Synthesis on Atom Computation

Dingchao Gao

Institute of Software Chinese Academy of Sciences

May 23, 2024

Quick Review of Atom Computation

- **Control Mechanisms:**
  - Neutral atoms controlled by laser lattice (stationary)
  - Laser tweezers (for movement)
- **Single Qubit Gate:**
  - $U3 : (\Omega\tau\cos\varphi, \Omega\tau\sin\varphi, \delta\tau)$
- **Multi-Qubit Gate:**
  - $C\mathcal{Z} : 2|gg\rangle\langle gg| - I$

Outline

1. Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures

2. Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors
FPQA-C: A Compilation Framework for Field Programmable Qubit Array

3. Discussion

Outline

1. **Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures**
    - Decomposition Strategy
    - Atom Movement Strategy

2. **Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors**
   FPQA-C: A Compilation Framework for Field Programmable Qubit Array
    - Tan
    - Wang

3. **Discussion**

Decomposition Strategy

- **Native Gate Set**
  - Local CZ gates.
  - Local $R_z(\lambda) : (0, 0, \frac{\lambda}{2})$ gates.
  - Global $GR(\theta, \phi) : (\frac{\theta}{2}\cos\phi, \frac{\theta}{2}\sin\phi, 0)$ gates.
- **Euler-Angle Decomposition**
  - Decompose any single-qubit gate $U3(\theta, \phi, \lambda)$ into Euler angles:

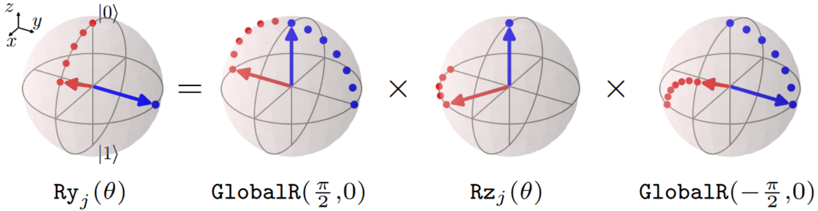
$$U3(\theta, \phi, \lambda) = R_z(\phi) R_y(\theta) R_z(\lambda)$$

Axial Decomposition

- **Decomposition of R_y Gate**
  - Decompose $R_y(\theta)$ gate into global and local gates:

$$\prod_j R_{y_j}(\theta_j) = GR\left(-\frac{\pi}{2}, 0\right)\left[\prod_j R_{z_j}(\theta_j)\right] GR\left(\frac{\pi}{2}, 0\right)$$

  - This involves a net global rotation of π.



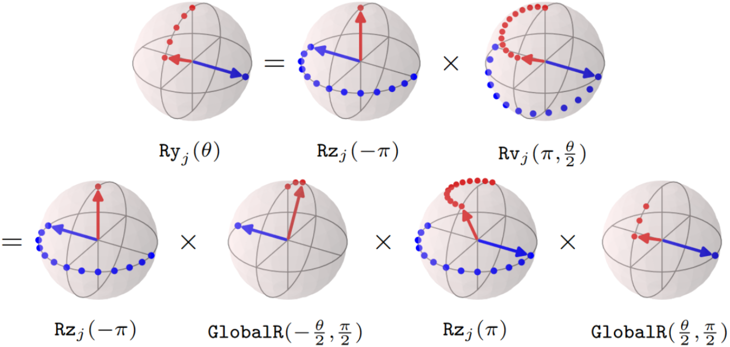
Transverse Decomposition

- **Optimized Decomposition of R_y Gate**
  - Decompose $R_y(\theta)$ gate using minimal global rotation angle:

$$R_y(\theta) = R_v\left(\pi, \frac{\theta}{2}\right) R_z(-\pi)$$

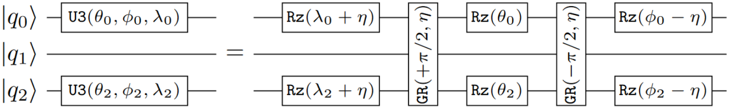
  - Further decompose R_v gate into global and local gates:

$$R_v(\xi, \omega) = GR(\omega, \frac{\pi}{2}) R_z(\xi) GR(-\omega, \frac{\pi}{2})$$

