Synthesis on Atom Computation

Dingchao Gao

Institute of Software Chinese Academy of Sciences

June 6, 2024

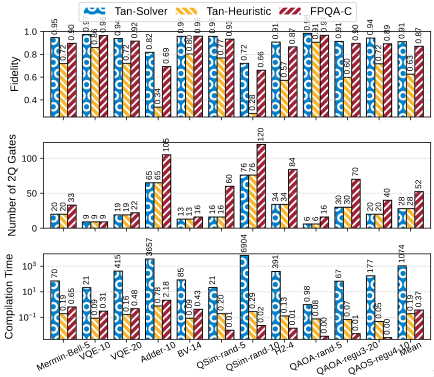
Outline

1. Compilation for Dynamically Field-Programmable Qubit Arrays with Efficient and Provably Near-Optimal Scheduling

2. Computational capabilities and compiler development for neutral atom quantum processors—connecting tool developers and hardware experts

Related Works

- ***Compiling Quantum Circuits for Dynamically Field-Programmable Neutral Atoms Array Processors***: Utilizes Z3 MST, but lacks scalability and fidelity considerations.
- ***FPQA-C: A Compilation Framework for Field Programmable Qubit Array***: Employs a rule-based algorithm, offering good scalability, but does not achieve the optimal count of 2Q gates.



Overview

- Background: Quantum computing with neutral atoms has advanced rapidly.
- Fidelity:

$$f = (f_1)^{g_1} \cdot \overbrace{(f_2)^{g_2} \cdot (f_{\text{exc}})^{|Q|S-2g_2}}^{\text{two-qubit gate}} \cdot \overbrace{(f_{\text{trans}})^{N_{\text{trans}}}}^{\text{atom transfer}} \cdot \overbrace{\prod_{q \in Q} (1 - T_q/T_2)}^{\textit{decoherence}}.$$

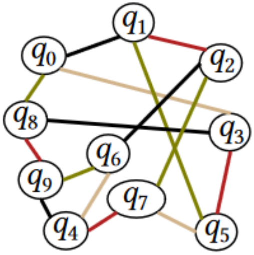
- Significant scalability: experiments with up to 6,100 qubits.
- The compilation process is broken down into three tasks: scheduling, placement, and routing.

Qubit Num	Gate Num	Scheduling	Placement	Routing	Codegen	Total
30	45	0.0008	137.32	0.0057	0.0184	137.35
60	60	0.0017	141.23	0.0124	0.0379	141.28
90	135	0.0023	144.43	0.0304	0.0630	144.52

Table: Timing Results for Different Qubit and Gate Numbers

Scheduling

- Scheduling is crucial for determining the sequence of operations. Graph edge coloring is used to model the scheduling problem.
- Each edge represents a two-qubit gate.
- Colors represent different stages.
- The goal is to minimize the number of stages while ensuring no two adjacent edges share the same color.



qubit interaction graph

stage	gate
0	g_3 g_6 g_7 g_{11}
1	g_0 g_5 g_9 g_{12}
2	g_2 g_4 g_8 g_{14}
3	g_1 g_{10} g_{13}

edge-color schedule

Scheduling: Graph Edge Coloring

Theorem (Vizing's Theorem)

For any simple graph G with maximum degree Δ, the chromatic index $\chi'(G)$ satisfies:

$$\Delta \leq \chi'(G) \leq \Delta + 1$$

where $\chi'(G)$ is the minimum number of colors needed to color the edges of G.

- There exists an algorithm with runtime $O(|V| \cdot |E|)$ that provides an edge coloring $\phi : E \to \{0, 1, 2, \dots, \Delta(G)\}$.
- The maximum gate count is $\binom{n}{2}$. Thus, the time complexity of scheduling is $O(n^3)$.

Placement

- Placement refers to assigning qubits to physical locations.
- Optimal placement minimizes the distance between interacting qubits.
- This reduces the need for long-distance routing, which can lower fidelity.

Formula (cost function)

$$\sum_{g(q,q')\in G} w_g \cdot dist(m(q), m(q'))$$

where w_g is the weight for gate g, m is the placement function from qubits to interaction sites, and dist is the ***Euclidean distance****.*

