

# Mind the Gap!

## Challenges and Opportunities in Closing the Algorithms-to-Devices Gap in Quantum Computing

Margaret Martonosi

Dept. of Computer Science

Princeton University

Work by Princeton Group Members and Alums:

Chuck Garcia

Prakash Murali

Wei Tang

Teague Tomesh

Esin Tureci

Ellie Vogel



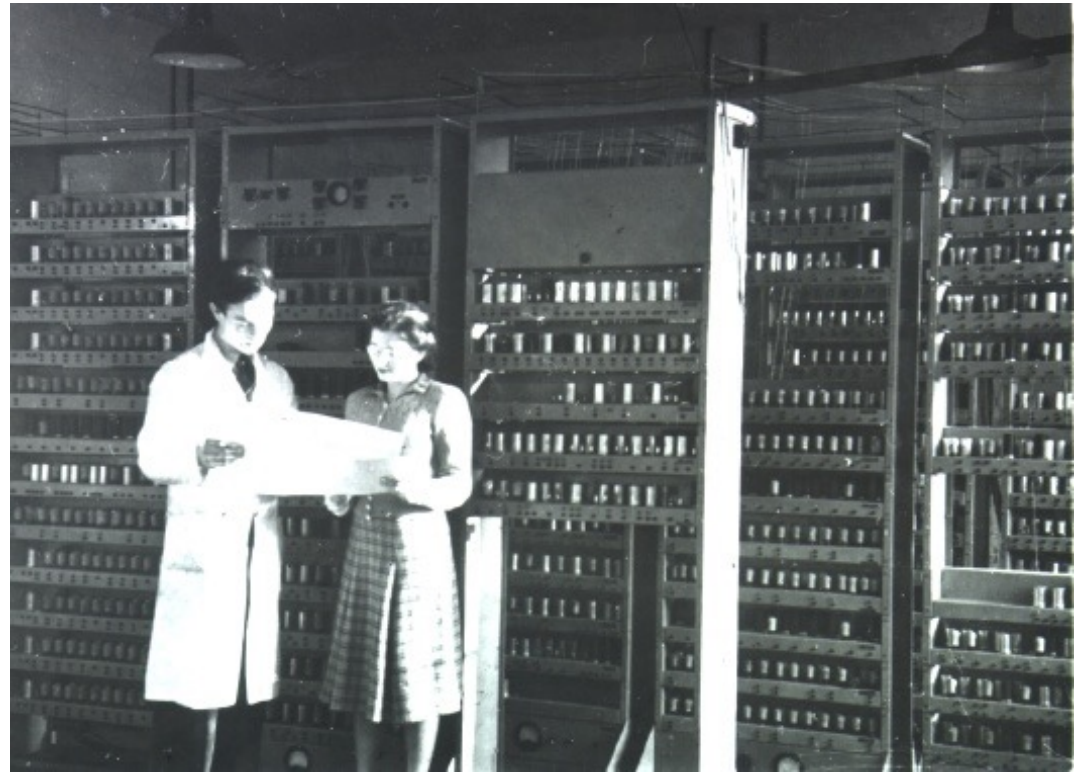
# An Analogy: Classical Computing in 1950

~1950's Classical Computing

Algorithms

Assembly Language

Vacuum Tubes, Relay Circuits



# Alan Turing in 1950: “Can Machines Think?”



A. M. Turing (1950) *Computing Machinery and Intelligence*. *Mind* 49: 433-460.

---

## COMPUTING MACHINERY AND INTELLIGENCE

By A. M. Turing

### 1. The Imitation Game

I propose to consider the question, "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think." The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words "machine" and "think" are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, "Can machines think?" is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.

# Classical Computing Today: Massive Scaling Achieved through Innovation and Smart Abstraction

~1950's Classical Computing

Algorithms

Assembly Language

Vacuum Tubes, Relay Circuits

Today's Classical Computing

Algorithms

High-Level Languages

Compiler OS

Architecture

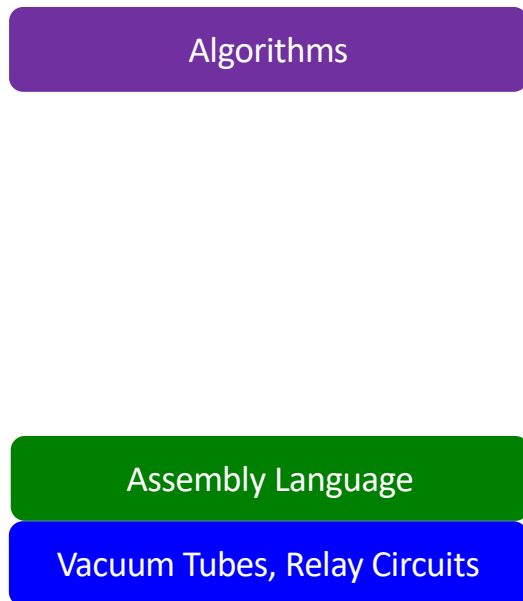
Modular hardware blocks:  
Gates, registers

