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

Alexander Hankin*
Harvard University
Cambridge, MA, USA
alexander.hankin@intel.com

Abdulrahman Mahmoud
Harvard University/MBZUAI
Cambridge, MA, USA

Mark Hempstead
Tufts University
Medford, MA, USA

David Brooks
Harvard University
Cambridge, MA, USA

Gu-Yeon Wei
Harvard University
Cambridge, MA, USA

***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 and scalability trade-offs in a trapped-ion quantum computing design and find that optimal scheduling of qubits can provide a 6.2× speedup in performance. Additionally, our analysis shows that scaling TI quantum computers *horizontally* (i.e., minimizing the use of weak links and maximizing the chain length) results in a 20% average speedup compared to using more chains for qubit placement.

***Index Terms*—quantum computing, ion trap, trapped ion, weak link, ion chain, analytical model, performance model, timing, scaling**

I. Introduction

Quantum computer (QC) designs have advanced rapidly over the last decade. Google [5], IBM [13], and IonQ [22] 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 [2]), Microsoft (Azure Quantum [6]), and Amazon (Braket [3]), 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 [4], [10], [15], [51]. For example, Grover's algorithm is a quantum search algorithm that requires only O(sqrt(N)) computations as compared to classical search which requires O(N), where N is the size of the function's domain [28]. Accelerated search has the potential to speed up many critical applications such as drug development [68] or simulated annealing [17], as well as advance many other computational problems.

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 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 (with more description provided in §II and §VII). Designing a QPU around a specific technology will have major

*Now at Intel Labs.

implications across the entire accelerator ecosystem, as qubit mapping, gate scheduling, and decoherence mechanisms are closely associated with the underlying qubit implementation. Additionally, the mechanisms (both physical and abstraction layers) for scaling up a QPU to support thousands of qubits for meaningful computations is closely related to the qubit technology.

The two leading candidates for qubit implementation are 1) superconducting qubits, as employed by IBM [13], Google [5], and Intel [30], which take a semiconductor-based approach around the Josephson Junction to achieve quantum mechanical properties for computing; and 2) trapped-ion qubits, as employed by IonQ [22] and Honeywell [52], which are built around trapping and control of atomic ions for their quantum behaviors. In the modern NISQ era where noise is a major contributing factor towards designing QC, the trapped-ion (TI) qubits have an advantage over superconducting qubits in their longer coherence times [65], as well as other benefits over superconducting qubits. On the flip side, a major drawback for TI qubits is that *scaling* to a large number of qubits is a fundamental and physical challenge. Quantum Charge Coupled Device (QCCD) based TI devices address this drawback to a limited degree, but there is still significant research effort needed to architect them and scale them to 50-100 qubits and beyond [48]. 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.

A high-level overview showing the architecture of a simple TI QC is shown in Figure 1. 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) [14], [18], [36]. 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 [43], [49], [57]. We call such connections between chains a *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, exploring the trade-off between "horizontal" and "vertical" scaling has not been performed before (§VI-B). 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 (§III, §IV, §V);
- Evaluation of optimal TI configuration for a range of quantum applications on QCCD-based TI systems (§VI-A);
- Novel analysis on the effect of the weak link and ion chain length on the performance and scalability of TI QCs from 50–100 qubits and beyond (§VI-B);
- Open-sourcing the codebase for large-scale experimentation.

II. Background

While quantum computing differs substantially from classical computing paradigms, we focus on the architectural perspective for designing a different type of accelerator. We first describe this paradigm shift, highlighting important distinctions from classical computing for designing a quantum accelerator (§II-A), followed by a deep dive into the trapped-ion qubit technology we consider (§II-B).

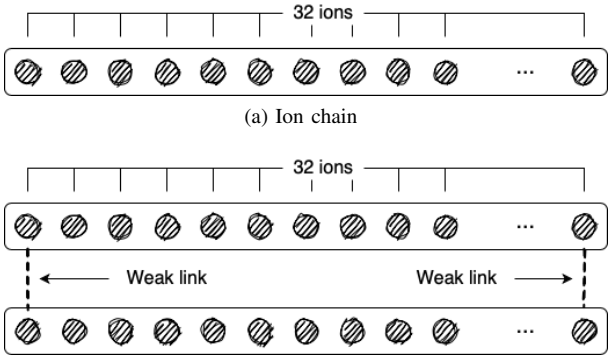
A. Paradigm Shift: Classical to Quantum Acceleration

Quantum computers (QCs) are still at a nascent stage in development, from the underlying physical implementation of qubits to the high-level programming and compilation of quantum programs. However, most computer architects agree that QC will not necessarily *replace* classical processors, but rather work alongside them to accelerate certain applications [19]. In fact, many quantum applications require either classical preprocessing or postprocessing, enforcing the need for co-development with traditional classical machines. To that end, it is more accurate to represent QCs as *quantum processing units* (QPUs), and design and evaluate them as a unique type of accelerator.

A QPU differs from classical accelerators in a couple of fundamental ways. First, quantum processors operate on *qubits* rather than classical digital bits. Qubits exhibit unique physical properties such as superposition and entanglement (which do not appear in classical bits), and also physically decohere over time. Decoherence is a particular challenge for the realization of fault tolerant (FT) quantum systems; today, most research operates in the *noisy, intermedate-scale quantum* era, or NISQ. Thus, quantum computing and reliability research go hand-in-hand in the design of quantum computing accelerators.

Second, a quantum computing architecture differs from its classical counterpart in that it does not obey a Von Neumann separation between compute and memory. Qubits represent both the compute fabric and the information flow in a QPU, and quantum measurements inherently break the computing system (commonly exemplified by Schrodinger's cats' thought experiment). Thus, many classical and convenient architectural abstractions and structures (e.g., caches, pipelining, etc) do not naturally extend to the quantum domain.

Third, while classical computers have built whole ecosystems around CMOS (and related) technologies, there are multiple



(a) Ion chain

(b) Multiple ion chains

Fig. 1: **High level overview of TI architecture**. Ions are organized in chains which can be linked together via *weak links*. The current physical limit for the number of ions in one chain is 32.

qubit technologies in development and analysis, and none have yet emerged as a clear "winner". The leading two candidates are superconducting qubits and trapped-ion (TI) qubits, while other potential qubit technologies exist such as the semiconductor spin qubits [32], linear optics [37], and Majorana qubits [33].

