Practical Quantum Computing

Lecture 01

Introduction

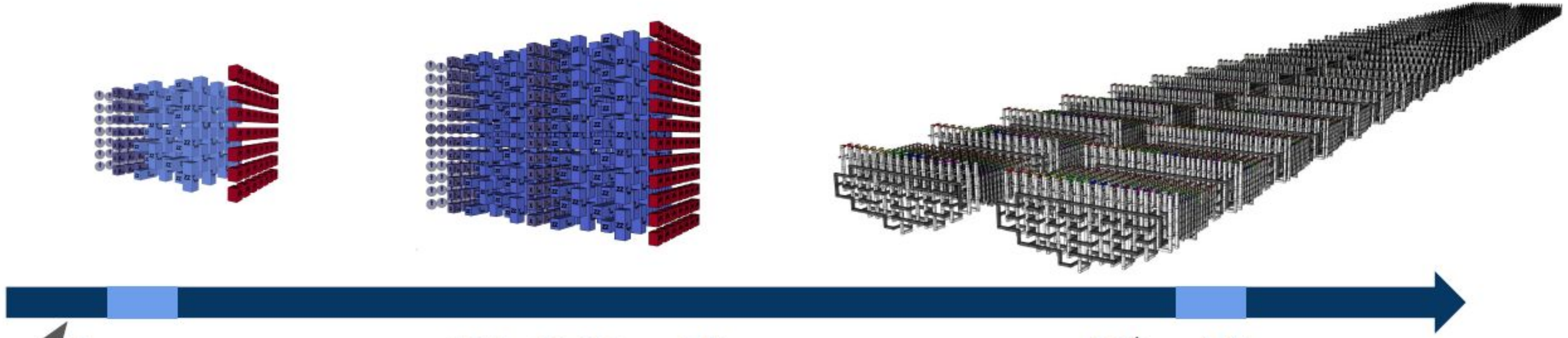
Bureaucracy

- See MyCourses page
  - Schedule
  - Grading
- Zulip chat - no private messages to TAs
  - Alex Ilov
  - Arshpreet Maan
- Python Programming
  - JupyterLab projects
  - PyCharm + git for Project3
-

Looking at history

Harvard Mark 1

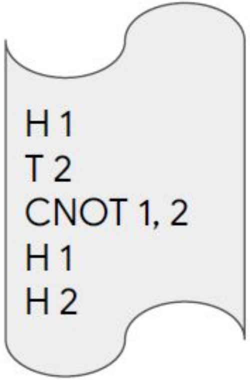
A Privileged Age



Are there impactful algorithms for noisy intermediate scale quantum (NISQ) processors?

Quantum Software

Fidelity: 90



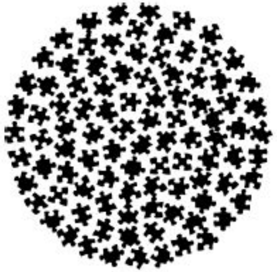
Write it on a piece of paper!

99

Useful to record Instructions.

Can still eyeball circuits.

99.9



At supremacy frontier.

Depth and gate minimization.

Simple modularity.

99.99

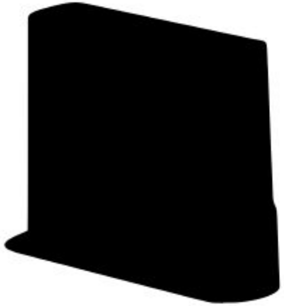
Complex modularity.

Automatic compiling.

Beginning of hardware independent abstractions.

...

99.9999999....

