Quantum Computing
Quantum Computing
Quantum Computing

Quantum

A research effort from Google AI that aims to build quantum processors and develop novel quantum algorithms to dramatically accelerate computational tasks for machine learning.
Quantum Computing

A research effort from Google AI that aims to build quantum processors and develop novel quantum algorithms to dramatically accelerate computational tasks for machine learning.
Quantum Computing

Quantum

A research effort from Google AI that aims to build quantum processors and develop novel quantum algorithms to dramatically accelerate computational tasks for machine learning.

Empowering the quantum revolution

Your path to powerful, scalable quantum computing starts here.

Watch now

Join us at the leading edge of opportunity

Quantum computing takes a giant leap forward from today’s technology—one that will forever alter our economic, industrial, academic, and societal landscape. In just hours or days, a quantum computer can solve complex problems that would otherwise take billions of years for today’s computers to solve. This has massive implications for research in healthcare, energy, environmental systems, smart materials, and more. From hardware to software, from development through deployment, Microsoft brings the only scalable quantum system to the broadest set of customers.
Quantum Computing

Department of Energy Announces $218 Million for Quantum Information Science

SEPTEMBER 24, 2018

Field Will Shape Future of Information Processing

WASHINGTON, D.C. – Today, the U.S. Department of Energy (DOE) announced $218 million in funding for 85 research awards in the important emerging field of Quantum Information Science (QIS). The awards were made in conjunction with the White House Summit on Advancing American Leadership in QIS, highlighting...
Why?
Why?

• Factoring large numbers
Why?

- Factoring large numbers
- Simulating small physical systems
Why?

- Factoring large numbers
- Simulating small physical systems
- Uncrackable crypto systems
- Solving linear equations
Challenges

In general
- Answer may be unknown
- Simulation is intractable
- Breakpoints break things (opening the box kills the cat)

In the near term
- Execution is expensive, and (highly) error prone
- Computing resources (e.g., qubits) are scarce
PL to the Rescue?
PL to the Rescue?

- Compilers and optimizations, language design, formal methods, approximate computing, probabilistic programming, …

- Sound familiar?
PL to the Rescue?

- Compilers and optimizations, language design, formal methods, approximate computing, probabilistic programming, ...

- Sound familiar?

- Goal of this talk: Get you interested to work on QC with your PL hat on!
Qubits

*Superposition*: Qubits can be in multiple states (0 or 1) at once
Qubits

\((\alpha, \beta)\)

Superposition: Qubits can be in multiple states (0 or 1) at once
Qubits

\[(\alpha \beta)\]

\[|\alpha|^2 + |\beta|^2 = 1\]

Superposition: Qubits can be in multiple states (0 or 1) at once
Qubits

\[
\begin{pmatrix}
1 \\
0
\end{pmatrix}
\quad
\begin{pmatrix}
\alpha \\
\beta
\end{pmatrix}
\]

\[|0\rangle\]

\[|\alpha|^2 + |\beta|^2 = 1\]

**Superposition:** Qubits can be in multiple states (0 or 1) at once
Qubits

\[
\begin{pmatrix}
1 \\
0
\end{pmatrix}
\begin{pmatrix}
\alpha \\
\beta
\end{pmatrix}
\begin{pmatrix}
0 \\
1
\end{pmatrix}
\]

\[
|\alpha|^2 + |\beta|^2 = 1
\]

Superposition: Qubits can be in multiple states (0 or 1) at once
Measurement: Looking at a qubit probabilistically turns it into a bit.
Measurement

\[ (\alpha) \]

\[ (\beta) \]

_ Measurement_: Looking at a qubit probabilistically turns it into a bit.
**Measurement**

\[
\begin{pmatrix}
\alpha \\
\beta
\end{pmatrix}
\]

*Measurement*: Looking at a qubit probabilistically turns it into a bit.
Measurement: Looking at a qubit probabilistically turns it into a bit.
Measurement: Looking at a qubit probabilistically turns it into a bit.
**Measurement**

$$|\alpha|^2 \rightarrow \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \rightarrow |\beta|^2$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{dice} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

