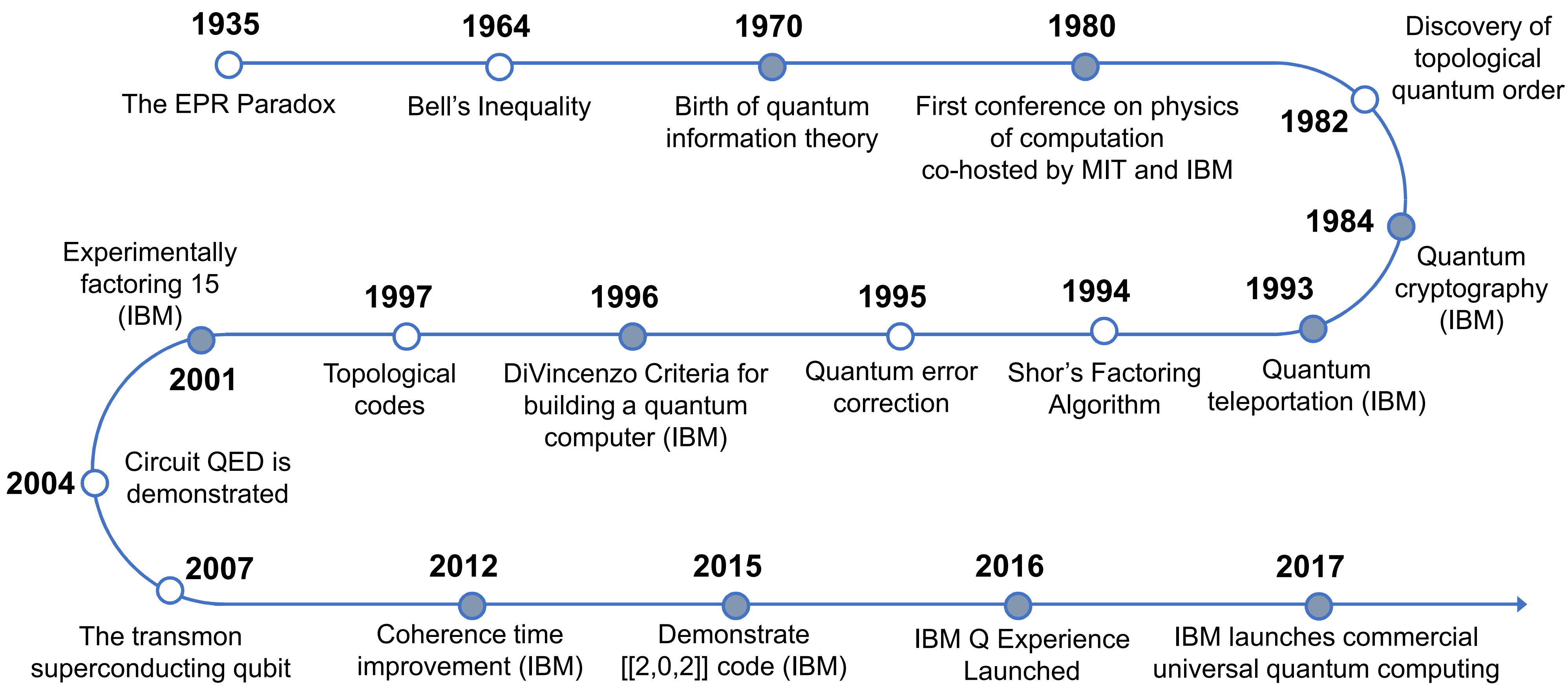
Efficient Programming of Near- and Long-Term Quantum Computers