Placement Strategies

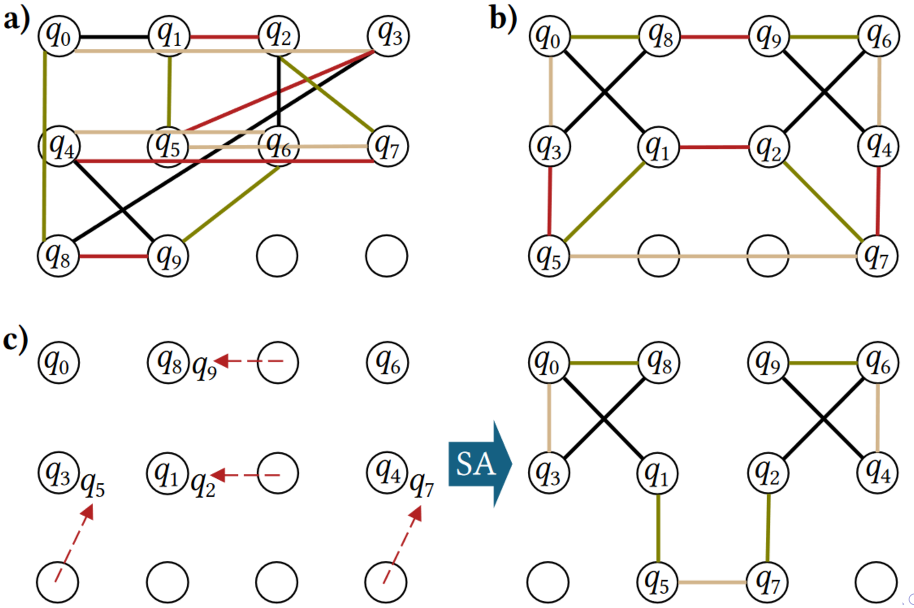
- Use of simulated annealing algorithms to find near-optimal solutions with a constant runtime.
- Balancing between computational efficiency and placement quality.

$$x \in \left[0, \max\left(\lfloor\sqrt{n}\rfloor + 4, x_{\max}\right)\right], y \in \left[0, \max\left(\lfloor\sqrt{n}\rfloor + 4, y_{\max}\right)\right]$$

Formula (weight)

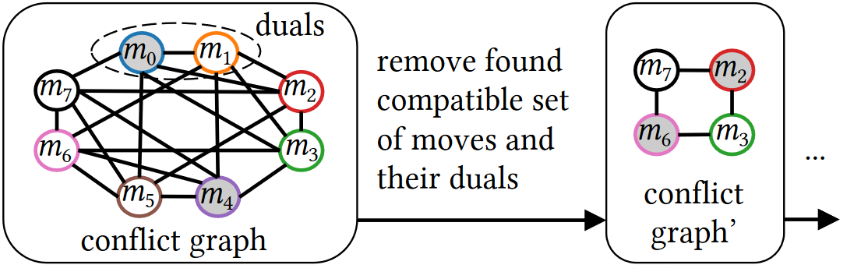
$$w_g = \begin{cases} 1, & \textit{static placement} \\ \max\left(0.1, 1 - 0.1 s_g\right), & \textit{dynamic placement} \end{cases}$$

Placement example



Routing

- Routing involves determining paths for qubits to move during computation. Ensuring minimal delay and avoiding congestion are key goals.
- A conflict graph represents the conflicts during the routing process.
- Nodes represent qubits movements, and edges represent conflicts.
- The problem is The Maximum Independent Set (MIS) problem, where one seeks to find the largest set of vertices in a graph such that no two vertices in the set are adjacent.



Greedy Algorithm for Bounded Degree Graphs

1). putting all vertices in a list (sorted by distance).

2). adding the first vertex to the IS.

3). removing all its neighbors from the list, and continuing 2-3.

Theorem

In bounded degree graphs, there are effective approximation algorithms with constant ratios. For example, a greedy algorithm that forms a maximal independent set by repeatedly choosing the vertex with the minimum degree and removing its neighbors achieves an approximation ratio of $(\Delta + 2)/3$ *for graphs with maximum degree* Δ. *Approximation hardness bounds for these cases were shown by Berman and Karpinski (1999).*

Routing Complexity

For the number of qubits n, the maximum number of gates is $n/2$, so the number of vertices is at most n ($|V| \leq n$):

- Checking conflicts for all pairs of vertices requires $O(|V|^2)$ time.
- Sorting the vertices requires $O(|V| \log |V|)$ time.
- The greedy algorithm requires $O(|V|^2)$ time. In the worst case, the greedy algorithm needs to be run $O(|V|)$ times.
- **In total, there can be $O(n)$ Rydberg stages, resulting in a routing time of $O(n^4)$.**

Only construct a graph on the first K vertices in the lists ($|V| = K$):

- **The windowed routing takse $O(n^2 \log n + n^2 K^2)$.**

Results and Comparison

- The compiler, Enola, shows significant improvements in performance.
- Achieves 3.7X stage reduction compared to existing works.
- Demonstrates 5.9X improvement in fidelity on benchmark sets.
- Highly scalable, capable of compiling circuits with up to 10,000 qubits within 30 minutes.
- Outperforms the current state of the art, OLSQ-DPQA.

