Hybrid Circuit Mapping: Leveraging the Full Spectrum of Computational Capabilities of Neutral Atom Quantum Computers

Ludwig Schmid* Sunghye Park† Robert Wille*‡
https://www.cda.cit.tum.de/research/quantum
*Technical University of Munich, Germany
†Pohang University of Science and Technology, Korea
‡Software Competence Center Hagenberg GmbH, Hagenberg, Austria
{ludwig.s.schmid,robert.wille}@tum.de;shpark96@postech.ac.kr

Abstract

Quantum computing based on *Neutral Atoms* (NAs) provides a wide range of computational capabilities, encompassing high-fidelity long-range interactions with native multi-qubit gates and the ability to shuttle arrays of qubits. While, previously, these capabilities have been studied individually, we propose a fast hybrid compiler to perform circuit mapping and routing utilizing *both* high-fidelity gate interactions and qubit shuttling. We delve into the intricacies of the compilation process when combining multiple capabilities and present effective solutions to address the resulting challenges. The final compilation strategy is then showcased across various hardware settings, revealing its versatility, and highlighting potential fidelity enhancements achieved through the strategic utilization of combined gate- and shuttling-based routing. With the additional multi-qubit gate support for both routing capabilities, the proposed approach is able to take advantage of the full spectrum of computational capabilities offered by NAs.

1 Introduction

Neutral Atoms (NAs) have emerged as a compelling choice for universal quantum computing [11, 13, 25, 26], showcasing a broad spectrum of computational capabilities that encompass high-fidelity, long-range interactions between qubits with native multi-qubit gate support [9, 11, 15], and remarkable scalability [3, 22]. In response to these advantages, dedicated software solutions, such as compilers [17, 21, 30], have been developed to optimize performance while adhering to hardware constraints. Moreover, Bluvstein et al. [6][5] have demonstrated the ability to dynamically rearrange qubit arrays during computation with high fidelity. Based on this experimental progress, further compilation strategies [7, 19, 27, 31, 32] have explored the potential of using qubit shuttling for circuit mapping and routing, presenting it as a promising alternative to conventional approaches reliant on SWAP gate insertion.

Nevertheless, previous work individually only studied a single aspect or a subset of the full spectrum of the computational capabilities of NAs. In particular, SWAP gate insertion and atom shuttling have been considered separately from each other. While these separate studies of the mapping capabilities facilitate the initial understanding, they neglect potentially better solutions, arising from the combined use of both gate-based and shuttling-based mapping throughout the compilation process.

In this work, a hybrid compilation approach is proposed to explore the potential advantage of leveraging gate-based SWAP insertion *and* shuttling-based atom rearrangements. In particular, this entails the compilation task of mapping and routing a provided quantum circuit to NA hardware. The resulting compilation process is able to choose between the two mapping capabilities for each gate individually within the circuit based on available hardware information. Challenges arising from the simultaneous consideration of both mapping capabilities are discussed, and corresponding solutions are proposed and integrated into a hybrid mapping process. The heuristic approach employs two capability-specific cost functions, which are designed for rapid evaluation and consider additional information to enhance parallelism by incorporating commutation rules and look-ahead functionality.

The approach is evaluated on a set of benchmark circuits with up to 200 qubits, considering different hardware information. The evaluations demonstrate the ability of the proposed approach to correctly identify the preferred mapping capability of different hardware configurations. In particular, it shows fidelity improvements for mixed hardware that is characterized by similar operation fidelities between entangling gates and shuttling. Furthermore, the performed evaluations indicate a correlation between circuit structure and preferential mapping capability, offering new research questions, enabled by the hybrid mapping process.

Overall, the proposed hybrid approach is a further step to take advantage of the potential provided by NAs, offering new possibilities for mapping quantum circuits to hardware. With the additional support for arbitrary-sized multi-qubit gates for both gate- and shuttling-based mapping, it represents the first proposal to leverage the full spectrum of computational capabilities offered by the NA hardware. The full code of the proposed approach is publicly available as part of the *Munich Quantum Toolkit* (MQT)[1].

The remainder of this paper is structured as follows: Section 2 provides a brief background on NAs and the task of circuit mapping employing gates or shuttling. In Section 3, the hybrid compilation approach is introduced, by first discussing challenges and the corresponding solutions, with the complete process overview discussed afterward. In Section 4, we summarize the results of the numerical evaluations considering different hardware parameters and compiler settings, demonstrating the potential for fidelity improvements, and finally, Section 5 concludes the paper.

[1] https://github.com/cda-tum/mqt-qmap

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3655959

2 Background

This section provides a brief summary of the computational capabilities of the NA platform [11, 13, 25, 26, 27] followed by the compilation task of mapping quantum circuits onto NA hardware.

2.1 Neutral Atom Computational Capabilities

In NAs, to realize a computational register, atoms are stored in optical dipole traps, such as optical lattices or optical tweezer traps. These are created by interfering laser beams that create an array of potential valleys, effectively trapping the atoms at specific coordinates. Within this work, we assume these traps to lay on a regular square lattice with lattice constant d. Recent experimental work has demonstrated continuous reloading of thousands of atoms [12, 18]. Common atom species include alkali or alkaline-earth-like atoms such as Rb or Sr. The computational states can be encoded in long-lived atomic states such as hyperfine- or nuclear spin-states, after laser-cooling the atoms down to their motional ground states. Single qubit gates are then realized by laser pulses on individually addressed qubits or the whole register using globally applied laser beams. Multi-qubit gates are realized using the Rydberg blockade effect to introduce a phase shift conditioned on the qubit states of the nearby excited atoms [9, 15].