*Measurement*: Looking at a qubit probabilistically turns it into a bit.
Measurement

\[ \left| \frac{1}{\sqrt{2}} \right|^2 \quad \left( \begin{array}{c} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{array} \right) \quad \left| \frac{1}{\sqrt{2}} \right|^2 \]

\[ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \]
Measurement

\[
\frac{1}{2} \rightarrow \left( \begin{array}{c} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{array} \right) \rightarrow \frac{1}{2}
\]

\[
\left( \begin{array}{c} 1 \\ 0 \end{array} \right) \rightarrow \left( \begin{array}{c} 0 \\ 1 \end{array} \right)
\]
Unitaries are operators that transform ("evolve") quantum states
Unitaries are operators that transform ("evolve") quantum states.
Unitaries are operators that transform ("evolve") quantum states.
Unitaries are operators that transform (“evolve”) quantum states.
**Unitaries**

\[
\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}
\]

\[
\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}
\]

*Unitaries* are operators that transform ("evolve") quantum states.
Unitaries

\[ H |0\rangle = |+\rangle \]

\[ \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \]

This is the Hadamard unitary, labeled \( H \)
This is the *Hadamard* unitary, labeled $H$.
Multiple Qubits

Multi-qubit states can be formed via the tensor product

\[
\left( \frac{1}{\sqrt{2}} \right) \otimes \left( \begin{array}{c} 0 \\ 1 \end{array} \right)
\]
Multiple Qubits

\[
\left( \frac{1}{\sqrt{2}} \right) \otimes \left( \begin{array} {c} 0 \\ 1 \end{array} \right) = \left( \begin{array} {c} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{array} \right)
\]

Multi-qubit states can be formed via the tensor product.
Multiple Qubits

Multi-qubit states can be formed via the tensor product

\[ \left( \frac{1}{\sqrt{2}} \right) \otimes \left( \begin{array}{c} 0 \\ 1 \end{array} \right) = \left( \begin{array}{c} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{array} \right) \]

\[ |+\rangle \otimes |0\rangle = |+\rangle |0\rangle \]
Multiqubit Unitaries

\[
\begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
0 \\
\frac{1}{\sqrt{2}}
\end{pmatrix}
\]
Multiqubit Unitaries

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
0 \\
\frac{1}{\sqrt{2}}
\end{pmatrix}
\]
Multiqubit Unitaries

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$
Multiqubit Unitaries

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
0 \\
\frac{1}{\sqrt{2}}
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
1 \\
0
\end{pmatrix}
\]

CNOT \( |+\rangle |0\rangle = \frac{1}{\sqrt{2}} ( |01\rangle + |10\rangle) \)
Measurement 2.0

\[ \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \]
Measurement 2.0

\[
\begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} \\
0
\end{pmatrix}
\]
Measurement 2.0

\[
\begin{pmatrix}
0 \\ \\
\frac{1}{\sqrt{2}} \\ \\
\frac{1}{\sqrt{2}} \\ \\
0
\end{pmatrix}
\]
Measurement 2.0
Measurement 2.0

\[
\begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} \\
0
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} \\
0
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 \\
0 \\
0 \\
0
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 \\
0 \\
1 \\
0
\end{pmatrix}
\]
Measurement 2.0

\[
\begin{pmatrix}
0 \\
1 \\
0 \\
0
\end{pmatrix} \rightarrow \begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} \\
0
\end{pmatrix}
\rightarrow \begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}
\]
Measurement 2.0

\[ \frac{1}{\sqrt{2}} \]

\[ \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \]

\[ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \]

\[ \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \]

\[ \frac{1}{2} \]

\[ (1) \otimes (0) \]