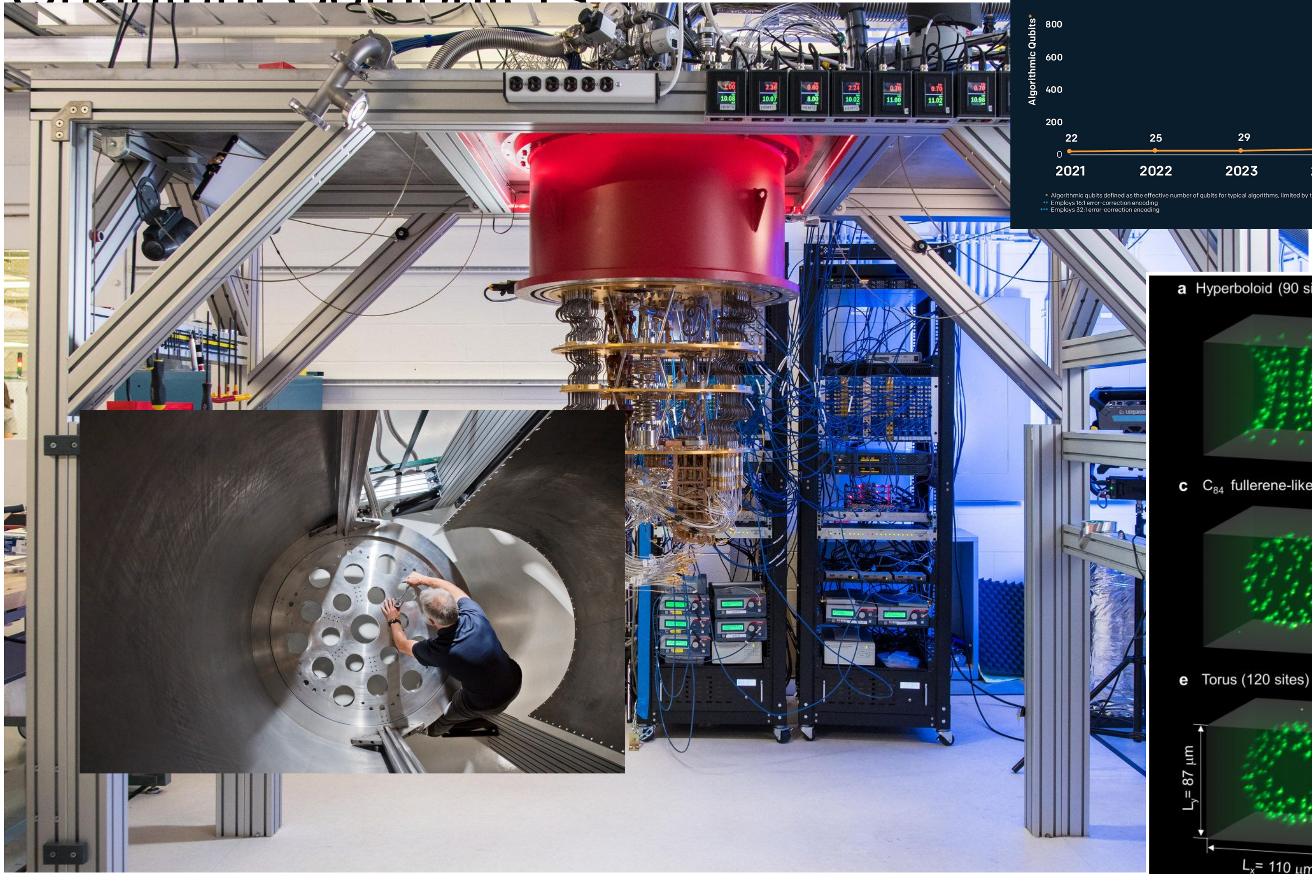


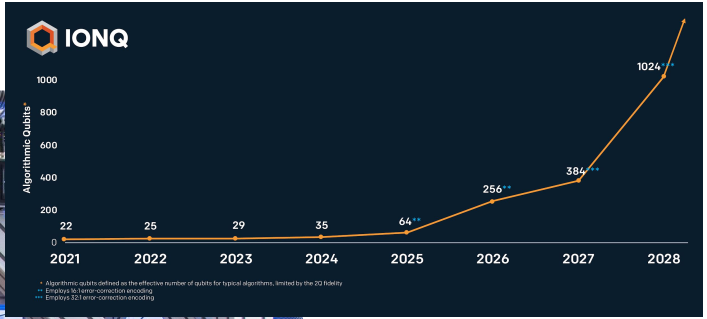
Architecture.

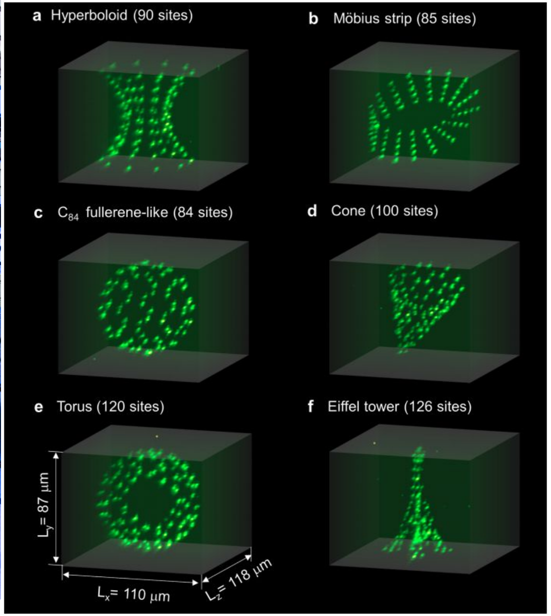
Operating systems.

High level languages.

