Multi-controlled Phase Gate Synthesis with ZX-calculus applied to Neutral Atom Hardware

Korbinian Staudacher[1]*   Ludwig Schmid[2]   Johannes Zeiher[3,4,5]
Robert Wille[2,6]   Dieter Kranzlmüller[1,7]

*Korbinian.Staudacher@nm.ifi.lmu.de

[1]MNM-Team, Ludwig-Maximilians-Universität München, 80538 Munich, Germany
[2]Chair for Design Automation, Technical University of Munich, 80333 Munich, Germany
[3]Fakultät für Physik, Ludwig-Maximilians-Universität München, 80799 Munich, Germany
[4]Max-Planck-Institut für Quantenoptik, 85748 Garching, Germany
[5]Munich Center for Quantum Science and Technology (MCQST), 80799 Munich, Germany
[6]Software Competence Center Hagenberg GmbH (SCCH), 4232 Hagenberg im Mühlkreis, Austria
[7]Leibniz Supercomputing Centre (LRZ), 85748 Garching, Germany

Quantum circuit synthesis describes the process of converting arbitrary unitary operations into a gate sequence of a fixed universal gate set, usually defined by the operations native to a given hardware platform. Most current synthesis algorithms are designed to synthesize towards a set of single-qubit rotations and an additional entangling two-qubit gate, such as CX, CZ, or the Mølmer–Sørensen gate. However, with the emergence of neutral atom-based hardware and their native support for gates with more than two qubits, synthesis approaches tailored to these new gate sets become necessary. In this work, we present an approach to synthesize (multi-) controlled phase gates using ZX-calculus. By representing quantum circuits as graph-like ZX-diagrams, one can utilize the distinct graph structure of diagonal gates to identify multi-controlled phase gates inherently present in some quantum circuits even if none were explicitly defined in the original circuit. We evaluate the approach on a wide range of benchmark circuits and compare them to the standard Qiskit synthesis regarding its circuit execution time for neutral atom-based hardware with native support of multi-controlled gates. Our results show possible advantages for current state-of-the-art hardware and represent the first exact synthesis algorithm supporting arbitrary-sized multi-controlled phase gates.

1 Introduction

Compiling and optimizing quantum algorithms towards hardware-specific constraints is indispensable to efficiently use currently available noisy quantum hardware with limited gate fidelities and coherence times. An important step of the compilation process is quantum circuit synthesis, converting arbitrary unitary operations to gate sequences natively supported by the hardware. State-of-the-art synthesis algorithms, such as [25,47], are often focused on a superconducting hardware setting and synthesize towards singular two-qubit gates, e.g., CX, and single-qubit gates. Such synthesis algorithms are less preferable for other hardware architectures, for instance, when gates acting on three or more qubits can be executed natively without decomposition, resulting in a reduced execution cost.

In this work, we propose an approach to synthesize quantum circuits towards single qubit gates and arbitrary-sized multi-controlled phase gates $C_nP(\varphi)$. To this end, we make use of the representation of a quantum circuit as a graph-like ZX-diagram where we can use powerful rewrite rules of the ZX-calculus to simplify diagrammatic structures [15]. This approach has shown to be a useful tool for tasks like hardware-agnostic circuit optimization or equivalence checking [23,39,49]. We show that $C_nP(\varphi)$ have a distinct representation in graph-like ZX-diagrams as a combination of so-called phase gadgets, which occur naturally in the diagrams when using a simplification strategy proposed in [23]. By modifying an

existing extraction algorithm from [4] to translate graph-like diagrams back to quantum circuits, we can specifically optimize towards extracting phase gadget combinations corresponding to $C_nP(\varphi)$ gates.

The benefit and potential of the resulting approach are shown by synthesizing gate functionality for the recently emerging neutral atoms platforms [17, 20, 32, 41–44]. Besides dynamic connectivity with atom rearrangements [7, 8, 46] and favorable properties regarding scalability and large-scale control [6, 8, 19, 35, 38], this technology offers native support for multi-controlled gates such as C_nP, and CZ_n [12, 14, 16, 21, 22, 34]. We integrate our extraction scheme into a full gate synthesis and optimization process and compare total execution times on hardware against Qiskit synthesis routines, considering current state-of-the-art parameters. Our results show promising advantages in the form of reduced execution times on different benchmark circuits.

The paper is structured as follows: In the first part, we give a basic introduction to ZX-calculus, including graph-like ZX-diagram simplification, and show how multi-controlled phase gates can be identified and extracted from the diagrams. In the second part, we focus on the application of the proposed approach to neutral-atom-specific gate synthesis and discuss its effect on the execution time.

2 Related work

So far, algorithms supporting the synthesis of gates acting on more than two qubits are mostly centered around the generation of Toffoli gates. Ref. [18] introduces a synthesis algorithm for classical logic reversible functions using multi-control Toffoli gates and there exist algorithms for synthesizing towards universal Toffoli gate sets [3], even with optimal numbers of Toffoli gates [33]. The synthesis of multi-controlled phase gates is less studied. Ref. [57] proposes an optimal synthesis algorithm, but restricted to diagonal unitaries as an input. For universal circuits, a recent framework for neutral atom systems [37] is able to synthesize circuits with *CCZ* gates. However, the synthesis process is based on non-exact numerical optimization procedures and does not consider more than three-qubit gates or arbitrary rotations.

3 Preliminaries

In this section we introduce the ZX-calculus fundamentals and describe how graph-like diagrams can be simplified and extracted to quantum circuits, which represents the basis of our synthesis approach. We only give a brief overview of ZX-calculus, for a more detailed introduction we refer to [13, 15, 56].

3.1 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 ZX-diagram. There are two types of spiders: The green Z-spiders and the red X-spiders. Spiders can be parametrized with an angle $\alpha \in [0, 2\pi)$ and correspond to two-dimensional matrices in Hilbert space:

Spiders can have any number of ingoing and outgoing wires and 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
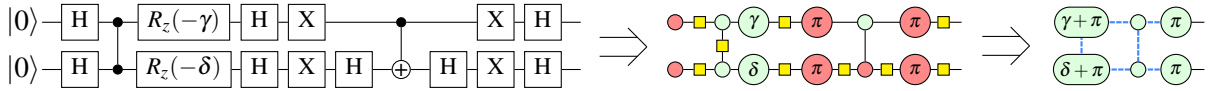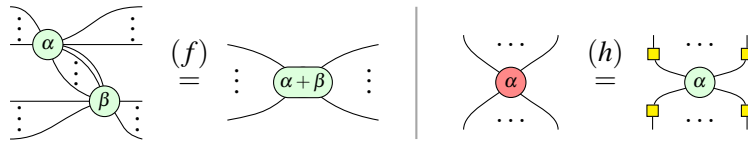
Figure 1: Translation of a two-qubit Grover search into a graph like ZX-diagram. Gates are replaced by their ZX-calculus counterpart and the diagram is made graph-like by repeated application of (f) and (h).

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, and wires exiting to the right are called output wires, with adjacent spiders defined as outputs O. We refer to the set of spiders $v \in I \cup O$ as *boundary spiders* and the complementary set of spiders $v \in V \setminus (I \cup O)$ as the *interior spiders*. The complements of the inputs and the outputs are defined as $\bar{I} = V \setminus I$ and $\overline{O} = V \setminus O$ respectively. We can write any quantum circuit as ZX-diagram by replacing gates with equivalent diagrams and use rules from ZX-calculus to modify them without changing the linear map. For instance, the following rules hold:

The fusion rule (f) allows to merge spiders of the same color together if they are connected by at least one normal wire and (h) allows to change 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 [53].
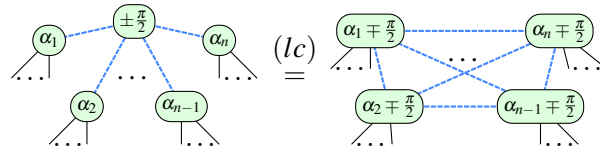
3.1.1 Graph-like diagrams

In this work, we consider the class of *graph-like* ZX-diagrams as introduced in [15], which allow us to represent any quantum computation as a graph of parametrized green Z-spiders and Hadamard wires. In those diagrams we represent Hadamard wires between spiders as dashed blue line instead of the yellow box for easier visualization. One can transform any ZX-diagram into an equivalent graph-like ZX-diagram by repeatedly applying standard ZX-rules [15] (c.f. Figure 1). This formalism provides a link between quantum computing and graph theory since the entire computation is captured by the *underlying graph* spanned by Hadamard wires, combined with phases of Z-spiders. Moreover, graph-like diagrams can be directly interpreted as measurement patterns in the model of measurement-based quantum computing (MBQC) [4, 10].