This theoretically allows realizations of m-qubit high-fidelity multi-controlled $C_{m-1}Z$ phase gates. For these gates to be executable, all participating qubits need to be within a certain *interaction radius* r_{int} to each other, where r_{int} depends on the atom species and the chosen Rydberg state. To reduce crosstalk between gates, parallel execution is only possible if qubits corresponding to different gates keep a distance of at least the *restriction radius* $r_{\text{restr}} \geq r_{\text{int}}$ to all qubits that execute another multi-qubit gate simultaneously. The resulting region is referred to as *restricted volume*.

Ex: 1. *The yellow region in Fig. 1a indicates the possible interaction candidates for multi-qubit gates for* $r_{\text{int}} = r_{\text{restr}} = 2d$. *The red region corresponds to the restricted volume for other multi-qubit gates to be executed simultaneously.*

In addition to these long-range interactions (for large r_{int}), NAs provide the capability to *shuttle* arrays of trapped atoms [5]. To this end, the qubits are loaded from the static *Spatial Light Modulator* (SLM) traps into a 2D *Acousto-Optic Deflector* (AOD). The AOD can be described by the corresponding x (column) and y (row) coordinates of the deflected laser, where each intersection defines a potential trap. Using more than one row and column allows for the shuttling of multiple qubits at once, each trapped in one of the intersecting coordinates. Each row/column coordinate can be activated, moved, and finally deactivated. This capability allows for arbitrary rearrangements of the atoms according to the two following constraints:

First, columns and rows are not allowed to cross each other, i.e., the ordering of the rows/columns always remains the same. To move beyond another trapped qubit the first has to be released back to a static SLM trap by deactivating the corresponding row and/or column and, then, cross as a second step only. Secondly, empty AOD intersections still represent a potential trap, disturbing qubits unintentionally when hovering or passing over them. The effect of these *ghost spots* can be circumvented by loading the qubits sequentially into the AOD traps, each with an additional offset movement to prevent the ghost spots from hovering over other qubits at any time.

Ex: 2. *Fig. 1b illustrates a scenario where* q_0 *needs to be shuttled to* q_1 *and* q_3, q_4 *to* q_2. *As a first step (1),* q_0 *is loaded by activating AOD coordinates at* $x_1 = 2d$ *and* $y_0 = d$. *To prevent problematic ghost spots with the following activations, a small offset move is applied to the coordinates. Then (2),* q_3 *and* q_4 *can be loaded simultaneously in the same row by activating* $x_0 = d$, $x_2 = 5d$ *and row* $y_1 = 3d$. *Note that due to the previous offset, all resulting ghost spots (light blue) are in the empty inter-qubit regions. The qubits can then be moved (3) to their destinations without crossing any other AOD coordinate and placed sequentially using again offset movements (4).*

2.2 Quantum Circuit Mapping

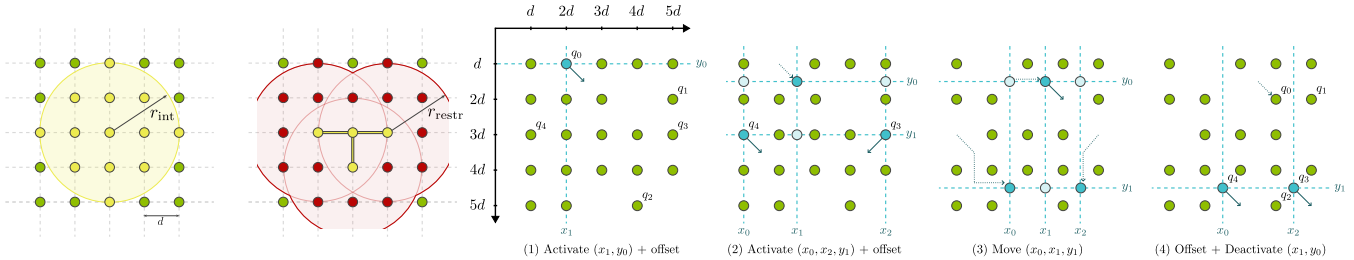
The task of *circuit mapping* consists of assigning circuit qubits $\mathbf{Q} = \{q_i\}_{i=0,\dots,n-1}$ to a set of physical qubits $\mathbf{P} = \{Q_a\}_{a=0,\dots,N-1}$. In addition, for NAs the mapping task becomes two-folded as the set of *physical qubits* $\mathbf{P}$ also has to be assigned to the possible *trap coordinates* $\mathbf{C} = \{C_\alpha\}_{\alpha=0,\dots,\mu}$. We assume $\mu = l^2 - 1 > m \geq n$ with a non-zero number of unoccupied coordinates on a regular $l \times l$ lattice. As a result, one has to consider two mapping steps: first, the *qubit mapping* f_{q} of assigning circuit qubits to physical hardware qubits, represented by trapped atoms. Secondly, the *atom mapping* f_{a} of assigning the physical qubits to the coordinates.

Given both mappings, one can define the *connectivity graph* $G = (\mathbf{P}, \mathbf{E})$, where $\mathbf{E}$ contains all pairs of physical qubits that can interact with each other, i.e., $(Q_a, Q_b) \in \mathbf{E} \Leftrightarrow d(Q_a, Q_b) \leq r_{\text{int}}$ with d the Euclidean distance. The set of qubits that are connected to Q are referred to as *vicinity* $V_{r_{\text{int}}}(Q)$ and $K_{r_{\text{int}}} = |V_{r_{\text{int}}}(Q)|$ as its *coordination number*.

