Optimized Compilation and Atom Shuttling for Reconfigurable Neutral Atom Arrays

Priyanshu Gourav
Dept. of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, India
priyanshu_gourav@kgpian.iitkgp.ac.in

Aditya Kumar Bharti
Dept. of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, India
akbaditya2005@kgpian.iitkgp.ac.in

Abhoy Kole
Cyber-Physical Systems
DFKI GmbH
Bremen, Germany
abhoy.kole@dfki.de

Indranil Sengupta
Dept. of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, India
isg@iitkgp.ac.in

Rolf Drechsler
Institute of Computer Science
University of Bremen / DFKI GmbH
Bremen, Germany
drechsler@uni-bremen.de

***Abstract*—Neutral atom quantum processors utilizing dynamic optical tweezers offer unique connectivity advantages through physical qubit transport (shuttling). However, minimizing the coherence loss associated with atom movement requires sophisticated compilation strategies. We propose a hybrid compilation strategy combining local heuristics and global matching. We introduce a tiered active-first planning strategy to minimize unnecessary reconfiguration and a Right-Filled Table (RFT) lookahead mechanism to optimize the placement of idle qubits. We primarily benchmark our approach against DasAtom, which represents the state-of-the-art for architectures relying exclusively on atom shuttling rather than error-prone SWAP operations. While DasAtom provides high fidelity through exhaustive global routing, its combinatorial optimization becomes a severe compilation bottleneck for large circuits. Our method significantly reduces compilation time (up to 30 times) while maintaining competitive fidelity ($< 0.6\%$ drop) on circuits with usable success probability (fidelity > 0.5). By trading a modest increase in movement operations for a highly scalable alternative, our approach effectively mitigates compilation bottlenecks. Furthermore, on sufficiently large grids, our framework achieves high parallelism, effectively narrowing the fidelity and runtime gap.**

***Index Terms*—Quantum Compilation, Neutral Atom Arrays, Reconfigurable Architectures, Atom Shuttling, Scalability.**

I. Introduction

Neutral atom arrays are a leading platform for scalable quantum computing, combining large qubit counts with long coherence times. Recent work has demonstrated reconfigurable arrays exceeding 6000 coherent atomic qubits [1], surpassing other modalities. Unlike superconducting systems with fixed coupling maps [2] or trapped ions constrained to low-dimensional geometries [3], neutral atom processors using Acousto-Optic Deflectors (AODs) support a natively scalable, reconfigurable 2D architecture [4]. Optical tweezers allow atoms to be physically transported between operations, enabling effective all-to-all connectivity and reducing reliance on SWAP gates [4], improving fidelity.

However, atom shuttling introduces non-negligible physical overhead. Finite transport times can induce motional heating, atom loss, and decoherence [4]. Consequently, compilers for neutral-atom quantum processors must minimize total transport distance to reduce latency and decoherence while mitigating the straggler effect, in which the slowest-moving atom determines the duration of each rearrangement stage which, in turn, reduces fidelity. Simultaneously, compilation must strictly enforce Rydberg blockade constraints, which prohibit concurrent multi-qubit interactions within a specified exclusion radius, thereby restricting gate concurrency [5].

Limitations of Prior Work: Static SWAP-based compilers [6], [7] incur heavy gate overheads and are unsuitable for shuttle-capable hardware, while dynamic compilation introduces an NP-Hard scheduling problem [8]. Existing dynamic solvers face a dichotomy: optimal ILP/SAT formulations scale only to small circuits, whereas state-of-the-art heuristic schedulers such as DasAtom [9], Atomique [10], and Enola [11] apply combinatorial optimization that becomes a bottleneck on large circuits (detailed in Sec. II).

In this paper, we propose a hybrid framework that injects future awareness into a scalable layer-wise planner. Our contributions are:

1) A *Right-Filled Table (RFT)* mechanism that provides lightweight lookahead, allowing the planner to pre-shuttle idle qubits toward future targets without the heavy cost of full trajectory optimization.
2) A *Tiered Assignment Strategy* that uses local heuristics to preserve array topology and reduce solver overhead.
3) An evaluation on dense (QTetris[1]) and organized (QFT) benchmarks showing significantly reduced compilation time at competitive fidelity versus global-optimization solvers such as DasAtom.

II. Background

To situate our approach, we summarize neutral atom hardware constraints and limitations of existing compilers.