B. Qubit Technologies

There are multiple competing technologies in the qubit space. As was the challenge in the early days of computing for harnessing and taming the classical properties of electrons, the challenge today is to control the quantum properties and leverage them for computing.

Trapped-ion (TI) qubits are one of the leading candidates in this space. TI qubits operate directly on atoms, leveraging the quantum mechanical properties via internal energy levels of the atoms. Common ions such as Ca+, Ba+, or Yb+ (amongst others) are used [19]. This is in contrast to Superconducting qubits, which are implemented with lithographically printed circuit elements. Superconducting qubits revolve around an electrical circuit element called a Josephson junction to create "artificial" atoms which exhibit desired quantum mechanical properties. Historically the leading (and only practical) qubit technology for many years, superconducting qubits can scale to large numbers if noise can be sufficiently reduced (hence operating close to 0 Kelvin). Modern challenges in the superconducting work is exploring how to address errors resulting from physical noise and qubit state decoherence, with quantum error correcting codes [20], randomized circuit designs [63], and frequent recalibration [38] a requirement for producing useful operational qubits.

A TI qubit is implemented by stabilizing an ion in physical space by "trapping" it in place to control it. This is done by applying an RF Paul trap to freeze the ion at a saddle-point, followed by physical cooling to reduce vibrations and decoherence. Microwave pulses (hyperfine qubits) or lasers (optical qubits) are used to control qubits and excite ions and transition from one state to another for computation, depending on the frequency of operation for the trapped ions.

Trapped ions have relatively long coherence times [65], which means that the qubits are long-lived to address reliability concerns. However, scalability is a challenge. TI qubits can be put into chains, as depicted in Figure 1, allowing nearby qubits to interact with one another (via gate operations, entanglement, etc) and a physical system is architected at the chain-granularity (for cooling, frequency manipulations, etc). However as chains become long, it becomes challenging to properly control individual qubits, and the fine-grained frequencies of the qubits must be extremely well controlled. So far, multiple published systems can address up to 32 ions through the use of a 32-channel acousto-optic modulator (AOM) [14], [18], [36].

Instead of attempting to scale ion chains *horizontally* by making chains longer and adding control for more qubits within a chain, Quantum Charge Coupled Device (QCCD) based TI devices have been proposed which allows us scale *vertically* by linking multiple chains together (Figure 1) [11], [34], [35], [39], [44], [46], [47], [52], [55], [56], [64], [66]. To connect two chains, a *weak link* is introduced between a qubit in the first chain and a qubit in the second chain. The physical novelty of linking chains vertically could potentially enable trapped-ions to scale up to more qubits; however, multiple chains introduces new constraints which need to be considered from a *performance* perspective.

In particular, quantum gates can only operate on qubits that belong to the same chain *or* the weak links connecting two chains together. This introduces a physical limitation on the types of qubit operations a higher level algorithm or compiler can schedule, since not all qubits can directly interact with one another. Furthermore, qubit operations on weak links introduce a heavy latency penalty due to the physical link used to communicate between two chains (as compared to local gate operations between two qubits on the same chain).

These physical attributes of trapped-ion architectures are commonly dismissed if focusing on a gate-level abstraction of the underlying qubit technology [61], [62]. Additionally, while most studies at the architectural level focus on the challenges of reliability [50], [59], functional correctness [8], [58], or scalability [16], [60], only one work has modeled the performance impact of QC as a function of the underlying trapped-ion technology [48]. Understanding the performance implications is a crucial piece of the puzzle while navigating the early stages of QPU design in the NISQ era of 50-100 qubits and beyond.

III. 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. In this work, we focus on the trapped-ion (TI) qubit technology as a promising hardware fabric for QPUs, and introduce a performance modeling framework, VelociTI, to address the challenge of scalability for TI architectures.

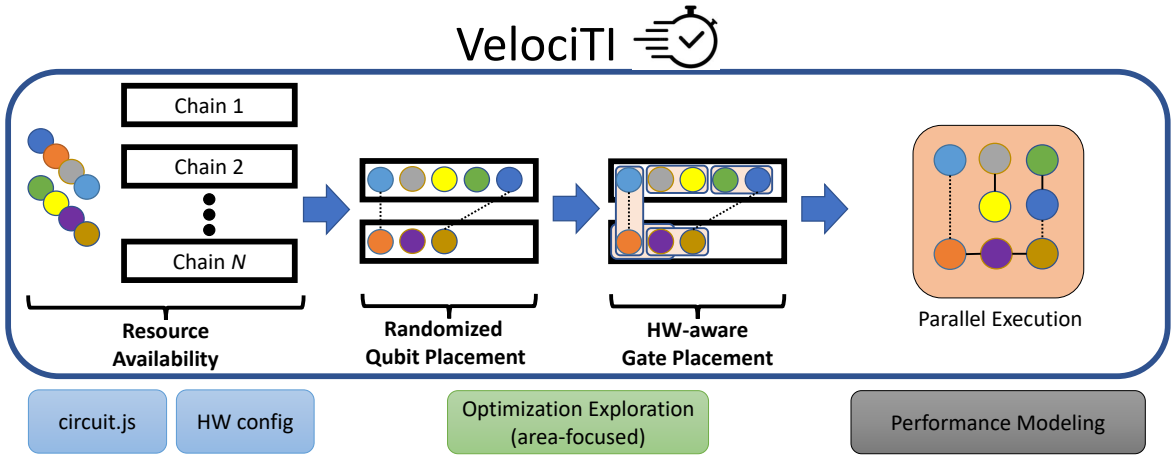


Fig. 2: **High-level overview of VelociTI**. VelociTI performs "place-and-route" for qubits and gates across ion-chains for a given optimization target (e.g., minimal area utilization) and measures the circuit timing using optimized performance models.

A. 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 [42], [54], 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 [48] and vertical scaling via weak links. To the best of our knowledge, no publicly available architecture-level modeling framework currently employs a TI chain weak link abstraction for conducting performance and scalability experimentation of TI systems beyond the NISQ-era scale (50-100 qubits). This elevates the need for simulation and modeling tools, like VelociTI, to understand the impact, benefits, and drawbacks of scaling QCCD-based TI QC systems which connect ion-chains via weak links.

