Received 12 October 2024; revised 16 February 2025; accepted 17 February 2025; date of publication 6 March 2025; date of current version 1 May 2025.

Digital Object Identifier 10.1109/TQE.2025.3548423

Quantum Circuit Compilation for Trapped-Ion Processors With the Drive-Through Architecture

CHE-MING CHANG[1,3], JIE-HONG ROLAND JIANG[1,2,3] (Member, IEEE), DAH-WEI CHIOU[2,3], TING HSU[3,4,5,6], AND GUIN-DAR LIN[3,4,5,6]

[1]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
[2]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
[3]Center for Quantum Science and Engineering, National Taiwan University, Taipei 10617, Taiwan
[4]Department of Physics, National Taiwan University, Taipei 10617, Taiwan
[5]Physics Division, National Center for Theoretical Sciences, Taipei 10617, Taiwan
[6]Trapped-Ion Quantum Computing Laboratory, Hon Hai Research Institute, Taipei 11492, Taiwan

Corresponding author: Jie-Hong Roland Jiang (e-mail: jhjiang@ntu.edu.tw).

This work was supported in part by the National Science and Technology Council of Taiwan under Grant 112-2112-M-002-001, Grant 112-2119-M-002-017, Grant 113-2112-M-002-025, and Grant 113-2119-M-002-024, in part by the Ministry of Education of Taiwan through the Featured Area Research Center Program within the framework of the Higher Education Sprout Project under Grant 113L900903, and in part by the NTU Core Consortium Project under Grant NTU-CC114L895002.

ABSTRACT Trapped-ion technologies stand out as leading contenders in the pursuit of quantum computing, due to their capacity for highly entangled qubits. Among many proposed trapped-ion architectures, the "drive-through" architecture has drawn increasing attention, notably for its remarkable ability to minimize heat generation, which is crucial for low-temperature operation and thermal noise reduction, thus reliable quantum computation. We present the first compilation system tailored for the drive-through architecture to achieve high-fidelity computation for intended quantum programs. Our approach accommodates the unique features of the new architecture that utilize transport gates to facilitate direct entanglement between static qubits and communication qubits. We optimize the qubit placement that changes over time for each trap, considering the cost of qubit swapping. Our method strategically balances the gate and swap distances, significantly improving the overall fidelity across various benchmarks.

INDEX TERMS Drive-through architecture, quantum circuit compilation, quantum computing (QC), qubit mapping, trapped ion.

NOMENCLATURE

$G_{\mathcal{G}} = (V_{\mathcal{G}}, E_{\mathcal{G}})$	Dependence graph of the circuit.
$V_{\mathcal{G}_\ell}$	ℓth layer of dependence graph.
$\ell_{\max}$	Circuit depth.
χ	Trap capacity.
G_{C_ℓ}	Connectivity graph of the ℓth layer.
$P_\ell = \{P_{\ell,1}, \ldots, P_{\ell,k}\}$	Partition of the ℓth layer.
T	# intertrap communications.
$\pi_{\ell,j}$ $(\pi_{\ell,j,0})$	(Initial) Configuration in layer ℓ trap j.
$\mathtt{gd}(\pi_{\ell,j})$	Gate distance of layer ℓ trap j.
$\mathtt{sd}(\pi_{\ell,j})$	SWAP distance of layer ℓ trap j.

I. INTRODUCTION

Quantum computing (QC) leverages the principles of quantum mechanics to solve problems intractable for classical computing. QC offers applications in diverse fields, from quantum chemistry [1], cryptography [2], to machine learning [3]. Despite theoretical evidence demonstrating the quantum advantages, the biggest challenge lies in building a scalable quantum computer with low error rates. Among various proposed quantum technologies, trapped-ion systems are one of the most promising [4], featuring high-fidelity quantum gates, state initialization, readout, and dense connectivity.

Several architectures have been proposed for scaling up trapped-ion quantum computers. The simplest architecture for trapped-ion QC is a single trap that stores all ions in a 1-D array. However, it faces limitations as the two-qubit gate speed decreases with an increasing number of ions [4]. The quantum charge-coupled device (QCCD) architecture [5] was then proposed to scale up near-term applications, which employs a modular approach consisting of several small traps. Ions are moved, or shuttled, to communicate between

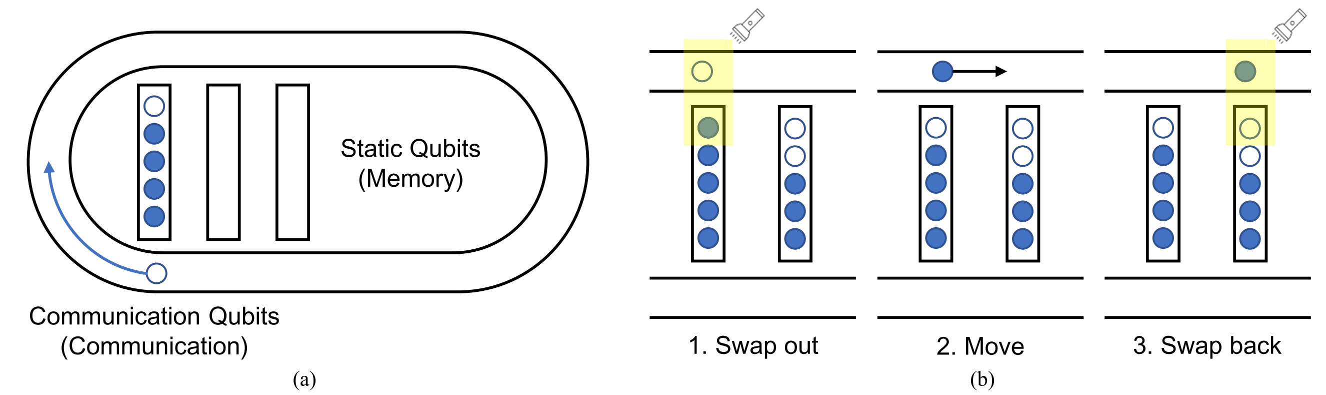


FIGURE 1. **(a) Drive-through architecture with static qubits in traps and communication qubits on the racetrack. Some qubits may be information-free (colored in white). (b) Methods for transporting the information of a static qubit to another trap.**

traps. Despite its advantages, QCCD still suffers from the complexity of precise control of the ion geometric configurations and the laser interactions [6].

To address the aforementioned problem, researchers in [7] proposed a novel "drive-through" architecture, which is the focus of our work. The advantages of this architecture are detailed in Section II-A4. As shown in Fig. 1(a), several traps are arranged in the middle, surrounded by a racetrack. Logical qubits in quantum algorithms are assigned to ions in these traps, referred to as static qubits, where standard quantum operations (single- and two-qubit gates) are performed. Simultaneously, several communication qubits move along a predesigned rail at a constant velocity on the racetrack, serving as the information commuters between traps, as illustrated in Fig. 1(b). A SWAP gate can be executed between a static and communication qubit; after the latter reaches its destination, another SWAP gate transfers the information back to the destination trap. Experiments have already demonstrated the feasibility of moving ions through such complex geometries [8].

Despite the progress in architectural exploration for trapped-ion processors, limited research has focused on designing qubit mapping algorithms specifically tailored to these architectures. Qubit mapping is a critical step in implementing quantum algorithms on hardware, assigning logical qubits from the program to physical qubits in the device while satisfying specific hardware constraints.

Most of the existing qubit mapping research has focused on superconducting devices, where qubit coupling is sparse and typically limited to nearest neighbors. Consequently, the primary goal in these systems is to minimize the number of SWAP gates inserted, as this helps reduce circuit depth, which is a critical optimization objective due to the short decoherence times of superconducting qubits. In contrast, trapped-ion systems offer a unique advantage with their significantly longer T_2 decoherence times, allowing for more relaxed constraints on circuit depth. Furthermore, since ions within a trap are fully coupled, most prior works focus on minimizing intertrap communication. This challenge resembles techniques used in distributed QC, where physical qubits remain in the same partition throughout the computation and rely on entanglement resources, such as Bell states, to enable interpartition communication. However, compilers designed for distributed QC [9], [10] are not directly applicable to trapped-ion systems, where ions must be shuttled between traps to perform entangling gates.

A. RELATED WORK

A bidirectional search method has been employed to improve initial qubit placement and reduce the number of inserted SWAP gates [11]. To optimize SWAP gate insertion and circuit depth, a look-ahead heuristic cost function combined with an A* search algorithm has been proposed in [12]. An architecture-agnostic methodology for qubit mapping has also been developed under the t|ket⟩ framework [13], [14]. More recently, a double-sourced shortest path method has been introduced to achieve optimal qubit mapping [15]. Beyond heuristic approaches, exact methods have also been explored to address the qubit mapping problem. These include satisfiability modulo theories (SMTs) [16], [17], [18] and incremental Boolean satisfiability problem (SAT)-solving techniques [19].

Most prior works [20], [21], [22], [23] on compilers for trapped-ion architectures focus on minimizing intertrap communication. Yet, these methods neglect the cost of reordering ion qubits in the traps and may lead to two-qubit gates operating on distant qubits, which can significantly degrade the fidelity. An SAT formulation of qubit mapping on QCCD devices with cyclic topologies has been proposed in [24], but this approach is not scalable to larger circuits. These challenges highlight the importance of an architecture-aware compiler that can optimize overall program fidelity, which is particularly crucial for quantum computers in the noisy intermediate-scale quantum (NISQ) era.

B. OUR CONTRIBUTIONS

The main contributions of this work are summarized as follows.

