# Towards a Cross-Compilation Strategy for Neutral Atoms Using Graph-like ZX-Diagrams (Work-in-Progress)

**Korbinian Staudacher**[1*], Ludwig Schmid[†], Wanja Sajko[*], and Robert Wille[†‡]
[1] Presenting Author
[*]MNM-Team, Ludwig-Maximilians-Universität München, Munich, Germany,
[†]Chair for Design Automation, Technical University of Munich, Munich, Germany
[‡]Software Competence Center Hagenberg GmbH, Hagenberg, Austria
korbinian.staudacher@nm.ifi.lmu.de; ludwig.s.schmid@tum.de; robert.wille@tum.de

***Abstract*—In this work-in-progress, we address the challenge of optimizing quantum circuit compilation for neutral atom-based quantum computers by proposing a novel cross-compilation scheme that integrates ZX-circuit extraction and hardware mapping processes. Leveraging ZX-calculus, our approach enables a more cohesive and informed decision-making process by allowing the extraction algorithm to generate multiple candidate circuits, which are then evaluated by the mapping algorithm. This iterative feedback loop ensures that the extraction paths are optimized according to the hardware's current configuration and physical parameters, potentially leading to improvements in compilation efficiency and hardware utilization. We present the fundamental idea and illustrate and discuss potential compilation improvements. This work-in-progress lays the idea for novel ZX-based quantum circuit compilation and offers a blueprint for adapting the cross-compilation approach to other extraction and mapping algorithms and hardware platforms.**

## I. Introduction

Executing quantum algorithms on real quantum hardware necessitates several preprocessing steps to translate general quantum operations into an executable form. Given the current limitations in qubit connectivity and error rates, these quantum circuits must be optimized for specific hardware architectures to achieve optimal results. This process, known as *compilation*, includes *circuit synthesis*, *transpilation*, and *mapping/routing*, with various algorithms and techniques available for each step and hardware type.

Typically, these compilation steps are performed independently, with the output of one step serving as the input for the next. For instance, Qiskit [1] offers a set of transpilation and mapping algorithms to optimize quantum circuits for IBM's hardware. However, this sequential approach can yield suboptimal results, as the output of one step may not be ideal for the subsequent step.

Commonly used, ZX-calculus has established itself as a powerful tool for quantum circuit transpilation and optimization. It involves converting a quantum circuit into a ZX-diagram, simplifying it using diagrammatic rules, and then converting it back into a quantum circuit through *circuit extraction*.

In this work-in-progress, we propose a novel approach to quantum circuit compilation based on ZX-calculus, which combines the circuit extraction and mapping steps into a process we term *cross-compilation*. This method communicates between the extraction and mapping algorithms, selecting the best extraction steps iteratively to construct a fully mapped quantum circuit that meets hardware constraints and is optimized for the given parameters.

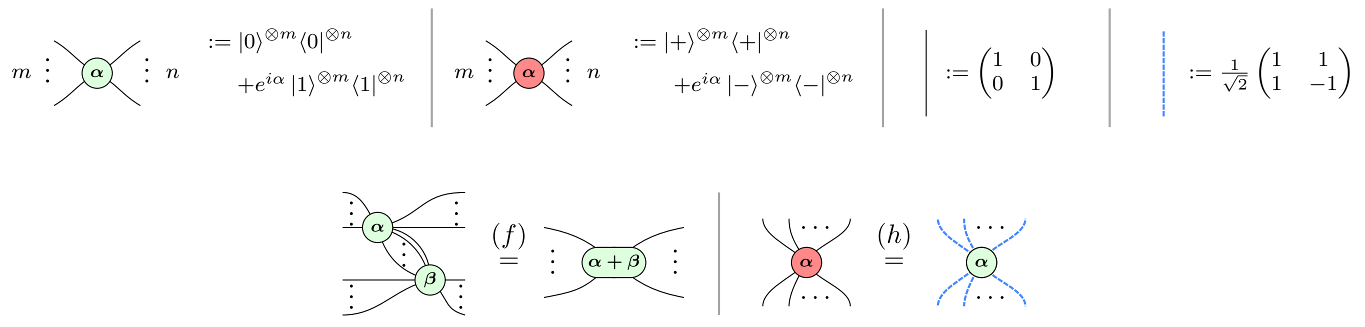
We discuss the underlying idea and illustrate the potential of this approach by applying it to the compilation of quantum circuits for neutral atom quantum computers. Specifically, the idea is based on combining an extended version of the ZX strategy of diagram simplification and extraction algorithm [2], [3] that supports multi-controlled phase gates [4] and the hybrid mapping algorithm from [5], which handles both SWAP gate insertion and shuttling operations available on neutral atom hardware.

