Atomique: A Quantum Compiler for Reconfigurable Neutral Atom Arrays

Hanrui Wang[1], Pengyu Liu[2], Daniel Bochen Tan[3], Yilian Liu[4], Jiaqi Gu[5]
David Z. Pan[6], Jason Cong[3], Umut A. Acar[2], Song Han[1]
[1]Massachusetts Institute of Technology, [2]Carnegie Mellon University, [3]University of California, Los Angeles, [4]Cornell University
[5]Arizona State University, [6]University of Texas at Austin

***Abstract*—The neutral atom array has gained prominence in quantum computing for its scalability and operation fidelity. Previous works focus on *fixed atom arrays (FAAs)* that require extensive SWAP operations for long-range interactions. This work explores a novel architecture *reconfigurable atom arrays* (RAAs), also known as *field programmable qubit arrays* (FPQAs), which allows for coherent atom movements during circuit execution under some constraints. Such atom movements, which are unique to this architecture, could reduce the cost of long-range interactions significantly if the atom movements could be scheduled strategically.**

In this work, we introduce Atomique, a compilation framework designed for qubit mapping, atom movement, and gate scheduling for RAA. Atomique contains a *qubit-array mapper* to decide the coarse-grained mapping of the qubits to arrays, leveraging MAX k-Cut on a constructed gate frequency graph to minimize SWAP overhead. Subsequently, a *qubit-atom mapper* determines the fine-grained mapping of qubits to specific atoms in the array and considers load balance to prevent hardware constraint violations. We further propose a router that identifies parallel gates, schedules them simultaneously, and reduces depth.

We evaluate Atomique across 20+ diverse benchmarks, including generic circuits (arbitrary, QASMBench, SupermarQ), quantum simulation, and QAOA circuits. Atomique consistently outperforms IBM Superconducting, FAA with long-range gates, and FAA with rectangular and triangular topologies, achieving significant reductions in depth and the number of two-qubit gates.

***Index Terms*—Quantum Computing; Quantum Compiler; Atom Array; Neutral Atom; Qubit Mapping; Gate Scheduling; Rydberg Atom**

I. Introduction

The landscape in the field of quantum computing is rapidly evolving. Alongside the development of superconducting systems featuring up to 433 qubits [35], [36], [68], [42], [31], [43], the field of neutral atom arrays is also making remarkable strides, in which the architecture of each atom can be used as a qubit. Infleqtion recently announced a groundbreaking 1600 qubit system [37]; Ebadi et al. [22] showcased a system with up to 289 qubits; QuEra introduced a 256-qubit system available on AWS [88], [25], [66]; Pasqal has unveiled a device with 324 qubits [70]; Atom Computing has developed technology supporting over 1000 qubit traps [6], [63]; and Bluvstein et al. [9] also demonstrate implementation of various error correction codes with neutral atom arrays.

This momentum has catalyzed the architecture community to focus on compiler development for these platforms that map and route qubits [48], [45], [61], [73], [21], [1], [27], [33]. Baker et al. [5] pioneered the first compiler framework tailored for neutral atom arrays, adapting qubit mapping strategies from superconducting systems and accommodating the unique advantages and constraints of neutral atoms, such as long-range interaction zones and sporadic atom loss. Building on this, Patel et al. (Geyser) [64] proposed enhancing the system's efficiency by resynthesizing circuits into three-qubit blocks to utilize native multiqubit operations.

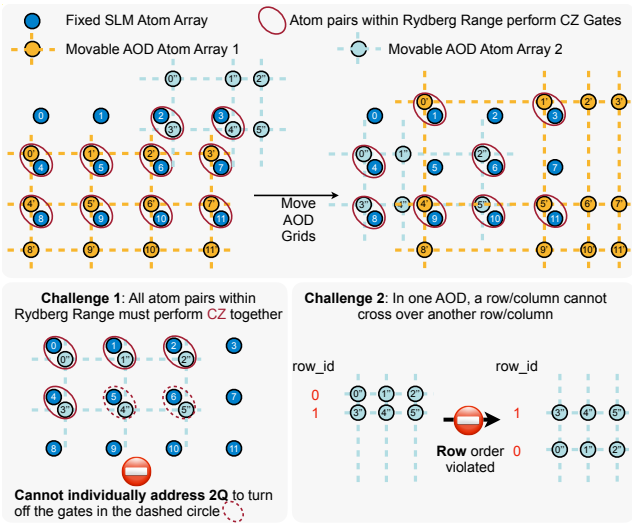


Fig. 1: RAA with fixed & movable atoms.

Although much of the existing research has focused on *fixed atom arrays (FAA)*, a more versatile architecture has surfaced: the *reconfigurable atom arrays* (RAAs), which is also known as *field programmable qubit arrays* (FPQAs). This architecture allows two atoms (qubits) to communicate if they are within the Rydberg radius. Furthermore, the architecture offers fixed atoms, realized using spatial light modulators (SLM), and movable atoms, realized by acousto-optic deflectors (AOD). The AOD atoms can be moved (programmatically) to communicate with other atoms [10], [75], introducing a dynamic coupling map between qubits determined by atom locations. This atom mobility makes it possible for otherwise distant atoms to communicate instead of performing extensive SWAP gates as in superconducting architectures. RAA also allows us to configure the number of qubits and their locations before execution and move atoms during execution, which is the reason why this architecture is given the name: field

979-8-3503-2658-1/24/$31.00 ©2024 IEEE
DOI 10.1109/ISCA59077.2024.00030

programmable qubit arrays.

Fig. 1 illustrates an example with two AOD arrays (represented by yellow and cyan dots) and one SLM array (represented by blue dots). The top subfigure shows a moving step during execution, in which we move the qubits so that the pairs within the Rydberg radius change allowing us to perform different two-qubit gates. Meanwhile, the movement must respect several constraints that the hardware imposes, as depicted in the lower sub-figures. Sec. II discusses more details about atom movement.

As shown in Fig. 2, one-qubit gates can be performed directly using lasers that target atoms from the front, while for two-qubit gates, atoms are moved next to each other and then activated using a laser light from the side, known as a *Rydberg laser*. Importantly, the Rydberg laser affects all atom pairs that are close together, automatically causing them to interact with each other through two-qubit gates. The main takeaway is that while this leads to high-parallel execution, it also means that we have to be very strategic about moving the atoms. We need to ensure that only the desired pairs of atoms are brought close for interaction, to prevent accidental operations between the wrong pairs. Tan et al. [75] presented the first RAA compiler, leveraging a satisfiability modulo theories (SMT) solver for qubit mapping and routing. Despite its ingenuity, the method faces scalability challenges due to the exponential time complexity of SMT solvers. Furthermore, it neglects the detrimental impact of atom transfers between the SLM and AOD arrays; such atom transfers can lead to atom loss [22]. Atom losses can become significant in iterative algorithms like QAOA or trotterized quantum simulations, leading to potential circuit failures.

In this paper, we introduce a scalable compilation framework for RAA, accounting for errors due to atom movements and transfers with additional support for multiple AOD arrays and varying array sizes. Our approach conceptualizes the hardware as a *complete multipartite coupling graph*, which represents all potential qubit couplings between movable array positions. We partition the execution of circuits into stages. In each stage, we select a subset of edges according to the target circuit to activate the two-qubit gates. Using this representation, we are able to reason about the best schedule for atom movements and laser activations.

This framework addresses two key challenges:

- The absence of individual addressability for two-qubit gates, requiring that the selected edge set must only contain atom pairs that need a two-qubit gate.
- The AOD arrays' row/column orders must be preserved, which may forbid two edges being in the same set.

To efficiently address these compilation challenges, we employ a two-tier hierarchical mapping strategy. The strategy divides the mapping problem into two separate problems: (1) mapping groups of qubits to arrays and (2) mapping qubits within an array to atoms in order to maximize the parallel execution of gates without violating the hardware constraints.

Firstly, the coarse-grained **qubit-array mapper** decides the assignments of qubits to atom arrays. The two-qubit gates between atoms from two different arrays can be executed with atom movement, whereas intra-array two-qubit gates cannot be directly executed and require additional SWAPs. Therefore, the mapper tries to maximize the two-qubit gates between arrays by formulating the problem as a MAX k-cut (k is the number of arrays) of a constructed gate frequency graph and solving it with an efficient heuristic.

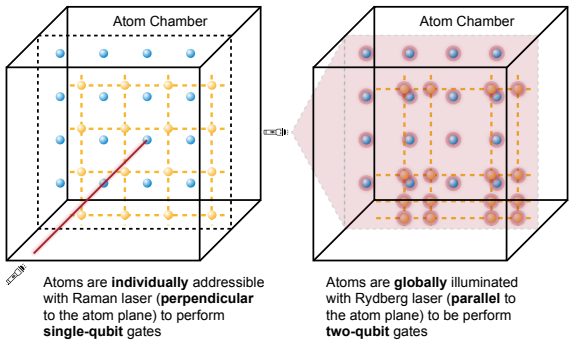


Fig. 2: RAA single-qubit gates and two-qubit gates.

Subsequently, at a more fine-grained level, we propose a **qubit-atom mapper** that maps the qubits to specific atoms in the array. For the SLM array, the qubit-atom mapper considers the load balance among rows and columns to adhere to specific requirements, such as maintaining the order of rows and columns during movements and avoiding overlaps between two rows or two columns. Then, the qubit-atom mapper strategically maps AOD atoms with frequent two-qubit gates to consistent locations across different arrays to increase the execution parallelism.

After completion of this two-level mapping, our **high-parallelism AOD atom router** is activated. In one iteration, it identifies a front layer of non-dependent gates in the transpiled circuit and selects a subset of these gates that satisfies all the hardware constraints. Then, it performs the movements of AOD rows and columns and illuminates the lasers for gate execution. The router works iteratively until the whole quantum circuit is completed.

We extensively evaluate Atomique on 20+ benchmarks of generic circuits (arbitrary, QASMBench, and SupermarQ circuits), Quantum simulation, and QAOA circuits with 5 to 100 qubits and 10 to 10^4 two-qubit gates. We estimate the circuit fidelity under realistic parameters while comprehensively modeling four aspects of movement overhead, including heating, cooling overhead, decoherence, and atom loss.

On average, Atomique achieves 5.6×, 3.4×, 3.5×, and 2.8× in two-qubit gate reduction, 3.7×, 3.5×, 3.2×, and 2.2× in depth reduction over IBM Superconducting, FAA with long-range gates [5], FAA with rectangular topology, and FAA with triangular topology. Atomique is 1000× faster in compilation than a solver-based [75] compiler with similar fidelity. It also reduces the number of pulses by up to 6.5× over Geyser [64].

II. RAA Background

RAA Qubit Movement. In RAAs, atoms are held in two kinds of traps as illustrated in Fig. 1: fixed traps generated by

a spatial light modulator (SLM) and movable traps created by crossed 2D acousto-optic deflectors (AODs). Each 2D AOD is composed of two intersecting 1D AODs, capturing atoms at the points of intersection. The system allows for manipulation of the X/Y coordinates of AOD columns/rows for aligned qubit movements. In particular, within the same AOD set, a column/row is restricted from crossing another (Fig. 1 Challenge 2). However, columns/rows from different AOD sets can intersect. While the figure depicts two AOD sets, control over additional sets has been experimentally validated [28].

RAA Gates. High-fidelity, individually addressable one-qubit gates are induced by a Raman laser (Fig. 2 left) [10], [46], [28]. The two-qubit entangling gates are achieved through the Rydberg blockade mechanism [38]. When two atoms are within the *Rydberg range* r_b and illuminated by a Rydberg laser, a controlled Z-rotation is induced (Fig. 2 right). We need to move two qubits within r_b of each other for two-qubit gates or to separate them by $\geq 2.5r_b$ when no gate is performed (Fig. 1 Challenge 1).

Pros & Cons of Neutral Atom Arrays in General. Neutral atoms enjoy the same benefits as other natural qubits, e.g., ions, compared to fabricated qubits, e.g., superconductors: high-fidelity gates, ease of calibration, and virtually limitless source. Compared to electrical traps used by ions, optical traps for neutral atoms can be fabricated as 2D rather than 1D and thus scale to much larger sizes, e.g., 1020 SLM traps demonstrated in [23] which can, in theory, trap 1020 atoms and then later demonstrated trapping of more than 1200 atoms in [63]. In the long run, the scalability of RAAs can still be challenging at a few-thousand-qubit scale because they are built from quantum optics experimental apparatus not initially designed for large-scale quantum computing. However, this limitation is comparative, as other platforms lag behind in scaling so far. Companies such as QuEra and PASQAL have begun to develop hardware with scalability in focus.

