EZCache: A Hierarchical Memory System for Zoned Neutral Atom Quantum Computers

Jiayi Zhong
Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
Shanghai, China
jiayialice323@gmail.com

Yuxin Deng*
MoE Key Laboratory of Interdisciplinary Research of Computation and Economics, Shanghai University of Finance and Economics and Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
Shanghai, China
yxdeng@msg.sufe.edu.cn

Hui Jiang
School of Computer Science and Technology, Chongqing University of Posts and Telecommunications
Chongqing, China
jianghui@cqupt.edu.cn

Jiacheng Feng
Swanson School of Engineering, University of Pittsburgh
Pittsburgh, PA
jif104@pitt.edu

Abstract

Long-distance atom shuttling between the storage zone (SZ) and the entangling zone (EZ) degrades the fidelity of quantum programs on large-scale neutral atom processors. Existing compilers often place entangling qubits near the zone boundary, underutilizing deeper EZ sites and repeatedly moving idle qubits across zones. With spatially selective laser excitation, only part of the EZ is illuminated for entangling gates while the rest stays unilluminated, which turns compilation into a constrained time-aware placement problem. We present **EZCache**, a hierarchical memory-system abstraction that uses the dark EZ region as a capacity-limited residency layer for idle qubits. It heuristically decides where entangling qubits execute in the illuminated EZ and which idle qubits stay resident versus return to the SZ based on near-future reuse. Across parallel entangling gates, EZCache suppresses decoherence and laser crosstalk from reducing unnecessary qubit shuttling by combining reuse-window residency with look-ahead idle parking. Simulations show that EZCache improves fidelity by 18.1% over PowerMove and 64.2% over ZAC, and reduces total movement by 52.4% over PowerMove.

CCS Concepts

• **Computer systems organization** → **Quantum computing**.

Keywords

Quantum Computing, Neutral Atom Array, Compiler

ACM Reference Format:
Jiayi Zhong, Yuxin Deng, Hui Jiang, and Jiacheng Feng. 2026. EZCache: A Hierarchical Memory System for Zoned Neutral Atom Quantum Computers. In *2026 International Conference on Supercomputing (ICS '26), July 06–09, 2026, Belfast, United Kingdom*, 12 pages. https://doi.org/10.1145/3797905.3807838

*Corresponding author.

ICS '26, Belfast, United Kingdom

ACM ISBN 979-8-4007-2522-7/2026/07
https://doi.org/10.1145/3797905.3807838

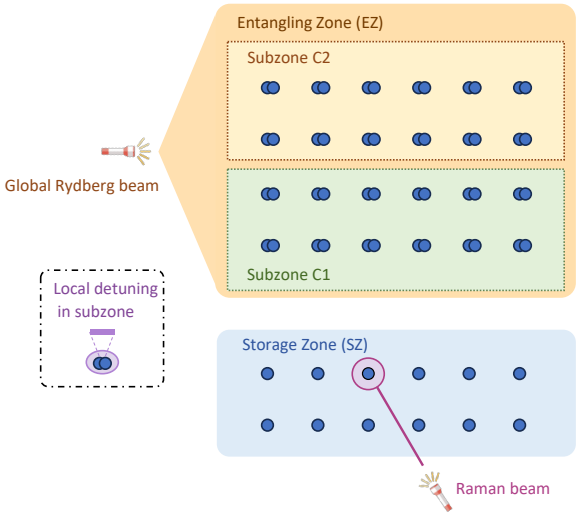

Figure 1: Illustration of a zoned neutral atom processor with spatially selective excitation. Blue dots refer to the atom sites.

1 Introduction

Large-scale quantum computing represents a transformative paradigm, with the potential to address computational challenges beyond classical systems, as exemplified by Shor's factorization and Grover's search algorithms [9, 17]. Neutral atom quantum computing (NAQC) platforms built on optical tweezer arrays are a strong candidate for scaling to this regime. Recent NAQC experiments have demonstrated programmable registers with hundreds to thousands of atoms, coherent atom transport, and Rydberg-based entangling operations with long lifetimes [3, 4, 6, 13, 22]. Many of these systems adopt a zoned layout to support reliable large-scale execution of quantum programs. As Figure 1 shows, the storage zone (SZ) keeps qubits away from the global entangling fields, so idle qubits are less exposed to errors related to entangling gates and crosstalk, while the entangling zone (EZ) concentrates the global Rydberg pulse where multi-qubit gates are performed [3].

Despite the fidelity benefits of zoning, the dominant cost at scale is still repeated atom shuttling, especially long-distance transfers between SZ and EZ across successive entangling rounds. This shuttling increases transport-induced loss and heating, and it also stretches the time between pulses, which leaves qubits idle longer and more vulnerable to decoherence and drift. Prior compilers make important progress on zoned compilation, but their behavior also exposes a structural inefficiency in how EZ space is used. NALAC formalizes routing on zoned neutral atom architectures [18]. Mantra reduces SZ–EZ shuttling by grouping entangling operations to increase locality [10]. ZAC and PowerMove further reduce back and forth movement by reusing qubits in EZ across nearby rounds [12, 14]. Even with these optimizations, entangling qubits are often placed near the SZ–EZ boundary to lower immediate movement, which leaves deeper EZ sites underused. As a result, qubits that do not participate in the current round are still frequently shuttled back to SZ, and congestion near the interface becomes a growing bottleneck as circuit size increase.

A recent hardware capability makes it possible to use EZ space more effectively during each global Rydberg pulse [13]. During the pulse, platforms can selectively suppress Rydberg excitation at chosen EZ sites by locally detuning atoms from the Rydberg transition using site-dependent energy shifts induced by off-resonant light, implemented with spatial light modulators, digital micromirror devices, or auxiliary beams [8, 11]. This creates a dark EZ region that remains unexcited during the pulse, so idle qubits can stay in EZ instead of being moved back to SZ after every round. As Figure 1 shows, if subzone C_1 is set as the illuminated one, then subzone C_2 can temporarily hold idle qubits.

However, this flexibility comes with new physical constraints that the compiler must manage in NAQC. The capacity in dark region is limited, and at pulse time only the qubits participating in the current round's entangling gates may occupy illuminated EZ execution sites. In addition, keeping idle qubits in EZ is not free over time. Even in the dark region, qubits accumulate time-dependent errors called decoherence, and nearby global Rydberg illumination can still have nonzero impact called laser crosstalk. These effects make EZ residency a time-aware resource: the compiler must decide not only how to place active qubits for the current pulse and where to park idle qubits, but also when it is better to return idle qubits to SZ rather than leaving them in EZ.

We present EZCache to address these problems. EZCache jointly decides, at each entangling stage, (i) execution site assignment for the entangling qubits in the illuminated EZ and (ii) residency and waiting locations for idle qubits under pulse time safety and dark capacity constraints. The goal is to minimize per-round qubit moving time determined by the slowest qubit transport, reduce overall transport distance as an indicator of motional heating and atom-loss risk, and cut unnecessary qubit shuttling across three zones including the dark EZ region. In this paper, an EZ subzone is a logical illumination choice on a fixed EZ tweezer array. It selects which EZ sites are illuminated in a stage, without changing the number or geometry of EZ sites.

Our contributions are as follows:

- We present a hierarchical memory-system abstraction for selective illumination on zoned NAQC, where SZ provides long-term storage and a dark EZ subregion holds idle qubits between entangling rounds; we also define time-aware reuse metadata based on next-use distance to guide qubit residency control.
- We design an EZCache compilation pipeline that runs once per entangling round. It heuristically selects which EZ subregion is illuminated and which is kept dark, coordinates qubits' admission and eviction between SZ and EZ, assigns execution sites to the current entangling gates, and places idle qubits at safe waiting sites according to a time-aware residency strategy that bounds EZ waiting with reuse windows and uses short look-ahead to choose waiting locations.
- On 1,016 benchmark instances across QAOA-rand, QAOA-regular, Bernstein–Vazirani, and Quantum Fourier Transform, EZCache improves duration of quantum circuits by 18.1% over PowerMove and 64.2% over ZAC on average, and reduces total moved distance by 52.4% over PowerMove. It scales with circuit size while consistently improving transport and fidelity across multiple quantum circuits.

The rest of this paper is organized as follows. Section 2 provides background on zoned neutral atom processors and our motivation. Section 3 formally describes the EZCache memory system and compilation objective under selective illumination. Section 4 describes the key techniques used in EZCache and presents the complete flow of the pipeline. Sections 5 and 6 evaluate EZCache and report experimental results. Section 7 compares with previous work relevant to our research. Section 8 concludes this work.

2 Background and Motivation

This section introduces the hardware model and stage based execution flow in zoned neutral atom quantum computing, and explains how spatially selective Rydberg excitation makes entangling zone management a residency control problem that spans multiple stages.

2.1 Hardware Implementation and Quantum Programs

Qubit movement. A neutral atom qubit is encoded in the internal states of an atom held in an optical tweezer. The device uses a spatial light modulator (SLM) to create a fixed array of trapping sites that hold atoms for storage and interaction [1, 7]. Atom transport uses movable tweezers steered by crossed 2D acousto optic deflectors (AOD). AOD picks up an atom from one SLM site, carries it across the array, and drops it onto another site. This supports long range transfers between SZ and EZ and short range rearrangements within a zone. Within one AOD set, the AOD rows and columns cannot cross, which limits which moves can happen at the same time. Different AOD sets can intersect, which increases parallelism [4].