The concept of cross-compilation is generalizable to other ZX extraction methods [6], mapping strategies, and quantum computing platforms such as superconducting qubits or trapped ions. Furthermore, we plan to publish the full code as open-source software in the *Munich Quantum Toolkit* [7], providing a blueprint for implementing similar cross-compilation schemes for various hardware.

The remainder of this paper is structured as follows: In Sec. II, we provide an overview of ZX-calculus and the extraction of quantum circuits from ZX-diagrams. Sec. III introduces the neutral atom quantum computing platform and the corresponding mapping problem. Sec. IV discusses the issue of suboptimal compilation due to the independent use of extraction and mapping algorithms, followed by an overview of the proposed solution in Sec. V. In Sec. VI, we briefly discuss the idea and its potential to improve existing compilation strategies. Finally, we conclude the work in Sec. VII.

## II. ZX Circuit Extraction

The following section gives a brief overview of the ZX-calculus and the extraction of quantum circuits from graph-like ZX-diagrams.



$m$
$\alpha$
$n$
$:= |0\rangle^{\otimes m}\langle 0|^{\otimes n} + e^{i\alpha}|1\rangle^{\otimes m}\langle 1|^{\otimes n}$
$:= |+\rangle^{\otimes m}\langle +|^{\otimes n} + e^{i\alpha}|-\rangle^{\otimes m}\langle -|^{\otimes n}$
$:= \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$
$:= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$
$\beta$
$(f)$
$\alpha+\beta$
$(h)$

Figure 1. Definition of Z-spiders and X-spiders, the identity and the hadamard wire, and two of the ZX-calculus rules, namely the fusion rule $(f)$ and the Hadamard rule $(h)$.

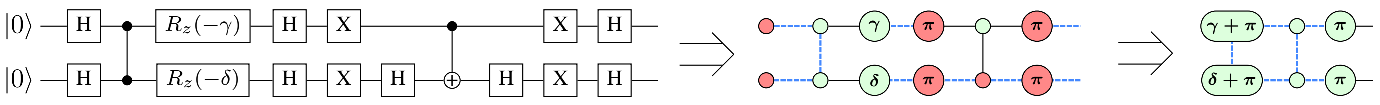


Figure 2. Example of a two-qubit Grover search circuit which is translated first to a ZX-diagram and then converted to a graph-like ZX-diagram using the rules from the ZX-calculus.

## A. ZX-Calculus

ZX-calculus is a diagrammatic language for reasoning about linear maps in quantum computing where nodes (spiders) and edges (wires) form an undirected graph called a ZX-diagram. There are two types of spiders: green *Z-spiders* and red *X-spiders*. Spiders can be parametrized with an angle $\alpha \in [0, 2\pi)$ and correspond to two-dimensional matrices in Hilbert space with the definition given in Figure 1. Spiders can have any number of ingoing and outgoing wires. We can compose two diagrams either horizontally by joining the outputs of one diagram with the inputs of the other (denoted by $\circ$), or vertically by placing them side by side (denoted by $\otimes$). This corresponds to the known dot and tensor product in Hilbert space. For convenience, we distinguish between two types of wires: *Normal wires*, representing the identity, and *Hadamard wires*, representing the Hadamard matrix. Wires entering the diagram from the left are called input wires, with the adjacent spiders defined as inputs $I$. Wires exiting to the right are called output wires, with adjacent spiders defined as outputs $O$.

**Example 1.** A spider with two wires (incoming and outgoing) corresponds to the following single-qubit gates: For $\alpha = 0$ the Z-spider corresponds to the identity matrix and the X-spider corresponds to the Hadamard matrix. For $\alpha = \pi$ the Z-spider corresponds to the Pauli-Z matrix and the X-spider corresponds to the Pauli-X matrix. ※

As ZX-diagrams represent linear maps, they constitute a graphical language for quantum circuits. This means any quantum circuit can be represented as a ZX-diagram by replacing gates with equivalent diagrams. The ZX-calculus provides a set of rules to manipulate ZX-diagrams without changing the linear map, which can be used to simplify the diagram and possibly the underlying quantum circuit.

**Example 2.** In Figure 2 a two-qubit Grover search circuit is translated to a ZX-diagram. This is done by replacing the above-mentioned single-qubit gates and replacing the Hadamard gate with Hadamard wires. The two-qubit gates are also replaced by their ZX-calculus counterparts. ※

The fusion rule $(f)$ allows merging spiders of the same color if they are connected by at least one normal wire, and $(h)$ allows changing the colors of spiders by flipping normal and Hadamard wires. All rules hold in both directions and are also valid with interchanged colors, so we can also split up spiders with $(f)$. There exists a complete graphical rule set for transforming ZX-diagrams [8].

