Closing the Gap
between Quantum Algorithms and Machines
with Hardware-Software Co-Design

Fred Chong
Seymour Goodman Professor
Department of Computer Science
University of Chicago

Lead PI, the EPiQC Project,
an NSF Expedition in Computing

With Ken Brown, Ike Chuang, Diana Franklin, Danielle Harlow, Aram Harrow, Margaret Martonosi, John Reppy, David Schuster, & Peter Shor
(UChicago, MIT, Princeton, Duke, UCSB)
Why Quantum Computing?

- Fundamentally change what is computable
  - The only means to potentially scale computation exponentially with the number of devices
- Solve currently intractable problems in chemistry, simulation, and optimization
  - Could lead to new nanoscale materials, better photovoltaics, better nitrogen fixation, and more
- A new industry and scaling curve to accelerate key applications
  - Not a full replacement for Moore’s Law, but perhaps helps in key domains
- Lead to more insights in classical computing
  - Previous insights in chemistry, physics and cryptography
  - Challenge classical algorithms to compete with quantum algorithms
Now is a privileged time in the history of science and technology, as we are witnessing the opening of the NISQ era (where NISQ = noisy intermediate-scale quantum).
– John Preskill, Caltech
The Algorithms to Machines Gap

<table>
<thead>
<tr>
<th>Year</th>
<th>Grover's Algorithm (Database search)</th>
<th>Shor's Factoring Algorithm (Crypto)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2000</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>2005</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>2010</td>
<td>1000</td>
<td>10000</td>
</tr>
<tr>
<td>2015</td>
<td>10000</td>
<td>100000</td>
</tr>
<tr>
<td>2020</td>
<td>100000</td>
<td>1000000</td>
</tr>
<tr>
<td>2025</td>
<td>1000000</td>
<td>#Qubits Needed</td>
</tr>
</tbody>
</table>

#Qubits Needed vs #Qubits Buildable

Gap!
The Algorithms to Machines Gap

- Grover's Algorithm (Database search)
- Shor's Factoring Alg. (Crypto)
- Quantum Sim, Q Chem, QAOA

#Qubits Needed vs. #Qubits Buildable

Year


Gap!
The Algorithms to Machines Gap

![Graph showing the gap between the number of qubits needed and the number of qubits buildable over time, with Grover's Algorithm (Database search) and Shor's Factoring Algorithm (Crypto).]
Closing the Gap: Software-Enabled Vertical Integration and Co-Design

- Grover's Algorithm (Database search)
- Shor's Factoring Alg. (Crypto)
- Quantum Sim, Q Chem, QAOA

Year:

Gap!

Co-Design

Categories:
- Algorithms
- Prog Lang
- Compiler
- Architecture
- Modeling
- Devices
Goal

Develop co-designed algorithms, SW, and HW to close the gap between algorithms and devices by 100-1000X, accelerating QC by 10-20 years.

Co-Design

- Grover's Algorithm (Database search)
- Shor's Factoring Alg. (Crypto)
- Quantum Sim, Q Chem, QAOA

Gap!

Result: Crossover by 2023!

Develop co-designed architectures, SW, and HW to close the gap between algorithms and devices by 100-1000X, accelerating QC by 10-20 years.
Space-Time Product Limits

2-qubit Gate Error $\sim 10^{-3}$

Diagram:
- $2\times 1024$
- $32\times 32$
- $2024\times 1$
- Qubits
- Gates
Space-Time Product Limits

2-qubit Gate Error $\sim 10^{-5}$

- Qubits: 2024x1
- Gates: 128x1024
“Good” Quantum Applications

- Compact problem representation
  - Functions, small molecules, small graphs
- High complexity computation
- Compact solution
- Easily-verifiable solution
- Co-processing with classical supercomputers
- Can exploit a small number of quantum kernels
Quantum Compiler Optimizations

- Similar to circuit synthesis for classical ASICs
- Program inputs often known at compile time
- Manage errors and precision
- Scarce resources
  - Every qubit and gate is important
Tool Flow

Scaffold tools, 41K lines of code, open source
epiqc.cs.uchicago.edu

https://github.com/epiqc/ScaffCC
Increasing Parallelism

- Compiler Optimizations:
  - Loop unrolling, constant propagation, inlining, function cloning, DAG scheduling

[Heckey+ ASPLOS 2015]
Aggregated Instructions for Optimal Control

[Shi+ ASPLOS19]
QAOA Example
More Parallel

More Serial
Aggregation Summary

- Breaks ISA abstraction
- Up to 10X latency reduction, mean 6X
- Especially useful for swaps
- Plan to apply to IBM Q and trapped ion machines
Noise-Adaptive Compiler Mappings

- SMT solver optimization for qubit and link variations in physical machines [Murali+ ASPLOS19]
- IBM Q Experience machines
Example: Avoid Error-Prone Links

(a) Bernstein-Vazirani Intermediate Representation

(b) Naive mapping

(c) Optimized mapping
Noise-Adaptive Results

![Bar chart showing success rates for various benchmarks using different simulators: Qiskit, T-SMT*, and R-SMT* with \( \omega = 0.5 \). The benchmarks include BV4, BV6, BV8, HS2, HS4, HS6, Toffoli, Fredkin, Or, Peres, QFT, and Adder.]
Noise-Adaptive Summary