VLSI Circuits

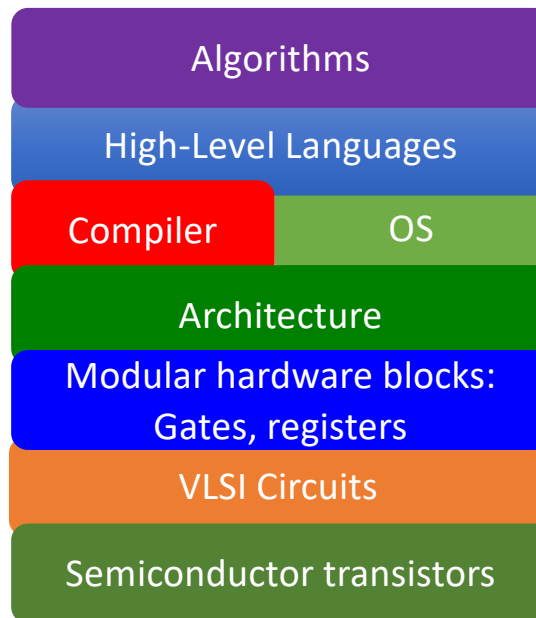
Semiconductor transistors

# Quantum Systems Today

## ~1950's Classical Computing



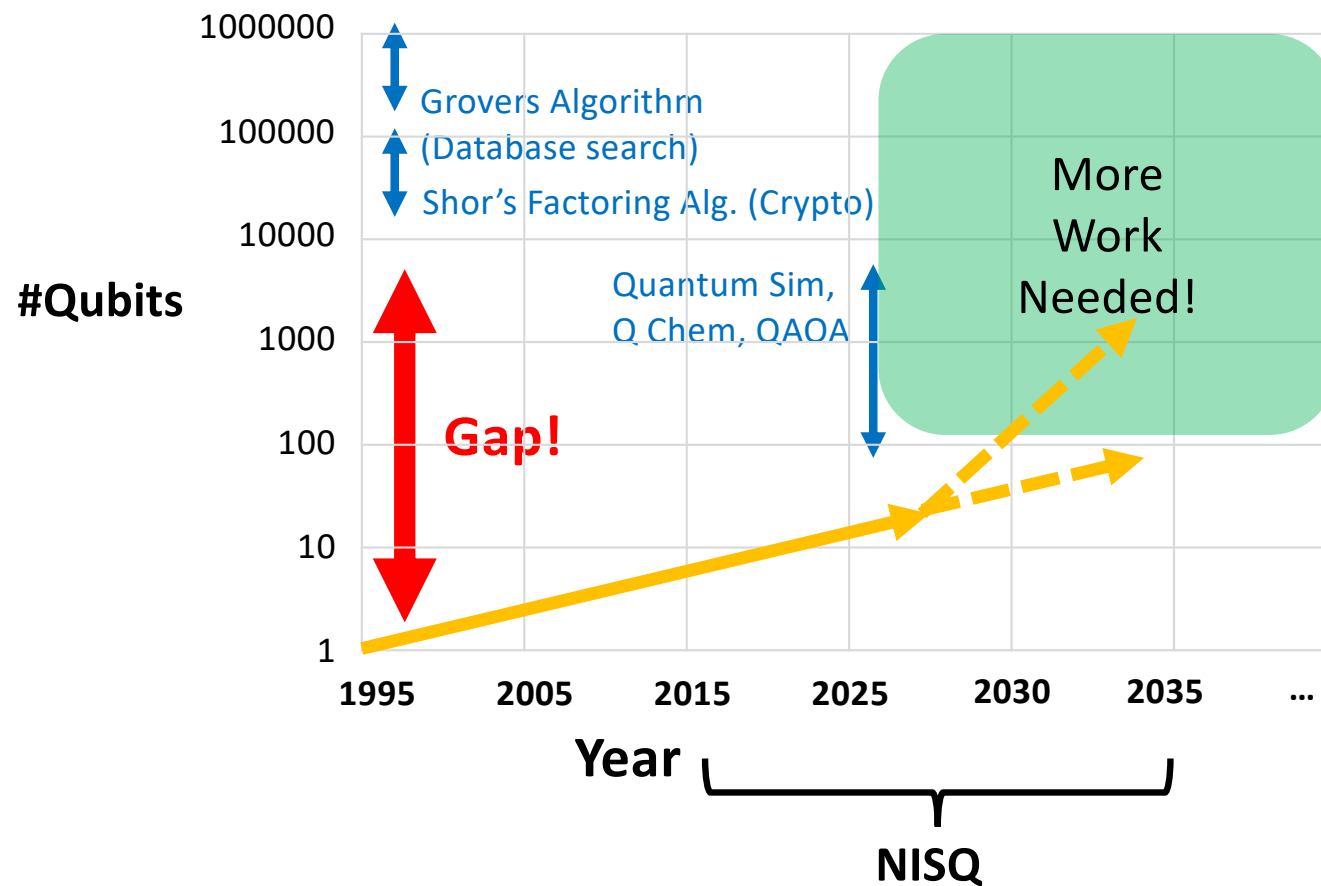
## Today's Classical Computing



## Quantum Systems



# Algorithms to Machines Gap: Status Check

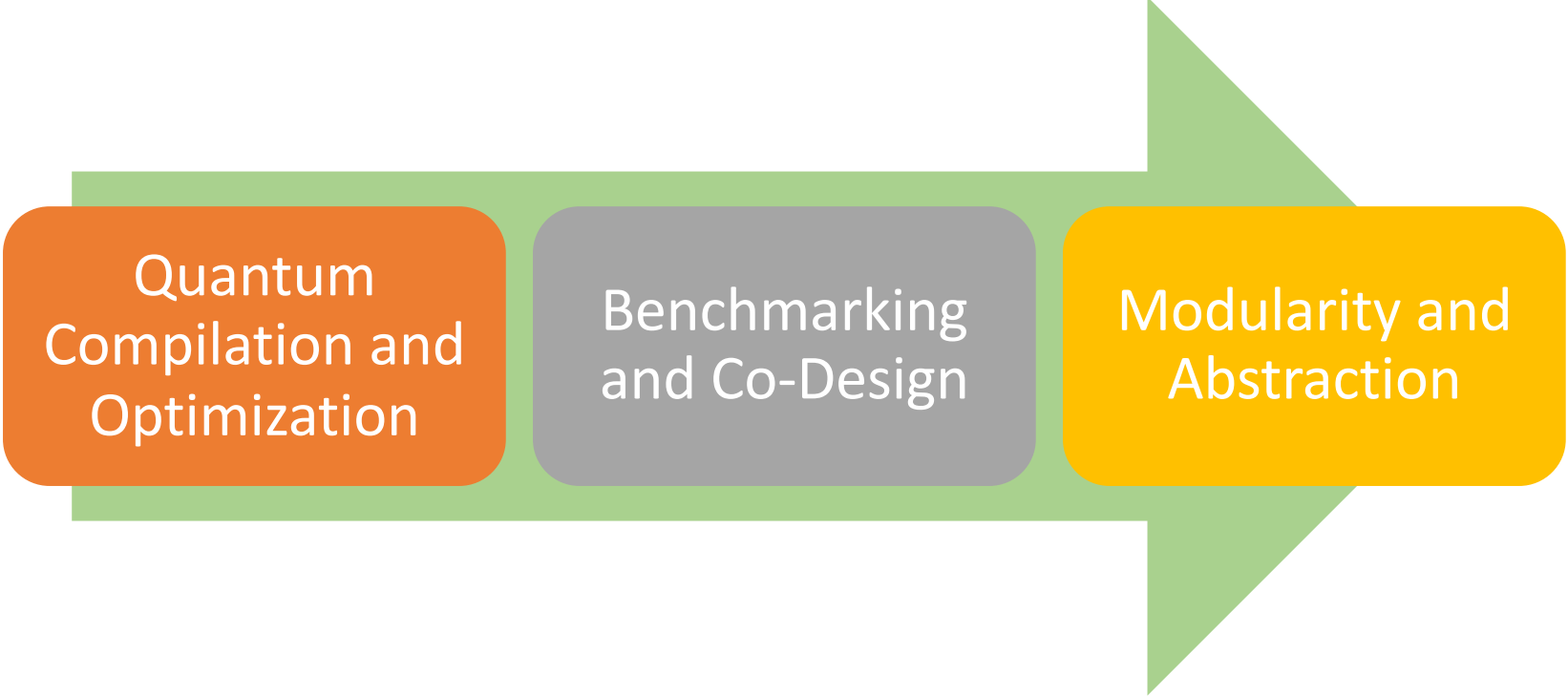


NISQ Era shows distinct “chapters” of evolution

1. Compilation and Mapping
2. Co-Design
3. Bridging NISQ and Fault-tolerance via Modularity and Abstraction

# This talk: Some examples and opportunities from each “NISQ Chapter”

---