## B. Graph-like ZX-diagrams

In this work, we consider the class of *graph-like* ZX-diagrams as introduced in [2], which allow us to represent any quantum computation as a graph of parametrized green Z-spiders and Hadamard wires. Formally, a ZX-diagram is graph-like iff:

1) All spiders are Z-spiders,
2) All wires between spiders are Hadamard wires,
3) There are no parallel Hadamard wires or self-loops,
4) Every Z-spider is connected to at most one input or one output, and
5) Every input or output is connected to a Z-spider.

One can transform any ZX-diagram into an equivalent graph-like ZX-diagram by repeatedly applying standard ZX-rules [2].

**Example 3.** The last step in Figure 2 shows the graph-like ZX-diagram of the two-qubit Grover search circuit. ※

The graph-like ZX-diagram can be used to perform diagram simplifications and optimizations [2]. In addition to the standard ZX-rules, this includes techniques such as *local complementation*, *pivoting*, and *phase gadget elimination* where the latter was introduced in [3]. These techniques are already

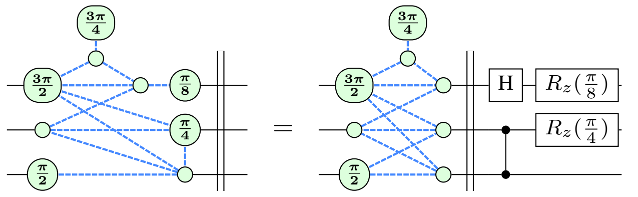


Figure 3. Basic circuit extraction from a graph-like ZX-diagram.

implemented in tools such as PyZX [9] and in this work we assume that the graph-like ZX-diagram is already fully simplified.

### C. Circuit Extraction

As a final step, one must extract a quantum circuit from the previously optimized graph-like ZX-diagram. This extraction is feasible in polynomial time if the underlying graph has some kind of *flow* [10], [11]. Here, we provide a brief overview of the extraction algorithm for graph-like ZX-diagrams with gflow as described in [10]. The algorithm extracts a quantum circuit by identifying suitable parts of the ZX-diagram and converting them into their equivalent quantum gates. These parts are then removed from the diagram, extracting one gate at a time, until only the inputs and outputs remain. During the process, a set of green Z-spiders, called the *frontier*, separates the extracted portion of the diagram from the unextracted part.

There are three basic extraction rules:

- **Phase:** Phases of frontier spiders are directly extracted as $R_z$ gates.
- **CZ:** Hadamard wires between frontier spiders are extracted as CZ gates.
- **Hadamard:** Hadamard wires where a frontier spider is only connected to a non-frontier spider are extracted as Hadamard gates.

**Example 4.** Figure 3 illustrates the three extraction rules. ※

If every spider in the frontier has at least two non-frontier neighbors, we can add wires of a frontier spider to another by placing a CX gate in the extracted circuit with the following effect:

1) **CX:** Extracting a CX gate with control $c$ and target qubit $t$ copies all Hadamard wires of the target frontier spider to the control qubit frontier spider. Double wires cancel each other.

**Example 5.** In Figure 4 a CX gate between control qubit $q_0$ and target $q_1$ is extracted. The two Hadamard wires of the frontier spider $q_1$ are copied to $q_0$ and cancel the existing Hadamard wires. After this CX gate, the rest of the diagram can be extracted, as shown on the right side. ※

In addition to this basic extraction, we also consider the extraction of multi-qubit gates as described in [4]. This extends the extraction to allow for multi-controlled phase gates of the form $C_{\mathrm{m}}P(\theta)$ with $m \geq 1$ control qubits.

## III. Neutral Atom Mapping

In this section, we introduce the neutral atom quantum computing platform and the corresponding mapping problem for *Neutral Atom* (NA) quantum computers.

### A. Neutral Atom Quantum Computing

NAs have emerged as a promising platform for universal quantum computing [12]–[15]. They offer a wide range of computational capabilities, including high-fidelity, long-range interactions between qubits, native multi-qubit gate support [15]–[17], and large-scale realizations [18], [19]. Additionally, Bluvstein *et al.* [20], [21] have demonstrated high-fidelity dynamic rearrangement and *shuttling* of qubit arrays during computation, allowing for arbitrary connectivity with some additional time overhead.

In recent years, the software community has increasingly focused on the NA platform, developing various solutions for NA-specific compilation tasks [5], [22]–[24]. Despite rapid experimental advancements, there is a risk that the available hardware may outpace the development of appropriate software solutions, preventing full utilization of the hardware's capabilities [25]. The role of the compiler is to translate high-level quantum algorithms into instructions executable by the hardware, utilizing available quantum gates and operations such as atom rearrangements.

### B. Hybrid Architecture

In neutral atom quantum computing, qubits are encoded in the electronic states of individual atoms, such as Rb, Sr, or Yb, which are trapped in optical tweezers [18]. These qubits are manipulated using laser beams, with single-qubit gates realized through state transitions driven by a combination of global and local lasers [13], [21]. Multi-qubit gates are implemented via the *Rydberg* blockade mechanism, which prevents the simultaneous excitation of two nearby atoms to a Rydberg state. This technique enables the creation of fast and high-fidelity CZ-gates [16], [21] on nearby atoms.