1) We develop the first compilation system to map a QASM description of a circuit to the low-level physical placement and movement on the drive-through architecture, which imposes constraints on intertrap communication through static qubits at the trap boundaries.
2) We present a scalable algorithm to optimize the dynamic placement for each trap. Our approach minimizes intertrap communication and optimizes intratrap configuration by considering the ion distances during SWAP gate execution [SWAP distance (SD)] and two-qubit gates [gate distance (GD)] simultaneously. To our knowledge, this is the first compiler work addressing both intertrap communication and intratrap ion configuration in modular architectures.
3) We conduct comprehensive experiments on various benchmarks to demonstrate that our approach consistently reduces total distance, thereby improving the overall program fidelity.

The rest of this article is organized as follows. Section II introduces the trapped-ion quantum computer and its qubit mapping problem. Section III formalizes the compilation problem for the drive-through architecture. Section IV details our compiling methods. Section V gives the experimental results. Finally, Section VI concludes this article and outlines future work.

II. BACKGROUND

A. TRAPPED-ION QUANTUM COMPUTER

1) ION CHAINS

In a trapped-ion quantum computer, quantum information is stored in the internal states of ions. These ions are first loaded in a fast-oscillating quadrupole potential and effectively captured, leading to crystallization that forms a 1-D ion chain via the Coulomb repulsion [25]. This configuration enables precise control and manipulation of individual ion qubits using lasers with specific frequencies. The entropy of the ions can be constantly removed by laser cooling [26], thus keeping the system at nearly zero temperature to improve the stability of the structure and robustness of quantum operations.

2) GATE IMPLEMENTATION BY LASER

In a trapped-ion system, a single-qubit gate can be achieved by illuminating the ions with a laser beam, driving coherence transition between two internal states [27]. A two-qubit gate, such as the Mølmer–Sørensen (MS) gate [28], utilizes laser beams to exert state-dependent forces on the participating ions, coupling the internal states of the two ions with the aid of the collective motion to build qubit entanglement. To consider the gate performance in simulations, we refer to [20] and [29] for an estimate of the gate time τ of amplitude modulation gates as

$$\tau(d) = 38 \times d + 10 \quad (\mu s) \tag{1}$$

where d is the number of ions that separate the two ions being entangled. In addition, the gate fidelity can be approximated as

$$F = 1 - \Gamma\tau - A(2\bar{n} + 1) \tag{2}$$

where Γ is the background heating rate of the trap, A is a scaling factor on the second term proportional to #qubits/ log(#qubits), and $\bar{n}$ is the motional mode or vibrational energy of an ion chain. Therefore, to achieve better fidelity during a two-qubit gate operation, it is crucial to minimize the distance between the two target qubits.

3) QCCD ARCHITECTURE

The QCCD architecture [5] employs a modular design of multiple small ion chains trapped in individual traps. The ions are then shuttled around different trapping zones to communicate between traps. Ion shuttling may involve three steps: 1) splitting the desired ion from an ion array in the original trapping zone; 2) moving the ion along a specific path to the destination zone; and 3) merging it into another ion array in the destination zone. These processes are time consuming [8], [30] and result in severe heat generation, introducing extra sources of error. It can be expected that shuttling control may eventually become a bottleneck that limits the computation speed and fidelity for trapped ion architectures.

4) DRIVE-THROUGH ARCHITECTURE

The drive-through architecture introduces a novel approach to trapped-ion systems by leveraging transport gates [6], [7], [31], [32], where quantum gates are executed when ions are transported through stationary optical beams. When a communication ion passes a static ion, a transport gate entangles two ions directly without slowing down the communication ion. This design eliminates the complex ion chain splitting and merging operations required in traditional architectures like QCCD. By avoiding such operations, this architecture reduces heating due to motional modes and vibrational energy, which is critical for achieving high-fidelity entangling gates. Moreover, it significantly shortens the recooling time, which is currently a major runtime bottleneck in trapped-ion systems [8], [30].

The fidelity of transport gates has been theoretically demonstrated to be comparable to conventional two-qubit gates, and the hardware implementation of this architecture has been proven and validated preliminarily [7]. However, compiling programs for the drive-through architecture presents new challenges. In [7], achieving high-fidelity transport gates requires proximity between communication and static ions, limiting the communication qubits' connectivity to static qubits near the trap's periphery.

In most trapped-ion systems, ions are shuttled to different traps for communication. We adopt a similar strategy in the drive-through architecture, exchanging the information of a static qubit with a communication qubit to allow the communication qubit to travel to different traps for further gate

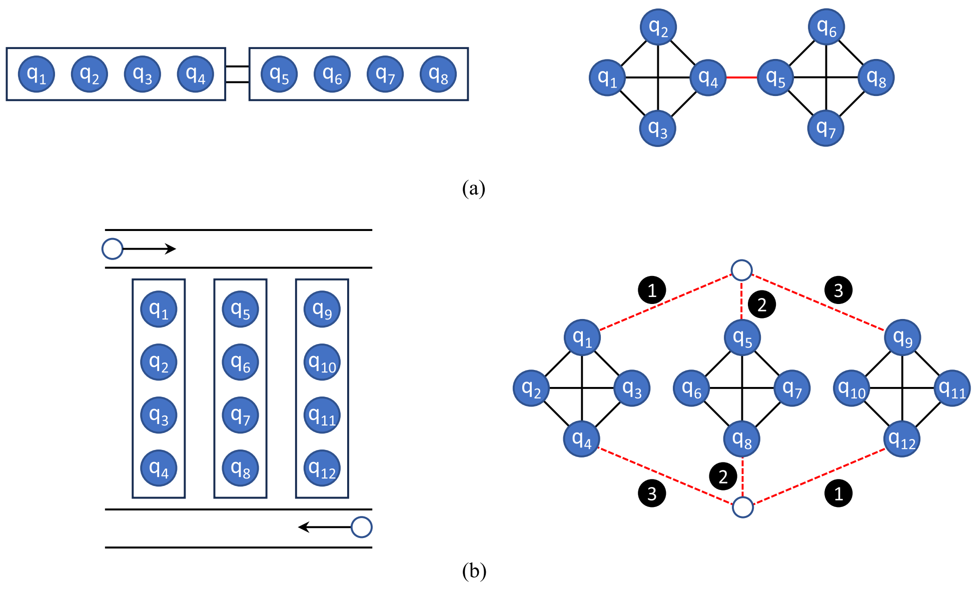


FIGURE 2. **Examples of transforming a trapped-ion architecture into a coupling graph. The red edges indicate the communication edges, where ions must be shuttled to different traps. (a) Linear QCCD architecture. (b) Drive-through architecture. Note that the communication edges in the drive-through architecture are available in specific orders.**

operations, as shown in Fig. 1(b). The operation of exchanging the information between a static qubit and a communication qubit is achieved by applying SWAP gates, using the same mechanism as in the case of a pair of static ions, rather than physically swapping their locations, to avoid slowing down the communication ion [7]. While this approach may increase the number of two-qubit gates, it simplifies the ion movement, reduces transport-induced heating, and improves overall routing efficiency.

Although the use of bridge gates (such as implementing a CNOT gate through a communication qubit) may reduce the number of logical two-qubit gates (requiring only four CNOT gates), it introduces significant inefficiencies in qubit routing. For example, implementing a CNOT(A, C) through a communication qubit B involves a sequence of CNOT gates: CNOT(B, C), CNOT(A, B), CNOT(B, C), and CNOT(A, B). This requires two complete cycles around the racetrack to complete the operation. Such repeated cycles generate excessive heat during ion transport, undermining the efficiency of this architecture. Only if the cost of ion movement was significantly reduced through technological advancements could employing bridge gates become a preferable option.

B. QUBIT MAPPING

As different quantum hardware only supports a predefined universal gate set known as the primitive gates, logic synthesis translates the quantum circuit into a lower level circuit description (e.g., QASM description) consisting of primitive gates from a particular gate set by performing gate decomposition and merging to optimize various objectives such as circuit depth or gate count [33]. The logical qubits within the synthesized circuit then need to be mapped onto the physical hardware so that the geometric configurations of the qubits are determined. This process is known as qubit mapping. However, the circuit potentially requires minor modifications (adding SWAP gates) to adhere to the coupling constraints imposed by the hardware.

The input of the qubit mapping problem typically consists of two components: the quantum circuit to be mapped, and the graph representing the coupling constraint of the hardware device. While most prior qubit mapping algorithms are primarily benchmarked on superconducting architectures, many can be adapted for trapped-ion processors if the coupling graph of the device is well defined. Fig. 2 illustrates two examples of how a trapped-ion architecture can be transformed into a coupling graph. In the linear QCCD system discussed in [21], the coupling graph can be modeled as several fully connected subgraphs linked by intertrap communication edges, as shown in Fig. 2(a). The coupling graph of the drive-through architecture can also be described using fully connected subgraphs, but with communication edges available in a specific sequence, emphasizing the dynamic nature of this architecture, as shown in Fig. 2(b). Due to its unique hardware constraints, most prior qubit mapping algorithms are not directly applicable to this new architecture.

III. PROBLEM FORMULATION

We formulate the quantum circuit compilation problem on the drive-through architectures as follows.

Problem 1 (The Drive-Through Architecture Compilation Problem): We are given an input quantum circuit C (in the QASM format) of primitive one- and two-qubit gates with

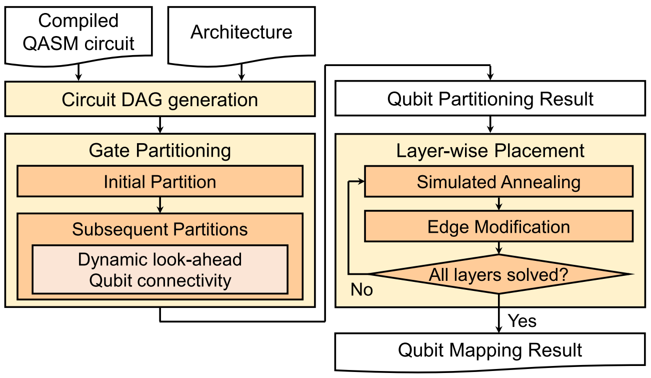


FIGURE 3. Compiler overview.

n logical qubits $Q = \{q_1, \ldots, q_n\}$ and given a drive-through architecture $\mathcal{A}$ in the form of Fig. 1(a) with k traps, each accommodating χ ions (i.e., static physical qubits) and one racetrack with sufficient communication ions (i.e., communication physical qubits), for $n \leq k\chi$. We are asked to provide a feasible execution of C on $\mathcal{A}$ such that the overall fidelity of the entire circuit execution is maximized.

