PL for Quantum Computation

Kartik Singhal

ks.cs.uchicago.edu

Oregon PL Summer School

Tue Jun 28, 2022

EPiQC

THE UNIVERSITY OF CHICAGO

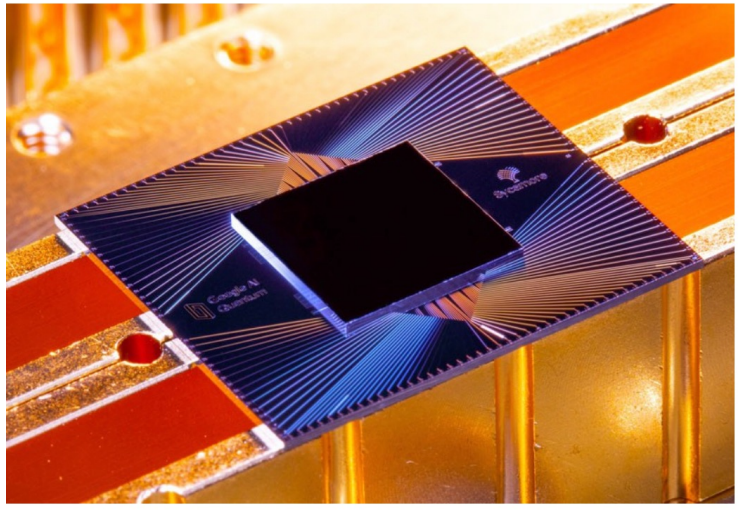
Quantum machines are here!

NEWS · 23 OCTOBER 2019

Hello quantum world! Google publishes landmark quantum supremacy claim

The company says that its quantum computer is the first to perform a calculation that would be practically impossible for a classical machine.

Elizabeth Gibney



The Sycamore chip is composed of 54 qubits, each made of superconducting loops. Credit: Erik Lucero

PDF version

RELATED ARTICLES

Beyond quantum supremacy



Quantum gold rush: the private funding pouring into quantum start-ups



Physicists propose football-pitch-sized quantum computer



REPORT

Quantum computational advantage using photons

Han-Sen Zhong[1,2,*], Hui Wang[1,2,*], Yu-Hao Deng[1,2,*], Ming-Cheng Chen[1,2,*], Li-Chao Peng[1,2], Yi…

+ See all authors and affiliations

Science 03 Dec 2020:
eabe8770
DOI: 10.1126/science.abe8770

Article | Figures & Data | Info & Metrics | eLetters | PDF

Abstract

Quantum computers promises to perform certain tasks that are believed to be intractable to classical computers. Boson sampling is such a task and is considered as a strong candidate to demonstrate the quantum computational advantage. We perform Gaussian boson sampling by sending 50 indistinguishable single-mode squeezed states into a 100-mode ultralow-loss interferometer with full connectivity and random matrix—the whole optical setup is phase-locked—and sampling the output using 100 high-efficiency single-photon detectors. The obtained samples are validated against plausible hypotheses exploiting thermal states, distinguishable photons, and uniform distribution. The photonic quantum computer generates up to 76 output photon clicks, which yields an output state-space dimension of 10^{30} and a sampling rate that is ~10^{14} faster than using the state-of-the-art simulation strategy and supercomputers.

The Extended Church-Turing Thesis is a foundational tenet in computer science, which states that a probabilistic Turing machine can efficiently simulate any process on a realistic physical device (*1*). In the 1980s, Richard Feynman observed that many-body quantum problems seemed difficult for classical computers due to the exponentially growing size of the quantum state Hilbert space. He proposed that a quantum computer would be a natural solution.

A long-term vision

“While a civil engineer as a professional provides specific guarantees about the efficacy and failure modes of a bridge, a software engineer for a software product, does not. Computing as a scientific discipline has come a long way, but as an engineering discipline, it has miles to go.”

We failed to achieve this goal with classical software over the last 70 years; quantum computing, as a fundamentally different way of computation, provides a unique opportunity to do things right in a principled manner.

Approaches to Software Reliability

- Social
  - Code reviews
  - Extreme/Pair programming
- Methodological
  - Design patterns
  - Test-driven development
  - Version control
  - Bug tracking
- Technological
  - "lint" tools, static analysis
  - Fuzzers, random testing
- Mathematical
  - Sound type systems
  - Formal verification



Less "formal": Lightweight, inexpensive techniques (that may miss problems)

This isn't a tradeoff… all of these methods should be used.

Even the most "formal" argument can still have holes:

- Did you prove the right thing?
- Do your assumptions match reality?
- Knuth: *"Beware of bugs in the above code; I have only proved it correct, not tried it."*

More "formal": eliminate *with certainty* as many problems as possible.

Slide from Benjamin Pierce's CIS500: Software Foundations, Lecture 1

The need for reliable quantum software

Reasoning techniques from classical computing do not transfer:

- State collapse on observation of state.
- Simulation beyond tens of qubits is expensive or impossible.
- The cost of running a program on a physical quantum computer is high; it gets worse if we run *incorrect* programs.
- Hence, it is important to ensure correctness statically.

Our approach:

- Adapt the best of classical static typing techniques to the quantum realm.
- Bring reasoning techniques as close to programming as possible.

