Quantum Compiling

Michele Amoretti$^{1,2}$
1. Department of Engineering and Architecture - University of Parma, Italy
2. Quantum Information Science @ University of Parma, Italy
www.qis.unipr.it
Contact: michele.amoretti@unipr.it

Quantum Computers @ Parma 28 October 2019
Outline

● Quantum Compilation Problem
● State of the Art
● Quantum Software @ UniPr
Quantum Compilation Problem

Fast, device-aware implementation of quantum algorithms

A good quantum compiler must translate an input program into the most efficient equivalent of itself, getting the most out of the available hardware.

The quantum compilation problem in general is NP-Hard.
Quantum Compilation Problem

On noisy intermediate-scale quantum (NISQ) devices:

- **Gate synthesis** = decomposition of an arbitrary unitary operation into a sequence of gates from a discrete set
- Compliance with the **coupling map**
- Noise awareness

Quality indicators:

- Circuit **depth**
- **Gate count**
- **Fidelity** of quantum states
Quantum Compilation Problem

- Initial mapping of logical qubits to physical ones
  Necessary but not sufficient

- Exchange of qubit states using SWAP gates
  Increment of circuit depth
Example: GHZ State

The Greenberger-Horne-Zeilinger state for $n$ qubits:

$$|0\rangle \otimes^n + |1\rangle \otimes^n \quad \frac{1}{\sqrt{2}}$$

It is an entangled quantum state that plays a prominent role in several quantum algorithms.

Example: GHZ State

The ideal quantum circuit that generates the GHZ state requires only H and CNOT gates.
Example: GHZ State

Ideal vs. compiled 8-qubit GHZ circuit (Qiskit v.0.12)
Outline

● Quantum Compilation Problem
● State of the Art
● Quantum Software @ UniPr
State of the Art

Zulehner et al., An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures, IEEE TCAD vol.38 no.7, 2019

- Partition circuit into layers
- For each layer find a CNOT compliant mapping
  - $m!/(m-n)!$ possible mappings with $m$ physical qubits and $n$ logical qubits
  - A* search algorithm
  - minimize additional operations to switch between subsequent mappings
State of the Art

Zulehner et al., 2019

A* search algorithm:

- start from current layer mapping
- explore possible subsequent mappings employing combinations of parallel SWAPs
- each node $x_i$ represents a partial mapping
- expand node $x_i$ that minimizes cost function $f(x_i) = g(x_i) + h(x_i)$
  - $g(x_i)$ is the fixed cost of current node as $f(x_{i-1}) + 7 \times \#SWAPs$
  - $h(x_i)$ is a heuristic to estimate the distance from a goal node
  - enhance with lookahead capabilities
State of the Art

Childs et al., *Circuit Transformations for Quantum Architectures*, TQC 2019

Architecture: $G(V,E)$

Circuit: directed acyclic graph, set of qubits $Q$ ($|Q| \leq |V|$), size $|C|$

1) **Greedy swap circuit transformation**: Time complexity $O(|C||E|diam(G))$
   - Best approach for minimizing circuit size

2) **Architecture-aware circuit transformation**:
   - qubit movement: addressed by a SWAP-based *permuter*, time complexity $< t_p$
   - qubit placement: addressed by a *mapper*, time complexity $< t_m$

   Time complexity $O(|C|(t_m + |V| + t_p))$

   Best approach for minimizing depth on grid architectures

Quantum Computers @ Parma, 28/10/2019
State of the Art

Jandura, 2018 (LookaheadSwap in Qiskit)

Mapping = logical qubits ↔ physical qubits

Given a mapping and a list of upcoming CNOTs, repeat 4 times:

1. Find 4 most promising swaps and calculate new mapping for each of them
2. For each mapping count how many CNOTs can be executed and remove them from the list

From the $4^4 = 256$ final mappings, choose the one that allowed for the most CNOTs to be executed and execute the first swap on the path to this mapping.