3.1.2 Diagram simplification

We can rewrite graph-like diagrams into equivalent simplified versions (i.e., decreasing the number of spiders or wires), by using graph-theoretic rewrite rules as shown in [15]:

Local complementation   Given an undirected graph G, local complementation on a vertex v (written $G \star v$) consists of flipping the edges between the neighbors of v. That is, after local complementation, every pair of neighbors of v is connected iff it was not connected before. In graph-like ZX-diagrams, we can use a rewrite rule based on local complementation (lc) to eliminate spiders with a phase of $\pm\frac{\pi}{2}$:

Pivoting A Pivot $G \wedge uv$ consists of three local complementations $(G \star u \star v \star u)$ applied on a pair of neighboring vertices u, v. We can use a similar rewrite rule (p) in graph-like ZX-diagrams to eliminate pairs of spiders with phase 0 or π:
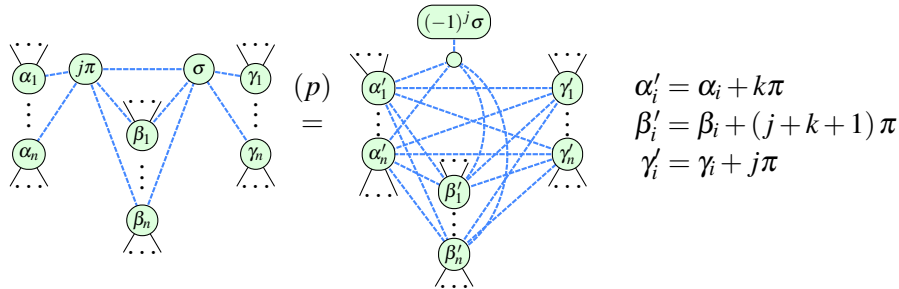

$$\alpha_i' = \alpha_i + k\pi$$
$$\beta_i' = \beta_i + (j + k + 1)\pi$$
$$\gamma_i' = \gamma_i + j\pi$$

By repeatedly applying those rules one can eliminate all interior spiders with phase $\pm\frac{\pi}{2}$ and every pair of interior spiders with phase 0 or π [15].

Phase gadgets One can further simplify ZX-diagrams with a slightly modified version of the pivot rule if we allow one spider of a pair to have a non-Clifford phase σ [23]. The non-Clifford spider does not get removed but is transformed into a so called *phase gadget*:

$$\alpha_i' = \alpha_i + k\pi$$
$$\beta_i' = \beta_i + (j + k + 1)\pi$$
$$\gamma_i' = \gamma_i + j\pi$$

In graph-like ZX-diagrams a phase gadget consists of a "top" spider exclusively connected to a phaseless "root" spider connected to other spiders. Simplifying graph-like diagrams with all three rules, we obtain a diagram where spiders either have a non-Clifford phase, are part of a phase gadget or a boundary.

3.1.3 Gflow in graph-like diagrams

Gflow is a graph-theoretic property for measurement patterns defined on labeled open graphs (G, I, O, λ), where $G = (V, E)$ is an undirected graph with vertices V and edges E, $I \subseteq V$, $O \subseteq V$ are the set of inputs resp. outputs, and λ is a labeling function assigning each vertex a measurement plane of the Bloch sphere in $\{XY, XZ, YZ\}$ [11]. A labeled open graph has gflow if there exists a map $g : \overline{O} \to \mathscr{P}(\overline{I})$ and a partial order $\prec$ over V, s.t. for all $v \in \overline{O}$:
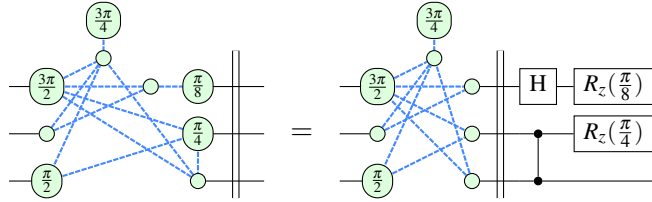
- If $w \in g(v)$ and $v \neq w$, then $v \prec w$.
- If $w \in Odd(g(v))$ and $v \neq w$, then $v \prec w$.
- If $\lambda(v) = XY$, then $v \notin g(v)$ and $v \in Odd(g(v))$.
- If $\lambda(v) = XZ$, then $v \in g(v)$ and $v \in Odd(g(v))$.

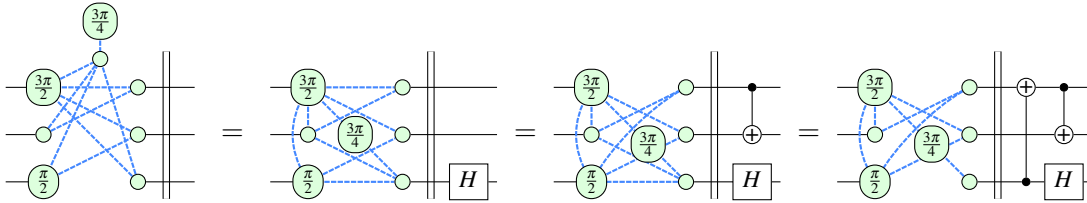- If $\lambda(v) = YZ$, then $v \in g(v)$ and $v \notin Odd(g(v))$.

In graph-like diagrams, we interpret the underlying graph as a labeled open graph with phase gadgets corresponding to YZ measurements[1] and other spiders corresponding to XY measurements [4]. Since the initial graph-like diagrams obtained from quantum circuits (as in Figure 1) have gflow [15] and all above rules preserve gflow [4], the simplified diagrams have gflow as well.

3.1.4 Circuit extraction

Extracting quantum circuits back from graph-like ZX-diagrams where the circuit has only as many qubits as there are outputs/inputs, is so far only possible in polynomial time if the underlying graph has some kind of flow [4, 48]. Here, we give a brief overview of the extraction algorithm for graph-like ZX-diagrams with gflow as described in [4]. The algorithm extracts a quantum circuit from a ZX-diagram by taking suitable parts of the diagram and creating their equivalent representation as a quantum gate within the circuit at the corresponding position. These parts are then removed from the diagram, extracting one gate at a time, until only the inputs and outputs of the diagram remain. During the process, a set of green Z-spiders called the *frontier* separates the extracted part of the diagram from the unextracted part. Phases of frontier spiders can be directly extracted as R_z gates, and Hadamard wires between frontier spiders as CZ gates. Furthermore, Hadamard wires where a frontier spider w is exclusively connected to a non-frontier spider v can be extracted as Hadamard gates with v replacing w in the frontier:

If every spider in the frontier has at least two non-frontier neighbors we can add wires of a frontier spider to the wires of another one by placing a CX gate on the extracted circuit. If all neighbors are measured in the XY plane, gflow ensures that there exists a combination of additions so that there remains a frontier spider with only a single neighbor. We can obtain such a combination by applying Gaussian elimination on the biadjacency matrix between the frontier vertices and their neighbors. Otherwise, if there are YZ-measured neighbors, we can transform them into XY measurements by applying a pivot on the neighbor and a connected frontier spider:

By repeating these procedures, we can transform the entire diagram into a quantum circuit.

4   Extracting controlled phase gates from graph-like ZX-diagrams

The process of transforming quantum circuits to graph-like ZX-diagrams, simplifying them, and re-extracting circuits can already be seen as an implicit synthesis algorithm to the gate set $\{R_z, H, CZ, CX\}$.

---

[1]The root spider is labeled as YZ measurement, while the top spider corresponds to a measurement effect which is omitted from the underlying graph.

Given the requirements of neutral atom platforms, it may be desirable to incorporate two- and multi-controlled phase gates of arbitrary rotations into this gate set. We first show how such gates are represented in graph-like ZX-diagrams, then how we incorporate this finding into the extraction algorithm.

4.1   Graph-like representation of controlled phase gates

The (multi-) controlled phase gate is a diagonal gate, meaning all non-zero entries of its corresponding matrix in the Z-basis are on its diagonal. Such gates can be represented as a semi-Boolean function $f : \{0,1\}^n \to \mathbb{C}$ which assigns a complex number to each basis state. Ref. [26] shows, that any semi-Boolean function $f(b) = a_b$ with $a_b \in \mathbb{C}$ and $b \in \mathbb{B}^n$ can be expressed in ZX-calculus as follows:

with

$c_i = \;\mid\;$ if $c_i = 1$, and

$c_i = \;$ if $c_i = 0$

$$\alpha_c = \frac{-1}{2^{n-1}} \sum_{b \in \mathbb{B}^n} f(b) \chi(b,c) \tag{1}$$

The part in the dashed box is repeated for every Boolean vector c in $\mathbb{B}^n$ and the grey box decomposes into n subdiagrams either connecting the corresponding lower and upper wire with a normal wire if the i-th element of the Boolean vector is 1, or disconnecting them if it is 0. Further, the phase α_c can be obtained by the formula on the right, where $\chi(b,c) = (-1)^{b \cdot c}$ corresponds to a parity function with $\cdot$ being the inner product: If b and c overlap in an odd number of elements it returns -1, else 1. This rule yields a combination of phase gadgets, and when applying the color change rule on the middle red spider, we obtain the same graph structures as introduced in the previous section. To model an n-controlled phase gate $C_nP(\varphi)$ as a semi-Boolean function, we take α as an all-zero vector of length 2^n except for its last entry being φ. Following Equation (1), one can transform the function to a ZX-diagram which has $2^n - 1$ phase gadgets split up into $\binom{k}{n}$ phase gadgets for $k \in \{1, \dots n\}$. For instance, the two and three-qubit-controlled phase gates have the following representation:

For arbitrary-sized multi-controlled phase gates, this generalizes to the following theorem:

Theorem 1 (Multi-controlled phase gates). *Let $\binom{S}{k}$ denote the set of all k-combinations of a set S and $PG(\alpha, N)$ denote a phase gadget with phase α connected to neighbors N which are empty Z-spiders. A n-qubit controlled phase gate $C_nP(\varphi)$ is equivalent to a graph-like ZX-diagram with outputs $O, |O| = n$ having a phase of α and phase gadgets[2]*

$$\prod_{k=2}^{n} \prod_{s \in \binom{O}{k}} PG((-1)^{k+1}\alpha, s), \qquad \alpha = \frac{\varphi}{2^{n-1}}$$

Proof. A graphical proof is given by the ZX representation of the diagonal gate and the corresponding proof in [26], a combinatorial proof can be found in [2]. We give an alternative combinatorial proof in Appendix B.   □

---

[2]Note, that the product notation here corresponds to the composition ∘ of ZX-diagrams.