Additionally, qubits can be dynamically rearranged during computation by *shuttling* atoms with high fidelity, allowing for arbitrary qubit connectivity [21]. Experimentally, this is achieved using two types of optical traps: 1. *Spatial Light Modulator* (SLM) traps, which define single static trap sites, and 2. *Acousto-Optic Deflector* (AOD) traps. AOD traps are formed by overlapping two 1D AODs (in the x and y directions) to create a 2D grid of trap sites at their intersections [21], [26]. Atoms can be transferred between trap types, referred to as *loading* (SLM $\rightarrow$ AOD) and *storing* (AOD $\rightarrow$ SLM). Qubits are shuttled by moving AOD rows and columns, thereby rearranging the trapped atoms. However, this process is constrained by the fixed ordering of rows and columns, meaning they cannot be swapped [25].

### C. Hybrid Mapping

The mapping problem in NA quantum computing involves translating a quantum circuit into a sequence of operations executable by the hardware. Based on the capabilities of NAs,

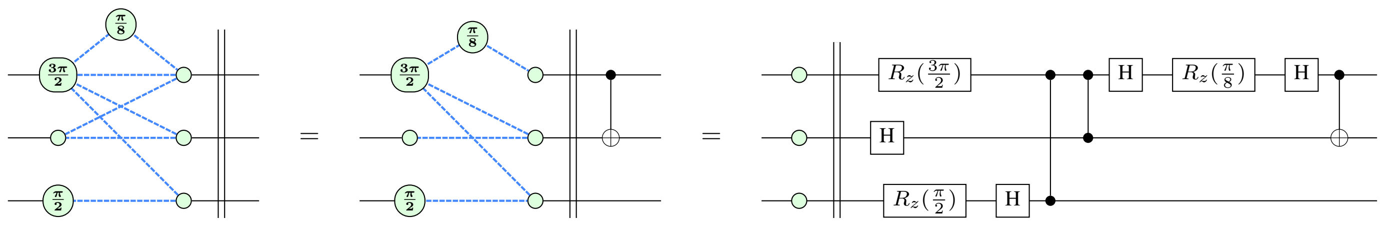


Figure 4. Full ZX circuit extraction of a graph-like diagram using a CX gate to rearrange the graph connections.

we encounter a dual mapping problem. First, in the *gate-based mapping* step, gates that are not trivially connected can be routed by modifying the qubit mapping using SWAP gate insertion [27], [28]. Second, in *shuttling-based mapping*, the qubit mapping remains unchanged while the connectivity graph is altered by moving atoms to new trap coordinates [21]. For shuttling, the special AOD constraint above of non-permuted AOD rows/columns has to be fulfilled. The goal is to minimize the number of SWAP gates and shuttling operations to reduce both the number of erroneous operations and the overall runtime, thereby minimizing decoherence [25].

Based on these considerations, several different approaches to the mapping problem have been developed. This includes NA-specific SWAP gate insertion [29] and shuttling-only based mapping using AODs to rearrange the qubits [22]–[24], [30], [31]. And, for the above described hybrid architecture, corresponding *hybrid mapping* strategies have been developed which aim to combine both capabilities to further optimize the result [5], [32].

For this work-in-progress, we consider the mapper from [5], which is available as an open-source project. It supports the following functionality used within this work:

- **Gate-based Mapping:** SWAP gate insertion to connect qubits that are not directly connected.
- **Shuttling-based Mapping:** Moving atoms next to each other to enable direct gate execution.
- **Multi-qubit Gate Support:** Mapping of multi-qubit gates (in particular multi-controlled phase gates $C_\mathrm{m}P(\theta)$).
- **Hybrid Mapping:** Automatic selection of the best mapping strategy based on the available hardware parameters.

We use this mapper due to its flexibility and applicability to different hardware architectures, but the ideas discussed in this work also generalize to other mapping strategies.

In the following, we provide a brief example to illustrate the potential improvements due to cross-compilation.

## IV. Considered Problem

To illustrate the potential of cross-compilation during ZX circuit extraction, let us revisit Figure 4 from Example 5. After adding the CX gate, the remaining ZX-graph can be directly extracted into a quantum circuit. The resulting circuit contains three entangling gates (2 CZ and one CX) and requires two-qubit gates between $q_0 \leftrightarrow q_2$ followed by $q_0 \leftrightarrow q_1$. Alternatively, another CX gate can be inserted between qubits two and three to further reduce edges in the ZX graph, then continue with the circuit extraction. This step and its result are shown in Figure 5, where the resulting circuit has four two-qubit gates (2 CZ and 2 CX) and requires connectivity between qubits $q_0 \leftrightarrow q_1$ and $q_1 \leftrightarrow q_2$.