Quantum operations. As shown in Figure 1, one-qubit gates are individually addressed by focused laser beams, while two-qubit gates are driven by a global Rydberg pulse over the EZ. We define a *Rydberg stage S* as one synchronized entangling round that executes a set of two-qubit gates in parallel. Before the pulse, the compiler repositions atoms to respect the finite Rydberg interaction range: each intended gate $(u, v) \in G(S)$ is placed within the blockade

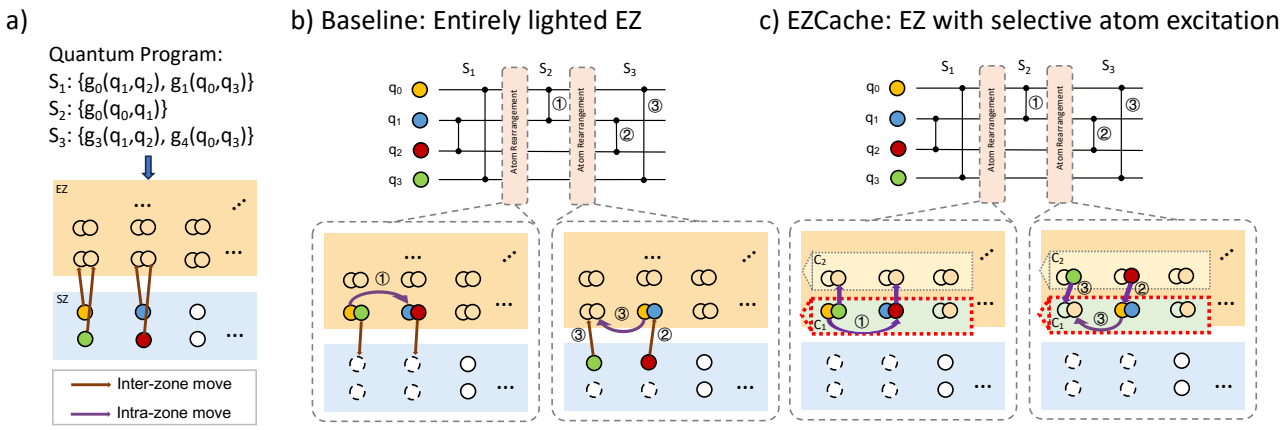

Figure 2: A motivating example. (a) A 4-qubit program scheduled into three Rydberg stages (S_1–S_3) with two-qubit gates. All qubits are initially mapped to storage-zone (SZ) sites. (b) Baseline compilation with an entirely illuminated entangling zone (EZ), where idle qubits cannot safely remain in EZ across stages; the circuit diagrams (black dots and lines) indicate the entangling operations in each stage, and the bottom panels illustrate the corresponding atom rearrangement, including four inter-zone moves. The numbered arrows (1)–(3) mark qubit movements for each two-qubit gate, not their execution order. (c) Circuit execution under spatially selective excitation. In each stage, EZ is partitioned into an illuminated execution subregion (C_1) and a dark safe subregion (C_2); the red dotted box indicates the subregion selected as the execution region, and the numbered arrows denote the gate-wise transport sequence.

radius, $\|\text{pos}(u)-\text{pos}(v)\| \le R_b$, so that the global pulse can entangle the pair, while all non-gate pairs $(i, j) \notin G(S)$ are kept farther apart, $\|\text{pos}(i) - \text{pos}(j)\| > R_b$, to suppress unintended Rydberg interactions [8].

Selective laser excitation. During a nominally global Rydberg pulse, platforms can suppress excitation on selected sites by locally detuning those atoms from the Rydberg transition using focused AC Stark shifts. Operationally, a site-dependent light field induces an energy shift Δ_i of the Rydberg transition at site i, so the effective detuning becomes $\delta_i = \delta_0 + \Delta_i$. When $|\delta_i|$ is made sufficiently large compared with the resonant Rabi frequency Ω, the excitation probability on that site is strongly suppressed, while nearby sites with $\Delta \approx 0$ remain addressable by the same pulse. This control has been demonstrated using spatial light modulators, digital micromirror devices, or auxiliary laser beams, enabling spatial patterns of "bright" (participating) and "dark" (suppressed) sites within the same tweezer array [8, 11].

Fidelity. The fidelity of the quantum circuit is affected by transport in two clear ways. First, long moves and many SZ–EZ crossings heat atoms and increase loss, so atoms are harder to recapture and gates become worse. Second, transport and rearrangement extend the time before each entangling pulse, so all qubits sit idle longer and accumulate decoherence and drift; recent neutral-atom processors report second-scale coherence times, so extra stage time can take a noticeable share of the coherence budget [3]. Therefore, cutting long-distance motion and avoiding repeated SZ–EZ shuttling directly improves reliability in staged execution.

2.2 Compilation Under Selective Illumination

The NAQC compilation — an informal overview. We consider a neutral atom device partitioned into a storage zone (SZ) and an entangling zone (EZ). A quantum program is scheduled into a sequence of synchronized Rydberg stages, where each stage contains a set of two-qubit gates executed under a pulse. For each stage, the compiler outputs a placement of all logical qubits onto physical trapping sites in SZ and EZ, and a movement plan that transforms the placement from one stage to the next. We categorize movements into: (i) *inter-zone transfers* that cross the SZ–EZ boundary, (ii) *intra-SZ moves* within SZ, and (iii) *intra-EZ moves* within EZ. A valid output must place the qubits of every two-qubit gate onto EZ sites that are close enough to interact, while keeping the overall transport cost low, including both the number of inter-zone transfers and the largest per-stage movement that determines stage preparation time under synchronized motion.

A motivating case. With selective excitation, only a subregion of EZ is illuminated during a pulse, so idle qubits may wait on dark EZ sites as long as they do not occupy illuminated sites at pulse time and enough dark capacity remains. Fig. 2(a) shows a three-stage circuit where (q_2, q_3) are active in S_1 and S_3 but idle in S_2, so they have a short idle window and are reused immediately. We use the distances shown in the figure: adjacent sites within a zone are spaced by 15 μm, and the SZ–EZ separation is 30 μm [14]. In the baseline in Fig. 2(b), idle qubits are forced to leave EZ during the pulse, so (q_2, q_3) make a round trip shuttling between SZ and EZ across $S_1 \to S_2$ and $S_2 \to S_3$, i.e., 4 inter-zone transfers and an extra distance of $4 \times 30 = 120$ μm (brown arrows). With selective excitation in Fig. 2(c), (q_2, q_3) can stay on dark EZ sites during S_2 and only perform short intra-EZ relocations; counting each purple intra-EZ arrow as one nearest-neighbor hop (15 μm), this reuse is supported by 4 intra-EZ moves for an extra distance of $4 \times 15 = 60$ μm (purple arrows). This change directly targets the metrics used later: it can reduce boundary crossings N_{inter}, and it can reduce

stage bottlenecks and movement volume (defined in Section 3) by replacing long SZ–EZ transfers with shorter intra-EZ moves.

2.3 Challenges

(1) Per-stage feasibility under synchronized execution. In each stage, the compiler must (i) place the qubits of every two-qubit gate onto EZ sites that satisfy the interaction-distance requirement, (ii) ensure that qubits not used in the stage are not located on illuminated sites during the pulse, and (iii) respect the finite number of available dark EZ sites for those waiting qubits. All qubits move between consecutive stages under collision-free and synchronized motion, so placement and movement are coupled across stages.

(2) Transport objective dominated by the slowest move. Stage preparation time is limited by the qubit that moves the farthest in that stage transition. Crossing the SZ–EZ boundary is typically longer than within-zone rearrangements, so unnecessary zone crossings can dominate the slowest-move cost. At the same time, keeping too many qubits inside EZ can increase contention and force longer within-EZ detours for some qubits.

(3) Capacity-limited residency decisions driven by near-future reuse. Dark EZ capacity is insufficient to hold all idle qubits. When a stage finishes, the compiler must decide which idle qubits remain in dark EZ sites and which are sent back to SZ, based on how soon they will be used again in later stages. Short-idle, soon-reused qubits should be prioritized for staying in EZ, while long-idle qubits should be evicted to SZ to free dark sites.

3 The EZCache Memory System

We formally describe our memory hierarchy system and introduce the main research objective in this section. We consider a NAQC architecture composed of a storage zone and an entangling zone. A circuit is executed in discrete *Rydberg stages* $\{S_t\}_{t=1}^{T}$, where each stage corresponds to one global entangling pulse with a fixed illumination pattern. Let G_t be the set of two-qubit gates executed in S_t. We denote the set of *active* qubits (atoms) in S_t by A_t (the qubits that appear in G_t) and the set of *idle* qubits by I_t, with $A_t \cap I_t = \emptyset$ and $A_t \cup I_t = Q$ for the logical-qubit set Q.

EZ partitions and illumination semantics. The EZ supports spatially selective Rydberg excitation: in each stage S_t, only a designated execution region is illuminated and can host entangling operations, while a disjoint region remains non-illuminated. We model this as a two-partition EZ with *Light-EZ* and *Dark-EZ* as shown in Figure 3. In stage S_t, the compiler assigns *roles* to these partitions. The illuminated execution role is denoted by $\mathrm{Exec}(S_t) \in \{\text{Light-EZ}, \text{Dark-EZ}\}$. The non-illuminated residency role is denoted by $\mathrm{Res}(S_t) \in \{\text{Light-EZ}, \text{Dark-EZ}\}$, with $\mathrm{Res}(S_t) \neq \mathrm{Exec}(S_t)$. These are logical roles that can change from stage to stage, rather than physically fixed regions. The key physical constraint is a safety invariant at the pulse time of stage S_t: an idle qubit must not be located in the illuminated region, i.e.,

$$I_t \cap \mathrm{Exec}(S_t) = \emptyset. \tag{1}$$

This invariant enforces that only qubits participating in the current entangling stage are exposed to Rydberg excitation.