We remark that our primary objective is to maximize the fidelity by the end of program execution using the gate errors defined by (2).

IV. METHODOLOGY

We summarize the notations used in this article in the Nomenclature. An overview of our algorithm is shown in Fig. 3. We first transform the QASM description into a directed acyclic graph (DAG) consisting of several layers. Then, we perform gate partitioning to assign gates to different traps. After partitioning, we solve the qubit placement trap by trap, optimizing the fidelity by simultaneously considering the program's SWAP gate and two-qubit gates. The qubit placement of each layer is slightly modified to serve as the initial placement for the next layer until all layers are solved.

A. CIRCUIT DAG GENERATION

We first transform the input QASM circuit description into a DAG $G_\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$, where each vertex $v_i \in V_\mathcal{G}$ denotes a two-qubit gate, and each edge $(v_i, v_j) \in E_\mathcal{G}$ denotes the precedence constraint that v_i should be executed before v_j. We remark that our definition aligns with prior work [11], [23]: A two-qubit gate on (q_i, q_j) can only be executed when all previous two-qubit gates on q_i or q_j have been executed. This definition does not consider the commutativity of two-qubit gates but directly follows the QASM description. The single-qubit gates are also not considered here because they could be executed locally on the target qubit without affecting the dependence [11]. The construction of the graph can be done in $O(|V_\mathcal{G}|)$ by traversing the QASM description of the circuit.

After the circuit DAG is constructed, we organize the DAG into several layers through as soon as possible scheduling. We define the ℓth layer of gate dependence graph $G_\mathcal{G}$ as a subset of gates $V_{\mathcal{G}_\ell} \subset V_\mathcal{G}$ with maximum path length ℓ from any vertex with in-degree 0. A layer consists of several independent gates that can be executed simultaneously. All layers can be initialized in $O(|V_\mathcal{G}|)$ by: 1) adding a source vertex s with in-degree 0 that connects to all input gates; 2) topological sorting of $G_\mathcal{G}$; and 3) finding the longest path from s to each vertex.

B. GATE PARTITIONING

After the dependence graph $G_\mathcal{G}$ is constructed, our next task is to assign quantum gates and qubits to different traps for execution. A partition of qubits is valid if each pair of qubits that requires gate execution is located in the same partition. The goal is to compute a sequence of valid partitions for each layer.

1) INITIAL PARTITION

We employ a greedy strategy to generate an initial partition based on qubit interactions from the first few layers of the circuit. For each gate involving a pair of qubits (q_i, q_j) in the first layer, we group q_i and q_j to the same cluster. This process is then repeated for the subsequent layers, adding additional qubits to clusters as long as the cluster size does not exceed the trap capacity χ. Once a cluster exceeds χ, no further qubits are added. Finally, small clusters are merged to create a balanced partition across all clusters.

2) SUBSEQUENT PARTITIONS

After the initial partition is generated, the partitions are dynamically updated for each layer. Intuitively, we aim to group interacting qubits together, while ensuring that the changes between consecutive partitions are minimized to reduce intertrap communication.

To achieve this, we first construct a connectivity graph G_{C_ℓ} for each layer ℓ. Each qubit represents a vertex in G_{C_ℓ}, and each gate (q_i, q_j) in the ℓth layer adds an edge between qubits q_i and q_j. Formally, the edge weight $w_{C_\ell}(q_m, q_n)$ accounts for the interaction between q_m and q_n over a small window of layers of depth d and is defined as

$$w_{C_\ell}(q_m, q_n) = \sum_{i=\ell}^{\min(\ell+d, \ell_{\max})} \mathbb{I}[(q_m, q_n) \in V_{\mathcal{G}_i}] f(i-\ell) \quad (3)$$

where $\mathbb{I}[\cdot]$ is the indicator function of whether q_m and q_n interact in layer i, $f(\cdot)$ is a decreasing positive function describing the decaying tendency, and $\ell_{\max}$ is the depth of the dependence graph. Once the connectivity graph is constructed, we perform min-cut partitioning on G_{C_ℓ} until a valid partition is achieved. A simple illustration of the partitioning on the connectivity graph is depicted in Fig. 4(c), with the valid partition $P_1 = \{P_{1,1} = \{q_2, q_4, q_8, q_9\}, P_{1,2} = \{q_0, q_3, q_5\}, P_{1,3} = \{q_1, q_6, q_7\}\}$.

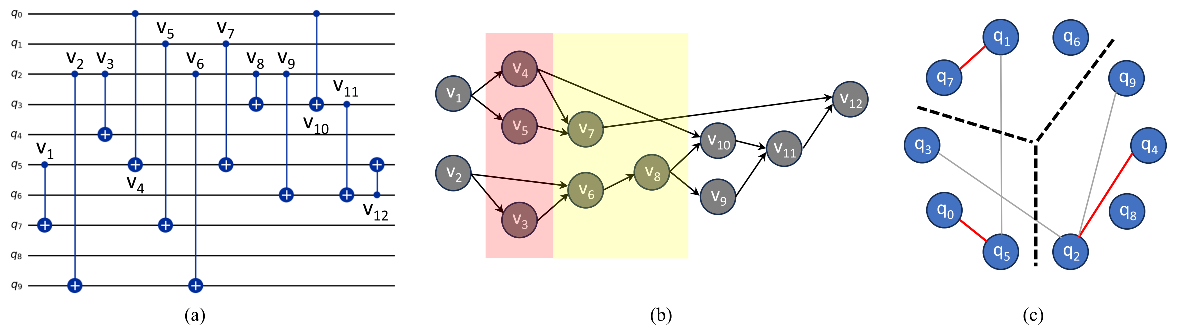


FIGURE 4. **Illustration of circuit DAG generation and the gate partitioning at $\ell = 1$. (a) Example program. (b) DAG generated following the order of the circuit in (a). (c) Valid $k = 3$ partition P_1 on the connectivity graph G_{C_ℓ} with a look-ahead depth $d = 2$ (weights are not shown). The corresponding edges are indicated in the highlighted part in (b).**

3) DYNAMIC LOOK-AHEAD DEPTH ADJUSTMENT

The look-ahead depth d determines how many additional layers are considered during partitioning, balancing the tradeoff between "global" information and processing time. Prior work [23] uses a constant $d = \ell_{\max}$ for all circuits. However, this approach overlooks the local structure of the circuit. We propose a method to dynamically adjust the look-ahead depth d according to the local circuit density. For sparse circuits with a low average number of two-qubit gates per layer, a larger d is preferable because it allows the partitioning algorithm to gather more information about upcoming qubit connections, leading to better partitioning decisions. On the other hand, the qubit connections are inherently more localized in dense circuits. In such cases, an excessive look-ahead d may increase program runtime without significant gains. By dynamically adjusting the look-ahead depth d, we tailor the partitioning process to the circuit's characteristics to improve both the solution quality and processing time.

4) MODIFIED k-WAY FIDUCCIA–MATTHEYSES ALGORITHM

Previous work [23] relies on the Kernighan–Lin (KL) algorithm [34] for partitioning, yet it has several limitations. While each iteration of the KL algorithm requires calculating the gain of swapping each vertex, the complexity could be $O(n^3)$, where n is the number of vertices. Second, the KL algorithm assumes a fixed number of logical qubits per trap, which restricts flexibility in partitioning. We adopt the Fiduccia–Mattheyses (FM) algorithm to address these issues. While utilizing implementation techniques in k-way FM algorithms [35] and the reduction from hypergraphs to graphs, we can achieve a linear-time complexity. Also, it allows different numbers of logical qubits to be stored in a trap in different time steps.

5) COST METRIC

A combinatorial metric is required to describe the intertrap communications. However, the cost metric described in [23] is not applicable when partition sizes vary between layers. We proposed another cost metric based on the maximum bipartite matching that can be applied to these cases.

Given the partition of two consecutive layers $P_\ell = \{P_{\ell,1}, \ldots, P_{\ell,k}\}$ and $P_{\ell+1} = \{P_{\ell+1,1}, \ldots, P_{\ell+1,k}\}$ of the qubit set S, we create an undirected bipartite graph with vertices $P_{\ell,i}, P_{\ell+1,i} \forall i = 1, \ldots, k$. An edge $(P_{\ell,i}, P_{\ell+1,j})$ with weight $|P_{\ell,i} \cap P_{\ell+1,j}| \forall i, j = 1, \ldots, k$, describes how many qubits are the same in trap i of layer ℓ and trap j of layer $\ell + 1$. The minimum number of qubits required to move between these two layers is then given by

$$T_\ell = |S| - \max_\sigma \sum_{j=1}^{k} |P_{\ell,j} \cap P_{\ell+1,\sigma(j)}| \tag{4}$$

where σ is a permutation over $\{1, \ldots, k\}$. The total number of intertrap communications is the sum of communications in each consecutive layer

$$T = \sum_{\ell=0}^{\ell_{\max}-1} T_\ell. \tag{5}$$

Empirical evaluation shows that $\sigma(x) = x$ holds most of the time (that is, trap j itself is the most similar to trap j in the next layer for all $j = 1, \ldots, k$). This suggests that the partitioning algorithm does not lead to excess movements between traps.