This indicates that, in addition to the non-unique graph-like representation of the circuit [2], there is a certain "degree of freedom" during the extraction procedure. As a result, the extracted quantum circuits can have different numbers of (two-qubit) gates with varying connectivity.

For circuit extraction, a typical metric is the number of entangling gates, as they are a major source of error on available hardware [33]. Thus, the first circuit would be preferable as it requires three entangling gates compared to four in the second circuit.

However, the problem becomes more complex when we also consider the task of mapping the resulting circuit to a specific hardware architecture. In particular, consider connectivity 1, shown in Figure 6, which depicts two coupling graphs representing possible qubit connectivities. While the second circuit can be executed directly on the hardware, as all required qubit connections ($q_0 \leftrightarrow q_1$ and $q_1 \leftrightarrow q_2$) are directly available, the first circuit contains an unavailable two-qubit gate connection ($q_0 \leftrightarrow q_2$). As a result, additional SWAP gates need to be inserted (e.g., between $q_0$ and $q_1$), which is realized by three CX gates on hardware. In total, this results in $3 + 3 = 6$ two-qubit gates for the first circuit, making the second circuit the better choice with its four two-qubit gates. For the second connectivity of Figure 6, on the other hand, the first circuit is again the better option.

While in this simple example, it might be possible to circumvent the additional SWAP gate by permuting the initial assignment of circuit qubits to hardware qubits, it clearly illustrates the interplay between circuit extraction and hardware mapping. A circuit that might be preferable based on metrics commonly used for circuit extraction, such as the number of two-qubit gates, might result in problematic mapping configurations and, therefore, in suboptimal execution on the final hardware. Although the example only considered SWAP gate insertions, the general idea also applies to other mapping strategies, such as shuttling and even hybrid mapping. To avoid this suboptimal interplay, we propose a cross-compilation scheme in this work, where the extraction and mapping algorithms communicate iteratively to avoid compilation steps that might be suboptimal for the opposite party.

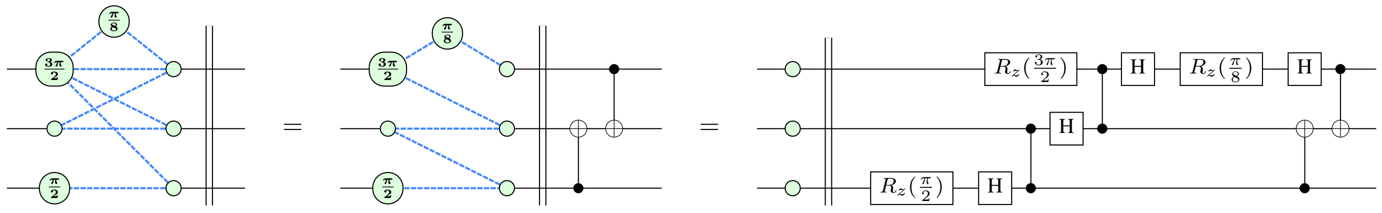


Figure 5. Alternative circuit extraction to Figure 4 with two CX gate insertions.

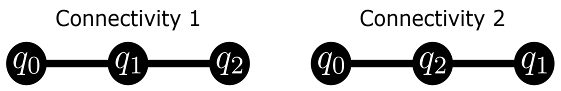


Figure 6. Example of two different qubit connectivities during the mapping step.

## V. Proposed Solution

In this section, we present a solution to address the issue of suboptimal compilation results arising from the disjoint use of circuit extraction and mapping algorithms.

### A. Solution Overview

Based on the example discussed above, we propose a *cross-compilation* scheme where the circuit extraction and mapping algorithms communicate to choose the optimal extraction step and iteratively construct the complete circuit. This idea is illustrated in Figure 7, with the circuit extraction algorithm on the left and the mapping algorithm on the right. At the beginning, or at any point during the extraction procedure, if multiple equivalent circuit extractions exist, they are "sent" to the mapping algorithm. These different circuits represent potential *extraction paths* for the extraction algorithm. The mapper analyzes these circuits to determine their suitability for the current hardware configuration. Based on a given metric, typically the number of two-qubit gates or some form of fidelity, the mapper selects the best option, performs the mapping, and reports the chosen extraction step back to the extraction algorithm. The extraction algorithm then continues along the reported path and restarts the feedback loop with the next extraction paths. Through this iterative back-and-forth communication, the entire graph is converted into a fully mapped quantum circuit.

In the following sections, we provide more details on the exact implementation and metrics used.

### B. Implementation Details

For the ZX-extraction phase, we utilize the "default" extraction algorithm as described in [10]. Since the extraction path choice can be non-unique, as illustrated above, we construct a path for each possible option. The extraction paths are generated using the default algorithm, which aims to minimize the number of CX gates required to continue the extraction. To increase the number of possible paths and, therefore, the search space, one can additionally consider circuit extractions with $k$ CX gates more than the minimum necessary, which we refer to as $k$-choice extraction. The parameter $k$ represents a hyperparameter, which allows a trade-off between search space exploration and algorithmic runtime.