[1]https://github.com/Huangyunqi/DasAtom/tree/main/Data/Q_Tetris

A. Neutral Atom Architecture

Neutral atom quantum processors trap qubits in optical potential wells. A *Spatial Light Modulator (SLM)* defines a large-scale static array of potential wells, providing a coherent environment for qubit storage and long-term idling. *Acousto-Optic Deflectors (AODs)* create movable tweezers that can pick up individual atoms and transport them across the 2D plane, enabling physical reconfiguration between gate cycles to achieve effectively all-to-all connectivity [12]. Entanglement is mediated via the *Rydberg blockade effect*: exciting an atom to a high-energy Rydberg state prevents simultaneous excitation of its neighbors. This defines the *restriction radius* (R_{res}), which is an exclusion zone where no other atoms can be excited to prevent crosstalk interference, and the smaller *interaction radius* (R_{int}), within which two-qubit CZ gates occur.

B. Compilation Challenges

Reconfigurable arrays introduce three primary spatio-temporal bottlenecks:

- **Transport Overhead**: Physical shuttling accumulates phase errors proportional to movement duration and risks atom heating or loss during trap hand-offs.
- **Geometric Packing**: Because $R_{\text{res}} > R_{\text{int}}$, each active gate blocks adjacent grid sites; maximizing parallelism requires packing interaction zones into the 2D plane without blockade violations.
- **Fragmentation**: Idle qubits may obstruct active paths. If these are naively evicted away from their future interaction partners, the resulting fragmentation necessitates long-distance, high-error transport in subsequent timesteps.

C. Prior Work

Early compilers relied on static layouts and SWAP-based routing [7]. The shift toward reconfigurable arrays introduced geometric shuttling, where *Tetris* [13] improves coherence via layout compaction but ignores transport overhead, and *Atomique* [10] suffers from fidelity loss due to mandatory SWAP gates in deep circuits.

Dynamic frameworks that strictly avoid SWAPs include *Enola* [11] and *DasAtom* [9]. While *Enola* uses lookahead-based scheduling, it scales poorly, with high compilation time on moderate-depth circuits. *DasAtom* establishes the state-of-the-art by maintaining stable layouts via subgraph isomorphism, empirically outperforming *Tetris*, *Enola*, and *Atomique* in fidelity by eliminating SWAPs and minimizing atom transfers [9]. However, *DasAtom* restricts the workspace to minimal $\sqrt{n}\times\sqrt{n}$ grids, limiting gate parallelism, and relies on NP-hard isomorphism checks that create computational bottlenecks for large-scale circuits. Similar scalability issues persist in optimal ILP-based schedulers [8].

In contrast, we prioritize scalability and maximal grid utilization. Rather than enforcing layout stability via combinatorial solvers, we embrace dynamic reconfiguration across the full grid to maximize parallel execution. By employing a lightweight Right-Filled Table (RFT) lookahead, we preemptively route atoms without the expensive solver overhead of DasAtom, providing robust temporal serialization on congested grids while significantly reducing compilation time for large-scale systems.

Choice of baseline. We benchmark exclusively against DasAtom, which leads Tetris, Enola, and Atomique on both geomean fidelity and geomean compile time on the QTetris suite (Table II of [9]); outperforming it on scalability therefore implies an advantage over the weaker baselines.

III. Proposed Method

We propose a dynamic compiler that accepts an arbitrary-sized quantum circuit, a neutral-atom grid architecture and the radius of interaction (R_{int}) as input (restriction radius (R_{res}) is taken as $2\times R_{\text{int}}$), and assigns logical qubits $Q = \{q_1, ..., q_n\}$ to time-varying physical grid locations

$$P(t) = \{\pi_q(t)\}_{q\in Q}, \quad \pi_q(t) = (x_q(t), y_q(t)) \in \mathbb{R}^2 \qquad (1)$$

at each discrete time step t, while satisfying interaction-range and Rydberg blockade constraints.

A. Overall Algorithm Framework

The compilation pipeline begins with a preprocessing stage that transforms the input quantum circuit into a format suitable for geometric planning, as illustrated in Fig. 1. The raw QASM circuit is first parsed into a Directed Acyclic Graph (DAG) to extract temporal dependencies. This DAG is then converted into a time-series interaction matrix. Using this matrix, the compiler performs a reverse scan (backward iteration from the circuit depth T_{max} down to 1) to generate the Right-Filled Table (RFT), which provides the lookahead data necessary for intelligent idle qubit placement (Sec. III-C)

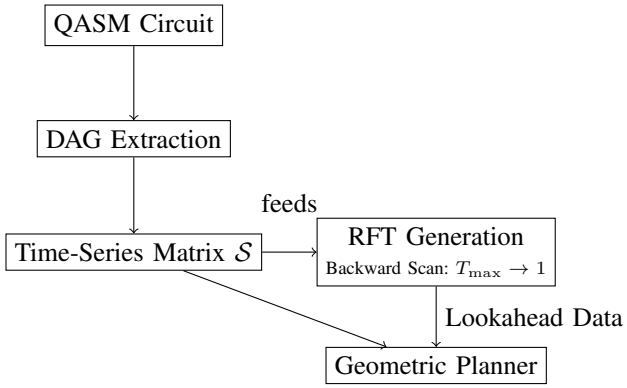


Fig. 1: **Data Flow of the Preprocessing Stage.** The raw circuit is converted into a time-series matrix $\mathcal{S}$. The RFT module then iterates over $\mathcal{S}$ in reverse (from $T_{\max}$ down to 1) to propagate future dependency information to earlier time steps, enabling intelligent idle parking. This reverse scan is an internal traversal order within the RFT Generation step.