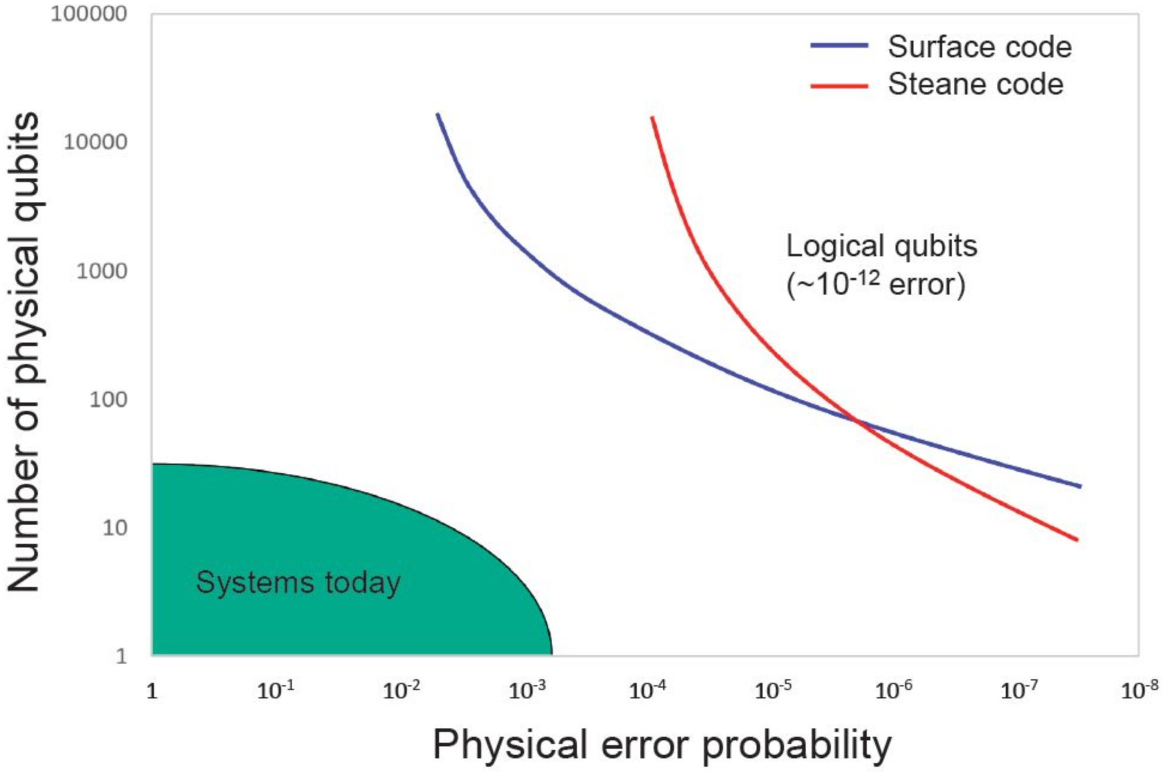
Quantum Computers







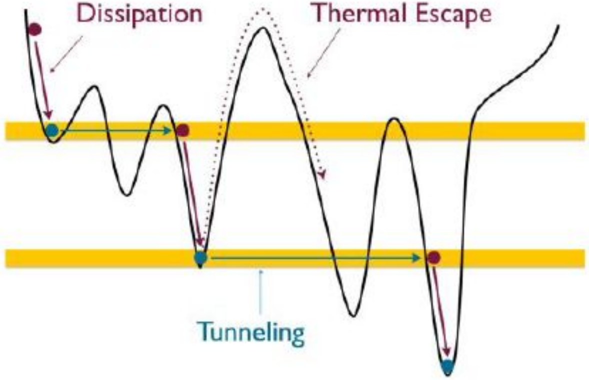
Fault-tolerance and the million of qubits



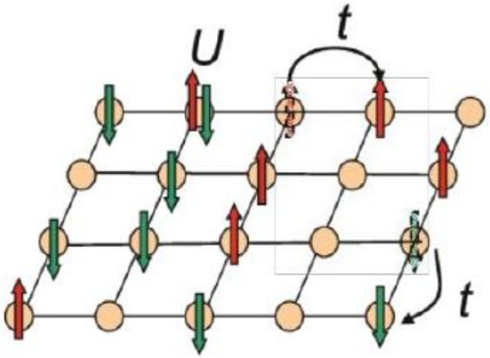
National Academies of Sciences, Engineering, and Medicine 2019. Quantum Computing: Progress and Prospects. Washington, DC: The National Academies Press.
https://doi.org/10.17226/25196.

Why NISQ?

Better Hardware



Better Algorithms



First Applications

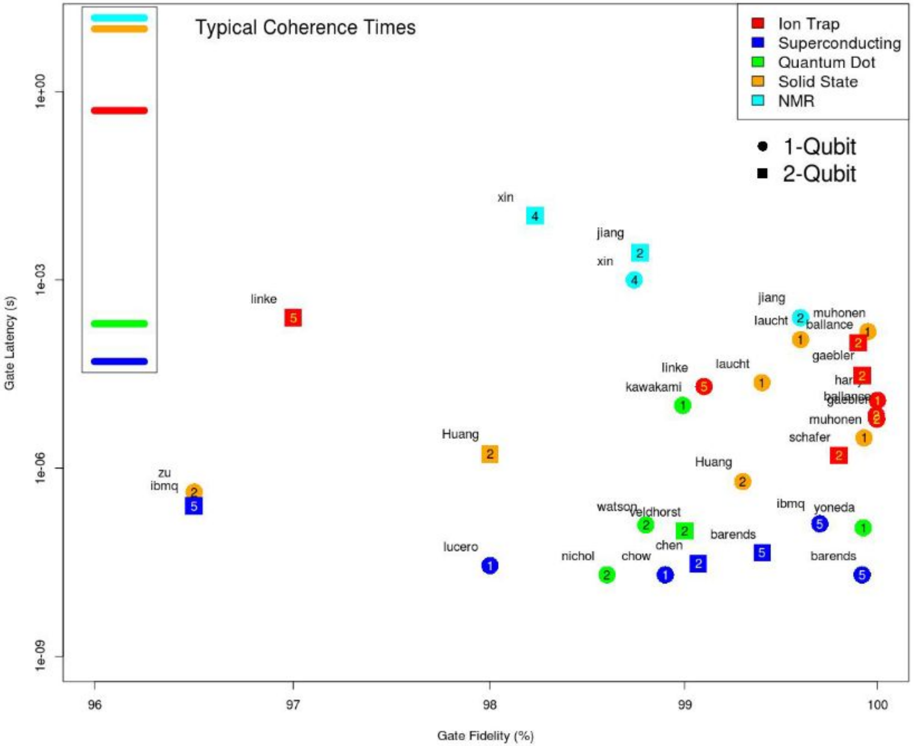


Fig. 1. Experimental latencies and fidelities of 1- and 2-qubit gates for different technologies. Approximate coherence times are shown in the inset for comparison. Higher fidelity and lower latency (relative to coherence time) are desirable. Numbers in points indicates total number of qubits in system. Technologies included are Ion Trap [192–196], Superconducting [80, 98, 204–208, 220, 220], Quantum Dot [198, 199, 201, 258], Solid State [221, 222, 259, 260], and NMR [31, 185]

Technology	Coherence Time (s)	1-Qubit Gate Latency (s)
Ion Trap	0.2 [192] - 0.5 [196]	1.6e-6 [193] - 2e-5 [196]
Superconductors	7.0e-6 [220] - 9.5e-5 [205]	2.0e-8 [80, 204, 207] - 1.30e-7 [98, 196]
Solid State Nuclear spin	0.6 [221]	1.12e-4 [222] - 1.5e-4 [221]
Solid State Electron spin	1e-3 [3]	3.0e-6 [221] - 2.3e-5 [222]
Quantum Dot	1e-6 [3, 225] - 4e-4 [200]	1e-9 [3] - 2e-8 [198]
NMR	16.7 [185]	2.5e-4 [185] - 1e-3 [31]