B. VelociTI Design and Workflow

Figure 2 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. Table I lists configuration parameters for VelociTI.
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.

	parameter	meaning
configured	q	number of 1-qubit gates
	p	number of 2-qubit gates
	δ	latency for 1-qubit gate
	γ	latency for 2-qubit gate inside chain
	$\alpha\gamma$	latency for 2-qubit gate between chains
	opt	chain optimization target
computed	c	number of chains
	w_{max}	maximum number of weak links
	w	number of weak links used

TABLE I: **Model parameters for the TI circuit abstraction used in VelociTI**. The weak link between chains is accounted for by penalty factor, α.

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.

C. Limitations

We emphasize that VelociTI is a performance and timing tool, and is not a tool for functional simulation of quantum algorithms. Unlike classical circuit design where gates are placed in a deterministic manner on input signals to generate outputs for functional correctness, our tool explores the mapping of qubits and gates on the ion-chain architecture to determine the expected runtime for an algorithm. This distinction and limitation is important, as VelociTI can be used to understand the expected runtime on a TI architecture and how an algorithm can potentially scale (as a function of the number of gates it requires and the underlying parallelism of the TI design), but our tool is not meant to be used to obtain the quantum computational result of a running algorithm. Additionally, we abstract the particular *type* of gate (e.g., a CNOT or Hadamard gate) based on the number of inputs it requires (e.g., 1-qubit gates or 2-qubit gates), and model the latency of the gate operation. Again, this benefits timing and performance, setting aside algorithmic correctness. Accurately modeling functional correctness at large number of qubits is an intractable problem, as that would imply a classical simulator outperforming a quantum processor. These all factor into our design decision to focus on the previously unexplored architecture-level performance modeling of NISQ-era and future TI systems, provided the absence of real multi-chain TI systems for experimentation, but we envision incorporating functional simulation for small systems as a future direction.

IV. 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.

A. Model parameters

The model parameters are the parameters which VelociTI uses to represent a quantum TI system. A summary of the model parameters is displayed in Table I. This includes the total number of qubits and the total number of gates. The other configured model parameters include 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 [7], [53], [57]. α is the penalty term for a 2-qubit gate latency that involves a weak link. opt represents 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.

B. Serial baseline performance

For any trapped-ion quantum circuit specified using the model parameters listed above, we can then derive an expression for baseline serial timing, t_{serial}. This relationship is described in Equation 1. t is a function of both the total latency for 1-qubit and 2-qubit gates. In the case of a 2-qubit gate, there is a possibility of a weak link being present. This is captured by Γ. The expression for Γ is shown in Equation 2. Γ is equivalent to the total latency of all the 2-qubit gates in the circuit.

$$t_{serial} = q\delta + \Gamma \tag{1}$$

$$\Gamma \equiv w\alpha\gamma + (p - w)\gamma \tag{2}$$

Serial operation modeling does not take advantage of intra-chain parallelism, indicating a worst-case runtime performance. While physically not a realistic option in a TI architecture, we consider this as a normalization baseline for measuring different possible parallel speed-ups, as described in the following section.

C. 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.

Figure 3 illustrates our graph representation of a quantum circuit used for performance modeling. Figure 3(a) shows a 7-qubit circuit where qubits q1–q4 are on one ion-chain (black qubits), and q5–q7 are on a separate ion-chain (red qubits). This circuit contains six 2-qubit gates.

The corresponding parallel performance model graph is shown in Figure 3(b)-(e). In this example, there are three "start nodes": (1) the 2-qubit gate operating on q1 and q2, (2) the 2-qubit gate operating between q3 and q4, and (3) the 2-qubit gate operating on q6 and q7. This is illustrated with 2 circles on the node in the graph representation. Notice that start-nodes have no incoming edges, and only contain outgoing edges. As a basic optimization, we save all start-nodes as a list in VelociTI, allowing us to maintain a record of start nodes when calculating the longest path in the circuit.

Each node is labeled with the qubits being operated on for each gate. If there are multiple gates operating on the same qubits, we use a unique identifier for each instance of the gate, incrementing from 1. This is analogous to a static-single assignment (SSA) format when labeling registers, in order to uniquely identify each gate in the circuit.

Edge weights are subsequently added to the graph, representing gate latencies in the system. For example, between nodes q3q4 and q4q5, we have an edge weight latency of $(1+\alpha)\gamma$ because the destination node/gate, q4q5, is a 2-qubit gate and there is a weak link involved ($\alpha\gamma$) plus the latency of q3q4 (γ) because it is a start node. So the total latency is $\alpha\gamma+\gamma=(1+\alpha)\gamma$. On the other hand, between q4q5 and q5q6 we have an edge weight of γ because q5q6 is a 2-qubit gate (no weak link) and the source node, q4q5, is not a start node.

D. 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. An overview of this process in shown in Figure 3(b)-(e).

Each subfigure indicates a computational chain of operations which could be performed in parallel on the TI architecture. 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. In the example illustrated, the parallel latency of the circuit is dictated by path (e), for a latency of $(1+\alpha)\gamma+\gamma$.

V. VelociTI Software Implementation

A. Implementation Details

VelociTI is implemented using Python v3.8 and designed with three main high-level priorities: flexibility, usability, and extensibility. Due to the rapid growth of quantum computing research, these design critera are required for rapid iteration and additional abstractions to be added to the framework. We use the NetworkX [1] graph library for the directed graph representation, along with a rich user input interface to expose all model parameters as input arguments to allow for easy design space exploration and scalability experiments through batch scripts. We add functionality to configure, save, and load existing circuits to the software via json configuration files. Finally, a test bench equipped with canonical unit tests is used to maintain tool correctness as the framework scales.