4.2   Adaption of the extraction algorithm

We adapt the algorithm described in 3.1.4 by including an additional C_nP gate extraction step between CZ and R_z extraction. For that, we carry out a pattern match on the phase gadgets which are exclusively connected to the outputs, i.e., the frontier. If we find a graph structure as described in Theorem 1 for n frontier spiders, we take the phase of the gadget which is connected to all n spiders as the desired phase α if n is odd, or $-\alpha$ if n is even, and adjust the phases of all other gadgets using the following two rewrites:

$$(4)$$

$$(5)$$

With Equation (4), we extract the unwanted part of an output phase as R_z gate, and with Equation (5), we split up phase gadgets into a part with the desired phase and another gadget so that the sum of the phases yields the original one. Both rewrites are sound in ZX-calculus: The first corresponds to an application of the fusion rule as mentioned in Section 3 and the second is a reversed version of the gadget fusion rule as shown in [23, Section D]. With adjusted phases, we extract the entire graph structure by removing it from the diagram and placing a $C_nP(\varphi)$ gate with $\varphi = \alpha \cdot 2^{n-1}$ on the circuit.

We can extend this procedure by also allowing the extraction of graph structures where some phase gadgets required for a C_nP extraction are missing in the diagram. Consider the following example, where we are initially missing two 2-ary phase gadgets with $-\frac{\pi}{4}$ to complete a $C_2P(\pi)$ structure:

$$(6)$$

By adding pairs of phase gadgets with opposite phases corresponding to the identity, we complete the required graph structure to extract the gate. Some inserted gadgets then remain in the diagram and are extracted later. If we always take the gadget with the most neighbors, complete the diagram to match a C_nP gate, and extract it, every phase gadget will get extracted as part of a C_nP gate at some point and we can entirely omit YZ spider eliminations via pivoting.

4.2.1   Preservation of gflow

For a complete translation of graph-like diagrams into quantum circuits, it is essential that all operations preserve gflow on the diagram. The rewrites of Equations 3-7 essentially reduce to deletions and insertions of phase gadgets, i.e., YZ measurements, connected to only outputs. Since the original extraction algorithm preserves gflow and it has been shown in [4, Lemma 3.4.] that the deletion of arbitrary YZ measurements preserves gflow, the same remains to be shown for the insertion case:

Lemma 1 (Insertion of YZ measurements on outputs). *Let $(g, \prec)$ be a gflow for (G, I, O, λ) and let $W \subseteq O$. Then (G', I, O, λ'), where $G' = (V', E')$ with $V' = V \cup \{x\}$, $\lambda'(x) = YZ$ and $E' = E \cup \{(x, w) | w \in W\}$ has a gflow.*

Proof. We provide the detailed proof in Appendix A.   □

4.2.2 Time complexity

The time complexity of the proposed approach in terms of elementary graph operations depends on whether we allow additional insertions of phase gadgets or not. Let k denote the number of spiders in a diagram and n the number of outputs:

- If we do not allow additional insertions of phase gadgets, we have approximately the same runtime as the original algorithm, namely $O(n^2k^2 + k^3)$ which is summed from the runtime for Gaussian elimination $O(n^2k)$, pivoting YZ measurements $O(k^2)$ and k steps in total [4]. Additionally, for our approach, we have to split at most k phase gadgets at each step, which adds another $O(nk)$ term to the elementary graph operations. Yet, this term gets absorbed by the complexity of the Gaussian elimination.

- If we allow phase gadget insertions to complete the graph structures corresponding to a C_nP gate, the complexity essentially becomes $O(2^{n+1}k)$. This is because, in the worst case, we would complete structures where there is only a single phase gadget connected to all n outputs, and we need to add $2 \cdot 2^n - n - 2$ additional gadgets. We want to emphasize that this worst-case complexity is unlikely to occur in practice. Yet, for larger circuits, it may be useful to limit the size of extractable C_nP gates to a constant.

5 Neutral Atom Circuit Synthesis

In the following, we want to apply the proposed extraction scheme to circuit synthesis for neutral atom (NA)-based hardware due to their native support of C_nP gates. Therefore, we briefly introduce the hardware capabilities [44], embed our proposed scheme into a complete synthesis procedure, and evaluate the effect of the C_nP extraction regarding the circuit execution time in comparison to Qiskits internal synthesis algorithm.

5.1 Neutral Atom Background

For NA-based quantum computers, qubit registers are realized by placing single atoms in optical dipole traps created by laser beams, referred to as optical lattices or optical tweezers. While arbitrary atom arrangements are possible, we assume a rectangular grid as illustrated in Figure 2. The qubit states can be encoded in long-lived internal atomic states such as hyperfine or nuclear spin states. Commonly employed atomic species include alkali or alkaline-earth(-like) atoms such as Rb and Sr, which provide suitable internal states with long coherence times. Gates are realized with specific laser pulses on the atoms using uniform global beams, with the possibility of addressing a whole register or individual qubits [17, 27]. Multi-qubit gates are based on the long-range interaction between close-by atoms excited to high-lying Rydberg states [16, 21, 28, 32, 34, 41, 42]. There exist different protocols to realize both single- and multi-qubit gates, and the preferred implementation depends on many parameters such as the chosen atom species, the respective qubit encoding, and the experimental setup. In this work, we focus on individually addressable $C_nP(\varphi)$ gates between neighboring atoms as a generalization of the often implemented CZ gate, which can be realized by tuning the accumulated phase during the Rydberg interaction to an arbitrary angle φ instead of π [28]. This might even result in improved gate times and fidelity due to the shorter time of the atom spent in the Rydberg state [16]. Regarding single-qubit gates, we assume a scheme between fast, individually addressable AC Stark shift beams, realizing local R_z rotations and slow, globally addressing microwave pulses performing a rotation about an axis in the
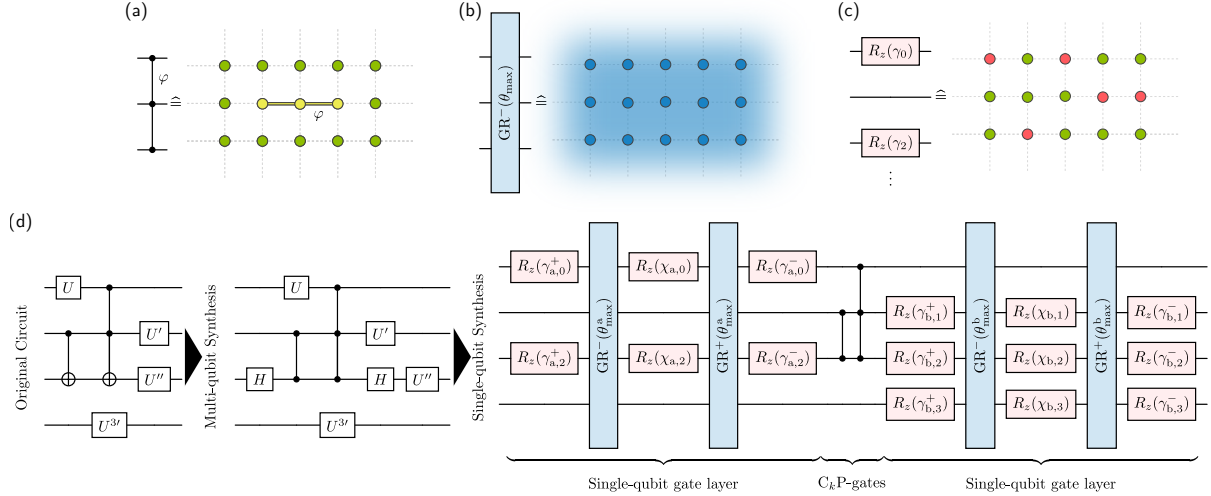
Figure 2: Illustration of the neutral atom gate capabilities and the process of synthesizing and scheduling quantum circuits to the hardware. **(a)** Native (multi-) controlled phase gates ($C_nP(\varphi)$), here shown for three qubits. **(b)** Global single-qubit rotations in the XY-plane. **(c)** Individually addressable Z-rotations ($R_z(\gamma)$). **(d)** Synthesis to alternating single- and multi-qubit layers: First, the synthesis of multi-qubit gates to C_nP gates. Second, the synthesis and scheduling of single-qubit gates into global XY-rotations and individually addressable Z-rotations according to the transversal decomposition of [36].

