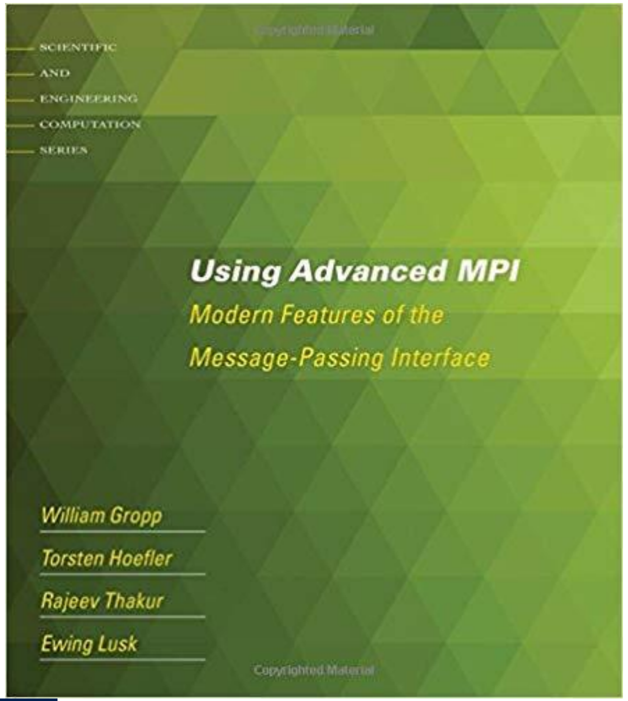
TORSTEN HOEFLER – ETH ZURICH & MICROSOFT QUANTUM

An HPC Systems Guy's View of Quantum Computing

Presented at TU Darmstadt, Germany, Jan. 2019

SPCL

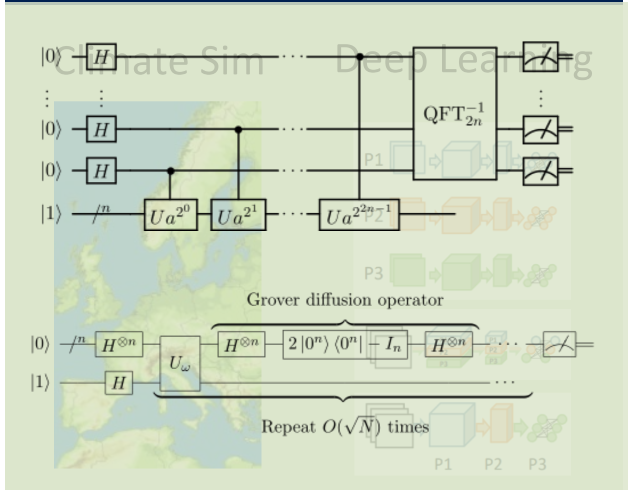
Who is this guy and what is he doing here?



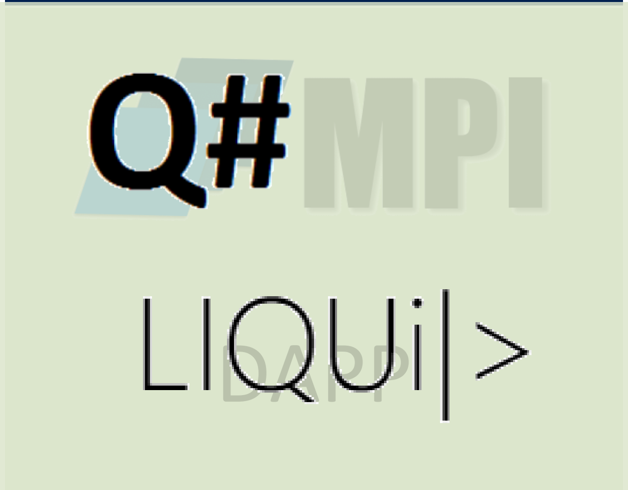
SPCL — 1 professor, 6 scientific staff, 13 PhD students