\[ (1) \otimes (1) \]
Measurement 2.0

\[
\begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} \\
0
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
1 \\
0
\end{pmatrix} \otimes \begin{pmatrix}
0 \\
1
\end{pmatrix} \Rightarrow
\begin{pmatrix}
0 \\
1
\end{pmatrix} \otimes \begin{pmatrix}
1 \\
0
\end{pmatrix}
\]
Measurement 2.0

\[ \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \]

\[ \frac{1}{2} \left( \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) + \frac{1}{2} \left( \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \]

\[ |01\rangle \]

\[ |10\rangle \]
Entanglement

\(\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}\)
Entanglement

\[ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \]
Entanglement

\[
\begin{pmatrix}
0 \\
1 \\
0 \\
0
\end{pmatrix} \otimes \begin{pmatrix}
1 \\
0 \\
1 \\
0
\end{pmatrix} = \begin{pmatrix}
0 \\
\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} \\
0
\end{pmatrix}
\]
Entangled qubits are not probabilistically independent — they cannot be decomposed. Connection at a distance!
Circuits

(1 0) → H → (0 1) → meas (CNOT)
Algorithms

• Common theme of many quantum algorithms
  • Compute over the state space in superposition, essentially doing work “in parallel”
  • measure to produce the final result (and cancel out noise)
Algorithms

• Common theme of many quantum algorithms
  • Compute over the state space in superposition, essentially doing work “in parallel”
  • measure to produce the final result (and cancel out noise)

• Several challenges
  • Algorithms are probabilistic — must ensure measurement likely to get the right answer
Algorithms

- Common theme of many quantum algorithms
- Compute over the state space in superposition, essentially doing work “in parallel”
- Measure to produce the final result (and cancel out noise)

Several challenges

- Algorithms are probabilistic — must ensure measurement likely to get the right answer
- Gates in a quantum circuit must be unitary, implying (e.g.,) reversibility — “ancillae” bits used to encode answers
Recall: Challenges

In general
- Algorithms hard to write; answer may be unknown
- Simulation is intractable (states exponential in # qubits)
Recall: Challenges

In general

- Algorithms hard to write; answer may be unknown
- Simulation is intractable (states exponential in # qubits)
- Breakpoints break things (opening the box kills the cat)

Noisy, Intermediate Scale Quantum (NISQ) Computing era

—Preskill
Recall: Challenges

In general

• Algorithms hard to write; answer may be unknown
• Simulation is intractable (states exponential in # qubits)
• Breakpoints break things (opening the box kills the cat)

In the near term

_Noisy, Intermediate Scale Quantum (NISQ) Computing era_

—Preskill
Recall: Challenges

In general

- Algorithms hard to write; answer may be unknown
- Simulation is intractable (states exponential in # qubits)
- Breakpoints break things (opening the box kills the cat)

In the near term

- Execution is expensive, and (highly) error prone

*Noisy, Intermediate Scale Quantum (NISQ) Computing era*

—Preskill
Recall: Challenges

In general

• Algorithms hard to write; answer may be unknown
• Simulation is intractable (states exponential in # qubits)
• Breakpoints break things (opening the box kills the cat)

In the near term

• Execution is expensive, and (highly) error prone
• Computing resources (e.g., qubits) are scarce

Noisy, Intermediate Scale Quantum (NISQ) Computing era

—Preskill
Limited Qubits, Connectivity

ibmq_16_melbourne v1.0.0

Queue: 1 runs

Accounts:
Hub: ibm-q
Group: open
Project: main

Single-qubit error rate
0: 1.288e-2
1: 2.305e-2

CNOT error rate
0: 3.845e-2
1: 1.525e-1

Qubits: 14
Online since: 2018-11-06
Basis gates: u1, u2, u3, cx, id

https://quantumexperience.ng.bluemix.net/qx/devices
Limited Qubits, Connectivity

ibmq_16_melbourne v1.0.0

Queue: 1 runs

ibm-q/open/main

Accounts:

Hub: ibm-q
Group: open
Project: main

Single-qubit error rate
0: 1.288e-2
1: 2.305e-2

CNOT error rate
0: 3.845e-2
1: 1.525e-1

Qubits: 14
Online since: 2018-11-06
Basis gates: u1, u2, u3, cx, id

https://quantumexperience.ng.bluemix.net/qx/devices
Limited Qubits, Connectivity

ibmq_16_melbourne v1.0.0

Single-qubit error rate
- Qubits: 14
- Online since: 2018-11-06
- Basis gates: u1, u2, u3, cx, id

CNOT error rate
- 1.288e-2
- 2.305e-2
- 3.845e-2
- 1.525e-1
Limited Qubits, Connectivity

ibmq_16_melbourne v1.0.0

Queue: 1 runs

ibm-q/open/main

Accounts:

Hub: ibm-q
Group: open
Project: main

Single-qubit error rate
1.288e-2
2.305e-2

CNOT error rate
3.845e-2
1.525e-1

Qubits: 14
Online since: 2018-11-06
Basis gates: u1, u2, u3, cx, id

https://quantumexperience.ng.bluemix.net/qx/devices
Limited Qubits, Connectivity

ibmq_16_melbourne v1.0.0

Queue: 1 runs

Accounts:
Hub: ibm-q
Group: open
Project: main

Single-qubit error rate
- 0: 1.288e-2
- 1: 2.305e-2

CNOT error rate
- 2: 3.845e-2
- 3: 1.525e-1

Qubits: 14
Online since: 2018-11-06
Basis gates: u1, u2, u3, cx, id

https://quantumexperience.ng.bluemix.net/qx/devices
Substantial Error Rates

ibmq_16_melbourne v1.0.0

Online
Queue: 1 runs

ibm-q/open/main

Accounts:
Hub: ibm-q
Group: open
Project: main

Single-qubit error rate
- Qubit 0: 1.288e-2
- Qubit 1: 2.305e-2

CNOT error rate
- CNOT: 3.845e-2
- CNOT: 1.525e-1

Qubits: 14
Online since: 2018-11-06
Basis gates: u1, u2, u3, cx, id

https://quantumexperience.ng.bluemix.net/qx/devices
Substantial Error Rates

ibmq_16_melbourne v1.0.0

Accounts:
Hub: ibm-q
Group: open
Project: main

Online
Queue: 1 runs

Single-qubit error rate
1.288e-2
2.305e-2

CNOT error rate
3.845e-2
1.525e-1

Qubits
14
Online since
2018-11-06
Basis gates
u1, u2, u3, cx, id

https://quantumexperience.ng.bluemix.net/qx/devices
Substantial Error Rates

ibmq_16_melbourne v1.0.0

- Online
- Queue: 1 runs
- Accounts:
  - Hub: ibm-q
  - Group: open
  - Project: main

- Single-qubit error rate:
  - 0: 1.288e-2
  - 1: 2.305e-2

- CNOT error rate:
  - 0: 3.845e-2
  - 1: 1.525e-1

- Qubits: 14
- Online since: 2018-11-06
- Basis gates: u1, u2, u3, cx, id

https://quantumexperience.ng.bluemix.net/qx/devices
Addressing the Challenges

In general
Addressing the Challenges

In general

- **Formal verification**: State goal, prove you reach it
  - “Beware of bugs in the above code; I have only proved it correct, not tried it.” —Knuth
Addressing the Challenges

In general

- **Formal verification**: State goal, prove you reach it
  - “Beware of bugs in the above code; I have only proved it correct, not tried it.” —Knuth
- **Language design**: Make algorithms more intuitive
Addressing the Challenges

In general

• **Formal verification**: State goal, prove you reach it
  • “Beware of bugs in the above code; I have only proved it correct, not tried it.” —Knuth

• **Language design**: Make algorithms more intuitive

In the near term (handling scarce, error-prone resources)

• **Compiler optimizations** (proved correct)
Addressing the Challenges

In general

- **Formal verification**: State goal, prove you reach it
  - “Beware of bugs in the above code; I have only proved it correct, not tried it.” —Knuth
- **Language design**: Make algorithms more intuitive