XY-plane for all qubits simultaneously. In particular, we assume the following gate definitions, where single-qubit gates are equivalent to the ones from [36] with GR as global XY-rotations applied to all n qubits, and $\hat{Y}$ being the Pauli-Y matrix:

$$C_nP(\varphi) \equiv \text{diag}(1, \ldots, 1, e^{i\varphi}), \quad R_z(\gamma^\pm) \equiv \text{diag}(e^{-i\gamma^\pm/2}, e^{-i\gamma^\pm/2}), \quad GR(\theta_{\max}) \equiv \exp\left(-i\frac{\theta}{2}\sum_{i=1}^{n}\hat{Y}_i\right) \quad (7)$$

In this setting, sets of arbitrary but simultaneous single-qubit gates on different qubits can be converted into two global illuminations interleaved with single-qubit Z-rotations. On unused qubits, the two complementary global rotations cancel out, effectively applying an identity operation. To convert single-qubit gates to this setting, we consider the transversal decomposition scheme introduced in [36] which is optimal in terms of global pulse time. An illustration of the gate capabilities and how to synthesize the respective single- and multi-qubit gates is shown in Figure 2.

5.2 Related Work

Recently, there has been a fast development of NA-specific compilation methods [5, 9, 30, 36, 37, 45, 51, 52, 54, 55] focusing almost exclusively on the mapping and scheduling tasks within the compilation. The only exceptions are [36], proposing the single-qubit synthesis also used in this work, and the Geyser framework [37] using numerical optimization to introduce additional CCZ gates. Nevertheless, both neglect the capability of NAs to natively execute controlled gates with arbitrary phases and, furthermore, the ability to directly execute multi-controlled gates for more than one control qubit. Although convenient, this neglects potentially shorter or simpler circuits using these specific capabilities of NAs. Therefore, in the following, we evaluate five different synthesis schemes to compare the commonly used naive synthesis algorithms with an NA-specific synthesis employing the ZX-approach of Section 4.2.

5.3 Evaluation Setup

The total gate synthesis process consists of the two steps illustrated in Figure 2: First, the synthesis of the multi-qubit gates into C_nP gates and arbitrary single-qubit rotations, and second, the synthesis and scheduling of single-qubit gates into the alternating global vs. local scheme, resulting in the fully synthesized circuit containing only gates from the native gate set of Equation (7). For the first step, we consider the following five schemes:

1. **Qiskit-default:** Circuits are converted into the Qiskit-supported native gate set of $\{U_3, CZ\}$ using the internal `transpile` function and setting the optimization level to three. This approach uses Qiskit internal schemes to decompose gates with more than two-qubit gates.

2. **No-decomp:** The Qiskit decomposition introduces a large overhead that can be bypassed for NAs. For better comparison, we propose this scheme which synthesizes to $\{U_3, C_nZ\}$ by replacing all (multi-) controlled gates by their $C_nZ\}$ equivalent and using the Qiskit-default approach for the single qubit gates.

3. **ZX-default:** The circuit is converted into a graph-like ZX diagram, and the default extraction algorithm of PyZX [24] is used to recreate a circuit.

4. **ZX-no-insert:** Similar to the default but the extraction scheme from Section 4.2 is used to synthesize C_nP gates.

5. **ZX-with-insert:** In addition to ZX-no-insert, we allow the insertion of additional phase gadgets, resulting in possibly more and larger phase gate extractions.

Since the ZX extraction sometimes produces redundant gates, we additionally apply a basic gate cancellation algorithm afterwards. In the second step, the transverse decomposition according to [36] is used to synthesize the single qubit gates. In this scheme, $\theta_{\max} \equiv \max_i \theta_i$ is defined as the maximum of the first Euler angle θ_i of any single qubit rotation in this layer. According to the discussion in [36], the total gate execution time scales linearly in this angle with the maximal duration at $\theta_{\max} = \pi$.

As for many single-qubit gates the actual moment of execution is not unique, it can be added to different layers. We thus use an additional greedy optimization step, not performed in [36], to check the possible positions of the single-qubit gates and assign them such as to minimize the overall $\theta_{\max}$. In particular, a gate with Euler angle θ is preferably assigned to a layer with $\theta_{\max} > \theta$, allowing the gate to be executed without increasing the gate time. As evaluation metrics, we compute both simple gate counts and the total circuit execution time T by scheduling the gates according to the illustration in Figure 2. We sum individual gate times, where we assume the gate execution time increases linearly with the rotation angle as follows:

$$T = \sum_{i=0}^{d} \frac{|max_\gamma(R_z(\gamma), i)|}{\pi} 100ns + \sum_{GR(\theta_{\max})} \frac{|\theta_{\max}|}{\pi} 100\mu s + \sum_{C_1P(\varphi)} \frac{|\varphi|}{\pi} 100ns + \sum_{C_nP(\varphi), n>1} \frac{|\varphi|}{\pi} 400ns \quad (8)$$

Here d denotes circuit depth, meaning we assume full parallel execution of the R_z if possible by taking the maximum angle of each layer. Multi-qubit gates are assumed to be executed in a sequential way. The gate times are $0.1\,\mu s$ for the R_z [46] and the $CP(\varphi = \pi)$ gate [31]. For all higher-weight controlled phase gates $C_{\geq 2}P$ we assume $0.4\,\mu s$ for $\varphi = \pi$. The dominating factor for circuit execution time are the slow global illuminations GR with $100\,\mu s$ [46]. Therefore, our main aim to use the scheme of Section 4.2 is to lower the number of global GR gates and, in this way, reduce to overall execution time.

For a comprehensive and rigorous evaluation, we chose circuits from three different benchmark collections, with their descriptions available online: **QASM-Bench(small)** [29] and **MQT-Bench** [40] contain

Table 1: Averaged reduction of execution time T relative to Qiskit-default. Negative percentages indicate an increased execution time.

	Circuits	Qiskit	No-decomp	ZX-default	ZX-no-insert	ZX-with-insert
QASM-Bench [29] [1]	35	0%	8%	2%	14%	26%
MQT-Bench [40] [2]	11	0%	0%	−44%	−16%	26%
Feynman-Bench [1] [3]	26	0%	63%	−23%	−16%	40%

[1] https://github.com/pnnl/QASMBench   [2] https://www.cda.cit.tum.de/mqtbench/   [3] https://github.com/meamy/feynman

various low-level benchmark circuits of different sizes and types with common quantum subroutines and algorithms. Additionally, we also consider the **Feynman-Bench** [1] collection. Created for formal methods, it contains different arithmetic circuits usually based on Toffoli gates.

The code used for the evaluations is available with an MIT license at Zenodo [50] allowing reproducibility and possible usage or integration into other compilation projects.

5.4   Results & Discussion

The five compilation schemes are evaluated on gate count and execution time T[ms] of the synthesized circuits, together with the algorithm runtime r[s]. We first discuss time reduction averaged over all circuits, summarized in Table 1, then, we highlight six examples shown in Table 2, which have been selected to best illustrate different cases within the dataset. The full dataset with all raw data is available at Zenodo [50]. On the QASM-Bench collection (1), the ZX-with-insert approach results in an average 26% reduction of execution time compared to the Qiskit internal synthesis improving 23 of the 35 circuits. Similar numbers result for the MQT-Bench (2) circuits with 26% reduction of execution time, improving 7 out of 11 circuits. For the Feynman benchmarks (3), results are mixed: While our scheme achieves a 40% reduction compared to Qiskit, improving 24 of 26 circuits, the No-decomp scheme has an even higher average reduction of execution time with 63%.

Considering the above part of Table 2 one can see how the synthesis approach described in this work is capable of successfully synthesizing C_nP gates. Since No-decomp just replaces multi-controlled gates by their C_nZ} equivalent, the corresponding column indicates the number of multi-controlled gates present in the original circuit. In comparison, one can then see that while No-insert only resynthesizes a few of the original gates, With-insert synthesizes more multi-qubit gates and is often able to create even higher-dimensioned controlled gates. This higher-controlled gate synthesis appears very dominantly in dense circuits such as qnn_10, corresponding to a quantum neural network circuit, but also in circuits that are natively built on controlled phase gates such as HHL.

Due to the controlled gates, the proposed approach is capable of effectively reducing the number of slow global GR gates in comparison to the regular ZX extraction scheme and Qiskit, which are not capable of synthesizing multi-controlled gates. Generally, a lower number of GR gates also results in a shorter circuit execution time with some exceptions, such as the hhl_n7, where the ZX synthesis has a longer execution time than the No-decomp scheme, although the number of absolute GR pulses is lower. This is likely due to an increased pulse time of the individual GR gates.