Furthermore, we also incorporate the extension proposed in [4], which generalizes the extraction to include multi-controlled phase gates of the form $C_{\mathrm{m}}P(\theta)$ in comparison to only CZ and CX of the original default extraction algorithm. The possibility to extract multi-controlled phase gates provides additional extraction paths and might circumvent the problems described in [4] where the extraction of large gates sometimes resulted in worse circuit fidelity.

For the mapping phase, we employ the hybrid mapping algorithm from [5], which supports both SWAP gate insertions and shuttling operations for hybrid architectures. The optimal path is determined by mapping the extracted circuits to the current hardware configuration, including qubit placement and connectivity. The best path is selected based on the *approximate success probability* (ASP), as defined in [25]:

$$\mathrm{ASP} = \exp\left(-\frac{t_{\mathrm{idle}}}{T_{\mathrm{eff}}}\right)\prod_{i=1}^{N} F(O_i) \tag{1}$$

where $F(O_i)$ represents the fidelity of the $i$-th operation, $t_{\mathrm{idle}}$ is the idle time, and $T_{\mathrm{eff}} = \frac{T_1 T_2}{T_1+T_2}$ denotes the effective coherence time. The ASP is used to assess the optimal mapping capability (whether SWAP gate insertion or shuttling is preferable) and accounts for hardware parameters such as operation fidelities and coherence times. Consequently, the optimal extraction path is influenced not only by the hardware's structure and capabilities but also by real-time hardware parameters, which may be updated immediately before the mapping step.

The extraction algorithm is implemented in Python using the PyZX library [9], while the mapping algorithm is implemented in C++ and relies on the hybrid mapper in MQT-QMAP [5]. Communication between the two algorithms is facilitated through Python bindings created with the PyBind library [34].

The complete code, along with detailed documentation, will be made publicly available and integrated into the *Munich Quantum Toolkit* (MQT) [7].

## VI. Discussion

While this work-in-progress primarily introduces the initial concept of *cross-compilation* for ZX circuit extraction, this novel strategy holds significant promise for several reasons:

First, it represents a direct generalization of the conventional approach, where quantum circuits are extracted and mapped

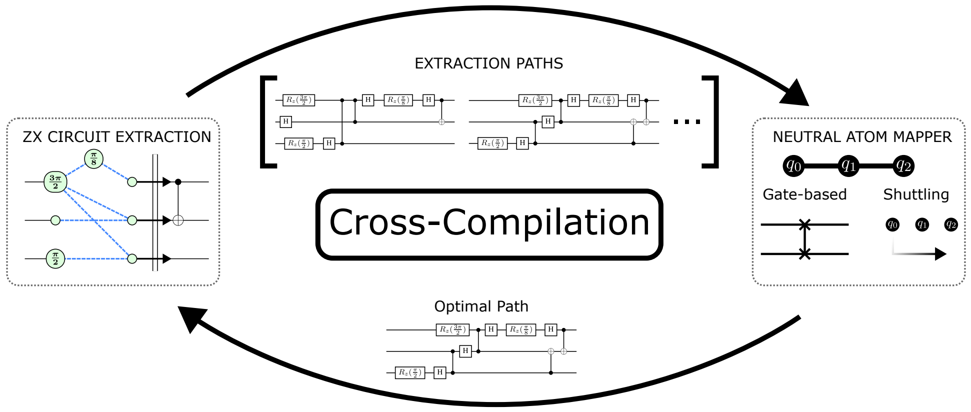


Figure 7. Overview of the cross-compilation strategy and the communication between circuit extraction and mapping.

separately. In particular, the traditional method of fully extracting the circuit before mapping it is a specific instance of our cross-compilation scheme, where only a single path (the fully extracted circuit) is provided to the mapper. Hence, the traditional approach sets a lower bound for extraction performance, and introducing iterative communication with local optimizations during extraction offers a promising avenue for improving compilation outcomes.

Second, as the approach is based on graph-like ZX-diagrams, it seamlessly integrates into the broader ZX-calculus framework. This allows it to benefit from sophisticated simplification rules, algorithms, and existing software frameworks like PyZX [9]. Consequently, it can take advantage of future enhancements in graph simplifications within the ZX community.

Third, while we illustrate the strategy with neutral atom hardware and multi-controlled phase gate extraction, the cross-compilation concept can be extended to other extraction algorithms and mapping tools. This includes more advanced ZX extraction algorithms based on heuristics for minimizing CX gate insertions [6] and T-gate optimized extraction [3]. Regarding mapping, available tools could be used to extract circuits favorable for various hardware architectures, such as neutral atoms, superconducting qubits, and trapped ions, employing different metrics like SWAP gate minimization or time-optimal mapping. Implementing cross-compilation requires only minor modifications to existing mappers, and using a common metric like the approximate success probability from Equation (1) offers a straightforward way to integrate existing mappers into the proposed strategy.