Ali Javadi-Abhari

IBM Research

History of Quantum Computing

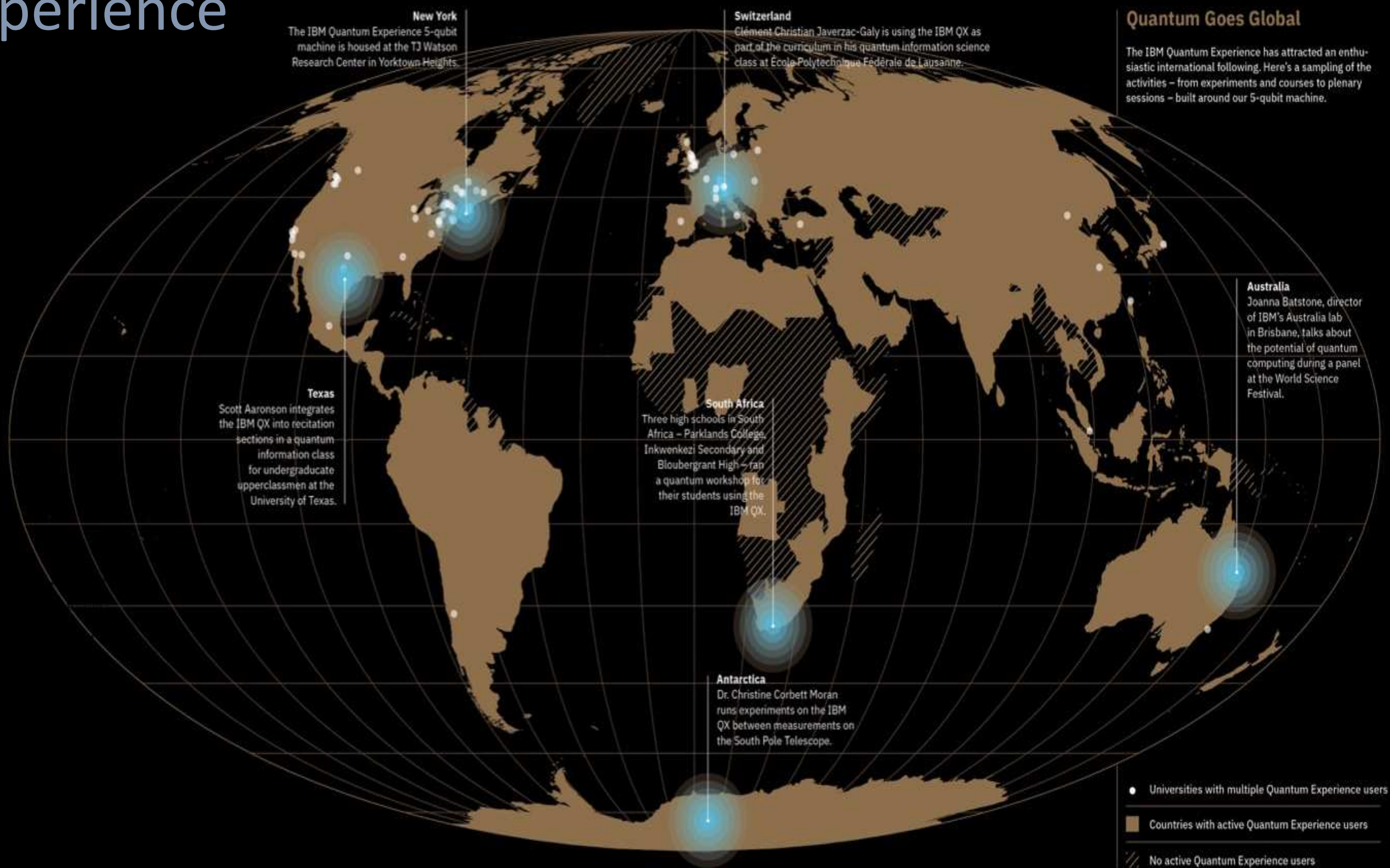


IBM Q Experience

80K
Unique Users

3 Million
Experiments Run

70
Scientific Papers

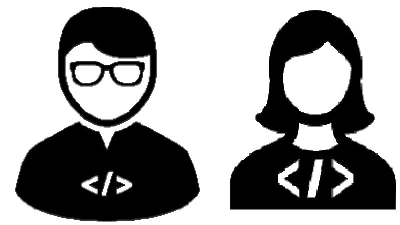


From Quantum Experience to Quantum Programs

QISKit

Build	Python Interface
Compile	Translate & Optimize

Quantum Developers

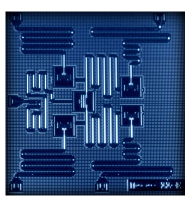


github.com/QISKit/qiskit-sdk-py/

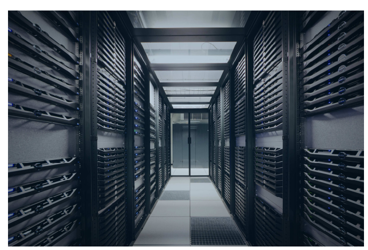
API

IBM Q Experience

Real Devices



Simulators

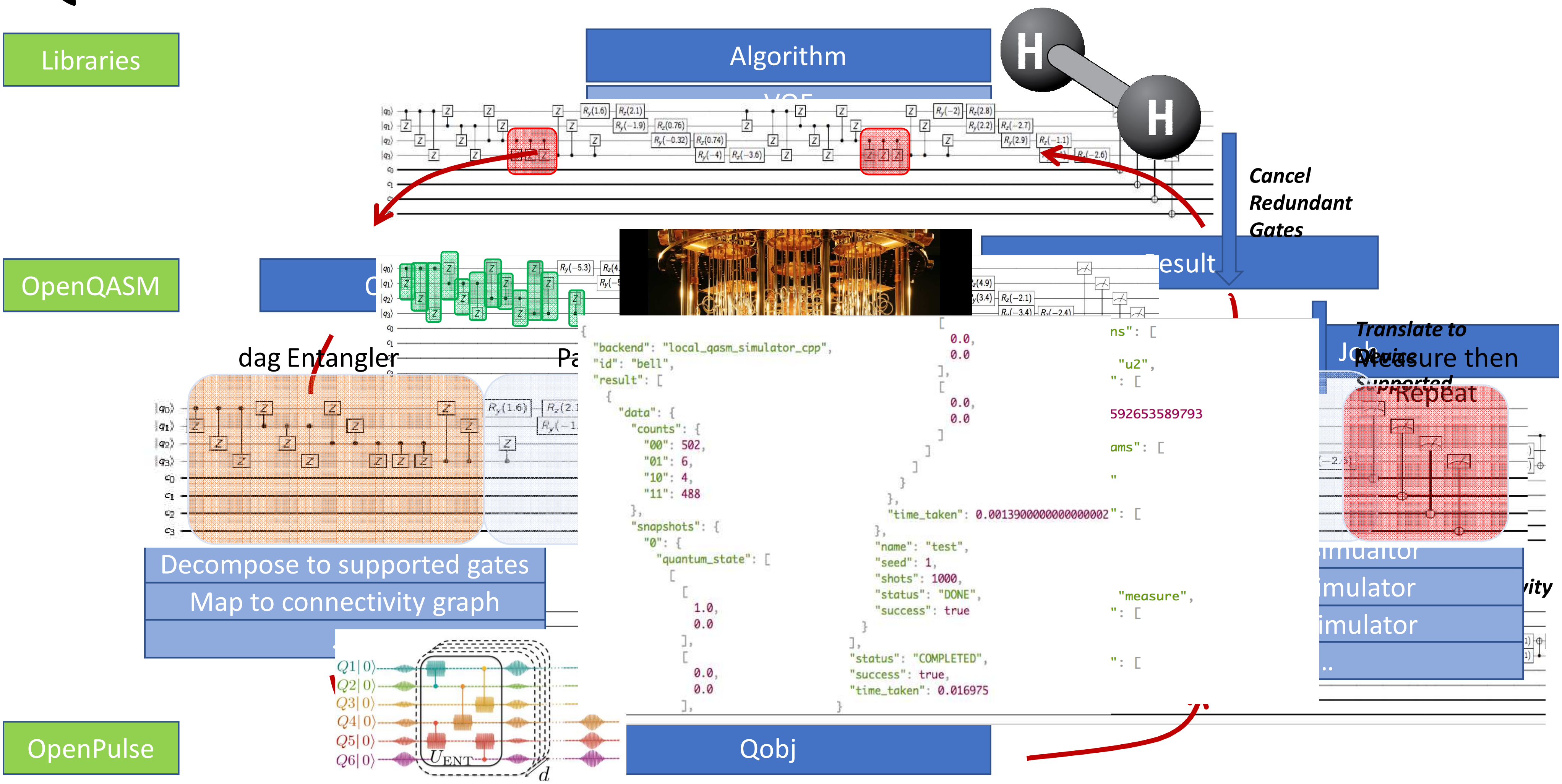


Laboratory

QISKit Architecture

User Abstraction Levels

QISKit Components



Challenges for near-term quantum computing

