Can You Beat Casey?

Neutral-atom quantum computing is advancing at an incredible pace. The era of analog devices is giving place to gate-based computers and, in particular, error correction protocols. Much of this is due to the amazing compilation design space of this platform, born out of the flexibility of entangling arbitrary pairs of qubits, a high degree of gate parallelization, as well as a wide variety of native gates.

Another reason why quantum computing is advancing at an incredible pace is the folks that have been putting amazing sweat equity in order to advance the field. Today, your main contender is QuEra's Quantum Systems Architect: Casey Duckering. Casey knows QuEra's systems inside and out and was one of the key people to optimize the circuits used in our recent demonstration of magic state distillation with logical qubits [1]. Your goal? To do better!

In this challenge, we will be providing you a set of circuits which may or may not be too wise in design. Even if they look good, they are probably not choosing the best gate set for neutral-atom systems, not ordering operations so that maximum parallelization is possible, and are definitely not aware of the fact that atoms can be moved around in order to entangle with distant neighbors. On this last point, while arbitrary connectivity is a beautiful asset, it should not be overused: the fewer atom movements, the better the fidelity of a circuit will be. So **we challenge you to use your arsenal of tools and knowledge about circuit design and compilation to optimize these circuits to a specific neutral-atom quantum computing layout while respecting some design rules** that we discuss right below.

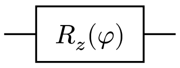
Design Rules

You will be using, or building on top of, QuEra's new language for representing and compiling circuits and operations on atoms, Bloqade-circuits. We also have a list of references that can help you understand some details of neutral-atom gate-based devices [1], [2], [3], as well as some neat tricks for circuit rewrites [4]. You can access these for free on the open-access repository arXiv. Still, use these sparingly, as the time is tight for the challenge. You can also use Quirk as a prototyping tool to help you think through circuit deformations/ transformations. Furthermore, you may want to investigate using third-party compilers to create a pipeline to pre-process circuits into better formats before fine-tuning choices specific for the neutral atom architecture you will be targeting below.

Here goes the summarized set of design rules we will use for our challenge. When in doubt, all options and possibilities defer to the constraints defined here.

Native Gate Set

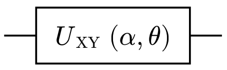
Neutral-atom quantum computers have some flexibility with regards to their native gate set, and a given choice depends on the specific hardware implementation. We will consider the following realistic scenario:



Z-rotation

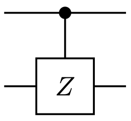
Z-rotations can be done globally (i.e. on all qubits for the same angle) or locally (i.e. on a sub-selection of qubits, with the same angle for all qubits in this subset). The choice of global or local rotations impart different errors in the qubits, with global rotations being strongly favored. In fact, global Z-rotations are substituted for simple delays in phases of subsequent gates, so they are free of errors and cost effectively zero time.

Global 1-qubit gates can be done without moving the atoms we will assume that can affect them both in the storage and processing sites. As for local rotations, atoms have to be moved to some of the 20 sites of the gate zone. Thus, the maximum number of qubits that can be chosen for local rotations in parallel is 20. Read more about our zones and geometric constraints below.



XY-rotation

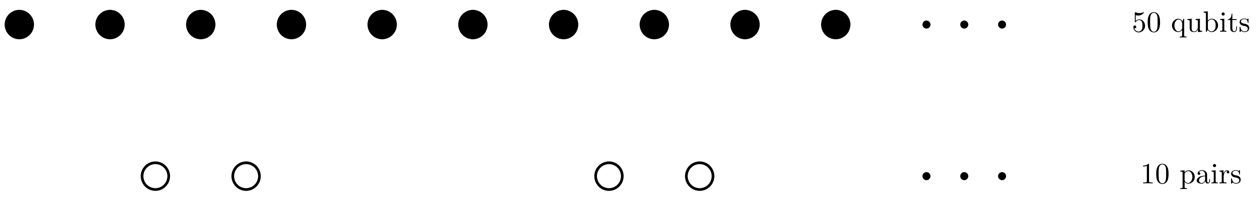
Similar to the Z rotations above, we can do rotations around any axis in the XY-plane both globally or locally. These rotations are defined by an arbitrary angle α on the plane, setting the rotation axis at an angle α in reference to the x-axis, with the rotation angle defined by θ. Once more, global operations are favored over local ones.