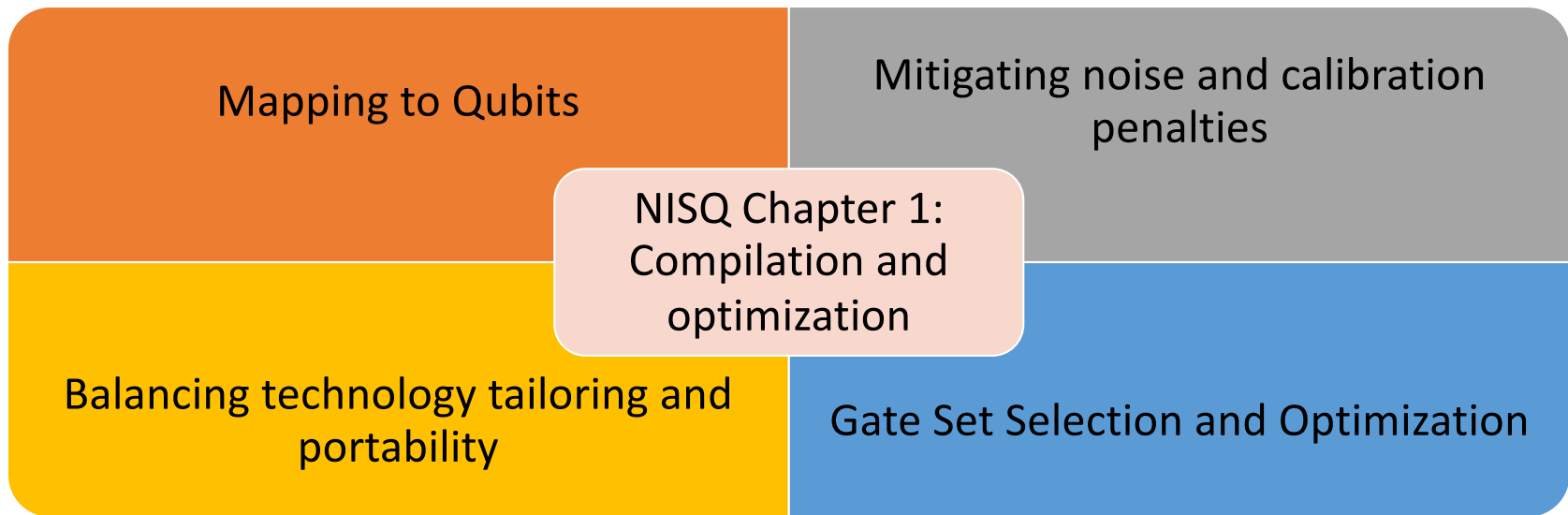


Quantum  
Compilation and  
Optimization

Benchmarking  
and Co-Design

Modularity and  
Abstraction

# Quantum Compilation and Optimization



Prakash Murali et al. ASPLOS'19. <https://doi.org/10.1145/3297858.3304075>

Prakash Murali, et al. ISCA 2019. <https://doi.org/10.1145/3307650.3322273>

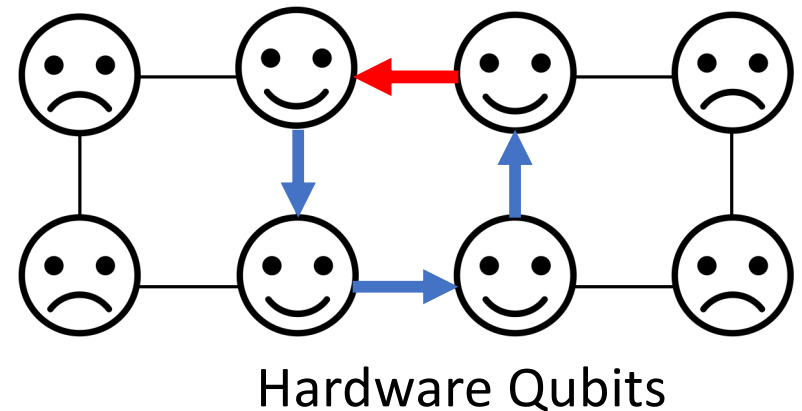
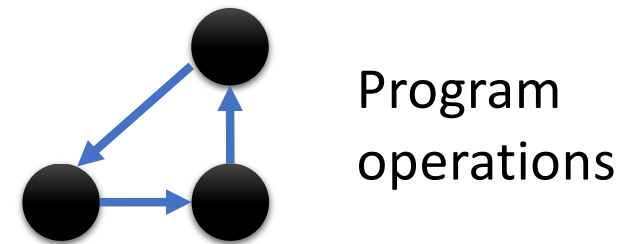
Ali JavadiAbhari et al. CF 2014. <https://mrmgroup.cs.princeton.edu/papers/CF2014.pdf>