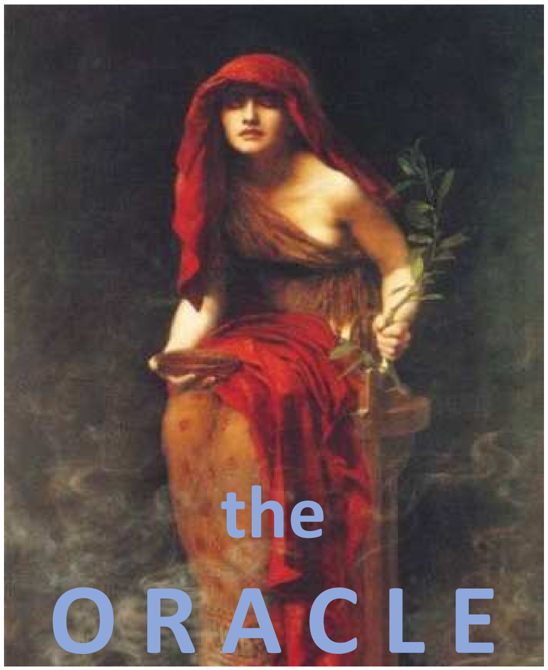
1. Don't have many qubits

2. Can't do many gates

    Gate error: gates are imperfect

    Relaxation: qubits do not retain state for long

Bernstein-Vazirani Algorithm*



Input (query) ← x_{n-1} … x_1 x_0

Secret Bitstring: s_{n-1} … s_1 s_0

Output (result) → $x_{n-1}s_{n-1} \oplus \ldots \oplus x_1 s_1 \oplus x_0 s_0$

Optimal classical strategy:

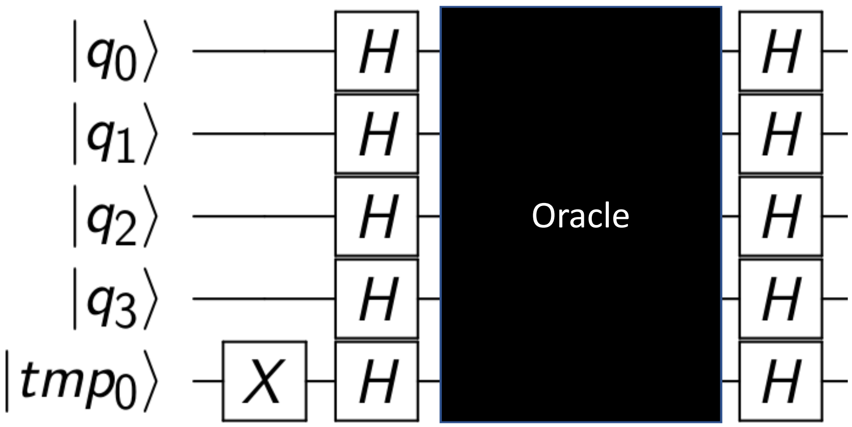
X = 1 0 … 0 0 (2^{n-1})
X = 0 1 … 0 0 (2^{n-2})
.
.
X = 0 0 … 1 0 (2)
X = 0 0 … 0 1 (1)

n tries

**E. Bernstein & U. Vazirani, STOC, 93*

Bernstein-Vazirani Solution

Wherever there’s CNOT (i.e. the secret bitstring has a `1`), phase kickback puts that control qubit in state |1>.