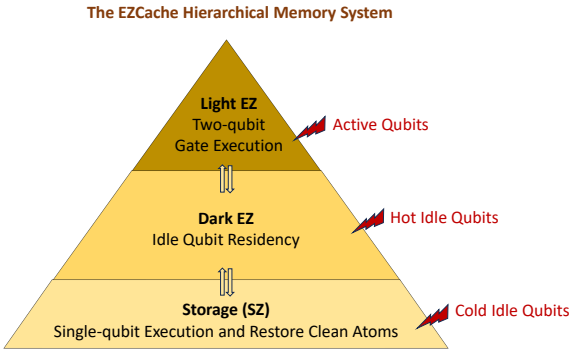

Figure 3: The three-layer EZCache system with different data types.

Cache-like residency mechanism. Selective illumination creates a strict separation between *execution* and *safe waiting* inside EZ. We interpret $\{\mathrm{SZ}, \mathrm{Res}(S_t)\}$ as the safe residency hierarchy across stages. SZ is the default long-lived safe region. The non-illuminated partition $\mathrm{Res}(S_t)$ is a capacity-limited residency cache inside EZ: it can safely hold qubits between stages without exposing them to the Rydberg pulse. In contrast, the illuminated partition $\mathrm{Exec}(S_t)$ is an execution region rather than a persistent storage region: qubits are brought into $\mathrm{Exec}(S_t)$ only to realize G_t under the entangling pulse, and Eq. (1) forces non-participating qubits to be absent from $\mathrm{Exec}(S_t)$ at stage time.

Resident and evict semantics of qubits. Residency is decided at stage boundaries. A qubit is *resident* at the boundary $S_t \to S_{t+1}$ if it is located in the non-illuminated partition $\mathrm{Res}(S_t)$ at the end of S_t and it stays inside EZ when transitioning to S_{t+1} (that is, it is not moved back to SZ at this boundary). A qubit is *evicted* at the boundary $S_t \to S_{t+1}$ if it is moved from $\mathrm{Res}(S_t)$ to SZ during this boundary transition. Let Cap be the number of trapping sites available for residency in $\mathrm{Res}(S_t)$. Note that an idle qubit currently in $\mathrm{Exec}(S_t)$ is first relocated to a site in $\mathrm{Park}(S_t)$ before the pulse; eviction refers exclusively to moving a qubit from $\mathrm{Res}(S_t)$ to SZ at a stage boundary. The boundary decision must satisfy the capacity constraint, so if more than Cap qubits compete for $\mathrm{Res}(S_t)$, some must be evicted to SZ even though $\mathrm{Res}(S_t)$ is safe.

Dark-EZ improves transport locality. The residency cache targets transport costs with a clear hierarchy. A transfer between SZ and EZ is a long move and often dominates both stage latency and transport-related errors. In contrast, moving atoms within EZ (between $\mathrm{Res}(S_t)$ and $\mathrm{Exec}(S_t)$) is usually much shorter. This makes *Dark-EZ* a locality enabler in two senses. First, it exploits *temporal locality*: many qubits become idle only for a small number of upcoming stages and will be used again soon, so keeping such qubits resident in the dark region avoids an immediate SZ round trip. Second, it improves *spatial locality*: while staying in the dark region, EZCache can place an idle qubit near likely future entangling sites, so that when the qubit becomes active again it needs a shorter move to reach a valid interaction site. Together, these locality effects yield two measurable outcomes: fewer inter-zone transfers (N_{inter}) and

smaller stage bottlenecks ($D_{\max}(t)$), which reduce stage latency under parallel motion.

Time-aware metadata for residency control. Residency decisions depend on *when* an idle qubit will next participate in an entangling stage. For each idle qubit $q \in I_t$, define the next-use distance

$$\Delta(q,t) = \text{next}(q,t) - t, \tag{2}$$

where $\text{next}(q,t)$ is the smallest stage index $t' > t$ such that $q \in A_{t'}$, and $\text{next}(q,t) = \infty$ if q never becomes active again. Given a reuse window $\omega \in \mathbb{Z}$ measured in *stage distance*, we classify q as *hot* if $\Delta(q,t) \le \omega$ and as *cold* otherwise. Equivalently, hot qubits are the qubits that will be used again within the next ω stages, so they are candidates for staying in $\text{Res}(\cdot)$ when capacity allows. Cold qubits have no near-term use within the window, so they are candidates for eviction to SZ to free dark sites. For hot qubits we further define an urgency score

$$\text{urg}(q,t) = \frac{\omega - \Delta(q,t) + 1}{\omega} \in (0,1], \tag{3}$$

which increases as the next entangling use approaches.

Atom transport objective. The stage-synchronous execution model implies that many atom moves can proceed in parallel, but each stage must wait for the slowest move before the Rydberg pulse can be applied. We model transport cost using a physical distance metric $d(\cdot,\cdot)$ between trapping sites. Let $\text{pos}_t(q)$ denote the location of qubit q right before executing stage S_t. The per-stage transport bottleneck is

$$D_{\max}(t) = \max_{q \in Q} d\big(\text{pos}_t(q), \text{pos}_{t+1}(q)\big), \tag{4}$$

which captures the longest single-qubit motion required to transition from S_t to S_{t+1} under parallel moves.

Definition 3.1 (Atom Transport Parallel Optimization Problem). *Given a circuit scheduled into T Rydberg stages $\{S_t\}_{t=1}^{T}$ with gate sets $\{G_t\}$ and active/idle qubit sets $\{A_t, I_t\}$, the compiler constructs a stage-indexed placement $\text{pos}_t : Q \to \text{Sites(SZ)} \cup \text{Sites(EZ)}$ for $t = 1, \ldots, T$ and chooses residency decisions at each boundary $t \to t{+}1$: each qubit currently in the dark region $\text{Res}(S_t)$ is either kept in EZ (stays in $\text{Res}(S_t)$) or evicted to SZ. These decisions must satisfy: (i) the safety invariant $I_t \cap \text{Exec}(S_t) = \emptyset$ in Eq. (1), (ii) the dark-region capacity constraint $|\text{Res}(S_t)| \le \text{Cap}$ for each t, and (iii) gate feasibility: for every $(u,v) \in G_t$, the placement must contain an executable pair of EZ sites within the Rydberg radius R_b, i.e., $d(\text{pos}_t(u), \text{pos}_t(v)) \le R_b$ with $\text{pos}_t(u), \text{pos}_t(v) \in \text{Sites(EZ)}$. The objective minimizes the parallel makespan*

$$M_{\text{exec}} = \sum_{t=1}^{T-1} D_{\max}(t), \qquad D_{\max}(t) = \max_{q \in Q} d\big(\text{pos}_t(q), \text{pos}_{t+1}(q)\big). \tag{5}$$

We additionally report the total movement volume $D_\Sigma = \sum_{t=1}^{T-1} \sum_{q \in Q} d(\text{pos}_t(q), \text{pos}_{t+1}(q))$ and the number of inter-zone transfers N_{inter} as secondary transport diagnostics.

4 Method

This section describes an EZCache compiler pass which follows the workflow in Figure 4 and computes a stage-indexed placement

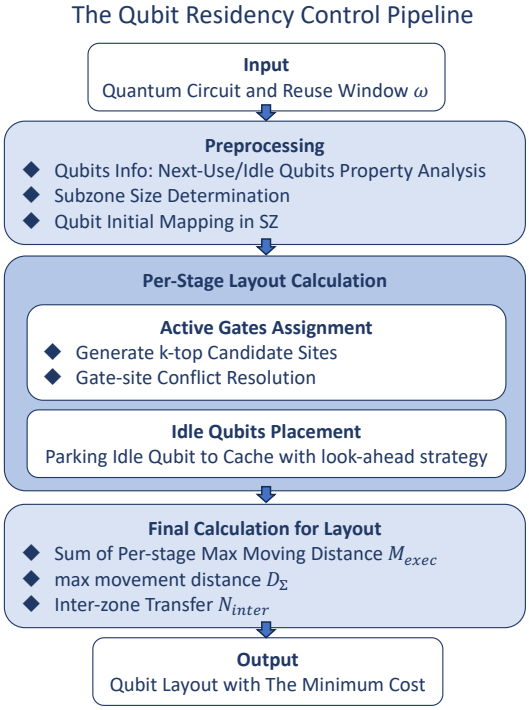

Figure 4: Workflow of the data management pipeline on EZCache.

by updating the layout state across Rydberg stages. We assume a fixed EZ layout with constant-speed atom motion. When estimating $D_{\max}(t)$ and D_Σ, we do not impose extra routing constraints from limited AOD resources. Instead, we treat parallel transport as unconstrained by the number of available AODs. Unless otherwise stated, all heuristic parameters are fixed across all experiments: the candidate-set size is $k = 8$, the reuse window is $\omega = 2$, the clearing and hot-qubit penalties are $(\lambda_f, \lambda_{\text{hot}}) = (0.5, 1)$, the active-gate movement balance is $\beta = 0.7$, and the immediate/look-ahead parking weights are $(\alpha, \gamma) = (1, 1)$.

4.1 Preprocessing

Preprocessing completes three tasks before per-stage control starts. First, it determines the sizes of the two EZ subzones using the stage decomposition. Let $\max_t |G_t|$ be the peak per-stage parallelism. We size $\text{Sites}(\text{Exec}(S_t))$ to accommodate this peak concurrency, and we size $\text{Sites}(\text{Res}(S_t))$ to provide a fixed residency capacity budget Cap used by the boundary decisions; these sizes are kept fixed during compilation. Second, given the reuse window ω, we precompute the time-aware metadata already defined in Section 3, in particular $\text{next}(q,t)$, $\Delta(q,t)$, and the derived hotness and urgency used by residency control and idle placement. Third, we initialize the layout state by mapping all logical qubits to SZ using the same initial mapping strategy as ZAC [12], which assigns each logical qubit to a distinct SZ site to form the placement before executing S_1.