In the near term (handling scarce, error-prone resources)

- **Compiler optimizations** (proved correct)
- **Reason about errors**, at the level of program and compiler
Our Research

- Verification Abstractions?
- Compilation Error Models Verification Optimization
- Semantics Mapping
- QWIRE, Robustness logic
- SQIRE
- (OpenQASM)
- (IBM QX, Rigetti, IonQ)

Hardware Description, Error Model
QWIRE

- QWIRE (POPL 2017) is a small language for describing quantum circuits. Features
  - A linear type system for enforcing no cloning
  - A denotational semantics in terms of density matrices
  - Embedded in Coq, with tactics for program verification

https://github.com/inQWIRE/QWIRE
Deutsch’s Algorithm

\[ |0\rangle \xrightarrow{H} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \xrightarrow{f} \frac{|0\rangle + (-1)^{f(0)}|1\rangle}{\sqrt{2}} \xrightarrow{H} \frac{|0\rangle + (-1)^{f(0)}|1\rangle}{\sqrt{2}} \xrightarrow{\text{meas}} \frac{|0\rangle + (-1)^{f(0)}|1\rangle}{\sqrt{2}} \]

**Definition deutsch (f : bool -> bool) :=**

```ocaml
let x ← H $ init0 $ ();
let y ← H $ init1 $ ();
let (x,y) ← (U f) $ (x,y);
let () ← discard $ meas $ y;
meas $ H $ x.
```
QWIRE in action
QWIRE in action

Compilation

$((x \land y) \land z)$
QWIRE in action

Compilation

$(x \land y) \land z$

Random Number Generation

$$H |0\rangle$$

Compilation

Random Number Generation

Compilation

Random Number Generation

Compilation

Random Number Generation
QWIRE in action

Compilation

$$\left( x \land y \right) \land z$$

Random Number Generation

Teleportation
QWIRE in action

Compilation

$$(x \land y) \land z$$

Random Number Generation

Teleportation

But: Nuts and bolts of proving things is surprisingly difficult!
• *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates.

\[
(Q, \lambda) \vdash P \leq \epsilon.
\]

judgment bounds the “distance” $\epsilon$ of the actual state, due to errors, from the ideal.
Reasoning about Errors

• *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates. Limitations:
Reasoning about Errors

• *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates. Limitations:

  • These bounds only *increase*: No way to verify error correction.
Reasoning about Errors

• *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates. Limitations:

  • These bounds only *increase*: No way to verify error correction.

  • Not designed for general reasoning
Reasoning about Errors

- *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates. Limitations:
  - These bounds only *increase*: No way to verify error correction.
  - Not designed for general reasoning
  - Not tied to hardware
Our Research

Compilation
Verification
Optimization
Mapping

QWIRE, Robustness logic

SQIRE

OpenQASM

IBM QX, Rigetti, IonQ

Hardware Description,
Our Research

QWIRE, Robustness logic

SQIRE

OpenQASM

IBM QX, Rigetti, IonQ

Compilation
Verification
Optimization
Mapping
SQIRe

- Simple IR for certified quantum compiler
  - Past compilers have had bugs. In SQIRe, can prove optimizations are bug free
  - Likewise, prove that mapping to hardware with limits makes sense
- Can also prove algorithm-level properties. Use of global register, rather than HOAS, a key
  - But not sure if key in principle, or just our practice
- Goal: Error-aware optimizations
Horizon: Higher-level Languages

- Current “high level” languages not very high level.
  - Traditional control operators over quantum circuits
- Instead: we want higher-level building blocks
  - Expose key components of quantum algorithms
- Q# is pushing in this direction, but still much to do
Future Work: You!

- There is lots of room for work in quantum computing that takes a PL perspective
  - Formal methods, compilers & optimization, language design, static analysis, hardware/software co-design …
- Join us!