The standard entangling gate for us will be the controlled-Z gate. To apply the entangling gate, atoms have to be put in proximity of each other. This is why the gate zone is structured the way it is, with pairs of sites to enable the gate to occur between atoms. The ability to shuttle atoms means you can pick any two atoms in the storage zone regardless of their distance apart and move them to the proper pair of sites in the gate zone. Furthermore, you can pick up multiple atoms and put them in the gate zone *in parallel.*

Register Geometry
To keep things simple, in this challenge we will consider a simplified version of a neutral-atom quantum processor. It will be defined by two zones: a **storage zone** with 50 sites, and a **gate zone** with 10 pairs of sites. Both of them are chosen to be 1-dimensional straight lines.

Here is an illustration to help you make the description above concrete:

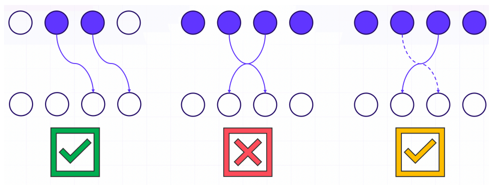


The first row with black dots correspond to the storage zone and contains your atoms. All host spots *can* be filled with qubits, but you can choose that filling with a maximum capacity of 50 atoms; atoms in this zone can go through 1-qubit rotations. The second row with pairs of white spots correspond to the gate zone. Atoms from the storage zone have to be moved to these white spots when you want to entangle them. After that, they move back to the storage zone.

Shuttling Dynamics

Here are the two simple rules which will be constraining your atom-moving strategies:

1. A single move operation may pick up any subset of atoms from one row and then place those atoms in another row (or somewhere else in the same row) under the constraint that the atoms are not reordered during the move and all atoms end up in the same destination row. Shortly, the spots that pick up and drag atoms while they move cannot cross paths. Crossings can be done in series though, but that costs more moves.



2. Atoms are not allowed to collide. A spot with one atom cannot get a second one.

Optimization goals

Here is the cost function you will be targeting to minimize

$$E \equiv \alpha(N_{\mathrm{G,Z}} + N_{\mathrm{G,XY}}) + \beta(N_{\mathrm{L,Z}} + N_{\mathrm{L,XY}}) + \gamma N_{\mathrm{CZ}} + \delta\left(\frac{T}{T_0}\right) + \eta N_{\mathrm{pick}}$$

Here, the variables correspond to:

- $N_{G,Z}$, $N_{\mathrm{G,XY}}$ - number of global 1-qubit gates (rotations around Z and an axis on the XY plane)
- $N_{L,Z}$, $N_{\mathrm{L,XY}}$ - number of local 1-qubit gates (rotations around Z and an axis on the XY plane)
- N_{CZ} - number of control-Z gates
- N_{pick} - number of times atoms are picked/dropped
- T - aggregate time of the circuit, normalized by the characteristic time $T_0 = 100\mu s$. This is typically dominated by the length of your longest move

The greek letters correspond to parameters that evaluate the relative weight of the different terms with respect to the general performance of the circuit. The corresponding values are

α	β	γ	δ	η
0.02	0.5	1.0	0.6	0.8

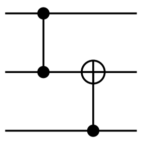
In sum: entanglement costs the most, and optimizing moves should come as the second priority. Then, global gates are favored over local ones.

Challenge Circuits

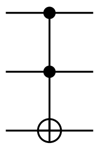
While you can manually input the circuits below into any SDK of your choice for further analysis or optimization, note that for the purposes of judging (and for your own use!) there are QASM files that correspond to each circuit in the `assets\qasm` folder, numbered to align with the order of circuits below.

You may also consult the `assets/examples` folder which contains examples of preparing the GHZ state using bloqade-circuits.