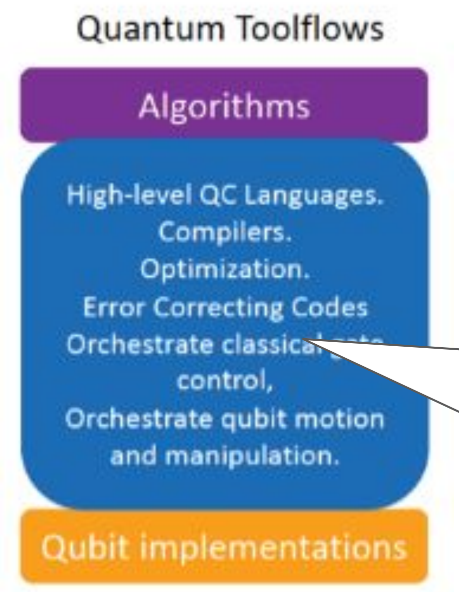
2-Qubit Gate Latency (s)	1-Qubit Gate Fidelity (%)	2-Qubit Gate Fidelity (%)	Mobile
5.4e-7 [193] - 2.5e-4 [196]	99.1 [196] - 99.9999 [195]	97 [196] - 99.9 [192]	YES
3.0e-8 [220] - 2.5e-7 [98, 196]	98 [206] - 99.92 [204]	96.5 [98, 196] - 99.4 [204]	NO
1.2e-4 [223]*	99.6 - [222] - 99.95 [221]	89 [224] - 96 [223]*	NO
1.2e-4 [223]*	99.4 [222] - 99.93 [221]	89 [224] - 96 [223]*	NO
1e-7 [201]	98.6 [198] - 99.9 [199]	90 [198]	NO
2.7e-3 [185] - 1.0e-2 [31]	98.74 [31] - 99.60 [185]	98.23 [31] - 98.77 [185]	NO

Table 1. Metrics for various quantum technologies.

* Nuclear/Electron Hybrid

Resch S, Karpuzcu UR. Quantum computing: An overview across the system stack. arXiv preprint arXiv:1905.07240. 2019 May 16.

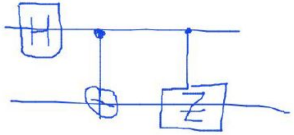
Some of the Quantum Software Philosophies



Cirq

Qiskit
Elements for building a quantum future

PENNYLANE



(ugly quantum circuit)

Open source Python frameworks for

Noisy Intermediate Scale Quantum (NISQ) algorithms

Some of the Quantum Software Philosophies

- Hardware details need to be part of programming abstractions as they greatly impact the viability of NISQ algorithms
- Hardware should drive features and diverse hardware will have diverse features
- Data structures and abstractions should match context in which they are used (**optimization, simulation, execution**)
- Optimize for workflows that validate heuristics algorithms and for rapid iteration in exploring minimally sized circuits.

Practical Quantum Computing - Quantum Software

Table 2: A Brief and Historical Summary of Quantum Programming Languages

Year	Language	Reference(s)	Semantics	Host Language	Paradigm
1996	Quantum Lambda Calculi	[181]	Denotational	lambda Calculus	Functional
1998	QCL	[206–209]		C	Imperative
2000	qGCL	[241, 312–314]	Operational	Pascal	Imperative
2003	λ_q	[282, 283]	Operational	Lambda Calculus	Functional
2003	Q language	[32, 33]		C++	Imperative
2004	QFC (QPL)	[245–247]	Denotational	Flowchart syntax (Textual syntax)	Functional
2005	QPAlg	[141, 160]		Process calculus	Other
2005	QML	[10, 11, 113]	Denotational	Syntax similar to Haskell	Functional
2004	CQP	[102–104]	Operational	Process calculus	Other
2005	cQPL	[180]	Denotational		Functional
2006	LanQ	[188–191]	Operational	C	Imperative
2008	NDQJava	[298]		Java	Imperative
2009	Cove	[227]		C#	Imperative
2011	QuECT	[48]		Java	Circuit
2012	Scaffold	[1, 138]		C (C++)	Imperative
2013	QuaFL	[162]		Haskell	Functional
2013	Quipper	[114, 115]	Operational	Haskell	Functional
2013	Chisel-Q	[175]		Scala	Imperative, functional
2014	LIQUi|⟩	[292]	Denotational	F#	Functional
2015	Proto-Quipper	[234, 237]		Haskell	Functional
2016	QASM	[212]		Assembly language	Imperative
2016	FJQuantum	[82]		Feather-weight Java	Imperative
2016	ProjectQ	[122, 266, 272]		Python	Imperative, functional
2016	pyQuil (Quil)	[259]		Python	Imperative
2017	Forest	[61, 259]		Python	Declarative
2017	OpenQASM	[66]		Assembly language	Imperative
2017	qPCF	[213, 215]		Lambda calculus	Functional
2017	QWIRE	[217]		Coq proof assistant	Circuit
2017	cQASM	[146]		Assembly language	Imperative
2017	Qiskit	[4, 232]		Python	Imperative, functional
2018	IQu	[214]		Idealized Algol	Imperative
2018	Strawberry Fields	[147, 148]		Python	Imperative, functional
2018	Blackbird	[147, 148]		Python	Imperative, functional
2018	QuantumOptics.jl	[157]		Julia	Imperative
2018	Cirq	[271]		Python	Imperative, functional
2018	Q#	[269]		C#	Imperative
2018	$Q\|SI\rangle$	[174]		.Net language	Imperative
2020	Silq	[35]		Python	Imperative, functional

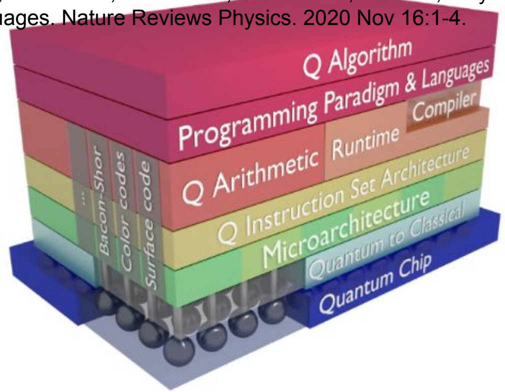
Zhao J. Quantum Software Engineering: Landscapes and Horizons. arXiv preprint arXiv:2007.07047. 2020 Jul 14.