With the lookahead data generated and a high-density initial layout constructed offline (Sec. III-B), the core of the framework is executed via a tiered geometric planner, detailed

in Alg. 1. The planner is designed to balance local stability with global feasibility by processing the circuit layer by layer.

To manage high gate density, a temporal serialization loop (Line 6) partitions active gates into blockade-safe batches. Within each batch, Phases 1–2 (Lines 8–23) apply lightweight local heuristics for lazy movement; Phase 3 (Lines 24–30) resolves the remainder via global Hungarian matching; Phase 4 (Lines 31–35) uses the RFT lookahead to optimize idle-qubit eviction.

Algorithm 1 Geometric Planner

Require: Circuit C, Grid (N_{row}, N_{col}), Interaction R_{int}, Static Layout L_{static}
Ensure: Sequence of Movement and Gate Instructions
1: **Preprocessing:** ▷ Explained in Sec. III-C
2: $\mathcal{S} \leftarrow$ TimeSeriesMatrix(C)
3: $\Phi \leftarrow$ RightFilledTable$(\mathcal{S})$
4: **for** $t = 1$ **to** $T_{\max}$ **do**
5: $Pairs_{pending} \leftarrow \{(q_i, q_j) \mid \mathcal{S}(i, j, t) = 1\}$ ▷ All active pairs at t
6: **while** $Pairs_{pending} \neq \emptyset$ **do** ▷ Temporal Serialization (Sec. III-D1)
7: $Slots_{accepted} \leftarrow \emptyset$
8: *// Phase 1: Wishlist Generation (Local Heuristics)* ▷ Sec. III-D2
9: $Wishlist \leftarrow \emptyset$
10: **for** $(q_i, q_j) \in Pairs_{pending}$ with current positions p_i, p_j **do**
11: **if** $\|p_i - p_j\| \leq R_{\text{int}}$ **then**
12: Add (p_i, p_j) to $Wishlist$ (Priority 0: Stay)
13: **else if** $\exists$ 1-qubit move yielding (p'_i, p_j) or (p_i, p'_j) where distance $\leq R_{\text{int}}$ **then**
14: Add the valid modified pair to $Wishlist$
15: **end if**
16: **end for**
17: *// Phase 2: Greedy Validation* ▷ Sec. III-D2
18: Sort $Wishlist$ by Priority $(0 \to 1)$
19: **for** target pair $(p_a, p_b) \in Wishlist$ **do**
20: IsFree $\leftarrow \forall (s_a, s_b) \in Slots_{accepted}, \{p_a, p_b\} \cap \{s_a, s_b\} = \emptyset$
21: IsSafe $\leftarrow \forall (s_a, s_b) \in Slots_{accepted}, \forall p \in \{p_a, p_b\}, \forall s \in \{s_a, s_b\}, \|p - s\| \geq R_{\text{res}}$
22: **if** IsFree **and** IsSafe **then**
23: Add (p_a, p_b) to $Slots_{accepted}$
24: **end if**
25: **end for**
26: *// Phase 3: Global Optimization (Subset that fits)* ▷ Sec. III-D3
27: $Movers \leftarrow Pairs_{pending} \setminus$ Pairs$(Slots_{accepted})$
28: $Slots_{static} \leftarrow \{s \in L_{static} \mid s$ is safe w.r.t $Slots_{accepted}\}$
29: $Num \leftarrow \min(|Movers|, |Slots_{static}|)$ ▷ Batch size
30: $Batch \leftarrow$ Select first Num pairs from $Movers$
31: CoM $\leftarrow \frac{1}{|Batch|} \sum_{q \in Batch} \pi_q(t)$ ▷ Center of movers
32: $Candidates \leftarrow$ Select Num from $Slots_{static}$ closest to CoM
33: Assign $Batch \to Candidates$ using **Hungarian Algorithm**
34: *// Phase 4: Idle Management (Eviction)* ▷ Sec. III-E
35: **for** $q_{evict} \in Qubits_{idle}$ displaced by Active Qubits **do**
36: $(q_{fut}, \delta) \leftarrow \Phi(q_{evict}, t)$ ▷ Future partner and time distance
37: $s^* \leftarrow \arg\min_{s \in Free} \|s - \pi_{q_{fut}}(t + \delta)\|$
38: **end for**
39: **Execute:** Generate SHUTTLE and GATE instructions
40: $Pairs_{pending} \leftarrow Pairs_{pending} \setminus (Batch \cup$ Pairs$(Slots_{accepted}))$
41: **end while**
42: **end for**

B. Monte Carlo Layout Optimization

To maximize hardware capacity for parallel gate execution, we perform a Monte Carlo-based layout optimization as an *offline preprocessing step* to construct a static layout L_{static} prior to the execution of the main algorithm. This step depends only on the grid geometry and hardware parameters (R_{int}, R_{res}), not on the specific circuit, and is therefore amortized across all circuits compiled on the same hardware configuration.

