Politecnico di Torino

Politecnico di Torino

Master's degree in Quantum Engineering

A.a. 2025/2026

Graduation Session March 2026

Compilation Strategies for Neutral Atom Quantum Architectures: A Comparative Evaluation

Supervisors:

Prof. Bartolomeo Montrucchio

Dr. Paolo Viviani

Dr. Giacomo Vitali

Candidate:

Pio Antonio Campana

Abstract

Neutral atom quantum computing has emerged as a promising platform for scalable quantum computation due to qubit uniformity, long coherence times, native long-range Rydberg interactions, and all-to-all connectivity enabled by atom shuttling. However, as system sizes grow, complex compilation strategies must be implemented to optimise qubit placement, routing, and scheduling while taking into account physical constraints such as restriction zones, transport limitations, and architecture-dependent connectivity. Managing such a complex scenario requires trade-offs in the compilation design to fully exploit the inherent parallelism of the architecture.

This thesis analyses compilation frameworks for neutral atom quantum computers across three architectural paradigms: static arrays, dynamically field-programmable qubit arrays (DPQAs), and zoned architectures. For each model, the placement, routing, and scheduling heuristics are examined and evaluated based on relevant metrics for Noisy Intermediate-Scale Quantum (NISQ) era: circuit fidelity, duration and compilation time.

Static array systems represent standard quantum computing technology, with a fixed set of atoms that interact based on the topology. DPQA systems allow for the reconfiguration of the topology during circuit execution thanks to atom shuttling, thus avoiding the need for additional SWAP gates. Zoned architecture is an evolution of the DPQA system, featuring separate zones for storage, entanglement, and readout.

The first contribution of this thesis is a unified framework for evaluating compilation methods in different neutral atom quantum computing architectures. Given an architecture, a compilation method, and a circuit, the framework enables a systematic and architecture-aware performance assessment. Focusing on zoned architecture, a bottleneck was identified: the serial back-and-forth movement between storage zones and entanglement zones between one Rydberg stage and another, causing an increase in circuit duration. Two solutions were proposed to address this issue. The first involves implementing a hybrid version of zoned architecture and static arrays, allowing for partial in-place interactions upon returning to the storage zone. The second solution involves implementing a combination of zoned architecture and DPQA, which allows both intra-zone and inter-zone movements to be exploited.

Table of Contents

List of Figures

Part I

Introduction

Chapter 1

Quantum computation models

Despite the exponential growth in computing power witnessed in recent years, various fields of research have demonstrated the intrinsic limitations of classical computing. This has necessitated the exploration of novel computing paradigms that facilitate the circumvention of these limitations. Quantum computing, a rapidly expanding domain of scientific research, is an example of this. It aims to overcome certain limitations by leveraging the principles of quantum mechanics. This computing paradigm was first proposed in 1982 by the physicist Richard Feynman [1]. The proposal aimed to simulate quantum systems with high accuracy, given that simulating such complex interactions is a substantial challenge for conventional computing systems, exhibiting exponential complexity. The simulation of quantum systems is imperative for advancements in domains such as physics, chemistry, and materials science. Subsequent to Feynman's address, quantum computing came into existence, and over the years, a plethora of computing models and architectures have been proposed.

Intensive research in this field soon led to the discovery that other problems, beyond the simulation of quantum systems, could also be solved more efficiently using this computational model. Notable examples of this are the algorithms proposed by scientists Lov Grover and Peter Shor. In fact, Grover's algorithm [2] was instrumental in facilitating the enhancement of unordered database search, thereby transitioning from $O(n)$ complexity to $O(\sqrt{n})$ complexity.

Conversely, Shor's algorithm [3] proposed a solution to the prime factorisation problem that is polynomial in complexity, thereby significantly outperforming the optimal classical solution, which necessitated exponential complexity. This algorithm attracted considerable interest due to the fact that the factorisation problem was employed for the protection of some cryptographic keys, and a rapid

resolution to this problem would have exposed a considerable proportion of the IT security of the era.

Despite the demonstration of the fundamental principles of these algorithms, their practical application remains hindered by the absence of a suitable platform for what is termed "utility-scale quantum computing". As previously stated, several hardware platforms have been proposed over the years, with considerable financial resources being allocated in the hope that one of these platforms would emerge as the successful solution to address real-world problems. After four decades of intensive research, the optimal hardware platform remains difficult to identify,as the management of quantum systems gives rise to several challenges. These systems exhibit a pronounced interaction with environmental noise, resulting in information decoherence, whereby information undergoes unwanted alterations over time, impeding the execution of operations with precision.

Among the most promising platforms today are superconducting qubits, trapped ions, neutral atoms, and photonic qubits. Each of these architectures has its own strengths, but also limitations. As previously stated, there is no predominant architecture, with numerous companies allocating substantial resources to their own initiatives. These efforts are directed towards the pursuit of quantum supremacy, defined as the point at which quantum computers will exceed the performance of classical computers in specific operations.

The present thesis will focus on neutral-atom quantum computing, and in particular on the problem of its circuit compilation, as the neutral-atom platform has a unique feature that needs to be managed and whose research is still in its infancy: atom shuttling.

The utilisation of neutral atom platforms has been demonstrated to be a highly effective solution for the construction of medium and large-scale quantum platforms. This is largely attributable to their architectural flexibility, long coherence times, high-fidelity operations, and straightforward control. The distinguishing feature of this platform is its all-to-all connectivity, whereby each qubit within the platform is capable of direct interaction with all others, thanks to atom shuttling. Shuttling is defined as the process of moving atoms within the platform for the purpose of bringing them closer together during computation operations, thereby enabling interaction between them. While this is a valuable asset, it must be managed meticulously to ensure optimal efficiency in circuit operations.

1.1 Digital quantum computing

The digital computation model, also known as gate-based model, represents the theoretical paradigm of reference for the design and analysis of quantum algorithms. This model was first formulated by David Deutsch, who in 1985 introduced the

concept of the universal quantum computer, in which the unitary evolution of quantum systems was exploited to perform computations in a more general way than the classical paradigm [4].

In this model, information is encoded in two-level quantum systems, called qubits, whose state is described by a normalised vector in a two-dimensional complex Hilbert space

$$\mathcal{H}_1 = \mathbb{C}^2. \tag{1.1}$$

A pure state of a single qubit can be expressed as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \qquad \alpha, \beta \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 = 1, \tag{1.2}$$

where $\{|0\rangle\,, |1\rangle$ represents the computational basis, used for initialising the system and for extracting the classical result at the end of the computation [5].

A quantum register composed of N qubits is described by the tensor Hilbert space

$$\mathcal{H}_N = \bigotimes_{i=1}^{N} \mathcal{H}_1 = (\mathbb{C}^2)^{\otimes N}, \tag{1.3}$$

The states belonging to this space can exhibit entanglement, a quantum feature that allows for non-separable correlations between qubits and is crucial in the computational advantage of quantum systems [5].

In the circuit model, the evolution of the system is described as a discrete sequence of unitary transformations, called quantum gates, applied to the qubit register. Formally, a gate acting on k qubits is represented by a unitary operator

$$U \in \mathrm{U}(2^k), \tag{1.4}$$

and the overall evolution of the state is given by

$$|\psi_{\text{out}}\rangle = U_m \cdots U_2 U_1\,|\psi_{\text{in}}\rangle\,, \tag{1.5}$$

where the order of the operators reflects the temporal structure of the quantum circuit. Single-qubit gates transform the state of a single qubit, while two- or multi-qubit gates allows the interaction of two or more of them [5].

As demonstrated by Barenco et al. in [6], any unitary transformation on the Hilbert space $\mathcal{H}_N$ can be approximated with arbitrary precision by the composition of a finite number of gates drawn from a universal set. In a neutral-atom quantum computer, a universally complete native gateset consists of arbitrary rotations on individual qubits and a Controlled-Z gate based on the Rydberg blockade, a concept that will be explored later.

At the end of the unitary evolution, the output of the quantum computation is obtained by a projective measurement of the qubits in the computational basis.

The measurement is described by a set of projection operators $\{|x\rangle\langle x|\}$, with $x \in \{0,1\}^N$, and the classical result x is observed with probability

$$P(x) = |\langle x|\psi_{\text{out}}\rangle|^2. \tag{1.6}$$

Due to the inherently probabilistic nature of the measurement, the execution of a quantum algorithm demands the circuit to be iterated a large number of times in order to estimate the probability distribution associated with the final state [5].

This model generates a conceptual separation between the algorithmic and physical levels, and the study of quantum compilation problems, in which an ideal circuit must be translated into a sequence of operations compatible with the constraints and imperfections of real hardware, becomes necessary.

1.2 Analog quantum computing

The analog quantum computing model is based on the direct exploitation of the natural dynamics of a physical quantum system to solve computational problems. Computation is achieved through a continuous evolution in time governed by a suitably engineered Hamiltonian. In this context, the algorithm is encoded directly in the structure of the system's Hamiltonian and its physical parameters.

Formally, the temporal evolution of the quantum state $|\psi(t)\rangle$ is described by the time-dependent Schrödinger equation [5]

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H(t)|\psi(t)\rangle, \tag{1.7}$$

where $H(t)$ represents the Hamiltonian of the system, which can be constant or vary slowly over time. The computation consists of preparing the system in a known initial state $|\psi(0)\rangle$ and letting it evolve under the action of $H(t)$ until a final time T, at the end of which the state $|\psi(T)\rangle$ encodes the solution to the problem.

A sub-case of analog computation is the model of quantum annealing and, more generally, adiabatic quantum computing [7]. In this scheme, the Hamiltonian of the system is expressed as an interpolation between an initial Hamiltonian H_0, whose ground state is easily prepared, and a final Hamiltonian H_P, which encodes the problem of interest:

$$H(t) = A(t)H_0 + B(t)H_P, \tag{1.8}$$

where the functions $A(t)$ and $B(t)$ vary slowly over time so as to satisfy $A(0) \gg B(0)$ and $A(T) \ll B(T)$. If the evolution is sufficiently slow with respect to the minimum energy gap between the ground state and the excited states, the adiabatic theorem guarantees that the system remains in its ground state, which at the final time coincides with the solution of the problem [8].

In the analog model, information is encoded in the energy levels of the system and in spin configurations, rather than in qubits manipulated individually by means of logic gates. The interactions are determined directly by the physical Hamiltonian, which may include many-body terms and long-range couplings that are difficult to simulate efficiently with digital circuits. This feature makes the analog paradigm well suited to the quantum simulation of complex physical systems, such as strongly correlated spin models or many-body dynamics [9].

Beyond optimization and quantum simulation, the analog paradigm has also been shown to be capable of reproducing the behavior of canonical quantum algorithms originally formulated within the gate-based model. A relevant example is the analog formulation of Grover's search algorithm. In this context, Farhi and Gutmann proposed a continuous-time quantum walk–based analog of Grover's algorithm, in which the search problem is encoded directly into a problem Hamiltonian and the system evolves continuously under its action [10]. They demonstrated that the marked state can be found in a time scaling as $O(\sqrt{N})$, as envisaged in the original work by Grover.

Aharonov et al. proved that any quantum circuit can be efficiently simulated by an adiabatic evolution with only polynomial overhead in time and resources [11]. This result establishes the polynomial equivalence between the two paradigms and implies that adiabatic quantum computation is universal in the computational complexity sense.

Chapter 2

Quantum hardware platforms

In 1996, David DiVincenzo formulated five necessary conditions to construct a quantum computer [12]. Going through these conditions will bring out the strengths and weaknesses of proposed technologies for quantum computing throughout the years.

1. A scalable physical system with well-characterized qubits: Not all physical systems can be qubits, since we cannot isolate two specific quantum states and manipulate them. Moreover, we need these systems to scale well to high numbers

2. Qubit initialization to a fiducial state: It should always be possible to prepare the qubits in the same initial state

3. Long relevant coherence times: Quantum information should last long enough to perform relevant operations

4. A universal set of quantum gates: A combination of single and two-qubit gates is needed to perform arbitrary operations

5. A qubit-specific measurement capability: In order to read the result of an algorithm, an accurate measurement of the final state is needed

Dr. Chris Monroe experimentally demonstrated the first historical quantum logic gate in 1995 by using trapped ions as qubits [13]. Thirty years have passed since then, but the recipe for the best quantum computer has yet to be found; however, some technologies emerged as strongly promising.

2.1 Superconducting technology

The first and probably most relevant technology that we will discuss is the superconducting qubit, a physical qubit implemented using superconducting electrical circuits based on Josephson junctions. This technology was experimentally demonstrated in 1999 [14], and research in this area is strongly supported by major industrial players such as Google and IBM, who consider superconducting platforms a leading candidate for large-scale quantum computing.

Superconductivity is a quantum phenomenon exhibited by certain materials below a critical temperature, where electrical resistance vanishes, and charge carriers form bound states known as Cooper pairs [15]. This property enables the realization of superconducting circuits. The key nonlinear element of superconducting qubits is the Josephson junction, an element composed of two superconductors separated by a thin insulating barrier, predicted by Brian Josephson in 1962 [16]. A Josephson junction allows coherent tunneling of Cooper pairs and introduces the anharmonicity required to isolate two energy levels.

In Figure 2.1, we can see three simple implementations of a superconducting qubit: in the charge qubit, the qubit states correspond to different numbers of Cooper pairs localized on a small superconducting island; in the flux qubit, quantum information is encoded in the direction of persistent currents circulating in a superconducting loop, which generate opposite magnetic fluxes; the phase qubit exploits the quantum dynamics of the superconducting phase difference across a current-biased Josephson junction, with the computational states associated with quantized energy levels in a tilted washboard potential [17].

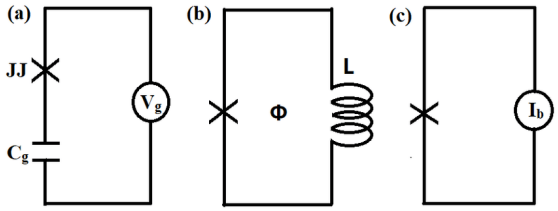

Figure 2.1: Superconducting qubit circuit diagrams. (a) Charge qubit. (b) Flux qubit. (c) Phase qubit. [17]

Superconducting technology is currently the most extensively researched and developed platform in the quantum computing landscape, owing to a combination of technological maturity and favorable engineering properties. One of its main strengths is scalability: the number of superconducting qubits can be increased using established microfabrication techniques, as demonstrated by IBM with the Condor processor comprising 1,121 qubits [18]. In addition, superconducting qubits can be strongly and controllably coupled, enabling reliable entangling operations.

Logic gates in superconducting architectures are relatively fast, with typical execution times on the order of tens of nanoseconds, allowing the implementation of deep circuits within the coherence time of the qubits. High gate fidelities can be achieved thanks to precise microwave control and optimized circuit design. Moreover, superconducting devices are compatible with conventional semiconductor technologies, facilitating the integration of control and readout electronics [19].

At the same time, this architecture presents several limitations. Qubit operation requires extreme cryogenic conditions, with temperatures close to absolute zero needed to maintain the superconducting state. Coherence times, although sufficient for many near-term applications, remain limited to the millisecond scale and are strongly affected by environmental noise [19].

From the perspective of quantum compilation, an additional constraint is imposed by the restricted qubit connectivity: two-qubit gates are only available between physically coupled qubits according to the hardware coupling map. As a result, compiling abstract quantum circuits often requires the insertion of additional routing operations, increasing circuit depth and negatively impacting the overall fidelity.

2.2 Trapped ion technology

This architecture relies on the possibility of holding an ion still inside a static or dynamic electromagnetic field, thanks to its electrical charge. Trapped ion quantum computers are composed of several ions organized in a crystal-like structure, thanks to the equilibrium between the confining potential and Coulomb repulsion between ions. The key technology is the Paul trap [20], which uses radio-frequency electric fields oscillating at 10-100 MHz to create a time-averaged quadrupole potential that confines ions along three spatial dimensions.

To implement the two-state system, the internal electronic states of single ions are used. Usually, two types of implementations are used: optical qubits, in which the two states are the ground state and an excited state; or hyperfine qubits, where the states are two ground states of different hyperfine levels. Quantum operations, both one-qubit and two-qubit, are applied by using accurately controlled laser impulses, which allow for manipulating internal states of the ions and coupling different qubits through collective vibrational modes of the ionic crystal [21].

Trapped ion technology features several advantages. First and foremost, it is characterized by extremely long coherence times, in the order of several minutes, one of the longest among all the quantum computing platforms. Quantum gates implemented in these systems display high fidelities. A strength point is the relative ease in the preparation of the fiducial state of the qubits, which can be reliably obtained with cooling and optical pumping techniques [21].

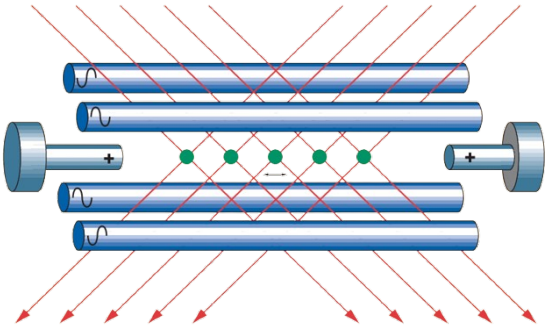

Figure 2.2: Schematic diagram of a 5-qubit trapped ion quantum computer. Laser beams stabilize ions and perform quantum operations on them [22].

Trapped ion quantum computers also exhibit important limitations. In particular, the execution times of gates are relatively slow, typically in the order of some microseconds; when raising the number of trapped ions, some issues related to controlling vibrational modes and selectively addressing single qubits arise, resulting in a difficult extension of the architecture to quantum registers with greater dimensions.

2.3 Photonic technology

The photonic qubit hardware platform differs significantly from the static qubit implementations described so far. It encodes quantum information in the different degrees of freedom of photons, particularly phase, polarisation and optical path. These photons are generated by optical sources, such as nonlinear crystals, each time a computation begins, and are then manipulated by passive and active optical elements, such as beam splitters, electro-optical modulators and phase shifters [23].

In this architecture, single-qubit gates can be achieved with ease. For instance, polarisation can be modified using half- and quarter-wave plates. Thanks to the maturity of photonics, the fidelity of these operations is very high. However, two-qubit operations are the main bottleneck of this technology as they are complex and probabilistic due to the fact that photons do not interact directly. The most commonly used protocol to create effective non-linearities is KLM. The CNOT gate is implemented using heralded Bell state measurement with a success probability of 1/9, which can be increased by employing additional ancilla photons and feed-forward techniques at the cost of increased resource overhead [23].

One of the main advantages of photonic technology is the high coherence, because photons weakly interact with the environment. This allows for preserving quantum information even over long distances, making photonic qubits particularly

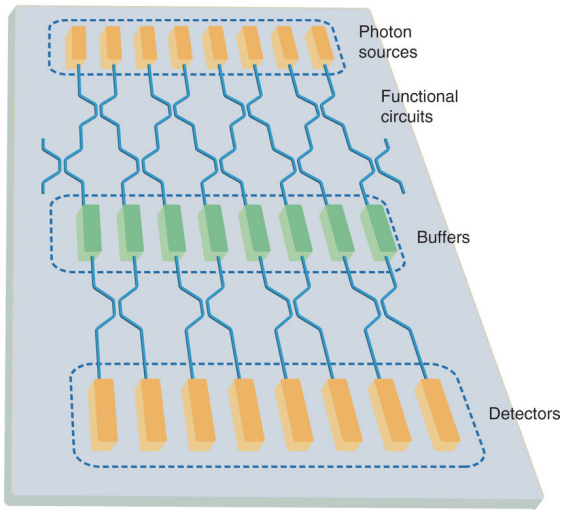

Figure 2.3: Conceptual diagram of a photonic quantum processor [24].

well-suited for applications of quantum information and entanglement distribution. Additionally, this technology is operated at room temperature, so no difficult control is needed. Photonic systems are also very fast, because operations are limited mainly to the propagation of optical signals [23].

