Compilation Visualizer for Neutral-Atom Gate-Based Quantum Computers

Quinn Manning[1], Daniil Shatokhin[1], Kevin Li[1], David Nizovsky[1], and Dmitrii Khitrin[1]

[1]Atomique, QuEra's 2024 challenge for QRISE

April 2024

1 Introduction

Recently, optical tweezers technology has been utilized to enhance the scalability of neutral-atom quantum computers [3, 2]. These devices facilitate all-to-all connectivity, thanks to the shuttling of qubits within the atomic trap [1], opening up significant possibilities for the application of quantum error correction (QEC) protocols. Neutral-atom quantum processors, such as those described in recent studies [1, 2], exploit the coherent transport of entangled atom arrays, enabling dynamic and programmable qubit interactions. However, the complex dynamics of atom shuttling necessitate sophisticated visualization and debugging tools that can accurately represent and manipulate the quantum state space. Moreover, as quantum algorithms increase in complexity, the ability to visually track and optimize their execution becomes crucial for both theoretical advancements and practical implementations.

In our project, we build upon the established compilation strategies as detailed in the literature [4, 5], which involve the incorporation of single-qubit gates within the frameworks of quantum circuit synthesis and the compilation visualizer.By incorporating single-qubit gates, it facilitates a more granular control over quantum operations, crucial for the accurate execution of generic quantum algorithms.

Moreover, our visualizer places a significant emphasis on logical qubit encoding protocols, and in this paper we focus on the $[[7, 1, 3]]$ Steane code in particular. This approach would transform original logical qubits into fully compiled programs that include comprehensive error correction routines. We demonstrate that Steane code initialization and stabilizer measurements for error correcting can be clearly visualized. The Steane code is instrumental in this context as it enables the construction of fault-tolerant quantum systems by systematically correcting errors that emerge during quantum computations.

In addition to these features, the visualizer also implements circuit decomposition. This process involves the translation of high-level, generic quantum circuits into a sequence of executable operations using gates that are native to the specific quantum hardware being used. Circuit decomposition is critical because it ensures that the synthesized circuits are not only theoretically sound but also practically executable on actual quantum machines. By doing so, the visualizer bridges the gap between theoretical quantum algorithm design and practical implementation on reconfigurable atom arrays, ensuring that algorithms perform as intended on these platforms.

This enhanced capability to decompose circuits into native gates, combined with the incorporation of error correction protocols and precise control mechanisms, positions our visualizer as a robust tool in the development and optimization of quantum algorithms for neutral-atom gate-based quantum computers. It provides a critical foundation for advancing the practical deployment and scalability of quantum computing technologies.

2 Atom Configuration SMT Solver

We took the SMT solver used previously by the UCLA group, and set the goal to build upon it by introducing more visualization capacities as well as single qubit gates. We managed to succeed in these tasks, however it would be good in the future to expand past two qubit gates to triple and quadruple qubit gates, as this multi-qubit functionality is a key advantage of neutral atom computers.

2.1 The Stage Breakdown

We take our circuit consisting of only native 1 qubit, and 2 qubit gates. We separate the 2, and use the SMT solver on the 2 qubit gates as shown previously. We then re-insert the 1 qubit gates where appropriate to preserve order. We ignore commutation optimization for now, but it should be looked on in the future.

3 Single-Qubit Gates

We assume that single-qubit gates can be executed arbitrarily as long as the original circuit order is preserved [2].

3.1 Pulse Parameters

The time evolution of Rydberg arrays is driven by Hamiltonian $H(t)$ defined as

$$H(t) = \frac{\Omega(t)}{2} \sum_i e^{i\phi(t)} |0_i\rangle \langle 1_i| + e^{-i\phi(t)} |1_i\rangle \langle 0_i| - \Delta(t) \sum_i \hat{n}_i + \sum_{i<j} \frac{C_6}{|\vec{x}_i - \vec{x}_j|^6} \hat{n}_i \hat{n}_j$$

where $|0\rangle \equiv |g\rangle$ is a ground state, $|1\rangle \equiv |r\rangle$ is a Rydberg state, $\hat{n}_i = |1_i\rangle \langle 1_i|$, and C_6 is the Rydberg interaction constant. Thus, the pulses are specified by the Rabi frequency Ω, phase of the Rabi drive ϕ, and detuning Δ.

We can obtain our Rabi frequency from the physical limitations of the system. From the axis of rotation, we can obtain our detuning frequency. Combined, along with a given desired angle, we can obtain the required pulse duration and desired waveform to perform a rotation whilst minimizing noise [6].