Pros & Cons of RAAs vs FAA. In FAA [28], [5], [64], [69], [7], fixed SLM atoms restrict qubit connectivity, requiring an additional AOD array to modulate Rydberg interactions between selected adjacent pairs. This limits scalability because twice as many SLM atoms and a similarly sized AOD array are needed compared to RAA. This also compromises `CZ` fidelity; e.g., 92.5% in FAA [28] versus 97.5% in RAA [10]. FAA needs SWAP gates ($\sim$3 `CZ`s) to route the qubits for two-qubit gates. In contrast, RAA's AOD movements are high-fidelity, limited primarily by coherence time. Ref. [10] estimates that with only 0.1% of coherence time lost, an AOD array could traverse a region hosting $\sim 2,000$ qubits. A drawback of atom movement and transfer is the risk of atom loss, jeopardizing the integrity of the entire quantum circuit. Thus, a careful comparison of these two routing methods is required, as we shall see in this paper.

Application-Specific Compilation. We contrast Atomique with three existing application-specific compilers: ZZ [2], [3], [4], 2QAN [45], and Paulihedral [50]. ZZ focuses on the commutation of ZZ gates in QAOA; our approach generalizes this by avoiding suboptimal layering of ZZ gates. 2QAN minimizes SWAPs by absorbing them into existing two-qubit unitaries, a technique less relevant for RAAs, which instead employs atom movement for routing. Paulihedral presents two strategies: 1) Optimizing Pauli term order to reduce circuit depth, a technique orthogonal to our work and applicable to pre-routing; 2) Tailoring each Pauli term for limited-connectivity NISQ hardware, but less applicable to RAAs.

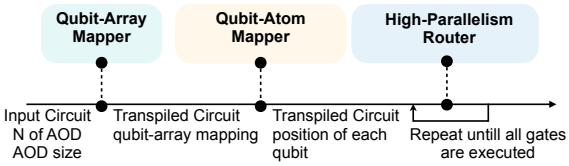


Fig. 3: Overall pipeline of Atomique.

III. Atomique Compilation Framework

The overall compilation flow is shown in Fig. 3. Our compiler contains a *qubit-array mapper* to determine the array in which each qubit should be placed and a *qubit atom mapper* to decide the specific position of each qubit inside the mapped array. After deciding all qubit positions, we also design a *high-parallelism router* to generate the movements and gate schedules of the rows and columns of the AOD array, respecting hardware constraints and maximizing parallelism.

As shown in Fig. 4, the RAA features one SLM array and multiple AOD arrays, represented by different colors in the Fig. 4 top. Qubits are fixed in the SLM array, and atom pairs in the SLM array will never fall within the Rydberg radius, precluding intra-array two-qubit gates. For atom pairs in the same AOD array, our compiler also avoids intra-array two-qubit gates because of the risk of atom loss. Consequently, two-qubit gates can only be performed between two different arrays. Since there is only one SLM array, at least one of the two arrays will be an AOD array, and thus the gate can be executed via atom movement. This can be represented by a complete multipartite graph for feasible two-qubit gate operations. As a result, the atoms in the same array are equivalent from the coupling map perspective. Therefore, we tackle the mapping problem hierarchically: firstly, deciding which array a qubit should be mapped to, and secondly, determining the specific position for the qubit in that array.

A. Qubit-Array Mapper

Ideally, if all gates in a circuit involve qubits in different arrays, there will be no SWAP overhead. To this end, Atomique tries to find a mapping to maximize the inter-array gates, and thus minimize SWAP overhead. This goal can be approached with a *two-qubit gate frequency graph* as shown in Fig. 4 bottom. In the graph, each vertex represents a qubit and an edge indicates that a gate exists between two qubits. The edge weights are determined by the frequency of 2-qubit gates between the qubit pairs. More gates between a qubit pair will result in a large edge weight. Then, finding the mapping that allows the most directly executable inter-array two-qubit gates is translated to a **MAX k-Cut** of the gate frequency graph, where the graph is divided into k

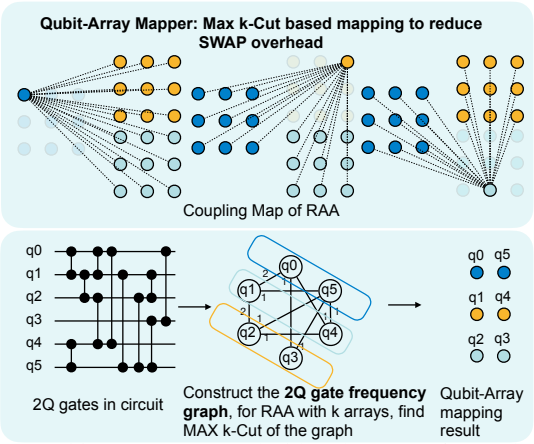


Fig. 4: Qubit-array mapper decides the array for each qubit with MAX k-Cut to reduce SWAP overhead.

Algorithm 1: Qubit-Array Mapper

Input : Quantum circuit C with n qubits
Input : Number of AODs m
Output: Mapping of qubits to AODs in dictionary M
$E \leftarrow n \times n$ zero matrix; `// Adjacency matrix for the gate frequency graph.`
$M \leftarrow$ empty dictionary; `// Maps qubits to AODs, M[i] are the qubits that are mapped to ith AOD`
for *each 2Q gate G in C* **do**
  $E[i][j] += \gamma^l$; `// G acts on qubit i and j, and lies in the l-th layer of the circuit.`
end
for $i = 1$ *to* n **do**
  $Bestcut \leftarrow 0$;
  $Mapto \leftarrow 0$;
  for $j = 1$ *to* m **do**
    $Currentcut \leftarrow \sum_{k \notin M[j]} E[i][k]$;
    if $Bestcut < Currentcut$ **then**
      $Bestcut \leftarrow Currentcut$;
      $Mapto \leftarrow j$;
    end
  end
  Add i to $M[Mapto]$;
end

partitions, corresponding to k arrays in RAA, with the aim of maximizing the summation of edge weights crossing different partitions. Although this problem is NP-hard, we adopt a straightforward greedy algorithm that yields an approximation of $1 - \frac{1}{k}$ [15]. The greedy algorithm decides the partition assignment of each vertex one by one while ensuring that each vertex can maximize the cut between partitions of already assigned vertices. The algorithm is outlined in Alg. 1.

For the calculation of the specific edge weight, each gate between a qubit pair will contribute a weight that exponentially decreases according to how many layers the gate is from the first layer in the circuit DAG and is controlled by a factor γ. We add decay because, for gates in the later layers, we have less control over the qubit positions, and the benefit this gate can have from a better mapping is less.

After finding the initial qubit-array mappings, we leverage the default SABRE [48] in Qiskit with the multipartite coupling graph to insert SWAP gates and pass the mapping and

Qubit	Involved 2Q Gate Count Rank
q16	1
q23	2
q4	3
q22	4
q5	5
q19	6
q24	7
q26	8
...	...

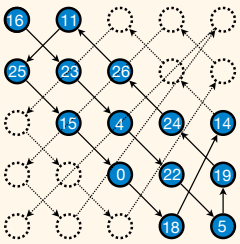


Fig. 5: Qubit-atom mapper first maps SLM qubits with "load balance mapping" to avoid potential constraint violations.

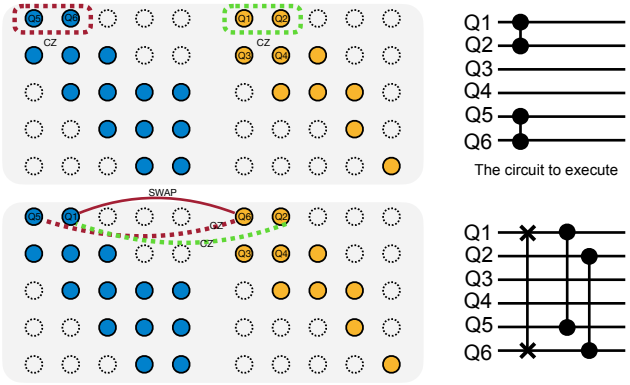


Fig. 6: An example showing why SWAP is still needed and how they are applied.

transpiled circuit to the qubit-atom mapper. Fig. 6 provides an example of how SWAP is inserted. In this instance, our aim is to execute two-qubit gates between Q1 and Q2, as well as between Q5 and Q6. However, due to hardware limitations, neither operation is directly feasible; certain movements may lead to undesired interactions, while others are physically prohibited. We address this issue by introducing an additional SWAP insertion step. Specifically, we swap Q1 with Q6 using three `CZ` and one-qubit gates, enabling the direct execution of all the two-qubit gates.

B. Qubit-Atom Mapper

Atomique further decides the fine-grained position of each qubit in each array. This step does not impact the SWAP overhead. Instead, it essentially focuses on selecting the optimal atom positions in the array to minimize circuit depth. It is crucial to note that once the atoms are positioned, the absolute positions of the SLM atoms and the relative positions of AOD atoms are immutable during circuit execution. Suboptimal mapping can diminish opportunities for parallel gate execution, thus elongating the circuit depth. In the worst-case scenario, all two-qubit gates need to be executed sequentially, even without circuit-level dependencies. Therefore, the primary goal of our qubit-to-atom mapping is to maximize parallel-gate executions. To achieve this, we perform qubit-atom mapping in two steps: firstly, map all SLM qubits, and secondly, map all AOD qubits.

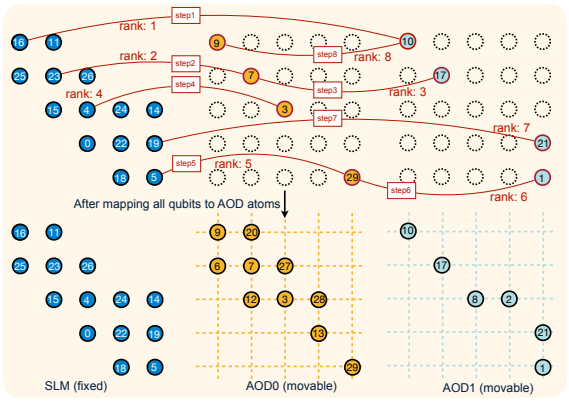


Fig. 7: Qubit-atom mapper then maps AOD qubits based on gate frequency rank.

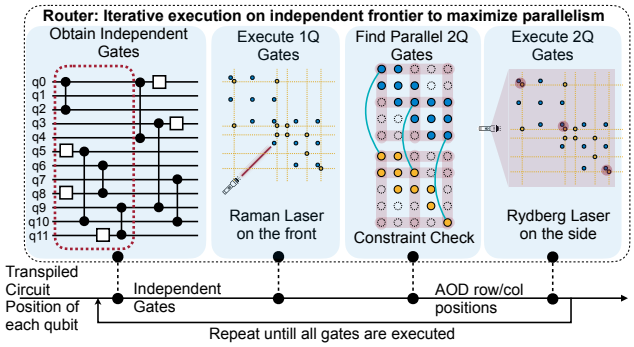


Fig. 8: The pipeline of the high-parallelism router.

First, it is crucial for qubit-atom mapping of SLM qubits following the design principle of load balance between rows/-columns. Specifically, we need to balance the number of atoms across rows because most qubits concentrating in a few rows will create more conflicts within the same row. The column load balance is similar. To this end, we propose the **load balance SLM mapping**, as illustrated in Fig. 5. We first sort the qubits in descending order based on the number of two-qubit gates they involved. Then, we map the qubits to atoms with a topological order starting from the upper-left corner, prioritizing filling the diagonal first and subsequently forming a spiral trajectory. This will ensure that the row-wise/column-wise sum of involved two-qubit gates of qubits is balanced across all rows/columns. The load balance can, in return, maximally avoid violations of the constraint1 on unwanted gates (Fig. 9) and constraint3 that rows/columns cannot overlap as illustrated (Fig. 11). Two atoms from a near-diagonal region have a greater chance of differing in row/column index.

Second, we need to map the AOD qubits. The design principle for AOD mapping is position alignment for frequent qubit pairs. Specifically, we propose **alignment AOD mapping** to make sure that the pairs of atoms of the same positions in two arrays will have highly frequency two-qubit gates so that we naturally enable higher parallelism if we move and align

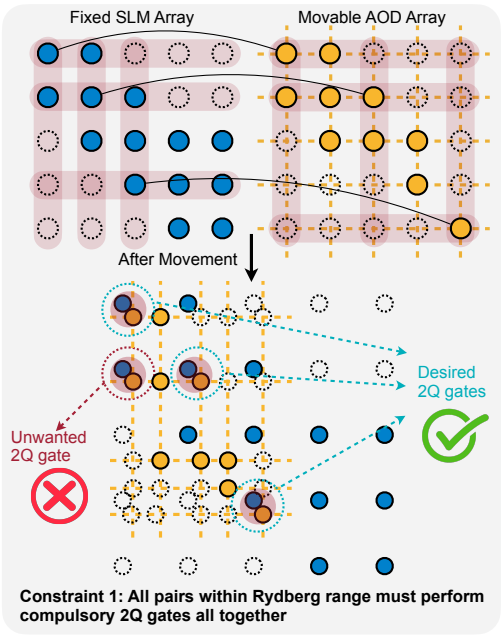


Fig. 9: The first constraint checked by the router.

two arrays. We first sort the frequency of qubit pairs with two-qubit gates; then, the frequent pairs will be mapped to the same spatial location in two arrays as the example shown in Fig. 7. The SLM qubits atom mapping has already been determined. The "rank" is determined by the two-qubit gate frequency in decreasing order. The Q16 and Q10 have the highest frequency, so we map Q10 to the same top left corner as Q16 in the SLM array. Similarly, for the second rank gate between Q23 and Q7, we map Q7 to the same row2 column2 position in the yellow AOD0 as Q23 in SLM. This mapping strategy will avoid the violation of Constraint 2 on row / column orders (Fig. 10) because the high-frequency gates will have a small relative displacement.