C. LAYERWISE PLACEMENT

After qubit partitioning, the next step is to determine the optimal placement of qubits within each trap at each layer to enhance gate execution efficiency. To achieve this, we employ simulated annealing (SA) to optimize qubit placement, considering both standard two-qubit gates and SWAP gates simultaneously. Our method minimizes the cost function

$$\mathrm{Cost}(\pi_{\ell,j}) = \underbrace{LA(\pi_{\ell,j})}_{\text{net force}} + \underbrace{B(\pi_{\ell,j})}_{\text{pull force}} + 3 \times \underbrace{\mathtt{sd}(\pi_{\ell,j}, \pi_{\ell,j,0})}_{\text{hold force}} \tag{6}$$

which consists of the following three terms.

1) *Net force*: It encourages logical qubits involved in two-qubit gates to be positioned as close as possible to minimize the gate distance and thus improve fidelity.

2) *Pull force*: It guides logical qubits that will leave the trap in subsequent layers toward the boundary in advance.
3) *Hold force*: It penalizes excessive SWAP operations to minimize the SWAP distance, thereby preserving fidelity.

In the cost function, the hold force is weighted by a factor of 3, since each SWAP gate is decomposed into three CNOT gates, with each CNOT gate further decomposed into one MS gate and several single-qubit operations.

In the following, we formally define each term and describe its calculation in the optimization process.

1) NET FORCE: MINIMIZING GATE DISTANCE

The goal of optimizing qubit placement is to minimize the distance between ion qubits that are involved in two-qubit gate operations, which directly impacts the fidelity of those gates. To achieve this, we define the gate distance as the sum of the distances between the qubits involved in two-qubit gates within the same layer. Formally, for trap j at the ℓth layer, the gate distance for permutation $\pi_{\ell,j}$ can be defined as

$$\mathtt{gd}(\pi_{\ell,j}) = \sum_{\substack{q_m,q_n \in P_{\ell,j} \\ (q_m,q_n)\in V_{\mathcal{G}_\ell}}} |\pi_{\ell,j}(q_m) - \pi_{\ell,j}(q_n)|. \tag{7}$$

In our setting, the objective is to minimize the gate distance within the current layer while also considering the effect on future layers. To model this, we draw from the minimum linear arrangement problem (MinLA), which is commonly used in very large scale integration design to minimize wirelength in 1-D placements [36]. The input of the MinLA problem is an undirected graph $G = (V, E)$ with nonnegative weight $c_{i,j}$ for each edge $(i, j) \in E$. The goal is to find a permutation $\pi : \{1, \ldots, |V|\} \to \{1, \ldots, |V|\}$ that minimizes the sum of weighted edge length

$$\pi^* := \arg\min_{\pi} LA(G, \pi) = \arg\min_{\pi} \sum_{(i,j)\in E} c_{i,j} |\pi(i) - \pi(j)|. \tag{8}$$

In our case, the graph for each layer ℓ and trap j can be derived from the connectivity graph G_{C_ℓ}, with the subgraph $G_{C_\ell}[P_{\ell,j}]$ representing the logical qubits in trap j. The edges in this subgraph correspond to qubit pairs involved in two-qubit gates. To account for future layers, we apply a decaying function f to adjust the edge weight. The net force can then be described by the MinLA cost to improve the fidelity of two-qubit gates in the program

$$LA(\pi_{\ell,j}) = \sum_{i=\ell}^{\min(\ell+d,\ell_{\max})} \sum_{\substack{q_m,q_n \in P_{\ell,j} \\ (q_m,q_n)\in V_{\mathcal{G}_i}}} |\pi_{\ell,j}(q_m) - \pi_{\ell,j}(q_n)| \times f(i-\ell). \tag{9}$$

TABLE 1. Example Partitioning Solution of the Program in Fig. 4(a)

$P_{\ell,k}$	$k=1$	$k=2$	$k=3$
$\ell=0$	$\{q_2,q_4,q_8,q_9\}$	$\{q_0,q_3\}$	$\{q_1,q_5,q_6,q_7\}$
$\ell=1$	$\{q_2,q_4,q_8,q_9\}$	$\{q_0,q_3,q_5\}$	$\{q_1,q_6,q_7\}$
$\ell=2$	$\{q_2,q_4,q_8,q_9\}$	$\{q_0,q_1,q_3,q_5\}$	$\{q_6,q_7\}$
$\ell=3$	$\{q_4,q_8,q_9\}$	$\{q_0,q_1,q_5\}$	$\{q_2,q_3,q_6,q_7\}$
$\ell=4$	$\{q_4,q_8,q_9\}$	$\{q_0,q_1,q_3,q_5\}$	$\{q_2,q_6,q_7\}$
$\ell=5$	$\{q_4,q_8,q_9\}$	$\{q_0,q_3,q_5,q_6\}$	$\{q_1,q_2,q_7\}$
$\ell=6$	$\{q_4,q_8,q_9\}$	$\{q_0,q_3,q_5,q_6\}$	$\{q_1,q_2,q_7\}$

2) PULL FORCE: GUIDING LEAVING QUBITS

To ensure that the transport gate can be executed within a small distance, we propose a preprocessing step that moves qubits, which will leave the trap in the next layer, to the boundary. Logical qubits $q \in P_{\ell,j} \setminus P_{\ell+1,j}$, which are leaving the trap, will be moved to the edge, and fixed during SA. After SA is finished, we remove those leaving qubits from $\pi_{\ell,j}$ and add new incoming qubits $q \in P_{\ell+1,j} \setminus P_{\ell,j}$ at the border to form the initial configuration $\pi_{\ell+1,j,0}$ of the next layer. This process can be continued for all layer's initial configurations.

Taking the program in Fig. 4(a) as an example again, a simple partition solution is given in Table 1. As seen from $\ell = 1$ to $\ell = 2$, q_1 leaves from trap 3 to trap 2, which is thus placed in the boundary of the trap in Fig. 5(b). Then, for the current layer $\ell = 2$, since qubits q_2 in trap 1 and q_3 in trap 2 are leaving to other traps, they are placed at the boundary, as shown in Fig. 5(c). After moving these ions to different traps, it forms the initial configuration of the next layer $\ell = 3$.

Since we have the information on the upcoming partitioning result, we can guide those leaving qubits to the border in the previous layers before they leave. We consider some leaving sets of trap j in layer ℓ $L_{\ell,j,i} = P_{\ell,j} \setminus P_{\ell+i,j}$, $i = 2, \ldots, d$, where $q \in L_{\ell,j,i}$ means that qubit q is in trap j in layer ℓ, and will leave after i layers. The boundary leaving cost for the leaving set $L_{\ell,j,i}$ is defined as

$$b(\ell, j, i) = \sum_{q\in L_{\ell,j,i}} \begin{cases} \pi_{\ell,j}(q), & q \text{ leaves from top} \\ |\pi_{\ell,j}(q) - \chi|, & q \text{ leaves from bottom} \end{cases} \tag{10}$$

where χ is the trap capacity. The departure direction (top or bottom part of the racetrack) depends on the relative position of the destination trap. To consider each upcoming leaving set, we add the following cost into SA:

$$B(\pi_{\ell,j}) = \sum_{i=2}^{\min(\ell+d,\ell_{\max})} b(\ell, j, i) \times f(i) \tag{11}$$

where f is a decreasing function. This cost essentially acts as a "pull," guiding these qubits toward the boundary before the actual layer of departure. We termed it the pull force.

3) HOLD FORCE: REDUCING SWAP OVERHEAD

As the configuration transforms during SA due to SWAP gates, the executing cost of these gates must also be considered. Thus, we incorporate the "distance" between the

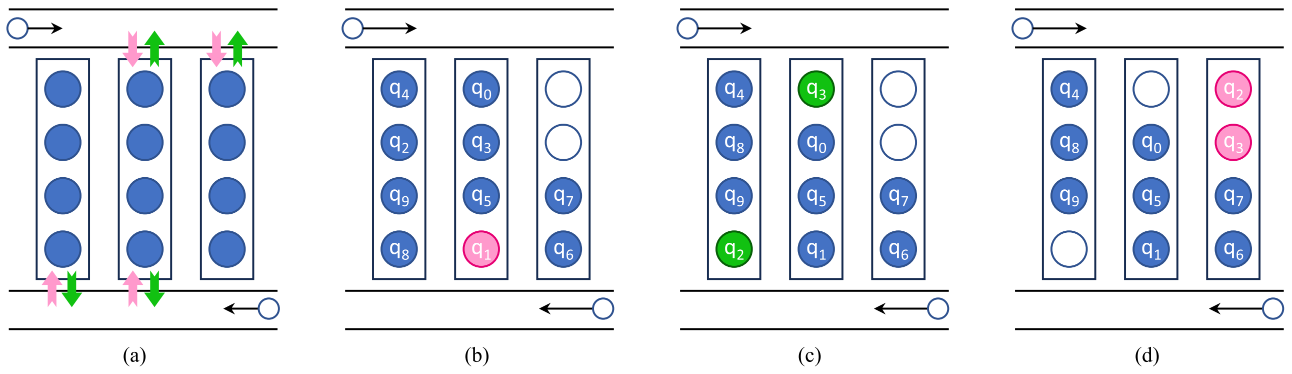


FIGURE 5. Illustration of layerwise placement from $\ell = 2$ to $\ell = 3$ using the partition solution in Table 1. (a) Entering (pink)/exiting (green) directions. (b) Configurations of $(\pi_{2,1,0}, \pi_{2,2,0}, \pi_{2,3,0})$. (c) Configurations of $(\pi_{2,1}, \pi_{2,2}, \pi_{2,3})$. Note that the edge modification relocates leaving qubits to the boundary. (d) Configurations of $(\pi_{3,1,0}, \pi_{3,2,0}, \pi_{3,3,0})$.

initial and final configurations in the SA cost metric. We termed it the hold force, which aims to stabilize the qubit configuration and reduce unnecessary transformations during optimization.