Awards

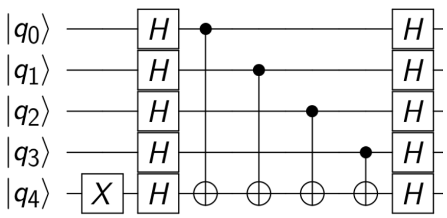
IBM Q Awards Developer Challenge

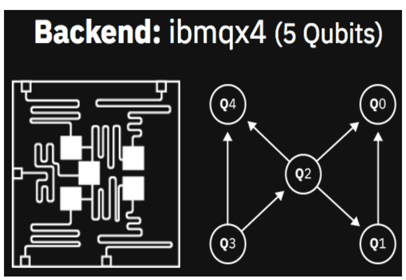
…t of Gate Errors and the Role of Software

Win $5000 in prizes

Deadline: 15th May 2018

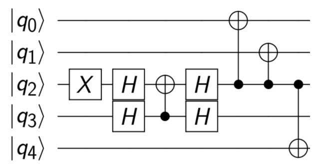
Programmed Circuit

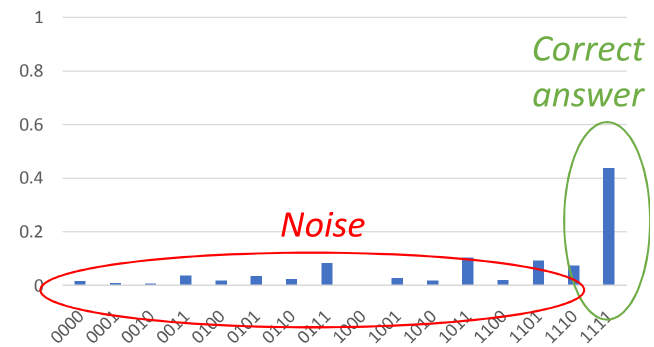




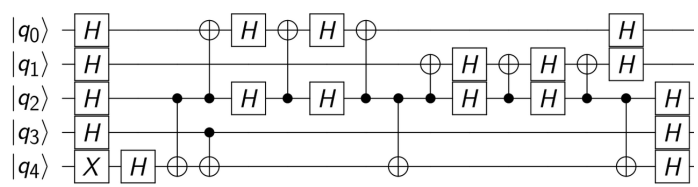
Device

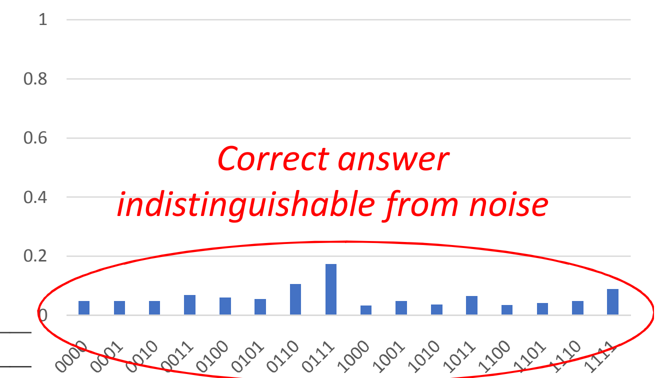
Good Compiled Circuit #1





Bad Compiled Circuit #2



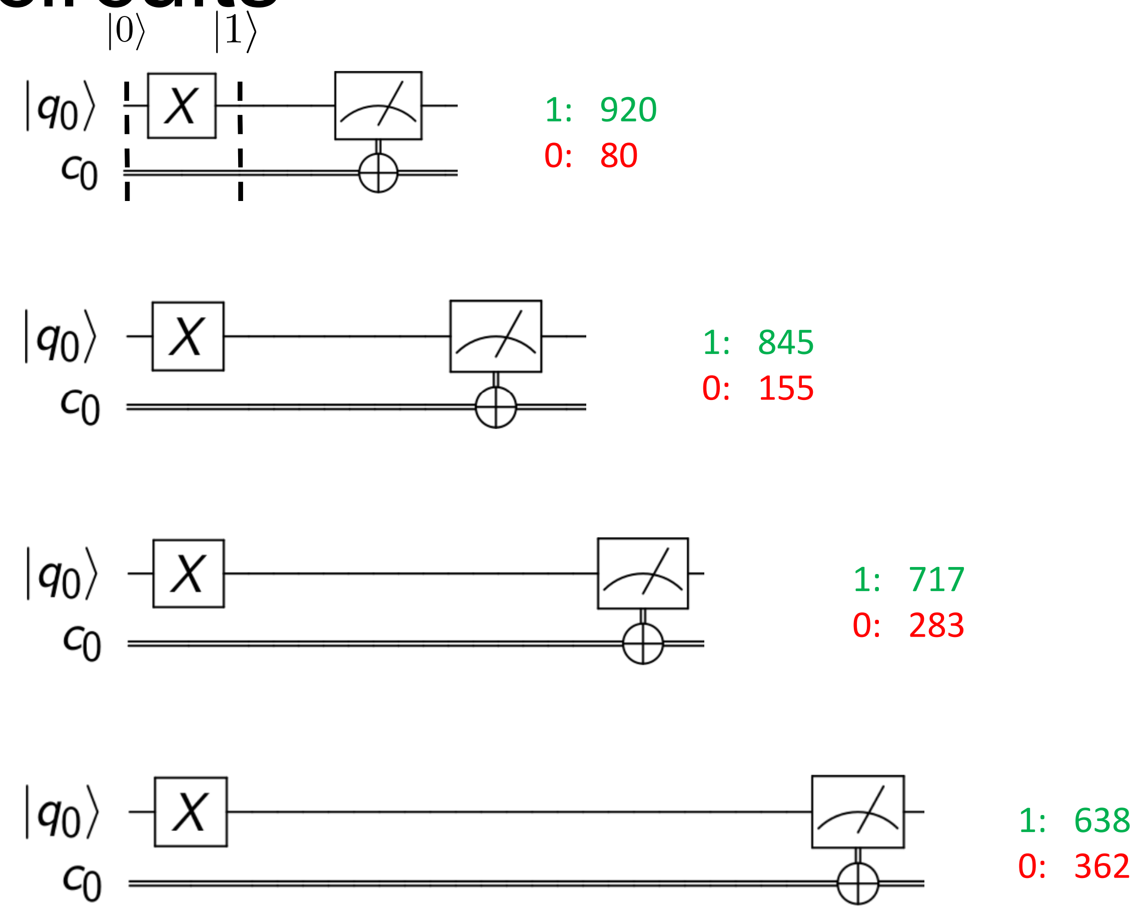


QISKit
Live Demo

Available here:

https://github.com/ajavadia/qiskit-sdk-py/blob/Demo/demo/Relaxation%20Demo.ipynb

Find qubit relaxation rate by running circuits

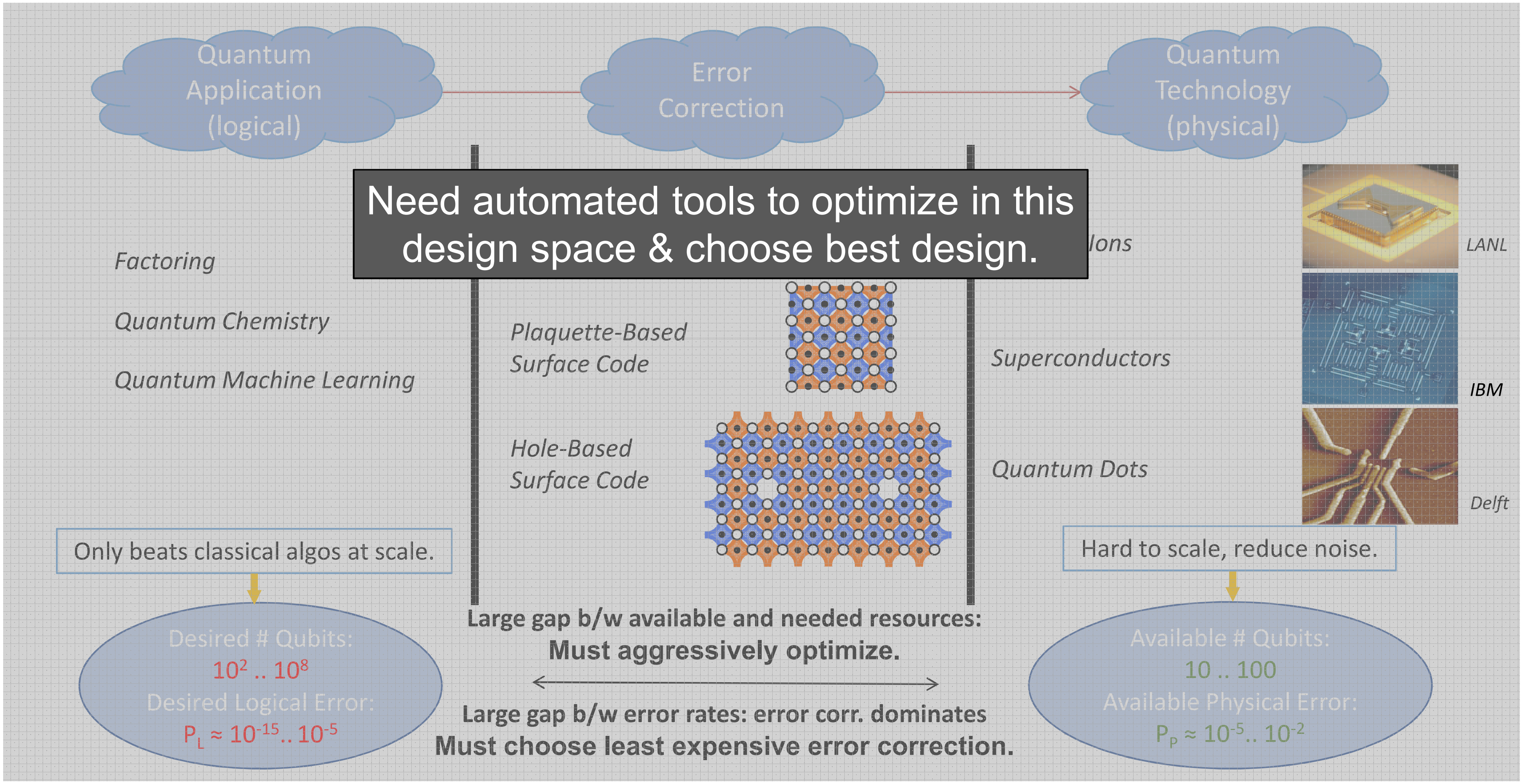


1. Put qubit in excited state and wait *variable amounts of time,* then measure.

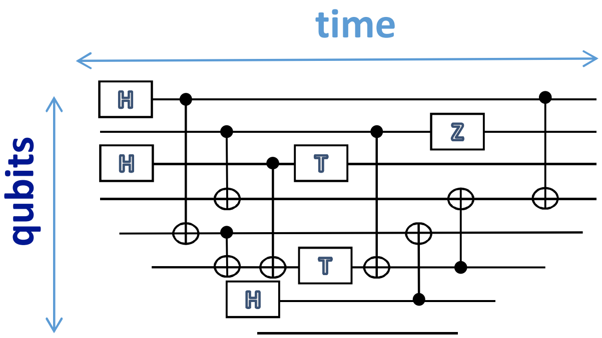
2. Repeat each circuit many times (e.g. 1000 shots) to approximate probability of unwanted |0> state in each.

