Software for Advancing Quantum Computing

Yufei Ding
Assistant Professor
Department of Computer Science
University of California, Santa Barbara
Software vs. Hardware

- How to divide the job between software and hardware optimizations?
  - Example: practical application, say Shor’s algorithm.
  - Example: QEC with 2 logic qubits with gate error rate at the level of $10^{-6}$ ~ $10^{-9}$.

- Be realistic.
  - More powerful hardware makes software easier.
  - More powerful software makes hardware easier.

- To find the boundary for software and hardware optimization is like an art and the boundary will change over time due to:
  - Overall Cost budget
  - State-of-the-art Hardware technology.
  - Specific Application.
  - Our understanding of the physics and system designs.

- A good software will
  - help resolve hardware constraints.
  - quantitively tell us how good the hardware need to be.
Lessons Learned from Classical World: Classical Compiler

Source program

Target program


Monolithic Compiler Two-pass Compiler Modular Optimizing Compiler Multicore/GPU Compiler Autotuning Compiler ML/Quantum Compiler

+ Error Diagnosis + Code Optimization

Our observation: From Monolithic to Modular to Architecture(Input)-aware to Domain-Specific.

- Modular with a good set of abstractions (intermediate representation) allows easy adaption.
- Get used to specialization towards whatever is needed.
My View of Three Important Set of Software Toolchains

My focus is more at top level:

- Toward Hardware Scaling Up.
- Toward Quantum Error Correction.

Other interesting toolchains for optimization at lower level:
e.g., EDA for qubit engineering.
Quantum Scaling-Up Strategies

Horizontal Scaling

Chiplet Quantum Computing
Distributed Quantum Computing

Vertical Scaling

- Ion Trap
  Gate Time: 1~100 μs

- Photonic
  Gate Time: 1 ns

- Superconducting
  Gate Time: 10~100 ns

Physics

CS

Software ecosystem would be different for different hardware architectures.
**A Synthesis Framework for Stitching Surface Code with Superconducting Quantum Devices**

Anbang Wu, Gushu Li, Hezi Zhang, Gian Giacomo Guerreschi, Yufei Ding, and Yuan Xie.

**Key Novelty:**
We can use a compiler to efficiently mitigate the constraints of limiting sparse 2-qubit connection on hardware.

**Software Infrastructure:**
1. Initial mapping
2. Efficient routing

---

**Paulihedral: A Generalized Block-Wise Compiler Optimization Framework For Quantum Simulation Kernels**

Gushu Li, Anbang Wu, Yunong Shii, Ali Javadi-Abhari, Yufei Ding, Yuan Xie.

- **Intermediate Representation (IR):**
  - Between source code and machine code
  - Right level of abstraction for efficient analysis and optimization.

**Key Novelty:**
What is a Good Quantum Intermediate representation? Principles for Guidance

**Abstraction:** Multi-Qubit Operation, i.e., Multi-Qubit Gate.

- Expressiveness: Efficiently + Universal
- Beneficial
- Easy Lowering

---

**UC SANTA BARBARA — PICASSO Lab**
**Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices**

Gushu Li, Yufei Ding, Yuan Xie

**Key Novelty:**
We propose to build a special compiler for optimizing quantum computing on large-scale chiplet quantum hardware. A new *highway* model is proposed to boost computation by enabling more concurrency in gate execution regardless of the distances among the involved qubits.