For these reasons, we believe that cross-compilation is an intriguing and promising strategy for ZX-based compilation. The next steps consist of evaluating the proposed concept and comparing it to already existing schemes—with the first results being expected until the IWQC2024.

## VII. Conclusion

In this work-in-progress, we introduced a novel circuit extraction and mapping scheme tailored for quantum computing based on neutral atom-based hardware using ZX-calculus. Rather than treating circuit extraction and mapping as separate tasks, we propose a *cross-compilation* approach in which the extraction algorithm first constructs possible extraction paths, which are then evaluated by the mapping algorithm. This method enables more informed decisions regarding the optimal extraction path, taking into account both the current hardware configuration and its physical parameters. We anticipate that this approach could enhance compilation results and make better use of hardware capabilities. Additionally, we plan to release the code as a reference implementation to facilitate adaptation of this approach for other researchers, enabling them to apply it to their own extraction and mapping algorithms and hardware setups.

## Acknowledgments

This work is partially supported by the German Federal Ministry of Education and Research (BMBF) under the funding program Quantum Technologies - From Basic Research to Market under contract number 13N16070. The authors acknowledge funding from the Munich Quantum Valley initiative (K5), which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. L.S. and R.W. acknowledge funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 101001318).

## References

[1] A. Javadi-Abhari *et al.*, *Quantum computing with Qiskit*, 2024. DOI: 10.48550/arXiv.2405.08810. arXiv: 2405.08810.

[2] R. Duncan, A. Kissinger, S. Perdrix, and J. v. d. Wetering, "Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus," en-GB, *Quantum*, vol. 4, p. 279, 2020, Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. DOI: 10.22331/q-2020-06-04-279.

[3] A. Kissinger and J. van de Wetering, "Reducing the number of non-Clifford gates in quantum circuits," *Physical Review A*, vol. 102, no. 2, p. 022 406, 2020. DOI: 10.1103/PhysRevA.102.022406.

[4] K. Staudacher, L. Schmid, J. Zeiher, R. Wille, and D. Kranzlmüller, *Multi-controlled Phase Gate Synthesis with ZX-calculus applied to Neutral Atom Hardware*, 2024. DOI: 10.48550/arXiv.2403.10864. arXiv: 2403.10864.

[5] L. Schmid, S. Park, S. Kang, and R. Wille, *Hybrid Circuit Mapping: Leveraging the Full Spectrum of Computational Capabilities of Neutral Atom Quantum Computers*, 2023. DOI: 10.48550/arXiv.2311.14164. arXiv: 2311.14164 [quant-ph].

[6] K. Staudacher, T. Guggemos, S. Grundner-Culemann, and W. Gehrke, "Reducing 2-qubit gate count for zx-calculus based quantum circuit optimization," in Proceedings 19th International Conference on *Quantum Physics and Logic,* Wolfson College, Oxford, UK, 27 June - 1 July 2022, vol. 394, Open Publishing Association, 2023, pp. 29–45. DOI: 10.4204/EPTCS.394.3.

[7] R. Wille *et al.*, "The MQT Handbook: A Summary of Design Automation Tools and Software for Quantum Computing," 2024. arXiv: 2405.17543, A live version of this document is available at https://mqt.readthedocs.io.

[8] R. Vilmart, "A Near-Optimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics," en, 2018.

[9] A. Kissinger and J. van de Wetering, "PyZX: Large Scale Automated Diagrammatic Reasoning," in Proceedings 16th International Conference on *Quantum Physics and Logic,* Chapman University, Orange, CA, USA., 10-14 June 2019, vol. 318, Open Publishing Association, 2020, pp. 229–241. DOI: 10.4204/EPTCS.318.14.

[10] M. Backens, H. Miller-Bakewell, G. de Felice, L. Lobski, and J. van de Wetering, "There and back again: A circuit extraction tale," en, *Quantum*, vol. 5, p. 421, 2021. DOI: 10.22331/q-2021-03-25-421.

[11] W. Simmons, "Relating measurement patterns to circuits via pauli flow," *Electronic Proceedings in Theoretical Computer Science*, vol. 343, pp. 50–101, 2021. DOI: 10.4204/eptcs.343.4.

[12] M. Saffman, T. G. Walker, and K. Mølmer, "Quantum information with Rydberg atoms," *Reviews of Modern Physics*, vol. 82, no. 3, pp. 2313–2363, 2010. DOI: 10.1103/RevModPhys.82.2313.

[13] M. Saffman, "Quantum computing with neutral atoms," *National Science Review*, vol. 6, no. 1, pp. 24–25, 2019. DOI: 10.1093/nsr/nwy088.