3. Find relaxation rate by fitting an exponential decay curve to the data.

Long-Term Road to Fault-Tolerant QC



Best Design Has Minimum Resource Usage

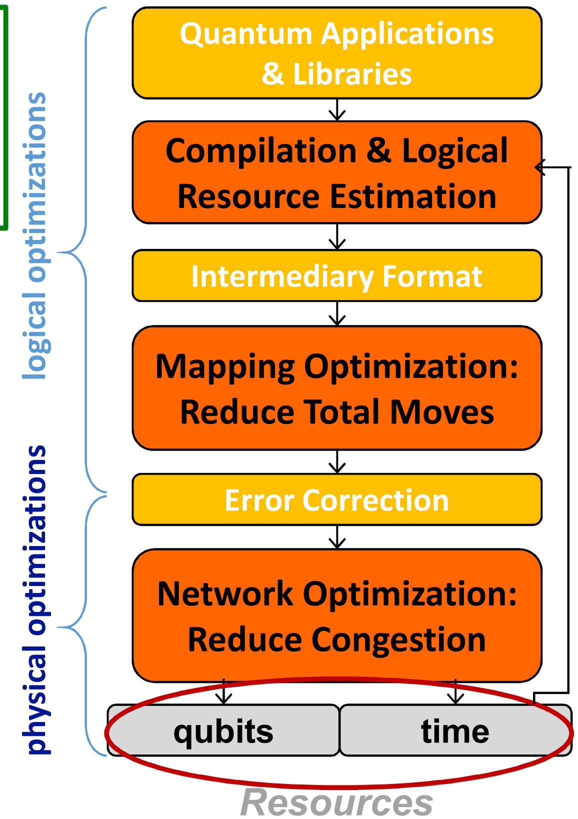


Main questions:

1) How many resources (qubit/time) required for a particular design?
2) To what extent can they be optimized?
3) What is the best design choice among many?

Scaffolds:

- First end-to-end toolflow for QC system design.
- One of first to examine full-scale applications, diverse technologies.

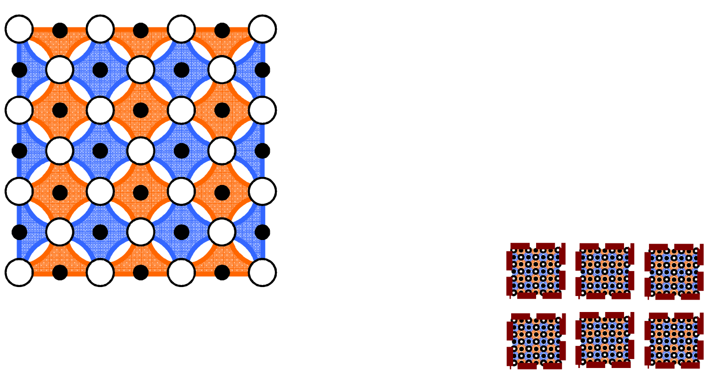


github.com/epiqc/ScaffCC/

Space Overhead of Error Correction

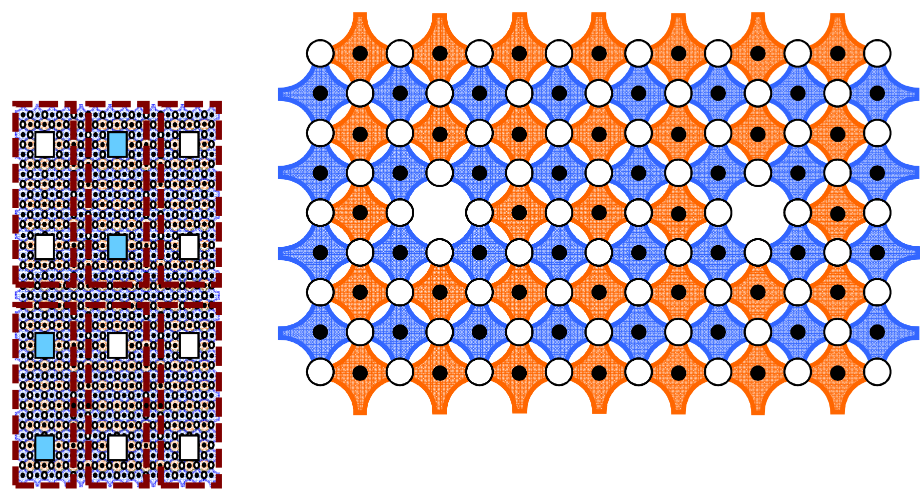
Plaquette-Based

+ Uses fewer physical qubits.



Hole-Based

- Uses more physical qubits.

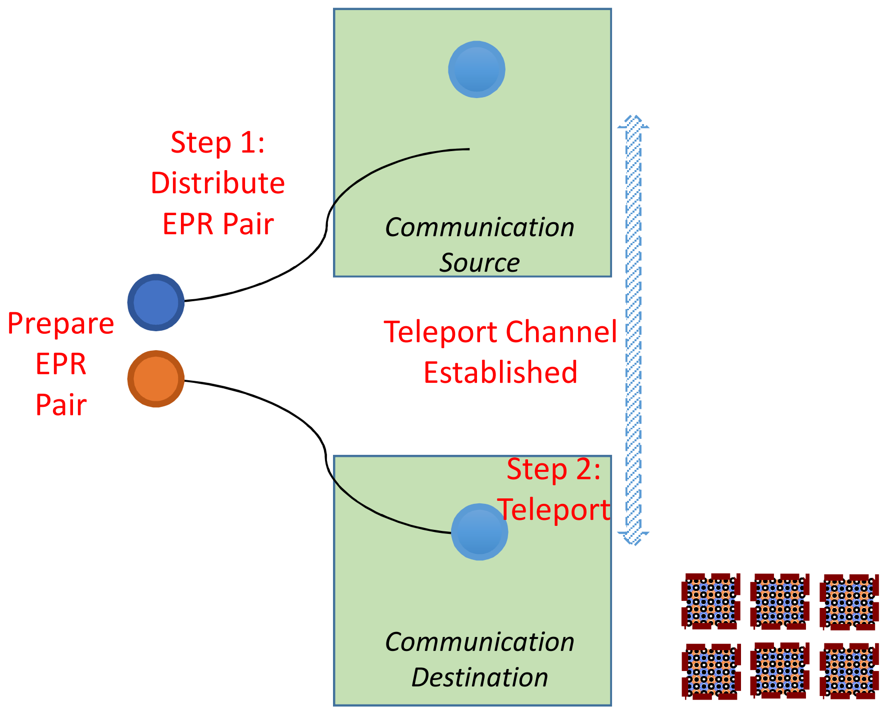


Time Overhead of Error Correction

Plaquette-Based: 2-step communication

- Qubits slowly move close to each other to interact.