The objective is to maximize the number of mutually compatible interaction slots—pairs of physical coordinates separated by at most R_{int}—that can be activated simultaneously without violating the blockade constraint. Specifically, for any two simultaneous two-qubit gates acting on logical pairs (q_i, q_j) and (q_k, q_l) at a time step t, their physical coordinate mappings must satisfy:

$$\min_{a \in \{i,j\}, b \in \{k,l\}} \|\pi_{q_a}(t) - \pi_{q_b}(t)\| \geq R_{\text{res}} \tag{2}$$

ensuring blockade-safe parallel execution.

We perform multiple trials of randomized greedy placement. In each trial, candidate interaction pairs are incrementally selected while ensuring that no two selected pairs conflict under the restriction radius constraint in Eq. 2. The best trial produces a high-density static layout L_{static}, which defines a set of blockade-safe interaction positions that can support maximal parallel execution. Each of the K trials runs in $\mathcal{O}(N_{\text{row}}^2 N_{\text{col}}^2)$ worst case (each candidate site is screened against up to $\mathcal{O}(N_{\text{row}} N_{\text{col}})$ already-placed positions via a vectorized blockade check); since the layout depends only on $(N_{\text{row}}, N_{\text{col}}, R_{\text{int}}, R_{\text{res}})$, it is computed once per hardware configuration and excluded from the per-circuit compile times in Sec. IV.

C. Preprocessing and Lookahead Generation

The compilation pipeline transforms high-level circuits into a discrete temporal representation for geometric planning through two stages: dependency extraction and lookahead map generation.

1) Temporal Interaction Mapping: To map instructions to a reconfigurable grid, we convert the circuit into a Directed Acyclic Graph (DAG), parallelizing commuting operations into discrete time layers $t = 1, \ldots, T_{\max}$, where $T_{\max}$ denotes the circuit depth. Interactions are represented by a time-series matrix $\mathcal{S}$, where $\mathcal{S}(i, j, t) \in \{0, 1\}$ indicates if a two-qubit gate between q_i and q_j executes at time t. Fig. 2 and Table I illustrate this mapping.

The compiler computes a sequence of spatial configurations $\{P(t)\}_{t=1}^{T_{\max}}$. For every scheduled gate at time step t, the interacting qubits must satisfy the interaction radius constraint using the L_2 norm:

$$\left\|\pi_{q_i}(t) - \pi_{q_j}(t)\right\| \leq R_{\text{int}}. \tag{3}$$

Specifically, the sequence must satisfy the following conditions: (i) active pairs obey the interaction constraint in Eq. 3, (ii) simultaneous gates satisfy the blockade exclusion radius R_{res} (as defined in Eq. 2), and (iii) movements between consecutive configurations $P(t)$ and $P(t+1)$ are realizable through valid, collision-free shuttling operations. Subject to these constraints, the compiler minimizes the cumulative transport cost across timesteps, defined here as the critical-path movement cost:

$$\text{Cost} = \sum_{t=1}^{T_{\max}-1} \sum_{b=1}^{B_t} \max_{q \in \mathcal{M}_{t,b}} \|\pi_q(t, b) - \pi_q(t, b-1)\| \tag{4}$$

where $\mathcal{M}_{t,b}$ is the set of qubits moving during batch b of layer t. Here, $\pi_q(t,0) \equiv \pi_q(t)$ and $\pi_q(t,B_t) \equiv \pi_q(t+1)$, with B_t denoting the number of sequential routing batches required to transition between layers t and $t+1$ without trajectory collisions. For $q \notin \mathcal{M}_{t,b}$, the qubit remains stationary. The total cost is thus the sum of the longest individual movements (bottlenecks) across all sequential batches.

2) Right-Filled Table (RFT): To minimize future transport penalties, we construct a lookahead map Φ by backward iteration from $T_{\max}$ to 1. For qubit q at time t, $\Phi(q,t) = (p,\delta)$ encodes its next interaction partner p and the remaining time δ until that interaction. The recursion is: (i) active $\leftarrow (p,0)$ and (ii) idle $\leftarrow (p',\delta'+1)$ where $(p',\delta') = \Phi(q,t+1)$.