4.2 Per-stage Compilation Pipeline

The input circuit is decomposed into Rydberg stages $\{S_t\}_{t=1}^{T}$. At each stage S_t, the compiler takes the stage sets (G_t, A_t, I_t), the current partition roles $(\mathrm{Exec}(S_t), \mathrm{Res}(S_t))$, and the inherited layout state (pos, Q), and outputs an updated layout state for the next stage. Here $\mathrm{pos}(q)$ is the current site of qubit q, and $Q(s)$ is the set of qubits currently occupying site s. We define the legal waiting and relocation candidate set for idle qubits as

$$\mathrm{Park}(S_t) = \mathrm{Sites}(\mathrm{Res}(S_t)) \cup \mathrm{Sites}(\mathrm{SZ}), \tag{6}$$

which contains all non-illuminated sites available during S_t. The controller updates the state in a fixed order: (1) it applies capacity-constrained residency control at the boundary by deciding which idle qubits stay in $\mathrm{Res}(S_t)$ and which idle qubits are evicted to SZ when the dark capacity would be exceeded; (2) it assigns one execution site in $\mathrm{Exec}(S_t)$ to each gate in G_t, and it enforces that different gates in the same stage use different execution sites; (3) after active qubits are moved to their execution sites, it assigns each idle qubit $q \in I_t$ a waiting site in $\mathrm{Park}(S_t)$, which by construction contains only non-illuminated locations; (4) it applies a final repair step right before the pulse to enforce Eq. (1): if any idle qubit is still located in $\mathrm{Exec}(S_t)$, it is relocated to a site in $\mathrm{Park}(S_t)$. We place this repair step last so intermediate clearing and gate realization do not need to maintain the invariant at every sub-step, while the final layout at pulse time always satisfies the safety condition.

Candidate set for relocating/parking idle qubits. Steps (1), (3), and (4) may need to move an idle qubit away from an illuminated location or away from a contested site. In all such cases, the destination must be a non-illuminated site. Therefore, all idle-qubit relocations and waiting placements are restricted to $\mathrm{Park}(S_t)$ defined in Eq. (6).

4.3 Active Gate Assignment

This module decides where each $g = (u, v) \in G_t$ executes inside $\mathrm{Exec}(S_t)$ and updates the layout accordingly. It follows three sub-steps: (i) candidate sites construction, (ii) distinct-site assignment, and (iii) placement realization with local clearing.

Generate candidate sites. Alg. 1 takes as input the active-gate set G_t, the execution-site set $\mathrm{Sites}(\mathrm{Exec}(S_t))$, and the current layout state (pos, Q), where $\mathrm{pos}(q)$ gives the current site of qubit q and $Q(s)$ gives the qubits currently occupying site s. It also uses the idle set I_t, the allowed relocation set $\mathcal{P}(q) \subseteq \mathrm{Park}(S_t)$ for each qubit q that needs to be moved away from a contested site, and the reuse-window metadata $(\mathrm{hot}, \mathrm{urg})$ to penalize moving qubits that will be used soon.

For each gate $g = (u, v) \in G_t$, the algorithm scans all execution sites $s \in \mathrm{Sites}(\mathrm{Exec}(S_t))$ (line 3–11) and assigns a rank value (line 10). The rank adds (i) the endpoint movement distance from the current sites $\mathrm{pos}(u)$ and $\mathrm{pos}(v)$ to s (line 4), and (ii) a clearing estimate when s is already occupied (line 6–9). To estimate clearing, it iterates over the current occupants of s other than u and v (line 6). Only idle occupants ($q \in I_t$) contribute to the estimate (line 7): each such blocker is hypothetically relocated to a feasible site in $\mathcal{P}(q)$, and the algorithm takes the cheapest relocation distance plus an urgency-weighted penalty when q is hot (line 8). The clearing estimate for s is the sum over all idle blockers encountered at s

Algorithm 1: Candidate site Construction for Active Gates in Stage S_t

Input: G_t; $\mathrm{Sites}(\mathrm{Exec}(S_t))$; layout (pos, Q); idle set I_t; feasible relocation sets $\mathcal{P}(\cdot) \subseteq \mathrm{Park}(S_t)$; metadata $(\mathrm{hot}, \mathrm{urg})$; parameters $(k, \lambda_f, \lambda_{\mathrm{hot}})$.
Output: Candidate sets $\{C(g)\}_{g \in G_t}$.

1 **foreach** $g = (u, v) \in G_t$ **do**
2 $\mathcal{L} \leftarrow \emptyset$;
3 **foreach** $s \in \mathrm{Sites}(\mathrm{Exec}(S_t))$ **do**
4 $f_{\mathrm{move}} \leftarrow d(\mathrm{pos}(u), s) + d(\mathrm{pos}(v), s)$;
5 $f_{\mathrm{free}} \leftarrow 0$;
6 **foreach** $q \in Q(s) \setminus \{u, v\}$ **do**
7 **if** $q \in I_t$ **then**
8 $f_{\mathrm{free}} \leftarrow f_{\mathrm{free}} + \min_{p \in \mathcal{P}(q)} \Big(d(\mathrm{pos}(q), p) + \lambda_{\mathrm{hot}} \cdot \mathrm{urg}(q, t) \cdot \mathbf{1}_{\mathrm{hot}(q)=1} \Big)$;
9 $\mathrm{rank}(g, s) \leftarrow f_{\mathrm{move}} + \lambda_f \cdot f_{\mathrm{free}}$;
10 $\mathcal{L} \leftarrow \mathcal{L} \cup \{(s, \mathrm{rank}(g, s))\}$;
11 $C(g) \leftarrow \mathrm{TopK}_k(\mathcal{L}$ by smallest rank$)$;
12 **return** $\{C(g)\}$;

(line 5–9). After evaluating all sites, the algorithm keeps the Top-k sites with the smallest ranks as $C(g)$ (line 12). The bounded size k limits the option set carried into the distinct-site assignment, while preserving multiple low-cost alternatives under site contention within the same stage.

Distinct-site assignment. Given $\{C(g)\}$, the pass selects one execution site per gate and enforces that different gates use different sites in the same stage. It processes gates in increasing order of $|C(g)|$, so gates with fewer options reserve a feasible site first. For each gate $g = (u, v)$ and each currently unused $s \in C(g)$, it evaluates a combined cost that prefers small per-gate movement, discourages large per-stage movement spikes, and includes the same clearing estimate used in candidate construction:

$$\mathrm{Cost}(g, s) = \beta \cdot \max(d_u, d_v) + (1 - \beta) \cdot (d_u + d_v) + f_{\mathrm{free}}(g, s), \tag{7}$$

where $d_u = d(\mathrm{pos}(u), s)$ and $d_v = d(\mathrm{pos}(v), s)$. The pass chooses the unused s with minimum $\mathrm{Cost}(g, s)$, assigns $\pi(g) \leftarrow s$, and marks s as used so later gates cannot reuse it in S_t.

Placement realization with local clearing. After the assignment $\pi(\cdot)$ is fixed, the pass updates (pos, Q) by executing the implied moves. For each gate $g = (u, v)$ with target $s = \pi(g)$, it first clears idle blockers currently occupying s by relocating each such blocker q to a feasible site in $\mathcal{P}(q) \subseteq \mathrm{Park}(S_t)$ using the same per-blocker minimization as in Alg. 1 (line 8). If a blocker belongs to another active gate in the same stage, it is handled when realizing that gate's own target site. Once s has enough free capacity, the pass moves u and v to s and updates $\mathrm{pos}(\cdot)$ and $Q(\cdot)$.

4.4 Idle Qubit Placement

This step takes the layout state (pos, Q) produced by the active-gate placement (Section 4.3) and updates it by assigning every idle qubit $q \in I_t$ to a safe waiting site. The candidate waiting sites are $\mathrm{Park}(S_t)$

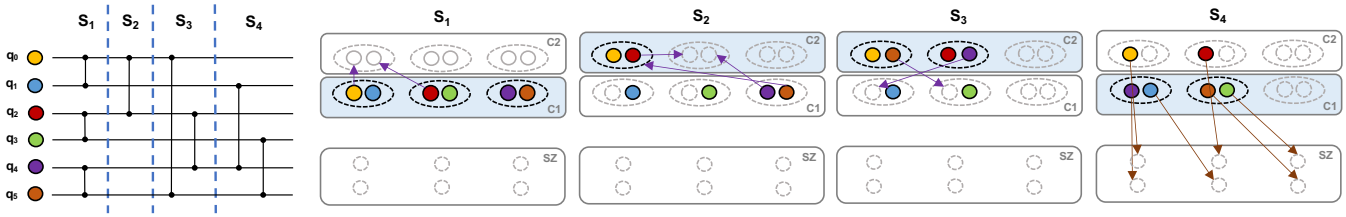

Figure 5: A 6-qubit running example scheduled into four Rydberg stages (S_1–S_4). Each panel shows the committed placement immediately before the pulse of stage S_t, with the illuminated execution subzone (enclosed by the blue rectangle) $\mathrm{Exec}(S_t)$ **alternating between** C1 **and** C2 **and the other subzone serving as the dark residency region** $\mathrm{Res}(S_t)$**; purple arrows indicate intra-zone movements and brown arrows show inter-zone moves.**

Algorithm 2: Idle Qubit Placement with Look-ahead Strategy in Stage S_t

Input: I_t; $\mathrm{Park}(S_t)$; current layout (pos, Q); $\mathrm{next}(q,t)$ and derived (hot, urg); parameters $(\alpha, \gamma, \lambda_{\mathrm{hot}}, k)$.
Output: Updated layout (pos, Q) after placing all $q \in I_t$.