P. Murali, et al, ISCA 2020. <https://ieeexplore.ieee.org/abstract/document/9138945>

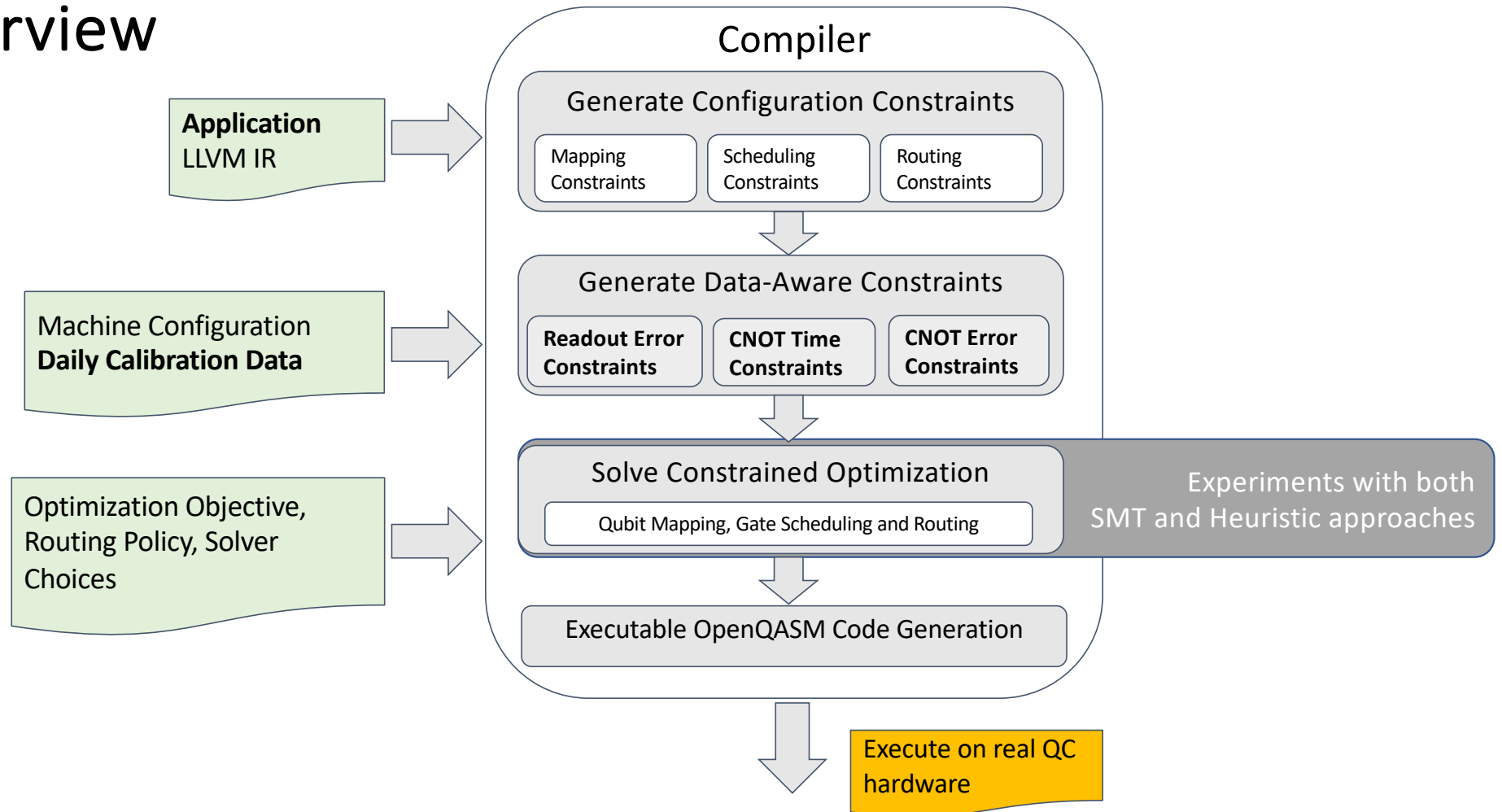


# Our Work: Noise-Adaptive Compilation

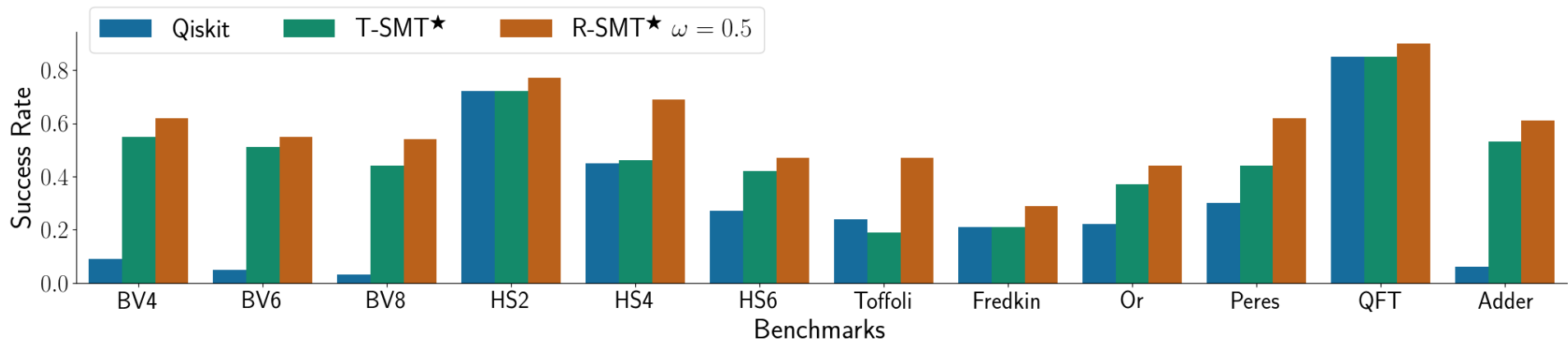
- Compile once-per-execution using noise characterization data
- Strategy 1: Place program qubits onto hardware qubits based on operation error rates
- Strategy 2: Reduce decoherence errors by scheduling gates to finish within the coherence time of each qubit
- Strategy 3: Reduce communication and use reliable routing paths, avoiding bad gates
- Full stack compiler targeted for a real QC system IBMQ16



# Compilation Overview



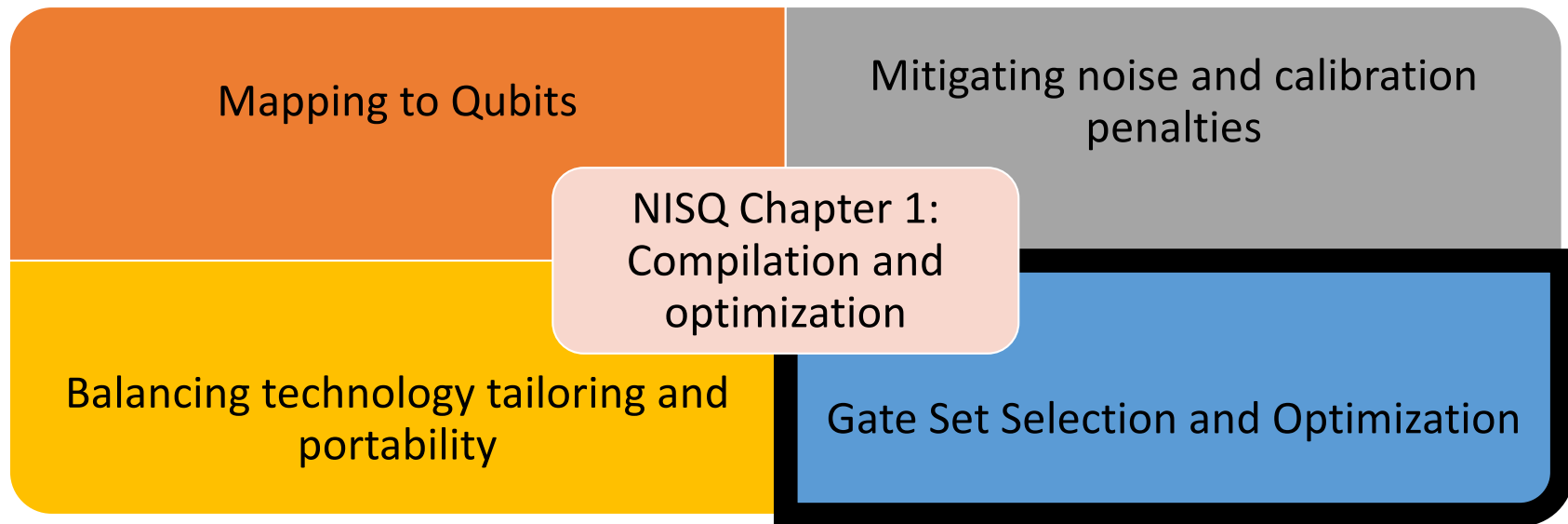
# Significant Boost in Program Success Rate



- Scaffold -> LLVM -> Z3 -> **IBM 16-qubit Rueschlikon**
- Each “run” is 8192 runs, with results a distribution.
- **Geomean 2.9X improvement in measured success rate over industry standard IBM Qiskit compiler**
  - Some benchmarks up to 18X
- **Up to 9.2X over noise-unaware optimized baseline.**
- Improvements because of optimal qubit movement, usage of best CNOTs and qubit readout units.