The final configuration $\pi_{\ell,j}$ is a permutation of the initial configuration $\pi_{\ell,j,0}$. To analyze the transformation, we decompose $\pi_{\ell,j}$ into its disjoint cycle decomposition [37] $C_{\pi_{\ell,j}} = \{c_1, \ldots, c_m\}$ by constructing a directed graph G with edges connecting each qubit's position in the initial configuration $\pi_{\ell,j,0}(i)$ to its position in the final configuration $\pi_{\ell,j}(i)$, where $i = 1, \ldots, \chi$ are the index of physical qubits in $P_{\ell,j}$. For each cycle $c_i \in C_{\pi_{\ell,j}}$, we compute the total displacement of physical qubits, which is the sum of the distances between two cyclically adjacent qubits in the cycle. The SWAP distance of $\pi_{\ell,j}$ is then defined as half of this total displacement, representing the movement needed to realize the permutation through SWAP gate operations

$$\begin{aligned}\mathtt{sd}(\pi_{\ell,j}, \pi_{\ell,j,0}) &= \frac{1}{2}\sum_{i} |\pi_{\ell,j,0}(i) - \pi_{\ell,j}(i)| \\ &= \frac{1}{2}\sum_{c\in C_{\pi_{\ell,j}}}\sum_{i\in c} |\pi_{\ell,j}(i) - \pi_{\ell,j}(i+1 \bmod |c|)|.\end{aligned} \tag{12}$$

Take the program in Fig. 5(b) and (c) as an example. The initial configuration is ordered qubit set (q_4, q_2, q_9, q_8), and the final configuration is (q_4, q_8, q_9, q_2). To analyze the transformation, we represent the initial configuration $\pi_{\ell,j,0}$ by indices $(1, 2, 3, 4)$. This corresponds to the permutation

$$\pi_{\ell,j} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix} \tag{13}$$

with a cycle decomposition $C_{\pi_{\ell,j}} = (1)(3)(42)$. Thus, the total displacement of physical qubits is $|1-1| + |3-3| + |2-4| + |4-2| = 4$ and the SWAP distance is 2. This result is intuitive as the only necessary swaps are between q_8 and q_2, which have a distance of 2. Note that this metric includes the cost of swapping qubits to the boundary, as described in Section IV-C2.

Since each node has only one outgoing edge, the SWAP operation defined here for two vertices is as follows. If $(v_i, v_{i+1}), (v_j, v_{j+1}) \in E$, then swapping v_i and v_j means to replace the above edges into (v_i, v_{j+1}) and (v_j, v_{i+1}), while the other edges remains the same. By only calculating the displacement difference on these incident edges, we can compute the cost difference more efficiently.

In summary, we optimize the dynamic qubit placement by three forces: net force, pull force, and hold force. The net force minimizes the gate distance to enhance two-qubit gate fidelity, the pull force guides qubits that will leave the trap to the boundary, and the hold force reduces unnecessary SWAP operations. By incorporating these forces into SA, we ensure that qubits are positioned in a way that improves overall execution fidelity while minimizing transport and SWAP overhead.

V. EVALUATION

A. EXPERIMENTAL SETUP

1) BENCHMARKS

We implemented our proposed algorithm in Python 3.7. Runtime tests were conducted on Ubuntu 20.04.6 LTS with Intel Xeon Silver 4208 at 2400 MHz with 256-GB memory. We benchmarked the performance of our algorithms on a wide range of application circuits including the Hardware Efficient Ansatz of the quantum approximate optimization algorithm (QAOA), Adder, and Supremacy circuit from [38] multiplier, quantum K-nearest neighbor (KNN), and quantum neural network (QNN) from [39], and the quantum volume (QV) and quantum Fourier transform (QFT), which are compiled by Qiskit [40]. Random circuits (Random M/L, where M stands for medium and L for large) were also generated to test patterns lacking in structured circuits. These testing circuits have around 50–250 qubits and contain thousands of two-qubit gates. Circuits of this size are large enough so that exact methods [16], [17], [18] could not find the optimum

TABLE 2. **Results of Intertrap Transport Requirements in Different Compiling Methods Under Trap Capacity 30**

Circuit	#Qubit	#Gate	Depth	#Trap	#Inter-trap transport SABRE-ext	#Inter-trap transport t|ket⟩-ext	#Inter-trap transport Ours
QNN	51	392	140	2	44	18	**6**
KNN	67	264	168	3	70	46	**21**
Adder 64	64	455	180	3	118	78	**26**
QFT 63	63	3400	245	3	558	286	**100**
Multiplier	75	6510	3555	3	1254	1578	**406**
QV	40	6000	300	2	1110	856	**614**
Supremacy	80	8500	421	3	1352	1912	**1037**
QAOA	80	7900	277	3	906	630	**397**
Random M	100	3000	189	4	2988	2790	**1735**
Random L	100	10000	319	4	10118	9492	**5979**

solution and are also the next major milestone for trapped-ion systems [20].

2) COMPILING METHODS

In our experiments, we compared our proposed method with two widely used qubit mapping tools: Qiskit [40] with its default mapping algorithms SABRE [11] and t|ket⟩ [14]. For SABRE, we use the standard placement and routing methods provided in Qiskit. We also set the optimization level to 0 since we do not want the tool to perform additional logic optimization to compare only the placement and routing results. For t|ket⟩, we use the noise-aware placement [using the two-qubit gate errors defined by (2)] method combined with the lexicographical comparison approach [13] for routing, both provided in t|ket⟩.

Although neither SABRE nor t|ket⟩ is specifically designed for the constraints of the drive-through architecture, we can still make a fair comparison with reasonable adjustments on the compiling results by utilizing the coupling graph shown in Fig. 2(b). Instead of adhering strictly to the communication order constraints imposed by the drive-through architecture, we create a complete graph for these boundary ion qubits. We denote the compiling methods here as SABRE-ext and t|ket⟩-ext as they are the extended versions of the original ones.

B. INTERTRAP COMMUNICATIONS

1) EVALUATION

In the drive-through architecture, it is not possible to implement a CNOT gate directly on these communication edges, since the static qubits must be physically shuttled to the other trap for entangling gate execution. Therefore, when a CNOT or SWAP gate appears on a communication edge in the compiled circuit, we count it as two intertrap communications to reflect the overhead accurately. In addition, suppose that consecutive CNOT gates are executed on the same communication edge without any other entangling gates operating on the incident qubits in between. In that case, they are still counted as only two intertrap communications.

Table 2 lists the statistics of the benchmarks and the results of the number of intertrap communications by different compiling algorithms. "#Gate" represents the two-qubit gate count in the circuit, and "Depth" indicates the depth of the circuit DAG. For our proposed algorithm, we ran the same experiment five times and reported the best-case result due to the randomness of SA. For trap capacity, to date, the largest trapped-ion systems have up to 32 qubits (IonQ [41] and Quantinuum [8]) per trap. Thus, we set the maximum trap capacity as a fixed number $\chi = 30$ for these experiments. As can be seen in the table, the gate partitioning approach considers the global structure of the circuit and thus minimizes the intertrap communication and outperforms the extended version of the qubit mapper SABRE-ext and t|ket⟩-ext in most large cases since these methods are not tailored for this kind of modular architecture.

2) EFFECT ON WEIGHT DECAYING FUNCTION AND LOOK-AHEAD DEPTH

We assign a baseline weight b to the gates in the current layer and apply an exponentially decaying weight $f(x) = b \times a^{-x}$ to the subsequent layers, where $x < d$. This allows us to assess the impact of different decaying tendencies (by tuning the parameter a) and various look-ahead depth d on the partitioning quality across three distinct types of circuits.

As illustrated in Fig. 6, the optimal choice for d and the decaying tendency is influenced by the circuit structure. For instance, the QNN typically exhibits a relatively sparse circuit structure, and thus, utilizing a larger d (around 60 to 70) can lead to better solution quality. In contrast, the number of intertrap communications for dense circuits such as QV or supremacy circuits eventually increases for larger d.

In addition, the runtime of partitioning increases as d increases due to larger processing data, as shown in Fig. 7. Interestingly, the runtime at $d = 0$ is higher than expected. This is because, at $d = 0$, the algorithm must frequently modify partitions without any upcoming information, resulting in inefficient runtime and poor quality. Therefore, considering too deep look-ahead information about the dense circuit can be nonbeneficial for minimizing the number of intertrap communications, and a suitable choice of d around 3–10 balances the optimization quality and processing time in such cases. In conclusion, our analysis indicates that a larger d is preferable for sparse circuits, while a smaller d is more appropriate for dense circuits.