Then start the whole algorithm again, until the circuit is completed.
State of the Art

Li et al., Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices, ASPLOS ‘19, 2019

**SWAP-based BidiREctional heuristic search (SABRE)**

- Preprocessing
  - Compute distance matrix $D$ over the coupling map
  - Generate DAG from circuit to represent two-qubit gates dependencies
  - Initialize the front layer $F$ as the set of two-qubit gates without unexecuted predecessors
  - Generate random initial mapping
Li et al., Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices, ASPLOS ‘19, 2019

SWAP-based heuristic

- Iterate over front layer $F$
  - Remove executable gates from $F$ and add their successor to $F$
  - For those gates in $F$ that cannot be executed, search for SWAPs to insert and update mapping
  - Select best SWAP to insert using an heuristic cost function based on distance matrix $D$
- Stop when $F$ is empty
State of the Art

Li et al., Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices, ASPLOS ‘19, 2019

Reverse traversal for bidirectional mapping

- From the original circuit generate a reverse circuit
- Use the final mapping of the first SWAP-based compilation as the initial mapping to compile the reverse circuit
- Use the final mapping of the reverse compilation to recompile the original circuit
- This initial mapping has the quality of taking into account all the gates information
Outline

- Quantum Compilation Problem
- State of the Art
- Quantum Software @ UniPr
Quantum Software @ UniPr

Team

- Michele Amoretti (PhD, associate professor)
- Davide Ferrari (PhD student)

Topics

- high performance computing (classical and quantum)
- quantum compiling
- quantum networking (protocols for distributed monitoring, anonymity, leader election, etc.)

Source code: [https://github.com/qis-unipr](https://github.com/qis-unipr)
Quantum Software @ UniPr

Research related to IBM Q

European Quantum Technology Conference (EQTC19)
The first international conference of the QT Flagship
Grenoble, France, February 18th - 22nd (2019)

Efficient quantum compiling for entanglement-based machine learning on IBM Q devices

Davide Ferrari*†,‡,§ and Michele Amoretti*†,§

Efficient Quantum Compiling for Quantum Chemistry Simulation on IBM Q

D. Ferrari,† I. Tavernelli,‡ M. Amoretti*

*michele.amoretti@unipr.it, †Department of Engineering and Architecture - University of Parma, Italy, ‡IBM Zurich Research Laboratory
Deterministic Pattern-Oriented Quantum Compiler (DPQC)

The DPQC software is the implementation of efficient and deterministic strategies for the compilation of quantum circuits characterized by peculiar sequences (patterns) of two-qubit operators.

By compilation we mean the transformation of abstract quantum circuits into circuits suited to the characteristics of the available hardware device, preserving their correctness and trying to maximize their efficiency.

Among the possible characteristics of a quantum computer, the DPQC software considers the coupling map and tries to minimize circuit depth and gate count.

SIAE filing n.2019002328, 11/9/2019
GHZ Results

Quantum Computers @ Parma, 28/10/2019
Quantum Chemistry Circuits

Used to compute the ground state wave function of simple molecular systems such as:

- Hydrogen $\text{H}_2$ with 4 qubits
- Lithium Hydride $\text{LiH}$ with 12 qubits
- Water $\text{H}_2\text{O}$ with 14 qubits
Characterized by repeated sequences of CNOTs
CNOT Sequences Without Gaps

- Algorithm `chain_layout()` analyzes the coupling map and orders the qubits in a chain.
- Every logical qubit $q_i$ is associated with the $i$-th physical qubit in the chain.
- Initial mapping satisfies all CNOTs.