ETH zürich — 6.5k staff, 20k students, focus on research

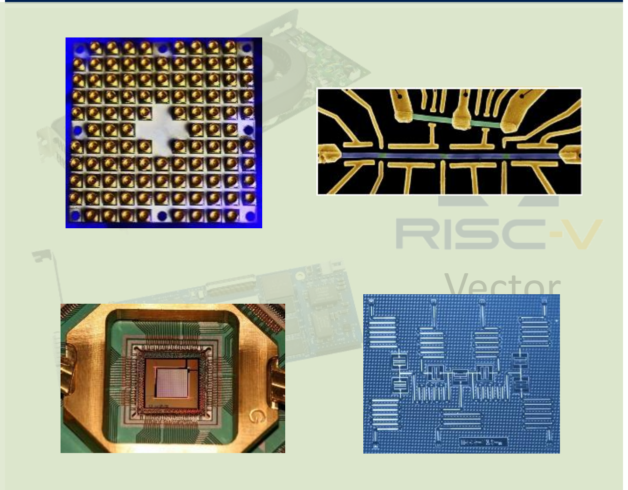
Applications

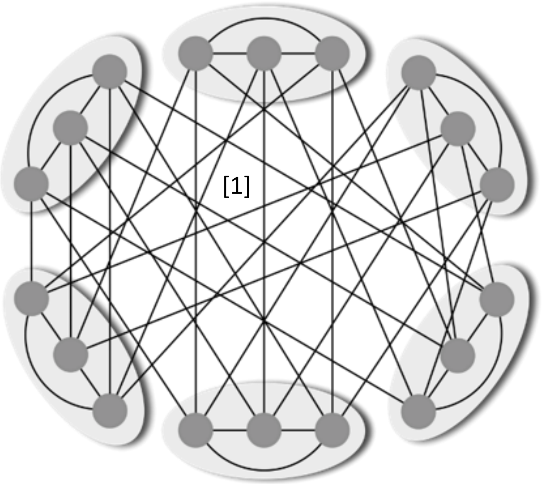


Programming Systems



Accelerator Hardware





[1] M. Besta, TH: Slim Fly: A Cost Effective Low-Diameter Network Topology, IEEE/ACM SC14, best student paper

Why I care – or – the end of computing as we know it!

Neuromorphic

- Asynchronous CMOS circuits
  - 1000x energy benefit
- Integrates compute (neurons) and memory/communication (synapses)
- Very specialized
  - Network and storage
  - Phrase your problem as inference!
- Even learning is hard
  - Comparatively little work
  - Suddenly much lower energy benefits …

image source: 21stcentury.com

Quantum

- Completely different paradigm
  - Concept of qubits
  - Bases on quantum mechanics (which only works in isolation)
- Many different ideas how to build
  - Ion trap (ions trapped in fields)
  - Optical
  - Spin-based
  - Superconducting
  - Majorana qubits
  - … (none proven to scale)
- Needs new algorithms to be useful
  - Algorithms are limited

image source: intel.com

Efficient CMOS

- FPGAs or CGRAs or GPUs
  - Have been around for a while
- Use transistors more efficiently
  - Accelerators
  - Custom architectures
  - Reconfigurable datapaths
- Adapt architecture to problem
  - Dataflow + Control Flow
- Cryogenic/superconducting ☺
- Main challenge
  - Programmability!
  - See our SC18 tutorial *“Productive Parallel Programming for FPGA”*

image source: intel.com

Using Hoare logic for quantum circuit optimization

Thomas Häner
Institute for Theoretical Physics
ETH Zurich
Zurich, Switzerland
haenert@phys.ethz.ch

Torsten Hoefler
Computer Science
ETH Zurich
Zurich, Switzerland
htor@inf.ethz.ch

Matthias Troyer
Institute for Theoretical Physics
ETH Zurich
Zurich, Switzerland
troyer@phys.ethz.ch

Abstract

By employing quantum mechanical phenomena such as superposition, entanglement, and interference, quantum computers promise to perform certain computations exponentially faster than any classical device. Precise control over these physical systems and proper shielding from unwanted interactions with the environment become more difficult as the space/time volume of the computation grows. Code optimization is thus crucial in order to reduce resource requirements to the greatest extent possible. Besides manual optimization, previous work has successfully adapted classical methods such as constant-folding and common subexpression elimination to the quantum domain. However, such classically-inspired methods fail to exploit certain optimization opportunities that arise due to entanglement. To address this insufficiency, we introduce an optimization methodology which employs Hoare triples in order to identify and exploit these optimization opportunities. We implement the optimizer using the Z3 Theorem Prover and the ProjectQ software framework for quantum computing and show that it is able to reduce the circuit area of our benchmarks by up to 5×.