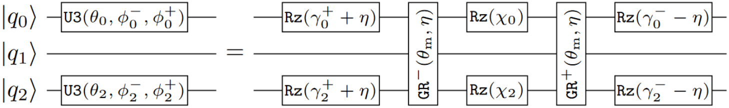


Addition step: Post-Processing and Optimizations

- **Axis of Rotation Adjustment**
  - Change the axis of rotation of global gates to eliminate redundant R_z gates.
- **Gate Merging**
  - Merge consecutive R_z gates to further reduce the gate count.



(a)



(b)

Comparison of Decomposition Methods

- **Axial vs. Transverse Decomposition**
  - Axial decomposition results in a net global rotation of π.
  - Transverse decomposition minimizes global rotation angle to $|\theta|$.
- **Advantages of Transverse Decomposition**
  - Up to 3.5x reduction in global gate pulse duration.
  - Up to 2.9x reduction in single-qubit gate execution time.

Summary

- Efficient decomposition of quantum circuits into native gate sets for neutral atom hardware.
- Transverse decomposition minimizes global rotation angles, leading to significant speedup.
- Post-processing optimizations further reduce gate count and improve execution time.

Outline

1. Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures

- Decomposition Strategy
- Atom Movement Strategy

2. Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors
FPQA-C: A Compilation Framework for Field Programmable Qubit Array

- Tan
- Wang

3. Discussion

Introduction to Atom Movement

- **Goal**
  - Utilize physical atom movement to optimize routing in quantum circuits.
  - Reduce overhead costs associated with traditional SWAP gate decompositions.
- **Challenges**
  - Maintain atom fidelity and avoid interference.
  - Optimize the movement paths to minimize execution time.

Atom Movement Constraints

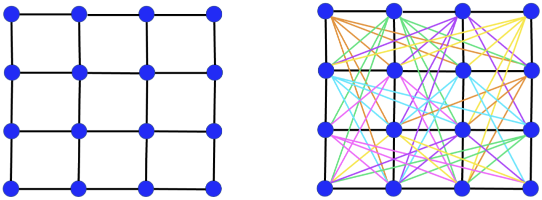
- **Threshold Distance**
  - When moving, an atom must stay at least d_{thr} away from any other atom to avoid interference.
- **Parallel Movement Constraints**
  - Atoms with the same initial x-coordinate can move horizontally together only if they end up with the same final x-coordinate.
  - Atoms with the same initial y-coordinate can move vertically together only if they end up with the same final y-coordinate.
- **Movement Speed**
  - Atoms must be moved at speeds not exceeding **0.55 μm/μs** to maintain qubit fidelity and entanglement.

Atom Array Configuration

- **Initial Mapping**
  - Each program qubit is assigned to a unique hardware qubit to minimize routing operations.
- **Displacement**
  - Atoms can be displaced from grid points by d_{thr} to facilitate SWAP operations without violating distance constraints.

Movement Graph

- **Graph Definition**
  - Nodes represent atom sites.
  - Edges indicate direct movement paths between sites without violating d_{thr}.
  - Edge weights represent the distance between atom sites.



Movement Costs and Optimization

- **Movement Costs**
  - **Duration**
    - Movement duration is the distance traveled divided by movement speed.
  - **Errors**
    - Atom movements incur idle errors but not gate errors.
  - **Parallelism**
    - Maximize parallel movements while adhering to constraints to optimize circuit duration.
- **Optimization Strategy**
  - Use movement graph to determine the best paths for atom movement.
  - Adjust initial displacements to facilitate efficient SWAP operations.

Comparing SWAP-based and Movement-based Routing

- **SWAP-based Routing**
  - SWAP gate decomposition results in high gate count and execution time.
- **Movement-based Routing**
  - Atom movement reduces routing overhead and improves fidelity.
  - Achieves up to 10x speedup and 2x improvement in fidelity.