Ex: 3. *The illustration in Fig. 2 shows a* 3×3 *lattice with* $|\mathbf{C}| = 9$ *SLM traps as well as* $|\mathbf{P}| = 4$ *physical and* $|\mathbf{Q}| = 3$ *circuit qubits to be mapped. The arrows indicate both qubit mapping* f_{q} *as well as an atom mapping* f_{a}. *For* $r_{\text{int}} = d$ *this results in* $\mathbf{E} = \{(Q_1, Q_2), (Q_2, Q_3)\}$ *and, e.g.,* $V(Q_2) = \{Q_1, Q_3\}$.

Due to the limited connectivity between all qubits, in the following step, G needs to be updated throughout circuit execution such that whenever a gate is executed, the corresponding physical qubits are in $\mathbf{E}$. This is referred to as *circuit routing* and is typically done by inserting SWAP gates into the circuit, swapping the qubit mapping assignment of two hardware qubits, and, this way, modifying G. On hardware, the SWAP gates can be realized using 3 CX gates or equivalently 3 CZ gates with additional single qubit rotations. In the last decade, multiple software tools to solve this problem for superconducting hardware have been developed, trying to minimize the number of SWAP gates [8, 16, 20, 23, 28, 33, 34]. For NAs, there are solutions taking into account variable interaction and restriction radii [2] as well as time-aware routing [17].

On the other hand, the shuttling capability of NAs additionally allows for changing the atom mapping by physically moving atoms to another lattice coordinate and modifying G this way. This has the advantage of not introducing additional error-prone CZ gates, but, depending on the hardware setup, may be significantly slower than SWAP gate insertion. Recent work has studied the potential of this novel mapping strategy, considering optimal solutions for a shuttling-only hardware setup by Tan et al. [29] and a shuttling-only heuristic routing algorithm by Nottingham et al. [19]. The latter allows the crossing of AOD rows/columns which allows the reduction of the shuttling problem to be tackled in a similar way to the SWAP gate insertion and can, therefore, not be compared directly to the shuttling constraints considered in this work.



(a) Interactions and restrictions for NA **(b) Qubit array shuttling with *Acusto-Optic Deflectors* (AOD) parallelization constraints.**

Figure 1: Computational capabilities of Neutral Atoms architectures

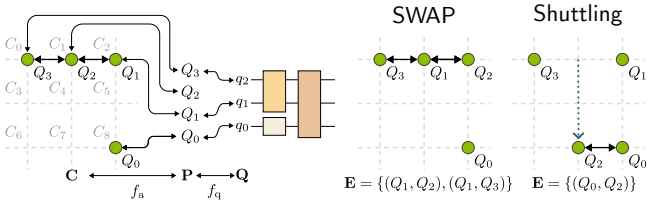


Figure 2: Qubit and atom mapping

Ex: 4. *Considering again Fig. 2, applying* SWAP(Q_1, Q_2)*, substitutes* (Q_2, Q_3) *with* (Q_1, Q_3) *in* $\mathbf{E}$*. Using shuttling, one can also modify* G *by placing* Q_2 *at* C_7 *and, this way resulting in* $\mathbf{E} = \{(Q_0, Q_2)\}$.

To evaluate the mapping results across different capabilities, common figures of merit such as SWAP gate count are no longer valid measures. A possible alternative [27] is the *approximate success probability* P defined as

$$P = \exp\left(-\frac{t_{\text{idle}}}{T_{\text{eff}}}\right) \prod_O \mathcal{F}_O\,, \quad T_{\text{eff}} = \frac{T_1 T_2}{T_1 + T_2}\,, \tag{1}$$

where the product runs over all circuit operations, T_1, T_2 are the coherence times of the system, and $\mathcal{F}_O \in [0, 1]$ is a measure for the average operation fidelity of operation O. The *total idle time* t_{idle} is defined as $t_{\text{idle}} = n \cdot T - \sum_O t_O$ with t_O the execution time of operation O and T is the total circuit runtime after scheduling, according to the constraints reviewed in Section 2.1.

3 Hybrid Mapping Problem

Considering the full spectrum of capabilities of NAs, as reviewed above, results in a two-fold mapping problem. First, in the *gate-based mapping* step, one can route gates that are not connected trivially by modifying the qubit mapping using SWAP gate insertion. Secondly, using *shuttling-based mapping*, the qubit mapping remains unaltered but the connectivity graph is modified by moving atoms to a new trap coordinate and, therefore, acting on the atom mapping. Recent approaches [2, 17, 19, 29] only considered one of the two cases separately, leaving untouched potential room for improvement. Motivated by that, this work proposes a compilation process that utilizes the two capabilities in a hybrid fashion, trying to employ the most suitable method to achieve the required connectivity. This imposes multiple challenges as illustrated in the following Section 3.1, with the corresponding solutions brought together to the proposed hybrid compilation approach reviewed in Section 3.2. Finally, Section 3.3 provides technical details such as the employed cost functions.

3.1 Challenges and Proposed Solutions

3.1.1 Increased Search Space The potential utilization of both mapping capabilities significantly increases the number of possible operations during the mapping process. This amplified search space may allow finding better solutions but it also imposes a challenge on the compiler that needs to decide between all possibilities. For the gate-based mapping, the set of possible SWAP operations corresponds to the union of possible SWAPs within r_{int} for all current gate qubits. In the worst case, this results in $O(nK_{r_{\text{int}}}/2)$, where n is the number of circuit qubits and $K_{r_{\text{int}}}$ is the coordination number. Since, as for many circuits, only a subset of all qubits participate in the next executable gates, this number will be significantly lower in most practical cases.

For shuttling-based mapping on the other hand, potentially any of the m physical atoms can be moved to any of the $\mu - m$ unoccupied coordinates. If one, furthermore, considers the possibility of moving away atoms from certain coordinates to free the space, effectively any possible rearrangement of qubits is possible, making a full search space exploration unfeasible.