[Murali et al. ASPLOS 2019]

# Quantum Compilation and Optimization



Prakash Murali et al. ASPLOS'19. <https://doi.org/10.1145/3297858.3304075>

Prakash Murali, et al. ISCA 2019. <https://doi.org/10.1145/3307650.3322273>

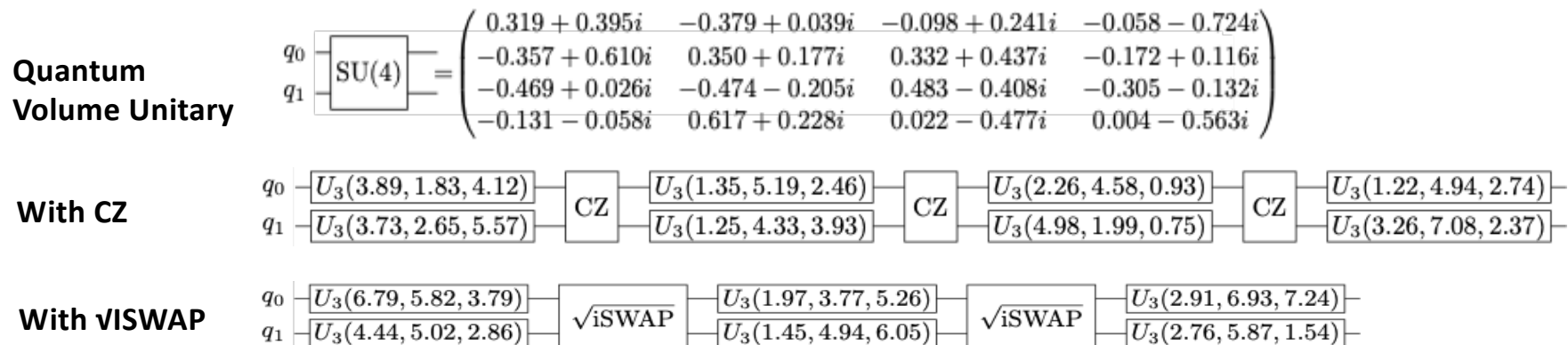
Ali JavadiAbhari et al. CF 2014. <https://mrmgroup.cs.princeton.edu/papers/CF2014.pdf>

P. Murali, et al, ISCA 2020. <https://ieeexplore.ieee.org/abstract/document/9138945>

...

# Instruction Set Design for NISQ Executions

- ISA choices affect the instruction count and fidelity of QC programs



- ISA choices also affect calibration costs
  - More gate types in the ISA => Higher calibration cost => Less time available for application computations
- Our work is the first to study the **expressivity vs. calibration tradeoff in QC**
  - Similar to RISC vs. CISC choices in classical computer architecture

### Instruction sets with a single type of two-qubit gate

IBM (CNOT), early Rigetti and Google's systems (CZ)

**Application:** Too restrictive, increases instruction count

**Hardware:** Easy to calibrate ( $\leq 1$  hour per day of calibration time on current systems)

$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

**Research Questions:** Is there an ISA which is highly expressive for applications and easy to calibrate? Will a small number of two-qubit gate types provide sufficient expressivity for NISQ applications? What are the calibration overheads of such an instruction set?

Example:  $fSim(\pi/2, \pi/6)$ ,  $fSim(\pi/4, 0)$ ,  $fSim(0, \pi)$ ,  $fSim(\pi/2, 0)$

### Instruction sets with continuous two-qubit gates

Rigetti:  $XY(\theta)$ , Google:  $fSim(\theta, \phi)$ ,

ETH:  $CPHASE(\theta)$

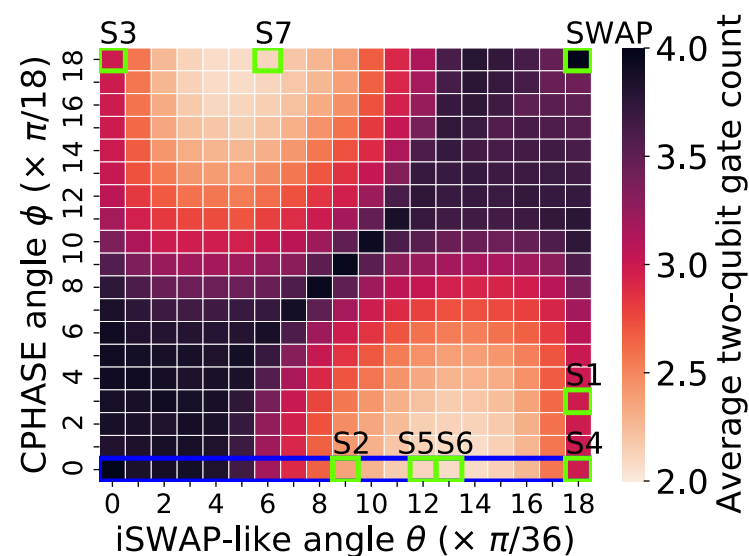
**Application:** Very expressive, lower instruction count

**Hardware:** Difficult to calibrate a large number of gate types

$$XY(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta/2) & i\sin(\theta/2) & 0 \\ 0 & i\sin(\theta/2) & \cos(\theta/2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

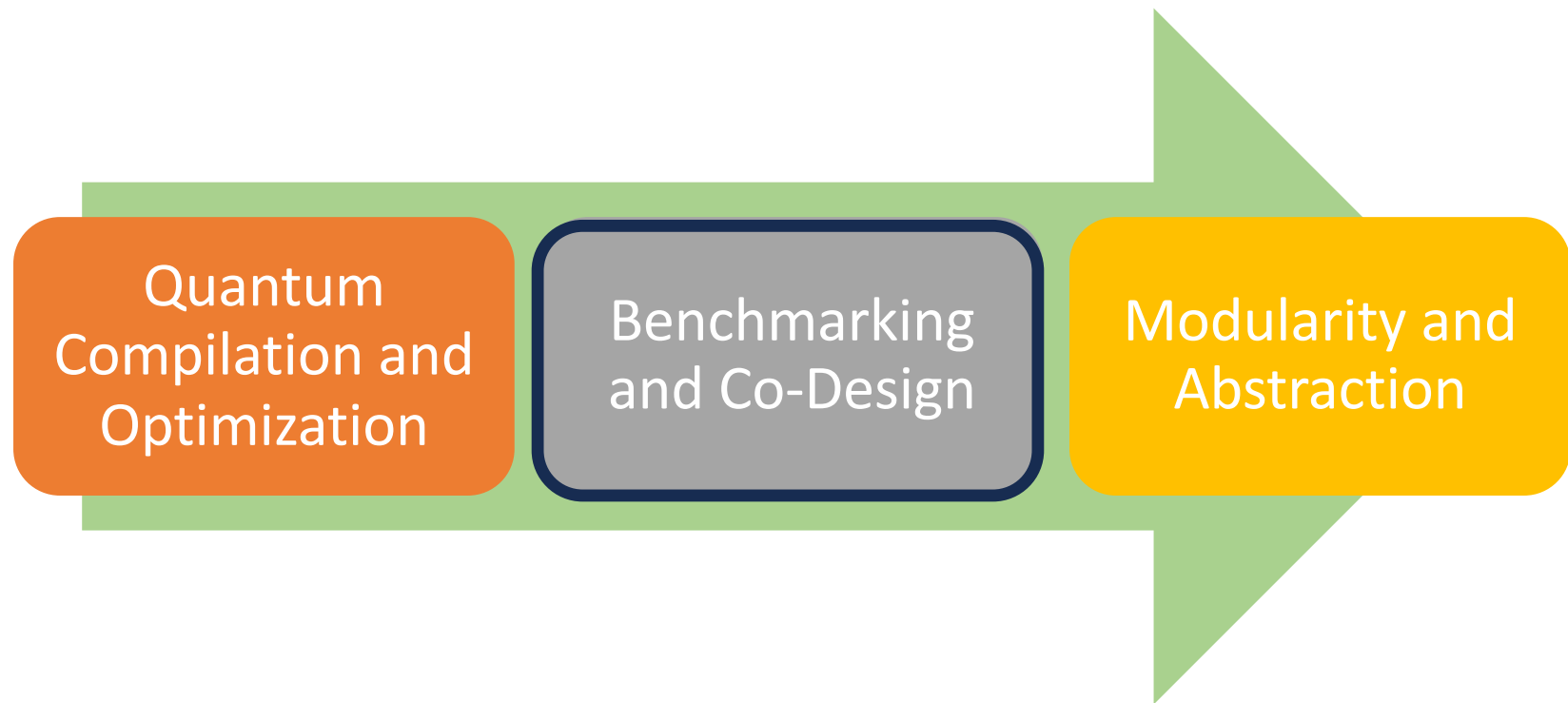
# Result: Use ISAs with 4-8 types of two-qubit gates