1 **foreach** $q \in I_t$ **do**
2 $t' \leftarrow \mathrm{next}(q,t)$;
3 **if** $t' < \infty$ **then**
4 Identify the next-use gate g_q of q in stage $S_{t'}$;
5 Construct a bounded set $C(g_q) \subseteq \mathrm{Sites}(\mathrm{Exec}(S_{t'}))$ of size k;
6 $p^\star \leftarrow \arg\min_{p \in \mathrm{Park}(S_t)} \mathrm{Score}(q,p)$;
7 Move q to $p^\star$ and update (pos, Q);
8 **return** (pos, Q);
9 **Function** $\mathrm{Score}(q,p)$**:**
10 $now \leftarrow d(\mathrm{pos}(q), p)$;
11 $future \leftarrow \mathbf{1}_{t'<\infty} \cdot \min_{s \in C(g_q)} d(p,s)$;
12 $pen \leftarrow \lambda_{\mathrm{hot}} \cdot \mathrm{urg}(q,t) \cdot \mathbf{1}_{\mathrm{hot}(q)=1} \cdot \mathbf{1}_{p \neq \mathrm{pos}(q)}$;
13 **return** $\alpha \cdot now + \gamma \cdot future + pen$;

(Eq. (6)), and the decision uses the reuse metadata precomputed in preprocessing, namely $\mathrm{next}(q,t)$ and the derived hotness/urgency labels.

Per-qubit scoring and selection (Algorithm 2). For each idle qubit q, Alg. 2 evaluates a score for each candidate site $p \in \mathrm{Park}(S_t)$ and selects $p^\star$ with the smallest score (line 6). The score has three parts (line 6): (i) the immediate move distance from the current site $\mathrm{pos}(q)$ to p, (ii) a bounded-horizon term that anticipates q's next use when $\mathrm{next}(q,t) < \infty$, and (iii) an urgency-weighted penalty that discourages moving a hot qubit unless needed. When $\mathrm{next}(q,t) < \infty$, the algorithm identifies the next-use two-qubit gate g_q incident to q in stage $S_{t'}$ (line 3), constructs a bounded candidate set $C(g_q) \subseteq \mathrm{Sites}(\mathrm{Exec}(S_{t'}))$ (line 4), and uses the minimum distance from p to this set as the look-ahead term (line 6). The bounded size k keeps the look-ahead computing stable across stages while preserving multiple near-best execution options for the next-use gate, which is needed because those execution sites can also be contended by other gates in $S_{t'}$.

Look-ahead candidate set. For an idle qubit q at stage t, let $t' = \mathrm{next}(q,t)$ be the next stage where q participates in a two-qubit gate. Let that gate be $g_q = (q,u)$ in stage $S_{t'}$. We build a small candidate set $C(g_q)$ of EZ sites for executing g_q. We enumerate EZ site pairs within the Rydberg radius R_b and keep only pairs that do not violate the per-stage collision constraint. We then keep the top-k pairs with the smallest estimated move distance from the current positions. Finally, $C(g_q)$ is the set of sites that appear in these top-k pairs. We use the same k for all experiments.

4.5 Final Layout Decision

At each stage S_t, the compiler must choose which EZ partition serves as the illuminated execution region. Concretely, it evaluates two candidates by swapping the roles of $\mathrm{Exec}(S_t)$ and $\mathrm{Res}(S_t)$: (i) $\mathrm{Exec}(S_t)$=Light-EZ and $\mathrm{Res}(S_t)$=Dark-EZ, or (ii) $\mathrm{Exec}(S_t)$=Dark-EZ and $\mathrm{Res}(S_t)$=Light-EZ. For each candidate, EZCache runs the same within-stage pipeline in Fig. 4 to obtain a complete post-stage layout pos_{t+1} that satisfies the safety invariant in Eq. (1).

To commit the stage output, EZCache compares the two candidate layouts using a deterministic tie-breaking rule derived from the transport diagnostics in Section 3. Let $D_\Sigma(t)$ and $D_{\max}(t)$ be the per-stage transport volume and bottleneck distance, and let $N_{\mathrm{inter}}(t)$ be the number of qubits that change zones between SZ and EZ during the $S_t \rightarrow S_{t+1}$ transition. We first prefer the candidate with smaller $D_{\max}(t)$ to reduce M_{exec} in Definition 3.1. If both candidates have the same $D_{\max}(t)$, we break ties by smaller $D_\Sigma(t)$; if still tied, we break ties by smaller $N_{\mathrm{inter}}(t)$. The selected candidate layout is committed as the inherited state for stage S_{t+1}, and the full placement trace $\{\mathrm{pos}_t\}_{t=1}^{T}$ is obtained by repeating this selection across stages.

4.6 A Running Example

Figure 5 shows a 6-qubit instance scheduled into four Rydberg stages. At each stage S_t, EZCache evaluates the two possible EZ role assignments and commits the lower-cost layout; the figure shows the committed choices, with $\mathrm{Exec}(S_1)$ = C1, $\mathrm{Exec}(S_2)$ = C2, $\mathrm{Exec}(S_3)$ = C2, and $\mathrm{Exec}(S_4)$ = C1. We denote the left/middle/right EZ site pairs within subzone Ci as Ci-L, Ci-M, and Ci-R. All qubits start in SZ, and we use reuse window $\omega = 2$. To illustrate one transition ($S_1 \rightarrow S_2$), in S_2 we have $G_2 = \{(q_0, q_2)\}$ and $I_2 = \{q_1, q_3, q_4, q_5\}$. Active-gate assignment constructs the candidate set $C((q_0,q_2))$ = {C2-L, C2-M}, commits the execution site $\pi((q_0,q_2))$ = C2-L, and moves q_0, q_2 to that site pair. Idle placement then keeps q_4, q_5 (with next-use distance $\Delta = 1$) in the dark

subzone $\mathrm{Res}(S_2)$ and places q_1, q_3 on the remaining sites, so no idle qubit occupies $\mathrm{Exec}(S_2)$ at pulse time. The same steps apply to later stages, and the safety invariant $I_t \cap \mathrm{Exec}(S_t) = \emptyset$ holds at every pulse.

4.7 Complexity

Let $n = |Q|$, $m_t = |G_t|$, $B_t = |\mathrm{Sites}(\mathrm{Exec}(S_t))|$, and $P_t = |\mathrm{Park}(S_t)|$. Assume each site has constant capacity and each relocation checks a bounded number of feasible targets. In stage S_t, *candidate-site construction* (Alg. 1) takes $O(m_t \cdot B_t)$ time, *distinct-site assignment* takes $O(m_t \log m_t + m_t \cdot k)$ time, *placement realization* takes $O(m_t)$ moves up to a constant factor, and *idle-qubit placement* (Alg. 2) takes $O(|I_t| \cdot P_t)$ time, with an additional $O(|I_t| \cdot B_{t'})$ factor when look-ahead is enabled and future candidate sets are constructed at stage $t' = \mathrm{next}(q, t)$.

5 Evaluation

Research questions. Our evaluation answers four questions that map to EZCache goals:

- **RQ1 (Transport, heating, and time reduction).** Does EZCache reduce (i) overall moving time dominated by the slowest transport, (ii) total transport distance as an indicator of motional heating and atom-loss risk, and (iii) inter-zone qubit shuttling between SZ and EZ, compared with prior zoned compilers?
- **RQ2 (Fidelity mechanisms).** Does safe residency in dark EZ regions improve circuit fidelity, and which mechanisms (excitation, transfer, decoherence) dominate?
- **RQ3 (Component necessity).** How much does each core component contribute (reuse window and look-ahead idle parking), and are they complementary?
- **RQ4 (Scalability).** How does EZCache scale with circuit size in compilation time?

Experimental setup and architecture. We evaluate EZCache on a zoned neutral atom architecture with a storage zone (SZ) and an entangling zone (EZ), using the same geometry assumptions as PowerMove: a 2D tweezer array with inter-trap spacing of 15 μm and an SZ–EZ separation of 30 μm [14]. We model AOD-controlled collective transport with the acceleration bound $a_{\mathrm{aod}} \le 2750\,\mathrm{m/s^2}$ [4]. We model trap transfer between SLM and AOD with duration $t_{\mathrm{tran}} = 15\,\mu$s [2], and we use gate durations $t_{1q} = 1\,\mu$s for one-qubit gate, $t_{\mathrm{CZ}} = 270$ ns for two-qubit gate, and $t_{\mathrm{exc}} = 270$ ns for one excitation of Rydberg pulse.

Baselines. We compare EZCache with two zoned neutral atom compilers, **ZAC** [12] and **PowerMove** [14]. ZAC emphasizes reuse-aware placement and load-balanced routing across stages to reduce data movement overhead and improve hardware utilization, and it introduces an intermediate representation that abstracts qubit motion into rearrangement jobs. PowerMove improves stage-to-stage transitions through a stage scheduler that reduces inter-zone interchange, a router that integrates allocation with movement without fixed intermediate layouts, and a collective-move scheduler that increases movement parallelism and favors longer SZ dwell time. In contrast, EZCache leverages selective illumination to keep a capacity-limited dark EZ region available as a residency layer, and it co-optimizes entangling-qubit execution placement with idle-qubit residency and waiting-site choices across stages.