Circuit #1
Let's start with a simple warm-up, with the two circuits below counting towards your scoring. First, let's just look at something unrealistic. Try to optimize the circuit below:

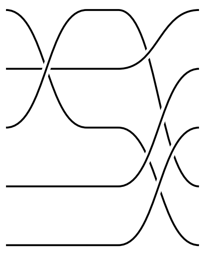


Now, let's add one more simple circuit to this warm-up exercise. Break a Toffoli gate into an efficient native circuit for our system:

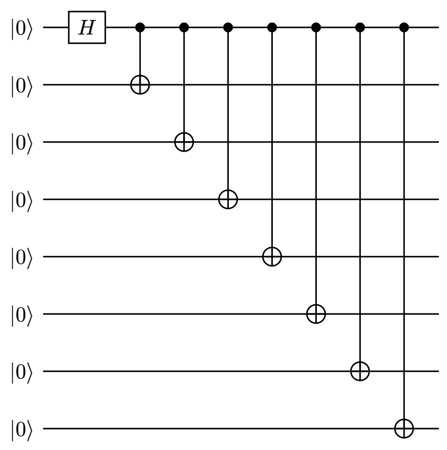


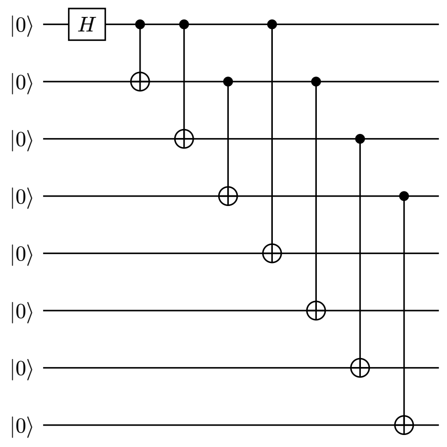
The following will NOT count towards your score but are left here to encourage you to think about how one could go about optimizing atom moves.

First, you can try to simplify this swap network:

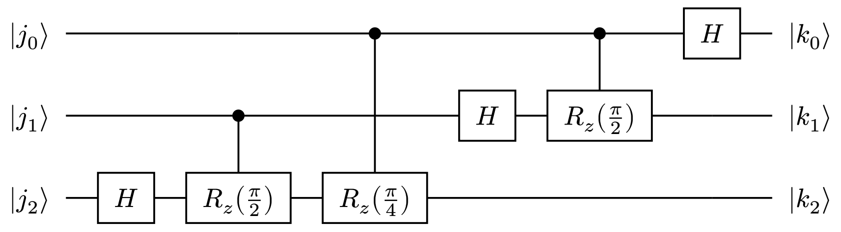


Then, take a look at the following two circuits. They are equivalent circuits for preparing a Greenberger-Horne-Zeilinger (GHZ) state but one is more optimal than the other for the architecture given to you.





Circuit #2
Now let's go to something more realistic. Let's optimize a standard quantum Fourier transform circuit:



Feeling this is too easy? Here are some side-challenges. Try to optimize the graph now for 16 trios of qubits in parallel! Or extend this to a larger Fourier transform over 20+ qubits.

Circuit #3
Scaling difficulty! QAOA on a graph. Assume you have the following graph:

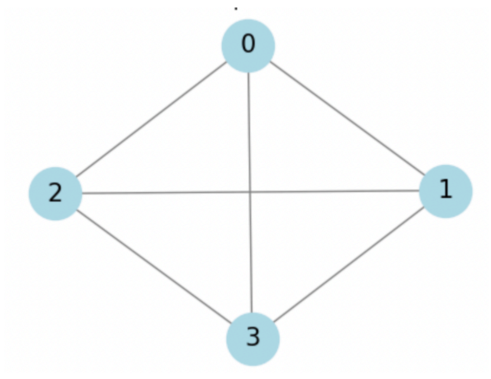
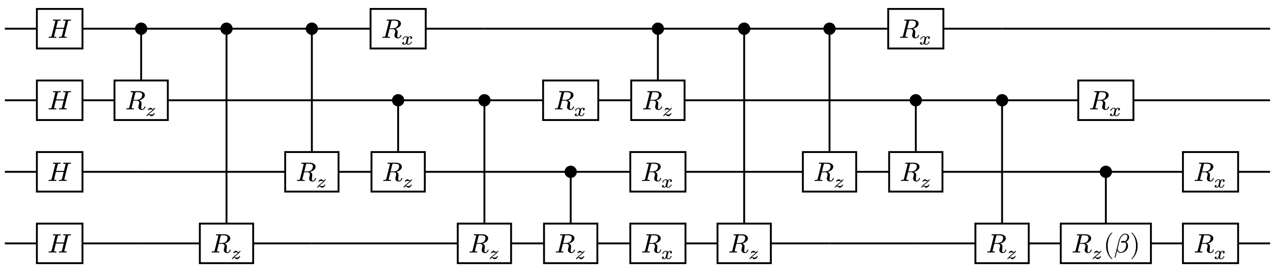