Summary

- Utilize physical atom movement to optimize routing in neutral atom quantum computers.
- Maintain fidelity and avoid interference with strategic movement constraints and optimizations.
- Significant improvements in execution speed and circuit fidelity compared to traditional SWAP-based routing.

Outline

1. Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures

2. Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors
FPQA-C: A Compilation Framework for Field Programmable Qubit Array

3. Discussion

DPQA/FPQA Architecture

- **Qubit Traps**
  - Qubits are held in optical traps generated by Acousto-Optic Deflectors (AODs) and Spatial Light Modulators (SLMs).
- **Reconfigurability**
  - AOD traps can move in rows and columns, allowing dynamic reconfiguration.
  - AOD rows/columns cannot cross over or overlap during movement.
- **Entangling Gates**
  - Entangling gates are applied using a Rydberg laser, effective when qubits are within a blockade range r_b.

Outline

1. **Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures**
    - Decomposition Strategy
    - Atom Movement Strategy

2. **Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors**
   FPQA-C: A Compilation Framework for Field Programmable Qubit Array
    - Tan
    - Wang

3. **Discussion**

Overview of the Method

- **Goal**
  - Develop a compiler for Dynamically Field-Programmable Qubit Arrays (DPQA).
  - Optimize the placement and routing of qubits to minimize circuit depth.
- **Approach**
  - Discretize the state space and formulate the problem as a Satisfiability Modulo Theories (SMT) problem.
  - Use an SMT solver to find optimal solutions.

Discretization of State Space

- **Space Domain**
  - Discretized into interaction sites to ensure qubits are either paired for gates or idle and well-separated.
- **Time Domain**
  - Discretized into stages where qubit positions are adjusted and gates are executed.

SMT Model Variables

- **stage** s: Represents the stage or step in the quantum circuit execution process.
- **qubit** q_i: Denotes the i-th qubit in the quantum circuit.
- **gate** g_j: Represents the j-th gate in the quantum circuit.
- **site indices** $x_{i,s}, y_{i,s}$: Coordinates (x, y) of the i-th qubit at stage s.
- **array index** $a_{i,s}$: Indicates whether the i-th qubit at stage s is in a stationary SLM trap ($a_{i,s} = 0$) or in a mobile AOD trap ($a_{i,s} = 1$).
- **AOD indices** $c_{i,s}, r_{i,s}$: Column c and row r indices for the i-th qubit at stage s in the AOD grid.
- **time coordinate** t_j: The time coordinate when the j-th gate is applied.

SMT Model Constraints

- **Spatial Constraints**
  - Qubits must be within the defined grid bounds:

$$0 \leq x_{i,s} < X, \quad 0 \leq y_{i,s} < Y \quad \forall i \in [0, N), s \in [0, S)$$

  - AOD moves:

$$(a_{i,s} = 1) \implies (c_{i,s+1} = c_{i,s} \wedge r_{i,s+1} = r_{i,s})$$

  - Site order implying row/column order enforing:

$$(x_{i,s} < x_{i',s}) \implies (c_{i,s} < c_{i',s}), \quad (y_{i,s} < y_{i',s}) \implies (r_{i,s} < r_{i',s})$$

  - AOD rows/columns cannot cross each other:

$$((a_{i,s} = 1) \wedge (a_{i',s} = 1) \wedge (c_{i,s} < c_{i',s})) \implies (x_{i,s+1} \leq x_{i',s+1}),$$
$$((a_{i,s} = 1) \wedge (a_{i',s} = 1) \wedge (r_{i,s} < r_{i',s})) \implies (y_{i,s+1} \leq y_{i',s+1}).$$