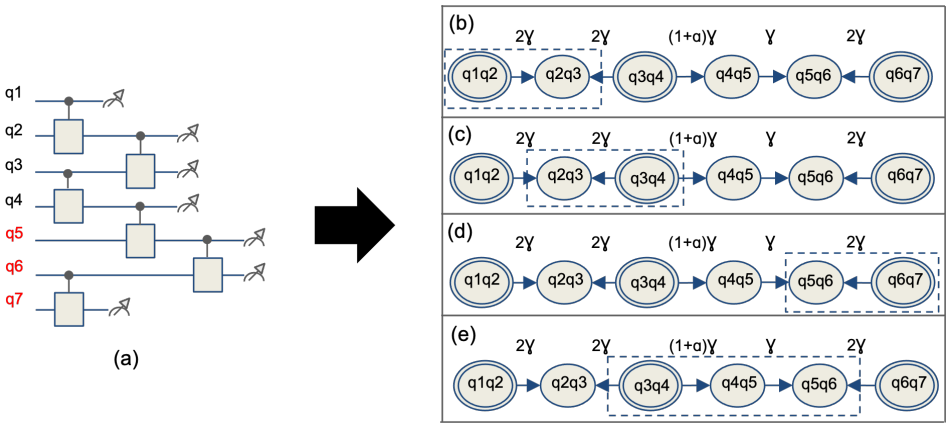


Fig. 3: **Directed graph representation of a TI architecture**. (a) Shows an example 2-chain (black, red), 7-qubit, 6 2-qubit gate quantum circuit. The corresponding directed graph representation is shown in (b)-(e). Each node has a label representing a gate operation, and nodes with two circles indicate "start nodes". Edge weight refers to the latency of an operation (the destination node) plus the latency of the source node/operation if the source node is a "start node" (See §IV-C).

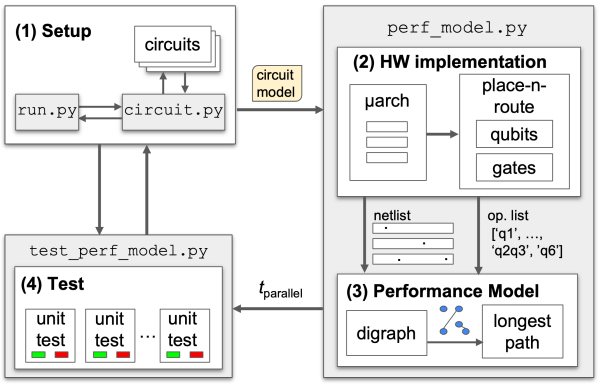


Fig. 4: **Overview of the software implementation of VelociTI**.

The software flow is divided into 4 main modules: (1) setup, (2) hardware implementation, (3) performance modeling, and (4) testing. An overview of the software flow is shown in Figure 4. First, the setup module either runs in normal mode or in test mode (with the test module). In normal mode, the setup module processes user inputs through command line arguments and sets the appropriate circuit model variables which will be fed into the performance modeling module. The setup module then instantiates a circuit using the circuit model shown in Table I. This is done through (a) direct specification of all model parameters, (b) by specifying a circuit from the library of circuits, or (c) choosing completely random circuits. The library of circuits contains the applications used in our evaluation (listed in Table II) as well as canonical test circuits that are used with the test bench. In test mode, the setup module also communicates with the test module model to initialize all unit tests in the testbench.

The hardware implementation module accepts the circuit model from the setup module and generates a netlist and operation order. To achieve this, intermediate microarchitecture blocks and place-and-route blocks determine from the circuit model: (i) a qubit mapping onto the available chains (i.e., netlist) where the number of chains is based on chain length and optimization target, and (ii) an operation order (i.e., op. list) that (a) is scheduled based on an optimization target and (b) respects the physical constraints of the TI hardware (i.e., weak links). Once the netlist and op. list are generated, the performance modeling module generates a directed graph based representation of the HW described by the netlist and the operation list, as described in §IV. The performance model finds all intra-chain parallel paths within the directed graph to ultimately compute the longest path in the entire circuit.

The test module is used when the software is run in test mode. It runs the testbench which contains unit tests of canonical circuit configurations which ensures sustained correctness of generic and corner cases as new abstractions and models are added to the software.

B. Tool Evaluation

To evaluate the performance and scalability of our software implementation, we measure the simulation time as quantum circuit size scales. This is shown in Fig. 5. We simulate random circuits and sweep number of qubits and number of 2-qubit gates up to the limit of NISQ-era TI systems (50-100 qubits). For each circuit, we do 35 simulation runs and each bar represents the average simulation time in seconds. We run the simulations on a desktop machine with a 6-core 3.1 GHz Intel Core i5 processor with 8 GB DDR4 memory running at 2667 MHz.

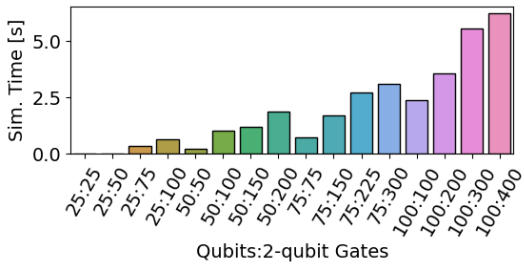


Fig. 5: **Software runtime versus quantum circuit size**.

Our results show an average simulation time of 0.63 seconds for a random circuit with 25 qubits and 100 2-qubit gates. For a random circuit with 100 qubits and 400 2-qubit gates, we measure a simulation time of 6.23 seconds, a 9.89× increase in simulation time. If we extrapolate this trend and assume that simulation time increases by 10× as number of qubits and number of 2-qubit gates scales by 4×, then we can predict that with 1,200 qubits and 4,800 2-qubit gates, the simulation time will be approximately 1 hour and 40 min. This is assuming that (1) no optimizations are performed to improve the runtime of the software implementation, and (2) the simulations are run on a commercial desktop machine; as such, this is a reasonable simulation time for classical performance simulators, and additional software engineering optimizations can further improve the toolchain.

VI. TI Performance and Scalability Evaluation

To demonstrates the use of VelociTI, we perform two case studies. First, 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. Second, we use the tool as a simulator to explore the design space of a QPU depending on how the various parameters may scale over time. For our case studies, we run VelociTI with 6 quantum computing applications: Supremacy [4], [41], Quantum Approximate Optimization Algorithm (QAOA) [23], [24], [29], [45], SquareRoot (Grover's search algorithm [28]), Quantum Fourier Transform (QFT) [9], Adder, and Bernstein-Vazirani (BV) [67]. We choose these applications because they provide a variety in (a) number of qubits, (b) number of 2-qubit gates, and (3) ratio of 2-qubit gates to qubits. An overview of these applications are shown in Table II. These applications have been used in previously published TI architecture work [48]. For our gate latency model, we use the latency values shown in Table III, which are consistent with prior work on published TI systems [7], [53]. We use a latency of 1 μs for a

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

TABLE II: Applications with attributes used in our evaluation.