The implementation of quantum computers based on photonic qubits also exhibits significant challenges: as previously discussed, the main issue resides in the implementation of two-qubit gates, given its probabilistic nature. Moreover, single-photon management is difficult, photons are lost, and large-scale integration of photonic components is difficult.

Chapter 3

Neutral atom technology

3.1 Qubit encoding

The encoding of quantum information in neutral atom computers is based on the use of internal electronic states, typically encoded in the levels of the hyperfine structure of the ground state of isotopes such as Rubidium-87 (^{87}Rb) or Strontium-88 (^{88}Sr) [25]. The computational basis is defined by the states $|0\rangle$ and $|1\rangle$, often chosen as "clock" states (e.g. $|F = 1, m_F = 0\rangle$ and $|F = 2, m_F = 0\rangle$ for ^{87}Rb) due to their insensitivity to external magnetic fields [25]. The quality of information storage in these states is limited by relaxation (T_1) and dephasing (T_2) times, which can reach the order of several seconds. To quantify the coherence limit of the system during processing, an effective coherence time (T_{eff}) parameter is defined, expressed by the formula:

$$T_{eff} = \frac{T_1 T_2}{T_1 + T_2} \tag{3.1}$$

The transitions for single-qubit gates are controlled by laser pulses that drive Rabi oscillations between $|0\rangle$ and $|1\rangle$. The dynamics of this manipulation are described by the single qubit control Hamiltonian:

$$\frac{H_1(t)}{\hbar} = \frac{\Omega(t)}{2} |0\rangle \langle 1| + \frac{\Omega^*(t)}{2} |1\rangle \langle 0| - \Delta(t) |1\rangle \langle 1| \tag{3.2}$$

where $\Omega(t)$ represents the effective Rabi frequency and $\Delta(t)$ the detuning of the laser with respect to atomic resonance [25].

To implement multi-qubit operations and generate entanglement, the system exploits temporary excitation towards Rydberg states $|r\rangle$, characterised by a high principal quantum number and strong dipole interactions. The interaction between two atoms in such states is dominated by the van der Waals potential (V_{vdW}),

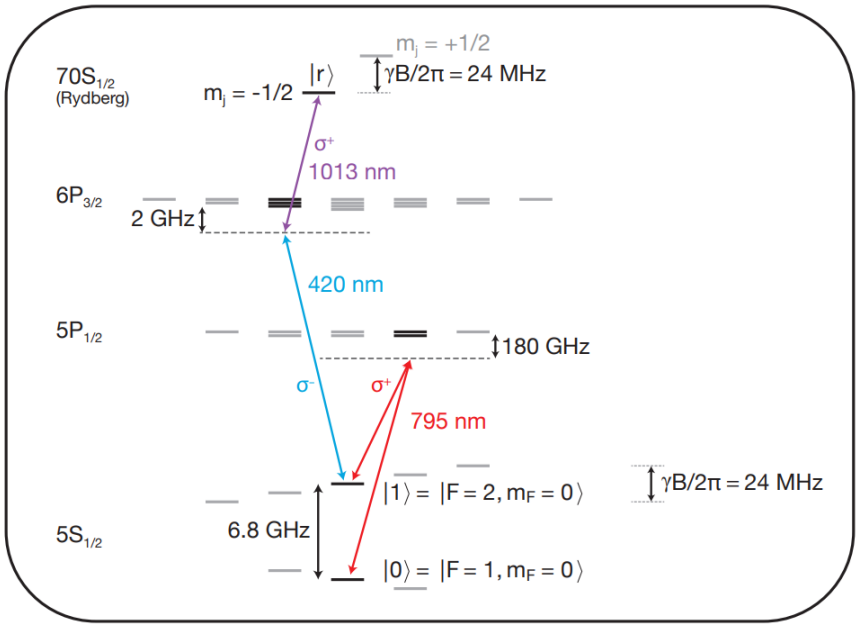

Figure 3.1: Level diagram showing key ^{87}Rb atomic levels used [26].

which scales with the sixth power of the distance d:

$$V_{vdW}(d) = \frac{C_6}{d^6} \tag{3.3}$$

This interaction introduces an energy penalty that prevents the simultaneous excitation of neighbouring atoms, a phenomenon known as Rydberg blockade [25]. The maximum distance within which this phenomenon is effective is defined by the blocking radius (r_b):

$$r_b \simeq \left(\frac{C_6}{\hbar\Omega_r}\right)^{1/6} \tag{3.4}$$

The evolution of a pair of qubits under the influence of a global Rydberg laser is governed by the following two-atom transition Hamiltonian:

$$\frac{H_2(t)}{\hbar} = \sum_{i=1}^{2}\left(\frac{\Omega_r(t)}{2}\left|1\right\rangle\left\langle r\right|_i + \frac{\Omega_r^*(t)}{2}\left|r\right\rangle\left\langle 1\right|_i - \Delta_r(t)\left|r\right\rangle\left\langle r\right|_i\right) + \frac{V_{vdW}}{\hbar}\left|r\right\rangle\left\langle r\right|_1 \otimes \left|r\right\rangle\left\langle r\right|_2 \tag{3.5}$$

By exploiting this dynamic, it is possible to implement universal logic gates such as the Controlled-Z (CZ) with high fidelity, taking advantage of the fact that the state $|11\rangle$ acquires a different phase compared to the other states of the basis due to the interaction V_{vdW} [25].

3.2 Optical traps: tweezers, SLM, AOD

The confinement and spatial manipulation of atoms in the quantum register are achieved using arrays of optical tweezers generated by far-off resonant lasers (usually between 810 nm and 830 nm) focused through customised high numerical aperture lenses [26]. Each trap has a diffraction-limited size (waist ≈ 0.9–$1\,\mu$m) and a potential depth U proportional to the laser intensity, with radial oscillation frequencies ω_0 reaching $2\pi \times 80$ kHz [26].

The creation of arbitrary geometries is entrusted to a Spatial Light Modulator (SLM), which acts as a programmable phase modulator. The SLM imprints a phase pattern $\phi(x, y)$ on the laser beam, which is converted into a specific intensity pattern in the focal plane of the lens, allowing the register to be scaled up to hundreds or thousands of qubits. To overcome the probabilistic loading limit (approximately 50%), the system uses 2D crossed Acousto-Optic Deflectors (AOD) to create mobile tweezers. These deflectors, controlled by radiofrequency (RF) signals generated by AWGs, allow atoms to be rearranged in real time to obtain defect-free arrays with unitary filling [26, 27].

To preserve information during transport, the trajectories of the AODs follow a cubic interpolation profile, which allows for movements 5-10 times faster than constant velocity profiles without inducing significant heating. The condition of adiabaticity necessary to avoid vibrational excitations requires that the acceleration a of the trap satisfies the relation:

$$ma\sigma \ll \hbar\omega_0$$

where m is the mass of the atom and σ is the extension of the qubit wave function in the ground state [28]. The efficiency of transport is quantified by the increase in the average vibrational quantum number ΔN after a displacement over a distance D in a time T, calculated as:

$$\Delta N = \frac{1}{2}\left(\frac{6D}{x_{zpf}\omega_0^2 T^2}\right)^2$$

where $x_{zpf} = \sqrt{\hbar/(2m\omega_0)}$ represents the size of the zero-point. Thanks to this dynamic control, it is possible to transport entangled qubits over distances greater than $110\,\mu$m while maintaining the fidelity of the state, enabling non-local and programmable connectivity within the processor [26].

3.3 AOD constraints

The dynamic movement of atoms in neutral atom processors is made possible by the use of Acousto-Optic Deflectors (AODs), which generate a series of laser beams

oriented along the x and y axes that intersect to create a grid of programmable optical traps in a 2D plane. Manipulation is achieved by modulating the frequencies of the RF signals sent to the AODs, allowing atoms to be moved between functional zones (such as from the storage zone to the entanglement zone) or to reconfigure the register's connectivity in real time [25]. However, the parallel movement of multiple qubits is subject to strict physical and geometric constraints arising from the nature of the AOD grid:

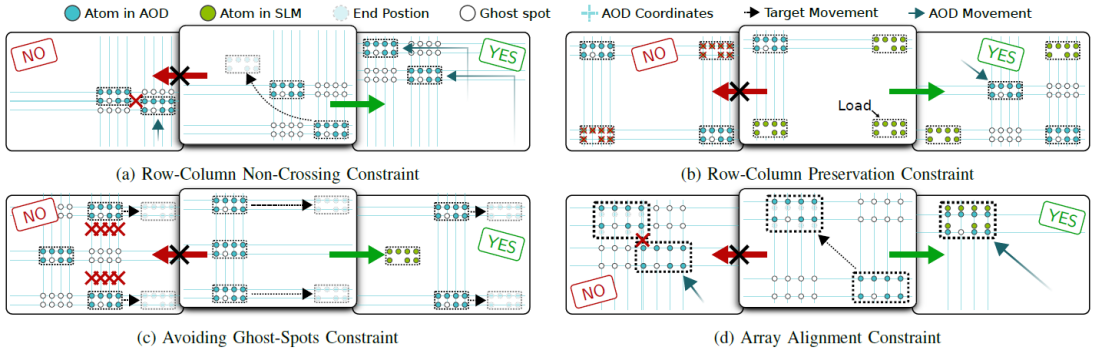

Figure 3.2: AOD constraints for shuttling operation [29].

- Preservation of Rows and Columns: Since each row (or column) is generated by a single control signal, all atoms sharing the same x or y coordinate must move simultaneously with the same displacement; it is therefore not possible to independently move two atoms located on the same row in different directions without moving the entire row [25, 29].
- Non-Crossing Constraint: The rows and columns of AODs must maintain a minimum safety distance (d_{min}) and can never cross each other to avoid overlapping trap potentials, which would result in the loss of atoms. This implies that the relative spatial order of the atoms captured in a single AOD must remain unchanged throughout transport [25, 29].
- Ghost Spots: Each intersection between active rows and columns potentially creates a trap; therefore, the system must be programmed to avoid so-called ghost spots, i.e. unwanted intersections that could accidentally capture or disturb atoms that should remain isolated [25, 29].
- Array Alignment: To execute logic gates that require the overlap of two groups of atoms (arrays), the system must avoid violating the non-crossing constraint by moving at least one of the two groups into static traps generated by a Spatial Light Modulator (SLM), which are not subject to the movement limitations of AODs [25, 29].

To maximise parallelism and reduce time overhead, advanced routing algorithms calculate trajectories that respect these constraints by attempting to move as many qubits as possible simultaneously to their interaction zones.

3.4 Qubit array architectures

Several spatial architectures have been proposed for neutral-atom quantum computers, each exploiting a different feature of the technology. In the following, the architectures will be divided into three main groups that operate in different ways.

3.4.1 Fixed atom array

The fixed atom array (FAA) architecture represents the traditional paradigm of neutral atom quantum computing, in which qubits are confined to static spatial positions within a regular grid generated by a SLM [25]. In this setup, single-qubit local operations are fundamental and are performed by directing highly focused laser beams onto individual atoms in the register. These pulses allow two-photon Raman transitions to be driven to achieve arbitrary rotations on the Bloch sphere with extremely high fidelity, exceeding 99.5% for single-qubit operations [27]. However, the fidelity of local two-qubit operations has not yet reached competitive percentages, settling at around 92.5% [30], a value that is not in line with state-of-the-art technologies.

Moreover, the connectivity of an FAA system is limited by the Rydberg interaction radius (r_{int}), i.e., the maximum distance within which the Rydberg blockade can mediate native logic gates between two or more atoms [25]. To establish entanglement between qubits located beyond this radius, the architecture must implement virtual SWAP gates. This method involves a high computational overhead, significantly increasing the total number of gates and the depth of the circuit, thus exposing the system to higher error and decoherence rates than architectures with dynamic connectivity [27].

3.4.2 Dynamically Field-Programmable Qubit Array

The Dynamically Field-Programmable Qubit Array (DPQA) architecture represents an evolution over static grid systems (FAA), introducing dynamic connectivity through coherent transport of atoms [26].

The DPQA model solves the gate overhead limitation by eliminating the need for virtual SWAPs thanks to the physical shuttling capability of atoms, manipulated by mobile optical tweezers controlled by AODs in combination with static traps generated by SLMs. Computation in a DPQA processor occurs through iterative cycles of loading, spatial reconfiguration of the array, storing, and gate application,

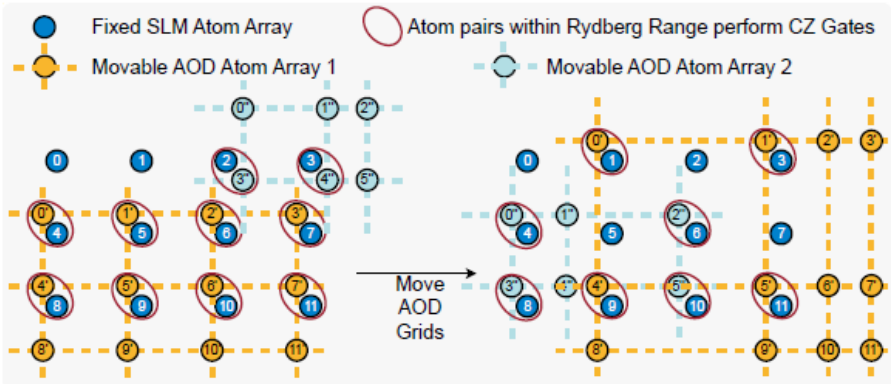

Figure 3.3: DPQA with fixed and movable atoms [31].

allowing qubits that have never interacted directly to be coupled without adding extra logic gates [25, 26]. This architecture enables highly parallel operations on entire rows or columns of atoms, optimising the execution of complex algorithms and the preparation of topological states. The coherent transport of atoms is able to preserve entanglement and quantum phase at distances greater than 100 μm in times significantly shorter than the coherence times of the system, allowing several reconfiguration steps during the execution of a quantum circuit [26].

Also, the DPQA model partially solves the problem relating to the low-fidelity local two-qubit operation: atoms can be arranged in a way that only interacting qubits stand in each other's blockade radius. This allows for the application of the higher-fidelity global two-qubit operation. However, this operation is still non-ideal because the global beam also affects non-interacting qubits, sometimes causing unwanted operations.

3.4.3 Zoned architecture

Zone architecture represents an advanced paradigm for quantum computing with neutral atoms, designed to overcome the connectivity and decoherence limitations of static lattice systems by spatially dividing the processor into distinct functional regions, such as the register zone for information storage, the interaction zone for gate execution, and the readout zone for measurement [26, 29]. The coherent transport of atoms between these areas is achieved by means of mobile optical tweezers controlled by AODs [32].

This configuration offers superior coherence protection, as the qubits remain shielded from the Rydberg and reading laser beams in the storage area, allowing effective coherence times (T_{eff}) to be maintained that can exceed one second [26]. A further key advantage is all-to-all dynamic connectivity: the physical displacement of entangled atoms eliminates the need to insert virtual SWAP gates, drastically

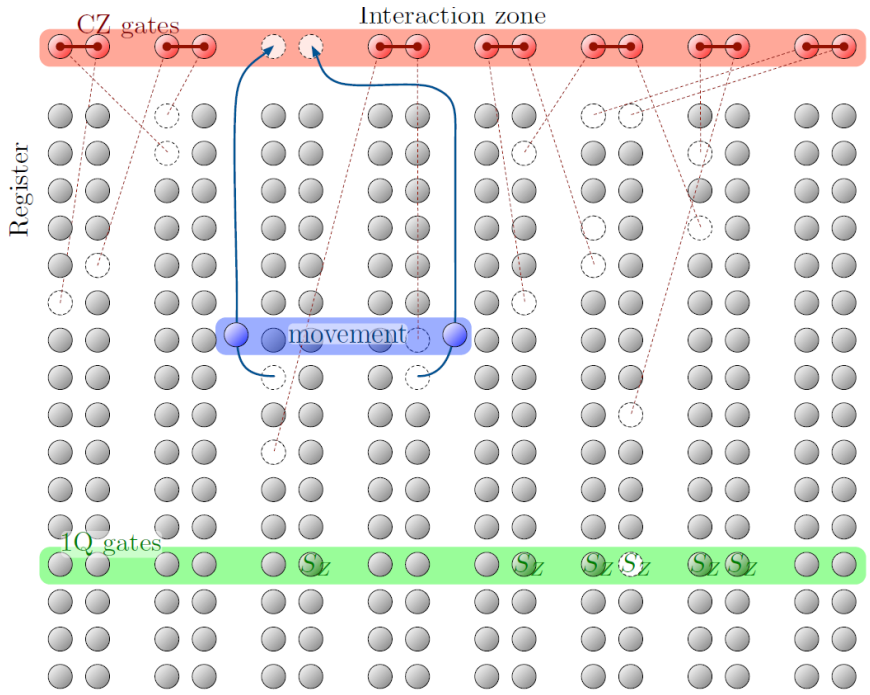

Figure 3.4: Experimental architecture: interaction zone on top, storage zone on bottom [32].

reducing circuit overhead and depth. Within the interaction zone, the use of global Rydberg pulses enables the execution of parallel multi-qubit gates [32]. Finally, the functional separation of the zones is crucial for Quantum Error Correction and mid-circuit measurements, as errors generated during gates in the IZ can be converted into atomic losses (erasure conversion) and detected in the readout zone without disturbing the data qubits in the register [32].

Chapter 4

Quantum compilation

Quantum compilation is the process by which an algorithm is transformed from an abstract, hardware-independent form into a description that can be executed on real hardware [25]. Transpilation plays a fundamental role in this process: it involves transforming one quantum circuit into another that is semantically equivalent, net of a global phase, while remaining compatible with the constraints imposed by the specific hardware platform on which the circuit will be executed [33]. The main constraints taken into account during transpilation are the native gate set and the connectivity topology of the architecture. Therefore, transpilation can be considered the primary tool for adapting the algorithmic level to the physical level of quantum computing.

The transpilation process follows three well-defined phases:

The first is synthesis or decomposition, where the abstract operations of the logic gates are rewritten in terms of a finite set of native gates supported by the hardware platform. This process introduces decompositions of individual gates, as well as the global rewriting of subcircuits, to achieve more efficient sections.

The second phase is mapping, which involves assigning logical qubits in the circuit to physical qubits in the hardware while taking the qubits' connectivity and noise characteristics into account.

The third phase is routing. This is necessary because direct interaction is only permitted between specific pairs of qubits; therefore, the initial mapping is not always sufficient to guarantee the executability of the circuit. This phase enables additional operations to be inserted to position the qubits required for multi-qubit gates adjacent to each other. This is often achieved using SWAP gates, which enable the logical exchange of the states of two qubits. While these operations preserve the circuit's functional correctness, they increase its depth and the number of gates [25].

In the Noisy Intermediate-Scale Quantum (NISQ) era, transpilation aims to produce an executable circuit and maximise the fidelity of the operations. This

is why the process must be noise-aware. The produced circuit must minimise the number of operations and its depth while adapting the transformations to the noise characteristics: in particular, it must maximise the fidelity of operations on the least noisy qubits.

Following transpilation, quantum compilation includes additional steps. In particular, there is temporal scheduling of operations in an attempt to make the most of the parallel nature of this computing paradigm. There is also a final translation to lower-level representations, such as pulse sequences [25].

The distinction between compilation and transpilation is particularly relevant when dealing with different hardware platforms, especially the neutral-atom platform, where the system's dynamics and the modes of interaction between qubits introduce new constraints and degrees of freedom.

Below, we will discuss the main transpilation methods used in Qiskit, an SDK developed by IBM for its quantum computers and the primary programming tool in modern quantum computing [33].

4.1 Circuit synthesis

The synthesis phase is the first step in transpilation, representing the translation of generic quantum circuit operations into those compatible with the target device.