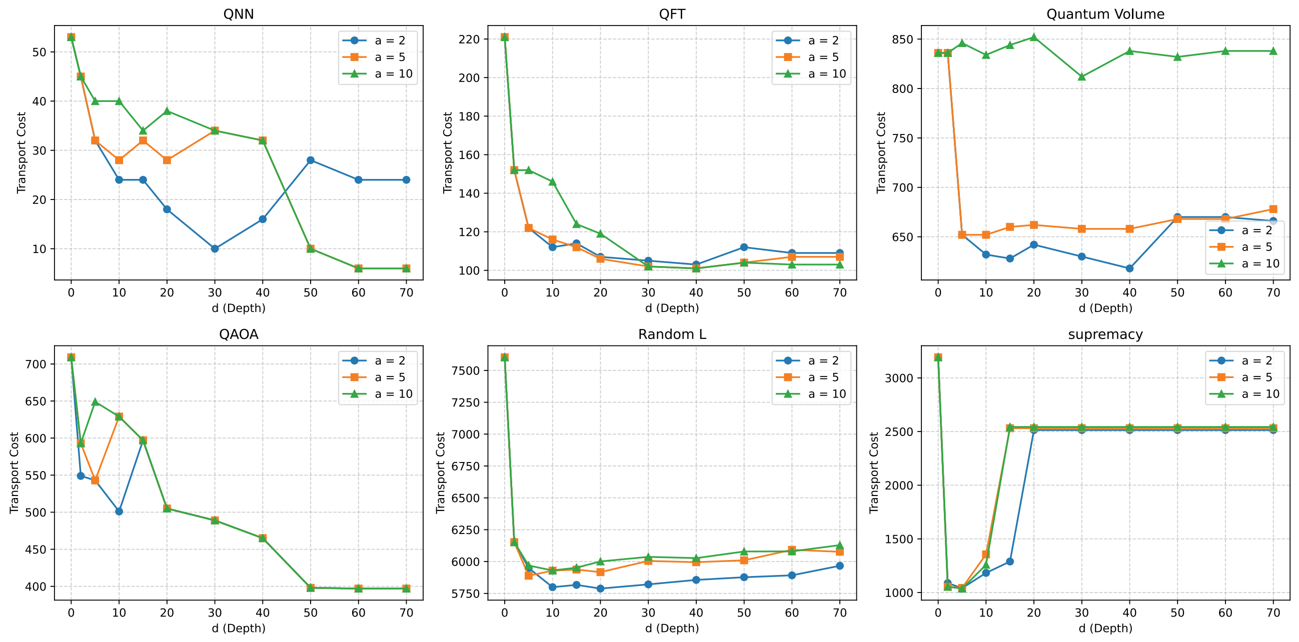


FIGURE 6. Number of intertrap transportation across various circuits with different look-ahead depth and decaying schemes.

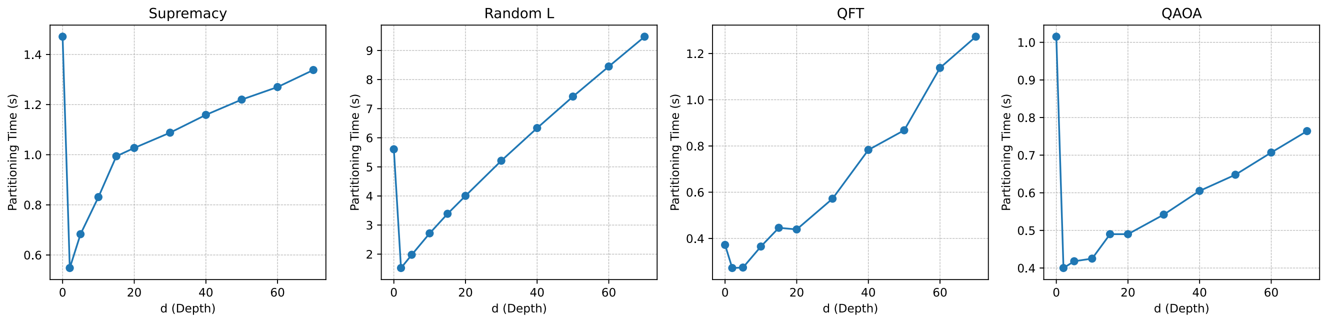


FIGURE 7. Gate partitioning time across various circuits with different look-ahead depths.

C. INTRATRAP QUBIT MAPPING

1) EVALUATION

We define the following distance metric to relate to the overall program fidelity.

1) The total gate distance is defined as the sum of gate distance in each trap and each layer, which is defined in Section IV-C

$$\mathrm{GD} = \sum_{\ell=1}^{\ell_{\max}} \sum_{j=1}^{k} \mathtt{gd}(\pi_{\ell,j}). \tag{14}$$

2) The total SWAP distance is defined as the sum of SWAP distance in each trap and each layer, which is defined in Section IV-C3

$$\mathrm{SD} = \sum_{\ell=1}^{\ell_{\max}} \sum_{j=1}^{k} \mathtt{sd}(\pi_{\ell,j}, \pi_{\ell,j,0}). \tag{15}$$

3) The total distance is defined as $\mathrm{TD} = \mathrm{GD} + 3\mathrm{SD}$. The SWAP distance is penalized by a factor of 3 since a SWAP gate can be decomposed into three CNOT gates, and all CNOT gates can be decomposed into one MS gate and some single-qubit gates. As the following shows, this cost metric accurately relates a combinatorial metric to the overall program fidelity.

Table 3 presents the GD, SD, and TD results after different compilation methods. To provide a rough idea of impacts on fidelity, we use a constant heating rate $\Gamma = 0.01$ (quanta per second) for a rough fidelity approximation for normal two-qubit gates as in (2). For the parameter $A(2\bar{n}+1)$, we use the numerical approximation as in [20] and [42]. For the fidelity of transport gates, we penalize the gate time by a factor of 2 as current transport gate technologies are limited by the velocity of the communication ions [32]. These gate fidelities are then multiplied to estimate the program fidelity.

TABLE 3. **Results of Running the Proposed Algorithm on Benchmark Circuits With Trap Capacity Set to 30**

Circuit	SABRE-ext		t|ket⟩-ext		Ours	
	$TD(GD, SD)$	Fidelity	$TD(GD, SD)$	Fidelity	$TD(GD, SD)$	Fidelity
QNN	5537 (4100, 479)	0.9648	3400 (3066, 167)	0.9678	3233 (2795, 146)	**0.9681**
KNN	4142 (2678, 488)	0.9630	4050 (3004, 523)	0.9673	2450 (1787, 221)	**0.9677**
Adder 64	7022 (4442, 860)	0.9411	6662 (4942, 860)	0.9459	4712 (3068, 548)	**0.9488**
QFT 63	66347 (43334, 7671)	0.6633	51136 (42400, 4368)	0.6813	49541 (43031, 2170)	**0.6871**
Multiplier	101682 (63864, 12606)	0.3755	97295 (66909, 15193)	0.3619	74999 (55922, 6359)	**0.4057**
QV	129669 (69285, 20128)	0.6377	106472 (68586, 18943)	0.6480	84353 (50720, 11211)	**0.6508**
Supremacy	130018 (83038, 15660)	0.2629	128331 (71809, 28261)	0.2421	71928 (56079, 5283)	**0.2781**
QAOA	77124 (73620, 1168)	0.3090	47039 (35627, 5706)	0.3331	17317 (15940, 459)	**0.3515**
Random M	175641 (33933, 47236)	0.3491	137226 (36996, 50115)	0.3607	105203 (32123, 24360)	**0.4168**
Random L	599695 (112540, 162385)	0.0300	463461 (123489, 169986)	0.0322	367787 (107621, 86722)	**0.0504**
Avg. Ratio	1.4379 (2.2872, 1.7464)	0.9019	1.3345 (2.4926, 1.4190)	0.9125	1.0000 (1.0000, 1.0000)	1.0000

TABLE 4. **Comparison of Different Intratrap Qubit Mapping Methods Using the Partitioning Result in Table 2**

Circuit	[20]		Ours	
	$TD(GD, SD)$	Fidelity	$TD(GD, SD)$	Fidelity
QNN	10823 (617, 3402)	0.9377	3233 (2795, 146)	**0.9681**
KNN	11748 (1335, 3471)	0.9352	2450 (1787, 221)	**0.9677**
Adder 64	11096 (1028, 3356)	0.8991	4712 (3068, 548)	**0.9488**
QFT 63	55646 (37373, 6091)	0.6642	49541 (43031, 2170)	**0.6871**
Multiplier	182024 (13757, 56089)	0.1500	74999 (55922, 6359)	**0.4057**
QV	99416 (50648, 16256)	0.6445	84353 (50720, 11211)	**0.6508**
Supremacy	384333 (50235, 111366)	0.0908	71928 (56079, 5283)	**0.2781**
QAOA	21820 (14500, 2440)	0.3432	17317 (15940, 459)	**0.3515**
Random M	196162 (20506, 58552)	0.3038	105203 (32123, 24360)	**0.4168**
Random L	689591 (68675, 206972)	0.0169	367787 (107621, 86722)	**0.0504**
Avg. Ratio	2.2162 (0.5745, 5.9070)	0.6899	1.0000 (1.0000, 1.0000)	1.0000

It is important to note that the heating rate $\Gamma = 0.01$ used here is significantly more optimistic than realistic values. For instance, the equivalent heating rate studied in [41] is approximately $\Gamma \sim 1$. Achieving $\Gamma \sim 0.01$ would require substantial advancements to realize systems with 50–100 qubits. However, if we had used $\Gamma \sim 1$, the overall fidelity would have been too low to allow meaningful comparisons for large-scale circuits. Therefore, we have chosen the highly idealized value of $\Gamma = 0.01$ to ensure a fair comparison. Table 3 is provided for comparison purposes only, and its numerical values should not be interpreted literally. In addition, single-qubit gates are neglected here as they can be implemented locally under the same operating conditions (within the trap), and they have identical effects on fidelity, assuming a constant heating rate Γ. Including single-qubit gates in the fidelity estimation would significantly lower the overall fidelity, rendering the comparison meaningless.

As shown in Table 3, our method achieves a notable reduction in total distance across various benchmarks, directly improving program fidelity due to the inverse relationship between ion distance and gate fidelity. In addition, reducing intertrap communication often minimizes SWAP distance by decreasing the number of edge modifications required to move qubits to the boundaries. These results underscore the importance of architecture-aware compilers in optimizing program fidelity, particularly for quantum computers operating in the NISQ era.