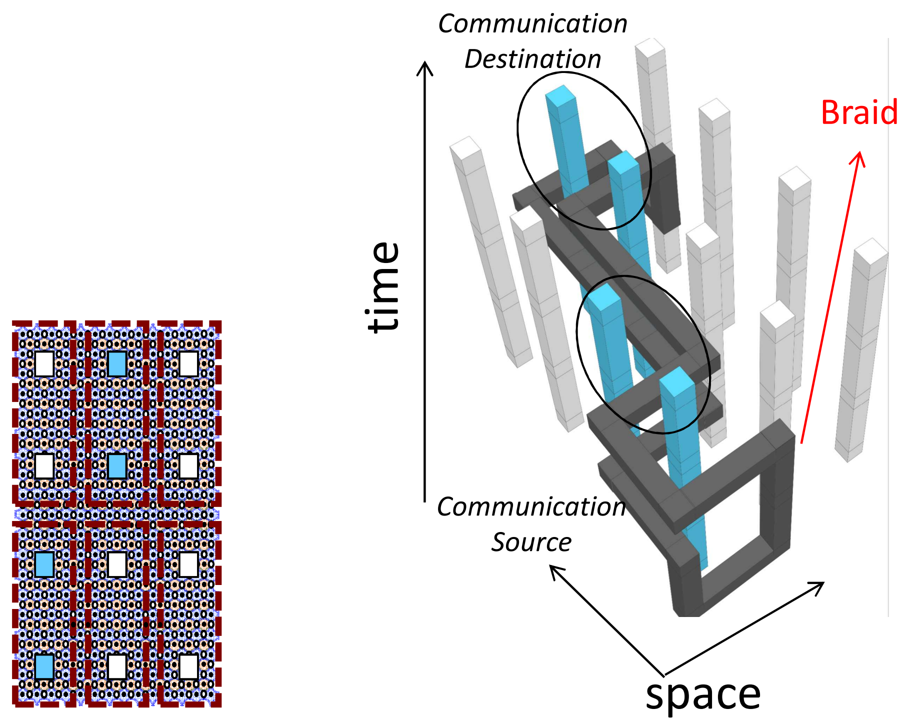
+ EPRs are decoupled from data: can be prefetched.



Hole-Based: 1-step communication

+ Braids move fast: n hops per cycle.

- Braids can't be pre-fetched.