This process operates on multiple levels of complexity:

First, the composite operations are expanded in terms of elementary instructions, thus transforming the circuit into a canonical form. Then, the gates not natively supported by the circuit are translated into a combination of native gates using symbolic rewriting rules or synthesis procedures based on matrix representation.

Alongside this functional translation, the synthesis phase integrates a series of optimisations to simplify the circuit structure. Since every physical operation is subject to noise and imperfections, this step is crucial and has a direct effect on the execution fidelity.

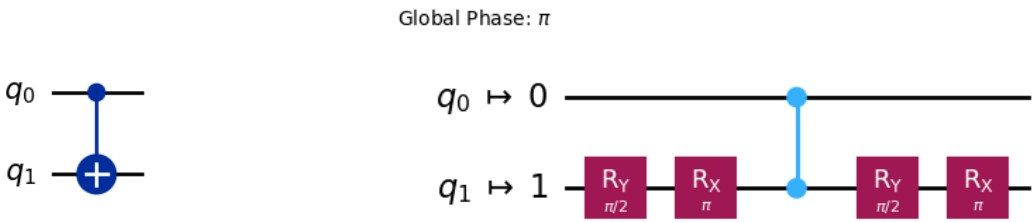

Figure 4.1: A CNOT gate and its decomposition in RX, RY, CZ gates to match neutral-atom platform constraints.

The main optimisation operations include merging logically equivalent single-qubit operation sequences, eliminating successive inverse or redundant operations, and exploiting the commutativity of certain gates to reduce circuit depth. These optimisations are applied iteratively until the circuit is as efficient as possible in terms of the total number of gates [33].

Synthesis should be understood as a trade-off between the accuracy of the unit representation and the physical cost of implementation.

4.2 Qubit mapping

The second step of transpilation is mapping. This involves assigning the logical qubits in the circuit to the physical qubits in the hardware architecture. The connectivity topology, noise, and coherence parameters obtained from calibration information are taken into account during this process.

In Qiskit, this step corresponds to the layout stage, in which an initial layout is calculated based on the selected method. Examples of algorithms include: VF2Layout, which searches for an isomorphism between the interaction graph of two qubits in the circuit and the physical connectivity graph. DenseLayout selects physical qubits with a high degree of connectivity and places the qubits that interact the most in the circuit on them. Finally, NoiseAdaptiveLayout uses calibration information to assign the most frequently used logical qubits to physical qubits with better noise parameters and coherence times [33].

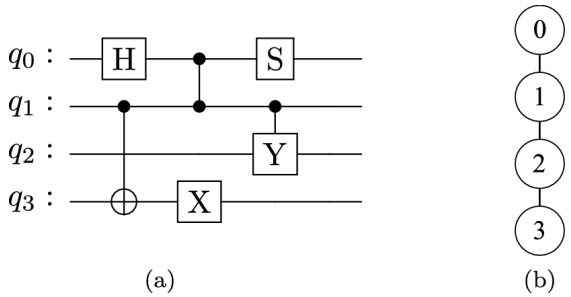

Figure 4.2: Representation of the mapping problem. Logical qubits $\{q_0, q_1, q_2, q_3\}$ in the circuit in (a) need to be mapped to physical qubits topology {0,1,2,3} in (b) [34].

At the end of this phase, an initial qubit layout is obtained that does not completely eliminate the possibility of connectivity problems. This problem will be addressed in the next step.

4.3 Qubit routing

The final step in transpilation is routing, whereby the circuit is modified to ensure compatibility with the platform topology and enable all multi-qubit operations. This is necessary because, as discussed in Section 1.2, some hardware has limited connectivity, meaning direct interaction between certain qubits is not permitted. Swap gates are inserted to bring the qubits involved in the required operations closer together, solving this problem.

Unfortunately, finding a routing that minimises the number of these additional gates is an NP-hard problem, so it is infeasible to solve this for large circuits, and sub-optimal solutions must be explored instead. Qiskit uses heuristic algorithms for swap insertion. The SABRE algorithm, which is used as the default routing method, is especially significant. It iteratively evaluates possible swap insertions and selects those that most effectively reduce the distance required for multi-qubit interactions, while balancing local and future constraints [33].

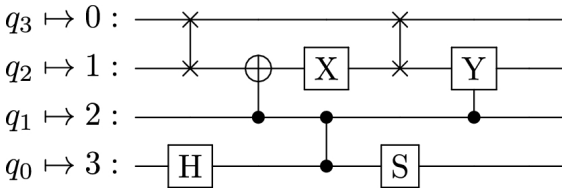

Figure 4.3: Representation of the routing problem. On the left of the circuit, the qubit mapping of Figure 4.2 is present. With this mapping, the interaction (q_1, q_3) is not possible, so a SWAP gate is inserted [34].

From a practical standpoint, the complexity of routing and the high cost of swap gates in the architecture mean that this phase accounts for most of the overhead in compilation, as each additional gate increases the total depth and exposure to errors.

4.4 Limits of Qiskit transpiler for neutral-atom platform

The Qiskit transpiler is a mature and effective tool for compiling quantum circuits, particularly on superconducting architectures characterised by static connectivity topology. However, the architectural assumptions underlying its compilation model are only partially compatible with the peculiarities of neutral-atom quantum computers, highlighting a conceptual discrepancy between the software model adopted and the underlying physics of such platforms.

In Qiskit, connectivity between qubits is modelled as a static graph, where nodes represent physical qubits and edges indicate directly realizable interactions. In this context, routing is implemented exclusively by inserting SWAP gates, which allow the quantum state of pairs of adjacent qubits in the graph to be exchanged. This approach assumes that the movement of qubits is a logical operation, without an explicit spatial counterpart. Although this assumption is adequate for platforms where qubits are physically fixed, it is limiting in the case of neutral atoms, where qubits are physical entities that are mobile in space and can be transported in a controlled manner using shuttling techniques.

In neutral-atom quantum computers, connectivity is not a static constraint but a dynamic resource that can be reconfigured during circuit execution. Multi-qubit operations, such as gates based on Rydberg interaction, require the atoms involved to be positioned within a specific spatial distance, but this condition can be satisfied by physically transporting the atoms rather than through a sequence of logical operations equivalent to SWAP. The routing model adopted by Qiskit does not capture this possibility, as it does not distinguish between the physical cost of a logical exchange operation and that of a spatial movement operation, nor does it allow for explicit optimisation of the trajectory and timing of qubit transport.

A further limitation lies in the fact that Qiskit's transpiler does not natively integrate continuous constraints related to space and time, such as the two-dimensional or three-dimensional geometry of optical traps, the maximum transport speed of atoms, or interference between simultaneous shuttling and laser control operations. As a result, the optimisation performed by the transpiler is confined to a purely discrete and combinatorial level, whereas compilation for neutral atom platforms requires co-design between the logical and physical levels, in which the spatial arrangement of atoms and the temporal sequence of operations play a fundamental role.

Given these premises, it is clear that mapping, routing, and scheduling problems must be considered altogether in order to best capture every degree of freedom in the architecture.

These considerations highlight how the direct use of general-purpose compilation tools, such as Qiskit's transpiler, is not sufficient to fully exploit the potential of neutral-atom architectures. In particular, the absence of an explicit model of shuttling and dynamic reconfigurability limits the ability to optimise the circuit and can lead to suboptimal solutions in terms of depth, fidelity, and scalability. These limitations motivate the need to develop dedicated compilation strategies that can natively incorporate the physical movement of qubits as a computational resource, and form the starting point for the work presented in the following chapters of this thesis.

Part II

Compilation frameworks for neutral-atom quantum computers

Chapter 5

Objectives and scope of the comparative analysis

The potential of neutral atoms as a platform for quantum computing discussed in the previous chapter has strongly driven research into its development. Several hardware architectures have been proposed, each associated with a specific compilation strategy involving physical constraints, connectivity models, and operational capabilities. This diversity does not allow, as in the case of superconducting architectures, the creation of a standard compilation model dependent solely on the connectivity map, but is constrained by the intrinsic mechanisms of the architecture: atom mobility, interaction mechanisms, and spatial organisation.

Given these assumptions, the objective of this chapter will be to provide a comparative study of the main state-of-the-art compilation frameworks for neutral atom quantum computing, with a specific emphasis on mapping, routing, and scheduling heuristics.

Unlike the standard quantum computing model, which has fixed-position qubit structures and predefined interactions, with a compilation model set by the IBM Qiskit framework, neutral atom systems are divided into a range of architectures that differ between two-dimensional static arrays, dynamically reconfigurable arrays, and zone architectures with rest and interaction zones.

Each of these paradigms introduces specific problems to be solved that must be handled by the compilation model, including: limitations imposed by the Rydberg blockade mechanism, constraints on parallel operations, costs arising from atom movement, and transport channel management. Consequently, given the wide range of problems to be managed, considerable reliance is placed on heuristic optimisations capable of navigating this very large search space while maintaining the necessary computational efficiency.

Throughout the analysis, the strategies are divided by architecture, and the

main algorithmic components of each, such as layout selection, qubit assignment, routing strategies, or scheduling policies, are identified. Special focus will be given to the methods that each architecture uses to mitigate or exploit specific hardware characteristics. In addition, the chapter will also highlight the limitations of current methodologies by looking for possible problems that can be solved with further development.

Despite growing interest and various proposals for compilation frameworks, there is currently no adequate means of comparing alternative heuristics; in fact, existing work evaluates performance using heterogeneous benchmarks, hardware-specific assumptions, or narrow metrics, making it difficult to select the ideal method for the architecture. This fragmentation motivates the necessity for a thorough assessment from an external perspective that can compare compilation approaches under consistent conditions and across different architectural models.

The detailed survey presented in this chapter is necessary to lay the conceptual and methodological foundations for building a structured evaluation framework presented in Part III. The analysis will also identify recurring bottlenecks, such as movement congestion, limited interaction zones, and scheduling conflicts that emerge across different architectures and significantly affect the computational and executive cost of compiled circuits.

5.1 Classification of neutral atom compilation architectures

Platforms for neutral atom quantum computing can be categorised based on the degree of spatial configurability of the atomic register and the possible mechanisms available to enable two-qubit interactions. When it comes to compilation, this classification is crucial because constraints such as available connectivity, movement operations, and available operations determine the complexity of mapping, routing, and scheduling strategies.

The first class of architecture is represented by static neutral atom arrays. In this architecture, atoms are positioned in a fixed 1-, 2-, or 3-dimensional geometry. As explained in Part I, the interaction between two qubits is mediated by the Rydberg blockade mechanism, so two-qubit gates are only possible between atoms positioned at a certain distance; consequently, it is possible to construct an effective connectivity graph determined by the geometric layout of the array, which cannot be modified during computation. For this type of architecture, compilation is well standardised considering that the problem can be traced back to that of compilation on superconducting computers, but an additional rule applies: in the case of Rydberg interactions, two operations can conflict if they are too close, and therefore these operations must be serialised.

The second class, defined as dynamically field-programmable qubit arrays (DPQA), allows the rearrangement of atoms during computation through controlled movements of optical tweezers. This possibility allows the interaction graph to be modified, achieving effective all-to-all connectivity. This allows interaction between distant qubits without the operational overhead of inserting swap gates. However, this movement of atoms introduces additional costs in terms of time and possible new sources of error. Compilation frameworks based on this architecture must include a movement-aware heuristic, capable of balancing the benefits of complete connectivity against the overhead introduced by movement operations. This architecture allows one of the problems of the basic architecture to be overcome, namely conflict zones for interactions; in fact, interacting pairs of atoms can be positioned at a distance that allows all possible operations to be performed in parallel. To take full advantage of this parallelism, a global radius is used for two-qubit operations: although this is more accurate than local operations, it still has drawbacks due to the possibility that atoms not participating in any interaction may still be excited, causing an error.

The third and final architecture is a direct evolution of the DPQA and is called the zone architecture. This name is due to the fact that the architecture is divided into: storage zones, where atoms that do not participate in two-qubit operations reside; entanglement zones, where atoms that will interact are moved; and readout zones, used for measurements. This division is designed to solve the problem of unintended excitations: the qubits that remain in storage will be protected from the global beam, but this will require greater computational and movement costs.

The rest of this chapter will describe the state-of-the-art compilation methods for each of the architectures described, indicating how specific hardware constraints modify the design of compilation workflows and influence the performance achievable by compiled quantum circuits.

Chapter 6

Heuristic compilation for fixed atom arrays

6.1 Interaction model and physical constraints

In fixed atom arrays, neutral atom qubits are arranged in two-dimensional grids defined by optical trapping potentials generated by optical tweezers. The hardware topology can be formalized as a graph $G = (V, E)$, where V represents physical qubit locations and an edge $(u, v) \in E$ exists if the Euclidean distance satisfies

$$d(u, v) \leq d_{max}.$$

d_{max} is a fixed parameter describing the maximal length at which two qubits can interact. When atoms are excited to Rydberg states during a two-qubit interaction, they induce a restriction zone around them within which additional excitations are physically forbidden [35]. Adopting the model

$$f(d) = \frac{d}{2},$$

parallel operations can happen only if their restriction zones do not overlap.

This geometric exclusion introduces a non-trivial trade-off between connectivity and parallelism. While a larger d_{max} reduces SWAP requirements, it simultaneously enlarges potential blockade overlap, increasing serialization. As a consequence, a balance between connectivity and serialization must be found during the compilation process [35].

6.2 Compilation pipeline and heuristic strategies

The compilation flow for fixed neutral atom arrays consists of three stages: mapping, routing, and scheduling. The objective is to minimize circuit depth while respecting

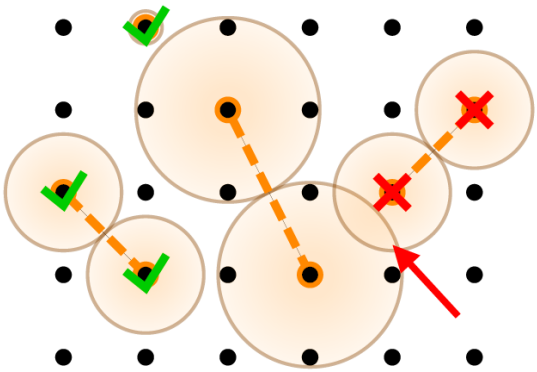

Figure 6.1: Examples of interactions on a neutral atom device. Gates can happen in parallel if their zones do not intersect [35]

restriction-zone constraints [35].

6.2.1 Lookahead-driven mapping

The mapping problem is solved through a lookahead weighting function defined over the circuit's directed acyclic graph (DAG), to prioritize near-term interactions without neglecting future dependencies. We separate the interactions into layers, whose operations can, in principle, be executed in parallel. The weight for an interaction between qubits u and v is defined as

$$w(u,v) = \sum_{l \geq l_c} e^{-|l-l_c|},$$

where l denotes the layer of the interaction and l_c is the current frontier layer. This exponentially decaying function ensures that imminent interactions receive higher priority, while still taking into account information about future operations [35].

The initial placement score for assigning qubit u to hardware location h is then computed as

$$s(u,h) = \sum_{v} d(h, \phi(v)) \cdot w(u,v),$$

where $\phi(v)$ denotes the physical position of already mapped qubits. In this formulation, the most operated qubits are positioned centrally to form a highly-interacting section with reduced routing overhead [35].

6.2.2 Routing with victim-aware scoring

When the distance between two qubits that have to interact exceeds d_{max}, SWAP gates are inserted. The routing strategy proposed incorporates a value for the target qubit's movement, and a value for the impact of the swapped qubit's displacement, defined as the victim qubit [35].

The routing score for swapping logical qubit u into hardware position h is given by

$$s(u,h) = \sum_v \Big[d(\phi(u),\phi(v)) - d(h,\phi(v))\Big] w(u,v) + \sum_v \Big[d(h,\phi(v)) - d(\phi(u),\phi(v))\Big] w(\phi^{-1}(h),v).$$

The first term measures the improvement in interaction distance for the target qubit. The second term penalizes excessive displacement of the victim qubit relative to its future interactions. This construction allows for preserving global efficiency and prevents local routing decisions from degrading mapping quality [35].

6.2.3 Scheduling Under restriction zones

Scheduling must ensure that simultaneously executed gates do not violate restriction zones. In this case, parallelism is determined not only by data dependencies but also by spatial compatibility. Gates are grouped into executable layers only if their corresponding restriction regions remain disjoint [35].

This constraint transforms scheduling into a geometric compatibility problem combined with the conventional dependency problem.

6.3 Performance implications of long-range connectivity

Conducted evaluations demonstrate a pronounced reduction in SWAP count as d_{max} increases. However, this benefit exhibits a rapid saturation effect. For most circuits, a moderate interaction radius, typically a value of 3 or 4, achieves near-optimal communication overhead reduction [35].

Given that a high connectivity in fixed atom arrays comes with the downside of serialization, this solution may be unsuitable for some circuits. In fact, highly parallel circuits experience depth inflation when restriction zones enforce serialization. In contrast, inherently serial circuits are minimally affected [35].

The availability of native multi-qubit gates provides an additional structural advantage. Direct implementation of the Toffoli gate reduces gate count by approximately a factor of six compared to its standard decompositions, an advantage

that can be exploited in hardware-aware compilation strategies.

Chapter 7

Dynamically Field-Programmable Qubit Arrays (DPQA)

The Dynamically Field-Programmable Qubit Arrays (DPQA) architecture represents an alternative approach for neutral atom quantum computing (NAQC hardware, exploiting the possibility of low-error atom movement. This paradigm actually shifts the perspective of the compilation process: the circuit is not adapted to the fixed hardware architecture, rather the architecture is adapted during execution to the circuit's necessities. Routing is no longer entrusted to SWAP gates, but it involves actual atomic transport.

The possibility to put atom close together allows us to restrict the Rydberg blockade radius, to incorporate only the necessary qubits within each other's interaction zone. This enables the utilisation of the higher-fidelity global pulse for two-qubit interactions, resulting in lower errors and higher parallelisation. This, therefore, requires that all participating atoms must be arranged before each interaction layer. However, the global nature of the pulse implies that idling qubits are also exposed to excitation noise, characterised by an excitation error probability f_{exc}. Minimising the number of such global Rydberg stages becomes one of the primary objectives of DPQA compilers.

Thus, in DPQA systems, layout synthesis is not merely a performance optimization but a necessary condition for correct and high-fidelity execution.

7.1 Hardware model: SLM and AOD traps

In DPQA, two trap configurations are used for storage and transport functionalities, employing two different optical subsystems:

Spatial Light Modulator (SLM) traps keep data qubits fixed in space. These traps, generated through static holographic light patterns, define a two-dimensional grid where qubits can remain idle without losing information for a long time.

Acousto-Optic Deflector (AOD) traps enable dynamic qubit movement. By steering acoustic waves, rows and columns of traps are translated across the plane, allowing atoms to be shuttled. The atom movement is not free of constraints, having to comply with four rules described in section 3.3. This results in a significant impact on the routing strategies.

7.2 Compilation frameworks for DPQA architectures

In DPQA systems, compilation plays a crucial role in enhancing circuit fidelity. Compilation strategies can be classified depending on how many movable atoms are available in the architecture. This difference will be clear in the next sections, where two complementary frameworks will be discussed. The first one is Q-Pilot compilation strategy, developed at MIT by Wang et al. [36], in which data qubits are fixed in SLM traps, and ancilla qubits are moved to enable two-qubit interactions. In the second one, ENOLA, in principle, all the qubits can switch from an SLM trap to an AOD trap and move, so that for each Rydberg stage, the shortest path to bring needed qubits together is chosen.

7.3 The Q-Pilot compiler

The core idea of Q-Pilot compilation strategy is the addition of movable ancillas inside the quantum computer architecture. These will serve as mediators for the long-range interactions between fixed-position data qubits [36].