Benchmarks. We evaluate four benchmark families at multiple circuit sizes, ranging from a few qubits up to 100 qubits. Among them, QAOA circuits contain consecutive entangling stages with dense two-qubit interactions, and thus serve as our primary stress tests for comparing transport and fidelity behavior across zoned compilers. The benchmark families are: (1) **QAOA-rand**: QAOA with random ZZ interactions, where each qubit pair is included as an interaction edge with distinct probability; (2) **QAOA-regular**: QAOA with ZZ interactions following a regular-graph connectivity pattern; (3) **Bernstein–Vazirani (BV)**: circuits generated from random secret strings, with approximately balanced 0/1 bits; (4) **Quantum Fourier Transform (QFT)**: standard QFT circuits at different qubit counts.

Stage decomposition and reuse window. Given the ordered two-qubit gate list, we build stages by a greedy maximal rule. We keep adding a gate to the current stage if it does not reuse any qubit. If it reuses a qubit, we start a new stage. This produces a deterministic stage sequence $\{S_t\}_{t=1}^{S}$. We use the same $\{S_t\}$ for EZCache, ZAC, and PowerMove. We do not apply extra circuit rewrites or re-scheduling outside the compared compilers. Unless stated otherwise, we set the reuse window to $\omega = 2$ for all methods. In ablations, we change ω only for EZCache to isolate the component effect.

5.1 Evaluation Metrics

We execute a circuit in S Rydberg stages. Within a stage, atom moves can overlap in time, and the stage completes when the slowest move completes. We assume a constant transport speed v as mentioned before, so movement time is proportional to movement distance.

Stage duration. Let $D_{\max}(t)$ be the maximum movement distance among all qubits that move in stage t. The movement time of stage t is therefore $T_{\mathrm{move}}(t) = D_{\max}(t)/v$. We report the sum of per-stage bottlenecks

$$\mathrm{SumDmax} = \sum_{t=1}^{S} D_{\max}(t), \tag{8}$$

which is directly proportional to the total movement time $\sum_{t=1}^{S} T_{\mathrm{move}}(t)$ under constant v.

Total movement distance. We also report

$$\mathrm{TotalDist} = \sum_{t=1}^{S} \sum_{q \in Q} d\big(\mathrm{pos}_{t-1}(q),\, \mathrm{pos}_t(q)\big), \tag{9}$$

which measures the total movement distance accumulated by all qubits across stages. We use it as a distance-based indicator of motional heating and the associated atom-loss risk, and to distinguish bottleneck reduction *SumDmax* from the total distance reduction *TotalDist*.

Inter-zone transfer count. To quantify costly crossings between SZ and EZ, we report the number of SZ–EZ boundary crossings N_{inter}.

Peak EZ idle occupancy. To study the effect of the reuse window parameter ω, we measure how many idle qubits remain in the EZ under different ω settings. Let $C_{\mathrm{EZ}}(t)$ denote the number of idle qubits placed in the dark EZ region at stage t. We define

$$C_{\mathrm{EZ}}^{\max} = \max_{t\in\{1,\dots,S\}} C_{\mathrm{EZ}}(t), \tag{10}$$

which captures the peak number of idle qubits staying in EZ across all stages. By varying ω, we report how this peak value changes, reflecting the tradeoff between longer EZ residency and potential waiting risk.

Fidelity model and diagnostics. Total circuit fidelity follows the same factorized model and parameter setting as PowerMove:

$$F_{\mathrm{tot}} = F_{2q}\cdot F_{\mathrm{exc}}\cdot F_{\mathrm{tran}}\cdot F_{\mathrm{dec}}. \tag{11}$$

Here $F_{2q} = \prod_{g\in\mathcal{G}_{2q}} f_{2q}$, where $\mathcal{G}_{2q}$ is the set of compiled two-qubit gates and f_{2q} is the per-gate CZ fidelity. The excitation factor is $F_{\mathrm{exc}} = f_{\mathrm{exc}}^{N_{\mathrm{exc}}}$, where N_{exc} counts excitation opportunities for idle qubits exposed to illuminated regions. The transfer factor is $F_{\mathrm{tran}} = f_{\mathrm{tran}}^{N_{\mathrm{tran}}}$, where N_{tran} counts trap-transfer and inter-zone (or cross-subzone) transport events. The decoherence factor is $F_{\mathrm{dec}} = \exp(-T_{\mathrm{idle}}/T_2)$, where T_{idle} is the accumulated waiting and transport time and T_2 is the coherence time.

Eq. (11) uses the neutral-atom values adopted in PowerMove [14]: single-qubit gates take $\sim 1\,\mu$s with 99.99% fidelity; CZ gates take 270 ns with $f_{2q} = 99.5\%$; trap transfer has fidelity $f_{\mathrm{tran}} = 99.9\%$; and idle qubits exposed to illuminated regions use $f_{\mathrm{exc}} = 99.75\%$. For RQ2, we report F_{tot} together with the component factors $(F_{2q}, F_{\mathrm{exc}}, F_{\mathrm{tran}}, F_{\mathrm{dec}})$ and the diagnostics $(N_{\mathrm{exc}}, N_{\mathrm{tran}}, N_{\mathrm{inter}})$.

To explain which factor drives the fidelity gap, we also use additive terms $E_x = -\log(F_x)$ for $x \in \{2q, \mathrm{exc}, \mathrm{tran}, \mathrm{dec}\}$. This is useful because Eq. (11) is a product, and taking $-\log(\cdot)$ turns it into a sum: $E_{\mathrm{tot}} = E_{2q} + E_{\mathrm{exc}} + E_{\mathrm{tran}} + E_{\mathrm{dec}}$. So each E_x becomes a directly comparable loss term in the same unit, and the stacked bars in Fig. 7 can show the relative importance of different error sources.

6 Results

6.1 RQ1 — Transport and Time Reduction

This subsection explains whether EZCache reduces transport distance, movement time, and SZ–EZ traffic for QAOA circuits. Fig. 6 reports the median and IQR of relative improvements on the two QAOA families (left: QAOA-rand; right: QAOA-regular). Table 1 lists several representative circuits and reports the raw values of SumDmax, TotalDist, and N_{inter} for PowerMove, ZAC, and EZCache, together with the corresponding improvements or reductions.

Total movement time reduction. Fig. 6 shows that, for both QAOA families, EZCache usually has a positive improvement on SumDmax compared with both PowerMove and ZAC across many n_q values (the median stays above 0% in most settings). This means the longest move in each stage becomes shorter more often, so the summed stage time (under constant-speed motion) becomes smaller. Table 1 gives concrete examples. For qaoa-rand with $n_q = 10$, SumDmax drops from 649.11 μm (PowerMove) and 1735.16 μm (ZAC) to 477.22 μm (1.36× and 3.64×). For qaoa-regular with $n_q = 50$, SumDmax drops from 14745.77 μm (PowerMove) and 22163.16 μm (ZAC) to 10801.55 μm (1.37× and 2.05×). Overall, the figure and the table both support point (iii) in RQ1: the total movement time across stages is reduced.

Atom-loss indicator reduction. Fig. 6 also shows that EZCache usually improves TotalDist on both QAOA families, so the total moved distance over all qubits becomes smaller in many sizes. This matters for point (ii) in RQ1, since TotalDist is used as a distance-based indicator of motional heating and atom-loss risk. Table 1 again provides examples. For qaoa-rand with $n_q = 10$, TotalDist drops from 1934.26 μm (PowerMove) and 3635.49 μm (ZAC) to 1323.69 μm (1.46× and 2.75×). For qaoa-regular with $n_q = 50$, TotalDist drops from 87327.72 μm (PowerMove) and 50296.22 μm (ZAC) to 31051.89 μm (2.81× and 1.62×). Together with the SumDmax results, this shows that EZCache reduces not only the stage bottleneck distance, but also the total movement volume.

Inter-zone transfer reduction. For point (i) in RQ1, N_{inter} directly measures how often qubits cross the SZ–EZ boundary. In Fig. 6, the improvement on N_{inter} is present but smaller and less uniform than the improvements on SumDmax and TotalDist, especially on QAOA-regular, where the median reduction is often close to 0% for some sizes. Table 1 shows both behaviors. For qaoa-rand with $n_q = 10$, N_{inter} drops from 30 (PowerMove) and 37 (ZAC) to 20 (33.3% and 45.9% reductions). For qaoa-regular with $n_q = 50$, N_{inter} drops from 390 (PowerMove) and 403 (ZAC) to 354 (9.2% and 12.2% reductions). There are also cases where N_{inter} does not change (e.g., BV at $n_q = 14$ and $n_q = 50$), which matches the near-zero reductions seen in parts of the figure. This separation is important for answering RQ1: EZCache consistently reduces movement time and total distance, while the SZ–EZ crossing count depends more on the circuit structure and feasibility constraints.