Figure 2: A 4-vertex 3-regular graph.

We build a simplified QAOA protocol for MaxCut on top of it as:



All $R_x = R_x(\alpha)$ and $R_z = R_z(\beta)$. The exact angles used in this circuit and which you are to be scored on are provided in the Cirq program in the `assets` folder on Github, under `QAOA_generator.ipynb`. This program can generate random 3-regular graphs and a simplified QAOA protocol for MaxCut on top of the graph. In this case, the above circuit and graph were obtained with the default arguments of `seed=1` and number of qubits `N=4`.

Your are encouraged to, but not required for scoring, to change the parameters to create larger or an ensemble of graphs and try out your optimization skills!

Circuit #4

To prepare you for the principles behind Circuit #5 and the ongoing theme of QEC (Quantum Error Correction) you'll see in the later circuits given to you we want to challenge you to optimize the encoding of a single qubit into Shor's 9-qubit [[9,1,3]] code. The [[n, k, d]] notation indicates that 9 physical qubits are used to encode 1 logical qubit of "distance" 3, with the relation $d = 2t + 1$ telling us that this code can correct any error on a singular ($t = 1$) qubit.

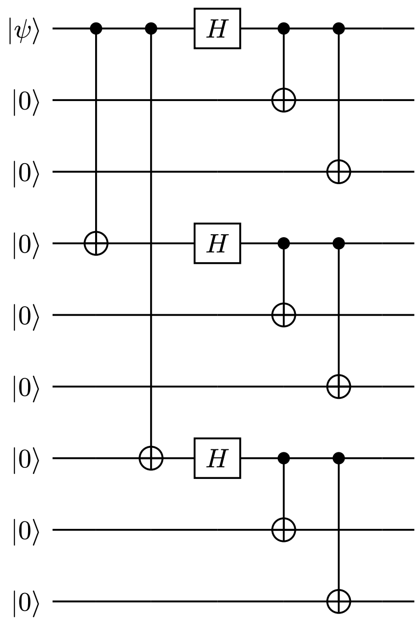
The goal is to take an arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, and encode it in logical states as $\left|\overline{\psi}\right\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle$, where

$$|\overline{0}\rangle = \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)$$

$$|\overline{1}\rangle = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)$$

When doing error correction, we usually omit the logical states, but we show them here to keep things concrete. More information about this encoding and its intuition for operation can be found in the original paper by Shor here

The encoding circuit is as follows:



Circuit #5

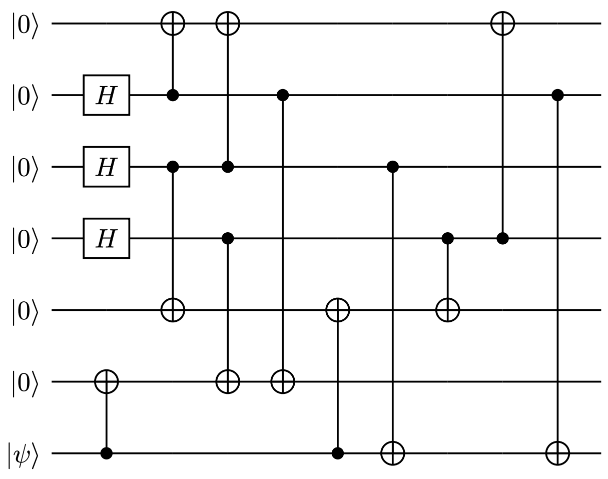
We now continue our theme of QEC by challenging you to optimize the encoding of a logical qubit in the Steane 7-qubit $[[7,1,3]]$ code (also known as the smallest color code!). This is the shorter, simpler encoding used in the recent logical magic state distillation work by QuEra [1].