Consider an n-qubit quantum state:

$$\Psi = \sum_{x=0}^{2^n-1} C_x \left|x\right\rangle .$$

Let C be a set of qubit pairs (j, j') on which controlled-Z gates must be applied. The state after parallel CZ application is:

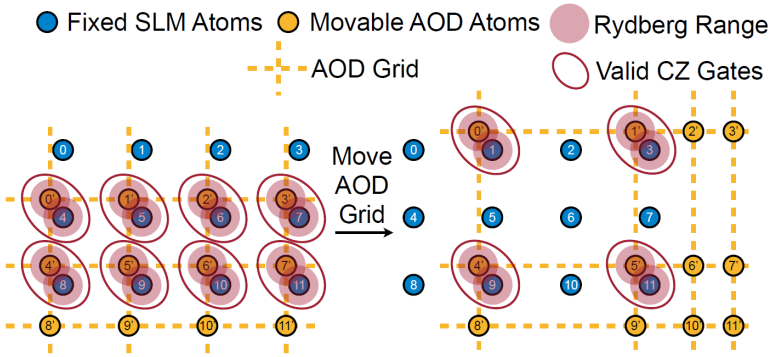

Figure 7.1: Dynamically Field-Programmable Qubit Arrays in Q-Pilot. While SLM (blue) atoms are fixed, AOD (yellow) atoms can be reconfigured during computation [36]

$$\Psi' = \sum_{x=0}^{2^n-1} C_x \prod_{(j,j')\in C} (-1)^{x_j x_{j'}} |x\rangle .$$

This is the fundamental two-qubit operation needed for quantum information processing in NAQC, and it is implemented in Q-Pilot through three phases [36]:

1. Ancilla creation: Ancillas, initialised to ground state $|0\rangle$ are initially placed next to control qubits, and a CNOT gate is applied, resulting in the state

   $$\Phi_1 = \sum_{x=0}^{2^n-1} C_x |\overline{xx}\rangle,$$

   In this way, the ancilla becomes entangled with the control qubit.

2. Interaction stage: Afterwards, ancillas are transported near target qubits, and CZ gates are applied.

3. Recycling: In the end, ancillas are brought back to control qubits and another CNOT is applied, in order to restore the $|0\rangle$ state, and obtain as a final state:

   $$\Phi_3 = \Psi' \otimes |0^n\rangle.$$

7.3.1 Generic router: Greedy scheduling under AOD constraints

The basic scheduling strategy for arbitrary circuits follows a greedy heuristic that complies with the non-crossing AOD constraints.

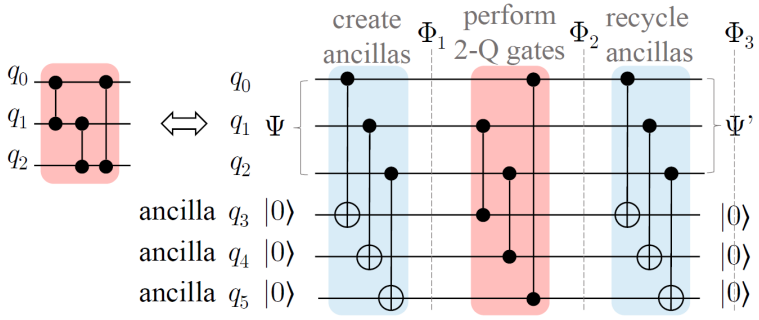

Figure 7.2: The three phases of ancilla routing in Q-Pilot [36]

The algorithm follows a predetermined sequence: First, a front layer of dependency-free CZ gates is extracted. The index of the first qubit sorts the gates; then each gate is tentatively added to the current stage, and the addition is accepted if the resulting AOD configuration complies with AOD constraints, otherwise the gate is deferred to a subsequent layer [36].

This construction allows for obtaining the maximal parallelism while ensuring compliance with the constraints.

7.3.2 Specific routing strategies

For Hamiltonian simulation the compilation objective changes. The goal becomes maximizing the fan-out from a selected root qubit.

A directed compatibility graph is constructed such that an edge exists from q_i to q_j if:

$$(q_j.row \geq q_i.row) \wedge (q_j.col \geq q_i.col).$$

In this way, the AOD movement preserves ordering constraints. Dynamic programming is then used to compute the longest path in this graph, which corresponds to the maximum number of gates that can be executed simultaneously while placing ancillas along a diagonal of the AOD grid, preserving maximum routing flexibility [36].

In QAOA circuits, interactions follow the graph structure of the cost Hamiltonian. Q-Pilot, therefore, attempts to saturate the AOD grid by maximizing legal SLM-AOD pairings per stage.

The heuristic proceeds by selecting the highest-priority interaction, then aligning compatible interactions along the same AOD row, and, in the end, searching vertically for configurations that maximize legal matches while avoiding spurious Rydberg interactions [36].

This strategy aims to execute as many graph edges as possible within a single global Rydberg pulse.

7.4 The Enola compiler

The Enola compiling strategy relies on the possibility of switching every data qubit from the SLM trap to the AOD trap, and vice versa. No additional ancilla qubits are required, thus all the atoms can be used to encode logical qubits.

7.4.1 Scheduling via edge coloring

Enola models scheduling as an edge-coloring problem on the interaction graph $G = (V, E)$, where vertices represent qubits and edges represent two-qubit gates [37].

The objective is to find a function:

$$\psi : E \to \mathbb{N}$$

such that if two edges share a vertex, they are assigned to different stages:

$$e \cap e' \neq \emptyset \Rightarrow \psi(e) \neq \psi(e').$$

Using the Misra-Gries edge-coloring algorithm, the number of stages S satisfies:

$$S \leq \chi'(G) + 1,$$

where $\chi'(G)$ is the chromatic index of G. Since $\chi'(G) \in \{\Delta(G), \Delta(G) + 1\}$, this guarantees near-optimal scheduling with polynomial complexity $O(n^3)$ [37].

7.4.2 Placement via fast simulated annealing

After scheduling, qubits must be mapped to physical sites that minimize transport throughout the circuit execution.

For this task, fast simulated annealing is adopted to minimize the following cost function:

$$\text{Cost} = \sum_{g(q,q') \in G} w_g \cdot \text{dist}(m(q), m(q')).$$

Dynamic weights are assigned as:

$$w_g = \max(0.1, 1 - 0.1 s_g),$$

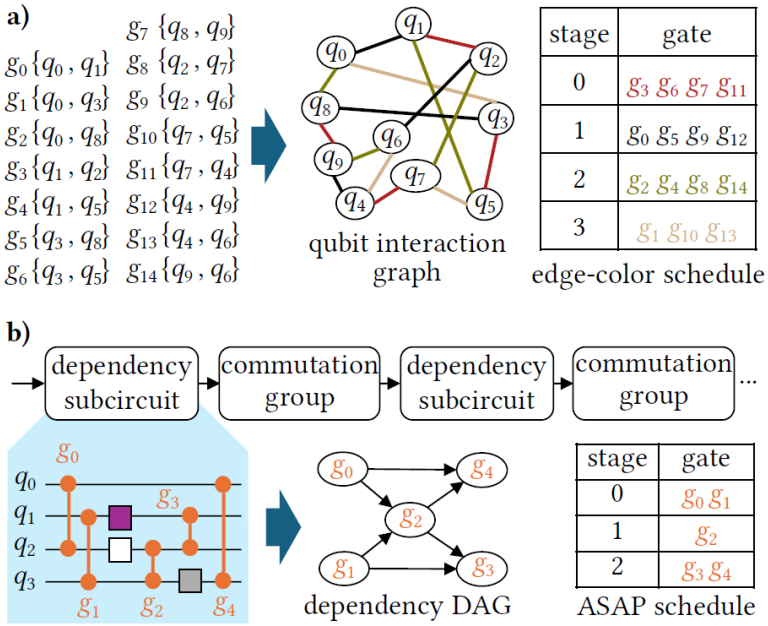

Figure 7.3: Scheduling in Enola. a) Edge coloring to schedule a group of commuting two-qubit gates. b) Division of generic circuits into dependency subcircuits and commutation groups [37]

where s_g is the number of preceding stages. The dynamic weights are constructed in such a way that imminent interactions are prioritized, while still taking into account the presence of future stages [37].

7.4.3 Routing via conflict graph and independent sets

Each movement in the architecture is represented as a tuple:

$$(src_x, src_y, dst_x, dst_y).$$

A conflict graph is created, posing the violations of AOD constraints as its edges. An example can be an ordering violation:

$$src_y(m) > src_y(m') \Rightarrow dst_y(m) > dst_y(m').$$

To parallelize movements, Enola extracts a Maximal Independent Set (MIS).

Two techniques are exploited to find the solution: The sortIS heuristic sorts movements by descending distance ($O(n \log n)$) and greedily builds the independent set ($O(n^2)$). For large systems, windowIS restricts analysis to the K longest movements. The total complexity is given by $O(n^2 \log n + n^2 K^2)$ [37].

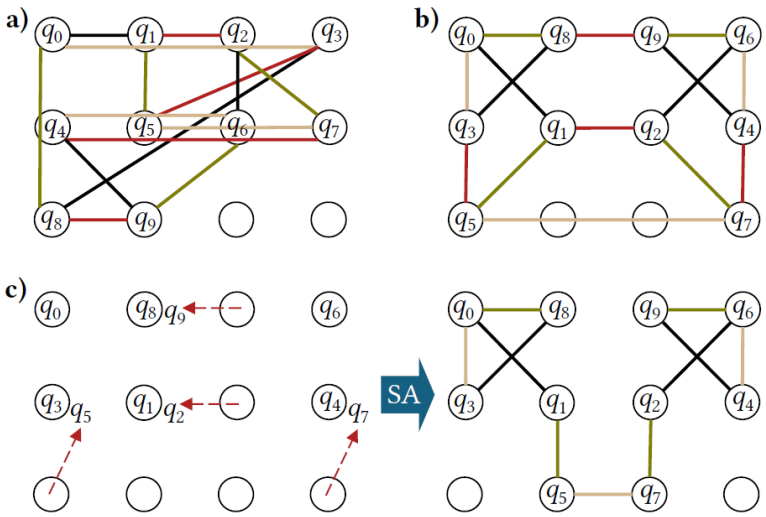

Figure 7.4: Placement in Enola. a) Trivial ordered placement. b) Optimized placement with SA based on distance. c) Dynamic placement (right) with SA after a Rydberg stage (left) is executed [37]

Chapter 8

Zoned architecture

Zoned neutral atom architectures represent an evolution of the monolithic DPQA systems, being divided into different operation zones. The idea of this division comes from the necessity to avoid unwanted excitations on idling qubits: in fact, in the DPQA architecture, since the CZ operations are obtained using a global beam, some non-participating qubits may be affected by excitation noise.

The proposed solution splits the architecture in a section where atoms interact, the entanglement zone, and a section where idle qubits stay, the storage zone. This allows to move the qubits of the current layer inside the entanglement zone and to shine the global beam only there, effectively shielding idle qubits in the storage zone, shifting the dominant error source from global laser exposure to atom transport.

However, this architectural advantage comes at the cost of increased compilation complexity: the search space is enlarged compared to DPQA systems. Placement, routing, and scheduling can no longer be treated independently.

Efficient exploitation of zoned architectures requires compilation strategies that explicitly account for transport constraints, parallel movement compatibility, and zone-specific resource allocation.

8.1 The NALAC compiler

Neutral Atom Logical Array Compiler (NALAC) is one of the first compilation solutions regarding zoned neutral atom architectures. NALAC aims to build a framework capable of exploiting quantum error correction (QEC) algorithms and achieving effective fault-tolerant quantum computing (FTQC) by treating arrays of atoms as a logical qubit and organising operations accordingly [29].

8.1.1 Logical Abstraction and Transversal Gates

In the FTQC paradigm, logical qubits are encoded across structured arrays of physical atoms. For example, in a Steane encoding, seven physical qubits arranged in a 2×4 layout form a single logical unit. Logical operations, particularly logical controlled-Z gates (CZ_L), are implemented transversally: each physical qubit in one array interacts with its corresponding partner in another array [29].

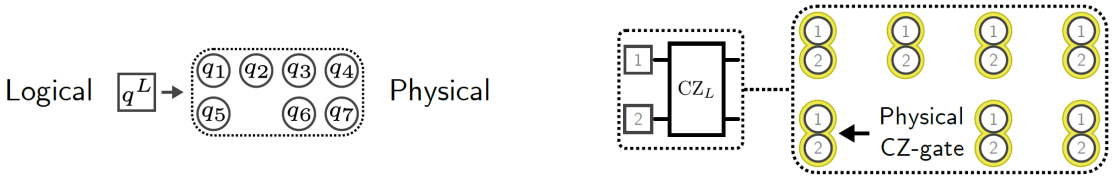

Figure 8.1: Logical qubit encoded across seven physical qubits in Steane code (left). Transversal logical Controlled-Z gate (right) [29]

Transversal gates are inherently fault-tolerant, as bitwise application prevents single physical errors from propagating across the logical block. However, in zoned architectures, executing CZ_L requires coordinated movement of entire logical arrays between storage and entangling zones. This transforms logical compilation into a constrained geometric scheduling problem.

NALAC, therefore, treats each logical array as an atomic routing entity, abstracted through a reference coordinate, drastically reducing the search space [29].

8.1.2 Abstract Shuttling Model and Mechanical Constraints

To translate logical operations into feasible physical movements, NALAC introduces an abstract model of the shuttling cycle:

1. **Loading**: Transfer of the atom from SLM traps to AOD traps.
2. **Moving**: Rearrangement within the architecture grid.
3. **Storing+Executing**: Transfer back to SLM traps in the entangling zone and application of the Rydberg pulse for two-qubit operation.

This cycle is constrained by the four AOD constraints introduced in Part I.

8.1.3 The NALAC Routing Framework

The core contribution of NALAC lies in transforming logical routing into a structured graph optimization problem that can be managed by polynomial-time heuristics.

NALAC begins by constructing an undirected interaction graph

$$G = (V, E),$$

where each vertex $v \in V$ represents a logical qubit array and each edge $(u, v) \in E$ corresponds to an executable transversal CZ_L operation present in the circuit to be compiled [29].

An operation is executable if it commutes with all preceding unexecuted operations, otherwise it will be moved to the next operational layer, for which a new interaction graph will be constructed.

Given the interaction graph, maximizing parallel transversal gates per shuttling cycle is equivalent to approximating a Max-Cut problem. Since exact Max-Cut is NP-hard, NALAC employs a Maximal Independent Set (MIS) heuristic:

1. Order vertices by descending degree.
2. Iteratively construct a maximal independent set $I \subseteq V$.
3. Assign I to AOD traps.
4. Assign $V \setminus I$ to SLM traps.

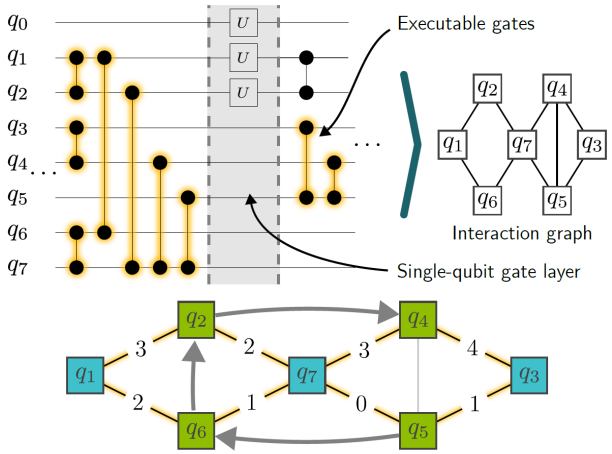

Figure 8.2: In the upper image, the construction of the interaction graph based on the commuting gates. In the lower image, the solution of the max-cut and edge-coloring problems of the interaction graph [29]

This strategy maximizes cross-partition edges, which correspond to transversal gates executable within a single entangling run.

After partitioning, NALAC schedules the transversal gates using a modified DSatur edge-coloring algorithm [29]. Each color represents a discrete entangling step.

Two additional constraints are imposed:

1. For any path of length two between AOD vertices, the direction of color inequalities must preserve a consistent ordering to prevent AOD crossing.

2. The least admissible color must not introduce cycles in the induced partial order graph.

Let $c(e)$ denote the color assigned to edge e. A color is admissible only if:

$$\forall e' \sim e, \quad c(e') \neq c(e),$$

and the induced directed graph remains acyclic. This guarantees that the final schedule is physically realizable and admits a topological ordering [29].

The edge coloring induces a partial order among logical qubits. NALAC performs a topological sort to derive a total order compatible with shuttling constraints.

This order determines horizontal placement in the entangling zone. Additionally, the compiler introduces resting positions: buffer coordinates ensuring that idle AOD arrays remain outside the Rydberg blockade radius R_b during entangling pulses. These buffers mitigate unintended interactions and reduce decoherence risk [29].

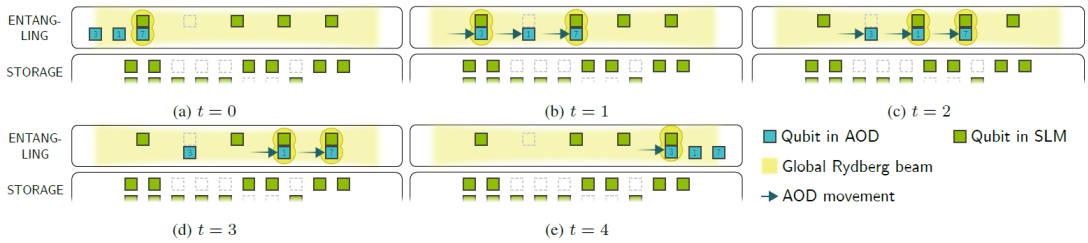

Figure 8.3: Time steps of the run obtained from the interaction graph in Figure 8.2 [29]

8.2 The ZAC compiler

ZAC has been proposed to exploit the full potential of multi-zone, multi-AOD systems, something that previous approaches failed to obtain due to stringent assumptions, like rigid layouts and no qubit reuse. Existing approaches fail to exploit the full potential of multi-zone, multi-AOD systems:

ZAC introduces a reuse-aware and load-balanced compilation framework designed explicitly for scalable zoned neutral atom architectures [38].

8.2.1 Hardware abstraction

As described in previous architectures, ZAC defines two fundamental traps, SLM traps for keeping atoms still in the architecture and AOD traps for transporting atoms within the architecture. The architecture is organised in three distinct regions: storage zone, entanglement zone and readout zone. To prevent unwanted Rydberg interactions, ZAC enforces a zone separation threshold of $d_sep \geq 10\mu m$ [38].

8.2.2 Movement cost modeling

Given a qubit placement $M = (m_0, m_1, ..., m_n)$, where $m_q = (x_q, y_q)$ are the coordinates of the q-th qubit in the storage zone, the movement cost for gate $g(q, q')$ is given by the gate cost function

$$gCost(g, \omega, M)$$

Where ω is the target Rydberg site. If qubits share the same SLM row (Parallel movement case):

$$gCost = \max\left(\sqrt{d(\omega, m_q)}, \sqrt{d(\omega, m_{q'})}\right)$$

If qubits lie on different rows (Sequential movement case):

$$gCost = \sqrt{d(\omega, m_q)} + \sqrt{d(\omega, m_{q'})}$$

The compiler explicitly favors placements enabling parallel transport, since parallel pickup reduces cumulative decoherence exposure [38].

8.2.3 Initial placement via Simulated Annealing

ZAC manages the initial qubit mapping to the storage zone using Simulated Annealing (SA).

The global objective function is:

$$cost(M) = \sum_{g \in G_2} w_g \cdot gCost(g, \omega_g^{near}, M)$$

where:

$$w_g = \max(0.1, 1 - 0.1(t - 1))$$

is a weight factor taking into account the Rydberg stage t at which the gate g is assigned [38].

The function represents the cost for the whole initial placement M. The cost is an estimation, taking into account the nearest Rydberg site ω_g^{near} to the two qubits participating gate g.