- Method: Configurable toolflow to study ISA choices using realistic noise simulations of Rigetti and Google systems.
- Use 4-8 gate types to improve expressivity, fidelity:
  - offers 1.5-2X reduction in instruction counts and fidelity, compared to a single gate type
  - near-optimal instruction counts and fidelity, compared to a fully continuous gate set
  - Diminishing returns in fidelity after 8+ types
- Tradeoff in calibration time:
  - 4-8X higher calibration time compared to a single gate type, but still practical on current systems
  - Two orders of magnitude lesser overhead than a continuous gate set
- More details on toolflow, application suite, calibration model, noise variations across gate types in the paper.



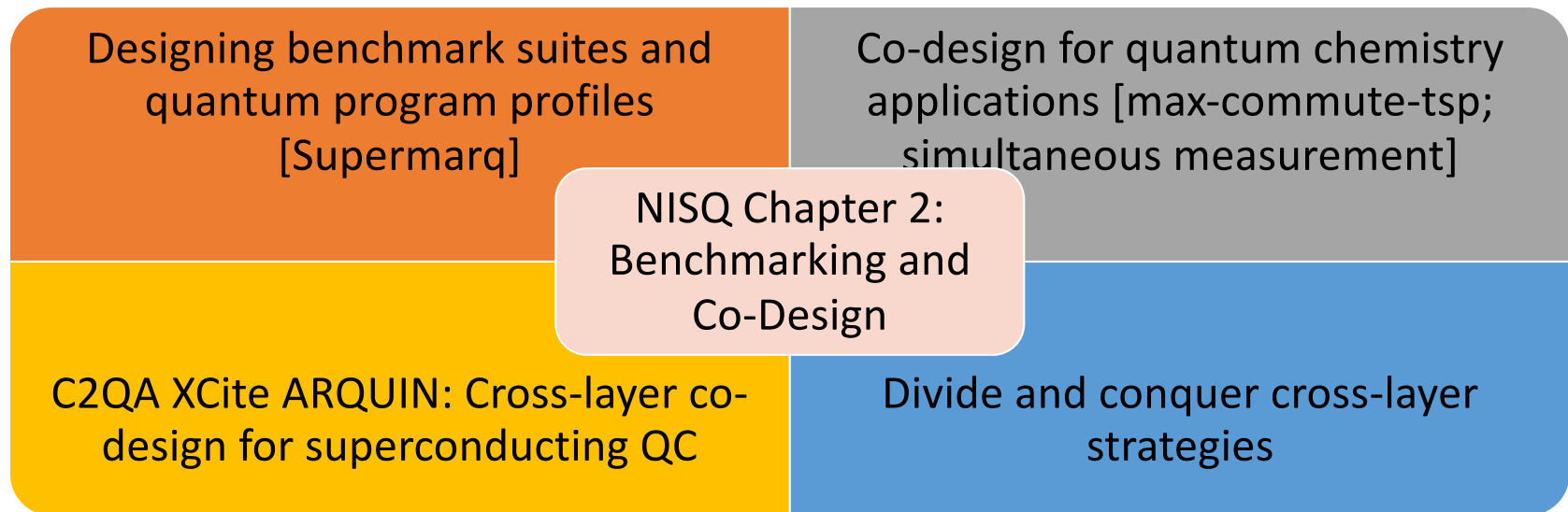
**QV Instruction count vs. Gate type**  
Choose gate types based on application + gate set characterization

# This talk: Some examples and opportunities from each “NISQ Chapter”





# NISQ Chapter 2: Benchmarking and Co-Design

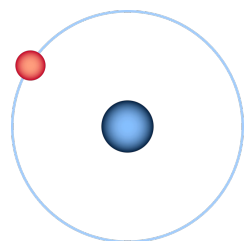
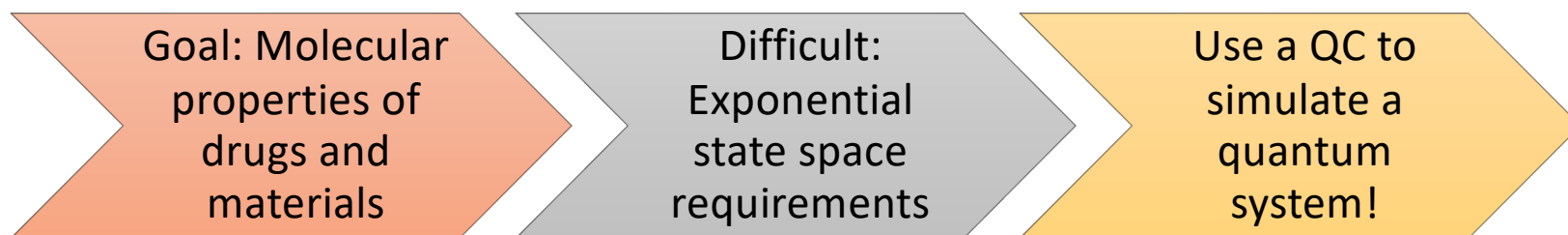


Teague Tomesh et al. HPCA '22 <https://arxiv.org/abs/2202.11045>

Teague Tomesh, et al. ICRC 2021. <https://mrmgroup.cs.princeton.edu/papers/tomesh-term-grouping.pdf>