**Preliminary Results:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Building blocks</th>
<th>Single chiplet</th>
<th>Connected chiplets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td><img src="image" alt="Square Diagram" /></td>
<td><img src="image" alt="Square Diagram" /></td>
<td><img src="image" alt="Square Diagram" /></td>
</tr>
<tr>
<td>Hexagon</td>
<td><img src="image" alt="Hexagon Diagram" /></td>
<td><img src="image" alt="Hexagon Diagram" /></td>
<td><img src="image" alt="Hexagon Diagram" /></td>
</tr>
<tr>
<td>Heavy Square</td>
<td><img src="image" alt="Heavy Square Diagram" /></td>
<td><img src="image" alt="Heavy Square Diagram" /></td>
<td><img src="image" alt="Heavy Square Diagram" /></td>
</tr>
<tr>
<td>Heavy Hexagon</td>
<td><img src="image" alt="Heavy Hexagon Diagram" /></td>
<td><img src="image" alt="Heavy Hexagon Diagram" /></td>
<td><img src="image" alt="Heavy Hexagon Diagram" /></td>
</tr>
</tbody>
</table>

On average (geomean), the circuit depth is reduced by **69.05%**, and the effective number of CNOTs is reduced by **27.25%**.

**Example:** Computation process on chiplets at the presence of highway (dark blue paths in (c)). Commutable gates in circuit (a) are aggregated to multi-target control gates (b) and executed simultaneously on the highway (d).
AutoComm: A Framework for Enabling Efficient Communication in Distributed Quantum Programs

Anbang Wu, Hezi Zhang, Gushu Li, Alireza Shabani, Yuan Xie, Yufei Ding

Key Novelty:
First Compiler that enables optimization for burst communication.

Insights: Burst communications enable more remote gates and thus reduce the ERP pair consumption, but they are often hidden in the original quantum programs.

Table:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Variable Name</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-qubit gates</td>
<td>$t_{\text{1q}}$</td>
<td>$\sim 0.1$ CX</td>
</tr>
<tr>
<td>CX and CZ gates</td>
<td>$t_{\text{2q}}$</td>
<td>1 CX</td>
</tr>
<tr>
<td>Measure</td>
<td>$t_{\text{ms}}$</td>
<td>5 CX</td>
</tr>
<tr>
<td>EPR preparation</td>
<td>$t_{\text{EP}}$</td>
<td>$\sim 12$ CX</td>
</tr>
<tr>
<td>One-bit classical</td>
<td>$t_{\text{cb}}$</td>
<td>$\sim 1$ CX</td>
</tr>
</tbody>
</table>

Compared to state-of-the-art DQC compilers, experimental results show that our proposed AutoComm can reduce the communication resource consumption and the program latency by 72.9% and 69.2% on average, respectively.

Example: Our compiler would enable communication aggregation with three key modules. (a) Identifying potential burst communication. (b) Linear merge. (c) Iterative refinement.
QEC is a Software Layer

Vertical Integration and Co-design

Applications
- Quantum simulation, Shor’s algorithm, Quantum ML, VQE...

QEC Code
- Surface code, LDPC code, Stabilizer code, Subsystem code...

Quantum Hardware
- Sparse coupling graph → special but critical types errors (correlated error, losing qubits) → more general noise modeling

Horizontal Expanding and Unification
A Synthesis Framework for Stitching Surface Code with Superconducting Quantum Devices
Anbang Wu, Gushu Li, Hezi Zhang, Gian Giacomo Guerreschi, Yufei Ding, and Yuan Xie.

Key Novelty:
We can use a compiler to mitigate such a structural-level mismatch.

Software Infrastructure:
We can systematically solve the mismatch: 1) Good abstraction; 2) Knowledge of beneficial and legal transformations; 3) An efficient search scheme.

Architecture and Compiler Support for Fault-tolerant Quantum Computing based on Code Switching
Anbang Wu, Keyi Yin, Andrew Cross, Ang Li, and Yufei Ding

Key Novelty:
We take (Steane code + RM code) as an example, to show the power of full-stack integration/codesign, from QEC-hardware codesign to compiler-application codesign.

1. Mapping of two code on the same area to facilitate code conversion.
2. When to apply the conversion?

Examples of placing multiple logical qubits. Green arrows denote logical CX directions. (a) connectivity 4. (b) connectivity 4 rotated. (c) connectivity 6. (d) connectivity 8.
Thank you!