4 Circuit Transpilation

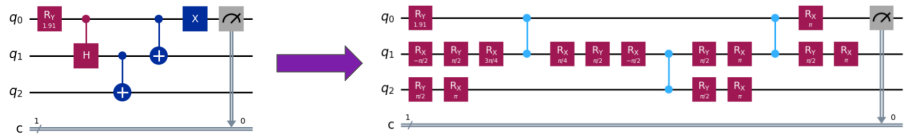
4.1 Circuit Deconstruction Using Qiskit

To deconstruct a generic quantum circuit, we first constrain the gates to a set, $[R_y(\theta), R_x(\theta), CZ]$, which can be implemented on Aquila via the Qiskit transpiler package. The plugin allows us to use the qiskit decomposition and optimizer, which optimizes any circuit around our native neutral atom gates. It first takes any non-native gates, such as $CNOT$, and decomposes them in terms of the native gates, before then optimizing around extra rotations. We are open to both a generic backend option, as well as custom backends the users can make for their personal systems.

Once we have a new circuit in terms of native gates, we seperate the single qubit and two-qubit gates, before sending the 2 qubit gates into the SMT solver, and we re-insert the single qubit gates afterwards. We note that a strong advantage of neutral atom computers is the ability to perform controlled z gates on more than 2 qubits at a time, however we omitted this feature to use the SMT solver in this project.

Rz is also omitted due to not being easily implemented via Raman pulses. Instead of pursuing a separate method of generating a native R_z, we for now stick purely to decomposing R_z in terms of $[R_x, R_y]$

We have included an example decomposition of a 3 qubit w state circuit:



4.2 SMT Order Preserving Circuit Diagram

To temporally orient the viewer with atom movement and gate execution in the visualizer array, we provide, at the bottom of our animation, a quantum circuit diagram generated by Qiskit. A red highlight pans through the circuit at each stage to denote which gate animation corresponds to which qubit in the visualizer array at any given stage. We think this enhances the accessibility of our visualizer by relating the atom array architecture to the familiar circuit representation.

5 QEC Animation Attempt

A QEC implementation should take a generic circuit and convert each qubit in that circuit to a logical bit which can undergo syndrome detection and error correction.

The initialization of Steane code can be easily implemented since compilation accounts for one-qubit gates (and, of course, for CZ gates) as well as measurements (the protocol for which is implemented into our pipeline in "steane.py" which creates two Steane codes for each user input qubit: one logical bit and one logical ancilla bit for syndrome detection). For visualization, the inclusion of readout zone is still required to animate the stabilizer measurements. We were also able to implement a protocol for stabilizer measurements ("stabilizer.py").

One oversight we had was adjusting, a, to indicate whether the qubit is in an AOD or SLM. We did not account for crossing AOD rows and the limit of one qubit per SLM in the initialization of the Steane codes.

6 Animation

Several improvements were made to the circuit visualization. The most significant change was including the quantum circuit image rendered using Qiskit and employing it to demonstrate which gate is being executed in the circuit at each given moment. It is worth noting that an equivalent circuit consisting only of basic gates is showcased, instead of the original input circuit. This detail makes it easier to visualize the gate execution, as its order is consistent with the output of the SMT solver.

The animation support for the single-qubit rotation gates, as well as circuit measurements, was also added. It is implemented by changing the color of the appropriate qubit and showing the parameters of the laser pulse needed to execute the gate.

7 Conclusions

The wealth of connectivity options that atom shuttling brings requires for sophisticated quantum circuit compilation methods. In our work, we improved the existent strategy by including one-qubit operations and mid-circuit measurements in the compilation process. Based on this, we were able to integrate the single-qubit gates into the compilation visualizer.

As we progress, the completion of the Steane code animation within the visualizer will be a priority. This feature is critical as it will provide a dynamic and intuitive representation of error correction processes, thereby enhancing the understanding and debugging capabilities of quantum algorithms. The animation of the Steane code will allow users to visually track error syndromes and corrections in real-time, offering a clearer insight into the fault-tolerance mechanisms at play.

Furthermore, the integration of a readout zone into the visualizer is planned as an essential enhancement. This addition will facilitate the observation and analysis of the final quantum state post-computation, providing crucial data that is imperative for the verification and validation of quantum algorithm outcomes.

References

[1] Dolev Bluvstein et al. "A quantum processor based on coherent transport of entangled atom arrays". In: *Nature* 604.7906 (2022), pp. 451–456.

[2] Dolev Bluvstein et al. "Logical quantum processor based on reconfigurable atom arrays". In: *Nature* 626.7997 (2024), pp. 58–65.

[3] Hannah J Manetsch et al. "A tweezer array with 6100 highly coherent atomic qubits". In: *arXiv preprint arXiv:2403.12021* (2024).

[4] Daniel Bochen Tan et al. "Compiling quantum circuits for dynamically field-programmable neutral atoms array processors". In: *Quantum* 8 (2024), p. 1281.

[5] UCLA VAST Lab. *Optimal Layout Synthesizer of Quantum Circuits for Dynamically Field-Programmable Qubits Array*. 2023. URL: `https://github.com/UCLA-VAST/DPQA`.