6.2 RQ2 — Fidelity Breakdown

This subsection explains whether safe residency in dark EZ regions improves fidelity, and which error sources explain the gap. Fig. 7 reports the breakdown using additive terms $E_x = -\log(F_x)$ for $x \in \{2q, \mathrm{exc}, \mathrm{tran}, \mathrm{dec}\}$. Across methods, E_{2q} stays similar because it is mainly decided by the number of compiled two-qubit gates and the fixed per-gate fidelity, so the fidelity difference mainly comes from E_{exc}, E_{tran}, and E_{dec}. In our QAOA settings, the excitation term is small for all methods because idle qubits are typically kept off illuminated execution sites, so E_{exc} is near zero on median and does not dominate the separation. Transfer-related loss is driven by transport and transfer events, and RQ1 already shows that long moves and SZ–EZ crossings often drop for QAOA sizes, which supports fewer transfer-related events and a smaller E_{tran}. The decoherence term is tied to accumulated waiting and transport time; a smaller per-stage bottleneck distance (SumDmax) means less stage-level movement time under constant-speed motion, so the accumulated time term becomes smaller and E_{dec} drops. For example, at n_q=50 we observe large reductions in both transfer and decoherence terms: on qaoa-rand, the median E_{tran} drops from 0.50 (ZAC) and 0.29 (PowerMove) to 0.27 (EZCache), while E_{dec} drops from 3.47 (ZAC) and 2.36 (PowerMove) to 2.00 (EZCache), reducing the total E_{tot} from 4.22 (ZAC) and 2.89 (PowerMove) to 2.52 (EZCache); on qaoa-regular, E_{tran} drops from 0.50 (ZAC) and 0.29 (PowerMove) to 0.27 (EZCache) and E_{dec} drops from 3.28 (ZAC)

Table 1: Representative RQ1 results. Each row is a benchmark instance with n_q qubits. SumDmax and TotalDist are in μm. N_{inter} is a count. PM refers to the Powermove method.

Benchmark		SumDmax (μm)					TotalDist (μm)					N_{inter} (count)				
Family	n_q	PM	ZAC	Ours	Improv. vs PM(x)	Improv. vs ZAC(x)	PM	ZAC	Ours	Improv. vs PM(x)	Improv. vs ZAC(x)	PM	ZAC	Ours	Reduct. vs PM (%)	Reduct. vs ZAC (%)
qaoa-rand	7	111.70	458.84	85.56	1.31	5.36	407.64	818.75	339.67	1.20	2.41	7	10	7	0.0	30.0
qaoa-rand	10	649.11	1735.16	477.22	1.36	3.64	1934.26	3635.49	1323.69	1.46	2.75	30	37	20	33.3	45.9
qaoa-regular	10	1326.75	3080.91	967.34	1.37	3.18	4063.87	6450.55	2334.83	1.74	2.76	54	60	38	29.6	36.7
qaoa-regular	50	14745.77	22163.16	10801.55	1.37	2.05	87327.72	50296.22	31051.89	2.81	1.62	390	403	354	9.2	12.2
qaoa-regular	60	13617.12	21850.81	10884.59	1.25	2.01	95234.39	50890.46	32687.17	2.91	1.56	376	392	339	9.8	13.5
bv	6	222.25	1166.41	159.08	1.40	7.33	680.66	2473.12	547.19	1.24	4.52	16	26	8	50.0	69.2
bv	14	903.60	2010.83	761.03	1.19	2.64	2089.69	4034.97	1626.34	1.28	2.48	26	26	26	0.0	0.0
bv	50	2472.84	3472.30	1689.27	1.46	2.06	7759.76	7123.22	3496.53	2.22	2.04	44	44	44	0.0	0.0
qft	18	12821.54	19035.55	10061.96	1.27	1.89	28832.24	40394.12	21452.85	1.34	1.88	336	336	330	1.8	1.8

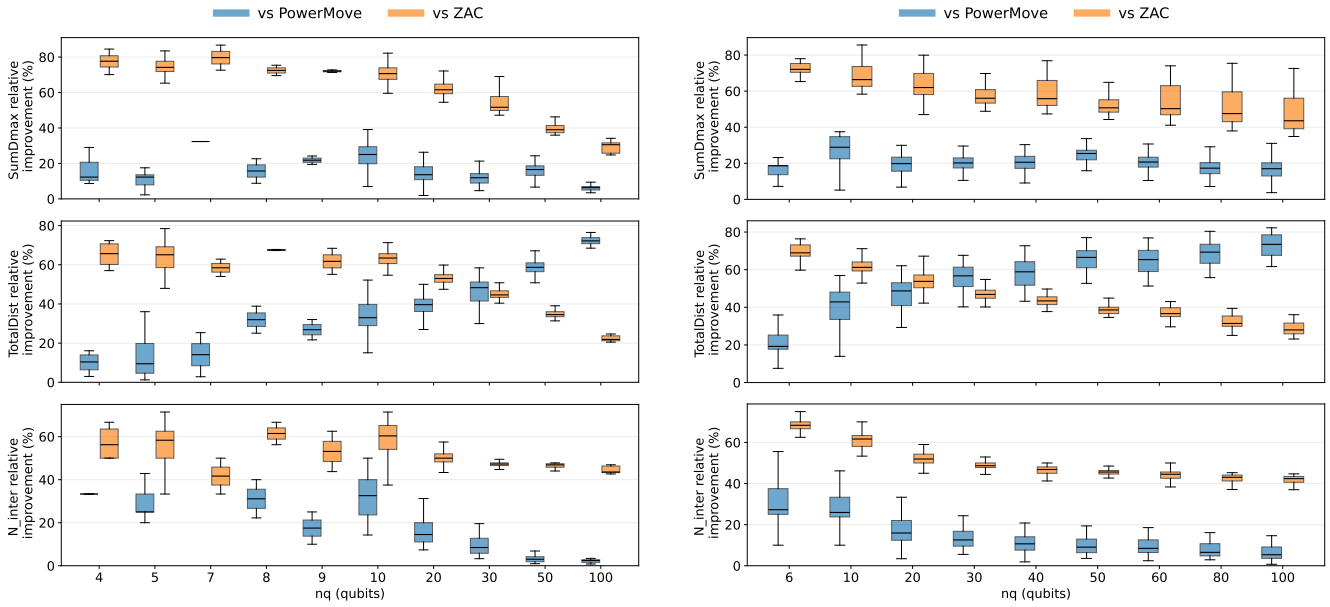

Figure 6: RQ1 instance-level improvements on QAOA families. Left: QAOA-rand; right: QAOA-regular. Each row reports EZCache's relative improvement (%) versus PowerMove and versus ZAC for SumDmax, TotalDist, and N_{inter} at each n_q (median with IQR, i.e., interquartile range, shown by the boxplots), where the relative improvement over a baseline $B \in \{\text{PowerMove}, \text{ZAC}\}$ is defined as $\frac{M_B - M_{\text{EZCache}}}{M_B} \times 100\%$, and M denotes the corresponding metric value.

and 2.44 (PowerMove) to 1.76 (EZCache), reducing E_{tot} from 4.03 (ZAC) and 2.97 (PowerMove) to 2.28 (EZCache).

6.3 RQ3 — Ablation and Sensitivity of Time-Aware Residency

This subsection evaluates the contribution of each core component and the sensitivity of the time-aware residency parameter. Fig. 8 ablates two components on the same base setting: no reuse-window residency ($\omega = 0$) and no look-ahead idle parking. We report two conservative variants that follow the same routing and pruning settings as the base, and use a fixed fallback rule to avoid regressions in SumDmax or N_{tran}. Reuse window enables reuse-window residency and then checks the resulting output against the base on the same circuit. If the enabled run increases either SumDmax or N_{tran} relative to the base, the variant falls back to the base output for that circuit. full algorithm further enables the look-ahead term and applies the same check against reuse window. This rule is applied to every tested circuit in the benchmark set, and the figure reports the resulting improvements versus the base for both SumDmax and N_{tran}.

The trends in Fig. 8 show that the reuse window component improves both metrics, and enabling look-ahead in full algorithm brings additional gains on top of it. For a concrete size example at n_q=20, full algorithm reduces N_{tran} by 12.1% on qaoa-rand and 13.4% on qaoa-regular on median, while also improving SumDmax by 3.1% and 3.0%, respectively.

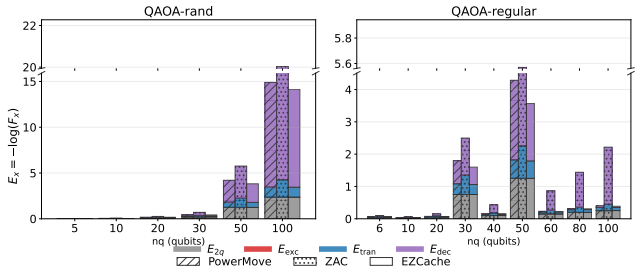

Figure 7: Breakdown via additive error contributions. We plot stacked $E_x = -\log(F_x)$ for $x \in \{2q, exc, tran, dec\}$, with methods (PowerMove, ZAC, EZCache) compared per n_q.

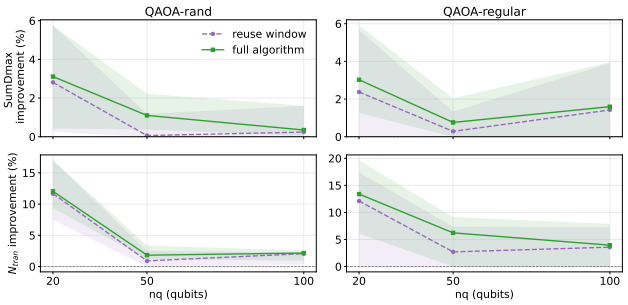

Figure 8: Ablation study. We compare (i) a base variant with ω=0 and f_{future}=0, (ii) *+reuse window* which enables reuse-window residency only when it does not worsen SumDmax nor N_{tran} versus the base, and (iii) *full-algorithm* which further enables look-ahead idle parking only when it does not worsen either metric versus *+reuse window*. Lines show the median improvement versus the base with IQR shading.

Fig. 9 further shows how the reuse window ω trades benefit and residency risk. The benefit axis is the N_{inter} improvement versus $\omega = 0$. The risk axis is $C^{EZ}_{\max}$, the peak idle occupancy in the dark EZ (defined in the metrics section). As ω increases, N_{inter} can improve further, but $C^{EZ}_{\max}$ can also increase. We use this plot to justify the default choice $\omega = 2$ under the stated $C^{EZ}_{\max}$ budget.