1 Introduction

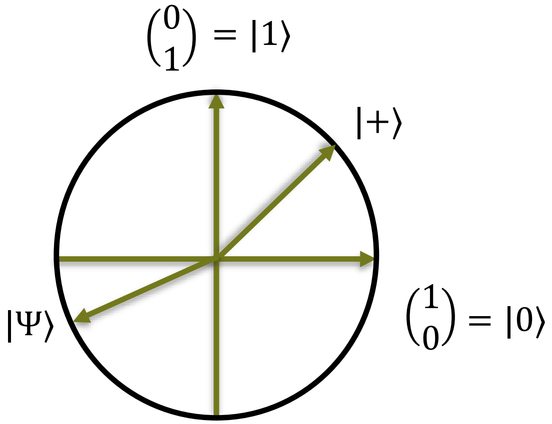
Quantum computers promise to solve certain computational tasks exponentially faster than classical computers. As a result, significant resources are being spent in order to make quantum computing become reality. In anticipation of the necessary layers of abstraction to facilitate software development, these packages include optimizing compilers. Inspired by previous work in the classical domain, these programs allow merging of quantum operations at various layers of abstraction [15, 34], e.g., merging of rotations that are applied successively to the same quantum bit (qubit), and using code annotations to identify patterns that are common in quantum computing, akin to pragma statements in classical computing [15, 34]. Further optimization opportunities can be created by employing a set of commutation relations [28] to reorder operations. In general, however, this approach incurs a cost that is exponential in the number of qubits that the reordered operations act upon. Furthermore, several methods have been developed for exact circuit synthesis with certain optimality guarantees [1, 10, 11, 23, 27]. However, these methods are not suitable for optimization of large quantum circuits.

Despite these efforts, most of the progress made in, e.g., the aforementioned quantum chemistry applications have been due to manual circuit optimization [17, 21] and the derivation and evaluation of superior error bounds [30]. This suggests that the capabilities of optimizing compilers may still be significantly improved.

To this end, one may first consider differences between manual and automatic optimization. For instance, in contrast to automatic methods, humans tend to harness additional information such as the circumstances under which a given subroutine is invoked. While compilers may not be able to infer the semantics of a given program and its subroutines

What is a qubit and how do I get one?

"I don't like it, and I'm sorry I ever had anything to do with it."
Schrödinger (about the probability interpretation of quantum mechanics)



$$|\Psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle = \alpha_0 \binom{1}{0} + \alpha_1 \binom{0}{1}$$

$$|\alpha_0|^2 + |\alpha_1|^2 = 1$$

For example: $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

A qubit can include a lot of information in α_0 and α_1 but can only sample one bit while losing all

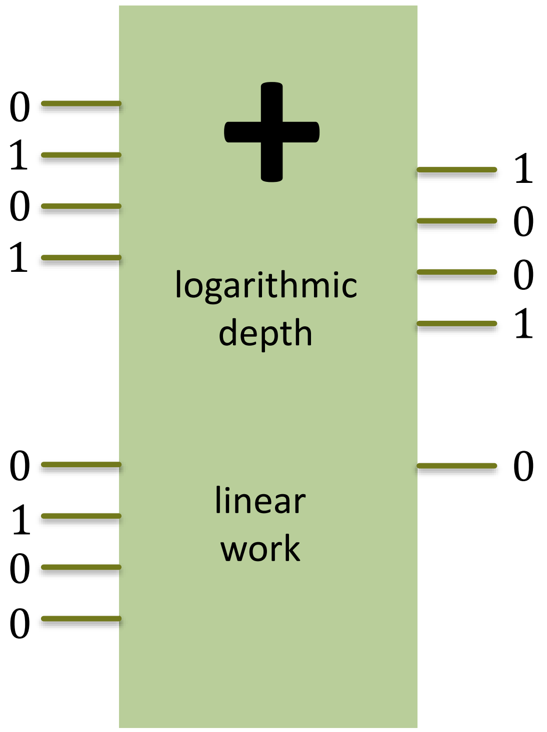
(encoding n bits takes $\Omega(n)$ operations)

n qubits live in a vector space of 2^n complex numbers (all combinations + entanglement)