The ZX approaches do not perform well on circuits that already contain close to optimal multi-controlled gates, for instance on circuits of the Feynman benchmark. Here, the approaches extract gates in a less efficient way, resulting in a gate and time overhead. In such cases, replacing multi-controlled gates without changing circuit structure as in the No-decomp scheme is the best option. This can also be seen when comparing the number of GR gates for the two adder circuits to the No-decomp scheme, where

Table 2: Evaluation results for six benchmarks, selected to illustrate both good and poor performance. Numbers after the names indicate the corresponding benchmark collection. The first table shows gate counts corresponding to the native gate set of Equation (7). The second table contains the total execution time T[ms] and the synthesis algorithm runtime r[s] on a consumer notebook.

	ZX Default		ZX No-insert			ZX With-insert						Qiskit Default		Own alternative No-decomp		
	GR	CP	GR	CP	C2P	GR	CP	C2P	C3P	C4P	C5P	GR	CP	GR	CP	C2P
hhl_n7 (1)	448	296	362	241	-	**306**	207	42	29	6	-	356	196	356	196	-
qft_10 (2)	90	140	36	75	-	**14**	62	14	-	-	-	44	105	44	105	-
qnn_10 (2)	106	334	62	199	-	**28**	159	48	26	8	1	76	188	76	188	-
gf2^7_mult (3)	214	956	134	447	19	**14**	22	114	-	-	-	194	300	18	6	49
rc_adder_6 (3)	76	100	84	95	-	62	91	4	-	-	-	86	93	**28**	27	11
qcla_adder_10 (3)	128	331	138	462	1	24	153	121	-	-	-	74	233	**18**	29	34

	ZX Default		ZX No-insert		ZX With-insert		Qiskit Default		Own alternative No-decomp	
	T	r	T	r	T	r	T	r	T	r
hhl_n7 (1)	11.07	0.085	8.71	0.156	7.70	0.239	5.45	0.122	**5.45**	0.277
qft_10 (2)	2.16	0.036	0.91	0.029	**0.35**	0.039	0.89	0.043	0.89	0.103
qnn_10 (2)	3.12	0.031	1.53	0.108	**0.74**	0.214	3.06	0.117	3.06	0.151
gf2^7_mult (3)	5.84	0.218	3.31	1.286	**0.40**	3.421	4.20	0.083	0.47	0.041
rc_adder_6 (3)	1.89	0.030	2.04	0.049	1.56	0.052	1.67	0.033	**0.71**	0.041
qcla_adder_10 (3)	3.26	1.326	3.48	0.448	0.66	2.063	1.63	0.073	**0.47**	0.078

the ZX approaches are not capable of reconstructing a similar efficient circuit structure. Since all ZX strategies yield inefficient circuits, it may be that in such cases the ZX-diagram simplification creates too complex graph structures.

Regarding algorithmic runtime, ZX performs similarly to the Qiskit internal synthesis. Note, however, the significant increase in runtime for qcla_adder and gf2^t_mult for the With-insert synthesis. This is likely the overhead due to the insertion of additional phase gadgets, resulting in worst-case exponential runtime as discussed in Section 4.2.2.

6 Conclusion

In this work we introduced a novel approach to synthesize quantum circuits to the universal gate set $\{H, R_z, C_nP\}$. As a key contribution, our approach is able to efficiently identify structures in graph-like ZX-diagrams that correspond to multi-controlled phase gates and extract them to quantum circuits. This allows the synthesis of such gates even if they were not present in the original circuit. Together with existing simplification strategies for ZX-diagrams, our approach can be used to synthesize arbitrary quantum circuits towards neutral atom architectures. Here, our synthesis often trades slow global pulse rotations for fast multi-controlled qubit gates and we are thus able to reduce execution time significantly for many common circuits. Further, this could also help hardware developers to evaluate whether increasing the number of qubits supported by multi-controlled phase gates is beneficial for certain problems in terms

of execution time and fidelity. In cases where the circuit already consists of optimized multi-controlled gates, such as circuits based on arithmetic functions, the synthesis may result in less efficient quantum circuits. This is likely due to overly complex graph structures resulting from ZX-diagram simplification. We leave it as a topic for further research whether in those cases more sophisticated strategies allow exploiting the phase gadget structures for multi-controlled phase gates synthesis without increasing the underlying graph structure complexity. Possible approaches include advanced heuristics applying the proposed scheme only to cases where it is likely to improve the circuit structure. We also want to mention that a similar synthesis approach could be done without ZX-calculus using the Pauli Dependency DAG representation of quantum circuits [48]. It has been shown that the diagram simplification rules from Section 3.1.2 are equivalent to reordering Pauli terms in a Pauli Dependency DAG and by identifying patterns of individual Pauli-Z terms similar to Theorem 1 we can then synthesize $C_n P$ gates. As future work, it would be interesting to see how these two versions compare.

Acknowledgments

References

[1] Matthew Amy (2019): *Towards Large-scale Functional Verification of Universal Quantum Circuits*. *Electronic Proceedings in Theoretical Computer Science* 287, pp. 1–21, doi:10.4204/EPTCS.287.1.

[2] Matthew Amy, Parsiad Azimzadeh & Michele Mosca (2018): *On the controlled-NOT complexity of controlled-NOT–phase circuits*. *Quantum Science and Technology* 4(1), p. 015002, doi:10.1088/2058-9565/aad8ca. Available at `https://dx.doi.org/10.1088/2058-9565/aad8ca`.

[3] Matthew Amy, Andrew N. Glaudell, Sarah Meng Li & Neil J. Ross (2023): *Improved Synthesis of Toffoli-Hadamard Circuits*, pp. 169–209. doi:10.1007/978-3-031-38100-3_12. arXiv:2305.11305.

[4] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski & John van de Wetering (2021): *There and back again: A circuit extraction tale*. *Quantum* 5, p. 421, doi:10.22331/q-2021-03-25-421. Available at `http://arxiv.org/abs/2003.01664`. ArXiv:2003.01664 [quant-ph].

[5] Jonathan M. Baker, Andrew Litteken, Casey Duckering, Henry Hoffmann, Hannes Bernien & Frederic T. Chong (2021): *Exploiting Long-Distance Interactions and Tolerating Atom Loss in Neutral Atom Quantum Architectures*, pp. 818–831. doi:10.1109/ISCA52012.2021.00069.

[6] Daniel Barredo, Sylvain de Léséleuc, Vincent Lienhard, Thierry Lahaye & Antoine Browaeys (2016): *An Atom-by-Atom Assembler of Defect-Free Arbitrary Two-Dimensional Atomic Arrays*. Science 354(6315), pp. 1021–1023, doi:10.1126/science.aah3778.

[7] Dolev Bluvstein, Simon J. Evered, Alexandra A. Geim, Sophie H. Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, J. Pablo Bonilla Ataides, Nishad Maskara, Iris Cong, Xun Gao, Pedro Sales Rodriguez, Thomas Karolyshyn, Giulia Semeghini, Michael J. Gullans, Markus Greiner, Vladan Vuletić & Mikhail D. Lukin (2023): *Logical Quantum Processor Based on Reconfigurable Atom Arrays*. Nature, pp. 1–3, doi:10.1038/s41586-023-06927-3.

[8] Dolev Bluvstein, Harry Levine, Giulia Semeghini, Tout T. Wang, Sepehr Ebadi, Marcin Kalinowski, Alexander Keesling, Nishad Maskara, Hannes Pichler, Markus Greiner, Vladan Vuletić & Mikhail D. Lukin (2022): *A Quantum Processor Based on Coherent Transport of Entangled Atom Arrays*. Nature 604(7906), pp. 451–456, doi:10.1038/s41586-022-04592-6.

[9] Sebastian Brandhofer, Ilia Polian & Hans Peter Büchler (2021): *Optimal Mapping for Near-Term Quantum Architectures Based on Rydberg Atoms*. In: *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–7, doi:10.1109/ICCAD51958.2021.9643490.

[10] Hans J Briegel, David E Browne, Wolfgang Dür, Robert Raussendorf & Maarten Van den Nest (2009): *Measurement-based quantum computation*. Nature Physics 5(1), pp. 19–26, doi:10.1038/nphys1157.

[11] Daniel E Browne, Elham Kashefi, Mehdi Mhalla & Simon Perdrix (2007): *Generalized flow and determinism in measurement-based quantum computation*. New Journal of Physics 9(8), p. 250, doi:10.1088/1367-2630/9/8/250. Available at `https://dx.doi.org/10.1088/1367-2630/9/8/250`.

[12] Alec Cao, William J. Eckner, Theodor Lukin Yelin, Aaron W. Young, Sven Jandura, Lingfeng Yan, Kyungtae Kim, Guido Pupillo, Jun Ye, Nelson Darkwah Oppong & Adam M. Kaufman (2024): *Multi-qubit gates and 'Schrödinger cat' states in an optical clock*. arXiv:2402.16289.