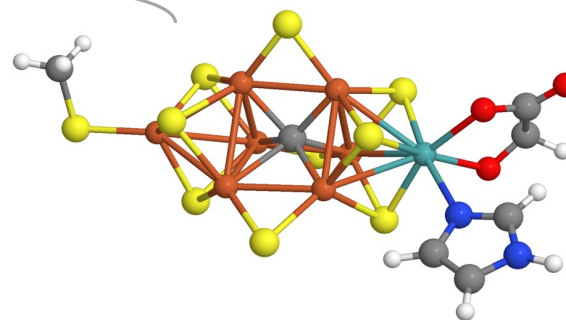
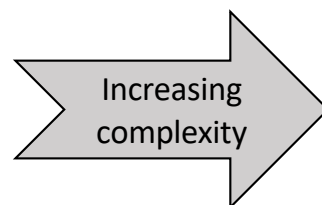
James Ang et al. ACM TQC 2024. <https://dl.acm.org/doi/pdf/10.1145/3674151>

# Example: Co-Design for Quantum Chemistry Apps



$$H = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} \\ \alpha_{2,1} & \alpha_{2,2} \end{pmatrix}$$

Hamiltonian describes allowed energy levels of the system

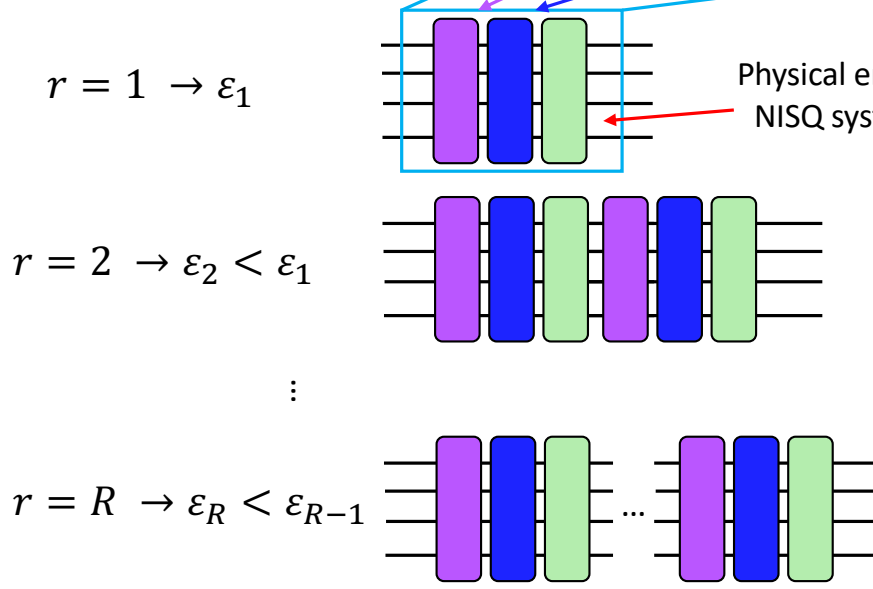


$$H = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,2^n} \\ \vdots & \ddots & \vdots \\ \alpha_{2^n,1} & \cdots & \alpha_{2^n,2^n} \end{pmatrix}$$

# Balancing algorithmic and physical error

- Trotterization  $\Rightarrow$  to simulate the system on a QC, we break the evolution up into discretized timesteps
- Physical error  $\Rightarrow$  Gates in real QC systems inject further error due to insufficient control fidelity

$$e^{-iHt} \approx \left( e^{-ic_1 P_1 \Delta t} e^{-ic_2 P_2 \Delta t} \dots e^{-ic_N P_N \Delta t} \right)^{t/\Delta t} + O(t\Delta t)$$



Physical error in NISQ systems

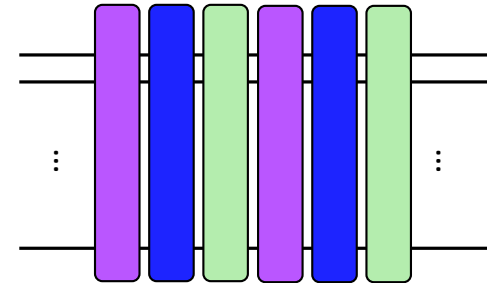
$r = t/\Delta t$   
 "Trotter number"

Algorithmic error

Hamiltonian simulation applications suffer from *algorithmic* and *physical* errors. Is there a way to simultaneously minimize both?

# Our solution: Full-stack co-design -> Commutation-aware compilation

- The QC sequentially simulates each Pauli term
- The compiler selects an ordering
  - Not all orderings equal



*Lexicographic (LEX):*  $H = 5 IIIY + 7 IYXX + 6 YIIY + 9 YXYX + 4 YYIY + 8 YYYY + 3 ZIII + 2 ZIZI + ZZZZ$

- Tranter et al. *J. Chem. Theory Comput.* 2018, 14, 11, 5617–5630 (mitigates physical errors well)

*Magnitude (MAG):*  $H = 9 YXYX + 8 YYYY + 7 IYXX + 6 YIIY + 5 IIIY + 4 YYIY + 3 ZIII + 2 ZIZI + ZZZZ$

- Tranter et al. *Entropy.* 2019, 21(12):1218 (mitigates algorithmic errors well)

## *Max-Commute-TSP (MCTSP)*

1. Group commuting terms
2. Order terms within each group
3. Clique-clique ordering heuristic


*Optimized Quantum Program Execution Ordering to Mitigate Errors in Simulations of Quantum Systems.* T Tomesh, et al.  
2021 International Conference on Rebooting Computing. Best Paper.

# Noiseless simulation reveals the impact of codesign

- Average over 79 benchmark molecules
- *Diamond distance* measures algorithmic error:

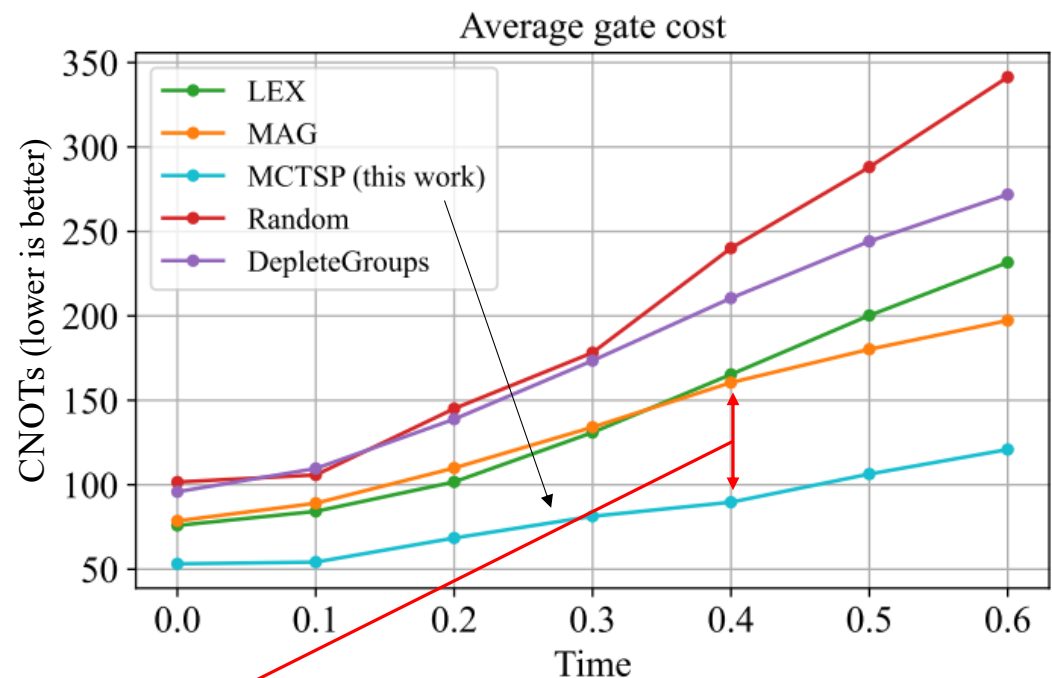
$$d_{\diamond}(\mathcal{E}, \mathcal{F}) := \epsilon = \|\mathcal{E} - \mathcal{F}\|_{\diamond}$$

$= \max_{\rho} \|(\mathcal{E} \otimes \mathbb{1})\rho - (\mathcal{F} \otimes \mathbb{1})\rho\|_1$


  
 quantum channels

## Simulation steps:

1. First, increase the Trotter number  $r$  until the error reaches a threshold  $\epsilon < 0.1$
2. Then, compute the number of CNOTs in the final quantum circuit.