Table 1 | **Overview of the languages surveyed in this Review**

Feature	Q#	Qiskit	Cirq	Quipper	Scaffold
Invocation	Standalone, usable from Python, C#, F#	Embedded into Python	Embedded into Python	Embedded into Haskell[a]	Standalone
Classical feedback	Yes	Yes[b]	No	Yes	Yes[c]
Adjoint generation	Yes	Yes	Yes	Yes	No
Resource estimation	Gate counts, number of qubits, depth and width, call graph profiling	Gate counts, number of qubits, depth and width	Gate counts, number of qubits	Gate counts, number of qubits, depth and width	Gate counts, number of qubits, depth[d]
Libraries	Standard, chemistry, numerics, ML	Standard, chemistry, optimization, finance, QCVV, ML	Standard, chemistry, ML	Standard, numerics	Standard[e]
Learning materials	Docs, tutorials, Katas	Docs, tutorials, textbook	Docs, tutorials	Docs[f], tutorials	Tutorials[g]

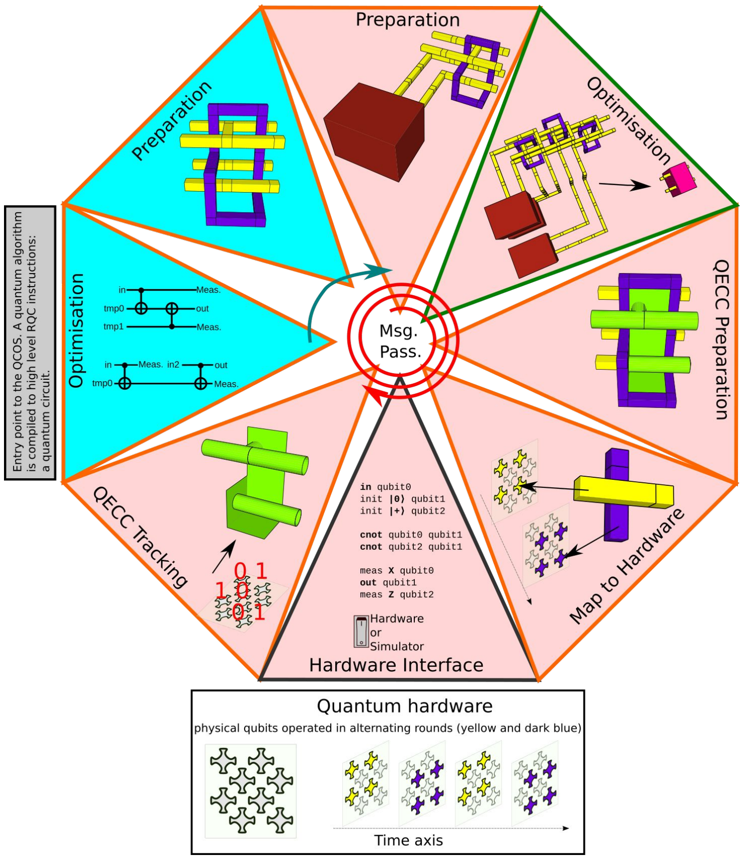
[a]Standalone versions such as Proto-Quipper-S and Proto-Quipper-M are proposed or under development. [b]Some restrictions apply regarding allowed types and language constructs in OpenQASM branching statements. [c]However, see relevant GitHub issue[122] regarding code generation for classical feedback. [d]Resources estimation includes different flavours of error correction (see REF.[123]). [e]See REF.[121] for the current selection of implemented algorithms. [f]Online API documentation available in REF.[124]. [g]Tutorials and manual in REFS[116,118]. ML, machine learning; QCVV, quantum characterization, verification and validation.

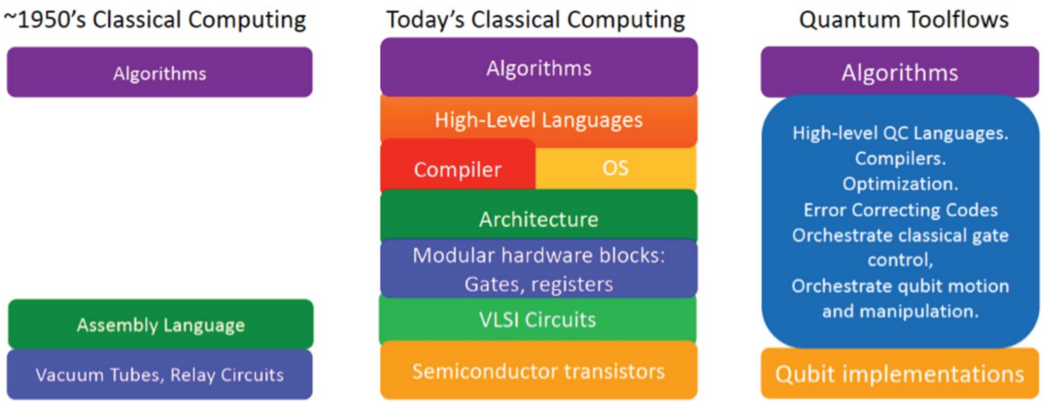
Heim B, Soeken M, Marshall S, Granade C, Roetteler M, Geller A, Troyer M, Svore K. Quantum programming languages. Nature Reviews Physics. 2020 Nov 16:1-4.