Gate Latencies	
Latency for 1-qubit gate [μs]	1
Latency for 2-qubit gate [μs]	100
Penalty for weak link	2.0, 1.8, 1.6, 1.4, 1.2, 1.0

TABLE III: Latency of gates in our evaluation [7], [53], [57]. Weak link penalty is described in §IV.

1-qubit gate and a latency of 100 μs for a 2-qubit gate. In the case of a weak link, we apply a penalty factor of 2 [57].

A. Case Study 1: 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 6. 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.

B. Case Study 2: Design Exploration and Scalability

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,

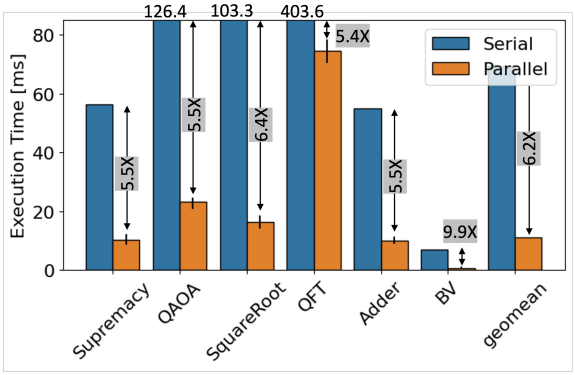


Fig. 6: **Estimated performance for the applications used in our evaluation running on a given practical TI hardware implementation**. Chain length of 16 ions is used, which is typical of NISQ-era QCCD-based TI systems [48].

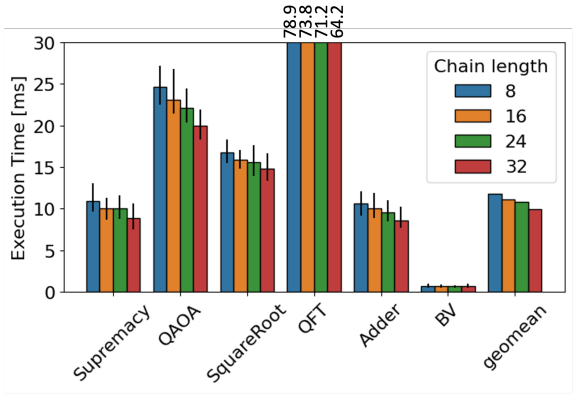


Fig. 7: **Estimated performance as a function of chain length.** for the applications used in our evaluation (see Table II and §VI-B). Chain length is swept within the achievable realm (8 ions to 32 ions).

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 7 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 BV with a speedup of only 11% between chain length of 8 qubits to 32 qubits. 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.

To further study the effect of chain size in parallel performance of TI architectures, we conduct a scalability experiment. In this case, we turn to a different measure of quantum computer evaluation: quantum volume. Quantum volume refers to square quantum circuit with N qubits and N/2 2-qubit gates. In the previous case study, we observed that the ratio of 2-qubit gates

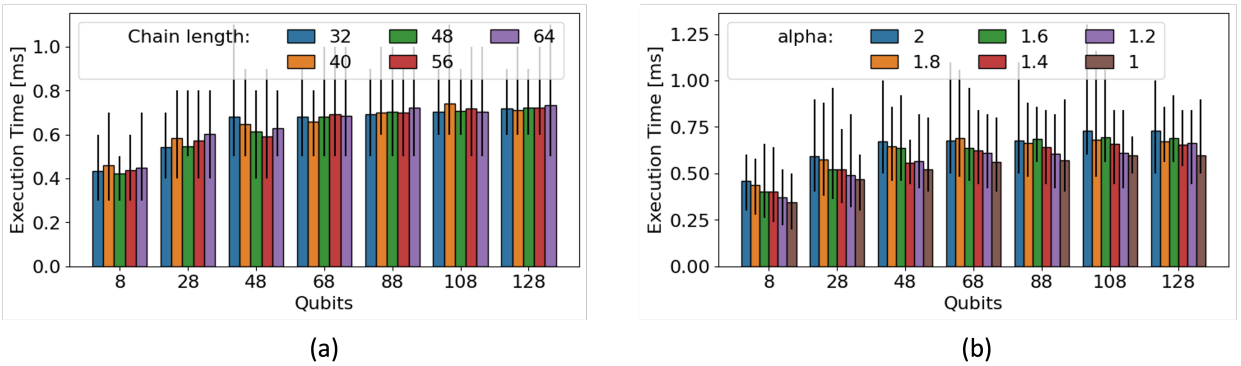


Fig. 8: **Effect of scaling chain length and weak link penalty on the performance of Quantum Volume (N qubits, N/2 2-qubit gates)**. (a) Shows the effect of doubling the 32-ion chain length by means of increments of 8 ions. Increasing the chain length, i.e. scaling horizontally, has trivial impact on execution time (consistent with Bernstein-Vazirani execution time in Fig. 7) (see §VI-A). (b) Shows the effect of reducing the weak link penalty from 2 down to 1, i.e. no weak link penalty. This results in a consistent reduction in parallel execution with an average speedup of 24% from α=2 to $\alpha = 1$.

to number of qubits had a significant affect on the speedup achieved by parallel execution over serial execution. We next explore if this ratio may have an effect on parallel execution time of a TI architecture with chain length and weak link penalty scaled *beyond the presently achievable limits*.

We scale ion chain length from 32 qubits to 64 qubits and reduce the weak link penalty α from 2 down to 1 (implying no weak link penalty). The objective for this study is to determine which type of scaling is the best method for maximizing TI execution time in the future. Figure 8(a) and 8(b) show execution time as a function of chain length and α, respectively, as quantum volume scales from 8 qubits to 128 qubits.