[14] L. Henriet *et al.*, "Quantum computing with neutral atoms," *Quantum*, vol. 4, p. 327, 2020. DOI: 10.22331/q-2020-09-21-327.

[15] T. M. Graham *et al.*, "Multi-qubit entanglement and algorithms on a neutral-atom quantum computer," *Nature*, vol. 604, no. 7906, pp. 457–462, 2022. DOI: 10.1038/s41586-022-04603-6.

[16] S. J. Evered *et al.*, "High-fidelity parallel entangling gates on a neutral-atom quantum computer," *Nature*, vol. 622, no. 7982, pp. 268–272, 2023. DOI: 10.1038/s41586-023-06481-y. arXiv: 2304.05420.

[17] H. Levine *et al.*, "Parallel Implementation of High-Fidelity Multiqubit Gates with Neutral Atoms," *Physical Review Letters*, vol. 123, no. 17, p. 170 503, 2019. DOI: 10.1103/PhysRevLett.123.170503.

[18] D. Barredo, S. de Léséleuc, V. Lienhard, T. Lahaye, and A. Browaeys, "An atom-by-atom assembler of defect-free arbitrary two-dimensional atomic arrays," *Science*, vol. 354, no. 6315, pp. 1021–1023, 2016. DOI: 10.1126/science.aah3778.

[19] L. Pause *et al.*, "Supercharged two-dimensional tweezer array with more than 1000 atomic qubits," *Optica*, vol. 11, no. 2, pp. 222–226, 2024. DOI: 10.1364/OPTICA.513551.

[20] D. Bluvstein *et al.*, "Logical quantum processor based on reconfigurable atom arrays," *Nature*, pp. 1–3, 2023. DOI: 10.1038/s41586-023-06927-3.

[21] D. Bluvstein *et al.*, "A quantum processor based on coherent transport of entangled atom arrays," *Nature*, vol. 604, no. 7906, pp. 451–456, 2022. DOI: 10.1038/s41586-022-04592-6.

[22] D. B. Tan, D. Bluvstein, M. D. Lukin, and J. Cong, "Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors," *Quantum*, vol. 8, p. 1281, 2024. DOI: 10.22331/q-2024-03-14-1281.

[23] D. B. Tan, W.-H. Lin, and J. Cong, *Compilation for Dynamically Field-Programmable Qubit Arrays with Efficient and Provably Near-Optimal Scheduling*, 2024. arXiv: 2405.15095.

[24] H. Wang *et al.*, *Q-Pilot: Field Programmable Quantum Array Compilation with Flying Ancillas*, 2023. DOI: 10.48550/arXiv.2311.16190. arXiv: 2311.16190.

[25] L. Schmid *et al.*, "Computational capabilities and compiler development for neutral atom quantum processors—connecting tool developers and hardware experts," *Quantum Science and Technology*, vol. 9, no. 3, p. 033 001, 2024. DOI: 10.1088/2058-9565/ad33ac.

[26] J. Beugnon *et al.*, "Two-dimensional transport and transfer of a single atomic qubit in optical tweezers," *Nature Physics*, vol. 3, no. 10, pp. 696–699, 2007. DOI: 10.1038/nphys698.

[27] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping Quantum Circuits to IBM QX Architectures Using the Minimal Number of SWAP and H Operations," in *Proceedings of the 56th Annual Design Automation Conference 2019*, Association for Computing Machinery, 2019, pp. 1–6. DOI: 10.1145/3316781.3317859.

[28] A. Cowtan *et al.*, "On the Qubit Routing Problem," in *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, vol. 135, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 5:1–5:32. DOI: 10.4230/LIPIcs.TQC.2019.5. eprint: 1902.08091.

[29] J. M. Baker *et al.*, "Exploiting Long-Distance Interactions and Tolerating Atom Loss in Neutral Atom Quan-

tum Architectures," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 818–831. DOI: 10.1109/ISCA52012.2021.00069.

[30] S. Brandhofer, I. Polian, and H. P. Büchler, "Optimal Mapping for Near-Term Quantum Architectures based on Rydberg Atoms," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–7. DOI: 10.1109/ICCAD51958.2021.9643490.

[31] H. Wang *et al.*, *FPQA-C: A Compilation Framework for Field Programmable Qubit Array*, 2023. DOI: 10.48550/arXiv.2311.15123. arXiv: 2311.15123.

[32] N. Nottingham *et al.*, *Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures*, 2023. DOI: 10.48550/arXiv.2307.14996. arXiv: 2307.14996.

[33] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018. DOI: 10.22331/q-2018-08-06-79.

[34] W. Jakob, J. Rhinelander, and D. Moldovan, *Pybind11 — seamless operability between c++11 and python*, https://github.com/pybind/pybind11, 2024.