The goal is to take an arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, and encode it in logical states as $|\overline{\psi}\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle$, where

$$|\overline{0}\rangle = \frac{1}{\sqrt{8}}(|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle$$
$$|0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle)$$
$$|\overline{1}\rangle = \frac{1}{\sqrt{8}}(|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle$$
$$|1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle)$$

More information about this encoding can be found here.

Here is a circuit that should do the job for you (more details here):



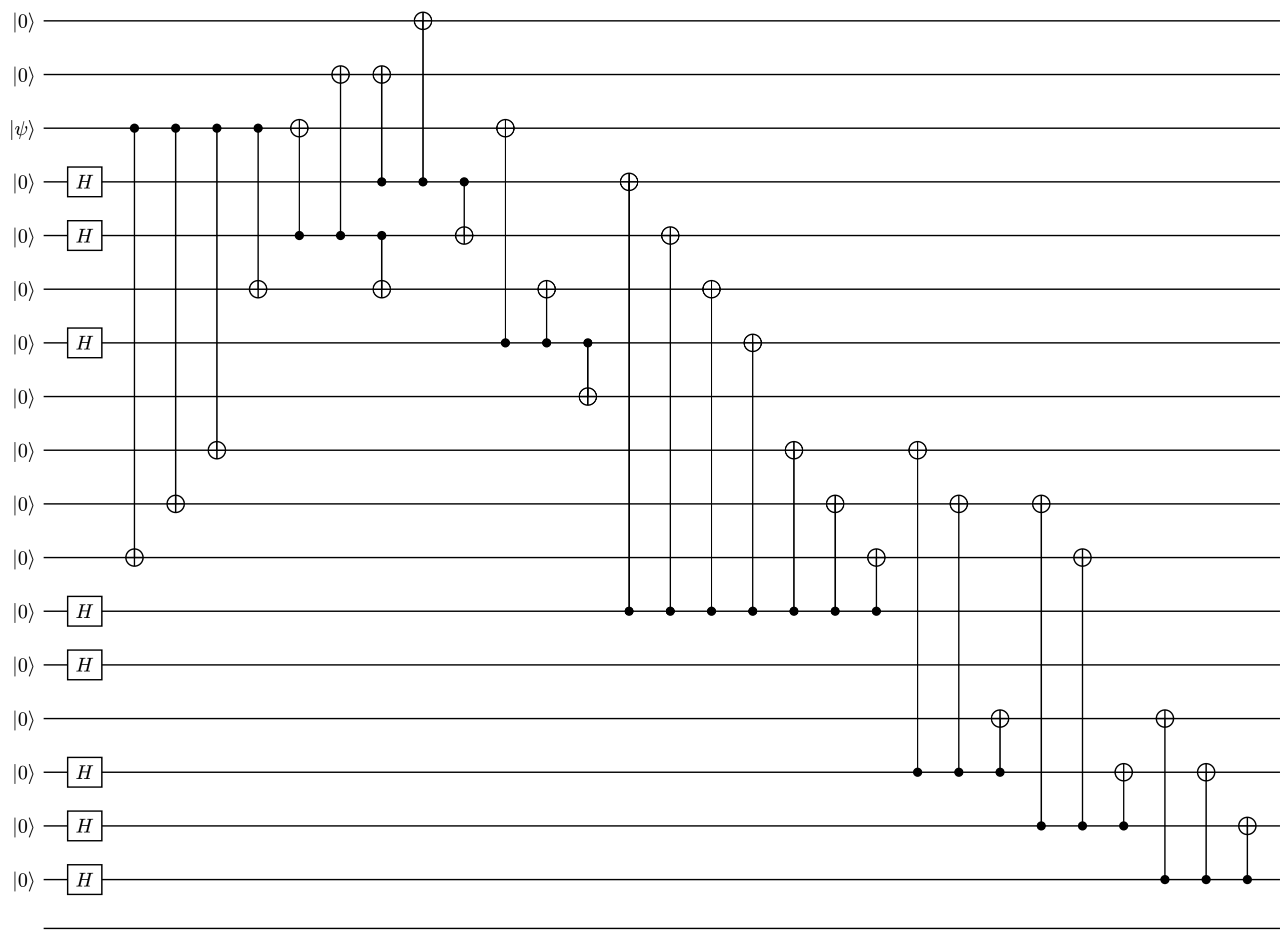
An optimization of this code for different geometric constraints was done in our recent work [1] (see Extended Data Figure 1). Find some tips on how we went about it in the supplementary material, specifically pages 11-12.