We make the following observations from our results. First, increasing chain length has a trivial impact on execution time as quantum volume scales. This is in fact consistent with the behavior of BV, another application with lower ratio of 2-qubit gates to number of qubits. Second, we observe that reducing the weak link penalty factor from 2 to 1 has a consistent effect on parallel execution as quantum volume scales, and it results in an average speedup of 24%. Third, execution time varies dramatically across application runs with the maximum difference between average execution time and maximum execution time for a single benchmark surpassing 50%. This makes sense given the low ratio of 2-qubit gates to number of qubits. In fact, quantum volume represents the lowest possible ratio and provides an interesting case study. For lower ratios of 2-qubit gates to qubits, the scheduler has less options and since we are using random scheduling, the scheduler is more likely to choose a less optimal schedule. This motivates the need for robust scheduling optimizations to prevent 50% or more performance degradation. This result also drives home the application dependence on architecting and scaling TI quantum computers.

Figure 9 shows a similar analysis but with different circuits. We keep the range of number of qubits the same, but this time we use a ratio of N qubits to 2·N 2-qubit gates to capture the effect of circuit composition on scaling potential of TI QCs. Figure 9(a) shows the same scaling of chain length, but this time we see a non-trivial impact of horizontal scaling, i.e., chain length scaling on performance. There is a marked difference in this effect for circuits with either less than or greater than 48 qubits. For less than 48 qubits, chain length scaling still has a trivial impact on performance; however, for 48, 68, 88, 108, and 128 qubits, we see an average speedup of up to 34% (48 qubits, 96 2-qubit gates). Figure 9(b) shows the effect of scaling the weak link penalty on performance. We observe an even greater impact of weak link penalty scaling on execution time than with Quantum Volume with an achieved speedup of up to 49% (128 qubits, 256 2-qubit gates).

These scalability studies indicate a substantial dependence of circuit composition in terms of the ratio of qubits to 2-qubit gates. This has a couple important implications for the future design of TI systems. First, TI design must be done with awareness of the characteristics of the application(s) that will run on the QPU. From Table II, we can see a variety of qubit to 2-qubit gate ratios. In the case of applications which have a higher ratio of 2-qubit gates to qubits, an increased chain length will boost performance while in the case of an application similar to quantum volume, increasing chain length may have a trivial impact on performance. On the other hand, we find that if device and physics researchers are able to improve the weak link connection (with current optics or other methods) between TI chains, this will result in a substantial speedup in performance.

VII. Related Work

As the surge in quantum computer research is relatively new, the quantum computing research community is still developing the foundational work on a robust quantum hardware-software stack. While there has been work at almost all levels of the computing stack for quantum computing, there has been more attention in certain areas. For example, at the software, compiler, and device/physics levels, there has been considerably more research than at the architecture-level. There has been scarce work in developing the crucial architecture-level abstracts and tools which bridge the gap between hardware and software.

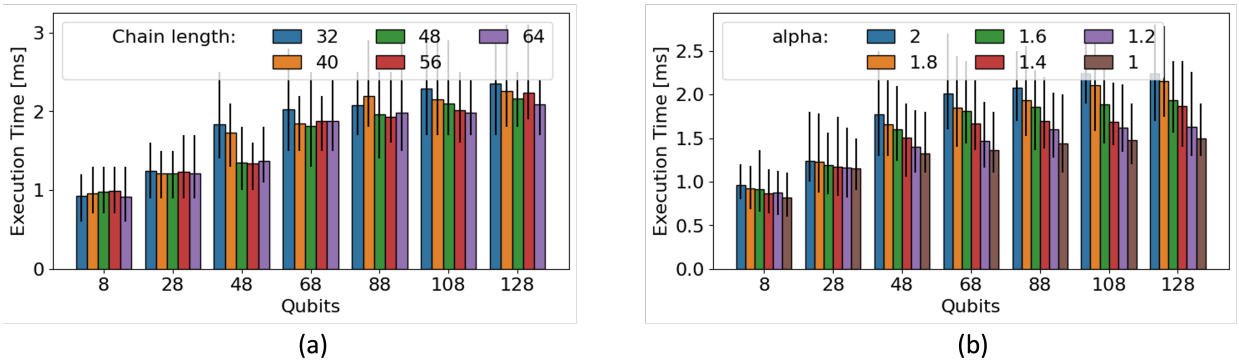


Fig. 9: **Effect of scaling chain length and weak link penalty on the performance of a TI system simulating random circuits with 2:1 ratio of 2-qubit gates to qubits** (§VI-B). Plot (a) shows an alternate result to that for quantum volume where chain length scaling from 32 ions to 64 ions now results in up to a 34% speedup. Plot (b) shows similar trend for reducing weak link penalty compared to quantum volume – with greater magnitude.

This becomes even more pronounced when venturing outside of superconducting-based QC research.

A. Quantum architecture

Most of the QC research at the architecture level has focused on Superconductor-based QCs [25]–[27], [31], [40]. Murali et al published the first comprehensive research study on architecting NISQ-era QCCD-based TI systems (50-100 qubits) [48]. In their work, they identify key observations, including the impact of design choices like trap capacity and connectivity on fidelity, and the impact of application on design. This is consistent with case study 2 in Section VI in this work which shows that application characteristics affect design choices when scaling TI systems. Finally, Murali et al show that the optimal entangling gate implementation and chain reordering method also varies by application.

B. Abstract architecture-level modeling

For decades, classical architects have been designing modeling tools to analyze and optimize designs of an emerging system. This complements the work at lower levels of the computing stack by filtering out designs which are not feasible at the architecture level. Modeling is a key enabler of future computing system development, and similar efforts should be replicated in the quantum computing space. For instance, Wattch [12] is an early power modeling framework for analysis and optimization of architecture designs, and NVSim is a timing, energy, and area modeling framework for emerging non-volatile memories published by Dong, Xu, Xie, and Jouppi [21]. Such studies and tools help pathfind in the early stages of a computer designs, as is extremely important in the quantum domain today.

VIII. Conclusion

Quantum computers are predicted to be "accelerators" in traditionally classical computing systems, and interest in Trapped Ion (TI) quantum computers, in particular, has grown recently due to their all-to-all intra-chain qubit connectivity and their longer coherence times than superconducting based quantum computers. In this work, we developed and concretized fundamental 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 two case studies which represent a couple of the use cases of a tool like VelociTI. First, 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. Second, we focus on design exploration and (i) find the best choice of chain length within the bounds of what is currently achievable for our application suite and (ii) determine the effect of scaling chain length and weak link penalty beyond their current limits on Quantum Volume. We observed that for (i), execution time improved by 20% on average from a chain length of 8 to 32, and for (ii), chain length scaling had little to no impact while reducing weak link penalty from 2.0 to 1.0 resulted in a performance improvement of 24% on average for Quantum Volume.