[13] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes*. Cambridge University Press, doi:10.1017/9781316219317.

[14] Clemens Dlaska, Kilian Ender, Glen Bigan Mbeng, Andreas Kruckenhauser, Wolfgang Lechner & Rick van Bijnen (2022): *Quantum Optimization via Four-Body Rydberg Gates*. Physical Review Letters 128(12), p. 120503, doi:10.1103/PhysRevLett.128.120503.

[15] Ross Duncan, Aleks Kissinger, Simon Perdrix & John van de Wetering (2020): *Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus*. Quantum 4, p. 279, doi:10.22331/q-2020-06-04-279. Available at `https://quantum-journal.org/papers/q-2020-06-04-279/`. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.

[16] Simon J. Evered, Dolev Bluvstein, Marcin Kalinowski, Sepehr Ebadi, Tom Manovitz, Hengyun Zhou, Sophie H. Li, Alexandra A. Geim, Tout T. Wang, Nishad Maskara, Harry Levine, Giulia Semeghini, Markus Greiner, Vladan Vuletić & Mikhail D. Lukin (2023): *High-Fidelity Parallel Entangling Gates on a Neutral-Atom Quantum Computer*. Nature 622(7982), pp. 268–272, doi:10.1038/s41586-023-06481-y. arXiv:2304.05420.

[17] T. M. Graham, Y. Song, J. Scott, C. Poole, L. Phuttitarn, K. Jooya, P. Eichler, X. Jiang, A. Marra, B. Grinkemeyer, M. Kwon, M. Ebert, J. Cherek, M. T. Lichtman, M. Gillette, J. Gilbert, D. Bowman, T. Ballance, C. Campbell, E. D. Dahl, O. Crawford, N. S. Blunt, B. Rogers, T. Noel & M. Saffman (2022): *Multi-Qubit Entanglement and Algorithms on a Neutral-Atom Quantum Computer*. Nature 604(7906), pp. 457–462, doi:10.1038/s41586-022-04603-6.

[18] Daniel Große, Robert Wille, Gerhard W Dueck & Rolf Drechsler (2009): *Exact multiple-control Toffoli network synthesis with SAT techniques*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28(5), pp. 703–715, doi:10.1109/TCAD.2009.2017215.

[19] Flavien Gyger, Maximilian Ammenwerth, Renhao Tao, Hendrik Timme, Stepan Snigirev, Immanuel Bloch & Johannes Zeiher (2024): *Continuous Operation of Large-Scale Atom Arrays in Optical Lattices*. arXiv:2402.04994.

[20] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond & Christophe Jurczak (2020): *Quantum Computing with Neutral Atoms*. Quantum 4, p. 327, doi:10.22331/q-2020-09-21-327.

[21] L. Isenhower, M. Saffman & K. Mølmer (2011): *Multibit CkNOT Quantum Gates via Rydberg Blockade*. Quantum Information Processing 10(6), p. 755, doi:10.1007/s11128-011-0292-4.

[22] Sven Jandura & Guido Pupillo (2022): *Time-Optimal Two- and Three-Qubit Gates for Rydberg Atoms*. Quantum 6, p. 712, doi:10.22331/q-2022-05-13-712.

[23] Aleks Kissinger & John van de Wetering (2020): *Reducing the Number of Non-Clifford Gates in Quantum Circuits*. Physical Review A 102(2), p. 022406, doi:10.1103/PhysRevA.102.022406.

[24] Aleks Kissinger & John van de Wetering (2020): *PyZX: Large Scale Automated Diagrammatic Reasoning* 318, pp. 229–241. doi:10.4204/EPTCS.318.14.

[25] Vadym Kliuchnikov, Dmitri Maslov & Michele Mosca (2013): *Fast and efficient exact synthesis of single-qubit unitaries generated by clifford and T gates*. Quantum Information & Computation 13(7-8), pp. 607–630, doi:10.5555/2535649.2535653.

[26] Stach Kuijpers, John van de Wetering & Aleks Kissinger: *Graphical fourier theory and the cost of quantum addition*. Available at `https://doi.org/10.48550/arXiv.1904.07551`.

[27] Harry Levine, Dolev Bluvstein, Alexander Keesling, Tout T. Wang, Sepehr Ebadi, Giulia Semeghini, Ahmed Omran, Markus Greiner, Vladan Vuletić & Mikhail D. Lukin (2022): *Dispersive Optical Systems for Scalable Raman Driving of Hyperfine Qubits*. Physical Review A 105(3), p. 032618, doi:10.1103/PhysRevA.105.032618.

[28] Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Tout T. Wang, Sepehr Ebadi, Hannes Bernien, Markus Greiner, Vladan Vuletić, Hannes Pichler & Mikhail D. Lukin (2019): *Parallel Implementation of High-Fidelity Multiqubit Gates with Neutral Atoms*. Physical Review Letters 123(17), p. 170503, doi:10.1103/PhysRevLett.123.170503.

[29] Ang Li, Samuel Stein, Sriram Krishnamoorthy & James Ang (2022): *QASMBench: A Low-level QASM Benchmark Suite for NISQ Evaluation and Simulation*, doi:10.48550/arXiv.2005.13018. arXiv:2005.13018.

[30] Yongshang Li, Yu Zhang, Mingyu Chen, Xiangyang Li & Peng Xu (2023): *Timing-Aware Qubit Mapping and Gate Scheduling Adapted to Neutral Atom Quantum Computing*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 1–1, doi:10.1109/TCAD.2023.3261244.

[31] Ivaylo S. Madjarov, Jacob P. Covey, Adam L. Shaw, Joonhee Choi, Anant Kale, Alexandre Cooper, Hannes Pichler, Vladimir Schkolnik, Jason R. Williams & Manuel Endres (2020): *High-Fidelity Entanglement and Detection of Alkaline-Earth Rydberg Atoms*. Nature Physics 16(8), pp. 857–861, doi:10.1038/s41567-020-0903-z.

[32] M Morgado & S Whitlock (2021): *Quantum simulation and computing with Rydberg-interacting qubits*. AVS Quantum Science 3(2), doi:10.1116/5.0036562. arXiv:https://pubs.aip.org/avs/aqs/article-pdf/doi/10.1116/5.0036562/19739152/023501_1_online.pdf.

[33] Priyanka Mukhopadhyay (2024): *Synthesizing Toffoli-optimal quantum circuits for arbitrary multi-qubit unitaries*. arXiv preprint arXiv:2401.08950. Available at `https://doi.org/10.48550/arXiv.2401.08950`.

[34] M. Müller, I. Lesanovsky, H. Weimer, H. P. Büchler & P. Zoller (2009): *Mesoscopic Rydberg Gate Based on Electromagnetically Induced Transparency*. Physical Review Letters 102(17), p. 170502, doi:10.1103/PhysRevLett.102.170502.

[35] M. A. Norcia, H. Kim, W. B. Cairncross, M. Stone, A. Ryou, M. Jaffe, M. O. Brown, K. Barnes, P. Battaglino, A. Brown, K. Cassella, C.-A. Chen, R. Coxe, D. Crow, J. Epstein, C. Griger, E. Halperin, F. Hummel, A. M. W. Jones, J. M. Kindem, J. King, K. Kotru, J. Lauigan, M. Li, M. Lu, E. Megidish, J. Marjanovic, M. McDonald, T. Mittiga, J. A. Muniz, S. Narayanaswami, C. Nishiguchi, T. Paule, K. A. Pawlak, L. S. Peng, K. L. Pudenz, A. Smull, D. Stack, M. Urbanek, R. J. M. van de Veerdonk, Z. Vendeiro, L. Wadleigh, T. Wilkason, T.-Y. Wu, X. Xie, E. Zalys-Geller, X. Zhang & B. J. Bloom (2024): *Iterative Assembly of ^{171}Yb Atom Arrays in Cavity-Enhanced Optical Lattices*, doi:10.48550/arXiv.2401.16177. arXiv:2401.16177.

[36] Natalia Nottingham, Michael A. Perlin, Ryan White, Hannes Bernien, Frederic T. Chong & Jonathan M. Baker (2023): *Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures*, doi:10.48550/arXiv.2307.14996. arXiv:2307.14996.

[37] Tirthak Patel, Daniel Silver & Devesh Tiwari (2022): *Geyser: A Compilation Framework for Quantum Computing with Neutral Atoms*. In: *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, Association for Computing Machinery, New York, NY, USA, pp. 383–395, doi:10.1145/3470496.3527428.

[38] Lars Pause, Lukas Sturm, Marcel Mittenbühler, Stephan Amann, Tilman Preuschoff, Dominik Schäffner, Malte Schlosser & Gerhard Birkl (2023): *Supercharged Two-Dimensional Tweezer Array with More than 1000 Atomic Qubits*, doi:10.48550/arXiv.2310.09191. arXiv:2310.09191.

[39] Tom Peham, Lukas Burgholzer & Robert Wille (2022): *Equivalence checking of quantum circuits with the ZX-calculus*. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 12(3), pp. 662–675, doi:10.1109/JETCAS.2022.3202204.