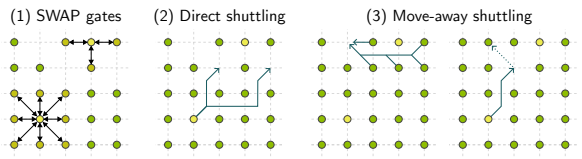
To reduce the search space, we only consider qubit movements that move the qubit directly in the vicinity of one of the other gate qubits. This can be done directly if there is an unoccupied coordinate. Otherwise, we select one of the nearby qubits to be moved away, resulting in a chain of two consecutive movements.

Ex: 5. *Fig. 3a illustrates the considered search space to connect two qubits for* $r_{\text{int}} = \sqrt{2}d$*. Shown are the possible operations for the three cases: (1) gate-based SWAPS, (2) directly shuttling to one of the free coordinates, and (3) requiring a move-away operation beforehand.*

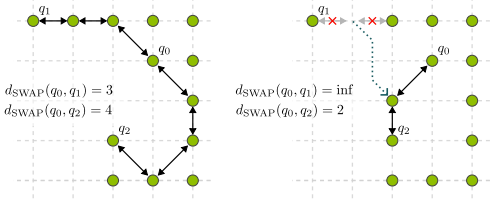
3.1.2 Mapping conflicts As both gate- and shuttling-based mapping directly affect the connectivity graph G, they can unintentionally conflict with each other. In particular, shuttling qubits away does not only affect the mapping of the moved qubits and their vicinity but it may also influence the optimal SWAP path for other qubits.

Ex: 6. *Fig. 3b illustrates the case, where a shuttling operation affects the number of necessary SWAPs* d_{SWAP} *also for gates that do not directly act on the moved qubit. In this case, the distance between* q_0 *and* q_2 *is reduced by the shuttling operation, while the connection* q_0 *to* q_1 *becomes impossible as the previous connection no longer exists.*

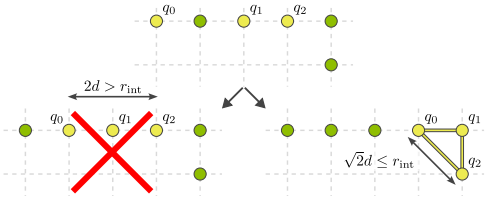
3.1.3 Multi-Qubit Gate-based Mapping As discussed in Section 2.1, we assume multi-qubit gates ($m \geq 3$) to be executable if all gate qubits are within the interaction radius of each other. This allows for different geometric arrangements of the qubits such as bulky clusters or other more spacious geometries for large r_{int}. While for static architectures the geometry is not a problem, the dynamic rearrangement of the qubits may result in a situation where there are not enough qubits or with the wrong geometric arrangement. As a result, one can not employ a distance-based cost function that directly drives qubits close to each other, but it actually has to be checked to see if the required geometric realization is possible and, if yes, where. This can be done with a breadth-first search, starting simultaneously from all gate qubits. If it is impossible to find a corresponding set of hardware qubits to execute the multi-qubit gate with SWAP gates, shuttling-based mapping must be employed instead.



(a) Increased search space: Considered operations.



(b) Mapping conflicts: Distance change by shuttling.



(c) Multi-qubit gate-based mapping: Positioning.

Figure 3: Challenges and proposed solutions.

Ex: 7. *Assuming $r_{\mathrm{int}} = \sqrt{2}d$, Fig. 3c illustrates a case where a distance-only "move-together" approach for multi-qubit gates mapping will fail. Due to the small r_{int}, the execution of the gate between qubits q_0,q_1 and q_2 requires a rectangular arrangement of the qubits. Therefore, driving the qubits close to each other results in a dead end. Instead, G has to be parsed to find the position with suitable geometry.*

3.2 Resulting Overall Mapping Process

Taking into account the discussed challenges and the respective proposed solution ideas, we propose the following overall hybrid mapping process to leverage gate-based and shuttling-based mapping. The process can be described as five major building blocks, illustrated graphically in Fig. 4.

(1) **Layer creation:** Creates a frontier layer f of gates that can be executed next, taking into account commutation rules. An additional lookahead layer l contains gates following the frontier layer up to a certain lookahead depth.

(2) **Decide for mapping capability:** For each gate, an estimate of the required number of SWAPs and shuttling operations is computed. Based on this estimate, an *approximate success probability* P_{g} and P_{s} according to Equation (1) is derived in both cases. After weighing the outcomes with α_{g} and α_{s} respectively the gate is assigned to the respective front/lookahead layers of gate-based f_{g}, l_{g} or shuttling-based $f_{\mathrm{s}}, l_{\mathrm{s}}$ mapping.

(3) **Gate-based Mapping:** Computes the best SWAP for all gates in f_{g} based on a distance-dependent cost function discussed in Section 3.3.1. For $m \geq 3$, a suitable position on the graph has to be found (use shuttling otherwise). This is repeated until at least one gate can be executed, resulting in updating the layers.

(4) **Shuttling-based Mapping:** Computes chains of possible shuttling operations and chooses the best one according to the cost function discussed in Section 3.3.2. This is again repeated until one of the gates can be executed in which case all layers are again updated. To prevent interference with the gate-based mapping, f_{s} is only considered if f_{g} is empty, meaning all necessary SWAP gates have already been applied.