Conclusion

- The compilation process for dynamically field-programmable qubit arrays involves scheduling, placement, and routing.
- The method provide near-optimal solutions for scheduling ($S_{opt} + 1$) and efficient strategies for placement and routing.
- Enola compiler achieves significant improvements in stage reduction and fidelity.
- Future work includes further optimization and exploring additional constraints.
- Open source availability: `https://github.com/UCLA-VAST/Enola`

Outline

1. Compilation for Dynamically Field-Programmable Qubit Arrays with Efficient and Provably Near-Optimal Scheduling

2. Computational capabilities and compiler development for neutral atom quantum processors—connecting tool developers and hardware experts

Compilation subroutines

Definition (Synthesis)

Given a quantum computation $U \in \mathbb{C}^{2^n \times 2^n}$ and the native platform gate set Σ^{native}, *synthesis* is the task to find a gate sequence

$$\tilde{U} = g_{N-1} \circ \cdots \circ g_0$$

with all $g_0, \ldots, g_N \in \Sigma^{\text{native}}$ and $U = \tilde{U}$ up to some small error.

Definition (Mapping)

Given a quantum circuit $U = g_{N-1} \circ \cdots \circ g_0$ on circuit qubits $\mathbf{Q}$ and a hardware configuration with physical qubits $\mathbf{P}$ and coupling map $G(\mathbf{P}, \mathbf{E})$. The task of *mapping* is to find a bijective function $f : \mathbf{Q} \to \mathbf{P}$ and an insertion of MOVE and SWAP operations such as

$$U = \cdots \circ \text{MOVE}(q_i) \circ \text{SWAP}(q_j, q_k) \circ g(q_i, q_j) \circ \cdots$$

Definition (Scheduling)

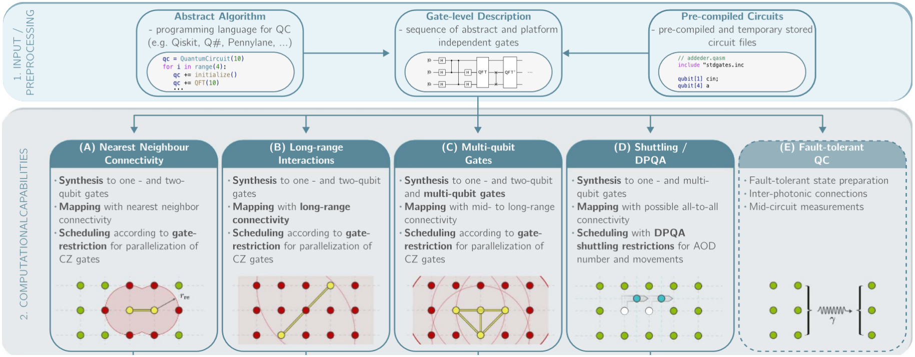
Given a quantum circuit U and its corresponding DAG representation D, the objective of *scheduling* is to determine the optimal timing for the gates to be executed while preserving the integrity of the DAG up to commutation rules.

Figures of merit

- Gate count
- Operations count
- Fidelity and runtime

$$P(U) = \exp\left(-\frac{t_{\text{idle}}}{T_{\text{eff}}}\right)\prod_{i=0}^{\tilde{N}} \mathcal{F}_{O_i}, \quad U = O_{N-1} \circ \cdots \circ O_0$$

Atom computation capabilities



Abstract Algorithm
- programming language for QC (e.g. Qiskit, Q#, Pennylane, ...)

```
qc = QuantumCircuit(10)
for i in range(4):
   qc += initialize()
   qc += QFT(10)
   ...
```

Gate-level Description
- sequence of abstract and platform independent gates

Pre-compiled Circuits
- pre-compiled and temporary stored circuit files

```
// addeder.qasm
include "stdgates.inc

qubit[1] cin;
qubit[4] a
```

(A) Nearest Neighbour Connectivity
- **Synthesis** to one - and two-qubit gates
- **Mapping** with nearest neighbor connectivity
- **Scheduling** according to **gate-restriction** for parallelization of CZ gates

(B) Long-range Interactions
- **Synthesis** to one - and two-qubit gates
- **Mapping** with **long-range connectivity**
- **Scheduling** according to **gate-restriction** for parallelization of CZ gates

(C) Multi-qubit Gates
- **Synthesis** to one - and two-qubit and **multi-qubit gates**
- **Mapping** with mid- to long-range connectivity
- **Scheduling** according to **gate-restriction** for parallelization of CZ gates

(D) Shuttling / DPQA
- **Synthesis** to one - and multi-qubit gates
- **Mapping** with possible all-to-all connectivity
- **Scheduling** with **DPQA shuttling restrictions** for AOD number and movements

(E) Fault-tolerant QC
- Fault-tolerant state preparation
- Inter-photonic connections
- Mid-circuit measurements