This guarantees that idle qubits retain knowledge of future partners. The last four columns of Table I present the RFT derived from matrix $\mathcal{S}$ for the circuit shown in Fig. 2.

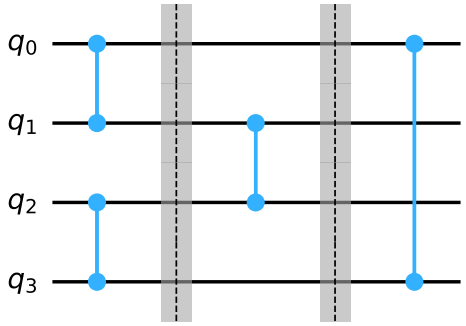


Fig. 2: Example 4-qubit quantum circuit scheduled over three discrete time layers.

TABLE I: Time-Series Matrix and Right-Filled Table (RFT) computed for the quantum circuit shown in Fig. 2.

Time-Series Matrix ($\mathcal{S}$)				Right-Filled Table (RFT)			
Pair	T1	T2	T3	Qubit	T1	T2	T3
(q_0,q_1)	1	0	0	q_0	$(q_1,0)$	$(q_3,1)$	$(q_3,0)$
(q_2,q_3)	1	0	0	q_1	$(q_0,0)$	$(q_2,0)$	–
(q_1,q_2)	0	1	0	q_2	$(q_3,0)$	$(q_1,0)$	–
(q_0,q_3)	0	0	1	q_3	$(q_2,0)$	$(q_0,1)$	$(q_0,0)$

D. Hybrid Slot Assignment Strategy

Mapping logical qubits to physical coordinates $P(t)$ is a combinatorial optimization problem. We propose a hybrid active-first strategy prioritizing localized, lazy movement via heuristics, falling back to global optimization only for unresolved pairs. To manage high gate density, this process executes in sequential batches per timestep t until all scheduled gates satisfy the interaction constraint (Eq. 3).

1) Temporal Serialization (Batching): If physical or blockade constraints prevent the simultaneous execution of all required gates at time t, the compiler enforces temporal serialization. Active pairs are partitioned into the largest possible blockade-compliant subsets. The compiler processes these batches sequentially, generating routing and gate instructions for the current subset before advancing to the remaining pending pairs A_t. This ensures architectural feasibility without modifying the original circuit.

2) Local Heuristics and Validation: For each active pair (q_i,q_j) in the current batch, the compiler evaluates a prioritized placement wishlist: (i) Priority 0 (Stay) retains the pair in situ if $\|\pi_{q_i}(t) - \pi_{q_j}(t)\| \le R_{\text{int}}$, and (ii) Priority 1 (Lazy Relocation) relocates exactly one qubit to an adjacent vacancy if it satisfies the interaction constraint. Candidates are greedily committed only if collision-free and blockade-safe ($\|\pi_{q_a}(t) - \pi_{q_b}(t)\| \ge R_{\text{res}}$) relative to already scheduled interactions.

3) Global Optimization via Hungarian Algorithm: Pairs unresolved by heuristics are classified as movers (M) and assigned to available blockade-compliant slots in the static layout. To prevent excessive global migration, we restrict the search space by computing the Center of Mass, $\text{CoM} = \frac{1}{|M|}\sum_{q\in M}\pi_q$, and filtering the physical slot list to retain only the $|M|$ closest candidate slots.

We map these pairs using a Minimum Weight Perfect Matching approach. We construct a cost matrix $\mathcal{C}$, where entry $\mathcal{C}_{k,l}$ defines the optimal transport distance for logical pair $k=(q_a,q_b)$, with current positions π_{q_a},π_{q_b}, to occupy physical slot $l=(p_1,p_2)$, accounting for orientation symmetry:

$$\mathcal{C}_{k,l} = \min\Big(\|\pi_{q_a}-p_1\| + \|\pi_{q_b}-p_2\|, \\ \|\pi_{q_a}-p_2\| + \|\pi_{q_b}-p_1\|\Big). \tag{5}$$

If a qubit is already at its target coordinate, its corresponding displacement penalty is zero. The Linear Sum Assignment (Hungarian) algorithm then solves for a matching that minimizes the total cost $\sum \mathcal{C}_{k,l}$. By matching movers to their most proximal slots via the CoM-constrained cost matrix, the solver effectively prevents the long-range intersecting paths typical of naive greedy mapping.

E. Lookahead-Aware Eviction Strategy

To minimize decoherence, our compiler applies a lazy movement principle, keeping idle qubits stationary. However, if an active qubit claims a physical site occupied by an idle qubit, the resident qubit must be relocated (eviction).