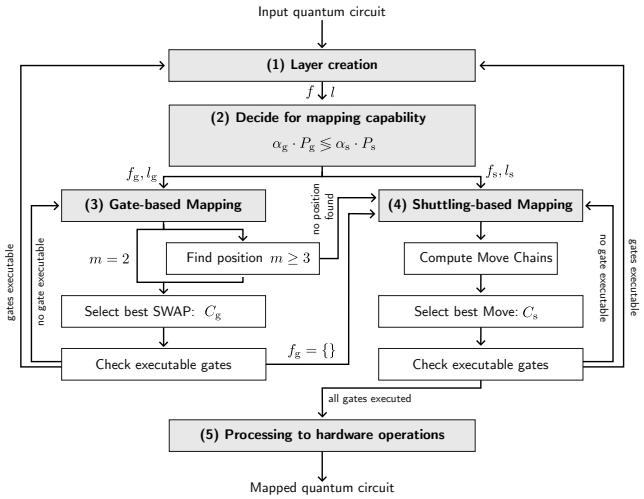


Figure 4: Resulting hybrid mapping process

(5) **Processing to hardware operations:** Finally, the SWAP gates are decomposed to natively supported CZ and single-qubit gates. Simultaneously, the shuttling operations are scheduled in parallel according to the AOD constraints and converted to native AOD operations, entailing AOD activation, deactivation, and movements of the AOD coordinates (see Example 2).

The resulting output can then be scheduled to compute characteristics such as the total circuit execution time T, or the total qubit idle time t_{idle}. This scheduling step also takes into account the restriction constraint regarding the parallel execution of multi-qubit gates discussed in Section 2.1.

In the following, some parts of the compilation process are discussed in more detail, in particular, the employed cost functions and multi-qubit mapping for $m \geq 3$.

3.3 Implementation Details

3.3.1 Gate-based Routing According to the previous discussions in Section 3.1.3, for the gate-based mapping it is necessary to differentiate between two and multi-qubit gates with $m \geq 3$. The former can be swapped close to each other, while the latter requires the search for a suitable geometric position to execute the gate. This position is found by employing a breadth-first search on G, starting simultaneously from all gate qubits and choosing the resulting position that requires the least number of SWAPS. Assuming a position P_g is found for gate g, the following cost function based on the two-qubit cost of Li et al. [16] is evaluated for each SWAP candidate S:

$$C_{\mathrm{g}}(S) = e^{-\lambda^t t(S)} \left[C_{\mathrm{g}}^{f}(S) + w_l \cdot C_{\mathrm{g}}^{l}(S) \right] \tag{2}$$

$$C_{\mathrm{g}}^{f}(S) = \sum_{g \in f_{\mathrm{g}},\, m=2} \Delta d_{\mathrm{SWAP}}(S, g) + \sum_{g \in f_{\mathrm{g}},\, m \geq 2} \Delta d_{\mathrm{SWAP}}(S, P_g), \tag{3}$$

where $\Delta d_{\mathrm{SWAP}}(S, g|P_g)$ returns the difference in the number of SWAPs for gate g or position P_g after the application of S. C_{g} computes the cost for the front and lookahead layer, with a *lookahead weighting factor* w_l. This sum is then weighted with an exponential factor constituted by a decay rate λ^t and a "last used" integer $t(S)$ to favor the use of SWAPs acting on different qubits to increase parallelism. In difference to Li et al. [16], $t(S)$ also takes into account restricted qubits due to the NA-specific constraint of r_{restr}. By choosing different values for $\lambda \in \mathbb{R}^+$, this allows a continuous

control between high parallelism and minimizing the absolute number of SWAP gates. Depending on the given hardware setup, one of the two figures of merits may be favorable over the other [27], and tuning λ allows for more hardware adaptive mapping compared to other NA compilers such as Baker et al. [2] or Li et al. [17].

3.3.2 Shuttling-based Routing As discussed in Section 3.1.1, considering all possible shuttling operations is unfeasible, and only the operations moving the qubits directly to their destination are taken into consideration. In general, there are two cases (as discussed in Example 5): a direct move M to an unoccupied coordinate, or a move combination (M_{away}, M) consisting of a first move-away operation, followed by the direct move to the now free coordinate. These moves are then combined into a chain of movements, listing all shuttling operations required to execute a certain gate, where the chain length is bounded by $2(m-1)$. This represents the worst case where all gate qubits have to use a move-away combination to make the gate executable. These movement chains are created for each gate qubit, always keeping chains of minimal length, based on the intuition that two move operations are unlikely to be faster than a direct single operation, even if they can be shuttled in parallel. The chains themselves are created by recursively choosing a qubit of the vicinity V of the central gate qubit and considering the possibility of moving all other gate qubits close to this position. This is done by first choosing qubits that allow for a direct move, resulting in a fast and close-to-optimal selection of interesting move operations. Each move chain is then evaluated by summing the following cost function over all individual moves M contained in the chain:

$$C_{\text{s}}(M) = C_{\text{s}}^{f}(M) + w_l\, C_{\text{s}}^{l}(M) + w^t C_{\text{parallel}}^{t}(M) \tag{4}$$

$$C_{\text{s}}^{f}(M) = \sum_{g \in f_{\text{s}}} \Delta d(M), \quad C_{\text{parallel}}^{t}(M) = \sum_{M^t} \Delta T(M, M^t)\,, \tag{5}$$

where the first two terms correspond to a distance reduction in the front and lookahead layer, similar to Equation (2). C_{parallel}^{t} takes into account if the the moves of the move chain can be executed in parallel to the last t move operations M^t, i.e.,

$$\Delta T(M, M^t) = \begin{cases} 0 & \text{, parallel loading \& shuttle} \\ t_{\text{act}} + t_{\text{deact}} & \text{, parallel shuttle} \\ t_{\text{act}} + s(M)/v + t_{\text{deact}} & \text{, else} \end{cases}$$

where $t_{(\text{de})\text{act}}$ is the (de)activation time of the AOD coordinates, v the shuttling velocity, and $s(M)$ the rectangular shuttling distance of movement M. By varying the *time weight* w^t one can, therefore, control the contribution of the parallelism constraint to the overall cost function and, therefore, it plays a similar role to λ^t in Equation (2) controlling the trade-off between choosing the most effective operation versus the one that can be executed best in parallel with previous operations.