Varsamopoulos S, Bertels K, Almudever CG. Comparing neural network based decoders for the surface code. IEEE Transactions on Computers. 2019 Oct 23;69(2):300-11.

Aggregated architecture





From: https://cra.org/ccc/events/quantum-computing/

Grover's Algorithm

For N = 1000 entries

- classical exhaustive search method needs 1000 steps
- Grover's algorithm needs approx. 32 steps

Grover's algorithm is a framework

- No exponential speedup like Shor's alg.
- Extended for different problems
    - cryptanalysis AES
    - combinatorial optimisation
    - travelling salesman

Quantum Resource Estimates of **Grover's** Key Search on ARIA
AK Chauhan, SK Sanadhya - International Conference on Security, Privacy ..., 2020 - Springer
... [10] studied the quantum circuits of **AES** and estimated the cost of quantum resources needed to apply **Grover's** algorithm to the **AES** oracle for key search. Almazrooie et al ... As a working example, they implemented the **AES Grover** oracle in Q# quantum programming language ...
☆ 99 Related articles

Solving Binary $\mathcal{MQ}$ with **Grover's** Algorithm
P Schwabe, B Westerbaan - ... Conference on Security, Privacy, and Applied ..., 2016 - Springer
... primitives. For example, in [GLRS16], Grassl, Langenberg, Roetteler, and **Steinwandt** describe how to attack **AES**-128 with **Grover's** algorithm using a quantum computer with 2953 logical qubits in time about \(2^{87}\). We note ...
☆ 99 Cited by 25 Related articles All 12 versions

Quantum **Grover** Attack on the Simplified-**AES**
M Almazrooie, R Abdullah, A Samsudin... - Proceedings of the 2018 ..., 2018 - dl.acm.org
... This paper is organized as follows: Sections 2 and 3 review the Simplified-**AES** (S-**AES**) cryptosystem and the quantum **Grover's** algorithm, respectively ... Figure 8. Applying **Grover** attack on S-**AES**. Figure 8 illustrates the complete model of the **Grover** attack against S-**AES** ...
☆ 99 Related articles

Applying Grover's algorithm to AES: quantum resource estimates

Markus Grassl[1], Brandon Langenberg[2], Martin Roetteler[3], and Rainer Steinwandt[2]

[1] Universität Erlangen-Nürnberg & Max Planck Institute for the Science of Light, Günther-Scharowsky-Straße 1, Bau 24, 91058 Erlangen, Germany, Markus.Grassl@fau.de
[2] Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, U.S.A., {blangenb,rsteinwa}@fau.edu
[3] Microsoft Research, One Microsoft Way, Redmond, WA 98052, U.S.A., martinro@microsoft.com

Abstract. We present quantum circuits to implement an exhaustive key search for the Advanced Encryption Standard (AES) and analyze the quantum resources required to carry out such an attack. We consider the overall circuit size, the number of qubits, and the circuit depth as measures for the cost of the presented quantum algorithms. Throughout, we focus on Clifford+T gates as the underlying fault-tolerant logical quantum gate set. In particular, for all three variants of AES (key size 128, 192, and 256 bit) that are standardized in FIPS-PUB 197, we establish precise bounds for the number of qubits and the number of elementary logical quantum gates that are needed to implement Grover's quantum algorithm to extract the key from a small number of AES plaintext-ciphertext pairs.

Keywords: quantum cryptanalysis, quantum circuits, Grover's algorithm, Advanced Encryption Standard

Fault-Tolerance and its Cost

For N = 1000 entries

- Grover's algorithm needs approx. 32 steps
- How long does a step take?
  - Depends on speed of quantum computer gates
  - Fault-tolerance, reliability of the computer
- Qubit can be affected by noise (e.g. depolarising noise)

$$\rho \to (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z)$$

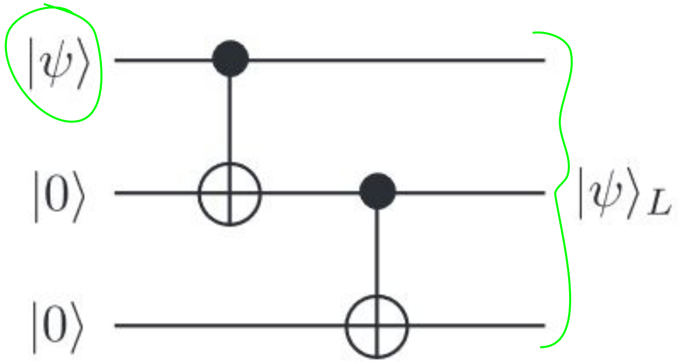
- Threshold theorem: *a quantum computer with noise can efficiently and accurately simulate an ideal quantum computer, if the level of noise is below a certain threshold*
  - Assuming threshold is not reached
  - Use methods to mitigate, detect, correct errors

Repetition and more complex codes

$|0\rangle_L = |000\rangle\,, \qquad |1\rangle_L = |111\rangle$

Introduce redundancy

$$|\text{GHZ}\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}.$$

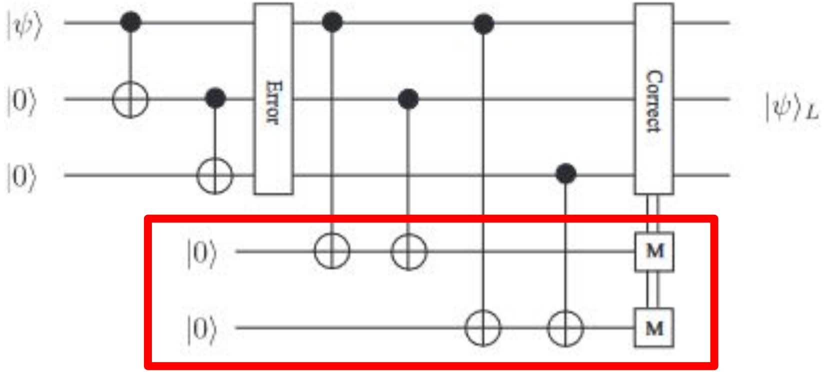


Circuit: Encoding a state in a logical state

$$\begin{aligned}\alpha|0\rangle + \beta|1\rangle \rightarrow\ & \alpha|0\rangle_L + \beta|1\rangle_L \\ &= \alpha|000\rangle + \beta|111\rangle \\ &= |\psi\rangle_L\,.\end{aligned}$$