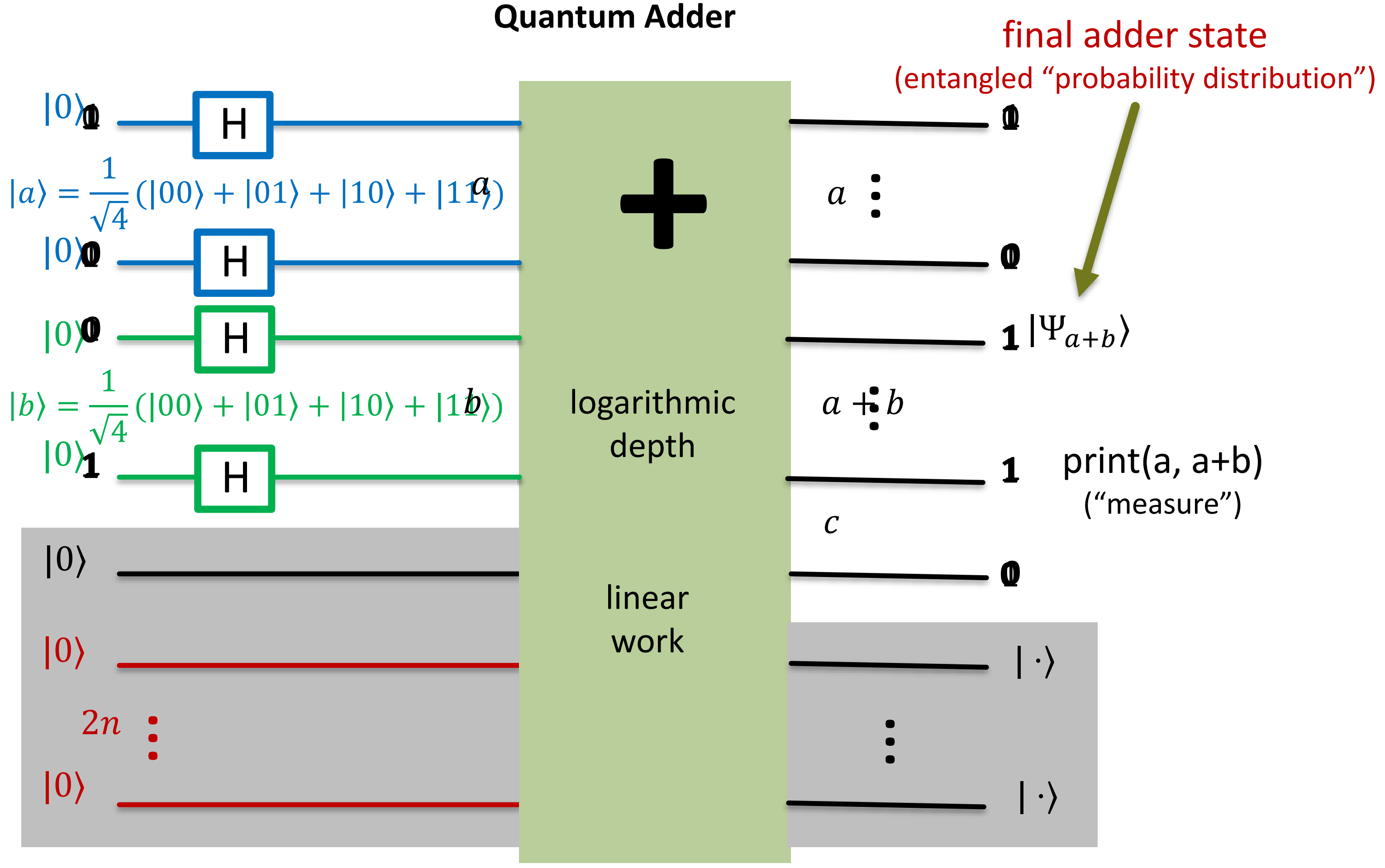
$$|\Psi_n\rangle = \sum_{i=0..2^n-1} \alpha_i |i\rangle \qquad \text{e.g., } |\Psi_2\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$$

Example: adding 2^n numbers in $O(\log n)$ time

Reminder: Classical Adder

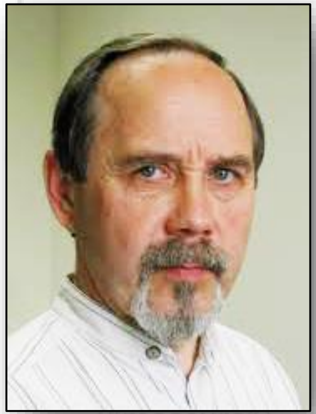


Quantum Adder



Example: adding 2^n numbers in $O(\log n)$ cycles

We add all 2^n numbers in parallel but only recover n classical bits!



A Corollary to Holevo’s Theorem (1973): ***at most n classical bits can be extracted from a quantum state with n qubits*** *even though that system requires $2^n - 1$ complex numbers to be represented!*