References

[1] "Networkx: Network analysis in python," https://networkx.org.
[2] 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.
[3] Amazon Web Services, "Amazon Braket," 2020. [Online]. Available: https://aws.amazon.com/braket/
[4] 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. [Online]. Available: https://doi.org/10.1038/s41586-019-1666-5

[5] 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.

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

[7] 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. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.117.060504

[8] S. Barz, J. F. Fitzsimons, E. Kashefi, and P. Walther, "Experimental verification of quantum computation," *Nature physics*, vol. 9, no. 11, pp. 727–731, 2013.

[9] 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. [Online]. Available: https://doi.org/10.1145/167088.167097

[10] 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. [Online]. Available: https://doi.org/10.1038/nature23474

[11] R. B. Blakestad, C. Ospelkaus, A. P. VanDevender, J. H. Wesenberg, M. J. Biercuk, D. Leibfried, and D. J. Wineland, "Near-ground-state transport of trapped-ion qubits through a multidimensional array," *Phys. Rev. A*, vol. 84, p. 032314, Sep 2011. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.84.032314

[12] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proceedings of the 27th Annual International Symposium on Computer Architecture*, ser. ISCA '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 83–94. [Online]. Available: https://doi-org.ezp-prod1.hul.harvard.edu/10.1145/339647.339657

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

[14] 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.

[15] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, Dec. 2019. [Online]. Available: https://doi.org/10.1038/s41567-019-0648-8

[16] D. Copsey, M. Oskin, F. Impens, T. Metodiev, A. Cross, F. T. Chong, I. L. Chuang, and J. Kubiatowicz, "Toward a scalable, silicon-based quantum computing architecture," *IEEE Journal of selected topics in quantum electronics*, vol. 9, no. 6, pp. 1552–1569, 2003.

[17] E. Crosson and A. W. Harrow, "Simulated quantum annealing can be exponentially faster than classical simulated annealing," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, oct 2016. [Online]. Available: https://doi.org/10.1109%2Ffocs.2016.81

[18] 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. [Online]. Available: https://doi.org/10.1038/nature18648

[19] Y. Ding and F. Chong, *Quantum Computer Systems: Research for Noisy Intermediate-Scale Quantum Computers*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2020. [Online]. Available: https://books.google.com/books?id=wgrtDwAAQBAJ

[20] D. P. DiVincenzo and P. W. Shor, "Fault-tolerant error correction with efficient quantum codes," *Physical review letters*, vol. 77, no. 15, p. 3260, 1996.

[21] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, vol. 31, no. 7, p. 994–1007, jul 2012. [Online]. Available: https://doi.org/10.1109/TCAD.2012.2185930

[22] 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. [Online]. Available: https://doi.org/10.1038%2Fs41586-021-03582-4

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

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

[25] X. Fu, L. Riesebos, M. A. Rol, J. van Straten, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, V. Newsum, K. K. L. Loh, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels, "eqasm: An executable quantum instruction set architecture," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 224–237.

[26] X. Fu, M. A. Rol, C. C. Bultink, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels, "An experimental microarchitecture for a superconducting quantum processor," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 813–825. [Online]. Available: https://doi.org/10.1145/3123939.3123952

[27] X. Fu, M. Rol, C. Bultink, H. van Someren, N. Khammassi, I. Ashraf, R. Vermeulen, J. Sterke, W. Vlothuizen, R. Schouten, C. Almudever, L. DiCarlo, and K. Bertels, "A microarchitecture for a superconducting quantum processor," *IEEE Micro*, vol. 38, pp. 40–47, 05 2018.

[28] 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. [Online]. Available: https://doi.org/10.1145/237814.237866

[29] M. P. Harrigan, K. J. Sung, M. Neeley, K. J. Satzinger, F. Arute, K. Arya, J. Atalaya, J. C. Bardin, R. Barends, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. Dunsworth, D. Eppens, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, S. Habegger, A. Ho, S. Hong, T. Huang, L. B. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P. V. Klimov, A. N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, M. Lindmark, M. Leib, O. Martin, J. M. Martinis, J. R. McClean, M. McEwen, A. Megrant, X. Mi, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, C. Neill, F. Neukart, M. Y. Niu, T. E. O'Brien, B. O'Gorman, E. Ostby, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N. C. Rubin, D. Sank, A. Skolik, V. Smelyanskiy, D. Strain, M. Streif, M. Szalay, A. Vainsencher, T. White, Z. J. Yao, P. Yeh, A. Zalcman, L. Zhou, H. Neven, D. Bacon, E. Lucero, E. Farhi, and R. Babbush, "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," *Nature Physics*, vol. 17, no. 3, pp. 332–336, feb 2021. [Online]. Available: https://doi.org/10.1038%2Fs41567-020-01105-y

[30] Intel, "100-Qubit Quantum Computing System Unveiled," 2021. [Online]. Available: https://www.tomshardware.com/news/atom-computing-unveils-100-qubit-quantum-computing-system

[31] A. Javadi-Abhari, P. Gokhale, A. Holmes, D. Franklin, K. R. Brown, M. Martonosi, and F. T. Chong, "Optimized surface code communication in superconducting quantum computers," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 692–705. [Online]. Available: https://doi.org/10.1145/3123939.3123949

[32] B. E. Kane, "A silicon-based nuclear spin quantum computer," *nature*, vol. 393, no. 6681, pp. 133–137, 1998.