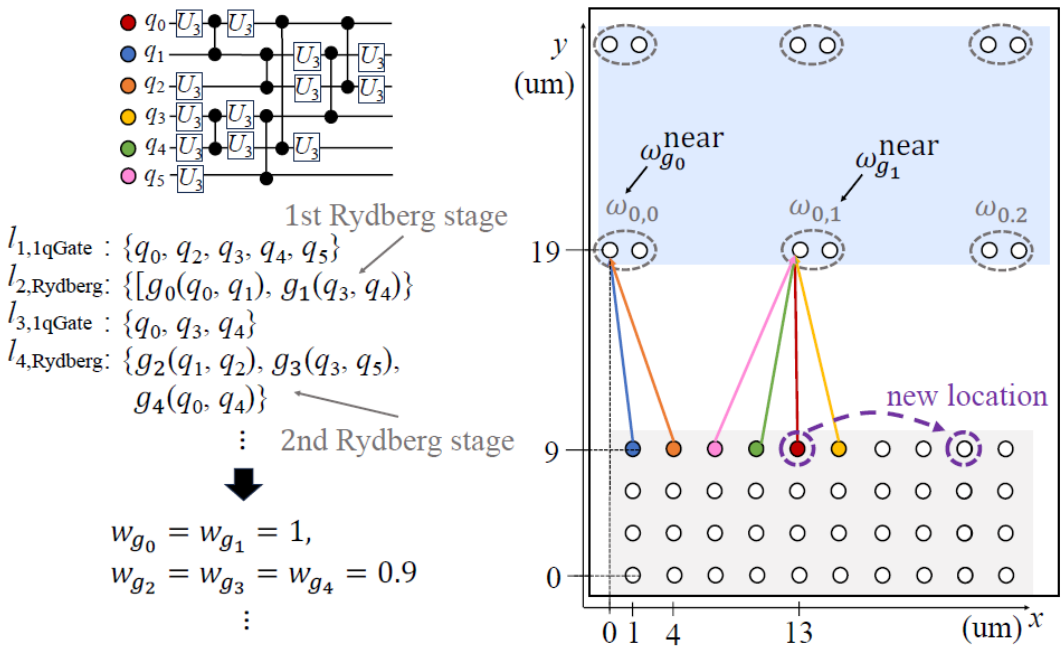

Figure 8.4: Initial placement in ZAC, with qubits of different colors assigned to spots in the storage zone, connected with the relative nearest Rydberg spot. A possible replacement with SA for red dot is depicted [38]

Simulated annealing performs stochastic local search over candidate mappings. At each iteration, small perturbations, e.g., swapping qubit positions, are evaluated, and non-improving moves are accepted with a probability that decreases with temperature. This allows the optimizer to escape local minima early in the search while gradually converging toward a low-cost configuration. The weighting scheme prioritizes early gates, where distance estimates are more reliable [38].

8.2.4 Routing through dynamic qubit reuse

ZAC manages the routing part by focusing on three main problems, with related solutions: Rydberg sites allocation, storage return, and reuse opportunities. The most impactful optimization is reuse-aware routing: instead of returning qubits to storage after each Rydberg stage, ZAC keeps qubits inside the entanglement zone if required in subsequent stages [38].

For newly required gates, ZAC formulates site allocation as a minimum-weight full matching problem of a bipartite graph composed of gates and Rydberg sites, solved through the Jonker-Volgenant algorithm, an optimized variant of the Hungarian method, iteratively refining dual variables and augmenting matchings to minimize total cost. This produces globally optimal gate-to-site assignments under the defined movement cost metric [38].

Reuse opportunities are modeled as a bipartite graph between currently occupied Rydberg sites and upcoming gate requirements. The Hopcroft-Karp algorithm is used to compute a maximum matching in the bipartite graph by alternating breadth-first and depth-first searches to identify multiple augmenting paths in parallel layers. This ensures maximal reuse while preventing site conflicts [38].

When qubits must return to storage, the problem is formulated similarly to gate placement, as a minimum-weight full matching problem of a bipartite graph, composed of non-reuse qubits and storage traps. The cost of the vertices in the graph is given by

$$Cost = \sqrt{d(s_{r,c}, m_q)} + \alpha \cdot \sqrt{d(s_{r,c}, m_{q'})}$$

This heuristic biases placement toward spatial proximity of future interaction partners, reducing transport in subsequent stages [38].

8.2.5 Multi-AOD scheduling and load balancing

Scheduling in ZAC is governed by two constraints: Trap dependencies and qubit dependencies. These define a scheduling DAG over rearrangement jobs.

ZAC assigns the longest rearrangement job to the first available AOD, maximizing hardware utilization and preventing idle mobile traps from becoming bottlenecks.

Because AOD control signals are one-dimensional, activating rows and columns forms a combinatorial rectangle, unintentionally picking up atoms at all intersections. ZAC avoids this via the parking maneuver: selected rows or columns are temporarily displaced during pickup to prevent unwanted atom capture [38].

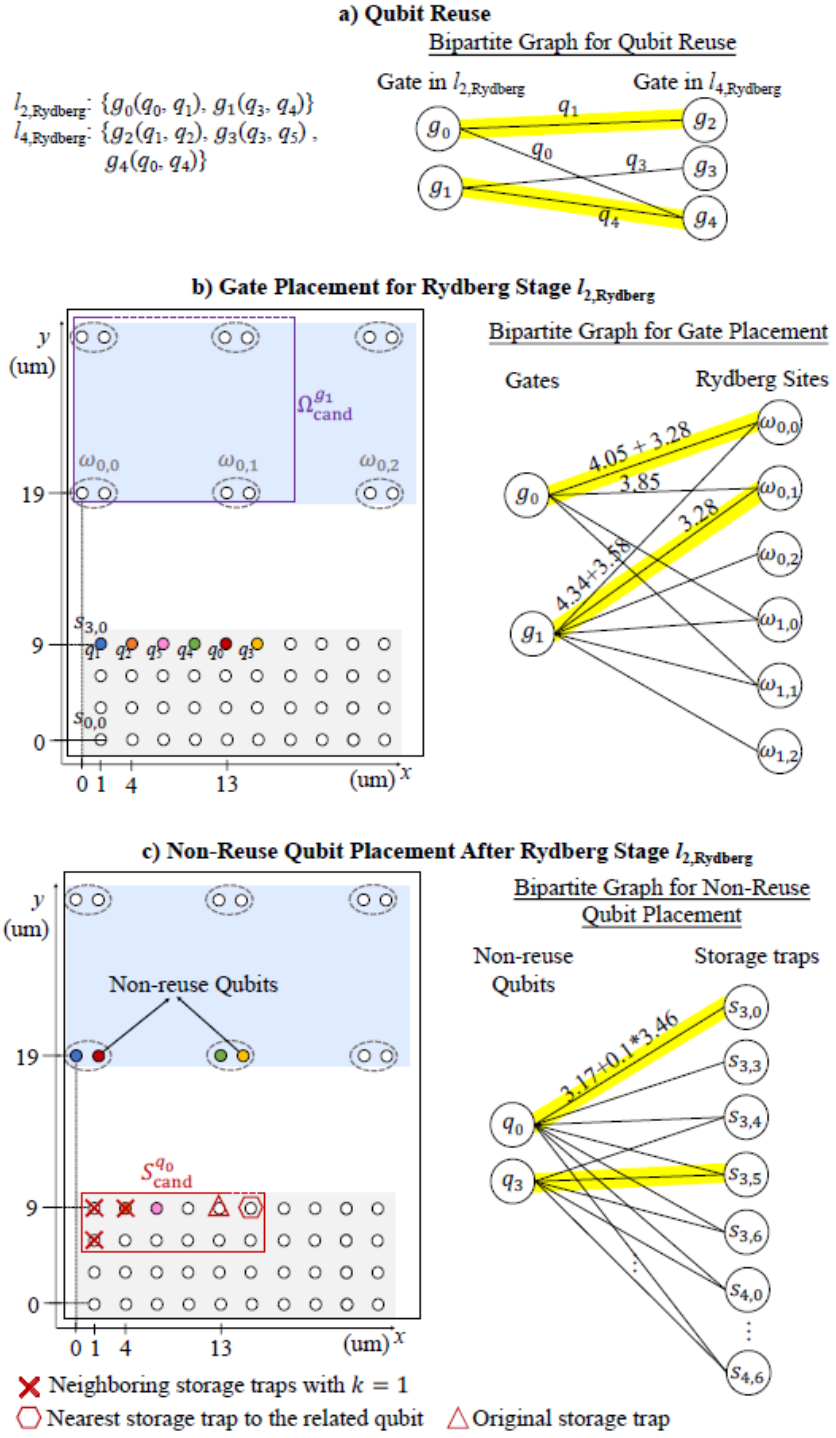

Figure 8.5: Heuristics for dynamic placement [38]

8.3 The MQT QMAP compiler

8.3.1 Compilation flow and the routing gap

The compilation flow for MQT QMAP consists of five stages: scheduling, reuse analysis, placement, routing, and code generation. Gates are first organized into alternating single- and two-qubit layers. Reuse analysis identifies qubits participating in consecutive entanglement stages to avoid unnecessary trap transfers, each incurring approximately 15 μs overhead [39]. This is taken from ZAC proposal, as it was considered a winning idea.

A critical limitation of earlier zoned compilers is the decoupling of placement and routing. Optimizing placement solely for minimal travel distance may generate configurations that violate the non-crossing constraint, forcing serialized rearrangement steps. This serialization effectively doubles the decoherence window for the corresponding layer.

Rather than treating routing as a post-processing step, QMAP evaluates the routability of a placement during the placement stage itself. This is achieved through a mathematically structured cost model that approximates the physical AOD constraints before actual routing is performed [39].

8.3.2 Formalization of the routing-aware placement problem

Exact routing computation for every candidate placement is computationally infeasible. Therefore, QMAP introduces a proxy cost function that captures routing complexity.

Given a placement p, movements are greedily grouped into compatible parallel sets G, where compatibility reflects row/column non-crossing and order preservation constraints. The primary placement cost is defined as:

$$cost(p) := \sum_{G \in groups(p)} \sqrt{d_{\max}(G)}$$

where $groups(p)$ denotes the set of parallel movement groups, $d_{\max}(G)$ is the maximum travel distance within group G. The square-root dependence reflects the physical transport relation $t \propto \sqrt{d/a}$ under constant acceleration [39].

This formulation naturally rewards fewer movement groups, larger parallel sets, and reduced maximal travel distance per group.

To achieve global optimization across layers, an extended cost function is introduced:

$$cost^*(p) := cost(p) + \sum_{a \in reused} cost_r(a) + \alpha \sum_{a \in placed} cost_l(a)$$

The reuse term $cost_r(a)$ incorporates a constant reward $-\gamma$ corresponding to the fidelity gain from avoiding two trap transfers. The look-ahead term $cost_l(a)$ estimates the cost of moving future interaction partners in subsequent layers; this anticipates future routing requirements and discourages placements that isolate upcoming partners. The parameter α regulates the influence of the look-ahead component: Larger α values increase global foresight, reducing the risk of blocking parallel movement in subsequent layers [39].

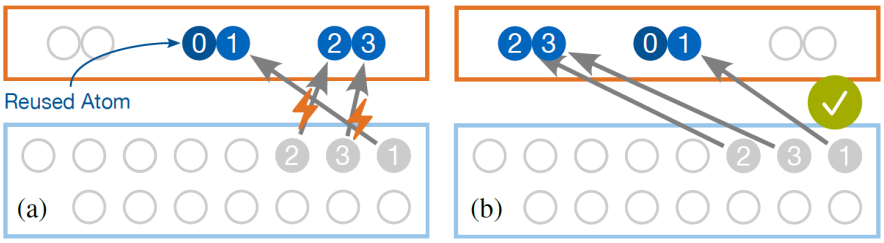

Figure 8.6: In (a), routing-agnostic placement, requiring two rearrangement steps to position qubits in the entanglement zone. In (b), routing-aware placement, only requiring one rearrangement step [39]

8.3.3 A* search strategy and state-space representation

To accelerate the search while maintaining high-quality solutions, a composite heuristic function $h^*(p)$ is defined. Because it includes an accelerating component, the resulting heuristic is non-admissible. This design choice intentionally trades strict optimality guarantees for computational tractability in extremely large search spaces.

The heuristics is composed of three parts:

- Admissible part: The admissible component provides a lower bound on the cost increase by assuming that all remaining atoms can be integrated into existing movement groups without introducing additional conflicts:

  $$h(p) = \max \sqrt{d_{\max}(G)} - \max \sqrt{d_{\min}(a)}$$

  Here $d_{\max}(G)$ denotes the maximal travel distance within a movement group G, while $d_{\min}(a)$ denotes the minimum achievable travel distance for atom a. This term captures the minimal possible additional cost under ideal grouping assumptions.

- Accelerating part: To anticipate routing conflicts early in the search, an accelerating component evaluates the standard deviation of key-value differences in the row and column mapping structures.

  If the spacing (gaps) between atoms in the target configuration deviates from their spacing in the source configuration, strictly parallel execution becomes mathematically impossible, forcing serialization of movements. Therefore, lower SD values indicate configurations that better preserve proportional spacing and parallel feasibility.

  To compensate for dimensional differences between zones, source indices (keys) are multiplied by a scaling factor (e.g., target columns/source columns) prior to SD computation. This normalization ensures that the heuristic consistently favors movements that preserve relative geometric structure across heterogeneous trap grids.

- Look-ahead part: To maintain sufficient search depth and avoid myopic decisions, the heuristic incorporates the average look-ahead cost $\overline{\text{cost}_l}(a)$ of all unplaced entities.

The final composite heuristic is therefore defined as:

$$h^*(p) := h(p) + \delta \cdot \left(\beta + \sum_{G \in \text{groups}(p)} \text{SD}(G) \right) \cdot |\{\text{unplaced}\}| + \sum_{a \in \text{unplaced}} \overline{\text{cost}_l}(a)$$

The parameter δ controls the strength of the accelerating component. Increasing δ enables a quicker search mode, prioritizing compilation speed at the potential expense of minimizing the total number of rearrangement steps.

8.3.4 Data structures for efficient compatibility verification

Efficient execution of the A* search is enabled by the use of binary search trees (BSTs) to verify compatibility of movements with physical routing constraints.

Each movement group maintains mappings of the form $k \mapsto v$ where k is the discrete source index (row or column), and v is the corresponding target trap index.

A new mapping is considered compatible with an existing group if and only if one of the following conditions holds:

1. Identity: If k already exists in the BST, it must map to the same v.
2. Order Preservation: If k is new, its target value v must satisfy

   $$v_{\text{lower}} < v < v_{\text{upper}}$$

   where v_{lower} and v_{upper} are the target values of its immediate predecessor and successor in the BST.

This order-preservation constraint prevents row/column crossing and merging violations, ensuring compliance with the physical limitations of AOD-based transport.

To further reduce the search space, an adjustable-size window centered around the nearest target trap is applied. Only traps within this window are considered during expansion. If no free traps are found, the window expands automatically. This adaptive pruning strategy significantly improves compilation performance while preserving high-quality placement solutions.

Part III

NAQC unified evaluation framework

Chapter 9

Problem formulation and design objectives

9.1 Architectural heterogeneity in neutral-atom platforms

Neutral-atom quantum computing has evolved along diverse architectural paradigms, each characterized by distinct physical constraints and operational mechanisms. Fixed atom arrays rely on static trap geometries with nearest-neighbor interactions between qubits. Dynamically field-programmable qubit arrays allow controlled atom displacement through acousto-optic deflectors, enabling flexible interaction patterns. Zoned storage-entanglement architectures introduce a further level of structural specialization by separating storage zones from interaction zones, improving the architecture's reliability. Each of these paradigms is defined by a combination of geometric topology, movement capabilities, interaction constraints, and gate operations characteristics. As a consequence, compilation cannot be architecture-independent: Routing decisions, movement scheduling, and interaction placement must explicitly account for the physical structure of the hardware.

Formally, we represent an architecture instance $\mathcal{A}$ as a parameter tuple

$$\mathcal{A} = (\mathcal{G}, \mathcal{M}, \mathcal{I}),$$

where $\mathcal{G}$ encodes geometrical structure, $\mathcal{M}$ defines movement capabilities, and $\mathcal{I}$ specifies interaction constraints. Given a logical circuit $\mathcal{C}$, compilation under architecture $\mathcal{A}$ produces a physical execution trace

$$\Phi(\mathcal{C}, \mathcal{A}) = \{\phi_1, \phi_2, \ldots, \phi_L\},$$

in which each layer ϕ_ℓ contains transport and interaction operations that respect the structural constraints of $\mathcal{A}$. Existing comparative studies frequently evaluate

such traces under heterogeneous modeling assumptions, obtaining unfair results. This lack of uniformity prevents controlled assessment of structural trade-offs.

9.2 Objective of the unified framework

The objective of the proposed framework is to embed heterogeneous architectural models and routing strategies inside a unified execution and evaluation structure. To formalize this objective, we define a unified evaluation operator

$$\mathcal{E} : (\mathcal{C}, \mathcal{A}, \mathcal{H}) \rightarrow (T, F),$$

where $\mathcal{C}$ denotes the logical circuit, $\mathcal{A}$ the architecture instance, and $\mathcal{H}$ the routing heuristic employed during compilation. The outputs are the total execution time T and the final circuit fidelity F. The key design principle is that $\mathcal{E}$ remains invariant across experiments. Only the architectural parameters and the selected heuristic vary. This ensures that performance differences arise from structural and algorithmic characteristics rather than inconsistencies in noise modeling or timing definitions.

Chapter 10

Unified execution and evaluation model

10.1 Circuit decomposition

The logical circuit $\mathcal{C}$ is first transpiled into the native neutral-atom gate set

$$\mathcal{G}_{native} = \{RX, RY, RZ, CZ\}.$$

Two-qubit operations are organized into ordered interaction layers

$$\mathcal{L}_{CZ} = \{L_1, L_2, \ldots, L_K\},$$

such that no qubit participates in more than one CZ gate within the same layer. These layers define an interaction ordering and capture dependency constraints among entangling operations. However, they do not yet represent physical execution units. Single-qubit gates are scheduled according to data dependencies and may appear before, after, or between interaction layers.

10.2 Operational execution layers

Physical execution is described through a sequence of operational layers

$$\mathcal{O} = \{O_1, O_2, \ldots, O_M\},$$

each representing a set of hardware-level operations that can be executed concurrently. An operational layer may contain transfer operations between trapping technologies, atom transport operations, two-qubit CZ gates, single-qubit rotations.

The ordered set of CZ layers $\mathcal{L}_{CZ}$ is therefore embedded within $\mathcal{O}$ as a subset of operational layers containing entangling gates:

$$\mathcal{L}_{CZ} \subseteq \mathcal{O}.$$

10.3 Layer duration model

For each operational layer O_m, let $T^{(m)}_{transfer}$ denote transfer time between SLM and AOD traps, $T^{(m)}_{move}$ denote atom transport time, $T^{(m)}_{2Q}$ denote the duration of parallel CZ gates, $T^{(m)}_{1Q}$ denote the duration of single-qubit gates. The duration of operational layer m is defined as the critical path across all operations it contains:

$$T_m = \max\left(T^{(m)}_{transfer}, T^{(m)}_{move}, T^{(m)}_{2Q}, T^{(m)}_{1Q}\right).$$

This formulation guarantees that every physical operation contributes to the execution time whenever it lies on the critical path. The total circuit duration is therefore

$$T = \sum_{m=1}^{M} T_m.$$

Transport time accounts for AOD serialization effects. If S_m compatible movement sublayers are required within operational layer O_m, and $d^{(s)}_{max}$ denotes the maximum displacement in sublayer s, then

$$T^{(m)}_{move} = \sum_{s=1}^{S_m} \frac{d^{(s)}_{max}}{v_{AOD}},$$

where v_{AOD} is the effective transport velocity.