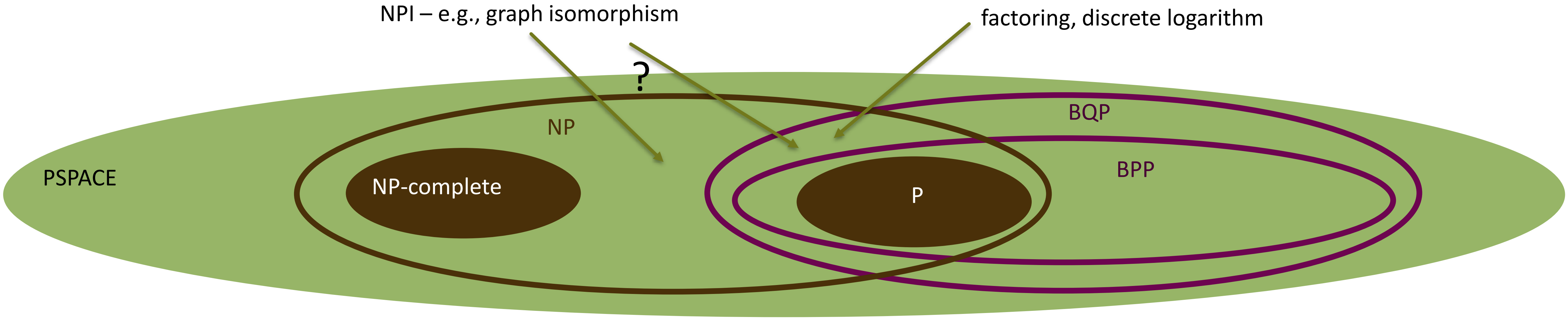
My corollary: *practical quantum algorithms read a linear-size input and modify an exponential-size quantum state such that the correct (polynomial size) output is likely to be measured.*

Question: Are quantum algorithms good at solving problems where a solution is verifiable efficiently (polynomial time)? Answer: Kind of ☺

So quantum computers can solve NP-complete problems!?

A problem is in NP if a solution can be verified deterministically in polynomial time.

- **Even with quantum computing, it seems that P ≠ NP (limited by linearity of operators). Quantum is at least as powerful as classic, thus, we do not know!**
- **New complexity class: Bounded-error Quantum Polynomial time (BQP)**
  - Quantum version of to **B**ounded-error **P**robabilistic **P**olynomial time (BPP)



Quantum algorithms are very complex (i.e., weird)

Most quantum programs recombine known algorithmic building blocks!

Amplitude Amplification

Amplify probability of the "right" output

- Using quantum interference
- E.g., Grover's search
- Often $O(\sqrt{2^n})$ iterations

Quantum Fourier Transform

DFT on amplitudes of a quantum state

- $O(n \log n)$ gates for 2^n elems
- Used in factoring and discrete logarithm

Phase Estimation

Measure eigenvalues of a unitary operator

- Used to compute eigenvectors
- Used to solve linear systems
- Determine eigenvalues in $O\left(\frac{1}{\epsilon}\right)$ gates

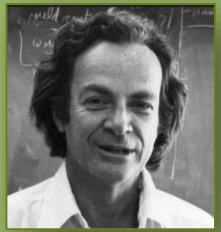
Quantum Walks

Speedup mixing times in randomized algorithms

- Quantum version of random walks
- Between quadratic and (rarely) exponential speedup

Hamiltonian Simulation

Simulate nature ☺



- Exponential speedup (over best known) classical algorithm for quantum effects in physics, chemistry, material science …. problems

Others

(not relevant for performance/HPC)

- Quantum teleportation
- EPR-pair based proofs/certificates
- Certified random number generation
- …

How does a quantum computer work?

Qubits are arranged on a (commonly 2D) substrate

Reuse big parts of process technology in microelectronics

Quantum systems are most naturally seen as accelerators

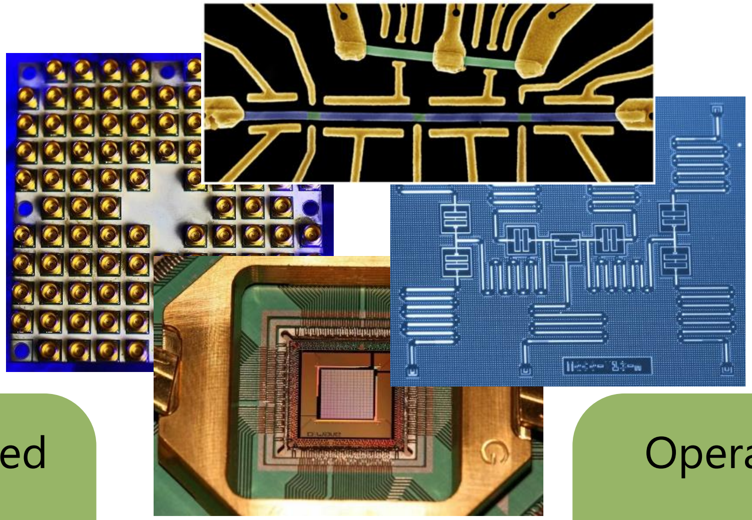
Work in close cooperation with a traditional control circuit

Quantum circuits use predication (no control flow)

Circuit view simplifies reasoning but requires classical envelope

Qubits are error prone, need to be highly isolated (major challenge)

