2024 Design, Automation & Test in Europe Conference (DATE 2024)

Towards Cycle-based Shuttling for Trapped-Ion Quantum Computers
(Extended Abstract)

Daniel Schoenberger[1] Stefan Hillmich[2] Matthias Brandl[3] Robert Wille[1,2]

[1] Chair for Design Automation, Technical University of Munich, Germany
[2] Software Competence Center Hagenberg GmbH, Austria
[3] Infineon Technologies AG, Germany
daniel.schoenberger@tum.de, stefan.hillmich@scch.at, matthias.brandl@infineon.com, robert.wille@tum.de
https://www.cda.cit.tum.de/research/quantum/

***Abstract*—The *Quantum Charge Coupled Device (QCCD)* architecture offers a modular solution to enable the realization of trapped-ion quantum computers with a large number of qubits. Within these devices, ions can be shuttled (moved) throughout the trap and through different dedicated zones. However, due to decoherence of the ions' quantum states, the qubits lose their quantum information over time. Thus, the shuttling needed for these shuttling operations should be minimized. In this extended abstract, we propose a concept towards a cycle-based heuristic approach to determining an efficient shuttling schedule for a given quantum circuit.**

I. Introduction

Quantum computing [1] as a new paradigm promises to solve certain problems which are computationally intractable on classical computers. Trapped-ion quantum computers are one of the most promising candidates to show quantum advantage in the foreseeable future [2]. However, the scaling of such machines requires corresponding tooling support to exploit their full potential. For ion traps in particular, efficiently moving, i.e., *shuttling*, the ions is an important problem, since unnecessary movement not only increases the required time but also the likelihood of errors due to decoherence. This makes determining efficient schedules of the movement paramount for useful computations in trapped-ion quantum computers. First solutions addressing this problem have been proposed, e.g., in [3]–[7]. However, the considered architectures are comparatively simple and do not cover a large part of possible architectures. In this work, we propose a concept of a cycle-based heuristic approach to generate an efficient shuttling schedule for a given quantum circuit.

II. Background

Trapped-ion quantum computers [5], [8]–[10] utilize ions as qubits, where the quantum state of each ion is manipulated using electromagnetic interactions. To this end, ions are isolated and held in a controlled environment by a combination of radio-frequency and quasi-static electric fields. Within the confines of a trap, multiple ions can be arranged in a chain-like configuration. However, while a single trap suffices for smaller quantum computers, as the number of ions increases, longer gate times give rise to different types of background errors. This makes it challenging to scale to more practical quantum algorithms that require more qubits. To improve the scalability of trapped-ion quantum computers, the *Quantum Charge Coupled Device* (QCCD) architecture [5], [11], [12] proposes to build modular systems, by connecting multiple linear trap sites that each may hold one chain of ions. Then, the main idea of the QCCD paradigm revolves around designating specific regions of the trap for specific functionalities. For instance, all quantum operations are performed in a dedicated *processing zone* that is specifically constructed for efficient qubit operations. The acquired quantum information may then be stored in a *memory zone*, which is shielded from potential disturbances and sources of decoherence. In linear traps, ion chains may block the way of each other, which would require slow interactions like chain reordering and reconfiguration to resolve this issue. To address this problem, junctions are implemented into the systems, connecting linear regions and form two-dimensional architectures. The extension to a second dimension allows ions to avoid the path of other ions without swapping or reconfiguration.

III. Concept for Cycle-based Shuttle Scheduling

Since all quantum operations are performed in the processing zone of the device, ions have to be shuttled to the processing zone to execute a quantum gate on the respective ions. A quantum circuit therefore dictates, which ions have to be moved between the memory zone and the processing zone. In this section, we propose the concept for a solution to determine an efficient shuttling schedule that moves the ion chains in a QCCD device to execute and minimize the execution time of a given circuit.

A. Graph Description

To determine an appropriate shuttling schedule, we represent the architecture of a memory zone and the interface to the processing zone in a QCCD device as an undirected graph. The edges of the graph represent the individual sites of linear traps (each site holding one ion chain). The nodes represent either junctions (termed major nodes) or connections between sites in one linear region (termed minor nodes). On this graph, the physically continuous movement of ion chains is discretized into time steps. At every time step, each chain is present at exactly one edge of the graph.

Example 1. *Consider the top part of Fig. 1. A QCCD architecture is illustrated on the left-hand side. On the right, a corresponding graph is given where the major nodes (green) form a grid of size* 3×3*. Minor nodes (black) mark the three individual sites between two junctions and one inbound and one outbound edge connect the memory zone to a processing zone.*

979-8-3503-4859-0/DATE24/© 2024 EDAA

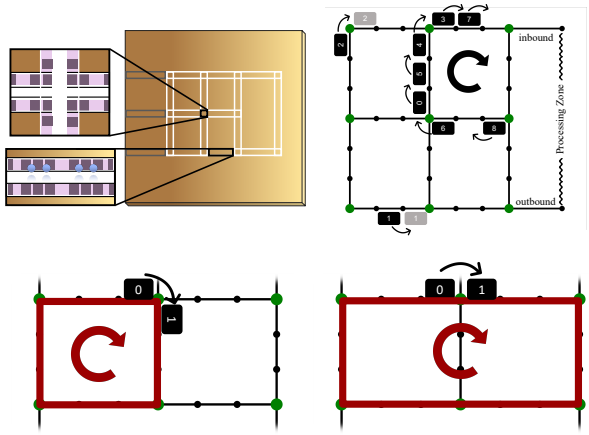

Fig. 1: Cycles in a QCCD architecture

B. General Idea

The defining problem of shuttling arises if an ion chain meets other chains on its path to the processing zone. In a trap filled with multiple ion chains, the shortest path may be blocked, since the chains can not directly swap places in a memory zone. These conflicts have to be resolved by moving the blocking chains away from the paths or change the path of the moving chains. For an increasing number of ion chains in the system this problem becomes increasingly more difficult to solve exactly, i.e., with the minimal number of time steps. To tackle this issue, the topology of the considered modular QCCD architectures offers an intuitive solution: Exploiting *cycles*. Cycles avoid conflicts and still move chains on their shortest path within the memory zone. On the graph representation, we refer to cycles as connected edges that form closed loops. If we form cycles along the shortest paths of the chains and move every chain one step on that cycle, we are able to shuttle individual chains along their optimal path while, at the same time, moving blocking chains away from that path. Additionally, because all junctions can be shuttled through in parallel, one turn of all edges on a cycle only takes one time step.