[40] Nils Quetschlich, Lukas Burgholzer & Robert Wille (2023): *MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing*. Quantum 7, p. 1062, doi:10.22331/q-2023-07-20-1062.

[41] M Saffman (2016): *Quantum computing with atomic qubits and Rydberg interactions: progress and challenges*. Journal of Physics B: Atomic, Molecular and Optical Physics 49(20), p. 202001, doi:10.1088/0953-4075/49/20/202001. Available at `https://dx.doi.org/10.1088/0953-4075/49/20/202001`.

[42] M. Saffman, T. G. Walker & K. Mølmer (2010): *Quantum Information with Rydberg Atoms*. Reviews of Modern Physics 82(3), pp. 2313–2363, doi:10.1103/RevModPhys.82.2313.

[43] Mark Saffman (2019): *Quantum Computing with Neutral Atoms*. National Science Review 6(1), pp. 24–25, doi:10.1093/nsr/nwy088.

[44] Ludwig Schmid, David F Locher, Manuel Rispler, Sebastian Blatt, Johannes Zeiher, Markus Müller & Robert Wille (2024): *Computational capabilities and compiler development for neutral atom quantum processors—connecting tool developers and hardware experts*. Quantum Science and Technology 9(3), p. 033001, doi:10.1088/2058-9565/ad33ac. Available at `https://dx.doi.org/10.1088/2058-9565/ad33ac`.

[45] Ludwig Schmid, Sunghye Park, Seokhyeong Kang & Robert Wille (2023): *Hybrid Circuit Mapping: Leveraging the Full Spectrum of Computational Capabilities of Neutral Atom Quantum Computers*, doi:10.48550/arXiv.2311.14164. arXiv:2311.14164.

[46] Adam L. Shaw, Ran Finkelstein, Richard Bing-Shiun Tsai, Pascal Scholl, Tai Hyun Yoon, Joonhee Choi & Manuel Endres (2024): *Multi-Ensemble Metrology by Programming Local Rotations with Atom Movements*. Nature Physics, pp. 1–7, doi:10.1038/s41567-023-02323-w.

[47] VV Shende, SS Bullock & IL Markov (2006): *Synthesis of quantum-logic circuits*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(6), pp. 1000–1010, doi:10.1109/TCAD.2005.855930.

[48] Will Simmons (2021): *Relating Measurement Patterns to Circuits via Pauli Flow*. Electronic Proceedings in Theoretical Computer Science 343, p. 50–101, doi:10.4204/eptcs.343.4.

[49] Korbinian Staudacher, Tobias Guggemos, Sophia Grundner-Culemann & Wolfgang Gehrke (2023): *Reducing 2-QuBit Gate Count for ZX-Calculus based Quantum Circuit Optimization*. Electronic Proceedings in Theoretical Computer Science 394, pp. 29–45, doi:10.4204/EPTCS.394.3.

[50] Korbinian Staudacher, Ludwig Schmid, Johannes Zeiher, Robert Wille & Dieter Kranzlmüller (2024): *Multi-Controlled Phase Gate Synthesis with ZX- Calculus*, doi:10.5281/zenodo.10730427.

[51] Daniel Bochen Tan, Dolev Bluvstein, Mikhail D. Lukin & Jason Cong (2024): *Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors*. Quantum 8, p. 1281, doi:10.22331/q-2024-03-14-1281.

[52] Daniel Bochen Tan, Shuohao Ping & Jason Cong (2024): *Depth-Optimal Addressing of 2D Qubit Array with 1D Controls Based on Exact Binary Matrix Factorization*. arXiv:2401.13807.

[53] Renaud Vilmart (2019): *A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics*, pp. 1–10. doi:10.1109/LICS.2019.8785765.

[54] Hanrui Wang, Pengyu Liu, Bochen Tan, Yilian Liu, Jiaqi Gu, David Z. Pan, Jason Cong, Umut Acar & Song Han (2023): *FPQA-C: A Compilation Framework for Field Programmable Qubit Array*, doi:10.48550/arXiv.2311.15123. arXiv:2311.15123.

[55] Hanrui Wang, Bochen Tan, Pengyu Liu, Yilian Liu, Jiaqi Gu, Jason Cong & Song Han (2023): *Q-Pilot: Field Programmable Quantum Array Compilation with Flying Ancillas*, doi:10.48550/arXiv.2311.16190. arXiv:2311.16190.

[56] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*. arXiv preprint *arXiv:2012.13966*. Available at `https://doi.org/10.48550/arXiv.2012.13966`.

[57] Shihao Zhang, Junda Wu & Lvzhou Li (2023): *Characterization, synthesis, and optimization of quantum circuits over multiple-control Z-rotation gates: A systematic study*. Phys. Rev. A 108, p. 022603, doi:10.1103/PhysRevA.108.022603. Available at `https://link.aps.org/doi/10.1103/PhysRevA.108.022603`.

A   Gadget insertion

Inserting YZ measurements on the outputs preserves gflow:

Corollary 1. *Let* $(g, \prec)$ *be a gflow for* (G, I, O, λ) *and let* $W \subseteq O$. *Then* (G', I, O, λ'), *where* $G' = (V', E')$ *with* $V' = V \cup \{x\}$, $\lambda'(x) = YZ$ *and* $E' = E \cup \{(x, w) | w \in W\}$ *has a gflow* $(g', \prec')$ *with following properties:*

- $g'(x) = \{x\}$,

- $\forall v \in V: g'(v) = g(v)$,

- $\prec'$ *is the transitive closure of* $\prec \cup \{(x, v) | v \in O\} \cup \{(v, x) | v \in V \setminus O\}$.

Proof. The only new correction set in g' is $g'(x) = \{x\}$, for all other vertices, it is the same as in g. Therefore, all conditions except $(g2)$ are trivially satisfied. For $(g2)$, we need to distinguish two cases for the new vertex x and vertices $v \in V \setminus \{x\}$:

- $x \in Odd(g(v))$: By definition, x is the last element in the partial order $\prec'$ of all non-outputs, thus $(g2)$ holds.

- $v \in Odd(g(x))$: $x \prec v$ holds, because $g(x) = \{x\}$ and $Odd(\{x\})$ only contains outputs which we chose to be after x in the partial order.

Note that $x \notin Odd(g(x))$ by definition. $\qquad \square$
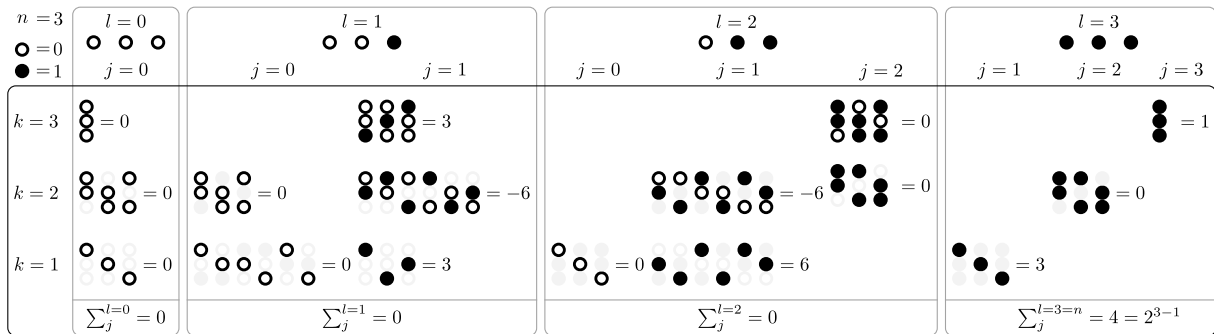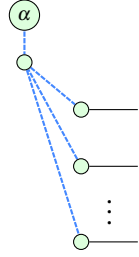
B   Alternative proof of Theorem 1

Figure 3: Illustration of the possible combinations and their contribution in Equation (12) for $n = 3$. For each possible value of $l = 0, \ldots, 3$ the combinations for the possible $k \leq n$ and $j \leq l$ are illustrated as three-circle circles, and their contribution to the sum is computed. The final row shows that the sums of the contributions fulfill the condition of Lemma 2.

This section provides an alternative, combinatorial proof of Theorem 1 instead of using the graphical approach discussed in the main part of the work. The overarching idea is to find a closed formula for the unitary defined by the ZX illustration by summing the corresponding phase contributions and showing that this corresponds to $\text{diag}(1, 1, \ldots, e^{i\alpha})$ for an arbitrary number of qubits n.

To find a closed formula for Theorem 1 consider the definition of a single phase gadget and its corresponding unitary action on the *n*-qubit basis states according to [23]:

$$U |x_1, \ldots, x_n\rangle = e^{i\alpha(x_1 \oplus \ldots \oplus x_n)} |x_1, \ldots, x_n\rangle \quad , \tag{9}$$

where the binary $x_i \in \{0,1\}, i = 1, \ldots, n$ label the basis states and $\oplus$ is the binary sum modulo two, i.e. XOR. Note that all phase gadgets of Theorem 1 can be written in such a way, summing only the x_i connected by Hadamard wires to the phase. The single qubit phases give an additional contribution of $e^{i\alpha x_j} |x_1, \ldots, x_n\rangle$ for each applied qubit j, corresponding to a single-qubit phase gadget.