For further technical details, we refer to the code, which is publicly available as part of the Munich Quantum Toolkit.

4 Numerical Evaluations

The advantage of the hybrid mapping approach proposed in this work is its flexibility when it comes to different hardware configurations. By leveraging SWAP gate insertion and shuttling, the mapper can choose for each gate in the front layer the currently favorable mapping capability with potential improvements regarding circuit runtime and average circuit fidelity. These benefits have been confirmed in experimental evaluations, with the main results of the evaluations summarized in this section.

4.1 Experimental Setup

We performed mapping experiments across three different hardware configurations whose corresponding settings are summarized in Table 1c and based on state-of-the-art hardware experiments [4, 5, 9, 14, 27]. They correspond to a (1) shuttling-optimized, a (2) gate-optimized, and a (3) mixed configuration that does not have a favorable mapping capability. The considered hardware size is a 15×15 lattice with $d = 3\,\mu\text{m}$ and $N = 200$ atoms in all cases.

As benchmarks, we utilized three representative circuits [24], namely the *Quantum Fourier Transform* (QFT), *Quantum Phase Estimation* (QPE), and a *graph state preparation circuit* (graph) for $n = 200$ qubits. To account for multi-qubit gates, we additionally considered three benchmark circuits representing classical binary reversible functions synthesized by [1] using C_mX gates with $m \leq 4$ denoted *bn*, *call*, and *gray* (Table 1b). For all circuits, C_mX have been decomposed to natively supported C_mZ gates.

For each hardware and circuit, the mapper is executed in three different modes: (A) gate-based only: $\alpha_{\text{s}} = 0$, (B) shuttling-based only: $\alpha_{\text{g}} = 0$, and (C) hybrid mapping utilizing the full hybrid compilation process proposed in Section 3. For the hybrid mode, different decision radios $\alpha = \alpha_{\text{g}}/\alpha_{\text{s}}$ are tested, keeping only the best. The remaining mapping parameters are $\lambda^t = 0$, $w_l = 0.1$, $w^t = 0.1$, and $t = 4$. A trivial identity mapping is chosen for the initial layout, i.e., $q_i \leftrightarrow Q_i \leftrightarrow C_i$ for all circuit qubits.

To evaluate the results, both the original and the mapped circuits are scheduled, taking again state-of-the-art hardware parameters of Table 1c. The difference in circuit execution time ΔT and the number of CZ gates ΔCZ are computed, where the latter corresponds to the eventually inserted and decomposed SWAP gates. Additionally, in both cases, a total average mapping fidelity is computed and compared taking the negative logarithm of the approximate success probability P ratio $\delta\mathcal{F} = -\log(P_{\text{mapped}}/P_{\text{original}})$ with a smaller value representing a smaller fidelity decrease caused by the mapping process. The results with the corresponding mapper runtimes (RT) in CPU seconds are listed in Table 1a with the full code and evaluation data available online [10].

4.2 Discussion

It is obvious that shuttling-only mapping results in $\Delta\text{CZ} = 0$, as no additional gates are added. For gate-based mapping, on the other hand, the inserted SWAP gates are decomposed and, therefore, result in a higher ΔCZ, while the time overhead ΔT is magnitudes of order smaller compared to the slower shuttling-only mapping. Nevertheless, for the (1) shuttling-optimized hardware it is still favorable to utilize the slow shuttling compared to the error-prone CZ gates, due to the long coherence times. Similarly, for the (2) gate-optimized hardware with improved CZ fidelity, SWAP gate insertion represents the preferred mapping capability as expected. The proposed hybrid mapper directly utilizes the hardware parameters to decide between the two capabilities and can, therefore, correctly identify the more suitable strategy, resulting in the best output in all shown cases. Moreover, the full flexibility of the proposed hybrid approach is demonstrated in the last row of Table 1a, corresponding to the (3) mixed hardware setup, representing reasonable operation fidelities for near-term devices. Instead of using the same computational capability through the whole mapping process, the hybrid

Table 1: Mapping Results for different NA hardware settings with different compilation strategies