Quantum error correction enabled the dream of quantum computers



Commonly limited to neighbor interactions between qubits

Limited range, may require swapping across chip

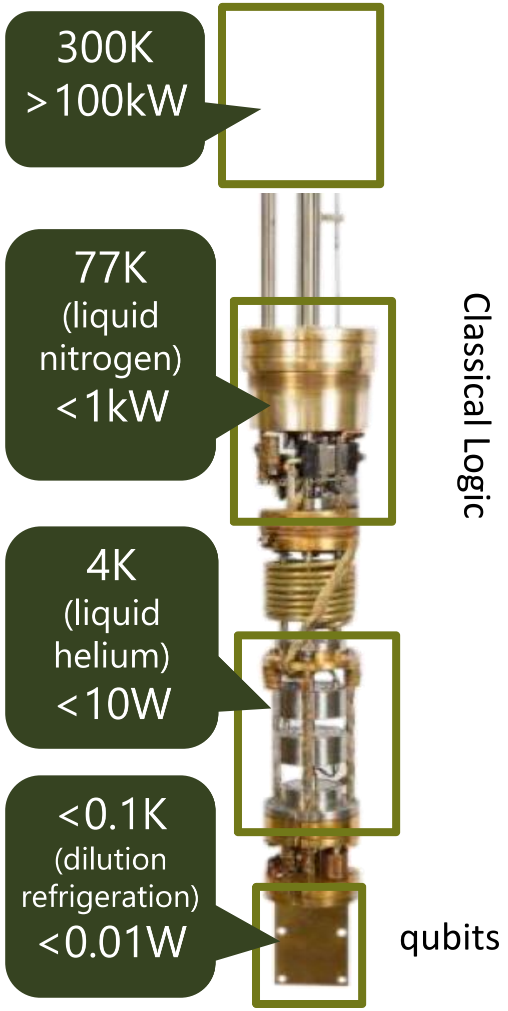
Operations ("gates") are applied to qubits in place!

As opposed to bits flowing through traditional computers!

Operations ("gates") have highly varying complexity

Some are literally free (classical tracking), some are very expensive

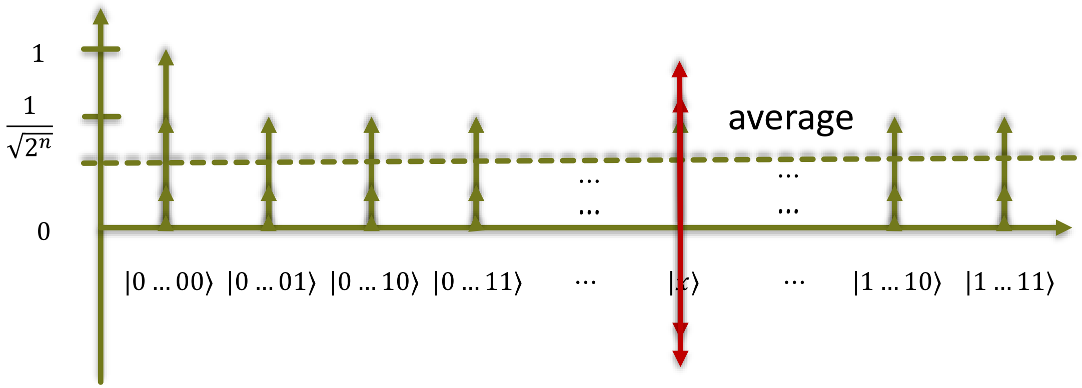
Hardware and software architecture for quantum computing



	qubits	bits	abstraction	
Logical layer	Quantum circuits	Instruction stream + data	Q# programming language Q intermediate representation	SW
Gate synthesis	Build Gates (T, Rotation, multi-control, …)	Factory control and qubit routing	Microcoded instructions	MW
Q error correction	QEC Codes (Steane etc., 1:n mapping)	Control for QEC (varies with code)	Microcoded instructions	MW
Physical control	Physical quantum state	Analog pulse generators	Qubit control pulses	HW

Full Example: Grover's search

- **Task: find $x \in D$ for which $f(x) = y$ (invert $f(x)$)**
  - Classical requires $O(|D|)$ queries
  - Quantum requires $O\left(\sqrt{|D|}\right)$ queries



allocate $\lceil \log_2 |D| \rceil$ qubits

Q# code

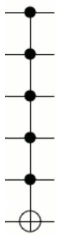
```
operation GroverSearch(n_searchQubits: Int): (Result[]) {
    body {
        mutable resultElement = new Result[n_searchQubits];

        using (qubits = Qubit[n_searchQubits]) {

            ApplyToEachCA(H, qubits); // qubits to uniform superposition

            let n_iterations = Floor(0.25 * PI()
                                  * Sqrt(ToDouble(2^n_searchQubits)));

            // Grover iteration
            for (nonce in 1..n_iterations) {
                OracleAND(qubits); // flips phase of desired state

                // apply Grover diffusion operator
                ApplyToEachCA(H, qubits);
                ApplyToEachCA(X, qubits);
                (Controlled Z)(qubits[1..n_searchQubits-1], qubits[0]);
                ApplyToEachCA(X, qubits);
                ApplyToEachCA(H, qubits);
            }
            set resultElement = MultiM(qubits);
        }
        return (resultElement);
    }
}
```