10.4 Fidelity model

Circuit fidelity is computed multiplicatively across independent error sources:

$$F = F_{1Q}^{g_1} \cdot F_{2Q}^{g_2} \cdot F_{tran}^{N_{tran}} \cdot F_{move} \cdot F_{dec} \cdot F_{exc}.$$

Here, g_1 and g_2 denote single- and two-qubit gate counts, associated with the one-qubit and two-qubit fidelities F_{1Q} and F_{2Q}, F_{tran} is the transfer operation fidelity, elevated to the number of transfer operations, and F_{move} and F_{dec} capture movement-induced errors and decoherence, respectively.

Decoherence effects are modeled through an exponential decay law

$$F_{\text{dec}} = \exp\left(-\frac{T_{\text{idle}}}{T_2}\right),$$

where T_{idle} represents the total idle time accumulated by a qubit during the execution of the circuit and T_2 denotes the coherence time of the physical system. This formulation follows the standard physical model used for quantum coherence

decay, where the probability of preserving the quantum state decreases exponentially with time due to interactions with the surrounding environment. T_{idle} includes all periods during which a qubit is not actively participating in gate operations but remains susceptible to decoherence processes. As a consequence, longer circuit durations directly translate into larger coherence losses, making execution time an important factor in the overall fidelity of the computation.

Movement-induced degradation is modeled to capture the errors introduced by repeated atom transport operations. Atom shuttling is performed through AOD-controlled displacements, which involve acceleration and deceleration phases. These velocity changes introduce mechanical jerk in the motion profile, which can transfer energy to the trapped atoms and lead to motional heating, that can reduce trapping stability, resulting in the loss of the atom.

To account for this phenomenon, movement fidelity is modeled as

$$F_{\text{move}} = \prod_i (1 - \epsilon_{\text{step}})^{n_i},$$

where ϵ_{step} represents the error probability associated with a single movement step, and n_i denotes the number of displacements required for the i-th atom. Consequently, a larger number of movement steps leads to an exponential reduction in the overall fidelity of the transport phase.

While parameter values differ across architectures, the functional structure of the fidelity model remains identical, ensuring controlled comparison.

Chapter 11

Architectural instantiation and routing strategies

Architectural instances are generated from a structured parameter set describing geometry and hardware constraints. These parameters include grid dimensions, trap spacing, zoning configuration, transfer rules between trapping technologies, transport velocity, and native gate durations.

Storage sites are defined as spatial coordinates

$$\mathcal{S} = \{s_{ij} \in \mathbb{R}^2\},$$

while entanglement sites are defined as pairs of spatial coordinates

$$\mathcal{C} = \{(c_{ij}^L, c_{ij}^R) \in \mathbb{R}^2 \times \mathbb{R}^2\}.$$

In zoned architectures, storage and interaction regions are partitioned into disjoint subsets,

$$\mathcal{S} = \bigcup_z \mathcal{S}_z, \qquad \mathcal{C} = \bigcup_k \mathcal{C}_k,$$

each subset corresponding to a physically distinct zone. Architectural instantiation therefore produces a concrete geometric substrate onto which logical qubits must be placed and routed.

11.1 Initial placement

Before routing can be performed, logical qubits must be assigned to physical storage sites. This initial placement defines the starting configuration of the system and significantly impacts subsequent routing cost.

Formally, initial placement is a mapping

$$\mathcal{P}_0 : Q \to \mathcal{S},$$

where Q is the set of logical qubits.

A simple placement strategy assigns qubits sequentially to storage sites following grid order. More refined approaches incorporate structural information from the circuit interaction graph. For example, qubits with high mutual interaction frequency can be placed spatially close in order to reduce expected transport distance in early layers.

In architectures with fixed atom arrays, initial placement plays an even more critical role. Since atoms cannot be physically transported, the initial mapping effectively determines which two-qubit interactions are directly executable and which require SWAP-based routing through intermediate qubits. In this case, placement becomes a graph embedding problem: the circuit interaction graph must be embedded into the hardware connectivity graph with minimal distortion. Thus, while mobile architectures allow geometric correction through transport, fixed arrays shift optimization pressure toward placement quality.

11.2 Routing as a geometric assignment problem

Given a two-qubit interaction layer L_k, routing determines how the participating qubits are positioned so that required interactions can be executed. In mobile-atom architectures, routing is realized through atom transport.

Let s_i denote the current position of qubit i, and let c_j denote an available interaction site. An assignment is represented by a permutation π mapping qubits to interaction sites. The aggregate displacement cost associated with π is

$$\sum_i \|s_i - c_{\pi(i)}\|.$$

The baseline routing strategy solves

$$\min_\pi \sum_i \|s_i - c_{\pi(i)}\|,$$

that is, it selects the one-to-one assignment minimizing total Euclidean displacement. The physical interpretation is direct: $\|s_i - c_{\pi(i)}\|$ represents the geometric distance an atom must travel.

Since transport time scales with displacement and movement errors accumulate over time, minimizing total distance reduces both execution time and motion-induced noise. Mathematically, this corresponds to a minimum-weight bipartite matching problem between active qubits and available interaction sites. However,

this formulation is locally optimal for a single layer and ignores AOD serialization constraints, spatial conflicts between simultaneous moves, and inter-layer spatial continuity.

11.3 Structure-aware routing strategies

In fixed arrays, routing does not involve physical movement of atoms. Instead, qubits interact only with predefined neighbors in a static connectivity graph. If two qubits in layer L_k are not adjacent in hardware, routing is achieved through SWAP operations along a path connecting them. Each SWAP modifies the logical-to-physical mapping, dynamically updating qubit positions in logical space while physical atoms remain fixed. In this setting, routing cost is determined by the number of SWAP operations added.

More advanced routing strategies incorporate structural information beyond geometric minimization.

Enola constructs a dependency graph capturing inter-layer constraints. It applies ASAP scheduling to move operations as early as possible while respecting dependencies, and uses edge coloring to maximize concurrency under hardware limitations.

NALAC partitions interaction layers using graph coloring to reduce AOD beam conflicts and movement serialization. By structuring interactions into compatible subsets, it enables more efficient parallel execution.

ZAC employs simulated annealing to refine placement and routing decisions. Rather than minimizing only local displacement, it explores global configurations that preserve spatial locality across layers. Bipartite matching is used to maintain consistency between consecutive interaction layers.

These approaches extend baseline routing by embedding geometric or graph-based assignment within broader structural optimization frameworks.

11.4 Inter-layer movement optimization

In mobile architectures, an additional optimization concerns qubits participating in consecutive interaction layers.

A naive execution model returns atoms to storage after each layer, followed by a new displacement for the next interaction. This introduces redundant motion and additional transfer overhead in zoned systems. Instead, when a qubit is active in consecutive layers, its trajectory can be merged into a direct displacement between interaction sites.

If $c^{(k)}$ and $c^{(k+1)}$ denote the assigned sites in layers L_k and L_{k+1}, the atom moves directly from $c^{(k)}$ to $c^{(k+1)}$. This reduces cumulative displacement, shortens

execution time, and lowers transport-induced error.

Overall, routing strategies, together with initial placement decisions, determine how logical circuits are instantiated onto heterogeneous neutral-atom architectures. Mobile arrays emphasize geometric transport optimization, while fixed arrays emphasize connectivity-aware placement and SWAP minimization.

The framework accommodates both paradigms within a unified abstraction, enabling systematic comparison across architectural models.

Chapter 12

Experimental Setup

12.1 Simulation platform

This contribution builds upon and extends the Master's thesis work of Nicolò Tosin (Dipartimento di Informatica, Università di Torino), which has not yet been published. The simulator reproduces the complete operational pipeline of a neutral-atom processor. This includes:

- parsing OpenQASM circuit descriptions,
- transpiling circuits into the native gate basis,
- placing atoms onto the storage lattice,
- scheduling interaction layers,
- computing atom movements through AOD-controlled optical tweezers,
- evaluating execution time and fidelity metrics.

The simulator models both the logical compilation stages and the physical execution constraints imposed by the hardware.

12.2 Hardware parameters

The simulated neutral-atom device consists of a storage lattice and a dedicated entanglement region.

The storage grid is configured as a 5×20 array, corresponding to 100 storage sites, with a spacing of $3\,\mu m$ between neighboring traps.

The entanglement zone is organized as 5×10 columns of interaction pairs. Within each pair, atoms are separated by $2\,\mu m$, while adjacent pairs are separated by $10\,\mu m$.

The vertical separation between the storage zone and the entanglement zone is $10\,\mu m$.

A sketch of the zoned architecture is present in Figure 12.1.

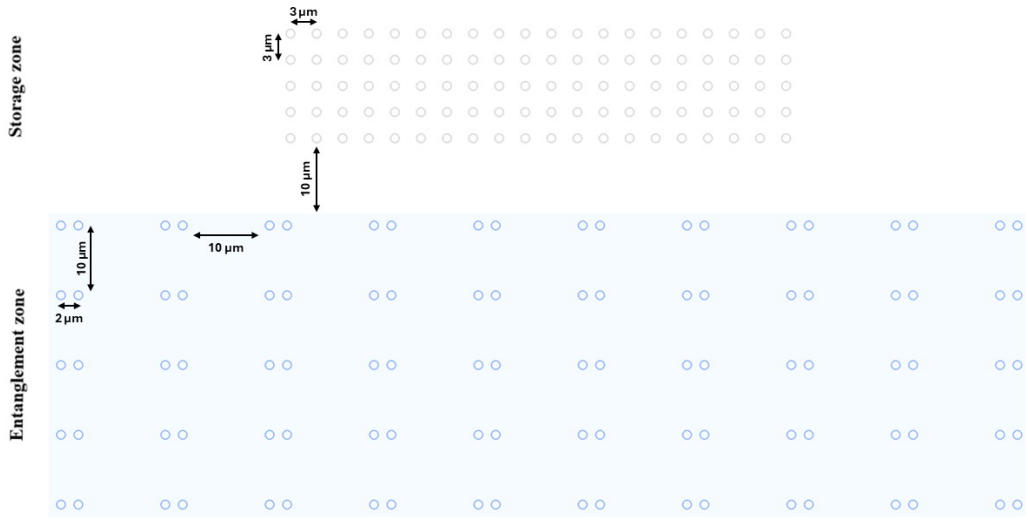

Figure 12.1: Sketch of zoned architecture used for experimental setup, with a storage zone on top and an entanglement zone on the bottom.

12.3 Fidelity and timing parameters

The fidelity model adopts parameter values consistent with experimental neutral-atom platforms reported in the literature. These parameters characterize both gate execution quality and error sources associated with atom transport and excitation processes.

Table 12.1 summarizes the fidelity and timing parameters used in the simulations.

Atom shuttling follows a triangular acceleration profile with acceleration a. Under this model, the time required to move an atom over a distance d is

$$t = 2\sqrt{\frac{d}{a}}.$$

Parameter	Value	Description
F_{1Q}	0.9997	Single-qubit gate fidelity
F_{2Q}	0.995	Two-qubit CZ gate fidelity
F_{exc}	0.9975	Excitation fidelity
F_{tran}	0.999	Atom transfer fidelity (SLM $\leftrightarrow$ AOD)
ϵ_{step}	0.001	Error per discrete AOD movement step
T_{1Q}	$0.52\,\mu s$	Single-qubit gate duration
T_{2Q}	$0.36\,ns$	Two-qubit gate duration
T_{tran}	$15\,\mu s$	Transfer time between SLM and AOD traps
T_2	$1.5\,s$	Coherence time
a	$2750\,m/s^2$	Atom shuttling maximal acceleration

Table 12.1: Hardware fidelity and timing parameters used in the simulations.

The duration of each operational layer is computed as the sequential sum of gate execution, transfer operations, and atom transport:

$$T_{layer} = T_{gate} + T_{transfer} + T_{move}.$$

This reflects the physical execution order of the operations, in which atoms are first picked from the storage trap, then transported, and finally deposited at the destination site before gate execution.

Chapter 13

Comparative results

This chapter presents the experimental evaluation of the different architectural configurations introduced in the previous sections. The objective of this analysis is to quantify how routing strategies and architectural constraints influence three key metrics of practical quantum execution: circuit fidelity, gate execution time, and total circuit duration.

Three architectural configurations are evaluated, each corresponding to a different combination of layout organization and routing heuristic.

- Fixed atom arrays constructed only by the 5×20 grid, compiled with Qiskit scheduler, taking into account the restriction zones to prevent cross-talk.
- Dynamically field-programmable qubit arrays: storage grid doubles its spots to allow the adjacency between interacting qubits. The atom movement problem is solved through the Enola heuristic.
- Zoned architecture: storage and entanglement zones are physically separated. The ZAC heuristic is exploited to route atoms during the computation.

For each benchmark circuit, the framework evaluates the compiled execution trace under identical physical assumptions. This ensures that differences in performance arise solely from architectural and routing properties rather than from modeling inconsistencies.

13.1 Benchmark circuits

To evaluate the routing strategies and architectural configurations within the unified framework, a diverse set of benchmark circuits was selected. The goal of the benchmark suite is to cover a broad spectrum of quantum workloads with different

structural properties, including varying qubit counts, interaction densities, and circuit topologies.

The benchmark suite consists of 15 quantum circuits belonging to several algorithmic families commonly used in quantum computing experiments. The number of qubits ranges from 10 to 78, allowing evaluation across both small and medium-scale problem instances.

All circuits are provided in OpenQASM format.

The benchmark suite includes circuits from the following algorithmic families:

- Arithmetic circuits: `adder_n10`, `multiply_n13`
- Bernstein–Vazirani algorithms: `bv_n19`, `bv_n30`, `bv_n70`
- Entangled state preparation: `cat_n35`, `ghz_n40`, `ghz_n78`, `wstate_n36`
- Quantum simulation workloads: `ising_n34`, `ising_n42`, `ising_n66`
- Machine learning primitives: `knn_n41`
- Fourier-based algorithms: `qft_n18`
- Verification circuits: `swap_test_n25`

These circuits exhibit different interaction patterns. For example, GHZ and CAT circuits contain highly structured interaction chains, while Bernstein–Vazirani circuits present shallow but wide interaction patterns. In contrast, Ising-model circuits generate dense entanglement layers with repeated qubit reuse.

For each circuit, the simulator records several metrics including the number of single- and two-qubit gates, the number of CZ layers, atom movements, execution time components, and final circuit fidelity.

13.2 Fidelity Analysis

Figure 17.1 reports the final circuit fidelity obtained for each benchmark across the evaluated architectures.

The most prominent trend emerging from the results is the advantage of the zoned architecture. Across almost all benchmarks, the zoned configuration achieves the highest final fidelity, consistently with state-of-the-art evaluations [38].

This behavior can be explained by the physical isolation provided by the storage–entanglement separation. In zoned architectures, atoms that are not currently participating in an interaction remain in the storage region and are therefore shielded from unintended laser excitation. This screening effect significantly reduces the probability of accidental excitation events, which represent one of the dominant error sources in neutral-atom devices.

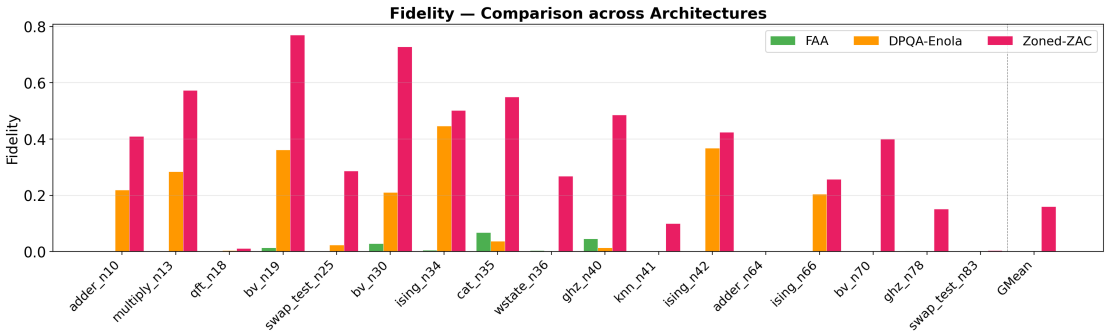

Figure 13.1: Circuit fidelity comparison across architectures.

In contrast, DPQA architectures execute interactions within a single shared region. While this provides flexibility for atom movement, it also exposes idle atoms to the excitation beams used for two-qubit operations. As a consequence, DPQA systems accumulate additional excitation errors over time, resulting in a lower overall fidelity.

This statement can be proved by calculating the average percentage of qubits participating in the CZ layers for each circuit: the circuits with the highest percentage, `ising_n34` and `ising_n42`, around 95%, are the ones with the lowest drop of fidelity from zoned architecture to DPQA, while the circuits with the lowest percentage, `cat_n35` and `bv_n70`, around 4%, are the ones with the highest drop of fidelity.

An even more pronounced fidelity degradation is observed in the fixed atom array architecture. The only way to let two atoms interact in this architecture without causing crosstalk is by employing local two-qubit gates, far less efficient than their global counterpart, with a fidelity around 92.5% [30]. Consequently, circuits compiled for FAA architectures obtain very low fidelities due to immature technology.

13.3 Total circuit duration

The total circuit duration ultimately determines the exposure of the system to decoherence. Figure 13.2 presents the full execution time for each benchmark.

The results reveal an important phenomenon: although FAA architectures have the highest gate execution time (Fig. 13.3), they still achieve the shortest overall circuit duration.

This apparent contradiction arises because transport time dominates the total execution cost in mobile-atom architectures. In FAA systems, atoms remain stationary throughout the computation, eliminating all movement overhead. As a

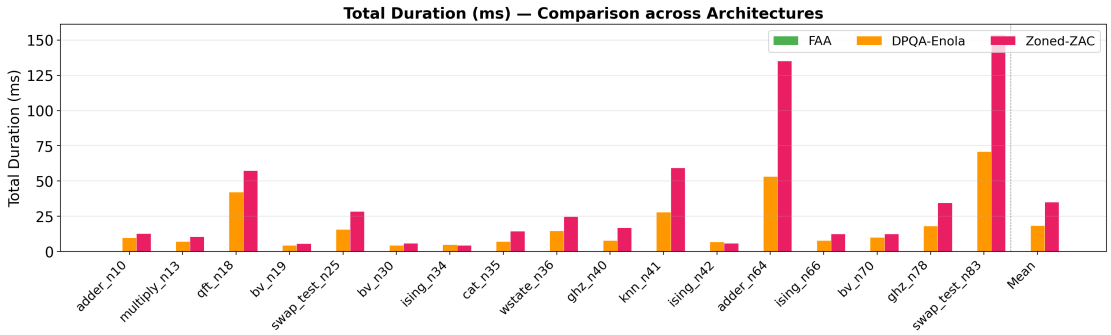

Figure 13.2: Total circuit duration comparison across architectures.

result, the total execution time consists only of gate operations.

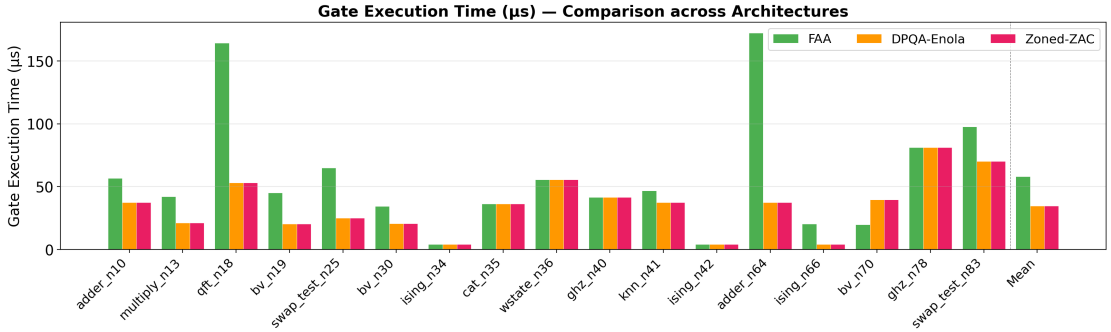

Figure 13.3: Gate execution time comparison across architectures.