C. High-Parallelism AOD Router

With the specific qubit-atom mappings and the transpiled circuit, we propose a router for AOD atom movements and gate scheduling. The router's responsibility is to generate the instructions on the movement of rows and columns of all AODs and decide when to turn on the lasers to perform gates. Meanwhile, it must respect all the hardware constraints during movements and optimize for high parallelism.

The pipeline is depicted in Fig. 8. The router first identifies independent "frontier gates" from a direct acyclic graph (DAG) circuit representation. Then, it executes all single-qubit gates by illuminating the Raman lasers. Although two-qubit gates do not have circuit-level dependencies and could theoretically be executed in parallel, moving them all to the corresponding atom may violate the hardware constraints. Therefore, the router greedily finds maximally legal parallel two-qubit gates. Starting with a single gate, the router incrementally adds gates to the set, assessing three hardware constraints. If violated, the two-qubit gate will not be added to the set but will be pushed

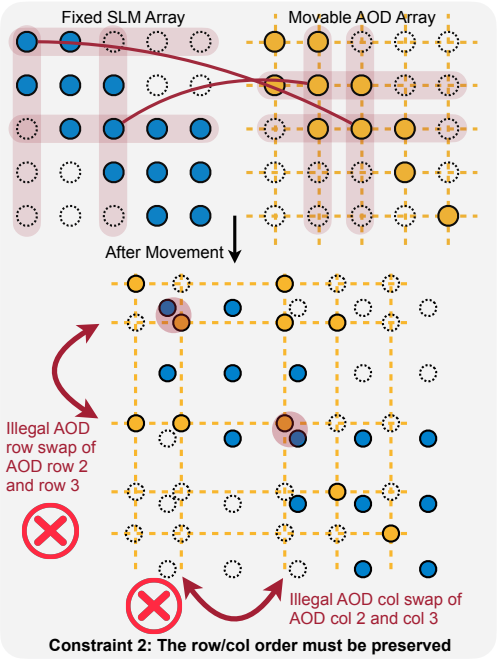


Fig. 10: The second constraint checked by the router.

back to the circuit DAG. After finding the legal parallel two-qubit gate set, the router will move the AOD rows and columns to the target positions and turn on the Rydberg laser to perform two-qubit gates. This router runs iteratively until the entire circuit is finished.

The first hardware constraint is that all pairs within the Rydberg range must perform gates together as in Fig. 9. In the figure's top part, three black connection lines indicate the valid gates in the circuit. If we perform the movement to perform those gates simultaneously, as in the bottom part, we will introduce an unwanted gate marked in red. Therefore, this set of gates cannot be executed in parallel. The second hardware constraint is shown in Fig. 10: the row order and column order of AOD arrays must be preserved, i.e., there cannot be a position swap of two rows/columns. In the figure, we show that to execute two two-qubit gates annotated by two red lines on the top, we have to swap row 2 and row 3 and swap column 2 and column 3, as illustrated in the bottom part. This is illegal and thus will not be accepted by the router. The third constraint is that the AOD rows and columns cannot overlap as in Fig. 11. To finish the two gate pairs connected by red lines, AOD rows 4 and 5 must be moved to the same position and overlap, which is not allowed.

IV. Atom Movement Overhead

We characterize atomic motion overhead through four metrics, as formulated in Eq. 1: additional two-qubit error due to heating, $F_{\text{mov\_heating}}$; atom loss due to heating, $F_{\text{mov\_loss}}$; additional overhead due to cooling, $F_{\text{mov\_cooling}}$; and decoherence during movement, $F_{\text{mov\_deco}}$.

$$F_{\text{mov}} = F_{\text{mov\_heating}} \times F_{\text{mov\_loss}} \times F_{\text{mov\_cooling}} \times F_{\text{mov\_deco}} \quad (1)$$

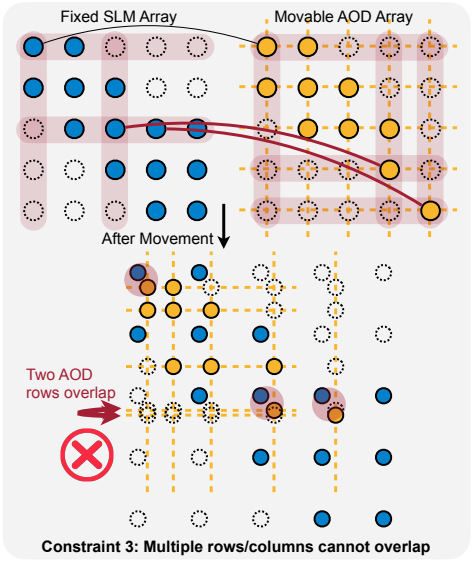


Fig. 11: The third constraint checked by the router.

Atom Heating. Atom movement causes heating and degrades two-qubit gate fidelity, necessitating the computation of the "vibrational quantum number", n_{vib}, for each atom. This quantity encapsulates an atom's cumulative vibrational energy due to heating. Our framework tracks each atom's n_{vib} based on motion history.

$$F_{\text{mov\_heating}} = \prod_{i=1}^{N_{2Q}} (1 - \lambda \times (1 - f_{2Q}) \times n_{\text{vib}}(i)) \quad (2)$$

According to [10], heating's fidelity impact is proportional to both two-qubit gate infidelity, $(1 - f_{2Q})$, and n_{vib}, as described in Eq. 2. In interactions involving an AOD and an SLM atom, the n_{vib} of the AOD atom is considered; for AOD-AOD interactions, the n_{vib} values are summed.

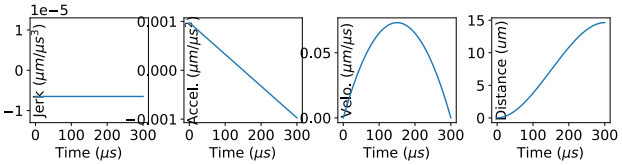


Fig. 12: The atom movement pattern.

Ref. [10] propose a constant negative jerk (the derivative of acceleration) strategy as delineated in Figure 12 to minimize n_{vib}, resulting in a linearly decreasing acceleration and a parabolic velocity profile. The incremental change in n_{vib} for a single movement is: $\Delta n_{\text{vib}} = \frac{1}{2}\left(\frac{6D/x_{\text{zpf}}}{\omega_0^2 T_{\text{mov}}^2}\right)^2$, where D is the movement distance; $x_{\text{zpf}} \equiv \sqrt{h/2\pi \cdot (2m\omega_0)}$ is the zero-point size of the particle; h is the Planck's constant; ω_0 is the trap frequency and T_{mov} is the duration of this move. From this equation, we can obtain the insights: $n_{\text{vib}} \propto N_{\text{move}}/T_{\text{mov}}^4$, indicating that a minor increase in T_{mov} allows a substantially greater N_{move} before the heating error is too large, albeit at the cost of increased decoherence error. This trade-off is further analyzed in Section V.

TABLE I: Hardware Parameters

Parameter	2Q fidelity	1Q fidelity	2Q gate T	1Q gate T	Coherence T	Atom distance
Neu. Atom	0.9975[10]	0.99992[10]	380ns[10]	625ns[10]	15s[10]	15µm[10]
Supercon.	0.9975	0.99992	480ns[34]	35.2ns[34]	801.2µs[34]	-

Parameter	T per move	Atom transfer T	Atom loss P	x_{zpf}	ω_0	λ
Neu. Atom	300µs[10]	15µs[10]	0.0068[16]	38nm[10]	80kHz[10]	0.109[10]

Here, we analyze when doing atom movement is better than adding SWAPs. According to [10], $x_{\text{zpf}} = 38\text{nm}$, and ω_0 can be between 40KHz and 80KHz. We use 80KHz. Movement time T_{mov} can be between 200µs to 400µs and we select 300µs. Atom distance, the separation between the rows and columns of the atom, needs to be greater than 6 times the Rydberg radius, which is $6 \times 2.5 = 15\mu m$. According to [10] Extended Fig. 5d movement path and Extended Fig. 6c in which the finite-temperature (Doppler) error rate is 0.3%, we compute the λ in Eq. 2 as 0.109. With these parameters, movement-induced n_{vib} increases are calculated to be $\Delta n_{\text{vib}} = 0.0054$ for a 1 hop movement, 0.13 for 5 hops, and 0.54 for 10 hops. In the experiments shown in Sec. V, the cost of routing two-qubit gates is around 2× the logical two-qubit gates in FAA. Eq. 2 says the additional error incurred by heating is λn_{vib} times the two-qubit error. So, when $n_{\text{vib}} < \frac{2}{\lambda} \approx 18$, which corresponds to fewer than 1000 gates, we expect better performance using RAA atom movement, since the overhead of movement will be less than that of additional SWAP gates.

Atom Loss. The heating and vibration also increase the atom loss probability. According to [10], the atom loss probability of moving and atom loss fidelity are: $P_{\text{mov\_loss}} = 1 - \frac{1}{2}\left(1 + \text{erf}\left[\frac{n_{\text{vib}}^{\max} - n_{\text{vib}}}{\sqrt{2 n_{\text{vib}}}}\right]\right)$, $F_{\text{mov\_loss}} = \prod_{i=1}^{N_{\text{move}}} \prod_j (1 - P_{\text{mov\_loss}}(i,j))$, where j indicates the moved atoms in a movement. In [10], $n_{\text{vib}}^{\max} = 33$. When $n_{\text{vib}} = 30$, $F_{\text{mov\_loss}} = 0.708$; when $n_{\text{vib}} = 20$, $F_{\text{mov\_loss}} = 0.998$; when $n_{\text{vib}} = 15$, $F_{\text{mov\_loss}} = 0.999998$. We see a fast improvement as n_{vib} is reduced. Thus, we establish a n_{vib} threshold of 15 for cooling, beyond which the impact on atom loss is negligible.

Cooling Overhead. When any atom's n_{vib} exceeds this threshold, cooling is initiated for the entire AOD array by swapping its quantum information with a pre-prepared, cooled AOD array initialized to the zero state. This information transfer requires two CZ gates for each atom, so the overhead is below. When the threshold is 15, cooling is required around every 100 gates. $F_{\text{mov\_cooling}} = \prod_i^{N_{\text{cooling}}} f_{2\text{Q}}^{2 \times N_{\text{AOD}}}$.

Decoherence. Finally, we model the decoherence due to the time overhead of moving atoms. $F_{\text{mov\_deco}} = \prod_i \exp(-N_i \times T_{\text{mov,i}}/T_1)$, where $T_{\text{mov,i}}$ is the moving time of stage i, and N_i is the number of qubits, including AOD and SLM used at stage i. In assessing fidelity overheads, we juxtapose additional SWAPs in FAA with decoherence costs in RAA. According to [10], the fidelity for a two-qubit (two-qubit) gate is $f_{2\text{Q}} = 0.975$, and $T_1 = 1.5s$. In the next section, our experiments show that the routing overhead for each two-qubit gate in FAA is around two additional two-qubit gates, reducing the overall fidelity by $f_{2\text{Q}}^2 = 0.95$. For RAA, the decoherence overhead of one additional two-qubit gate in a 10-qubit circuit is $F_{\text{mov\_deco}} = \exp(-300 \times 10^{-6}/1.5 \times 10) = 0.998$. This overhead scales to 0.99 and 0.98 for 50- and 100-qubit circuits, respectively. Consequently, RAA's overhead is smaller than adding two two-qubit gates in FAA. Remarkably, this advantage will persist in the forecasted future. Suppose that both $f_{2\text{Q}}$ and T_1 are scaled by an order of magnitude to 0.9975 and $15s$, respectively. The revised overhead for 10, 50, and 100 qubits becomes 0.9998, 0.999, 0.998, while the FAA routing overhead will be 0.995, maintaining RAA's benefit.