[33] T. Karzig, C. Knapp, R. M. Lutchyn, P. Bonderson, M. B. Hastings, C. Nayak, J. Alicea, K. Flensberg, S. Plugge, Y. Oreg, C. M. Marcus, and M. H. Freedman, "Scalable designs for quasiparticle-poisoning-protected topological quantum computation with majorana zero modes," *Phys. Rev. B*, vol. 95, p. 235305, Jun 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.95.235305

[34] V. Kaushal, B. Lekitsch, A. Stahl, J. Hilder, D. Pijn, C. Schmiegelow, A. Bermudez, M. Müller, F. Schmidt-Kaler, and U. Poschinger, "Shuttling-based trapped-ion quantum information processing," *AVS*

Quantum Science, vol. 2, no. 1, p. 014101, 2020, _eprint: https://doi.org/10.1116/1.5126186. [Online]. Available: https://doi.org/10.1116/1.5126186

[35] D. Kielpinski, C. Monroe, and D. J. Wineland, "Architecture for a large-scale ion-trap quantum computer," *Nature*, vol. 417, no. 6890, pp. 709–711, Jun. 2002. [Online]. Available: https://doi.org/10.1038/nature00784

[36] 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. [Online]. Available: http://opg.optica.org/abstract.cfm?URI=QUANTUM-2020-QM6A.2

[37] E. Knill, R. Laflamme, and G. J. Milburn, "A scheme for efficient quantum computation with linear optics," *nature*, vol. 409, no. 6816, pp. 46–52, 2001.

[38] L. Lao, P. Murali, M. Martonosi, and D. Browne, "Designing calibration and expressivity-efficient instruction sets for quantum computing," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 846–859.

[39] P. H. Leung, K. A. Landsman, C. Figgatt, N. M. Linke, C. Monroe, and K. R. Brown, "Robust 2-qubit gates in a linear ion crystal using a frequency-modulated driving force," *Phys. Rev. Lett.*, vol. 120, p. 020501, Jan 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.120.020501

[40] G. Li, Y. Ding, and Y. Xie, "Sanq: A simulation framework for architecting noisy intermediate-scale quantum computing system," 2019. [Online]. Available: https://arxiv.org/abs/1904.11590

[41] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, "Quantum supremacy is both closer and farther than it appears," 2018. [Online]. Available: https://arxiv.org/abs/1807.10749

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

[43] 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.

[44] T. S. Metodi, D. D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang, "A quantum logic array microarchitecture: Scalable quantum data movement and computation," in *Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 38. USA: IEEE Computer Society, 2005, p. 305–318. [Online]. Available: https://doi.org/10.1109/MICRO.2005.9

[45] 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. [Online]. Available: https://doi.org/10.1088/2058-9565/aab822

[46] C. Monroe and J. Kim, "Scaling the ion trap quantum processor," *Science*, vol. 339, no. 6124, pp. 1164–1169, 2013. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1231298

[47] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, "Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects," *Phys. Rev. A*, vol. 89, p. 022317, Feb 2014. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.89.022317

[48] 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. [Online]. Available: https://doi-org.ezp-prod1.hul.harvard.edu/10.1109/ISCA45697.2020.00051

[49] 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.

[50] D. Oliveira, E. Giusto, E. Dri, N. Casciola, B. Baheri, Q. Guan, B. Montrucchio, and P. Rech, "Qufi: a quantum fault injector to measure the reliability of qubits and quantum circuits," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022, pp. 137–149.

[51] 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

[52] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, "Demonstration of the trapped-ion quantum CCD computer architecture," *Nature*, vol. 592, no. 7853, pp. 209–213, Apr. 2021. [Online]. Available: https://doi.org/10.1038/s41586-021-03318-4

[53] 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. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.2.020343

[54] 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.

[55] A. K. Ratcliffe, R. L. Taylor, J. J. Hope, and A. R. Carvalho, "Scaling trapped ion quantum computers using fast gates and microtraps," *Physical Review Letters*, vol. 120, no. 22, may 2018. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.120.220501

[56] S. Sargaran and N. Mohammadzadeh, "Saqip: A scalable architecture for quantum information processors," *ACM Trans. Archit. Code Optim.*, vol. 16, no. 2, apr 2019. [Online]. Available: https://doi.org/10.1145/3311879

[57] 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. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.124.110501

[58] Y. Takeuchi and T. Morimae, "Verification of many-qubit states," *Physical Review X*, vol. 8, no. 2, p. 021060, 2018.

[59] M. A. Thornton, "Introduction to quantum computation reliability," in *2020 IEEE International Test Conference (ITC)*, 2020, pp. 1–10.

[60] T. Tomesh, P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Viszlai, X.-C. Wu, N. Hardavellas, M. R. Martonosi, and F. T. Chong, "Supermarq: A scalable quantum benchmark suite," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 587–603.

[61] T. Tomesh, K. Gui, P. Gokhale, Y. Shi, F. T. Chong, M. Martonosi, and M. Suchara, "Optimized quantum program execution ordering to mitigate errors in simulations of quantum systems," in *2021 International Conference on Rebooting Computing (ICRC)*, 2021, pp. 1–13.

[62] T. Tomesh and M. Martonosi, "Quantum codesign," *IEEE Micro*, vol. 41, no. 5, pp. 33–40, 2021.

[63] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling," *Physical Review A*, vol. 94, no. 5, p. 052325, 2016.

[64] Y. Wan, R. Jördens, S. D. Erickson, J. J. Wu, R. Bowler, T. R. Tan, P.-Y. Hou, D. J. Wineland, A. C. Wilson, and D. Leibfried, "Ion Transport and Reordering in a 2D Trap Array," *Advanced Quantum Technologies*, vol. 3, no. 11, p. 2000028, 2020, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qute.202000028. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.202000028

[65] P. Wang, C.-Y. Luan, M. Qiao, M. Um, J. Zhang, Y. Wang, X. Yuan, M. Gu, J. Zhang, and K. Kim, "Single ion qubit with estimated coherence time exceeding one hour," *Nature Communications*, vol. 12, no. 1, jan 2021. [Online]. Available: https://doi.org/10.1038%2Fs41467-020-20330-w

[66] K. Wright, J. M. Amini, D. L. Faircloth, C. Volin, S. C. Doret, H. Hayden, C.-S. Pai, D. W. Landgren, D. Denison, T. Killian, R. E. Slusher, and A. W. Harter, "Reliable transport through a microfabricated$\less$i$\greater$X$\less$/i$\greater$-junction surface-electrode ion trap," *New Journal of Physics*, vol. 15, no. 3, p. 033004, Mar. 2013, publisher: IOP Publishing. [Online]. Available: https://doi.org/10.1088/1367-2630/15/3/033004

[67] 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.

[68] T. Zhou, D. Huang, and A. Caflisch, "Quantum mechanical methods for drug design," *Current topics in medicinal chemistry*, vol. 10, no. 1, pp. 33–45, 2010.