(a) Mapping results

Hardware Settings			Compiler Settings: (A) Shuttling-based only ΔCZ	(A) ΔT [µs]	(A) $\delta\mathcal{F}$	(A) RT [s]	(B) Gate-based only ΔCZ	(B) ΔT [µs]	(B) $\delta\mathcal{F}$	(B) RT [s]	(C) Proposed Hybrid Approach ΔCZ	(C) ΔT [µs]	(C) $\delta\mathcal{F}$	(C) RT [s]	α
(1) Shuttling		*graph*	**0**	6.3	**0.45**	4.9	1086	**0.1**	7.37	4.9	**0**	6.3	**0.45**	4.9	
		qft	**0**	357.0	**25.20**	60.6	12141	**5.4**	88.47	60.7	**0**	357.0	**25.20**	60.4	
		qpe	**0**	313.5	**22.79**	61.7	11928	**3.5**	87.23	61.4	**0**	313.5	**22.79**	61.7	
		bn	**0**	11.0	**1.21**	0.3	522	**0.1**	4.06	0.3	**0**	11.0	**1.21**	0.3	
		call	**0**	16.6	**1.27**	35.8	1146	**0.3**	8.10	35.5	**0**	16.6	**1.27**	35.5	
		gray	**0**	5.8	**0.44**	0.5	483	**0.1**	3.34	0.5	**0**	5.8	**0.44**	0.5	1
(2) Gate		*graph*	**0**	11.8	0.94	2.8	324	**0.0**	**0.10**	0.4	324	**0.0**	**0.10**	0.4	
		qft	**0**	465.7	32.15	70.2	2733	**2.3**	**0.96**	17.4	2733	**2.3**	**0.96**	17.4	
		qpe	**0**	563.4	39.80	50.2	3021	**2.3**	**1.05**	17.4	3021	**2.3**	**1.05**	17.5	
		bn	**0**	29.9	2.12	0.6	132	**0.0**	**0.04**	0.1	132	**0.0**	**0.04**	0.1	
		call	**0**	28.9	2.07	28.2	309	**0.1**	**0.10**	26.4	309	**0.1**	**0.10**	26.4	
		gray	**0**	9.2	0.64	0.2	72	**0.0**	**0.02**	0.0	72	**0.0**	**0.02**	0.0	
(3) Mixed		*graph*	**0**	9.9	**0.59**	4.6	846	**0.1**	2.54	0.4	**0**	9.9	**0.59**	4.6	0.95
		qft	**0**	607.3	36.08	67.9	8898	**4.4**	27.24	18.3	6867	54.1	**24.09**	24.0	1.04
		qpe	**0**	613.7	36.51	59.6	7980	**2.9**	24.41	20.1	6987	26.9	**22.82**	22.9	1.06
		bn	**0**	21.6	1.28	0.3	492	**0.2**	1.47	0.1	78	10.0	**0.83**	0.2	1.01
		call	**0**	34.3	2.04	37.9	693	**0.2**	2.09	35.0	363	11.4	**1.77**	36.3	1.01
		gray	**0**	8.6	0.52	0.5	285	**0.1**	0.86	0.1	66	4.5	**0.46**	0.2	0.99

(b) Benchmark descriptions

Name	n	nCZ	nC_2Z	nC_3Z
graph	200	215	0	0
qft	200	9998	0	0
qpe	200	10340	0	0
bn	48	133	87	0
call	25	0	192	56
gray	33	0	62	0

(c) Hardware settings

Paramters	Shuttle	Gate	Mixed
$r_{\text{int}} = r_{\text{rest}}$	2	4.5	2.5
$\mathcal{F}_{U_3}$	0.995	0.9999	0.999 [14]
$\mathcal{F}_{CZ}$	0.994	0.9995	0.995 [9]
$\mathcal{F}_{\text{Shuttling}}$	1[5]	0.999	0.9999
t_{U_3} [µs]	0.5 [14]		
t_{CZ} [µs]	0.2 [9]		
t_{CCZ} [µs]	0.4		
t_{CCCZ} [µs]	0.6		
v [µm µs^{-1}]	0.55 [5]	0.2	0.3
$t_{\text{act/deact}}$ [µs]	20 [4]	50	40
T1 [µs]	100000000 [9]		
T2 [µs]	1500000 [9]		

ΔCZ and ΔT represent the difference in the number of CZ gates and circuit execution time, respectively. $\delta\mathcal{F}$ measures the relative fidelity decrease, taking the negative logarithm of the approximate success probability (less is better). RT is the runtime of the mapping process in CPU seconds, $\alpha = \alpha_g/\alpha_s$ is the ratio between gate- and shuttling-based mapping.

mapper can choose the most suitable way to map each gate within the circuit, resulting in a combination of SWAP gates and shuttling moves. By leveraging both mapping capabilities, the hybrid mapper can reduce the fidelity decrease $\delta\mathcal{F}$ caused by the mapping for all considered benchmarks except for the graph-state preparation. Here, the mapper correctly identifies the shuttling-only approach to still be the best choice. Note that the optimal ratio α between gate- and shuttling-mapping varies for different circuits, indicating a connection between circuit structure and preferred mapping capability. The proposed hybrid mapper allows, for the first time, to study this correlation, with a systematic study left for future work.

5 Conclusions

In this work, we proposed a hybrid compilation methodology, incorporating both gate-based SWAP insertion and shuttling-based qubit array rearrangements, i.e., both mapping capabilities of the Neutral Atom (NA) platform. We addressed challenges arising from the simultaneous utilization of both mapping capabilities and integrated corresponding solutions into an overall hybrid compilation process, inclusive of direct support for multi-qubit gates. The versatility of the proposed approach has been demonstrated across diverse hardware configurations, showcasing its adaptability and potential for enhancing fidelity. By applying the proposed mapping process to quantum circuits featuring up to 200 qubits, we have shown its effectiveness in harnessing the complete spectrum of computational capabilities provided by NAs, encompassing high-fidelity long-range interactions, native multi-qubit gate support, and qubit shuttling. This opens new possibilities for future studies and compiler development for the NA platform.

Acknowledgments

The authors thank Johannes Zeiher for fruitful discussions regarding Neutral Atoms and Tom Peham for helpful *C++* advice.

L.S. and R.W. acknowledge funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program, grant agreement No. 101001318, and MILLENION, grant agreement No. 101114305. This work was part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus.