TABLE II: Benchmarks

Name	Type	Dataset	Num. Qubits	Num. 2Q gates	Num. 1Q gates	2Q Gate per Q	Degree per Q	Tan-Solver	Tan-IterP
HHL	Generic	QASMBench	7	196	794	56.0	5.7	timeout	solved
Mermin-Bell	Generic	SupermarQ	10	67	30	13.4	7.6	timeout	solved
QV	Generic	QASMBench	32	1536	4096	96	19.7	timeout	solved
BV50	Generic	QASMBench	50	22	100	0.9	0.9	timeout	solved
BV70	Generic	QASMBench	70	36	417	1.0	1.0	timeout	solved
QSim-rand-20	QSim	-	20	180	275	18.0	4.2	timeout	solved
QSim-rand-40	QSim	-	40	380	567	19.0	5.7	timeout	solved
H2	QSim	Molecule	4	40	54	20.0	3.0	solved	solved
LiH	QSim	Molecule	6	1134	1602	283.5	6.0	timeout	solved
QAOA-rand-10	QAOA	-	10	27	10	5.4	5.4	timeout	solved
QAOA-rand-20	QAOA	-	20	80	20	8.0	8.0	timeout	solved
QAOA-regu5-40	QAOA	-	40	100	40	5.0	5.0	timeout	solved
QAOA-regu6-100	QAOA	-	100	300	100	6.0	6.0	timeout	timeout
Mermin-Bell	Generic	SupermarQ	5	19	15	7.6	4.0	solved	solved
VQE	Generic	SupermarQ	10	9	40	1.8	1.8	solved	solved
VQE	Generic	SupermarQ	20	19	80	1.9	1.9	solved	solved
Adder	Generic	QASMBench	10	65	101	13.0	2.6	solved	solved
BV	Generic	QASMBench	14	13	81	1.9	1.9	solved	solved
QSim-rand-5	QSim	-	5	20	29	8.0	2.4	solved	solved
QSim-rand-10	QSim	-	10	80	122	16.0	4.6	solved	solved
QAOA-rand-5	QAOA	-	5	6	5	2.4	2.4	solved	solved
QAOA-regu3-20	QAOA	-	20	30	20	3.0	3.0	solved	solved
QAOA-regu4-10	QAOA	-	10	20	10	4.0	4.0	solved	solved

V. Evaluation

A. Evaluation Methodology

Fidelity Estimation. We model the fidelity with four terms: $F = F_{1\text{Q}} \times F_{2\text{Q}} \times F_{\text{transfer}} \times F_{\text{mov}}$. F_{mov} has been discussed in Sec. IV. $F_{1\text{Q}}$ is the fidelity impact of executing one-qubit gates: $F_{1\text{Q}} = f_{1\text{Q}}^{N_{1\text{Q}}} \times \exp(-T_{1\text{Q}}/T_1 \times N)$, where $f_{1\text{Q}}$ is the fidelity of a single one-qubit gate; $N_{1\text{Q}}$ is the total number of one-qubit gate after compilation; $T_{1\text{Q}}$ is the cumulative time of one-qubit gates; T_1 is the coherence time; and N is the total number of qubits. Two-qubit fidelity is similar: $F_{2\text{Q}} = f_{2\text{Q}}^{N_{2\text{Q}}} \times \exp(-T_{2\text{Q}}/T_1 \times N)$. F_{transfer} is the fidelity impact of performing atom transfer between SLM and AOD: $F_{\text{transfer}} = (1 - P_{\text{loss}})^{N_{\text{transfer}}} \times \exp(-T_{\text{transfer}}/T_1 \times N)$, where P_{loss} is the atom loss probability; N_{transfer} is the number of transfers; and T_{transfer} is the cumulative time of transfers.

We build our own fidelity model because current commercial hardware or simulators cannot satisfy our requirements. For example, the hardware available from QuEra is fundamentally different from ours because it lacks movable atoms for now. Also, their hardware supports only pulse-level analog quantum computing, and the corresponding simulator, Bloqade, supports only Hamiltonian simulation. Neither supports the application of a quantum gate, requiring us to program the Hamiltonian directly. Investigating the correct pulse for a gate is beyond the scope of this paper. Finally, Bloqade does not support noisy simulation, thus preventing us from evaluating the performance of various compilers under noise.

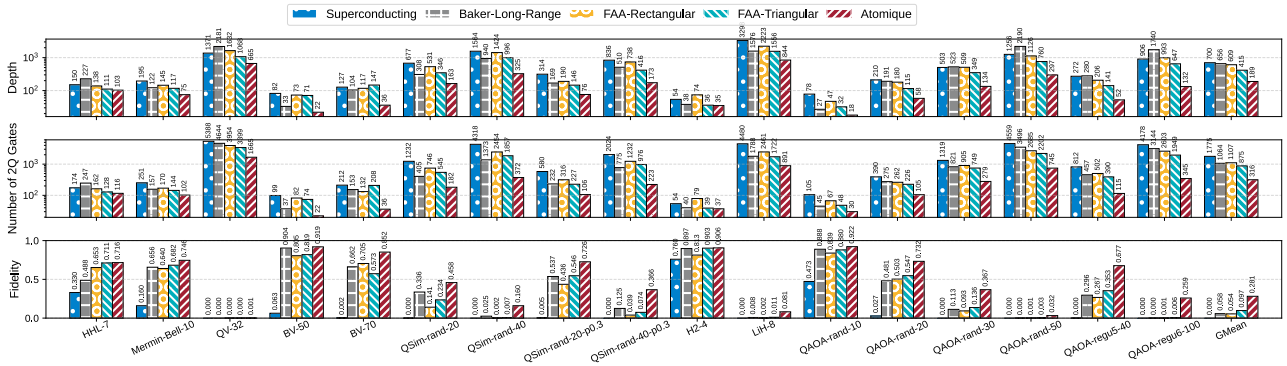


Fig. 13: Circuit depth, fidelity, two-qubit gate count comparisons between different architectures.

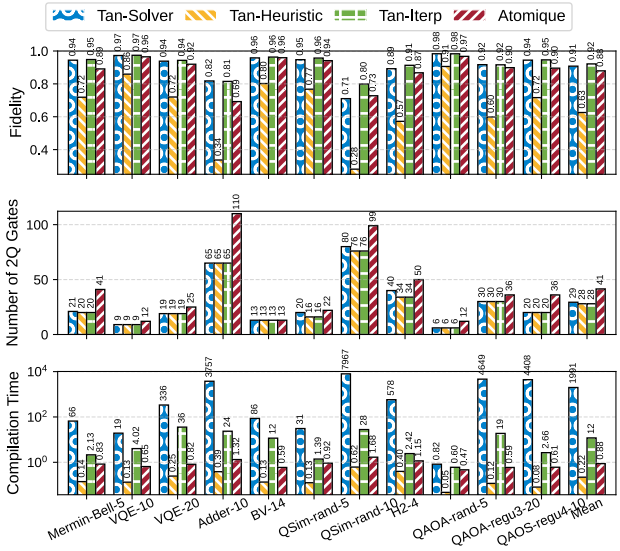


Fig. 14: Comparisons to Tan-Solver, Tan-Heuristic, and Tan-IterP. Atomique has similar fidelity but over 1000× faster.

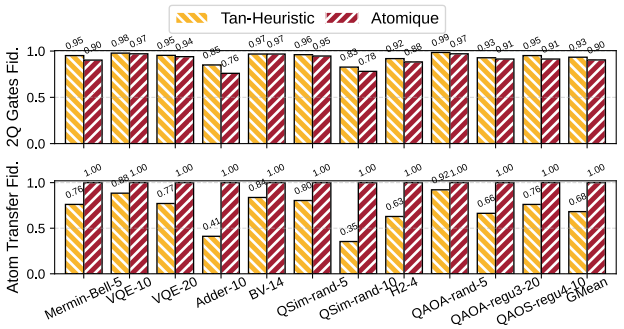


Fig. 15: Breakdown of major error sources of Tan-Heuristic and Atomique.

Hardware Parameters. In the neutral atom device cited from [10], a CZ gate is realized through two global Rydberg pulses, each lasting 190ns, leading to a two-qubit gate duration of 380ns and fidelity of 0.9975. Arbitrary single-qubit gates employ a 'BB1' pulse sequence, achieving a one-qubit gate time of 625ns and a fidelity of 0.9992, calculated based on a scattering error of 2×10^{-4} per π pulse. The atom transfer time is $15\mu s$ with a loss probability of 0.0068 [5]. Parameters such as SLM atom distance, move time, coherence time, $x_{\text{zpf}}, \omega_0, \lambda$ have been addressed in Section IV. Our default configuration is 10×10 topology with 1 SLM array and 2 AOD arrays. For baseline architectures, we equalize qubit numbers with those reported in Atomique. For superconducting qubits, parameters are derived from the IBMQ Experience platform [34]. Fidelity metrics for two-qubit and one-qubit gates align with those for neutral atoms for unbiased comparisons. We scale up the coherence time for superconducting and neutral atom devices by 10x and scale down their two-qubit and one-qubit gate errors to make evaluation on large quantum circuits possible.

Tab. I summarizes the parameters.

Benchmarks. To assess compiler performance, we utilize three benchmark categories: algorithmic (generic), quantum simulation (QSim), and Quantum Approximate Optimization Algorithm (QAOA). Their characteristics are detailed in Table II. The metric "Degree per Q" represents the average count of unique qubits that interact with a given qubit. Algorithmic circuits are sourced from SupermarQ [81] and QASMBench [47]. QSim circuits are randomly generated with a probability of 0.5 for a qubit to exhibit a non-I Pauli operator, and each circuit comprises ten Pauli strings. For molecular Hamiltonians, we consider H2 and LiH. QAOA circuits are constructed by randomly placing ZZ gates between all pairs of qubits with a probability of 0.5. In addition, QAOA circuits based on regular graphs are generated, in which ZZ interactions are placed to qubit pairs with an edge in the regular graph. These benchmarks span a broad spectrum featuring a variety number of qubits and circuit depth.

Baselines. To show the unique advantage of RAA, we establish a comparison with four other distinct architectures, each using a specialized compiler, along with two additional compilers for RAA. (1) **Superconducting**: Utilizing IBM's 127-qubit Washington superconducting machine with a heavy hexagon coupling graph. (2) **Baker-Long-Range**: A fixed rectangular array of atoms with long-range interactions. The compiler is as proposed by Baker et al. [5], and the maximum interaction range is set to four times the Rydberg radius. We

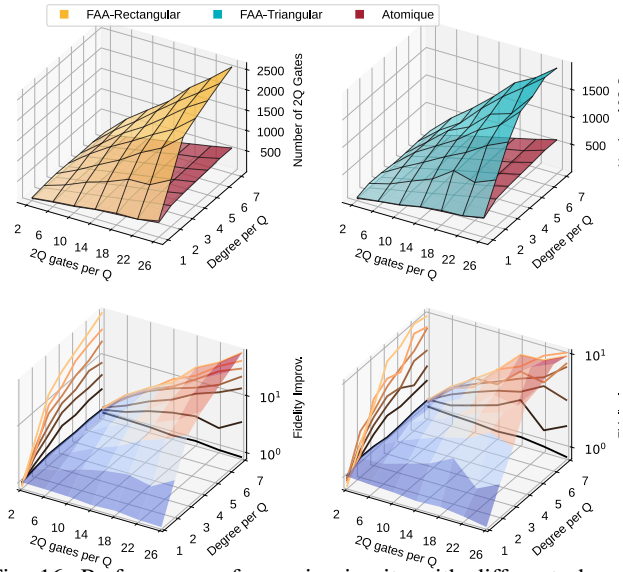


Fig. 16: Performance of generic circuits with different characteristics on different architectures.

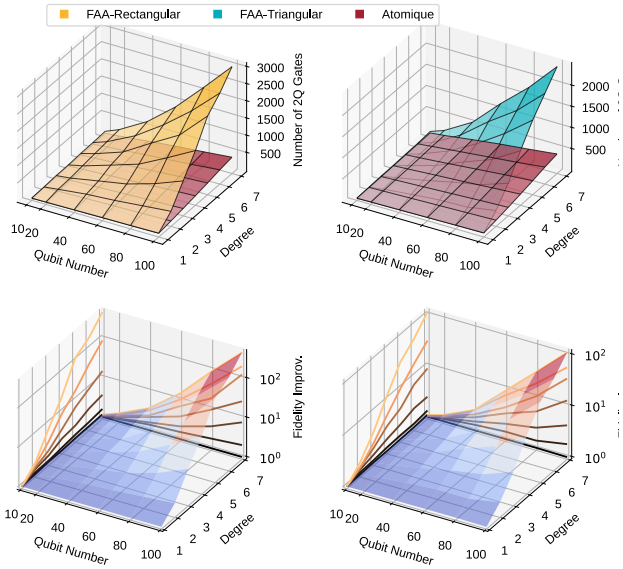


Fig. 17: Performance of QAOA circuits with different characteristics on different architectures.

use the open-source by the authors. (3) **FAA-Rectangular**: A fixed rectangular array of atoms that permits only nearest-neighbor interactions. (4) **FAA-Triangular**: Fixed triangular arrays of atoms introduced in [64]. (5) **Tan-Solver** and **Tan-IterP**: we compare our compiler with two solver-based compilers [75], [78]. Ref. [78] provides a tool, OLSQ-DPQA, where a parameter 'optimal ratio' can be set to switch between the two methods: 1 for **Tan-Solver** and to 0 for **Tan-IterP**. In the latter case, the SMT formulation is relaxed in a greedy manner, 'iterative peeling', to support compiling larger circuits. A few settings are worth mentioning: 1) we let OLSQ-DPQA use 16x16 atoms per SLM or AOD array, 2) the internal Z3 SMT solver [19] has version 4.13.0.0, and 3) we impose a

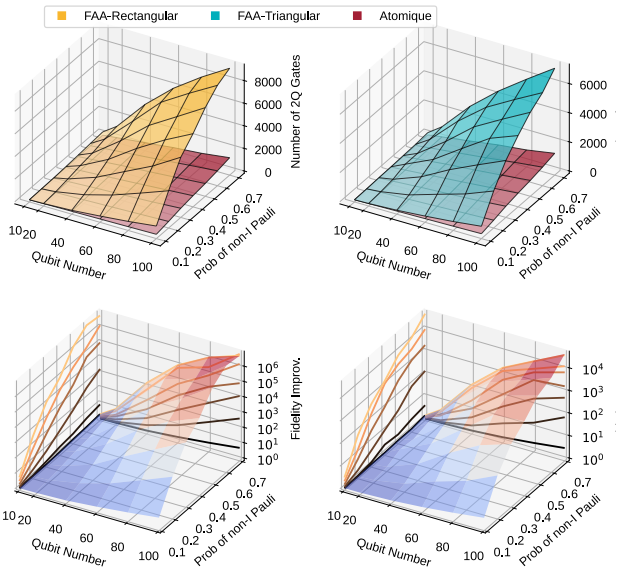


Fig. 18: Performance of Quantum Simulation circuits with different characteristics on different architectures.