To optimize these forced relocations, we introduce a partner-optimal eviction strategy. Since the evicted qubit q_{idle} already incurs a transport penalty, we maximize utility by routing it toward its next interaction partner q_{fut} (predicted by the RFT). The compiler selects a parking site s^* from the available set $A = Grid \setminus P(t)$ that minimizes this future distance:

$$s^* = \arg\min_{s\in A} \left\|s - \pi_{q_{fut}}(t+k)\right\| \tag{6}$$

for $k \ge 0$. If no future interactions exist, q_{idle} defaults to the array's geometric center.

As shown in Fig. 3, a naive greedy approach relocates q_{idle} to the nearest vacancy (Spot A). Instead, our strategy selects Spot B. While immediate displacement is comparable, Spot B minimizes subsequent transport to q_{fut}. This future-

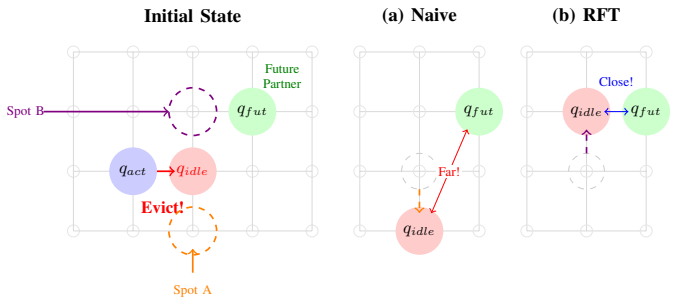


Fig. 3: **Eviction Logic.** The initial state (left) shows q_{idle} obstructing an active operation. (a) Naive placement may move q_{idle} to Spot A, causing high future transport cost. (b) Our RFT-aware method parks at Spot B, ensuring proximity to future partner q_{fut}.

awareness reduces cumulative transport overhead and shortens subsequent movement phases, improving effective coherence.

Post-assignment, a conflict-free shuttling strategy routes the physical qubits. It constructs a conflict graph of intersecting 2D paths and computes Maximal Independent Sets to execute non-overlapping movements in parallel batches. Cyclic dependencies (e.g., coordinate swaps) are safely resolved by transiently parking one qubit in the nearest available empty spot.

F. Complexity Analysis

Given N logical qubits and circuit depth T, the total number of two-qubit gates satisfies $G \leq NT/2$, the overall time complexity simplifies to $\mathcal{O}(TN^3)$, which is dominated by the $\mathcal{O}(k^3)$ Hungarian solver per batch, where k denotes the number of unresolved movers. Since $\sum k_{\text{batch}} \leq N$ per timestep, the worst-case cubic scaling is preserved.

Preprocessing requires $\mathcal{O}(N^2T)$ time. Space complexity is $\mathcal{O}(N^2T)$ for the time-indexed interaction matrix and $\mathcal{O}(NT)$ for the RFT structure; for $N, T \approx 1000$ this fits in ≈ 67 MB, comfortably scaling to projected architectures [1] exceeding 6,000 qubits.

IV. Experimental Results

We implemented our framework in Python 3.13.2 and evaluated it on the QFT and QTetris benchmark suites. All compilation experiments were conducted on an Intel Core i5 8th-generation processor with 8 GB RAM. Circuit fidelity is estimated using the DasAtom model [9], which assumes ideal transport ($f_{\text{trans}} = 1.0$) and calculates success probability via time-dependent decoherence and two-qubit gate errors:

$$F = \exp\left(-\frac{T_{\text{idle}}}{T_2}\right) \times f_{CZ}^{N_{CZ}} \tag{7}$$

where T_{idle} is the cumulative idle duration, $T_2 = 1.5$ s is the coherence time constant, N_{CZ} is the total CZ gate count, and $f_{CZ} = 0.995$ is the individual CZ fidelity. The source code is publicly available at https://github.com/ipash185/NAQC_Compiler.

A. Scalability and Parallelism

Fig. 4 illustrates scaling behavior for QFT circuits (5–60 qubits) mapped onto 11^2, 20^2, and 35^2 grid layouts. While DasAtom's compilation time (T_{comp}) [9] grows exponentially—exceeding 1,100 s for a 60-qubit circuit on a 35^2 grid—our hybrid heuristic resolves the layout in under 1 s (Fig. 4a), ensuring scalability. Although $|L_{\text{static}}|$ grows with grid area, local heuristics (Phases 1–2) resolve most pairs in QFT-like workloads, so the per-batch slot-filter cost is dominated by per-circuit preprocessing, keeping T_{comp} nearly grid-insensitive.

Increasing grid size reduces both circuit depth (T_{run}) and shuttling operations for both approaches (Fig. 4b, 4c). Assuming ideal shuttling, these results confirm that larger grids successfully enhance parallelism and execution efficiency.