In contrast, both DPQA and zoned architectures must repeatedly move atoms in order to align interacting qubits. Even when transport is highly optimized, the cumulative movement time often exceeds the cost of the gate operations themselves.

Among the mobile architectures, the zoned model exhibits the largest total execution time. This is a direct consequence of the storage–entanglement separation: each two-qubit interaction requires atoms to travel from the storage zone to the entanglement zone and back. These round-trip movements increase the average displacement distance and therefore increase the movement time. DPQA architectures avoid this overhead by operating within a single unified region, where atoms can be rearranged without crossing zone boundaries. As a result, their movement trajectories tend to be shorter, leading to lower total execution time compared to the zoned configuration.

Despite this, in some architectures, the zoned circuit duration is comparable to, or even lower than, the DPQA circuit. This is caused by a smarter heuristic that prevents the unnecessary movement of atoms participating in an interaction two

times in a row, the so-called reuse strategy.

This can be proved by calculating the average reuse of qubits by the ZAC heuristic for `ising` circuits: it stands at approximately 82% of the qubits used, demonstrating that the non-movement of these qubits reduces the circuit duration.

13.4 Discussion

Taken together, the results highlight the complex interplay between fidelity and transport overhead in neutral-atom quantum processors.

Fixed atom arrays minimize movement costs but suffer from a technology gap, which degrades fidelity.

DPQA architectures provide flexible routing with moderate transport overhead, but remain susceptible to excitation errors affecting idle atoms.

Zoned architectures introduce additional movement cost due to the separation between storage and entanglement regions, yet they provide a decisive advantage in terms of circuit fidelity by protecting idle atoms from unwanted interactions.

These observations motivate the architectural enhancements proposed in the following chapter, which aim to retain the fidelity advantages of zoned systems while reducing their transport overhead.

Part IV

Proposed architectural enhancements

Chapter 14

Mitigating transport bottlenecks in zoned architectures

14.1 Motivation

The evaluation presented in the previous chapter shows that zoned neutral-atom architectures provide the best fidelity performance among the considered paradigms. The separation between storage and entanglement regions allows entangling operations to be executed in controlled environments with reduced crosstalk and improved addressing selectivity.

However, the same architectural property that enables high-fidelity operation also introduces a structural limitation. In standard zoned systems, each two-qubit interaction requires atoms to travel from storage traps to an interaction region and return afterwards. As circuit size increases, these repeated storage-interaction-storage cycles accumulate and frequently dominate the total execution time.

Simulation results within the unified framework confirm that transport operations often exceed gate execution time by an order of magnitude. This phenomenon becomes particularly evident in large circuits, where limited AOD parallelism and entanglement-zone contention prevent full movement parallelization.

Consequently, improving the scalability of zoned architectures requires addressing the transport bottleneck without sacrificing the fidelity advantages of spatially separated interaction regions.

This chapter introduces two architectural extensions designed to reduce transport overhead while preserving the fundamental principles of zoned neutral-atom systems:

- the Zoned-Static architecture, which introduces adaptive in-place execution

inside the storage region,

- the Zoned-DPQA architecture, which exploits intra-zone movements typical of DPQA in zoned architecture to reduce movement overhead.

Both approaches aim to reduce unnecessary atom displacement and improve execution efficiency while maintaining compatibility with existing neutral-atom hardware capabilities.

14.2 Transport bottleneck in zoned architectures

In conventional zoned systems, the execution of a two-qubit gate between qubits q_i and q_j follows a fixed sequence: extraction of the atoms from storage traps, transport to a designated interaction site, execution of the entangling gate, and return transport to storage.

If $d_{i,k}$ denotes the geometric distance between the storage position of qubit i and its assigned interaction site in layer k, the transport time for that qubit can be approximated as

$$T_{\text{move}}^{(i,k)} = \frac{2d_{i,k}}{v},$$

where v represents the effective transport velocity.

Because multiple atoms share the same AOD resources, the duration of the movement phase for a layer is bounded by

$$T_{\text{move}}^{(k)} = \max_{i \in Q_k} \frac{2d_{i,k}}{v},$$

where Q_k denotes the set of qubits participating in layer k.

The total duration of the layer therefore becomes

$$T_{\text{layer}}^{(k)} = T_{\text{move}}^{(k)} + T_{\text{gate}} + 2T_{\text{trans}}.$$

In practice, the unified framework simulations show that

$$T_{\text{move}}^{(k)} \gg T_{\text{gate}} + 2T_{\text{trans}},$$

making movement the dominant contributor to execution time.

This bottleneck is often independent of the specific circuit being executed. Even with optimal routing, the architectural requirement of returning atoms to storage after each interaction layer forces repeated displacement cycles.

The architectural extensions proposed in this chapter focus on reducing the number and cost of these transport phases.

Chapter 15

Zoned-Static architecture

15.1 Architectural concept and physical layout

The Zoned-Static architecture extends the conventional zoned neutral-atom model by enabling two-qubit operations directly within the storage lattice whenever the participating atoms are geometrically adjacent, as in the FAA architecture.

In standard zoned architectures, storage traps serve exclusively as passive locations where atoms remain idle between interaction phases. Two-qubit gates are executed only within a dedicated entanglement region, requiring atoms to be transported from storage to interaction sites and returned afterward.

The Zoned-Static model instead treats the storage lattice as an active computational substrate. When two qubits are already located within the interaction radius required for Rydberg-mediated gates, the interaction can be executed directly in place without any atom displacement.

Formally, let

$$Q = \{q_1, q_2, \ldots, q_n\}$$

be the set of qubits in the circuit and let

$$\mathcal{S} = \{s_i \in \mathbb{R}^2\}$$

denote the set of storage trap positions. If two qubits q_i and q_j are mapped to storage sites

$$\Pi(q_i) = s_a, \qquad \Pi(q_j) = s_b,$$

their interaction can be executed in place if

$$\|s_a - s_b\| \leq d_{\text{int}},$$

where d_{int} is the Rydberg interaction distance.

If this condition is not satisfied, the qubits are transported to a dedicated entanglement zone following the standard zoned execution procedure.

This adaptive execution policy allows the system to dynamically select between in-place interactions and conventional zoned execution, reducing unnecessary atom transport when spatial locality is present.

The physical layout of the Zoned-Static architecture is identical to that of the baseline zoned architecture.

The system consists of two spatially separated regions:

- a storage region containing a dense lattice of traps,
- an entanglement region optimized for two-qubit gate execution.

Since the physical construction is identical to the conventional zoned architecture, the proposed extension does not require modifications to the hardware layout.

15.2 Initial placement strategy

The effectiveness of the first-layer in-place execution depends on the initial spatial arrangement of qubits. The Zoned-Static architecture, therefore, employs an interaction-aware initial placement strategy designed to maximize the probability of executing the first entangling layer directly within the storage lattice.

Let L_1 denote the first two-qubit interaction layer of the circuit and let

$$E_1 = \{(q_i, q_j)\}$$

be the set of interacting qubit pairs in that layer.

The initial placement defines a mapping

$$\Pi_0 : Q \to \mathcal{S}.$$

Pairs of qubits in E_1 are greedily assigned to adjacent storage traps whenever possible, satisfying

$$\|\Pi_0(q_i) - \Pi_0(q_j)\| \leq d_{\text{int}}.$$

This ensures that all interactions in the first layer can be executed in place without requiring any atom movement.

Qubits that cannot be paired during this process are assigned to the next available storage sites in the lattice.

By construction, this strategy guarantees that the first interaction layer can be executed entirely within the storage region, eliminating the transport overhead for that layer.

15.3 Adaptive interaction policy

After the first layer has been executed in place, subsequent interaction layers are scheduled using an adaptive execution policy.

Let L_k denote the k-th interaction layer and let Q_k be the set of qubits participating in that layer. The compiler first checks whether all required interactions can be executed within the current storage configuration.

If for every pair $(q_i, q_j) \in L_k$ the condition

$$\|\Pi_{k-1}(q_i) - \Pi_{k-1}(q_j)\| \leq d_{\text{int}}$$

holds, the entire layer is executed directly in the storage lattice.

Otherwise, the qubits participating in the layer are transported to the entanglement zone and assigned to interaction sites using the ZAC placement heuristic.

After executing the layer in the entanglement zone, the scheduler selects between two possible strategies for the subsequent layers:

- Storage execution strategy: All participating qubits are returned to the storage lattice, enabling potential in-place interactions in future layers.
- Entanglement-zone reuse strategy: he ZAC reuse mechanism is applied, moving only the necessary qubits to satisfy the interactions in the next layer while keeping others in the entanglement region.

The scheduler selects the strategy that minimizes the number of required AOD movement layers.

The choice is then performed for any subsequent layer.

15.4 Impact on transport overhead

When interactions are executed directly within the storage lattice, both the transport and transfer phases are eliminated.

If $T^{(k)}_{\text{move}}$ denotes the movement time for layer k, in-place execution implies

$$T^{(k)}_{\text{move}} = 0.$$

This reduction can significantly decrease the total circuit execution time in workloads that exhibit moderate spatial locality.

Furthermore, reducing atom transport decreases the exposure to motional heating and transfer-related atom loss, improving the overall reliability of the computation.

However, executing interactions in the storage region may disrupt certain reuse strategies typically exploited in zoned architectures, since atoms must remain in their storage traps during in-place layers.

Despite this limitation, the reduction in transport overhead often compensates for the loss of reuse opportunities, especially in circuits with repeated interactions among nearby qubits.

Chapter 16

Zoned-DPQA architecture

16.1 Architectural overview

The Zoned-DPQA architecture extends the conventional zoned neutral-atom model by introducing multiple interaction-capable regions.

The goal of this design is to combine two complementary architectural advantages:

- the shorter intra-zone movement characteristic of DPQA architectures,
- the reuse mechanisms and excitation shielding typical of zoned systems.

By allowing interactions to be executed in multiple regions, the architecture increases routing flexibility and reduces the average displacement required to bring interacting qubits together.

16.2 Physical layout

Let the architecture consist of two zones

$$\mathcal{Z} = \{Z_A, Z_B\}.$$

Each zone Z_z contains a set of paired traps

$$\mathcal{T}_z = \{(t^L_{z,i}, t^R_{z,i})\}_{i=1}^{N_z},$$

where $t^L_{z,i}$ and $t^R_{z,i}$ denote the left and right traps of the i-th pair in zone z.

Each pair forms a potential interaction site for two-qubit gates. The distance between the traps in a pair is fixed to the Rydberg interaction distance d_{int}, allowing controlled two-qubit operations.

During storage phases, atoms occupy the left trap of each pair:

$$q_i \to t^L_{z,i}.$$

When an interaction between qubits q_i and q_j is scheduled, the atoms are repositioned within the pair so that

$$q_i \to t^L_{z,k}, \quad q_j \to t^R_{z,k},$$

bringing them within the required interaction radius.

The two zones are separated by a vertical distance D_z.

16.3 Initial placement strategy

Let $Q = \{q_1, q_2, \ldots, q_n\}$ denote the set of qubits in the circuit and let L_1 denote the first interaction layer.

The initial placement defines a mapping

$$\Pi_0 : Q \to \mathcal{T}_A \cup \mathcal{T}_B.$$

The strategy aims to minimize the transport required to execute the first entangling layer. All qubits participating in interactions in L_1 are therefore placed in Zone A:

$$q_i \in L_1 \;\Rightarrow\; \Pi_0(q_i) \in \mathcal{T}_A.$$

The remaining qubits are placed in Zone B:

$$q_i \notin L_1 \;\Rightarrow\; \Pi_0(q_i) \in \mathcal{T}_B.$$

Within each zone, qubits are ordered according to their interaction frequency in the circuit interaction graph: qubits with higher interaction degree are placed closer to the boundary between the zones. This strategy reduces the expected displacement for qubits that frequently participate in interactions across zones.

16.4 Dynamic zone selection

For each interaction layer L_k, the compiler selects the zone in which the layer will be executed.

Let Q_k denote the set of qubits involved in layer L_k. For each candidate zone $z \in \mathcal{Z}$, the algorithm estimates the number of atoms that must be transferred from the opposite zone:

$$M_{\text{inter}}(z,k) = \left|\{q_i \in Q_k \mid \Pi_{k-1}(q_i) \notin Z_z\}\right|.$$

The execution zone is selected as

$$z^* = \arg\min_{z\in\mathcal{Z}} M_{\text{inter}}(z,k).$$

If both zones require the same number of inter-zone transfers, the selected zone maximizes the number of already-local qubits:

$$z^* = \arg\max_{z\in\mathcal{Z}} \left|\{q_i \in Q_k \mid \Pi_{k-1}(q_i) \in Z_z\}\right|.$$

This dynamic zone selection strategy reduces long-distance atom transport and increases routing flexibility.

16.5 Movement scheduling

Once the execution zone z^* is selected, the required atom movements are divided into two categories: Inter-zone movements, where atoms are transported between Z_A and Z_B, using ZAC heuristic; Intra-zone movements, where atoms are rearranged within the selected zone to form interaction pairs, exploiting Enola heuristic,

Once all required qubits reside in the selected zone, intra-zone rearrangement is performed.

16.6 Architectural comparison

The two proposed architectures address the transport bottleneck using complementary strategies.

- Zoned-Static reduces transport by enabling local in-place interactions when adjacency is available.
- Zoned-DPQA reduces transport by dynamically selecting the most efficient interaction region and balancing workload across zones.

The first approach is simpler and requires minimal hardware changes, while the second introduces greater architectural flexibility at the cost of more complex scheduling.

Both models preserve the fundamental advantages of zoned neutral-atom architectures while mitigating the dominant transport overhead identified in the previous analysis.

Chapter 17

Comparative evaluation

This chapter evaluates the performance of the proposed architectural extensions using the unified simulation framework introduced in the previous chapters. The goal of this analysis is to quantify how the proposed strategies mitigate the transport overhead of zoned architectures while preserving their key advantage in terms of circuit fidelity.

The comparison includes three architectural configurations:

- Zoned-ZAC: the baseline zoned architecture using the ZAC placement and routing heuristic.
- Zoned-FAA: the proposed Zoned-Static architecture, which enables in-place execution within the storage zone.
- Zoned-DPQA: the proposed dual-zone architecture that dynamically selects the most convenient execution region.

All experiments are performed using the hardware configuration and benchmark suite described in Part III.

17.1 Experimental assumptions

The evaluation assumes an improvement in the fidelity of local two-qubit operations within the storage lattice. Specifically, we assume that local two-qubit gates can reach a fidelity of approximately 99%.

This assumption is motivated by ongoing experimental progress in neutral-atom platforms, where improved laser addressing and interaction control are expected to increase the reliability of local Rydberg-mediated gates.

Under this assumption, executing interactions directly within the storage region may become competitive with the traditional strategy of moving atoms to a

dedicated entanglement zone. While in-place gates may still exhibit slightly lower fidelity than global interaction-zone operations, eliminating the transport and transfer phases can compensate for this difference in certain scenarios.

The proposed architectures are therefore evaluated under the assumption that local operations can achieve sufficiently high fidelity to justify this trade-off.

17.2 Circuit fidelity comparison

Figure 17.1 shows the final circuit fidelity obtained for each benchmark circuit across the evaluated architectures.

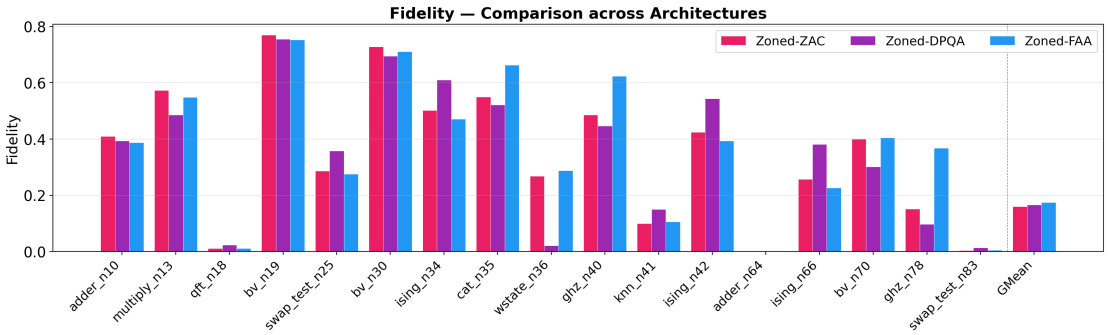

Figure 17.1: Circuit fidelity comparison across architectures.

The results show that the fidelity values obtained with the Zoned-ZAC and Zoned-DPQA architectures are largely comparable across the benchmark suite. This observation confirms that the proposed extensions preserve the fundamental advantage of zoned architectures: the shielding of idle atoms from unintended excitation during entangling operations.

Both the architectures shield idle qubits either by performing two-qubit operations in another zone, or by performing local two-qubit operations. This shielding effect significantly reduces the probability of accidental excitation events and remains a key contributor to the overall circuit fidelity.

The Zoned-FAA architecture exhibits similar behavior in many benchmarks. In this configuration, some interaction layers can be executed directly within the storage lattice when the participating qubits are already adjacent. When this occurs, the transport and transfer phases are avoided, eliminating the associated error sources.

However, the fidelity improvements of this strategy depend on the spatial structure of the circuit. When adjacency opportunities are limited, the architecture falls back to the standard zoned execution strategy, leading to results similar to the baseline architecture.

Overall, the results demonstrate that both proposed architectures maintain fidelity levels comparable to the baseline zoned system, confirming that the shielding advantage of zoned architectures is preserved.

17.3 Execution time analysis

While fidelity remains comparable across architectures, more significant variations appear in the total circuit duration. Figure 17.2 reports the execution time for each benchmark.

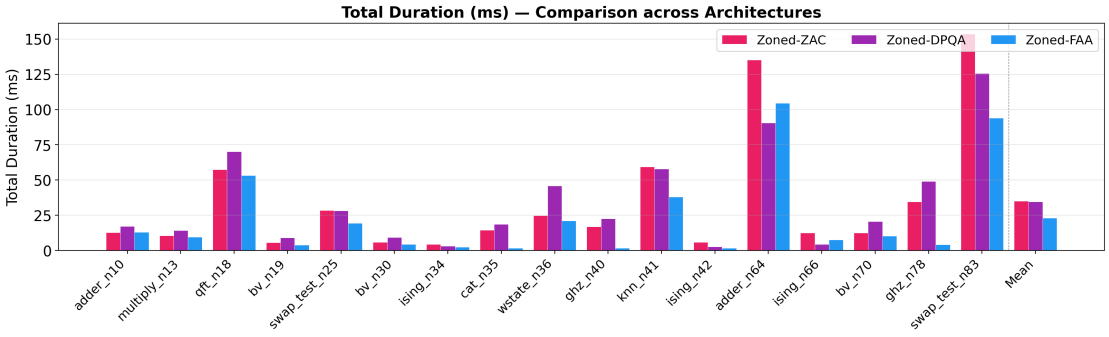

Figure 17.2: Total circuit duration comparison across architectures.

The results show that the average execution time of the Zoned-DPQA architecture is close to that of the baseline Zoned-ZAC configuration, while the Zoned-FAA architecture is the most advantageous strategy according to this metric.

In particular, the fastest circuits are the ones with the highest percentage of in-place CZ layers, as can be seen in Figure 17.3.

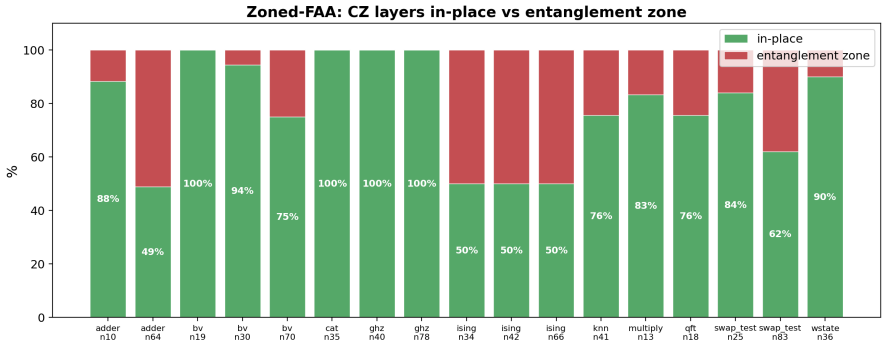

Figure 17.3: Percentage of CZ layers in storage zone and entanglement zone in Zoned-FAA architecture.