- Up to 18X better reliability (2.9X mean)
- SMT may not scale beyond 500 qubits
  - New work improves compile time by optimizing independent gate rather than total reliability
- New work targeting UMD trapped ion machine
  - Adapts to pairwise interactions
- Better than native IBM, Rigetti, and UMD SW
Quantum Memory Management

- Quantum computations use a lot of ancilla
  - Reversible arithmetic
    - NAND -> 3-in-3-out
  - Indirect measurements
  - Error correction
- Ancilla can be reused
  - After measurement
  - After uncomputation
- Tradeoff between qubit and gate usage
Qubit Savings vs Gate Overhead

Uncomputation efficiency

Gate overhead for uncomputing applications

Percent increase in number of gates
Space-Time Product

Space-time cost of applications with uncompute

Space-time cost (as percentage of original)

<table>
<thead>
<tr>
<th></th>
<th>BF</th>
<th>SHA-1</th>
<th>BWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>25</td>
<td>50</td>
<td>125</td>
</tr>
</tbody>
</table>
How do I know if my QC program is correct?

- Check implementation against a formal specification
- Check general quantum properties
  - No-cloning, entanglement, uncomputation
- Checks based on programmer assertions (quantum simulation)
- Heuristic bug-finding systems
  [Altadmri SIGCSE15]
- Can we check useful properties in polynomial time for programs with quantum supremacy?
Quantum Simulation

- **Relates to verification and quantum supremacy**

- Using distributed memory
  - 45 qubits [Haner+ SC17]

- Using tensor contraction:
  - Find ordering of contractions to most rapidly decrease graph complexity

- Approximate simulation
  - Match the accuracy of a real machine
Partial Simulation

- Polynomial simulation of Clifford + small number of T gates [Bravyi PRL16]
- Polynomial simulation of a specific instance given known input and output?
  - Allows simulation of test vectors
How can tools help the programmer?

- Visualize properties (e.g., entanglement)
- Visualize qubit usage and memory leaks
- Library building blocks and templates
- Program synthesis techniques
HHL (Linear Systems) Entanglement
Program Synthesis by Sketching

Counter-example guided synthesis (CEGS)


[Solar-Lezama ASPLOS 06]
Quantum Teleportation

- A family of protocols!

\[ U_1 = U_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 
0.8 & -\frac{1}{\sqrt{2}} & 0.6 & \frac{1}{\sqrt{2}} \\
-0.6 & \frac{1}{\sqrt{2}} & 0.8 & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & 0.8 & -\frac{1}{\sqrt{2}} & 0.6 \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0.6 & -0.8 
\end{pmatrix} \]

\[ U_3 = 4 \begin{pmatrix} 
0.07 & 0.49 \\
-0.49 & 0.07 
\end{pmatrix}^{-1}, \quad U_4 = 4 \begin{pmatrix} 
0.01 & 0.565 \\
-0.565 & 0.01 
\end{pmatrix}^{-1}, \quad U_5 = 4 \begin{pmatrix} 
0.424 & 0.07 \\
0.07 & -0.424 
\end{pmatrix}^{-1}, \quad U_6 = 4 \begin{pmatrix} 
0.49 & -0.07 \\
0.07 & 0.49 
\end{pmatrix}^{-1} \]
What are the right abstractions?

- Specification Languages
  - Coq, Hamiltonians
- Programming Languages
  - Scaffold, Quipper, Q#, Quil …
- Instruction-Set Architectures
  - OpenQASM
- Physical Control
  - OpenPulse
Specialization vs Abstraction

Gap?

Short-term SW

Long-term SW

qubits

100 1000 10000 100000
Tutorial: Grand Challenges and Research Tools for Quantum Computing

Link to Tutorial Videos on Youtube

An NSF Expedition in Computing

ENABLING PRACTICAL-SCALE QUANTUM COMPUTING

epiqc.cs.uchicago.edu
Summary

- QC is at a historic time
- Software and architecture can generate key insights and accelerate progress
- With the right models and abstractions, classical techniques can have significant impact
How do we map data?

- Static spectral and graph partitioners
- Map for clustering
  - Probably necessary to get to 1000 qubits
- Map for irregular physical constraints
  - Qubit couplings, hardware defects
- Granularity of mappings
- Interaction with qubit reuse

Spectral communities for 2-level Bravyi-Haah magic-state factory
How do we control noise?

- Repeated executions [Temme+ PRL17]
  - Powers of 2 gate times (measure nth-order noise)
  - Random gate errors (simulate noise)
- Machine learning [Mavadia+ NatureComm17]
  - Need to minimize measurements
- Machine learning + control algorithms?
Classical Control and Computation

- Temperature boundaries and interconnect constraints [Tannu+ Micro17]
  - Cryo-cmos: high power, but lower cost to cool 4k
  - Superconducting: expensive memory, low power, but expensive to cool to 10mk

- Real-time control: hard for GHz speeds
  - Adaptive algorithms, ML

- Error decoding
  - Fast, simple decoder in superconducting logic
  - Trade frequency of decoding for quality