As the binary sum does not depend on the order of the x_i but only on their value, we introduce the following notation, where we assume that l entries in the sum are non-zero, resulting in a non-zero-sum whenever l is odd:

$$x_1 \oplus \ldots \oplus x_n \quad = \quad \frac{(-1)^{l+1} + 1}{2} \quad = \quad \mathrm{mod}_2(l) \quad = \quad \begin{cases} 1 & , l \text{ is odd} \\ 0 & , l \text{ is even} \end{cases} . \tag{10}$$

Considering again Equation (2) one can see that there are two contributions to the total accumulated phase. First, for each qubit, a single-qubit phase α is added. Second, for each possible combination of length k of all the x_i, there is a phase gadget with phase $(-1)^{k+1}\alpha$. For the $C_1P(2\alpha)$ gate, this reduces to a single $k = 2$ phase gadget of phase $-\alpha$. For the $C_2P(4\alpha)$ gate, on the other hand, there are $\binom{3}{2} = 3$ phase gadgets of size $k = 2$ and angle $-\alpha$ and a single $(\binom{3}{3} = 1)$ $k = 3$ gadget with angle α. In general, for a n qubit gate, there are $\binom{n}{k}$ combinations for phase gadgets of size $k = 1, \ldots, n$. A combination contributes to the total phase if the number l of non-zero entries in the direct sum of Equation (9) is odd, resulting in an additional phase $\pm\alpha$. Otherwise, the combination does not contribute to the phase. To express the number of possible combinations depending on l, the $\binom{n}{k}$ combinations for a length k can also be expressed as choosing j variables from the l one-valued variables and choosing $k - j$ variables from the $n - l$ zero-valued variables and summing over all possible j:

$$\binom{n}{k} = \sum_{j=0}^{k} \binom{l}{j} \binom{n-l}{k-j}. \tag{11}$$

This relation is known as the *Vandermonde identity*. An illustration of the possible combinations depending on k and l is shown in Figure 3 for the simple case $n = 3$.

Multiplying the unitaries of all these phase gadgets corresponds to summing the accumulated phases with the appropriate sign, converting the problem of Theorem 1 into a summation of the appropriate phases with a corresponding sign. For the total structure to represent a multi-controlled phase gate, the phases have to vanish for all possible basis states $|x_1, \ldots, x_n\rangle$ except for $|1, \ldots 1\rangle$ where they have to sum to $2^{n-1}\alpha$.

Based on these considerations, an equivalent statement of Theorem 1 can be formulated, dropping the illustrations of the ZX-calculus and formulating the multi-controlled phase gate extraction as a purely combinatorial problem, focusing on the accumulated phase. Theorem 1 then directly follows from this Lemma based on the considerations above and using $e^{i\alpha \cdot 0} = 1$.

Lemma 2 (Multi-controlled phase gate). *For n binary variables $x_1 \ldots x_n$ of which l is non-zero, summing the modulo two sum over all possible combinations of length k with sign $(-1)^{k+1}$, it holds:*

$$\sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{\min(k,l)}\binom{l}{j}\binom{n-l}{k-j}\mathrm{mod}_2(j) = \begin{cases} 2^{n-1} & , \text{ if } n=l \\ 0 & , \text{ else} \end{cases}. \tag{12}$$

Where the $\min(k,l)$ results from the fact that the number of ones in the current combination j cannot be larger than the length k of the combination, nor the total number of ones l available.

Proof. The proof is two-fold. First, the $n = l$ case is shown explicitly, while the case $n \neq l$ is shown by induction in both variables n and l. Also, note that

$$\sum_{j=0}^{\min(k,l)}\binom{l}{j}\binom{n-l}{k-j} = \sum_{j=0}^{k}\binom{l}{j}\binom{n-l}{k-j} = \sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j} \tag{13}$$

as for $k > l$ the first binomial coefficient vanishes in all additional cases, and for $l > k$ the second, as $k - j < 0$ in these cases. This also becomes clear from the illustration in Figure 3 where the vanishing combinations are either non-existent or only contain zero entries. These identities are used multiple times in the following proof.

Case $n = l$:

If $n = l$ all variables are one and, therefore, $\min(k,l) = k$. Furthermore, the second binomial coefficient is non-zero only in the $j = k$ case, where it equals 1. This results in

$$\sum_{k=1}^{n}(-1)^{k+1}\binom{n}{k}\frac{(-1)^{k+1}+1}{2} = \frac{1}{2}\left[\sum_{k=1}^{n}\binom{n}{k}(-1)^{k+1} + \sum_{k=1}^{n}\binom{n}{k}\right] = \frac{1}{2}[1+2^n-1] = 2^{n-1},$$

using the regular and the alternating binomial sum, directly showing the first part of Lemma 2.

Case $n \neq l$:

Proving Equation (12) for arbitrary n and $l < n$ is done by induction. Therefore, showing the term to be zero for $l = 0$ and arbitrary n as the base case and then performing the induction step both in n and in l. *Base case $l = 0, n$:* In this case the second sum reduces to the $j = 0$ case, trivially giving zero, independetly for all n. In other words, as all variables are zero, the sum in Equation (9) always gives zero. *Induction step $n \to n+1$:* Inserting this step into Equation (9) and using the recurrence relation of the binomial coefficient $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$ and the abbreviation $\xi := \mathrm{mod}_2(j)$ one gets

$$\sum_{k=1}^{n+1}(-1)^{k+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j-1}\xi + (-1)^n + \underbrace{\sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j}\xi}_{=0 \text{ (Base case)}}$$

$$+ (-1)^n\sum_{j=0}^{l}\binom{l}{j}\cancel{\binom{n-l}{n+1-j}}\xi$$

$$= \sum_{k=0}^{n}(-1)^k\sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j}\frac{(-1)^j+1}{2} = \sum_{k=0}^{n}(-1)^{k+1}\sum_{j=0}^{k}\binom{l}{j}\binom{n-l}{k-j}\frac{(-1)^{j+1}+1-2}{2}$$

$$= (-1)^1 \underbrace{\binom{l}{0}\binom{0-l}{0-0}\frac{-2}{2}}_{k=0 \text{ case}} + \underbrace{\sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n-l}{k-j}\xi}_{=0 \text{ (Base case)}} + \underbrace{\sum_{k=0}^{n}(-1)^{k+1}\sum_{j=0}^{k}\binom{l}{j}\binom{n-l}{k-j}}_{\text{Vandermonde}}$$

$$= 1 + 0 - 1 = 0 \quad ,$$

writing the $n+1$ term separately to recover the base case. The term in the second line vanishes due to the lower part of the binomial coefficient always being larger than the top part. Going to the third line, an index shift in k is performed, and then using Equation (13) with an additional reinserted (-1) factor to recover the original form of ξ. Separating the $k=0$ case and the additional introduced -2 term, the base case can be inserted again, resulting in zero after using again the Vandermonde identity and the alternating binomial sum. In a similar fashion, also the induction step in l can be shown.

Induction step $l \to l+1$: With the base case for arbitrary n and the corresponding induction step in n, one can, in the following, assume the base case to be true for arbitrary n, in particular for $n' = n-1$. Performing the induction step in $l \to l+1$ and again using the recurrence relation, one gets

$$\sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{l+1}\binom{l+1}{j}\binom{n-(l+1)}{k-j}\xi$$

$$= \sum_{k=1}^{n}(-1)^{k+1}\sum_{j=0}^{l+1}\binom{l}{j-1}\binom{n'-l}{k-j}\xi + \underbrace{\sum_{k=1}^{n'}(-1)^{k+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n'-l}{k-j}\xi}_{=0 \text{ (Base case for } n')}$$

$$+ \underbrace{\sum_{k=1}^{n}(-1)^{k+1}\binom{l}{l+1}\binom{n'-l}{k-l-1}}_{j=l+1 \text{ case}} + \underbrace{(-1)^{n+1}\sum_{j=0}^{l}\binom{l}{j}\binom{n'-l}{n-j}\xi}_{k=n \text{ case}}$$

$$= \sum_{k=1}^{n}(-1)^{k+1}\sum_{j=-1}^{l}\binom{l}{j}\binom{n'-l}{k-j-1}\frac{(-1)^{j}+1}{2}$$

$$= \sum_{k=0}^{n'}(-1)^{k}\sum_{j=0}^{l}\binom{l}{j}\binom{n'-l}{k-j}\frac{(-1)^{j+1}+1}{2}$$

$$= 0 \quad ,$$

with the base case for n' used in the second line and the additional terms vanishing because either the first or the second binomial coefficient is zero. In the following two lines, first, an index shift in j is performed, and then, secondly, in k. The result is the same formula as in the previous calculation, just for n' and therefore, also vanishes.

This concludes the induction step in l, concluding also the $n \neq l$ case of Equation (12) and therefore proving Lemma 2.

$\square$