A more detailed inspection reveals that the relative performance strongly depends on the structure of the circuit.

For some benchmarks, the Zoned-DPQA architecture achieves noticeable reductions in execution time. These improvements occur when the dynamic selection of the execution zone successfully reduces the required atom displacement, allowing interactions to be executed closer to the qubits' current locations. This situation is particularly evident for `ising` circuits, where the percentage of intra-zone movements is well above that of inter-zone movement, as depicted in Figure 17.4.

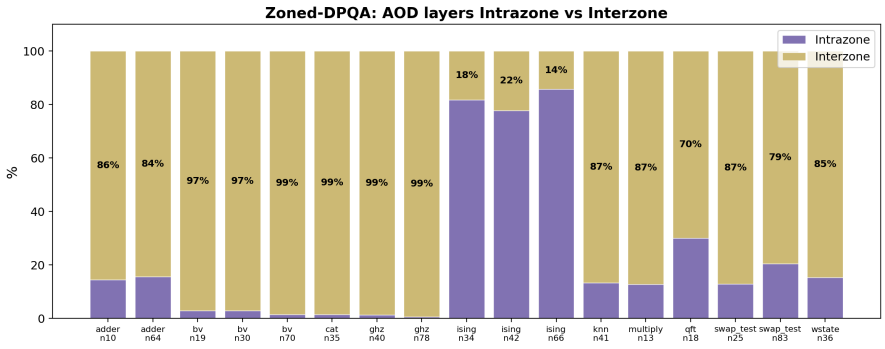

Figure 17.4: Percentage of inter-zone and intra-zone movements in Zoned-DPQA architecture.

In other circuits, however, the additional flexibility introduced by multiple zones may lead to additional coordination overhead, resulting in slightly longer execution times compared to the baseline configuration.

A similar circuit-dependent behavior can be observed in the Zoned-FAA architecture. In this case, however, the maximal duration is limited by the Zoned-ZAC architecture duration, since in the worst case, all the layers revert to the basic ZAC operations.

These results highlight an important property of transport-aware architectures: execution time improvements depend strongly on the interaction structure of the quantum circuit.

17.4 Discussion

The experimental results highlight two key observations.

First, both proposed architectures maintain circuit fidelity comparable to the baseline zoned architecture. This confirms that the shielding effect provided by zoned layouts remains intact even when introducing additional architectural flexibility.

Second, the overall circuit duration remains comparable across architectures, but performance differences emerge depending on the interaction patterns of individual circuits. Some circuits benefit significantly from reduced transport overhead, while others exhibit similar or slightly higher execution times.

This variability suggests that architectural flexibility and adaptive routing strategies are important for optimizing performance across heterogeneous workloads.

Chapter 18

Conclusions and Future Work

18.1 Summary of contributions

This thesis presented a unified framework for the compilation and evaluation of quantum circuits on neutral-atom quantum computing architectures. The main objective was to provide a common environment for comparing different architectural paradigms and compilation heuristics under consistent assumptions regarding physical parameters, scheduling policies, and performance metrics.

The framework integrates multiple state-of-the-art compilation strategies and supports several architectural models, including fixed atom arrays (FAA), dynamic programmable quantum arrays (DPQA), and zoned architectures. By implementing these approaches within the same simulation infrastructure, it becomes possible to perform fair and systematic comparisons in terms of execution time and circuit fidelity.

A key contribution of this work is the modular design of the framework. Placement strategies, routing algorithms, and scheduling heuristics are implemented as interchangeable components, allowing different combinations of techniques to be evaluated on the same circuit workloads. This design enables a detailed analysis of how architectural constraints and compilation strategies jointly influence the overall performance of quantum programs.

Experimental results obtained with the framework highlight the strengths and limitations of the considered architectures. In particular, zoned architectures demonstrate a clear advantage in terms of fidelity thanks to the shielding of idle atoms from unwanted excitation, which represents one of the dominant error sources in DPQA-style architectures. At the same time, the analysis reveals that atom transport can become a dominant contributor to circuit execution time, especially

when interactions require repeated transfers between storage and entanglement regions.

To address these limitations, two hybrid architectural extensions were proposed. The first, referred to as the Zoned-Static architecture, enables the execution of two-qubit gates directly within the storage lattice whenever atoms are already within the interaction radius, reducing unnecessary transport operations. The second, the Zoned-DPQA architecture, introduces multiple interaction-capable zones, allowing the compiler to dynamically select the execution region that minimizes inter-zone atom transfers.

Simulation results show that these hybrid models preserve the fidelity advantages of zoned architectures while introducing additional flexibility in atom routing. In particular, the shielding effect provided by zoned designs is maintained, while the additional configurability allows the system to reduce transport overhead in circuits exhibiting favorable interaction patterns.

18.2 Extensibility of the framework

An important aspect of the developed framework is its extensibility. The compilation pipeline has been designed so that new placement strategies, routing heuristics, and scheduling algorithms can be easily integrated as independent modules.

This modularity is particularly important in the context of neutral-atom quantum computing, where compilation techniques are evolving rapidly. As more efficient heuristics or routing algorithms are developed, they can be incorporated into the framework without requiring modifications to the overall architecture of the simulator.

As a result, the framework can serve not only as an evaluation tool for current compilation strategies, but also as a platform for testing and benchmarking future improvements in quantum compilation techniques.

18.3 Architecture-aware compilation

Another key observation emerging from the experimental analysis is that the efficiency of the compilation process strongly depends on the structure of the input quantum circuit. Circuits with different interaction patterns, qubit reuse characteristics, and spatial locality properties may benefit from different combinations of architectural models and compilation heuristics.

For this reason, a promising direction is to extend the framework so that it can act as an intermediate step in the compilation process. Instead of committing to a single architecture and routing heuristic, the compiler could evaluate multiple

architecture–heuristic combinations and automatically select the one that produces the most efficient implementation for the given circuit.

Such an adaptive compilation approach would enable the system to exploit the strengths of different architectures depending on the characteristics of the workload. For example, circuits with high interaction locality may benefit from static in-place execution strategies, while circuits with irregular interaction patterns may perform better under dynamic routing schemes.

18.4 Future work

Several directions can be explored to further extend the results presented in this thesis.

First, the hybrid architectures proposed in this work could benefit from more advanced compilation heuristics specifically designed for their structural characteristics. For example, placement algorithms that explicitly consider both storage adjacency and entanglement-zone accessibility could further reduce the number of required atom movements. Similarly, routing algorithms that jointly optimize intra-zone and inter-zone movements may improve the level of achievable parallelism.

Second, more sophisticated cost models could be incorporated into the compiler. The current implementation primarily evaluates execution time and fidelity, but future work could include additional physical effects such as heating during transport, atom loss probabilities, or more detailed error propagation models.

Another promising direction concerns the development of machine-learning-based compilation strategies. A learning-based compiler could analyze circuit characteristics and automatically predict the most efficient architecture and heuristic combination, reducing the need for exhaustive simulation across multiple configurations.

Finally, the framework could be extended to support larger-scale simulations and additional architectural variants, such as multi-zone systems with more than two interaction regions or heterogeneous layouts combining different trap densities.

Overall, the results presented in this thesis demonstrate that both architectural design and compilation strategies play a critical role in determining the performance of neutral-atom quantum processors. The proposed framework provides a flexible platform for exploring these design trade-offs and can serve as a foundation for future research in architecture-aware quantum compilation.

Bibliography

[1] Richard P Feynman. «Simulating physics with computers». In: *International journal of theoretical physics* 21.6/7 (1982), pp. 467–488 (cit. on p. 2).

[2] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. arXiv: quant-ph/9605043 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9605043 (cit. on p. 2).

[3] Peter W. Shor. «Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer». In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 1095-7111. DOI: 10.1137/s0097539795293172. URL: http://dx.doi.org/10.1137/S0097539795293172 (cit. on p. 2).

[4] David Deutsch. «Quantum theory, the Church–Turing principle and the universal quantum computer». In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (July 1985), pp. 97–117. ISSN: 0080-4630. DOI: 10.1098/rspa.1985.0070. eprint: https://royalsocietypublishing.org/rspa/article-pdf/400/1818/97/65933/rspa.1985.0070.pdf. URL: https://doi.org/10.1098/rspa.1985.0070 (cit. on p. 4).

[5] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. 10th anniversary ed. Cambridge ; New York: Cambridge University Press, 2010. ISBN: 978-1-107-00217-3 (cit. on pp. 4, 5).

[6] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. «Elementary gates for quantum computation». In: *Physical Review A* 52.5 (Nov. 1995), pp. 3457–3467. ISSN: 1094-1622. DOI: 10.1103/physreva.52.3457. URL: http://dx.doi.org/10.1103/PhysRevA.52.3457 (cit. on p. 4).

[7] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. «A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem». In: *Science*

292.5516 (Apr. 2001), pp. 472–475. ISSN: 1095-9203. DOI: 10.1126/science.1057726. URL: http://dx.doi.org/10.1126/science.1057726 (cit. on p. 5).

[8] M. Born and V. Fock. «Beweis des Adiabatensatzes». In: *Zeitschrift für Physik* 51.3 (Mar. 1928), pp. 165–180. ISSN: 0044-3328. DOI: 10.1007/BF01343193. URL: https://doi.org/10.1007/BF01343193 (cit. on p. 5).

[9] I. M. Georgescu, S. Ashhab, and Franco Nori. «Quantum simulation». In: *Reviews of Modern Physics* 86.1 (Mar. 2014), pp. 153–185. ISSN: 1539-0756. DOI: 10.1103/revmodphys.86.153. URL: http://dx.doi.org/10.1103/RevModPhys.86.153 (cit. on p. 6).

[10] Edward Farhi and Sam Gutmann. *An Analog Analogue of a Digital Quantum Computation*. 1996. arXiv: quant-ph/9612026 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9612026 (cit. on p. 6).

[11] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. *Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation*. 2005. arXiv: quant-ph/0405098 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/0405098 (cit. on p. 6).

[12] David P. DiVincenzo. *Topics in Quantum Computers*. 1996. arXiv: cond-mat/9612126 [cond-mat.mes-hall]. URL: https://arxiv.org/abs/cond-mat/9612126 (cit. on p. 7).

[13] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland. «Demonstration of a Fundamental Quantum Logic Gate». In: *Phys. Rev. Lett.* 75 (25 Dec. 1995), pp. 4714–4717. DOI: 10.1103/PhysRevLett.75.4714. URL: https://link.aps.org/doi/10.1103/PhysRevLett.75.4714 (cit. on p. 7).

[14] Y. Nakamura, Yu. A. Pashkin, and J. S. Tsai. «Coherent control of macroscopic quantum states in a single-Cooper-pair box». In: *Nature* 398.6730 (Apr. 1999), pp. 786–788. ISSN: 1476-4687. DOI: 10.1038/19718. URL: http://dx.doi.org/10.1038/19718 (cit. on p. 8).

[15] J. Bardeen, L. N. Cooper, and J. R. Schrieffer. «Theory of Superconductivity». In: *Phys. Rev.* 108 (5 Dec. 1957), pp. 1175–1204. DOI: 10.1103/PhysRev.108.1175. URL: https://link.aps.org/doi/10.1103/PhysRev.108.1175 (cit. on p. 8).

[16] B.D. Josephson. «Possible new effects in superconductive tunnelling». In: *Physics Letters* 1.7 (1962), pp. 251–253. ISSN: 0031-9163. DOI: https://doi.org/10.1016/0031-9163(62)91369-0. URL: https://www.sciencedirect.com/science/article/pii/0031916362913690 (cit. on p. 8).

[17] He-Liang Huang, Dachao Wu, Daojin Fan, and Xiaobo Zhu. «Superconducting quantum computing: a review». In: *Science China Information Sciences* 63.8 (July 2020). ISSN: 1869-1919. DOI: 10.1007/s11432-020-2881-9. URL: http://dx.doi.org/10.1007/s11432-020-2881-9 (cit. on p. 8).

[18] Muhammad AbuGhanem. «IBM quantum computers: evolution, performance, and future directions». In: *The Journal of Supercomputing* 81.5 (Apr. 2025). ISSN: 1573-0484. DOI: 10.1007/s11227-025-07047-7. URL: http://dx.doi.org/10.1007/s11227-025-07047-7 (cit. on p. 8).

[19] G Wendin. «Quantum information processing with superconducting circuits: a review». In: *Reports on Progress in Physics* 80.10 (Sept. 2017), p. 106001. ISSN: 1361-6633. DOI: 10.1088/1361-6633/aa7e1a. URL: http://dx.doi.org/10.1088/1361-6633/aa7e1a (cit. on p. 9).

[20] Wolfgang Paul and Helmut Steinwedel. In: *Zeitschrift für Naturforschung A* 8.7 (1953), pp. 448–450. DOI: doi:10.1515/zna-1953-0710. URL: https://doi.org/10.1515/zna-1953-0710 (cit. on p. 9).

[21] Colin D. Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M. Sage. «Trapped-ion quantum computing: Progress and challenges». In: *Applied Physics Reviews* 6.2 (May 2019). ISSN: 1931-9401. DOI: 10.1063/1.5088164. URL: http://dx.doi.org/10.1063/1.5088164 (cit. on p. 9).

[22] A.M. Steane and E.G. Rieffel. «Beyond bits: the future of quantum information processing». In: *Computer* 33.1 (2000), pp. 38–45. DOI: 10.1109/2.816267 (cit. on p. 10).

[23] Fulvio Flamini, Nicolò Spagnolo, and Fabio Sciarrino. «Photonic quantum information processing: a review». In: *Reports on Progress in Physics* 82.1 (Nov. 2018), p. 016001. ISSN: 1361-6633. DOI: 10.1088/1361-6633/aad5b2. URL: http://dx.doi.org/10.1088/1361-6633/aad5b2 (cit. on pp. 10, 11).

[24] Nobuyuki Matsuda and Hiroki Takesue. In: *Nanophotonics* 5.3 (2016), pp. 440–455. DOI: doi:10.1515/nanoph-2015-0148. URL: https://doi.org/10.1515/nanoph-2015-0148 (cit. on p. 11).

[25] Ludwig Schmid, David F Locher, Manuel Rispler, Sebastian Blatt, Johannes Zeiher, Markus Müller, and Robert Wille. «Computational capabilities and compiler development for neutral atom quantum processors—connecting tool developers and hardware experts». In: *Quantum Science and Technology* 9.3 (Apr. 2024), p. 033001. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ad33ac. URL: http://dx.doi.org/10.1088/2058-9565/ad33ac (cit. on pp. 12, 13, 15–17, 19, 20).

[26] Dolev Bluvstein et al. «A quantum processor based on coherent transport of entangled atom arrays». In: *Nature* 604.7906 (Apr. 2022), pp. 451–456. ISSN: 1476-4687. DOI: 10.1038/s41586-022-04592-6. URL: http://dx.doi.org/10.1038/s41586-022-04592-6 (cit. on pp. 13, 14, 16, 17).

[27] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. «Quantum computing with neutral atoms». In: *Quantum* 4 (Sept. 2020), p. 327. ISSN: 2521-327X. DOI: 10.22331/q-2020-09-21-327. URL: http://dx.doi.org/10.22331/q-2020-09-21-327 (cit. on pp. 14, 16).

[28] Jérôme Beugnon et al. «Two-dimensional transport and transfer of a single atomic qubit in optical tweezers». In: *Nature Physics* 3.10 (Aug. 2007), pp. 696–699. ISSN: 1745-2481. DOI: 10.1038/nphys698. URL: http://dx.doi.org/10.1038/nphys698 (cit. on p. 14).

[29] Yannick Stade, Ludwig Schmid, Lukas Burgholzer, and Robert Wille. «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)*. IEEE, Sept. 2024, pp. 784–795. DOI: 10.1109/qce60285.2024.00098. URL: http://dx.doi.org/10.1109/QCE60285.2024.00098 (cit. on pp. 15, 17, 39–42).

[30] T. M. Graham et al. «Multi-qubit entanglement and algorithms on a neutral-atom quantum computer». In: *Nature* 604.7906 (Apr. 2022), pp. 457–462. ISSN: 1476-4687. DOI: 10.1038/s41586-022-04603-6. URL: http://dx.doi.org/10.1038/s41586-022-04603-6 (cit. on pp. 16, 66).

[31] Hanrui Wang, Pengyu Liu, Daniel Bochen Tan, Yilian Liu, Jiaqi Gu, David Z. Pan, Jason Cong, Umut A. Acar, and Song Han. *Atomique: A Quantum Compiler for Reconfigurable Neutral Atom Arrays*. 2024. arXiv: 2311.15123 [quant-ph]. URL: https://arxiv.org/abs/2311.15123 (cit. on p. 17).

[32] Ben W. Reichardt et al. «Fault-tolerant quantum computation with a neutral atom processor». In: (Nov. 2024). arXiv: 2411.11822 [quant-ph] (cit. on pp. 17, 18).

[33] *Qiskit Transpiler Documentation*. IBM Quantum / Qiskit API documentation. Transpilation overview and API reference for the Qiskit transpiler. 2025. URL: https://qiskit.qotlabs.org/docs/api/qiskit/transpiler (cit. on pp. 19–22).

[34] Friedrich Wagner, Andreas Bärmann, Frauke Liers, and Markus Weissenbäck. «Improving Quantum Computation by Optimized Qubit Routing». In: *Journal of Optimization Theory and Applications* 197 (May 2023), pp. 1–34. DOI: 10.1007/s10957-023-02229-w (cit. on pp. 21, 22).

[35] Jonathan M. Baker, Andrew Litteken, Casey Duckering, Henry Hoffmann, Hannes Bernien, and Frederic T. Chong. «Exploiting Long-Distance Interactions and Tolerating Atom Loss in Neutral Atom Quantum Architectures». In: *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 2021, pp. 818–831. DOI: 10.1109/ISCA52012.2021.00069 (cit. on pp. 28–30).

[36] Hanrui Wang, Daniel Bochen Tan, Pengyu Liu, Yilian Liu, Jiaqi Gu, Jason Cong, and Song Han. *Q-Pilot: Field Programmable Qubit Array Compilation with Flying Ancillas*. 2024. arXiv: 2311.16190 [quant-ph]. URL: https://arxiv.org/abs/2311.16190 (cit. on pp. 33–35).

[37] Daniel Bochen Tan, Wan-Hsuan Lin, and Jason Cong. «Compilation for Dynamically Field-Programmable Qubit Arrays with Efficient and Provably Near-Optimal Scheduling». In: *Proceedings of the 30th Asia and South Pacific Design Automation Conference*. ASPDAC '25. ACM, Jan. 2025, pp. 921–929. DOI: 10.1145/3658617.3697778. URL: http://dx.doi.org/10.1145/3658617.3697778 (cit. on pp. 36–38).

[38] Wan-Hsuan Lin, Daniel Bochen Tan, and Jason Cong. *Reuse-Aware Compilation for Zoned Quantum Architectures Based on Neutral Atoms*. 2024. arXiv: 2411.11784 [quant-ph]. URL: https://arxiv.org/abs/2411.11784 (cit. on pp. 42–46, 65).

[39] Yannick Stade, Wan-Hsuan Lin, Jason Cong, and Robert Wille. «Routing-Aware Placement for Zoned Neutral Atom-based Quantum Computing». In: *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, Oct. 2025, pp. 1–9. DOI: 10.1109/iccad66269.2025.11240721. URL: http://dx.doi.org/10.1109/ICCAD66269.2025.11240721 (cit. on pp. 47, 48).