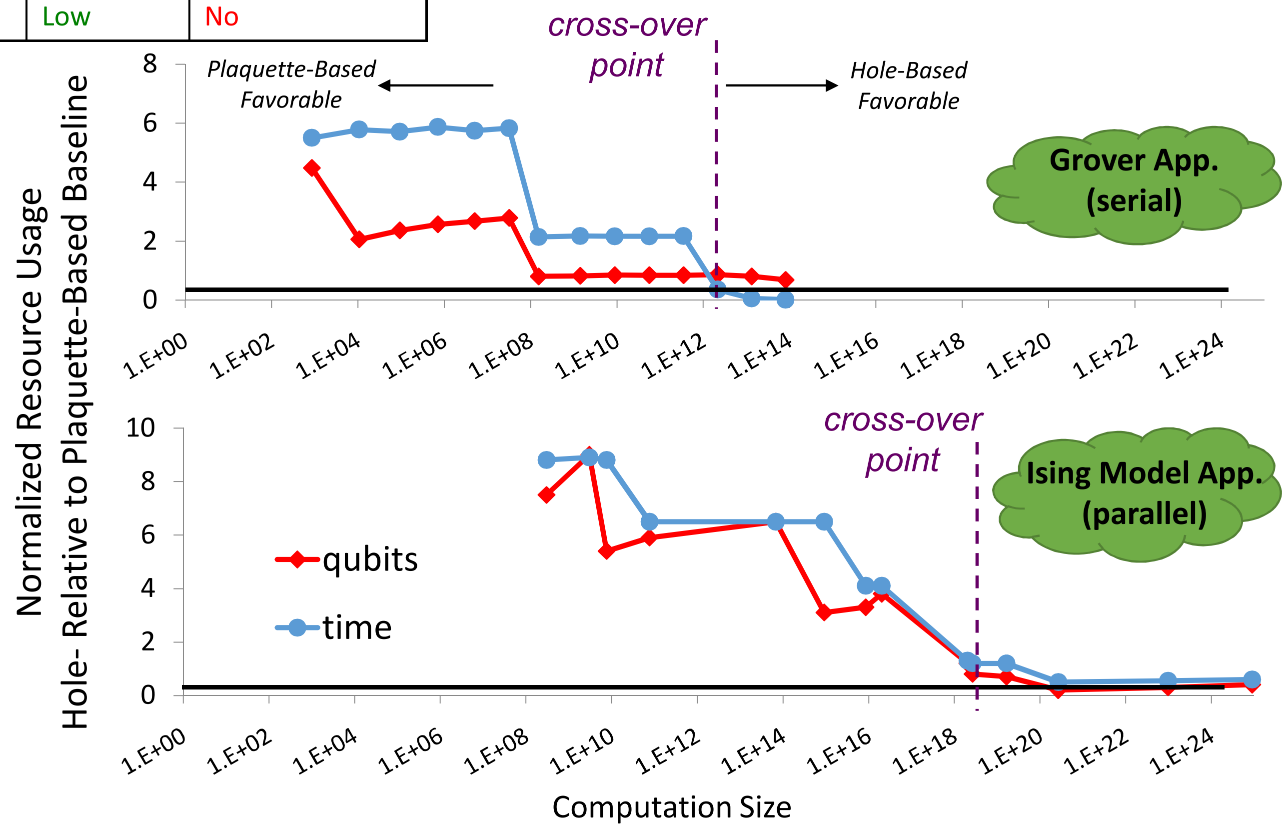
Application Dictates Code Favorability

Error Correction Type	Communication Method	Space (Qubits)	Time (Latency)	Pre-fetchable?
Plaquette-Based	Teleportations	Low	High	Yes
Hole-Based	Braids	High	Low	No

<u>Cross-over point:</u>
The computation size at which hole-based has better (lower) *space x time,* compared to plaquette-based.

<u>Cross-over point occurs much later for parallel apps:</u>
Plaquette-based code stays better for longer. Due to ability to schedule EPR pre-distribution around congestion.

<u>Tools are needed for these insights:</u>
Much of prior work had assumed hole-based to be better by default.



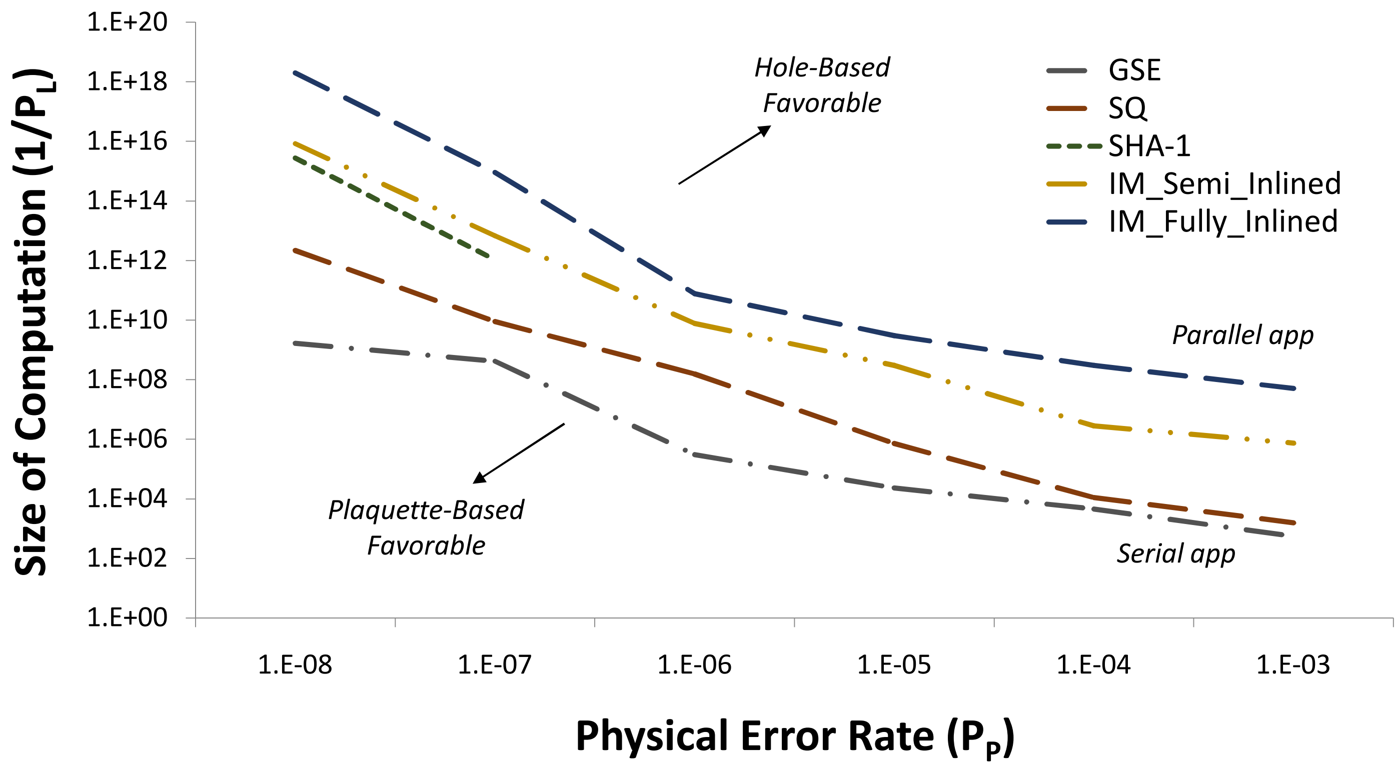
Co-Design of Applications, QEC Codes, Devices

Co-design for Maximum Benefit:

+ Very different technologies, with different constraints

+ Very different applications, with different characteristics

+ Very different Error Correction Codes, with different overheads



Javadi-Abhari et al. MICRO 17

Get Involved!

qiskit.org *Explore*

github.com/qiskit/qiskit-tutorial *Learn more*

github.com/qiskit *Contribute code*

qiskit.slack.com *Help define*

Awards **IBM Q Awards** Developer Challenge *Win $5000 in prizes*
Deadline: 15th May 2018