24-hour timeout. (6) **Tan-Heuristic**: We also compare with a heuristic strategy introduced in [75]. It allows circuit execution on RAA by positioning all qubits in the SLM and utilizing atom transfers and movements. For a `CZ` gate involving qubits i and j, the heuristic moves one qubit to the other via an AOD atom to perform the `CZ`, and then repositions it. This heuristic serves as an additional baseline. For experimental consistency, all tests run on a server equipped with a 40-core Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz and 192GB of RAM. All baselines are using Qiskit Optimization Level 3 with SABRE [48].

B. Main Results

Comparison of RAA with Other Quantum Architectures. Fig. 13 shows Atomique outperforming four alternative architectures in fidelity, two-qubit gate count, and depth (number of parallel two-qubit layers), exceeding the strongest baseline, FAA-Triangular, by factors of 2.7×, 2.8×, and 2.2×, respectively. Large and complicated circuits such as QSimrand benefit the most due to higher connectivity and lower SWAP overhead. In simpler circuits, such as H2 simulations, different architectures perform comparably.

Comparison of Atomique with Other Compilers on RAA. As indicated in the last column of Table II, Tan-Solver [75] is infeasible for circuits with more than 20 qubits within a 24-hour limit. Consequently, smaller circuits are used for comparisons in Fig. 14. For a fair comparison, Atomique employs a single AOD, as two baselines lack multi-AOD support. Tan-Heuristic [75] exhibits the fastest compilation time but suffers from low fidelity due to excessive atom transfers, making it unsuitable for large circuits. In contrast, Atomique maintains an average fidelity of 88%, comparable to solver-based compilers, while being more than 2,000 times faster for small-scale problems. This speed advantage is exacerbated for

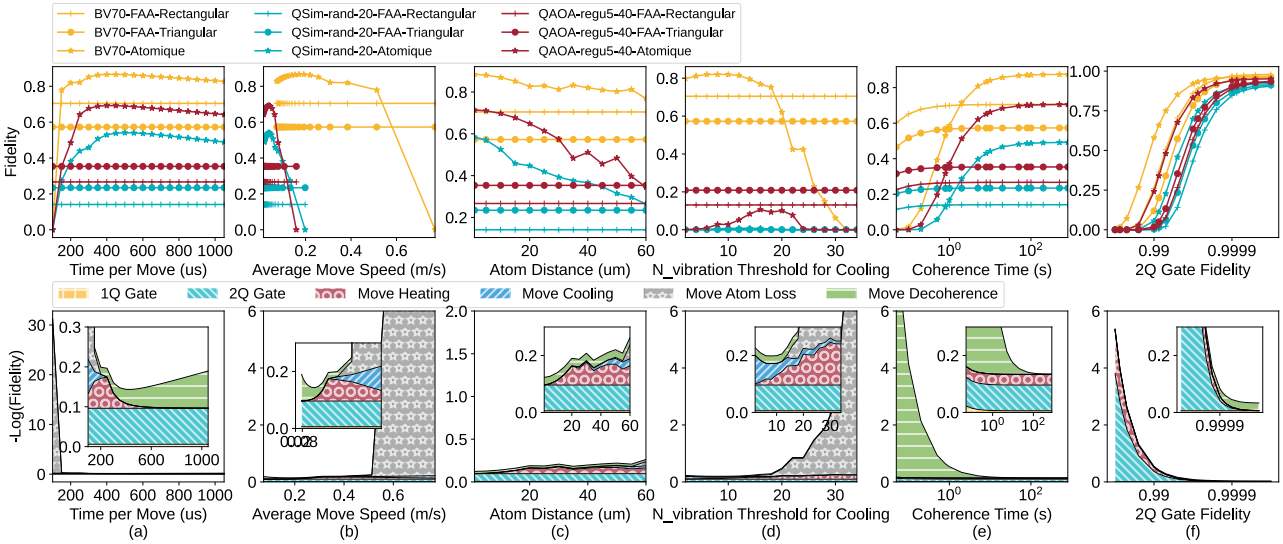


Fig. 19: Sensitivity analysis of various hardware parameters.

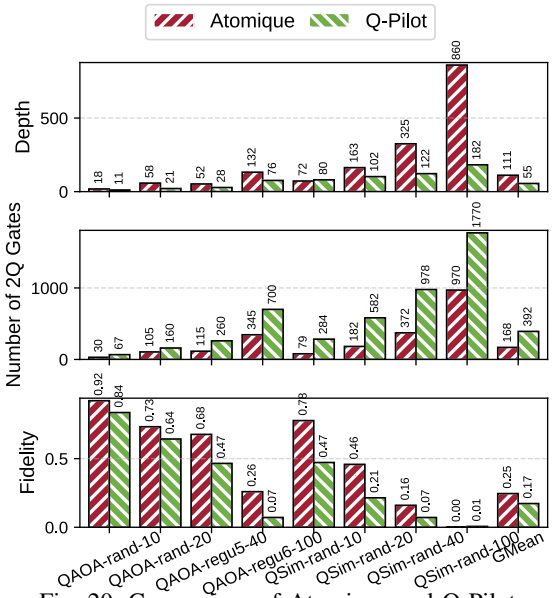


Fig. 20: Comparison of Atomique and Q-Pilot.

larger problems as the solver-based compiler times out due to exponential time and memory requirements. In Fig. 15, we can see that although the two-qubit fidelity loss is smaller in Tan-Heuristic, the fidelity loss from atom transfer is much more significant than Atomique.

We also compare Atomique with a recently-proposed compiler Q-Pilot [85], which focuses on reducing circuit depth for specific applications (QAOA and QSIM). Fig. 20 shows that although Q-Pilot has less depth than Atomique, Atomique has better overall fidelity, showing a better balance between depth, two-qubit gates, and atom movement.

Comparison of Atomique with Geyser. Geyser [64] suggests using multi-qubit gates to synthesize quantum circuit targets by minimizing the pulses used to perform multi-qubit gates. Tab. III shows the number of pulses used for multi-qubit gates for our compiler and Geyser. Since we lack experimental data to demonstrate the fidelity of multi-qubit gates, we follow Geyser's recommendation by employing the number of pulses as a proxy for the gate's fidelity. The more pulses a gate uses, the less fidelity it has. A gate involving n qubits necessitates $2n-1$ pulses. We limit the maximum function call in the dual-annealing process to 10^5 to ensure equitable comparisons. The data presented in the table clearly indicates that our compiler significantly surpasses Geyser, achieving a reduction of up to **6.5**$\times$ in the number of pulses.

C. Analysis

What Kind of Circuits are More Suitable for RAA?

(a) Generic (Arbitrary) Circuits. We analyze the efficacy of RAA on generic circuits in Fig. 16. Here, we generate random circuits with 40 qubits and varying average two-qubit gates per qubit and qubit degrees. Two-qubit gate per qubit indicates how many gates involve a certain qubit. The degree is the number of unique qubits with which a given qubit interacts. The two-qubit gate count informs the circuit depth, whereas the degree indicates the gate locality. In the figure, each surface's X and Y axes represent the two-qubit gate per qubit and qubit degree, respectively. The Z axis is the number of two-qubit gates and the relative fidelity improvement for the first and second rows. In the second row, we also project the surface into plane A (the Y-Z plane colored green) and plane B (the X-Z plane colored blue). Each line in plane A represents the fidelity improvement under a fixed two-qubit gate per qubit and varying degrees. In contrast, each line in plane B represents the fidelity improvement under a fixed degree and varying two-qubit gate per qubit. This setting is similar in Fig. 17 and Fig. 18. The results show the following insights clearly: (1) Atomique excels in **high-degree circuits**. In low-degree, well-localized circuits, Atomique's performance falls marginally behind FAA due to higher decoherence incurred by

TABLE III: Number of multi-qubit pulses (lower the better)

Benchmark	HHL-7	Mermin-Bell-10	QV-32	BV-50	BV-70
Geyser [64]	486	564	11803	432	655
Atomique	**348**	**306**	**4995**	**66**	**108**

movement. (2) **Deeper circuits benefit more from Atomique**, evidenced by widening fidelity gaps as the two-qubit gate count per qubit increases.

(b) QAOA and Quantum Simulation Circuits. In Fig. 17 and 18, we present results for QAOA and QSim circuits with varying properties. For QAOA, we generate regular graphs with different degrees. For QSim, we change the probability for each Hamiltonian term being a non-I operator. The graph degree in QAOA and probability of non-I terms resemble the qubit degree discussed in the previous paragraph, as they all indicate the locality of the problem. The previous insights are still valid here. The more non-local the program is, the higher the advantage Atomique is. In addition, Atomique is better for circuits with more qubits and depth.

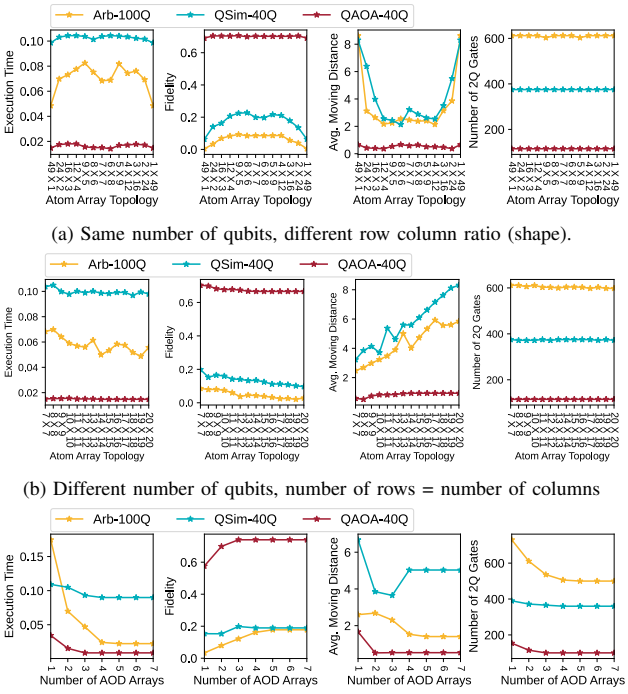
Sensitivity Analysis on Hardware Parameters. In Fig. 19, we conduct a sensitivity analysis on six key hardware parameters. The top row displays the circuit fidelities for three benchmarks—BV-70, QSim-rand-20, and QAOA-regu5-40 under different parameter settings. Comparative data for Atomique, FAA-Rectangular, and FAA-Triangular are also provided. The second row elucidates the error breakdown for BV-70 by calculating $-\log(\text{fidelity})$. Various error sources are color-coded, including one-qubit gate error, two-qubit gate error, movement heating error, cooling overhead error, atom loss during movement, and movement decoherence error.

(a) Time per Move. We vary the time per movement from 100 to 1000µs. According to Eq. 2, shorter times increase the vibrational quantum number (n_{vib}), causing significant atom loss. In the extreme case, where T_per_move smaller than 150µ, the heating of one single step can exceed N_{max}, making the cooling procedure no use. When T_per_move is smaller than 200µ, the cooling can help reduce movement heating error but inevitably incurs large cooling overhead. As T_per_move increases, the decoherence error dominates. The optimal setting is around 300µs, where the decoherence and heating are comparable, *within realistic 200-400µs range* [10].

(b) Average Move Speed. We also present the previous results in a different way by showing the atom movement speed, which is inversely related to T_per_move. Faster speeds induce atom loss, while slower speeds elevate decoherence error. QAOA and QSim benchmarks exhibit faster optimal speeds than arbitrary circuits, attributable to their smaller size.