2) EFFECT ON INTRATRAP MAPPING METHOD

To evaluate the effect of the intratrap mapping result, we further compared our method with the heuristic ion reordering strategies implemented in [20] and [21]. These approaches assign weights to qubits based on the number of gates they participate in and then reorder qubits in descending order of weight. For a fair comparison, we applied our partitioning result in Table 2 to both approaches.

While the heuristic method significantly reduces the GD by optimizing the mapping of ion qubits within traps, it also leads to a considerable increase in the SD, as shown in Table 4. This indicates a tradeoff between minimizing gate distances and the additional overhead introduced by frequent ion swaps. In contrast, our method strikes a better balance by simultaneously considering both GD and SD, resulting in a substantial improvement in overall fidelity.

3) EFFECT ON TRAP CAPACITY CHOICES

We also conducted several experiments on the same circuit with different trap capacities as in Table 5 to examine the compiler's performance for different trap capacities to reflect the tradeoff between intertrap communications and intratrap configurations. As trap capacity increases, the number of required traps decreases, shortening the compilation time and reducing the number of intertrap communications—a

TABLE 5. Results of Running the Proposed Algorithm on `Adder 102` With Different Choices of Trap Capacity χ and Heating Rate Γ

χ	# Trap	GD	SD	# Inter-trap Transports	Fidelity ($\Gamma = 0.01$)	Fidelity ($\Gamma = 0.1$)	Fidelity ($\Gamma = 1$)	Compilation time (s)
60	2	14613	1213	40	**0.8521**	0.7996	0.6025	3.15
40	3	10677	1215	57	0.8490	0.8073	0.6453	3.97
30	4	8150	1213	66	0.8473	0.8126	0.6747	4.88
24	5	6933	867	69	0.8466	0.8181	0.7029	5.47
20	6	5526	797	73	0.8462	**0.8223**	**0.7239**	6.36
15	8	4027	1683	274	0.8192	0.7914	0.6787	8.08
10	12	2623	1770	410	0.7889	0.7639	0.6620	12.29

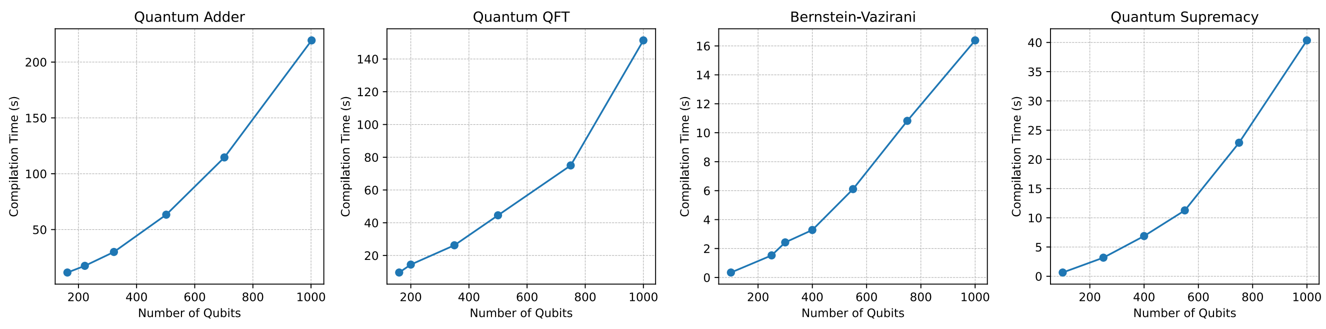


FIGURE 8. Compilation time for a series of circuits of different sizes.

major bottleneck in current trapped-ion technologies. However, larger traps introduce more complex intratrap configuration transformations, leading to higher GD and SD. On the other hand, using smaller traps minimizes intratrap complexity. Still, the increased number of traps significantly raises the number of intertrap communications. This higher communication overhead can impact gate scheduling, introducing delays that degrade performance and fidelity.

4) EFFECT ON PHYSICAL PARAMETERS

Simulated fidelity also depends on the heating rate Γ. For instance, as shown in Table 5, a lower heating rate improves the fidelity of all configurations. As the heating rate increases, the associated intratrap costs for GD and SD become more dominant than the intertrap communications. Thus, we conclude that the optimal trap capacity lies in the middle, where the tradeoff between intratrap complexity and intertrap communication is well balanced. It also depends on how effectively the technology, such as the heating rate or the transport gates in our drive-through architecture, can mitigate the penalties associated with intertrap communications.

5) SCALABILITY

To evaluate the scalability of our algorithm, we run a family of circuits with different sizes and record their average running time, as shown in Fig. 8. We also adjusted the trap size proportionally with the number of qubits to reflect advancements in trapped-ion technology. The results demonstrate that our compilation method exhibits an $O(n^2)$ runtime growth, which aligns with the $O(n^2)$ growth of two-qubit gates for circuits with $O(n)$ depth examined here. Our analytical method could trade a reasonable amount of runtime for solution quality, which is crucial for circuit sizes in the NISQ era. In contrast, exact methods such as SMT are often limited to small circuit sizes due to the computational complexity, and the runtime may grow exponentially as the circuit size increases. This limitation makes exact methods impractical for large modular architectures, highlighting the advantage of our scalable approach.

VI. CONCLUSION AND FUTURE WORK

Researchers have recently shown increasing interest in new QC hardware, and several compilers tailored for various architectures have been proposed. The drive-through architecture provides a promising approach toward large-scale trapped-ion quantum computers. This article proposed a scalable compilation flow for the drive-through architecture, considering the effect of both intertrap communications and the gates in the intended program. Our method substantially reduces the total distance of circuit execution, which can be crucial for total fidelity improvement, especially for quantum computers in the NISQ era.

Future work will focus on exploring and optimizing different architectural designs, including configurations with additional traps or racetracks, which may require more fine-grained mappings of partitions to specific traps for improved performance. In addition, addressing intertrap bottlenecks will require optimizing communication protocols. This could include strategies like scheduling traps with a higher number of two-qubit gates for execution priority or designing specific routing algorithms for communication qubits, ultimately minimizing intertrap transportation latency.

Our algorithm could also be extended to other similar modular architectures that communicate through near-boundary qubits, such as the modular universal scalable ion trap quantum computer [43], where different processor nodes could communicate through communication qubits at the border via an optical interface or other trapped-ion architectures with cyclic topologies [24]. Our compilation flow can potentially guide the trapped-ion quantum compiler design in the near future.

ACKNOWLEDGMENT

The authors thank IBM Quantum Hub at National Taiwan University and the Quantum Technology Cloud Computing Center at National Cheng Kung University for supporting access to quantum devices.

REFERENCES

[1] A. Kandala et al., "Hardware-efficient variational quantum Eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, Sep. 2017, doi: 10.1038/nature23879.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997, doi: 10.1137/s0097539795293172.

[3] 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, doi: 10.1038/nature23474.

[4] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021314, doi: 10.1063/1.5088164.

[5] 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, doi: 10.1038/nature00784.

[6] D. Leibfried, E. Knill, C. Ospelkaus, and D. J. Wineland, "Transport quantum logic gates for trapped ions," *Phys. Rev. A*, vol. 76, no. 3, Sep. 2007, Art. no. 032324, doi: 10.1103/physreva.76.032324.

[7] W. H. Png, T. Hsu, T.-W. Liu, M.-S. Chang, and G.-D. Lin, "Non-stop quantum entangling gate between a stationary ion qubit and a mobile one," *Bull. Amer. Phys. Soc.*, vol. 68, no. 7, Jun. 2023. [Online]. Available: https://meetings.aps.org/Meeting/DAMOP23/Session/U09.8

[8] S. A. Moses et al., "A race-track trapped-ion quantum processor," *Phys. Rev. X.*, vol. 13, no. 4, Dec. 2023, Art. no. 041052, doi: 10.1103/physrevx.13.041052.

[9] D. Ferrari, A. S. Cacciapuoti, M. Amoretti, and M. Caleffi, "Compiler design for distributed quantum computing," *IEEE Trans. Quantum Eng.*, vol. 2, 2021, Art. no. 4100720, doi: 10.1109/tqe.2021.3053921.

[10] I. Ghodsollahee, Z. Davarzani, M. Zomorodi, P. Pławiak, M. Houshmand, and M. Houshmand, "Connectivity matrix model of quantum circuits and its application to distributed quantum circuit optimization," *Quantum Inf. Process*, vol. 20, no. 7, Jul. 2021, Art. no. 235, doi: 10.1007/s11128-021-03170-5.

[11] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-Era quantum devices," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, Apr. 2019, pp. 1001–1014, doi: 10.1145/3297858.3304023.

[12] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1226–1236, Jul. 2019, doi: 10.1109/tcad.2018.2846658.

[13] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, "On the qubit routing problem," in *Proc. 14th Conf. Theory Quantum Comput., Commun. Cryptogr.*, 2024, pp. 5:1–5:32, doi: 10.4230/LIPIcs.TQC.2019.5.

[14] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t$|ket\rangle$: A retargetable compiler for NISQ devices," *Quantum Sci. Technol.*, vol. 6, Apr. 2020, Art. no. 014003, doi: 10.1088/2058-9565/ab8e92.

[15] C.-Y. Cheng, C.-Y. Yang, Y.-H. Kuo, R.-C. Wang, H.-C. Cheng, and C.-Y. Huang, "Robust Qubit mapping algorithm via double-source optimal routing on large quantum circuits," *ACM Trans. Quantum Comput.*, vol. 5, pp. 1–26, Aug. 2024, doi: 10.1145/3680291.

[16] B. Tan and J. Cong, "Optimal layout synthesis for quantum computing," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Dec. 2020, pp. 1–9, doi: 10.1145/3400302.3415620.