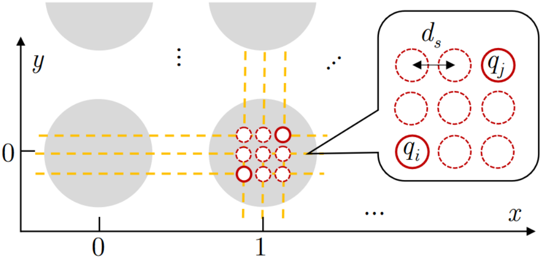
SMT Model Constraints

- **Interaction Constraints**
  - Qubits must be within r_b for entangling gates:

$$((a_{i,s-1} = 1) \wedge (a_{i',s-1} = 1) \wedge (c_{i,s-1} - c_{i',s-1} \geq C_{\mathsf{STK}})) \implies (x_{i,s} > x_{i',s}),$$
$$((a_{i,s-1} = 1) \wedge (a_{i',s-1} = 1) \wedge (r_{i,s-1} - r_{i',s-1} \geq R_{\mathsf{STK}})) \implies (y_{i,s} > y_{i',s}).$$

- **Circuit-Dependent Constraints**
  - Connectivity ensures:

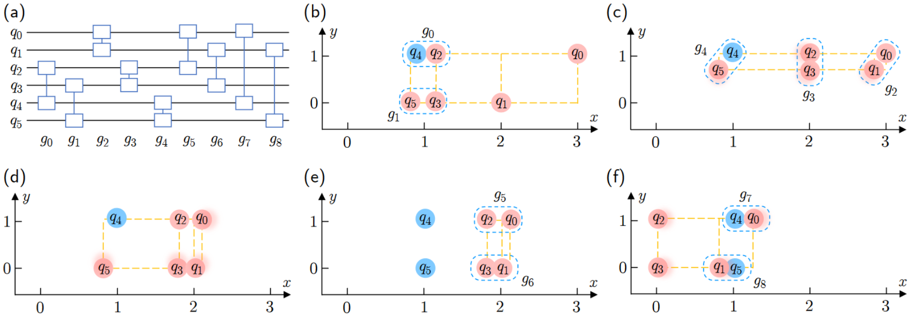
$$(t_j = s) \implies (x_{i,s} = x_{i',s} \wedge y_{i,s} = y_{i',s})$$



Scalability of the Model

- **Variables:** $5NS + G$
- **Constraints:** $O(G^2 + GS + N^2 S)$
- **Bits to Represent Variables:** $NS \log(2XYRC) + G \log(S)$
- **Worst-Case Runtime:** $O((N_{SLM} N_{AOD})^{NS} \cdot S^G)$
- **Shallow Circuit Regime and sparse graphs ($G = O(n)$):**
  - Bits required: $O(N \log(N_{SLM} N_{AOD}))$
  - Constraints: $O(N^2)$

Example of Compiled Quantum Circuit



Greedy Heuristic Method

- **Steps:**
  - Construct a ”single-step” SMT model with two stages.
  - Optimize the number of gates executed in the second stage.
  - Append the solution to the full solution and remove executed gates.
  - Repeat until fewer than 5% of gates remain.
- **Benefit:** Significantly faster than the optimal compiler with some sacrifice in optimality.

Experimental Results

- **Benchmarks:**
  - Evaluated on random graphs.
  - Circuits with 10 to 90 qubits.
- **Main Findings:**
  - For 90-qubit circuits: 5.1x fewer two-qubit gates compared to fixed planar architecture.
- **Error Sources:**
  - AOD movements cause 27x less infidelity.
- **Compiler Performance:**
  - Hybrid approach faster than optimal compiler.
  - Compiles up to 90-qubit circuits within a day.

Summary

- Utilize reconfigurability and non-local connectivity of DPQA for efficient quantum circuit compilation.
- Formulate constraints as an SMT problem for optimal solutions.
- Implement hybrid approach for scalability.

Outline

Compilation Framework

- **Qubit-Array Mapper**
  - Uses MAX k-Cut on a gate frequency graph to minimize SWAP overhead.
  - Coarse-grained mapping of qubits to arrays.
- **Qubit-Atom Mapper**
  - Fine-grained mapping of qubits to specific atoms in the array.
  - Load balance to prevent hardware constraint violations.
- **High-Parallelism Router**
  - Iteratively identifies parallelizable 2Q gates.
  - Decides atom movements and gate executions to maximize parallelism.