(c) Atom Distance. We vary the SLM atom separation from 1µm to 60µm. Larger distances proportionally elevate n_{vib} with D^2 rate, increasing heating error. Cooling operations stabilize this error but introduce overhead, dominating at 60µm. Atomique outperforms baselines for distances under 40µm. In experiments, we adopt the 15µm setting from [10], where the heating error is minimal.

(d) n_{vib} for Cooling. We employ cooling to mitigate the heating and atom loss caused by excessive n_{vib}. Frequent cooling introduces two-qubit gate errors. Using a 60µs atom distance to examine trade-offs, we find a low threshold causes significant cooling errors, while a high threshold results in severe atom loss. *An optimal threshold range of 12 to 25 minimizes overall impact*; we use 15 in our experiments.



(a) Same number of qubits, different row column ratio (shape).

(b) Different number of qubits, number of rows = number of columns

(c) Different number of AOD arrays

Fig. 21: Sensitivity analysis of array topology.

(e) Coherence Time. Coherence time impacts more on RAA than to FAA since RAA execution time is much longer due to movements. With movement time (300µs) significantly exceeding one-qubit and two-qubit gate times (625ns, 380ns), RAA benefits more from extended coherence time. Comparative analysis shows RAA outperforms when coherence time exceeds 1s, a threshold generally met in practice as per [10] and [55].

(f) two-qubit Gate Fidelity. For fidelities below 0.9999, two-qubit gate errors dominate, as indicated in (f). Remarkably, FAA and Geyser outperform RAA when two-qubit fidelity exceeds 0.9999, owing to minimal SWAP overheads. Nonetheless, under current achievable two-qubit fidelity and coherence time [10], [24] Atomique remains more reliable.

Impact of the Array Topology. We also examine the execution time, fidelity, average moving distance (mm), and two-qubit gate count under different array configurations. Benchmarks are 100Q arbitrary circuit with ten gates per qubit; 40Q simulation with $p = 0.5$ probability being non-I and ten strings; 40Q QAOA with 5-regular graph.

(a) Different Row-Col Ratio under Same Qubit Number Fig. 21(a) examines the impact of array shape while maintaining the same overall atom count. For all three tasks, square arrays maximize fidelity due to shorter movement distances, as

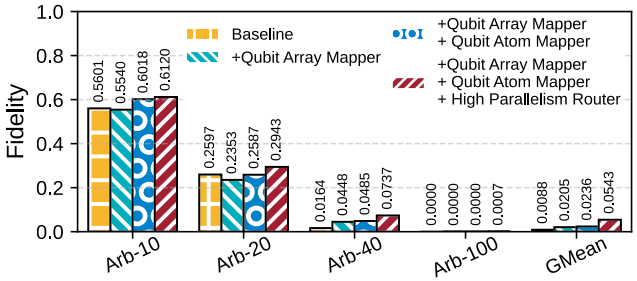


Fig. 22: Breakdown of improvement of compiler techniques.

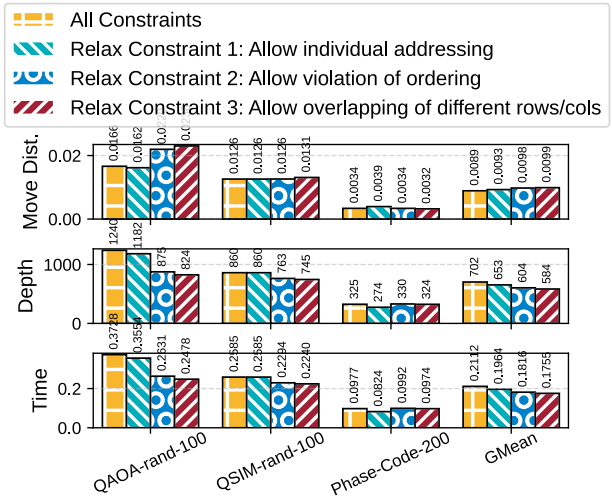


Fig. 23: Performance of Atomique after relaxing one of the three constraints.

the distance plot shows. However, the execution time increases slightly for square-like topology because they impose stronger constraints for the movement and reduce the parallelism.

(b) Different Qubit Number, Row=Col. Fig. 21(b) analyzes the effects of varying array sizes from 7×7 to 20×20. The optimal fidelity is achieved for all three tasks with a 7×7 array. As the array size expands, Atomique preferentially maps qubits to diagonal atoms, increasing parallelism and reducing execution time. However, this diagonal mapping lengthens moving distances without significantly altering the two-qubit gate count, leading to decreased fidelity due to *heating*.

(c) Different Number of AOD Arrays. Fig. 21(c) examines the impact of the number of AODs, ranging from 1 to 7. Additional AODs enhance the coupling map, and the relaxation of order-preserving constraints allows for more efficient movements—these two reasons combined reduce the two-qubit gate count and execution time and increase the fidelity.

Breakdown of Technique-Induced Improvements. Fig. 22 delineates the contributions of individual techniques to overall performance. We evaluated random circuits with 26 gates per qubit. For the baseline, we adopt the qiskit's dense mapping instead of the greedy mapping that maximizes the inter-array cut; we also use a random qubit-atom mapper rather than the load balance mapper; finally, we use a serial router that performs one gate at a time instead of parallelizing the gate. For the other experiments, we add these techniques cumulatively.

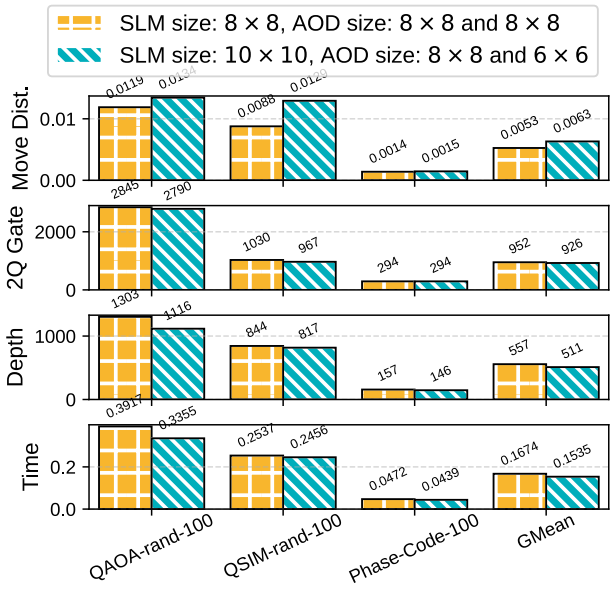


Fig. 24: Performance with different sizes across AOD layers.

On average, our qubit-array mapper gives rise to 3.53× better fidelity than the dense mapper and subsequent diagonal-first qubit-atom mapper gives rise to 1.19× higher fidelity. Finally, the high parallelism router brings 2.59× fidelity improvement. Combining all three methods, we have **10.9**× higher fidelity.

Relaxation of Hardware Constraints. Our compiler also enables toggling the constraints associated with moving AODs. Fig. 23 presents the outcomes of loosening the three constraints detailed in Fig. 9, 10, and 11. The count of two-qubit gates remains unchanged, as these constraints only influence the scheduling of the two-qubit gates. The depth is reduced, attributed to the increased flexibility in gate scheduling, which concurrently lowers the execution time. Conversely, the average movement distance sees an uptick due to the enhanced movement flexibility. The results indicate that among the constraints, easing the third yields the most significant improvement in performance.

Variability in AOD sizes. Our compiler accommodates AODs featuring distinct sizes across layers. In Fig. 24, we showcase outcomes for two configurations: one with uniform AOD and SLM dimensions, and another with varying dimensions. The disparity in SLM and AOD dimensions facilitates a reduction in the number of two-qubit gates as well as a decrease in both depth and execution time, as the varied sizes provide the compiler with enhanced flexibility in qubit mapping. Nonetheless, an increase in the moving distance is an inevitable consequence of the enlarged SLM dimensions.

Overlapping under Extreme Conditions. We analyze the performance when the number of logical qubits required by the circuit closely approaches the number of qubits available in the hardware. In Fig. 25, we compile circuits with 100 qubits on hardware where each AOD or SLM ranges in size from 6×6 (108 physical qubits in total) to 10×10 (300

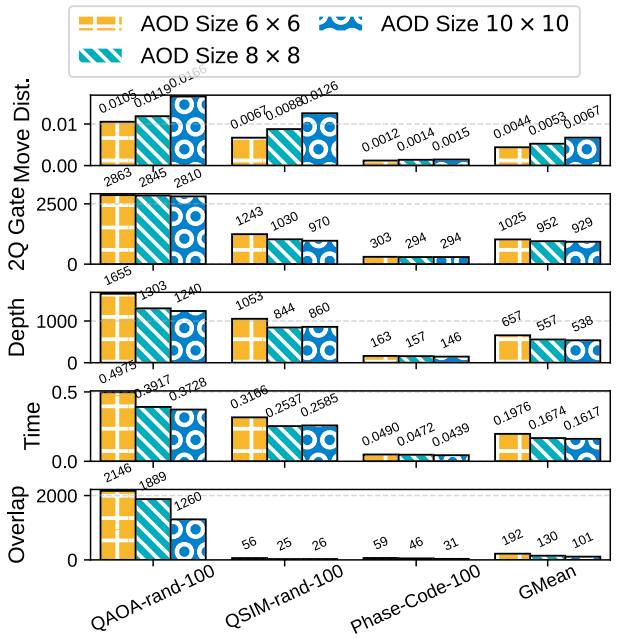


Fig. 25: Overlap and other quantities when the number of logical qubits is close to that of physical qubits.

physical qubits in total). Aside from an expected decrease in two-qubit gate depth and execution time, as well as an increase in moving distance, we also record the number of overlaps during compilation. This is defined as the case when the compiler fails to insert more two-qubit gates into the same layer due to a violation of Constraint 3. From the figure, it is evident that increasing the AOD size can aid in reducing overlaps, thereby optimizing gate scheduling. Furthermore, the overlapping behavior is highly application-dependent.

Additional `CNOT` Due to SWAP Insertion. We record the number of `CNOT` gates added due to SWAP insertion during compilation, which is the difference in the count of `CNOT`s before and after this step. From Fig. 26, it is evident that the additional `CNOT`s when using Atomique are consistently lower than those of all baselines. This performance is particularly notable in benchmarks such as BV and QSim.

VI. Related Works

Compilers for Emerging Quantum Architectures In addition to widely used superconducting quantum computers, researchers have recently shown increasing interest in new quantum computing hardwares such as neutral-atom machines and trapped-ion machines. In this work, we have discussed the previous work design for neutral-atom machines [5], [64], [75]. In addition, a number of compilers designed for trapped ion systems have been proposed in recent years, including TILT [87] for a linear chain of trapped ions, Ref. [60] for quantum charge coupled device-based trapped ion architectures, and Ref. [44] for shuttling-based trapped ion architectures. Many compilation techniques for various settings have also been proposed [67], [79], [18], [65], [59], [80], [48], [58],

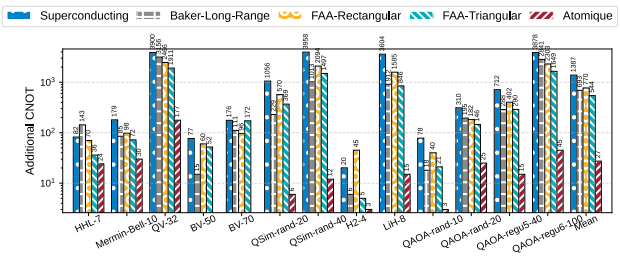


Fig. 26: Comparisons of additional `CNOT` caused by SWAP-insertion among different architectures.

[53], [91], [57], [54], [12], [89], [29], [83], [8], [51], [17], [84], [90], [62], [87], [82], [30], [20], [50], [45], [14]. In this work, we focus on the emerging RAA devices implemented with dynamically reconfigurable atom arrays, which require new techniques to fully exploit their power. Therefore, we design a compilation framework that delivers low-depth compiled circuits with high scalability.

Qubit Mapping and Instruction Scheduling Since noise is the bottleneck of NISQ machines, many noise-adaptive quantum compilation techniques have been proposed [67], [79], [18], [65], [32]. Examples include various gate errors which can be suppressed by qubit mapping [59], [80], [48], [58], [53], [91], composite pulses [57], [54], [12], [89], dynamical decoupling [29], [83], [8], [51], [17], randomized compiling [84], hidden inverses [90], instruction scheduling [62], [87], frequency tuning [82], [30], [20], parallel execution on multiple machines [74], algorithm-aware compilation [40], [41], [39], [50], [45], [14], qubit-specific basis gate [52], [49], and vairous pulse level optimizations [13]. Qubit mapping and routing, also named quantum layout synthesis or qubit allocation/placement, has been a popular research topic in the quantum computing community [48], [86], [61], [91], [77], [76], [56], [71], [93], [92], [58], [26]. Instruction scheduling has also been widely explored [62], [87], [72], [74], [52], [49]. The most relevant previous work is Brandhofer *et al.* [11] which considers a hypothetical atom array architecture with "1D displacement" reconfigurability, a much more restrictive architecture than the RAA we discussed in this paper.