C. Cycle-based Shuttling

The concept of cycles works in all QCCD architectures, in which closed loops can be formed. To sketch how to use cycles to create a shuttling schedule, we formulate an approach for grid-like architectures. These architectures are connected only by "X"-junctions, i.e., the angles of all junctions are 90°. This means that each face of the graph is a rectangle. We start by moving all chains within two junctions as far as possible on their path. Afterwards, chains may traverse a junction to reach a neighbor linear region. In case the next edge is blocked by another chain, cycles are formed along the shortest path of the shuttling chains and all chains on the cycle are moved in the same direction. The architecture's shape enables constructing cycles in a straightforward way, since every rectangle of the grid forms a closed loop. Depending on the direction of the desired movement, two different cycles are being constructed. All move operations within the memory zone are covered by these two cycles since the memory zone is a symmetric grid.

Example 2. *Since we consider a grid of "X"-junctions, a movement through a junction can either be horizontal or vertical. Both moves are exemplified in the bottom part of Fig. 1. The smallest possible cycle for a vertical move requires exactly one rectangle of the grid (left), while for the horizontal move, the cycle has to be expanded to two rectangles (right).*

As mentioned before, a quantum circuit dictates which ions have to be in the processing zone to execute a quantum gate. To ensure that chains arrive in the correct order at the inbound edge, we implement a priority queue on top of the algorithm. Chains only move along their shortest path and potentially form cycles if all chains, which are needed before the considered chain, are closer to the inbound edge. In other words, the chain that is needed first always moves, while the next one only if it is further away from the inbound edge than the first chain, and so on. Further, cycles may overlap and attempt to move chains in opposite directions. To avoid this, we allow only non-overlapping cycles to move at the same time, prioritized again by which ion is needed sooner.

IV. Conclusions

Trapped-ion quantum computers provide a modular design that promises good scalability with QCCD architectures. Still, efficient classical design tools are required to tap into this potential. In this extended abstract, we proposed a concept for generating efficient shuttling schedules and illustrated the resulting movements within QCCD devices. To this end, we use a graph-based abstraction of the underlying hardware to discretize the problem of moving ion chains. Further, we exploit the topology for conflict-free shuttling through cycles in the graph. This approach provides a blueprint for a scalable solution for determining efficient shuttling schedules of state-of-the-art and future QCCD devices. Remaining edge cases, e.g., shuttling through the processing zone, may be solved straightforwardly by either creating cycles fitted to the processing zone or direct paths to free edges. The full implementation of the proposed approach and the evaluation of the resulting solution are left for future work. These future results will be made available at https://github.com/cda-tum/mqt-ion-shuttler.

Acknowledgments

This work was funded under the European Union's Horizon 2020 research and innovation programme (DA QC, grant agreement No. 101001318 and MILLENION, grant agreement No. 101114305), the State of Upper Austria in the frame of the COMET program, the QuantumReady project (FFG 896217) within Quantum Austria (managed by the FFG), and was part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus.

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. 2016.
[2] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, p. 021 314, 2019.
[3] T. Schmale *et al.*, "Backend compiler phases for trapped-ion quantum computers," in *Int'l Conf. on Quantum Software*, 2022.
[4] J. Durandau *et al.*, "Automated Generation of Shuttling Sequences for a Linear Segmented Ion Trap Quantum Computer," *Quantum*, vol. 7, p. 1175, 2023.
[5] P. Murali, D. M. Debroy, K. R. Brown, and M. Martonosi, "Architecting noisy intermediate-scale trapped ion quantum computers," in *Int'l Symp. on Computer Architecture*, 2020, pp. 529–542.
[6] D. Schoenberger, S. Hillmich, M. Brandl, and R. Wille, "Using Boolean satisfiability for exact shuttling in trapped-ion quantum computers," in *Asia and South-Pacific Design Automation Conf.*, 2024.
[7] L. Schmid *et al.*, *Computational capabilities and compiler development for neutral atom quantum processors: Connecting tool developers and hardware experts*, 2023. arXiv: 2309.08656 [quant-ph].
[8] J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Phys. Rev. Lett.*, vol. 74, pp. 4091–4094, 20 1995.
[9] T. P. Harty *et al.*, "High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit," *Phys. Rev. Lett.*, vol. 113, p. 220 501, 22 2014.
[10] S. Debnath *et al.*, "Demonstration of a small programmable quantum computer with atomic qubits," *Nature*, vol. 536, no. 7614, pp. 63–66, 2016.
[11] D. Kielpinski, C. Monroe, and D. J. Wineland, "Architecture for a large-scale ion-trap quantum computer," *Nature*, vol. 417, no. 6890, pp. 709–711, 2002.
[12] J. M. Pino *et al.*, "Demonstration of the trapped-ion quantum CCD computer architecture," *Nature*, vol. 592, no. 7853, pp. 209–213, 2021.