[17] T.-A. Wu, Y.-J. Jiang, and S.-Y. Fang, "A robust quantum layout synthesis algorithm with a Qubit mapping checker," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Oct. 2022, pp. 1–9, doi: 10.1145/3508352.3549394.

[18] W.-H. Lin, J. Kimko, B. Tan, N. Bjørner, and J. Cong, "Scalable optimal layout synthesis for NISQ quantum processors," in *Proc. 60th ACM/IEEE Des. Autom. Conf.*, Jul. 2023, pp. 1–6, doi: 10.1109/dac56929.2023.10247760.

[19] I. Shaik and J. Van de Pol, "Optimal layout synthesis for deep quantum circuits on NISQ processors with 100+ Qubits," in *Proc. 27th Int. Conf. Theory Appl. Satisfiability Test.*, Aug. 2024, vol. 305, pp. 26:1–26:18, doi: 10.4230/LIPIcs.SAT.2024.26.

[20] P. Murali, D. M. Debroy, K. R. Brown, and M. Martonosi, "Architecting noisy intermediate-scale trapped ion quantum computers," in *Proc. IEEE/ACM Int. Symp. Comput. Archit.*, Sep. 2020, pp. 529–542, doi: 10.1109/ISCA45697.2020.00051.

[21] A. A. Saki, R. O. Topaloglu, and S. Ghosh, "Muzzle the shuttle: Efficient compilation for multi-trap trapped-ion quantum computers," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, Mar. 2022, pp. 322–327, doi: 10.23919/date54114.2022.9774619.

[22] E. Nikahd, N. Mohammadzadeh, M. Sedighi, and M. S. Zamani, "Automated window-based partitioning of quantum circuits," *Phys. Scr.*, vol. 96, no. 3, Jan. 2021, Art. no. 035102, doi: 10.1088/1402-4896/abd57c.

[23] J. M. Baker, C. Duckering, A. Hoover, and F. T. Chong, "Time-sliced quantum circuit partitioning for modular architectures," in *Proc. 17th Int. Conf. Comput. Front.*, May 2020, pp. 98–107, doi: 10.1145/3387902.3392617.

[24] D. Schoenberger, S. Hillmich, M. Brandl, and R. Wille, "Using Boolean satisfiability for exact shuttling in trapped-ion quantum computers," in *Proc. 29th Asia South Pacific Des. Autom. Conf.*, Jan. 2024, pp. 127–133, doi: 10.1109/ASP-DAC58780.2024.10473902.

[25] W. Paul, "Electromagnetic traps for charged and neutral particles," *Rev. Mod. Phys.*, vol. 62, no. 3, pp. 531–540, Jul. 1990, doi: 10.1103/revmodphys.62.531.

[26] J. Eschner, G. Morigi, F. Schmidt-Kaler, and R. Blatt, "Laser cooling of trapped ions," *J. Opt. Soc. Amer. B*, vol. 20, no. 5, May 2003, Art. no. 1003, doi: 10.1364/josab.20.001003.

[27] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland, "Quantum dynamics of single trapped ions," *Rev. Mod. Phys.*, vol. 75, no. 1, pp. 281–324, Mar. 2003, doi: 10.1103/revmodphys.75.281.

[28] A. Sørensen and K. Mølmer, "Quantum computation with ions in thermal motion," *Phys. Rev. Lett.*, vol. 82, no. 9, pp. 1971–1974, Mar. 1999, doi: 10.1103/physrevlett.82.1971.

[29] C. J. Trout et al., "Simulating the performance of a distance-3 surface code in a linear ion trap," *New J. Phys.*, vol. 20, no. 4, Mar. 2018, Art. no. 043038, doi: 10.1088/1367-2630/aab341.

[30] J. M. Pino et al., "Demonstration of the trapped-ion quantum CCD computer architecture," *Nature*, vol. 592, no. 7853, pp. 209–213, Apr. 2021, doi: 10.1038/s41586-021-03318-4.

[31] L. E. de Clercq et al., "Parallel transport quantum logic gates with trapped ions," *Phys. Rev. Lett.*, vol. 116, no. 8, Feb. 2016, Art. no. 080502, doi: 10.1103/physrevlett.116.080502.

[32] H. N. Tinkey, C. R. Clark, B. C. Sawyer, and K. R. Brown, "Transport-enabled entangling gate for trapped ions," *Phys. Rev. Lett.*, vol. 128, no. 5, Jan. 2022, Art. no. 050502, doi: 10.1103/physrevlett.128.050502.

[33] G. De Micheli, J.-H. R. Jiang, R. Rand, K. Smith, and M. Soeken, "Advances in quantum computation and quantum technologies: A design automation perspective," *IEEE Trans. Emerg. Sel. Top. Circuits Syst.*, vol. 12, no. 3, pp. 584–601, Sep. 2022, doi: 10.1109/jetcas.2022.3205174.

[34] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 291–307, Feb. 1970, doi: 10.1002/j.1538-7305.1970.tb01770.x.

[35] Y. Akhremtsev, T. Heuer, P. Sanders, and S. Schlag, "Engineering a direct k-way hypergraph partitioning algorithm," in *Proc. 19th Workshop Algorithm Eng. Exp.*, Jan. 2017, pp. 28–42, doi: 10.1137/1.9781611974768.3.

[36] H. Seitz, "Contributions to the minimum linear arrangement problem," Ph.D. dissertation, Univ. Heidelberg, Heidelberg, Germany, 2024. [Online]. Available: http://archiv.ub.uni-heidelberg.de/volltextserver/10578/1/MinLA_Thesis_Seitz.pdf

[37] A. Labarre, "Lower bounding edit distances between permutations," *SIAM J. Discrete Math*, vol. 27, no. 3, pp. 1410–1428, Jan. 2013, doi: 10.1137/13090897x.
[38] T. Tomesh, "Quantum circuit generator: Python package for automated generation of different types of quantum circuits," *GitHub*, 2019. Accessed: 3 Oct. 2024. [Online]. Available: https://github.com/teaguetomesh/quantum_circuit_generator/tree/master
[39] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "QASMBench: A low-level quantum benchmark suite for NISQ evaluation and simulation," *ACM Trans. Quantum Comput.*, vol. 4, pp. 1–26, Jul. 2022, doi: 10.1145/3550488.
[40] A. Javadi-Abhari et al., "Quantum computing with Qiskit," May 2024, *arXiv:2405.08810*, doi: 10.48550/arxiv.2405.08810.
[41] J.-S. Chen et al., "Benchmarking a trapped-ion quantum computer with 30 qubits," *Quantum*, vol. 8, Art. no. 1516, doi: 10.22331/q-2024-11-07-1516.
[42] Y. Wu, S.-T. Wang, and L.-M. Duan, "Noise analysis for high-fidelity quantum entangling gates in an anharmonic linear Paul trap," *Phys. Rev. A*, vol. 97, no. 6, Jun. 2018, Art. no. 062325, doi: 10.1103/physreva.97.062325.
[43] C. Monroe and J. Kim, "Scaling the ion trap quantum processor," *Science*, vol. 339, no. 6124, pp. 1164–1169, Mar. 2013, doi: 10.1126/science.1231298.

Che-Ming Chang received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2024.

He previously interned with the AI for EDA Group, IBM Research Almaden Lab, San Jose, CA, USA. His research interests include electronic design automation (EDA) with emerging technologies and the application of artificial intelligence in EDA.

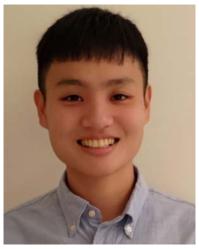
Jie-Hong Roland Jiang (Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1996 and 1998, respectively, and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 2004.

He is currently a Professor with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, where he also leads the Applied Logic and Computation Laboratory. His research interests include logic synthesis, formal verification, electronic design automation, and computation models of biological and physical systems.

Dr. Jiang is a member of Phi Tau Phi and the Association for Computing Machinery.

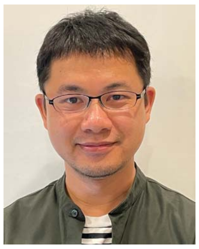
Dah-Wei Chiou received the B.S. and M.S. degrees from National Tsing Hua University, Hsinchu, Taiwan, in 1996 and 1998, respectively, and the Ph.D. degree from the University of California at Berkeley, Berkeley, CA, USA, in 2006, all in physics.

From 2020 to 2022, he was an Assistant Professor with the Department of Physics, National Sun Yat-sen University, Kaohsiung, Taiwan. Since 2023, he has been a Postdoctoral Researcher with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan.



Ting Hsu received the B.S. and M.S. degrees in physics from National Taiwan University, Taipei, Taiwan, in 2015 and 2019, respectively.

From 2019 to 2021, she was a Research Assistant with National Taiwan University. Since 2021, she has been a Research Assistant with the Hon Hai Research Institute, Taipei.



Guin-Dar Lin received the B.S. degree in electrical engineering and M.S. degree in physics from National Taiwan University, Taipei, Taiwan, in 1998 and 2000, respectively, and the Ph.D. degree in physics from the University of Michigan, Ann Arbor, MI, USA, in 2010.

He was a Postdoctoral Researcher with the University of Connecticut, Mansfield, CT, USA, from 2010 to 2012, and with the Institute for Theoretical Atomic, Molecular, and Optical Physics, Harvard-Smithsonian Center for Astrophysics and the Harvard Physics Department, Cambridge, MA, USA, from 2010 to 2013. From 2014 to 2020, he was an Assistant Professor with the Department of Physics, National Taiwan University, where he has been an Associate Professor since 2020. Since 2021, he has also been the Director of the Trapped Ion Quantum Computing Laboratory, Hon Hai Research Institute, Taipei.