B. Empirical Evaluation

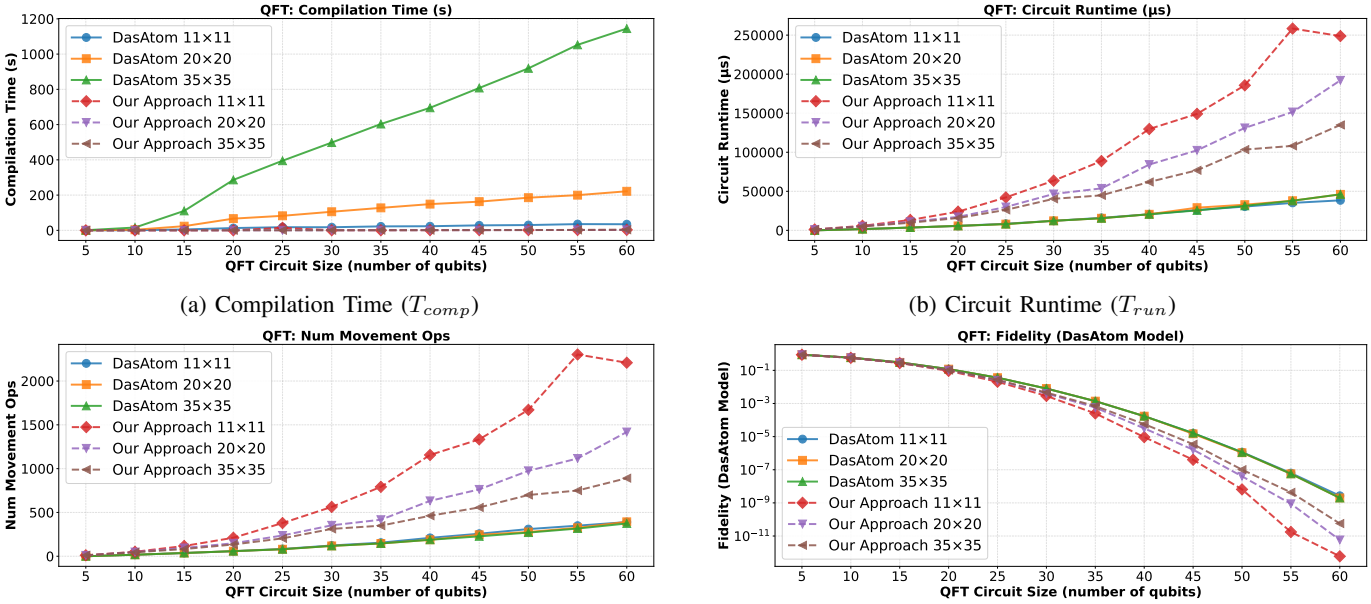
Table II benchmarks the `QTetris` suite on 11^2 and 35^2 grids to assess how hardware density impacts scheduling. We compare DasAtom and our proposed framework across Compilation Time (T_{comp}), Fidelity (Eq. 7), and Movement Operations (N_{move}).

- **Compilation Overhead:** For small `QTetris` circuits ($N \approx 16$), compilation time differences are marginal. Our framework's scalability advantage emerges primarily at larger problem sizes, as evidenced by the QFT scaling.
- **Fidelity vs. Movement Trade-off:** Our method prioritizes compilation speed, achieving large compilation time gains with acceptable fidelity loss in certain benchmarks. However, prioritizing rapid, localized heuristics over exhaustive global routing naturally increases total movement operations (N_{move}).
- **Grid Sizing:** Expanding grid capacity (e.g., 35^2) reduces congestion, improving our method's runtime, fidelity, and movement counts. While DasAtom's rigorous optimization ultimately yields fewer movements and marginally higher fidelity, our approach completely bypasses its combinatorial bottleneck, compiling 60-qubit circuits in under a second rather than hours.

V. Conclusion

We presented a multi-tiered compilation framework for reconfigurable neutral atom arrays to balance qubit connectivity with transport-induced decoherence. By integrating global Hungarian optimization for active gates with a Right-Filled Table (RFT) lookahead for idle qubits, we effectively managed physical movement without compromising fidelity.

Our results demonstrate that scalable compilation for neutral atom systems can be achieved without full global optimization, opening a path toward practical large-scale quantum programs. We currently assume ideal transport and per-pair CZ control; modeling movement-induced noise and extending the framework to DPQA-style architectures with global Rydberg pulses (which would require enforcing zone-wise gate compatibility in Phases 2–3) and to error-corrected zoned architectures



(a) Compilation Time (T_{comp})

(b) Circuit Runtime (T_{run})

(c) Movement Ops (N_{move})

(d) Fidelity (DasAtom Model)

Fig. 4: **QFT Scaling Analysis.** Comparison of (a) compilation scalability, (b) total runtime, (c) movement overhead, and (d) fidelity. Solid lines denote DasAtom [9]; dashed lines denote our Proposed framework. Note the logarithmic scale on T_{comp} and Fidelity. Compilation time scales polynomially for our method, while DasAtom [9] shows exponential growth.