- In general, the problem of finding a path that visits every node only once is NP-complete.
- `chain_layout()` exploits the qubit indexing and the graph structure to find a path with a deterministic and efficient approach.
- Computational complexity $O(m)$, where $m$ is the number of physical qubits.
CNOT Sequences Without Gaps

IBM Q20 Ideal

IBM Q20 Tokyo

Quantum Computers @ Parma, 28/10/2019
CNOT Sequences With Gaps

- Initial mapping is not enough
- Algorithm `path()` moves the qubits $q_1$ and $q_2$ involved in a remote CNOT close to each other:
  - iteratively computes the SWAP path between $q_1$ and $q_2$ neighbors
  - evaluates the distance between $q_x$ and $q_y$ as $|x - y|$ with $x$ and $y$ being their physical qubit indices in the coupling map
  - uses a cost function to choose which qubit to add to the path

$$f(x, y) = |x - y| \times \text{SWAP\_DEPTH}$$

- updates $q_1$ and $q_2$ mapping after every iteration
- Computational complexity $O(m)$
# UCCSD Results

<table>
<thead>
<tr>
<th>IBM Q20 Tokyo</th>
<th>Circuit Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Circuit Name</strong></td>
<td>Input Circuit</td>
</tr>
<tr>
<td>H2_UCCSD</td>
<td>82</td>
</tr>
<tr>
<td>LiH_UCCSD</td>
<td>8845</td>
</tr>
<tr>
<td>H2O_UCCSD</td>
<td>15388</td>
</tr>
<tr>
<td>Random20_UCCSD</td>
<td>125638</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IBM Q20 Tokyo</th>
<th>Compilation Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Circuit Name</strong></td>
<td>DPQC</td>
</tr>
<tr>
<td>H2_UCCSD</td>
<td>10.18</td>
</tr>
<tr>
<td>LiH_UCCSD</td>
<td>246.52</td>
</tr>
<tr>
<td>H2O_UCCSD</td>
<td>346.32</td>
</tr>
<tr>
<td>Random20_UCCSD</td>
<td>2351.45</td>
</tr>
</tbody>
</table>
Sequence Offset

8 qubits chain with offset=12

- $0 \leq \text{offset} < (m-n)$

- Used to map the qubits in the circuit to the qubits in the chain in the interval $\{0,..,n + \text{offset}\}$ instead of $\{0,..,n\}$
Sequence Offset

![Graph showing the UCCSD LiH sequence offset with depth and offset values.](image)
Repeated sequences of inverted CNOT cascades

- CNOT cascades are equivalent to CNOT sequences
- Algorithm `analyze_circuit()` detects CNOT cascades and turns them into CNOT sequences
- Computational complexity $O(gn)$ with $g$ being the number of gates in the circuit
CNOT Cascade

Inverted CNOT cascade

- CNOT gates can be inverted by applying $H$ gates before and after involved qubits
- `analyze_circuit()` also detects inverted CNOT cascades and turns them into CNOT cascades
- Algorithm `gate_cancellation()` deletes double CNOT and double $H$ gates
- Computational complexity $O(dn^2)$
## RyRz Results

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>IBM Q20 Tokyo</th>
<th>Circuit Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Input Circuit</td>
</tr>
<tr>
<td>H2_RyRz</td>
<td>73</td>
<td>56</td>
</tr>
<tr>
<td>LiH_RyRz</td>
<td>223</td>
<td>216</td>
</tr>
<tr>
<td>H2O_RyRz</td>
<td>273</td>
<td>256</td>
</tr>
<tr>
<td>Random20_RyRz</td>
<td>393</td>
<td>376</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IBM Q20 Tokyo</th>
<th>Compilation Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DPQC</td>
</tr>
<tr>
<td>H2_RyRz</td>
<td>0.81</td>
</tr>
<tr>
<td>LiH_RyRz</td>
<td>2.26</td>
</tr>
<tr>
<td>H2O_RyRz</td>
<td>3.20</td>
</tr>
<tr>
<td>Random20_RyRz</td>
<td>6.10</td>
</tr>
</tbody>
</table>
Future Work

- Testing the proposed approach with the **new coupling maps**
- Using extra qubits
- Implementing an **automatic offset selection** algorithm
- Studying **other circuit patterns**
Future Work

While effective, circuit depth and gate count are only pseudo-objectives. In reality, what matters is the **fidelity** of a computation when run on actual quantum hardware.

NISQ compilers excel when more **information** is made available to them **from the device**.

IBM Q **device properties** are shared openly and can be benchmarked, resulting in a bunch of compiler innovations.


Thank You!

Questions?