[*] Devitt et. al, *Quantum Error Correction for Beginners*, available at https://arxiv.org/pdf/0905.2794.pdf

Syndromes, Correction, Flags



Ancillae used for syndrome measurement

Error Location	Final State, $\|data\rangle \|ancilla\rangle$
No Error	$\alpha\|000\rangle\|00\rangle + \beta\|111\rangle\|00\rangle$
Qubit 1	$\alpha\|100\rangle\|11\rangle + \beta\|011\rangle\|11\rangle$
Qubit 2	$\alpha\|010\rangle\|10\rangle + \beta\|101\rangle\|10\rangle$
Qubit 3	$\alpha\|001\rangle\|01\rangle + \beta\|110\rangle\|01\rangle$

- Syndrome measurements *have to be repeated*
- Repetition code protects only against a single type of error: detects two errors, corrects one

Digitization of noise is based on the observation that any interaction between a set of qubits and environment can be expressed in the form

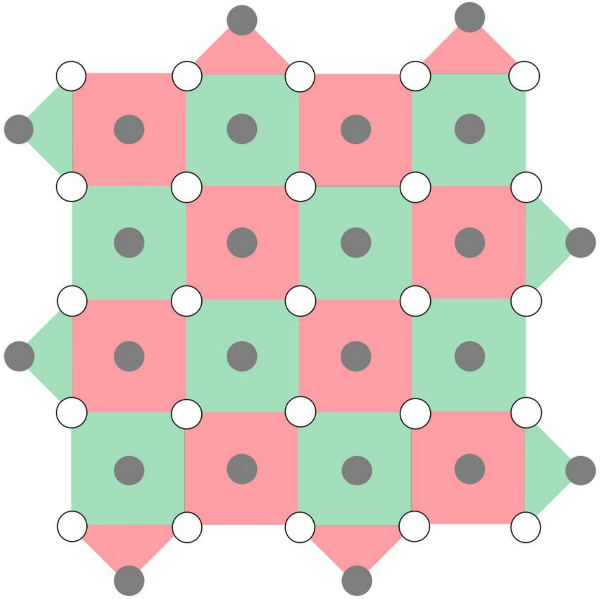
$$G = c_I\sigma_I + c_x\sigma_x + c_y\sigma_y + c_z\sigma_z$$

where,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Need to protect against *phase errors*, too

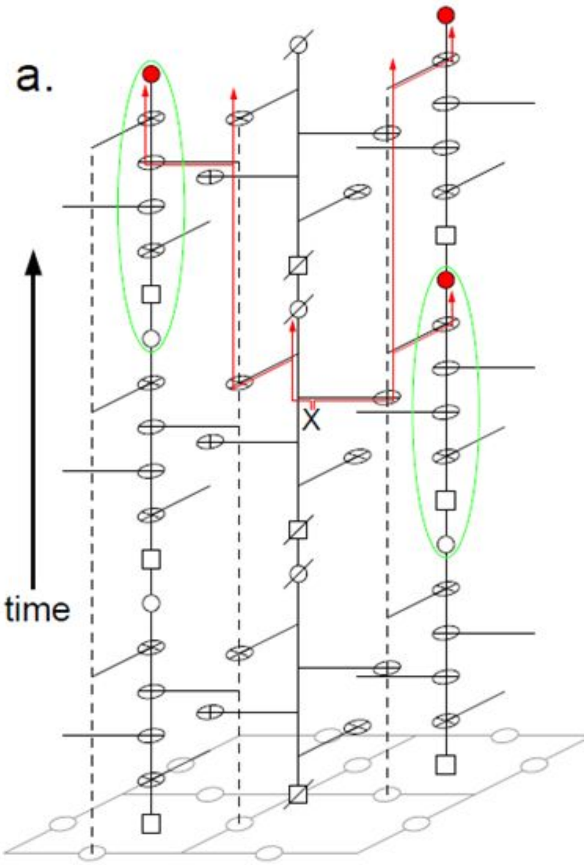
Surface code



detect X (red), detect Z (green)
measure Z stabilizers, measure X stabilizers

$$|\mathrm{GHZ}\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}.$$

stabilizers
-1 XXX
+1 ZZI
+1 ZIZ



Cost of Error Correction

Computer

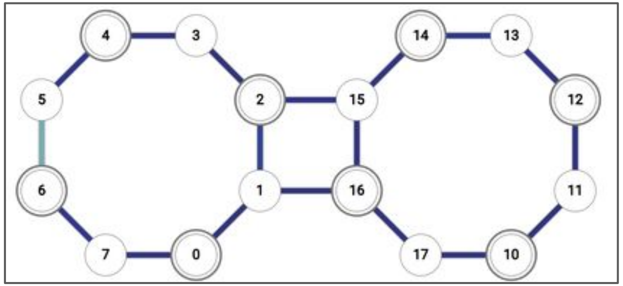
- Gate duration
- Qubit connectivity
- Qubit and gate quality, realistic noise models