Quantum Computation

The four postulates of quantum mechanics:

1. States

   Unit vectors in a complex vector space, $\mathbb{C}^n$
   aka, Hilbert space, $\mathcal{H}$

2. Dynamics

   Unitary matrices acting on state space

3. Measurement

   Getting information out of a quantum system

4. Combining systems

   State spaces *A* and *B* can be combined by taking their tensor product, $A \otimes B$

Computational basis

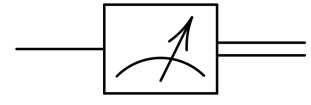
$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\psi = \alpha|0\rangle + \beta|1\rangle$$

$$|\alpha|^2 + |\beta|^2 = 1$$

Superposition

$$X.\psi = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.\psi = \alpha|1\rangle + \beta|0\rangle$$



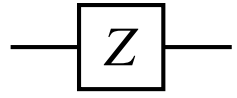
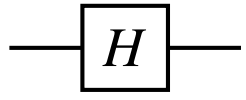
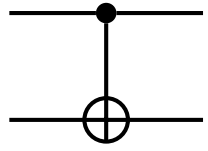
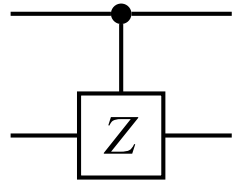
Primitive commands

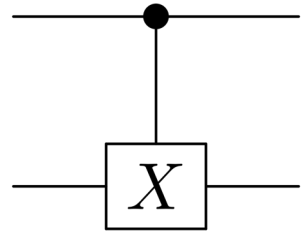
`init` *M*

`apply` *U* `to` *M*

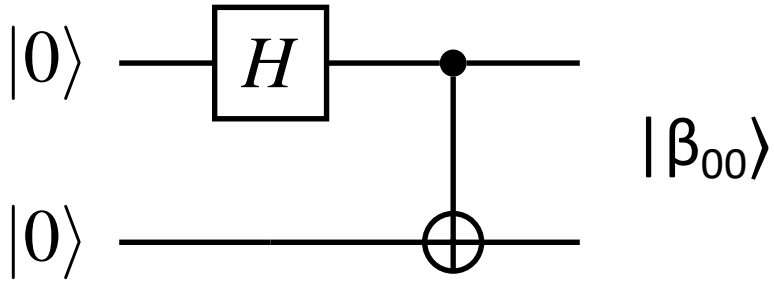
`meas` *M*

Commonly used unitary matrices and their gate notation

Z	Pauli Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
H	Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
CX	controlled-NOT		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
CZ	controlled-Z		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$



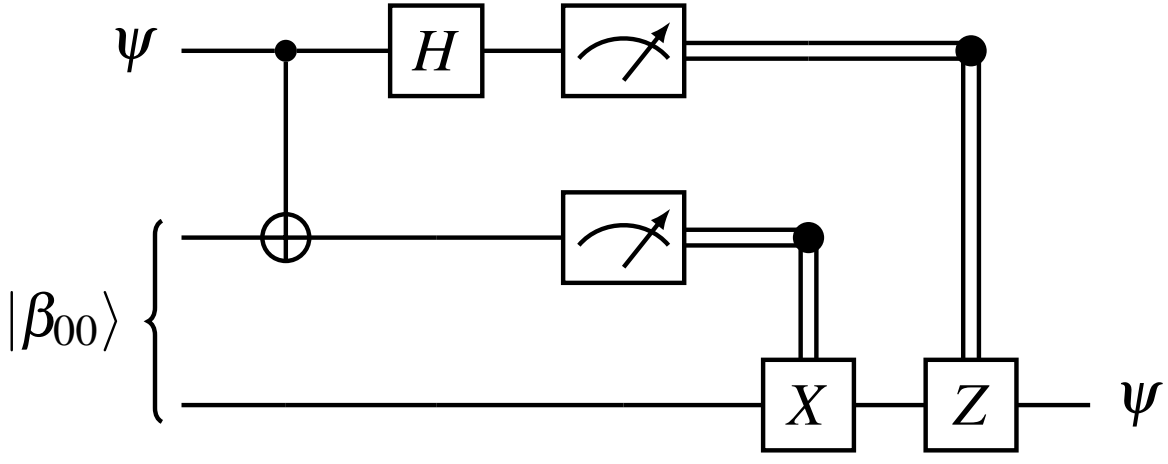
Bell state preparation




```
bell00 : unit → QST (a, b): (qbit⊗qbit)
bell00 = λx.do
           a ← init 0
           apply H to a
           b ← init 0
           apply CX to (a, b)
           return (a, b)
```

$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Quantum teleportation




```
teleport : q: qbit -> QST b: qbit
teleport = λx.do
             (a, b) ← bell00 ()
             apply CX to (q, a)
             apply H to q
             x ← meas q
             y ← meas a
             apply CX to (y, b)
             apply CZ to (x, b)
             return b
```

Publications

Search... Conference paper Date

Kartik Singhal, Kesha Hietala, Sarah Marshall, Robert Rand (2022). Q# as a Quantum Algorithmic Language. QPL 2022.

PDF Cite Code Project Venue

Robert Rand, Aarthi Sundaram, Kartik Singhal, Brad Lackey (2020). Gottesman Types for Quantum Programs. QPL 2020.

PDF Cite DOI Venue

Kartik Singhal, John Reppy (2020). Quantum Hoare Type Theory: Extended Abstract. QPL 2020.

PDF Cite Project Slides Video DOI Venue

Resources

- Quantum Programming Languages & Verification Bibliography: quantumpl.github.io OR git.io/qpl-bib

- PLanQC @ ICFP 2022: icfp22.sigplan.org/series/planqc

- QPL Proceedings: dblp.org/db/conf/qpl/index.html