For an extra challenge, optimize the circuit for 7 (physical) qubits being encoded in parallel.

Bonus circuit

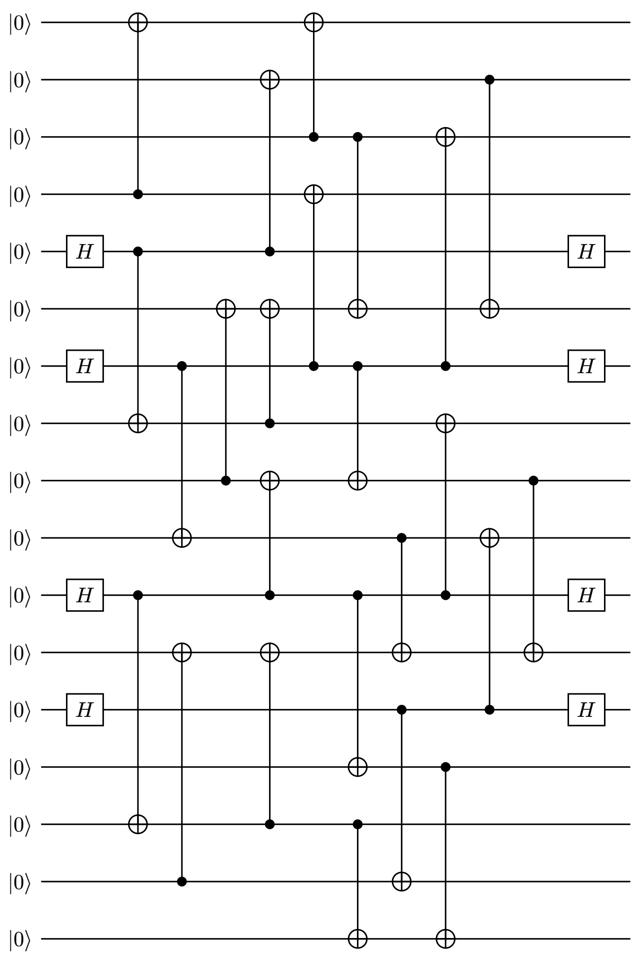
Feeling **bold**? We have some bonus challenges for you. Remember, succeeding at these will only count as tie-breakers, in case multiple teams succeed at solving the base 5 challenges above. So don't get stuck here without finishing the main challenge!

For your first bonus challenge, we propose you do a 17-qubit encoding that brings us to [[17,1,5]] extension of Steane's code. Here goes an un-optimized version of it, for an arbitrary state $|\psi\rangle$ to be encoded into the logical qubit:



Notice this is a bonus challenge. Validating changes in this circuit and scaling number of logical encodings can get pretty hard. You can try and prove that your optimization is both valid and improves the original circuit. Happy optimization!

And if you are really done, here comes a surface code (encoding just the 0 state) for you to optimize as well:



Extra Activities

We will be welcoming side activities that improve visualization, argumentation, or creative pipelines that get you to your solution faster. While these may not count directly to the performance of your cost function, you may want to pursue such directions in order to help you argue that your choice is indeed optimal or better than those of your competitors. These activities may count in case of the need for tie-breaking between competing teams:

Bibliography

[1] P. S. Rodriguez *et al.*, "Experimental Demonstration of Logical Magic State Distillation." [Online]. Available: https://arxiv.org/abs/2412.15165

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

[3] D. Bluvstein *et al.*, "Logical quantum processor based on reconfigurable atom arrays," *Nature*, vol. 626, no. 7997, pp. 58–65, Dec. 2023, doi: 10.1038/s41586-023-06927-3.

[4] W. Schober, "Extended quantum circuit diagrams." [Online]. Available: https://arxiv.org/abs/2410.02946