Qubit-Array Mapping Method

- **Gate Frequency Graph**
  - Vertices represent qubits, edges represent gates.
  - Edge weights determined by gate frequency.
- **MAX k-Cut**
  - Finds mapping that maximizes inter-array 2Q gates.
  - Greedy algorithm used for approximation.

Qubit-Atom Mapping Method

- **Load Balance Mapping**
  - Ensures balanced distribution of qubits across rows and columns.
  - Avoids constraint violations from dense qubit clusters.
- **Alignment Mapping**
  - Maps qubit pairs with frequent 2Q gates to the same positions in different arrays.
  - Enhances parallelism by aligning high-frequency gates.

High-Parallelism Router

- **Non-Dependent Frontier Gates**
  - Identifies and schedules gates with no dependencies.
- **Constraint Checks**
  - Ensures movements do not violate spatial or interaction constraints.
  - Greedily adds gates to the parallel set, checking each for legality.

Experimental Setup

- **Benchmarks**
  - Diverse set of benchmarks including QASMBench, SupermarQ, Quantum Simulation, and QAOA circuits.
- **Metrics**
  - 2Q gate count, circuit depth, and fidelity.
  - Comparisons with IBM superconducting, FAA with long-range gates, and other topologies.

Unique Experimental Approach

- **Comprehensive Simulations**
  - Evaluates logical error rates, execution times, and physical qubit requirements.
  - Includes realistic modeling of movement overheads (heating, cooling, decoherence, atom loss).
- **Scalability and Performance**
  - Demonstrates significant reductions in 2Q gate count and circuit depth.
  - Achieves up to 1000x faster compilation speed compared to solver-based methods.

Summary

- FPQA-C effectively addresses qubit mapping, atom movement, and gate scheduling in FPQA.
- Incorporates innovative methods to handle hardware constraints and optimize performance.
- Experimental results highlight the advantages of FPQA-C over traditional approaches in terms of gate count, depth, and fidelity.

Outline

1. Decomposing and Routing Quantum Circuits Under Constraints for Neutral Atom Architectures

2. Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors
FPQA-C: A Compilation Framework for Field Programmable Qubit Array

3. Discussion

CZ ? C$\mathcal{Z}$

- $C\mathcal{Z}$: $2|gg\rangle\langle gg| - I$, while CZ : $I - 2|11\rangle\langle 11|$
- in gererally:

$$
\begin{aligned}
|gg\rangle &\rightarrow |gg\rangle \\
|ge\rangle &\rightarrow |ge\rangle e^{i\phi_1} \\
|eg\rangle &\rightarrow |eg\rangle e^{i\phi_1} \\
|ee\rangle &\rightarrow |ee\rangle e^{i(2\phi_1+\pi)}
\end{aligned}
$$

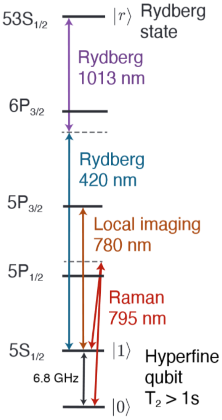


Figure: Level structure for ^{87}Rb atoms, with the relevant atomic transitions employed in *Quera paper*.

How to use mcz

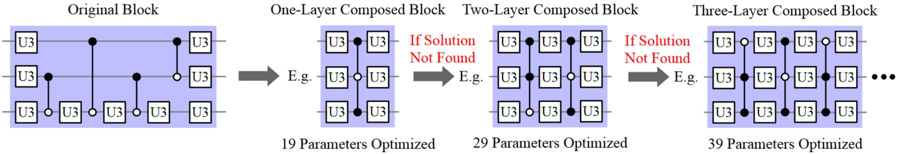


Figure: main idea in Gyser paper

How to use mcz

-
- More flexible U3 gate decomposition
- Oracle design using z group gate