References

[1] S. Adarsh et al. SyReC Synthesizer: An MQT tool for synthesis of reversible circuits. *Software Impacts*, 2022.
[2] J. M. Baker et al. Exploiting Long-Distance Interactions and Tolerating Atom Loss in Neutral Atom Quantum Architectures. In *ACM/IEEE Int'l Symposium on Computer Architecture*, pages 818–831, 2021. DOI: 10.1109/ISCA52012.2021.00069.
[3] D. Barredo et al. An atom-by-atom assembler of defect-free arbitrary two-dimensional atomic arrays. *Science*, 354(6315):1021–1023, 2016. DOI: 10.1126/science.aah3778.
[4] J. Beugnon et al. Two-dimensional transport and transfer of a single atomic qubit in optical tweezers. *Nature Physics*, 3(10):696–699, 2007. DOI: 10.1038/nphys698.
[5] D. Bluvstein et al. A quantum processor based on coherent transport of entangled atom arrays. *Nature*, 604(7906):451–456, 2022. DOI: 10.1038/s41586-022-04592-6.
[6] D. Bluvstein et al. Logical quantum processor based on reconfigurable atom arrays. *Nature*:1–3, 2023. DOI: 10.1038/s41586-023-06927-3.
[7] S. Brandhofer et al. Optimal Mapping for Near-Term Quantum Architectures based on Rydberg Atoms. In *Int'l Conf. on CAD*, pages 1–7, 2021. DOI: 10.1109/ICCAD51958.2021.9643490.
[8] A. Cowtan et al. On the qubit routing problem. In W. van Dam et al., editors, *Theory of quantum computation, communication and cryptography*, 2019.
[9] S. J. Evered et al. High-fidelity parallel entangling gates on a neutral-atom quantum computer. *Nature*, 622(7982):268–272, 2023. DOI: 10.1038/s41586-023-06481-y.
[10] Full code and evaluation data. URL: https://github.com/lsschmid/mqt-qmap/tree/hybrid-mapper.
[11] T. M. Graham et al. Multi-qubit entanglement and algorithms on a neutral-atom quantum computer. *Nature*, 604(7906):457–462, 2022. DOI: 10.1038/s41586-022-04603-6.
[12] F. Gyger et al. Continuous operation of large-scale atom arrays in optical lattices, 2024. arXiv: 2402.04994.
[13] L. Henriet et al. Quantum computing with neutral atoms. *Quantum*, 4:327, 2020. DOI: 10.22331/q-2020-09-21-327.
[14] H. Levine et al. Dispersive optical systems for scalable Raman driving of hyperfine qubits. *Physical Review A*, 105(3):032618, 2022. DOI: 10.1103/PhysRevA.105.032618.
[15] H. Levine et al. Parallel Implementation of High-Fidelity Multiqubit Gates with Neutral Atoms. *Physical Review Letters*, 123(17):170503, 2019. DOI: 10.1103/PhysRevLett.123.170503.
[16] G. Li et al. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. In *Int'l Conf. On Architectural Support for Programming Languages and Operating Systems*, 2019. DOI: 10.1145/3297858.3304023.
[17] Y. Li et al. Timing-Aware Qubit Mapping and Gate Scheduling Adapted to Neutral Atom Quantum Computing. *IEEE Trans. on CAD of Integrated Circuits and Systems*:1–1, 2023. DOI: 10.1109/TCAD.2023.3261244.
[18] M. A. Norcia et al. Iterative assembly of yb atom arrays in cavity-enhanced optical lattices, 2024. arXiv: 2401.16177.
[19] N. Nottingham et al. Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures, 2023. arXiv: 2307.14996.
[20] S. Park et al. A fast and scalable qubit-mapping method for noisy intermediate-scale quantum computers. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, pages 13–18, New York, NY, USA. Association for Computing Machinery, Aug. 2022. DOI: 10.1145/3489517.3530402.
[21] T. Patel et al. Geyser: a compilation framework for quantum computing with neutral atoms. In *Int'l Symposium on Computer Architecture*, pages 383–395, 2022. DOI: 10.1145/3470496.3527428.
[22] L. Pause et al. Supercharged two-dimensional tweezer array with more than 1000 atomic qubits. *Optica*, 11(2):222–226, Feb. 2024. DOI: 10.1364/OPTICA.513551.
[23] T. Peham et al. On Optimal Subarchitectures for Quantum Circuit Mapping. *ACM Transactions on Quantum Computing*, 4(4):23:1–23:20, 2023. DOI: 10.1145/3593594.
[24] N. Quetschlich et al. MQT Bench: benchmarking software and design automation tools for quantum computing. *Quantum*, 2023.
[25] M. Saffman. Quantum computing with atomic qubits and Rydberg interactions: progress and challenges. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 49(20):202001, 2016. DOI: 10.1088/0953-4075/49/20/202001.
[26] M. Saffman et al. Quantum information with Rydberg atoms. *Reviews of Modern Physics*, 82(3):2313–2363, 2010. DOI: 10.1103/RevModPhys.82.2313.
[27] L. Schmid et al. Computational capabilities and compiler development for neutral atom quantum processors—connecting tool developers and hardware experts. *Quantum Science and Technology*, 9(3):033001, Apr. 2024. DOI: 10.1088/2058-9565/ad33ac.
[28] B. Tan et al. Optimal Layout Synthesis for Quantum Computing. In *Int'l Conf. on CAD*, pages 1–9, 2020.
[29] D. B. Tan et al. Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors. *Quantum*, 8:1281, Mar. 2024. DOI: 10.22331/q-2024-03-14-1281.
[30] D. B. Tan et al. Depth-Optimal Addressing of 2D Qubit Array with 1D Controls Based on Exact Binary Matrix Factorization, 2024. arXiv: 2401.13807.
[31] H. Wang et al. FPQA-C: A Compilation Framework for Field Programmable Qubit Array, 2023. arXiv: 2311.15123.
[32] H. Wang et al. Q-Pilot: Field Programmable Quantum Array Compilation with Flying Ancillas, 2023. arXiv: 2311.16190.
[33] R. Wille et al. Mapping Quantum Circuits to IBM QX Architectures Using the Minimal Number of SWAP and H Operations. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, pages 1–6, New York, NY, USA. Association for Computing Machinery, 2019. DOI: 10.1145/3316781.3317859.
[34] A. Zulehner et al. An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2019. DOI: 10.1109/TCAD.2018.2846658.