VII. Conclusion and Outlook

We architect the emerging *reconfigurable atom arrays* (RAA) with moveable atoms at runtime. In this highly flexible RAA, we propose a qubit mapper and a router to reduce the SWAP overhead and maintain high parallelism of two-qubit gate executions. Compared to other architectures or compilers, our Atomique significantly reduces the depth and two-qubit gates. We hope Atomique opens up an avenue for future RAA.

VIII. Acknowledgement

This work is partially supported by NSF grant 442511-CJ-22291, CCF-1901381, CCF-2115104, CCF-2119352, and CCF-2107241, MIT-IBM Watson AI Lab, and Qualcomm Innovation Fellowship. The authors would like to thank Dolev Bluvstein, Mikhail D. Lukin, and Hengyun Zhou for valuable discussions on neutral atom arrays.

REFERENCES

[1] M. Alam, A. Ash-Saki, and S. Ghosh, "Circuit compilation methodologies for quantum approximate optimization algorithm," in *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, October 17-21, 2020*. IEEE, 2020, pp. 215–228. [Online]. Available: https://doi.org/10.1109/MICRO50266.2020.00029

[2] M. Alam, A. Ash-Saki, and S. Ghosh, "Circuit compilation methodologies for quantum approximate optimization algorithm," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 215–228.

[3] ——, "An efficient circuit compilation flow for quantum approximate optimization algorithm," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[4] M. Alam, A. Ash-Saki, J. Li, A. Chattopadhyay, and S. Ghosh, "Noise resilient compilation policies for quantum approximate optimization algorithm," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–7.

[5] J. M. Baker, A. Litteken, C. Duckering, H. Hoffmann, H. Bernien, and F. T. Chong, "Exploiting long-distance interactions and tolerating atom loss in neutral atom quantum architectures," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 818–831.

[6] K. Barnes, P. Battaglino, B. J. Bloom, K. Cassella, R. Coxe, N. Crisosto, J. P. King, S. S. Kondov, K. Kotru, S. C. Larsen, and et al., "Assembly and coherent control of a register of nuclear spin qubits," *Nature Communications*, vol. 13, no. 1, May 2022.

[7] J. Beugnon, C. Tuchendler, H. Marion, A. Gaëtan, Y. Miroshnychenko, Y. R. Sortais, A. M. Lance, M. P. Jones, G. Messin, A. Browaeys *et al.*, "Two-dimensional transport and transfer of a single atomic qubit in optical tweezers," *Nature Physics*, vol. 3, no. 10, pp. 696–699, 2007.

[8] M. J. Biercuk, H. Uys, A. P. VanDevender, N. Shiga, W. M. Itano, and J. J. Bollinger, "Optimized dynamical decoupling in a model quantum memory," *Nature*, vol. 458, no. 7241, pp. 996–1000, 2009.

[9] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter *et al.*, "Logical quantum processor based on reconfigurable atom arrays," *Nature*, pp. 1–3, 2023.

[10] D. Bluvstein, H. Levine, G. Semeghini, T. T. Wang, S. Ebadi, M. Kalinowski, A. Keesling, N. Maskara, H. Pichler, M. Greiner *et al.*, "A quantum processor based on coherent transport of entangled atom arrays," *Nature*, vol. 604, no. 7906, pp. 451–456, 2022.

[11] S. Brandhofer, H. P. Büchler, and I. Polian, "Optimal mapping for near-term quantum architectures based on Rydberg atoms," in *Proceedings of the 40th IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '21. Munich, Germany: Association for Computing Machinery, Nov. 2021.

[12] K. R. Brown, A. W. Harrow, and I. L. Chuang, "Arbitrarily accurate composite pulse sequences," *Physical Review A*, vol. 70, no. 5, p. 052318, 2004.

[13] Y. Chen, Y. Jin, F. Hua, A. Hayes, A. Li, Y. Shi, and E. Z. Zhang, "A pulse generation framework with augmented program-aware basis gates and criticality analysis," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 773–786.

[14] J. Cheng, H. Wang, Z. Liang, Y. Shi, S. Han, and X. Qian, "Topgen: Topology-aware bottom-up generator for variational quantum circuits," *arXiv preprint arXiv:2210.08190*, 2022.

[15] A. Coja-Oghlan, C. Moore, and V. Sanwalani, "Max k-cut and approximating the chromatic number of random graphs," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2003, pp. 200–211.

[16] J. P. Covey, I. S. Madjarov, A. Cooper, and M. Endres, "2000-times repeated imaging of strontium atoms in clock-magic tweezer arrays," *Physical review letters*, vol. 122, no. 17, p. 173201, 2019.

[17] P. Das, S. Tannu, S. Dangwal, and M. Qureshi, "Adapt: Mitigating idling errors in qubits via adaptive dynamical decoupling," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 950–962.

[18] P. Das, S. Tannu, and M. Qureshi, "Jigsaw: Boosting fidelity of nisq programs via measurement subsetting," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 937–949.

[19] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and J. Rehof, Eds. Berlin, Heidelberg: Springer, 2008, pp. 337–340.

[20] Y. Ding, P. Gokhale, S. F. Lin, R. Rines, T. Propson, and F. T. Chong, "Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 201–214.

[21] C. Duckering, J. M. Baker, A. Litteken, and F. T. Chong, "Orchestrated trios: compiling for efficient communication in quantum programs with 3-qubit gates," in *ASPLOS '21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual Event, USA, April 19-23, 2021*, T. Sherwood, E. D. Berger, and C. Kozyrakis, Eds. ACM, 2021, pp. 375–385. [Online]. Available: https://doi.org/10.1145/3445814.3446718

[22] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler, S.-T. Wang, M. Greiner, V. Vuletić, and M. D. Lukin, "Quantum optimization of maximum independent set using rydberg atom arrays," *Science*, vol. 376, no. 6598, pp. 1209–1215, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.abo6587

[23] S. Ebadi, T. T. Wang, H. Levine, A. Keesling, G. Semeghini, A. Omran, D. Bluvstein, R. Samajdar, H. Pichler, W. W. Ho *et al.*, "Quantum phases of matter on a 256-atom programmable quantum simulator," *Nature*, vol. 595, no. 7866, pp. 227–232, 2021.

[24] S. J. Evered, D. Bluvstein, M. Kalinowski, S. Ebadi, T. Manovitz, H. Zhou, S. H. Li, A. A. Geim, T. T. Wang, N. Maskara *et al.*, "High-fidelity parallel entangling gates on a neutral atom quantum computer," *arXiv preprint arXiv:2304.05420*, 2023.

[25] S. J. Evered, D. Bluvstein, M. Kalinowski, S. Ebadi, T. Manovitz, H. Zhou, S. H. Li, A. A. Geim, T. T. Wang, N. Maskara, H. Levine, G. Semeghini, M. Greiner, V. Vuletic, and M. D. Lukin, "High-fidelity parallel entangling gates on a neutral atom quantum computer," 2023.

[26] H. Fan, C. Guo, and W. Luk, "Optimizing quantum circuit placement via machine learning," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 19–24. [Online]. Available: https://doi.org/10.1145/3489517.3530403

[27] P. Gokhale, A. Javadi-Abhari, N. Earnest, Y. Shi, and F. T. Chong, "Optimized quantum compilation for near-term algorithms with open-pulse," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 186–200.

[28] T. M. Graham, Y. Song, J. Scott, C. Poole, L. Phuttitarn, K. Jooya, P. Eichler, X. Jiang, A. Marra, B. Grinkemeyer, M. Kwon, M. Ebert, J. Cherek, M. T. Lichtman, M. Gillette, J. Gilbert, D. Bowman, T. Ballance, C. Campbell, E. D. Dahl, O. Crawford, N. S. Blunt, B. Rogers, T. Noel, and M. Saffman, "Multi-qubit entanglement and algorithms on a neutral-atom quantum computer," *Nature*, vol. 604, no. 7906, pp. 457–462, Apr. 2022. [Online]. Available: https://www.nature.com/articles/s41586-022-04603-6

[29] E. L. Hahn, "Spin echoes," *Physical review*, vol. 80, no. 4, p. 580, 1950.

[30] F. Helmer, M. Mariantoni, A. G. Fowler, J. von Delft, E. Solano, and F. Marquardt, "Cavity grid for scalable quantum computation with superconducting circuits," *EPL (Europhysics Letters)*, vol. 85, no. 5, p. 50007, 2009.

[31] J. Hsu, "Ces 2018: Intel's 49-qubit chip shoots for quantum supremacy," https://spectrum.ieee.org/tech-talk/computing/hardware/intels-49qubit-chip-aims-for-quantum-supremacy.

[32] F. Hua, Y. Jin, A. Li, C. Liu, M. Wang, Y. Chen, C. Zhang, A. Hayes, S. Stein, M. Guo, Y. Huang, and E. Z. Zhang, "A synergistic compilation workflow for tackling crosstalk in quantum machines," 2023.

[33] F. Hua, M. Wang, G. Li, B. Peng, C. Liu, M. Zheng, S. A. Stein, Y. Ding, E. Z. Zhang, T. S. Humble, and A. Li, "Qasmtrans: A QASM quantum transpiler framework for NISQ devices," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W 2023, Denver, CO, USA, November 12-17, 2023*. ACM, 2023, pp. 1468–1477. [Online]. Available: https://doi.org/10.1145/3624062.3624222

[34] IBM, "IBM quantum," https://quantum-computing.ibm.com.

[35] ——, "IBM unveils 400 qubit-plus quantum processor and next-generation ibm quantum system two," https://newsroom.ibm.com/2022-

11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two.

[36] ——, "IBM unveils breakthrough 127-qubit quantum processor," https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor.

[37] Infleqtion, https://www.infleqtion.com/news/infleqtion-unveils-5-year-quantum-computing-roadmap-advancing-plans-to-commercialize-quantum-at-scale, Feb 2024.

[38] D. Jaksch, J. I. Cirac, P. Zoller, S. L. Rolston, R. Côté, and M. D. Lukin, "Fast quantum gates for neutral atoms," *Phys. Rev. Lett.*, vol. 85, pp. 2208–2211, Sep 2000. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.85.2208

[39] Y. Jin, X. Gao, M. Guo, H. Chen, F. Hua, C. Zhang, and E. Z. Zhang, "Quantum fourier transformation circuits compilation," 2023.

[40] Y. Jin, F. Hua, Y. Chen, A. Hayes, C. Zhang, and E. Z. Zhang, "Exploiting the regular structure of modern quantum architectures for compiling and optimizing programs with permutable operators," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4*, ser. ASPLOS '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 108–124. [Online]. Available: https://doi.org/10.1145/3623278.3624751

[41] Y. Jin, Z. Li, F. Hua, Y. Chen, H. Chen, Y. Huang, and E. Z. Zhang, "Tetris: A compilation framework for vqe applications," 2023.

[42] J. Kelly, "A preview of bristlecone, google's new quantum processor," https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html.

[43] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Applied Physics Reviews*, vol. 6, no. 2, p. 021318, 2019.

[44] F. Kreppel, C. Melzer, J. Wagner, J. Hilder, U. Poschinger, F. Schmidt-Kaler, and A. Brinkmann, "Quantum Circuit Compiler for a Shuttling-Based Trapped-Ion Quantum Computer," 7 2022.

[45] L. Lao and D. E. Browne, "2qan: A quantum compiler for 2-local qubit hamiltonian simulation algorithms," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 351–365.

[46] H. Levine, A. Keesling, G. Semeghini, A. Omran, T. T. Wang, S. Ebadi, H. Bernien, M. Greiner, V. Vuletić, H. Pichler, and M. D. Lukin, "Parallel implementation of high-fidelity multiqubit gates with neutral atoms," *Phys. Rev. Lett.*, vol. 123, p. 170503, Oct 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.123.170503

[47] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, feb 2023. [Online]. Available: https://doi.org/10.1145/3550488

[48] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for nisq-era quantum devices," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 1001–1014.

[49] G. Li, Y. Shi, and A. Javadi-Abhari, "Software-hardware co-optimization for computational chemistry on superconducting quantum processors," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 832–845.

[50] G. Li, A. Wu, Y. Shi, A. Javadi-Abhari, Y. Ding, and Y. Xie, "Paulihedral: a generalized block-wise compiler optimization framework for quantum simulation kernels," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 554–569.

[51] D. A. Lidar, "Review of decoherence free subspaces, noiseless subsystems, and dynamical decoupling," *Adv. Chem. Phys*, vol. 154, pp. 295–354, 2014.

[52] S. F. Lin, S. Sussman, C. Duckering, P. S. Mundada, J. M. Baker, R. S. Kumar, A. A. Houck, and F. T. Chong, "Let each quantum bit choose its basis gates," *arXiv preprint arXiv:2208.13380*, 2022.