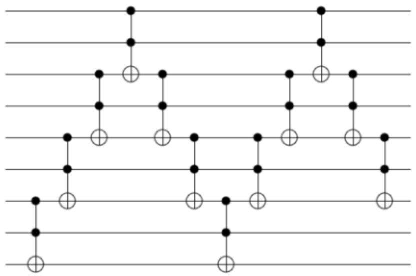
Quadratic speedup? Grover on a real machine

Performance estimates must be understood to be believed (inspired by Donald Knuth's "An algorithm must be seen to be believed")

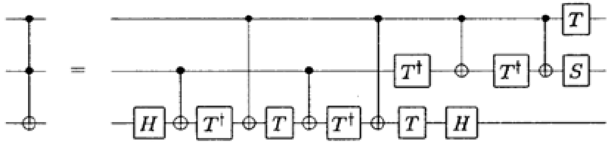
1. **Query complexity model – how algorithms are developed**
   - $T = \left\lfloor \frac{\pi}{4}\sqrt{2^n} \right\rfloor$ queries ($|D| = 2^n$ - represented by n bits)

2. **Express (oracle and diffusion operator) as n-bit unitary**
   - Assuming O n-bit operations for oracle!
   - $T = O\left\lfloor \frac{\pi}{4}\sqrt{2^n} \right\rfloor$ n-bit operations - $T_t = \left\lfloor \frac{\pi}{4}\sqrt{2^n} \right\rfloor$

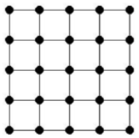




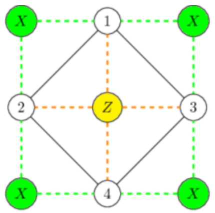
3. **Decompose unitary into two-bit (+arbitrary rotation) gates**
   - $T = O_2\left\lfloor \frac{\pi}{4}\sqrt{2^n} \right\rfloor \cdot 2(n-1)$ elementary operations - $T_t = \left\lfloor \frac{\pi}{4}\sqrt{2^n} \right\rfloor \cdot 4(n-1)$
4. **Design approximate implementations in discrete gate set (using HTHT...)**
   - $T = O_{\bar{2}}\left\lfloor \frac{\pi}{4}\sqrt{2^n} \right\rfloor \cdot 2(n-1)$ discrete T gate operations - $T_t = \left\lfloor \frac{\pi}{4}\sqrt{2^n} \right\rfloor \cdot 48(n-1)$



5. **Mapping to real hardware (swaps and teleport)**
   - Not to simple to model, depends on oracle – potentially $\Theta\left(\sqrt{2^n}\right)$ slowdown



6. **Quantum error correction**
   - Not so simple, depends on quality of physical bits and circuit depth, huge constant slowdown



Quadratic speedup? Grover on a real machine

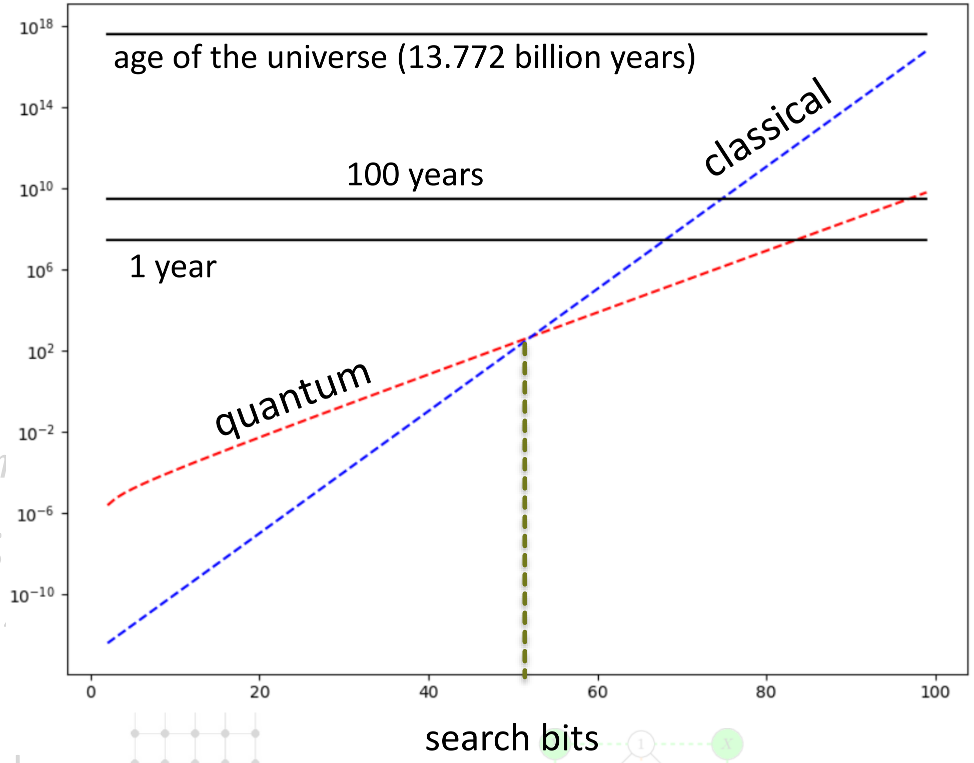
Performance estimates must be understood to be believed (inspired by Donald Knuth's "An algorithm must be seen to be believed")