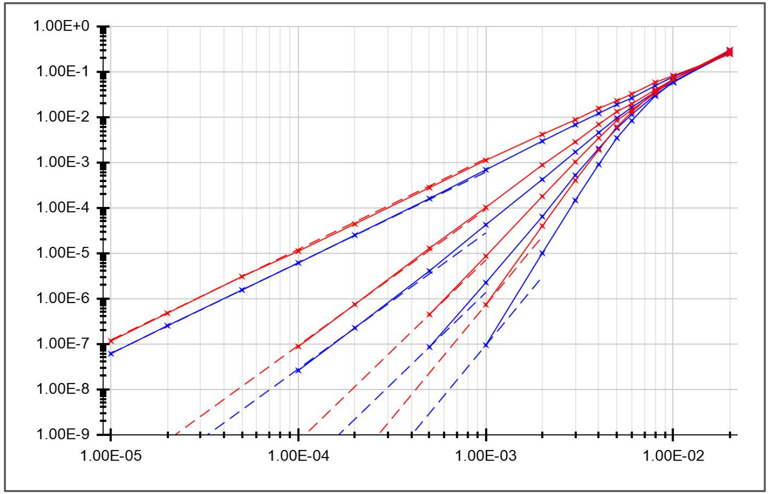
Code distance

- number of physical qubits
- number of syndrome measurements in time

Decoder performance

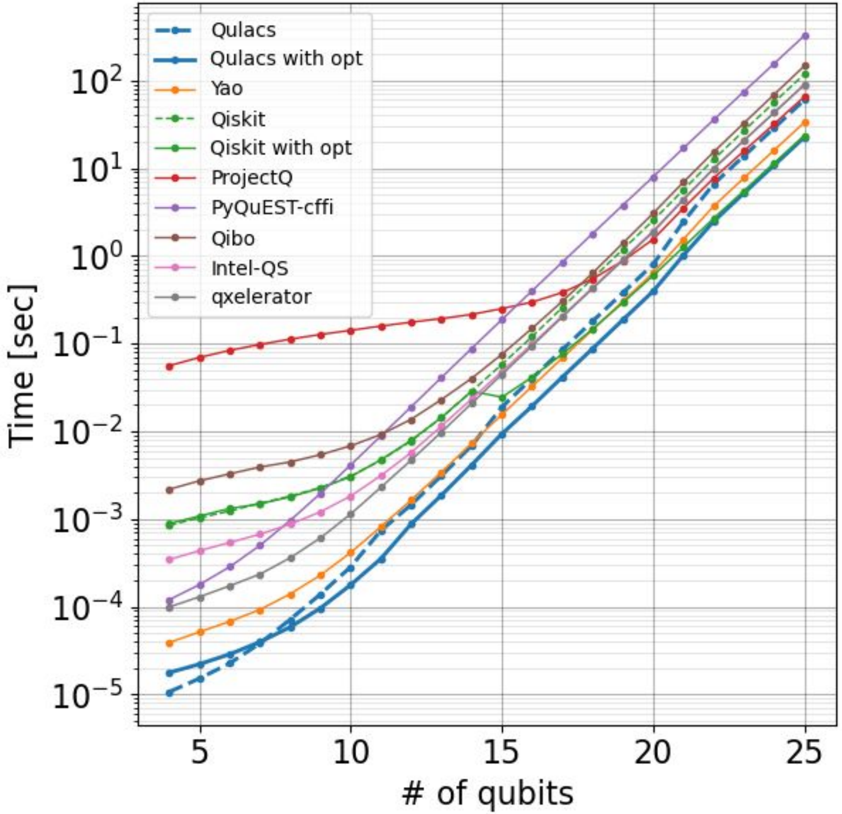
- what is the error suppression rate? (code dist.)
- how fast does it operate? (infl. code distance)

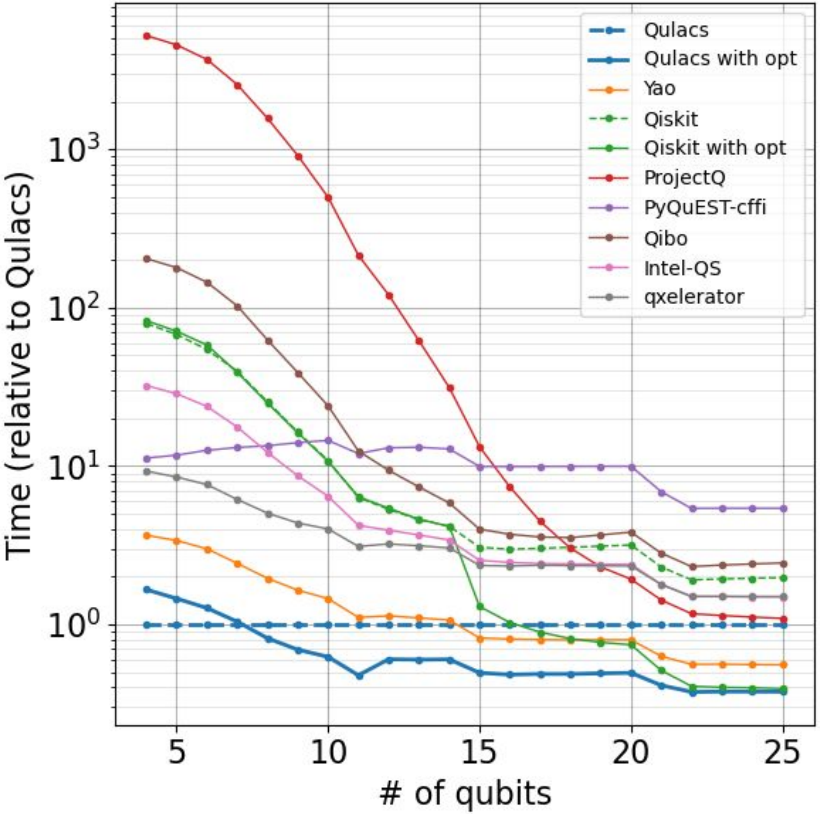




TOTAL: time overhead -> could negate Grover speed-up if not done right

Quantum circuit simulators





Space-time volume of a quantum computation