[53] L. Liu and X. Dou, "Qucloud: A new qubit mapping mechanism for multi-programming quantum computing in cloud environment," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 167–178.

[54] G. H. Low, T. J. Yoder, and I. L. Chuang, "Optimal arbitrarily accurate composite pulse sequences," *Physical Review A*, vol. 89, no. 2, p. 022341, 2014.

[55] I. S. Madjarov, J. P. Covey, A. L. Shaw, J. Choi, A. Kale, A. Cooper, H. Pichler, V. Schkolnik, J. R. Williams, and M. Endres, "High-fidelity entanglement and detection of alkaline-earth rydberg atoms," *Nature Physics*, vol. 16, no. 8, pp. 857–861, 2020.

[56] D. Maslov, S. M. Falconer, and M. Mosca, "Quantum circuit placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 752–763, Apr. 2008.

[57] J. T. Merrill and K. R. Brown, "Progress in compensating pulse sequences for quantum computation," *Quantum Information and Computation for Chemistry*, pp. 241–294, 2014.

[58] A. Molavi, A. Xu, M. Diges, L. Pick, S. Tannu, and A. Albarghouthi, "Qubit mapping and routing via maxsat," *arXiv preprint arXiv:2208.13679*, 2022.

[59] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 1015–1029.

[60] 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/10.1109/ISCA45697.2020.00051

[61] P. Murali, N. M. Linke, M. Martonosi, A. J. Abhari, N. H. Nguyen, and C. H. Alderete, "Full-stack, real-system quantum computer studies: Architectural comparisons and design insights," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA '19. Phoenix, Arizona: ACM Press, 2019, pp. 527–540.

[62] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 1001–1016.

[63] M. A. Norcia, H. Kim, W. B. Cairncross, M. Stone, A. Ryou, M. Jaffe, M. O. Brown, K. Barnes, P. Battaglino, T. C. Bohdanowicz, A. Brown, K. Cassella, C. A. Chen, R. Coxe, D. Crow, J. Epstein, C. Griger, E. Halperin, F. Hummel, A. M. W. Jones, J. M. Kindem, J. King, K. Kotru, J. Lauigan, M. Li, M. Lu, E. Megidish, J. Marjanovic, M. McDonald, T. Mittiga, J. A. Muniz, S. Narayanaswami, C. Nishiguchi, T. Paule, K. A. Pawlak, L. S. Peng, K. L. Pudenz, D. R. Perez, A. Smull, D. Stack, M. Urbanek, R. J. M. van de Veerdonk, Z. Vendeiro, L. Wadleigh, T. Wilkason, T. Y. Wu, X. Xie, E. Zalys-Geller, X. Zhang, and B. J. Bloom, "Iterative assembly of ^{171}yb atom arrays in cavity-enhanced optical lattices," 2024.

[64] T. Patel, D. Silver, and D. Tiwari, "Geyser: A compilation framework for quantum computing with neutral atoms," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 383–395. [Online]. Available: https://doi.org/10.1145/3470496.3527428

[65] T. Patel and D. Tiwari, "Qraft: Reverse your quantum circuit and know the correct program output," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 443–455. [Online]. Available: https://doi.org/10.1145/3445814.3446743

[66] QuEra, "Quera launches their 256 qubit analog quantum processor on aws," https://quantumcomputingreport.com/quera-launches-their-256-qubit-analog-quantum-processor-on-aws/.

[67] G. S. Ravi, K. N. Smith, P. Gokhale, A. Mari, N. Earnest, A. Javadi-Abhari, and F. T. Chong, "Vaqem: A variational approach to quantum error mitigation," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 288–303.

[68] Rigetti, "Rigetti quantum," https://www.rigetti.com/.

[69] M. Saffman, "Quantum computing with atomic qubits and rydberg interactions: progress and challenges," *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 49, no. 20, p. 202001, oct 2016. [Online]. Available: https://dx.doi.org/10.1088/0953-4075/49/20/202001

[70] K.-N. Schymik, B. Ximenez, E. Bloch, D. Dreon, A. Signoles, F. Nogrette, D. Barredo, A. Browaeys, and T. Lahaye, "In situ equalization of single-atom loading in large-scale optical tweezer arrays," *Phys. Rev. A*, vol. 106, p. 022611, Aug 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.106.022611

[71] M. Y. Siraichi, V. F. dos Santos, S. Collange, and F. M. Q. Pereira, "Qubit allocation," in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, ser. CGO '18. Vienna, Austria: ACM Press, 2018, pp. 113–125.

[72] K. N. Smith, G. S. Ravi, P. Murali, J. M. Baker, N. Earnest, A. Javadi-Abhari, and F. T. Chong, "Error mitigation in quantum computers through instruction scheduling," 2021. [Online]. Available: https://arxiv.org/abs/2105.01760

[73] K. N. Smith and M. A. Thornton, "A quantum computational compiler and design tool for technology-specific targets," in *Proceedings of the 46th International Symposium on Computer Architecture, ISCA 2019, Phoenix, AZ, USA, June 22-26, 2019*, S. B. Manne, H. C. Hunter, and E. R. Altman, Eds. ACM, 2019, pp. 579–588. [Online]. Available: https://doi.org/10.1145/3307650.3322262

[74] S. Stein, N. Wiebe, Y. Ding, P. Bo, K. Kowalski, N. Baker, J. Ang, and A. Li, "Eqc: ensembled quantum computing for variational quantum algorithms," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 59–71.

[75] B. Tan, D. Bluvstein, D. M. Lukin, and J. Cong, "Qubit mapping for reconfigurable atom arrays," *ICCAD*, 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3508352.3549331

[76] B. Tan and J. Cong, "Optimal layout synthesis for quantum computing," in *Proceedings of the 39th IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '20. Virtual Event, USA: Association for Computing Machinery, Jul. 2020.

[77] ——, "Optimal qubit mapping with simultaneous gate absorption," in *Proceedings of the 40th IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '21. Munich, Germany: Association for Computing Machinery, Nov. 2021.

[78] D. B. Tan, D. Bluvstein, D. M. Lukin, and J. Cong, "Compiling quantum circuits for dynamically field-programmable neutral atoms array processors," *Quantum*, vol. 8, 2024. [Online]. Available: https://doi.org/10.22331/q-2024-03-14-1281

[79] S. Tannu, P. Das, R. Ayanzadeh, and M. Qureshi, "Hammer: Boosting fidelity of noisy quantum circuits by exploiting hamming behavior of erroneous outcomes," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 529–540. [Online]. Available: https://doi.org/10.1145/3503222.3507703

[80] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 987–999.

[81] 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)*. IEEE, 2022, pp. 587–603.

[82] R. Versluis, S. Poletto, N. Khammassi, B. Tarasinski, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo, "Scalable quantum circuit and control for a superconducting surface code," *Physical Review Applied*, vol. 8, no. 3, p. 034021, 2017.

[83] L. Viola, E. Knill, and S. Lloyd, "Dynamical decoupling of open quantum systems," *Physical Review Letters*, vol. 82, no. 12, p. 2417, 1999.

[84] 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.

[85] H. Wang, B. Tan, P. Liu, Y. Liu, J. Gu, J. Cong, and S. Han, "Q-pilot: Field programmable qubit array compilation with flying ancillas," in *2024 Design Automation Conference (DAC)*, 2024.

[86] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. Las Vegas, NV, USA: ACM Press, 2019.

[87] X.-C. Wu, D. M. Debroy, Y. Ding, J. M. Baker, Y. Alexeev, K. R. Brown, and F. T. Chong, "Tilt: Achieving higher fidelity on a trapped-ion linear-tape quantum computing architecture," *arXiv preprint arXiv:2010.15876*, 2020.

[88] J. Wurtz, A. Bylinskii, B. Braverman, J. Amato-Grill, S. H. Cantu, F. Huber, A. Lukin, F. Liu, P. Weinberg, J. Long *et al.*, "Aquila: Quera's 256-qubit neutral-atom quantum computer," *arXiv preprint arXiv:2306.11727*, 2023.

[89] L. Xie, J. Zhai, Z. Zhang, J. Allcock, S. Zhang, and Y.-C. Zheng, "Suppressing zz crosstalk of quantum computers through pulse and scheduling co-optimization," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 499–513.

[90] B. Zhang, S. Majumder, P. H. Leung, S. Crain, Y. Wang, C. Fang, D. M. Debroy, J. Kim, and K. R. Brown, "Hidden inverses: Coherent error cancellation at the circuit level," *arXiv preprint arXiv:2104.01119*, 2021.

[91] C. Zhang, A. B. Hayes, L. Qiu, Y. Jin, Y. Chen, and E. Z. Zhang, "Time-optimal qubit mapping," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 360–374.

[92] X. Zhou, Y. Feng, and S. Li, "A Monte Carlo Tree Search Framework for Quantum Circuit Transformation," *arXiv:2008.09331 [quant-ph]*, Aug. 2020, arXiv: 2008.09331.

[93] A. Zulehner, A. Paler, and R. Wille, "Efficient mapping of quantum circuits to the IBM QX architectures," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Dresden, Germany: IEEE, Mar. 2018, pp. 1135–1138.

APPENDIX

A. Abstract

Our paper introduces a new compiler, Atomique, which is tailored for qubit mapping, atom movement, and gate scheduling of the RAAs.

The artifact contains the Atomique compiler, which is this paper's main contribution, baseline compilers used for comparison, benchmarks, and necessary scripts to run experiments and draw figures.

B. Artifact check-list (meta-information)

- **Algorithm:** The compiler is mainly built from three components, which are our original algorithms:
  - Qubit Array Mapper.
  - Qubit Atom Mapper.
  - High Parallelism Router.
- **Program:** We use a variety of benchmarks to test our compiler, including:
  - Algorithm circuit from QASMBench.
  - Algorithm circuit from SupermarQ.
  - QAOA circuit generated by us.
  - QSim circuit generated by us.

  All are public. We have already included them in the artifact for ease of use.
- **Compilation:** Python 3.9 with Qiskit 0.38.0. All are public-available.
- **Run-time environment:** Python with Qiskit and Pytorch. We tested it only on Linux; however, it should be able to run on any platform that supports Python, Qiskit, and Pytorch.
- **Hardware:** There is no specific hardware requirement. To complete the experiments in a reasonable amount of time, a CPU with 8 cores and 16 GB of memory is recommended.
- **Run-time state:**
  - Not OS-specific.
  - No need for root access.
  - Not run-time sensitive.
- **Execution:**
  - No specific condition is required.
  - The data for each figure can be generated in less than 10 minutes, except Tab.3. The baseline Geyser takes 10 hours to a day to complete.
- **Metrics:**
  - two-qubits gate count.
  - one-qubit gate count.
  - Fidelity.
  - Circuit depth.
  - Compilation time.

  In general, the generated figures should match those presented in the paper.
- **Output:** Figures 12-26 of the article and the necessary data for Tables 2 and 3.
- **Experiments:** Please see Readme.md in the artifact for more detailed instructions on running the experiments.
- **How much disk space required (approximately)?:** <10 GB
- **How much time is needed to prepare workflow (approximately)?:** <10 minutes
- **How much time is needed to complete experiments (approximately)?:** <2 hours if excluding the Geyser baseline.
- **Publicly available?:** Our compiler is available at https://doi.org/10.5281/zenodo.10895509.
- **Code licenses (if publicly available)?:** MIT license.
- **Workflow framework used?:** Qiskit and PyTorch.
- **Archived (provide DOI)?:** https://doi.org/10.5281/zenodo.10895509

C. Description

1) How to access

The artifact is available at the following link: https://doi.org/10.5281/zenodo.10895509

2) Hardware dependencies

To complete the experiments in a reasonable amount of time, a CPU with 8 cores and 16 GB of memory is recommended.

3) Software dependencies

The artifact is implemented in Python and requires several packages, such as PyTorch and Qiskit. The complete list of required packages is in requirements.txt in the artifact folder.

4) Data sets

All the benchmarks are provided in the artifact.

D. Installation

1) **Install Required Packages**: Assuming you are using Anaconda to manage the Python environment, install all the necessary Python packages listed in the `requirements.txt` file using the following command:

```
conda create -n atomique
conda activate atomique
conda install python==3.9
pip install -r requirements.txt
```

2) **Set Up Environment Variable**: Export the Python path to the base directory to ensure the scripts run correctly.

```
export PYTHONPATH=.
```

E. Experiment workflow

To run all the experiments, run

```
./all_script
```

F. Evaluation and expected results

To evaluate the functionality of Atomique, the quantum circuit should be successfully compiled into basic operations supported by RAA, and some statistics of the compiled circuit should also be output.

For the evaluation of Atomique, all the figures presented in the paper should be reproduced using the artifact. The figures are stored under the plot folder after running the experiments.

G. Experiment customization

The user can write their own benchmarks and configs to customize the experiments by following the provided example benchmarks and configs.

H. Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-and-badging-current
- http://cTuning.org/ae/submission-20201122.html
- http://cTuning.org/ae/reviewing-20201122.html