TABLE II: Comprehensive Performance Benchmark on a Representative Subset of the QTetris Suite, comparing DasAtom and Proposed Method on 11^2 and 35^2 grid layouts.

Circuit	**Compilation Time (T_{comp}, s)**				**Fidelity (DasAtom Model)**				**Movement Operations (N_{move})**			
	11^2		35^2		11^2		35^2		11^2		35^2	
	DasAtom	Proposed	DasAtom	Proposed	DasAtom	Proposed	DasAtom	Proposed	DasAtom	Proposed	DasAtom	Proposed
4mod5-v1_22	0.051	0.090	0.894	0.023	0.946	0.944	0.946	0.942	0	2	0	5
adr4_197	5.145	1.454	76.358	1.432	5.0e-4	3.0e-4	5.0e-4	3.0e-4	55	470	55	459
alu-v0_27	0.101	0.069	0.974	0.027	0.918	0.909	0.918	0.908	0	9	0	10
bv_n16	0.145	0.063	2.842	0.042	0.926	0.919	0.926	0.920	2	9	2	8
cm152a_212	0.921	0.394	7.602	0.484	0.069	0.058	0.069	0.060	6	159	6	126
dc1_220	2.300	0.622	23.663	0.757	0.015	0.012	0.015	0.012	23	262	23	264
ising_model_16	0.123	0.085	0.723	0.047	0.471	0.459	0.471	0.465	0	22	0	9
misex1_241	26.977	1.921	643.382	2.435	2.6e-5	1.3e-5	2.6e-5	1.2e-5	52	689	52	710
qft_16	8.537	0.093	199.612	0.095	0.290	0.285	0.290	0.290	34	41	34	27
qv_n16_d5	91.010	0.037	2394.494	0.028	0.542	0.538	0.542	0.544	10	14	10	4
rd53_251	0.760	0.534	6.475	0.604	0.059	0.050	0.059	0.049	15	153	15	166
rd73_252	5.354	1.859	66.999	2.211	8.5e-6	4.1e-6	8.5e-6	4.1e-6	83	704	83	702
square_root_7	227.547	2.496	4571.890	2.808	1.7e-7	6.9e-8	1.7e-7	6.0e-8	106	909	106	866
sym6_145	2.560	1.310	18.933	1.855	1.9e-4	1.1e-4	1.9e-4	1.1e-4	40	504	40	577
z4_268	3.545	1.031	52.019	1.519	0.001	7.3e-4	0.001	7.5e-4	43	441	43	412

with mid-circuit measurement and feed-forward control remain future work.

REFERENCES

[1] H. J. Manetsch, G. Nomura, E. Bataille *et al.*, "A tweezer array with 6100 highly coherent atomic qubits," *Nature*, vol. 647, no. 8088, pp. 60–67, 2025.

[2] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[3] C. D. Bruzewicz *et al.*, "Trapped-ion quantum computing: Progress and challenges," *Appl. Phys. Rev.*, vol. 6, no. 2, 2019.

[4] D. Bluvstein *et al.*, "A quantum processor based on coherent transport of entangled atom arrays," *Nature*, vol. 604, no. 7906, pp. 451–456, 2022.

[5] M. Saffman, T. G. Walker, and K. Mølmer, "Quantum information with rydberg atoms," *Rev. Mod. Phys.*, vol. 82, no. 3, p. 2313, 2010.

[6] J. Wurtz *et al.*, "Aquila: Quera's 256-qubit neutral-atom quantum computer," *arXiv preprint arXiv:2306.11727*, 2023.

[7] P. Murali *et al.*, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proc. ASPLOS*, 2019, pp. 1015–1029.

[8] D. B. Tan *et al.*, "Compiling quantum circuits for dynamically field-programmable neutral atoms array processors," *Quantum*, vol. 8, p. 1281, 2024.

[9] Y. Huang, D. Gao, S. Ying, and S. Li, "Dasatom: A divide-and-shuttle atom approach to quantum circuit transformation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 44, no. 8, pp. 2966–2979, 2025.

[10] H. Wang *et al.*, "Atomique: A quantum compiler for reconfigurable neutral atom arrays," in *Proc. ISCA*, 2024.

[11] D. B. Tan *et al.*, "Compilation for dynamically field-programmable qubit arrays with efficient and provably near-optimal scheduling," in *Proc. ASP-DAC*, 2025.

[12] D. Bluvstein *et al.*, "Logical quantum processor based on reconfigurable atom arrays," *Nature*, vol. 626, no. 7997, pp. 58–65, 2024.

[13] Y. Jin, Z. Li, F. Hua *et al.*, "Tetris: A compilation framework for vqa applications in quantum computing," in *Proc. ISCA*, 2024, pp. 293–308.