Quantum computer with logical error rates $\leq 10^{-24}$ and gate times of 10^{-6}s vs. classical at 1 teraop/s.

38 billion years

	#gates		depth		#qubits
k	T	Clifford	T	overall	
128	$1.19 \cdot 2^{86}$	$1.55 \cdot 2^{86}$	$1.06 \cdot 2^{80}$	$1.16 \cdot 2^{81}$	2,953
192	$1.81 \cdot 2^{118}$	$1.17 \cdot 2^{119}$	$1.21 \cdot 2^{112}$	$1.33 \cdot 2^{113}$	4,449
256	$1.41 \cdot 2^{151}$	$1.83 \cdot 2^{151}$	$1.44 \cdot 2^{144}$	$1.57 \cdot 2^{145}$	6,681

Table 5. Quantum resource estimates for Grover's algorithm to attack AES-k, where $k \in \{128, 192, 256\}$.



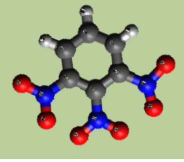
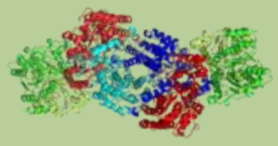
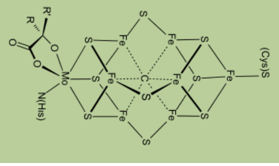
from Grassl et al.: "Applying Grover's algorithm to AES: quantum resource estimates", arXiv:1512.04965

Microsoft Quantum

Real applications?

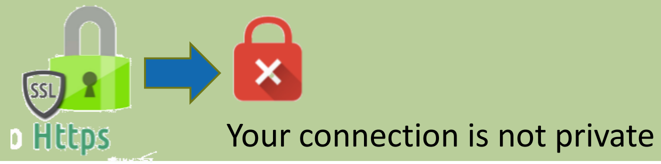
Quantum Chemistry/Physics

- Original idea by Feynman – use quantum effects to evaluate quantum effects
- Design catalysts, exotic materials, …

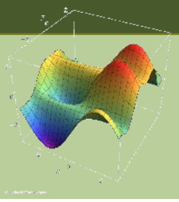
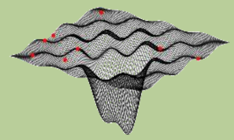
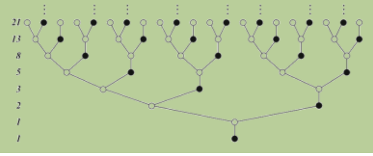
Breaking encryption & bitcoin

- Big hype – destructive impact – single-shot (but big) business case
- Not trivial (requires arithmetic) but possible



Accelerating heuristical solvers

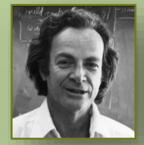
- Quadratic speedup can be very powerful!
- Requires much more detailed resource analysis → systems problem

Quantum machine learning

- Feynman may argue: "quantum advantage" assumes that circuits cannot be simulated classically → they represent very complex functions that could be of use in ML?


GOOGLE, ALIBABA SPAR OVER TIMELINE FOR 'QUANTUM SUPREMACY'



Thanks!

- Special thanks to Matthias Troyer and Doug Carmean

- Thanks to: Thomas Haener, Damian Steiger, Martin Roetteler, Nathan Wiebe, Mike Upton, Bettina Heim, Vadym Kliuchnikov, Jeongwan Haah, Dave Wecker, Krysta Svore

- And the whole MSFT Quantum / QuArC team!

All used images belong to the respective owners – source list in appendix