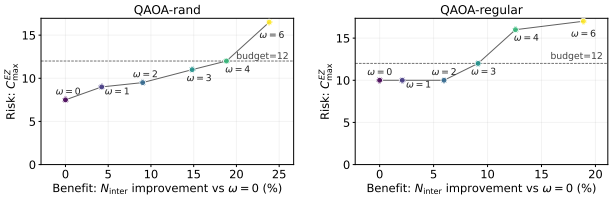

Figure 9: Reuse-window robustness. Each point corresponds to a reuse window ω, showing benefit (N_{inter} improvement versus $\omega = 0$) versus residency risk ($C^{EZ}_{\max}$). We also mark a risk budget $C^{EZ}_{\max} = 12$, which sets an upper bound on the allowed peak idle occupancy in the dark EZ.

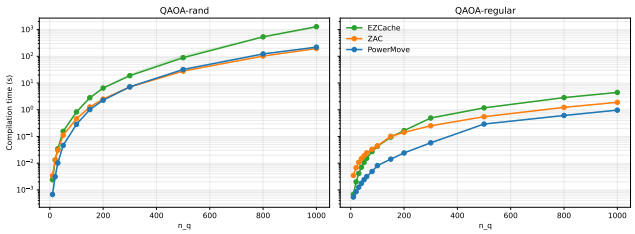

Figure 10: Scalability of compilation time versus n_q (log scale). We report wall-clock time of the compiler pass, excluding circuit generation, parsing, and file I/O. Points are per-instance measurements; lines show the median trend with IQR.

6.4 RQ4 — Scalability

This subsection explains how compilation time scales with circuit size. Fig. 10 reports compilation time T_{comp} on a log scale. Only the core compiler pass is timed, to avoid circuit parsing and file I/O effects. For EZCache, the timing uses one call to the stage-wise router with the default $\omega = 2$ and the same pruning settings as other experiments. PowerMove and ZAC are timed under the same stage sequence and environment. Per-instance measurements are summarized by the median and IQR. On qaoa-rand, the median T_{comp} at n_q=100 is 0.279 s (PowerMove), 0.429 s (ZAC), and 0.416 s (EZCache) under the same stage sequence. On qaoa-regular, the median T_{comp} at n_q=100 is 0.007 s (PowerMove), 0.044 s (ZAC), and 0.025 s (EZCache).

7 Related Work

Neutral atom platforms and scalable tweezer arrays. Optical-tweezer neutral atom systems scale from defect-free assembled arrays [1, 7] to thousands of trapped atoms [13]. Surveys summarize Rydberg interactions and the control stack [5, 15, 16, 21], motivating architectures that separate storage from entanglement where movement and in-zone waiting matter at scale.

Compilers for reconfigurable arrays and zoned architectures. Compilers for reconfigurable arrays optimize movement and scheduling under dynamic traps, e.g., OLSQ-DPQA [19] and Atomique [20]. For zoned designs, NALAC, Mantra, ZAC, and PowerMove exploit SZ–EZ separation and reuse to reduce transport [10, 12, 14, 18]. EZCache instead assumes selective illumination inside EZ and treats EZ residency as a capacity-limited state that can persist across stages, making admission, eviction, and intra-EZ migration explicit decisions under a safety invariant.

8 Conclusions

We present EZCache, a compilation framework that uses spatially selective illumination to make EZ residency a managed, capacity-limited state across Rydberg stages. By coordinating (i) qubits admission and eviction between SZ and EZ, (ii) active-gate placement in the illuminated region, and (iii) safe idle parking in dark EZ sites, EZCache replaces many long SZ–EZ transfers with short intra-EZ moves. Across 1,016 instances, EZCache improves transport cost and movement volume over prior zoned compilers, with mechanism-level fidelity accounting and scalability results.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant No. 2023YFA1009403, the National Natural Science Foundation of China under Grant Nos. 62472175 and 62502061, the "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software under Grant No. 22510750100, the Shanghai Frontiers Science Center of Molecule Intelligent Syntheses, the Natural Science Foundation of Chongqing under Grant No. CSTB2025NSCQ- GPX1300, and the Science and Technology Research Program of Chongqing Municipal Education Commission under Grant No. KJQN202500629.

References

[1] Daniel Barredo, Sylvain de Léséleuc, Vincent Lienhard, Thierry Lahaye, and Antoine Browaeys. 2016. An atom-by-atom assembler of defect-free arbitrary two-dimensional atomic arrays. *Science* 354, 6315 (2016), 1021–1023. doi:10.1126/science.aah3778

[2] Jérôme Beugnon, Charles Tuchendler, Harold Marion, Gaëtan, Yevhen Miroshnychenko, Yvan R. P. Sortais, Andrew M. Lance, Matthew P. A. Jones, Gaetan Messin, Antoine Browaeys, et al. 2007. Two-dimensional transport and transfer of a single atomic qubit in optical tweezers. *Nature Physics* 3, 10 (2007), 696–699.

[3] Dolev Bluvstein, Simon J Evered, Alexandra A Geim, Sophie H Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, et al. 2024. Logical quantum processor based on reconfigurable atom arrays. *Nature* 626, 7997 (2024), 58–65. doi:10.1038/s41586-023-06927-3

[4] Dolev Bluvstein, Harry Levine, Giulia Semeghini, Tout T. Wang, Sepehr Ebadi, Marcin Kalinowski, Alexander Keesling, Nishad Maskara, Hannes Pichler, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. 2022. A quantum processor based on coherent transport of entangled atom arrays. *Nature* 604 (2022), 451–456. doi:10.1038/s41586-022-04592-6

[5] Antoine Browaeys and Thierry Lahaye. 2020. Many-body physics with individually controlled Rydberg atoms. *Nature Physics* 16 (2020), 132–142. doi:10.1038/s41567-019-0733-z

[6] Sepehr Ebadi and et al. 2021. Quantum phases of matter on a 256-atom programmable quantum simulator. *Nature* 595 (2021), 227–232. doi:10.1038/s41586-021-03582-4

[7] Manuel Endres et al. 2016. Atom-by-atom assembly of defect-free one-dimensional cold atom arrays. *Science* 354, 6315 (2016), 1024–1027. doi:10.1126/science.aah3752

[8] Simon J. Evered, Dolev Bluvstein, Marcin Kalinowski, and et al. 2023. High-fidelity parallel entangling gates on a neutral-atom quantum computer. *Nature* 622 (2023), 268–272. doi:10.1038/s41586-023-06481-y

[9] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 212–219.

[10] Enhyeok Jang, Youngmin Kim, Hyungseok Kim, Seungwoo Choi, Yipeng Huang, and Won Woo Ro. 2025. Qubit Movement-Optimized Program Generation on Zoned Neutral Atom Processors. In *Proceedings of the 23rd ACM/IEEE International Symposium on Code Generation and Optimization*. 459–475.

[11] Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Tout T. Wang, Sepehr Ebadi, Hannes Bernien, Markus Greiner, Vladan Vuletić, Hannes Pichler, and Mikhail D. Lukin. 2019. Parallel Implementation of High-Fidelity Multiqubit Gates with Neutral Atoms. *Phys. Rev. Lett.* 123 (2019), 170503. doi:10.1103/PhysRevLett.123.170503

[12] Wan-Hsuan Lin, Daniel Bochen Tan, and Jason Cong. 2025. Reuse-aware compilation for zoned quantum architectures based on neutral atoms. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 127–142.

[13] Hannah J. Manetsch, Gyohei Nomura, Elie Bataille, others, and Manuel Endres. 2025. A tweezer array with 6,100 highly coherent atomic qubits. *Nature* 647 (2025), 60–67. doi:10.1038/s41586-025-09641-4

[14] Jixuan Ruan, Xiang Fang, Hezi Zhang, Ang Li, Travis Humble, and Yufei Ding. 2025. PowerMove: Optimizing Compilation for Neutral Atom Quantum Computers with Zoned Architecture. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '25)*. doi:10.1145/3676642.3736128

[15] Mark Saffman, T. G. Walker, and Klaus Mølmer. 2010. Quantum information with Rydberg atoms. *Reviews of Modern Physics* 82, 3 (2010), 2313–2363. doi:10.1103/RevModPhys.82.2313

[16] Ludwig Schmid et al. 2024. Computational capabilities and compiler development for neutral-atom quantum computing. *Quantum Science and Technology* 9, 3 (2024), 033001.

[17] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* 41, 2 (1999), 303–332. doi:10.1137/S0036144598347011

[18] Yannick Stade, Ludwig Schmid, Lukas Burgholzer, and Robert Wille. 2024. An abstract model and efficient routing for logical entangling gates on zoned neutral atom architectures. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Vol. 1. IEEE, 784–795.

[19] Daniel Bochen Tan, Dolev Bluvstein, Mikhail D. Lukin, and Jason Cong. 2024. Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors. *Quantum* 8 (2024), 1281.

[20] Hanrui Wang, Pengyu Liu, Daniel Bochen Tan, Yilian Liu, Jiaqi Gu, David Z. Pan, Jason Cong, Umut A. Acar, and Song Han. 2023. Atomique: A Quantum Compiler for Reconfigurable Neutral Atom Arrays. arXiv preprint. arXiv:2311.15123.

[21] Karen Wintersperger et al. 2023. Neutral atom quantum computing hardware: performance and end-user perspective. *EPJ Quantum Technology* (2023). doi:10.1140/epjqt/s40507-023-00190-1

[22] Jonathan Wurtz and et al. 2023. Aquila: QuEra's 256-qubit neutral-atom quantum computer. *arXiv preprint arXiv:2306.11727* (2023).