Mitigating both algorithmic and physical errors results in 40% fewer CNOTs than LEX and MAG

# Real HW executions show further importance of NISQ codesign

## Experimental steps:

1. Prepare an initial state
2. Generate an HS circuit with a particular ordering
3. Measure the *Hellinger infidelity* ( $1 - H_D$ )

$$H_D(P, Q) = \frac{1}{\sqrt{2}} \|\sqrt{P} - \sqrt{Q}\|_2$$

  
 probability distributions

(lower is better)

Molecule	Diamond Dist. ( $\epsilon$ ) <i>lex, mag, mctsp</i>	2-qubit gates <i>lex, mag, mctsp</i>	Hellinger Inf. (%) <i>lex, mag, mctsp</i>
$C_2H_4$	2.9, 1.8e-3, 1.8e-3	55, 49, 41	75.9, 55.2, <b>53.8</b>
$Cl_2$	1.9, 1.4e-4, 1.4e-4	47, 53, 37	62.2, 56.6, <b>54.2</b>
$C_2H_2$	1.9, 1.3e-3, 1.3e-3	47, 53, 37	62.4, <b>57.2</b> , 57.7
$F_2$	<b>1.1, 3.0e-3, 3.0e-3</b>	<b>47, 53, 37</b>	<b>71.0, 56.7, 52.2</b>
$N_2$	1.7, 1.2e-3, 1.2e-3	47, 53, 37	61.9, 54.6, <b>54.1</b>
$O_2$	3.1, 4.1e-3, 4.1e-3	41, 41, 27	59.6, 59.7, <b>46.5</b>

Mitigating algorithmic errors is crucial even in the NISQ regime

# Takeaways: Application-specific co-design

- Cross-layer focus enable coordinated mitigation of both algorithmic and physical errors
- Compiler benefits remain relevant even in the fault-tolerant era
- Similar techniques can be adapted to other applications

Optimization of simultaneous measurement for variational quantum eigensolver applications

*P Gokhale, et al.*

*2020 IEEE International Conference on Quantum Computing and Engineering. Best Paper.*

$\mathcal{O}(N^3)$  Measurement Cost for Variational Quantum Eigensolver on Molecular Hamiltonians.

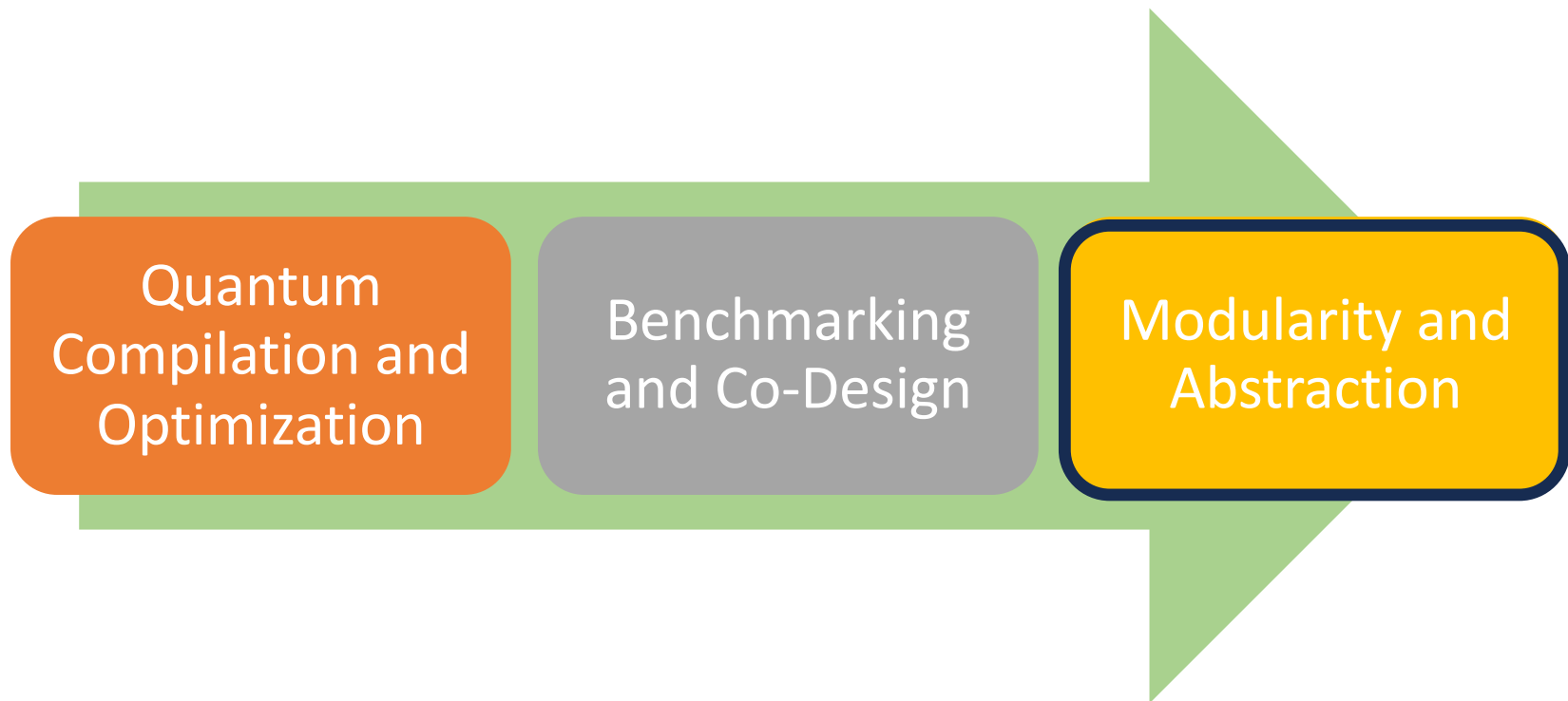
*P Gokhale, et al. IEEE Transactions on Quantum Engineering, 2020*

Divide and Conquer for Combinatorial Optimization and Distributed Quantum Computation

*T. Tomesh, et al. 2023 IEEE International Conference on Quantum Computing and Engineering*

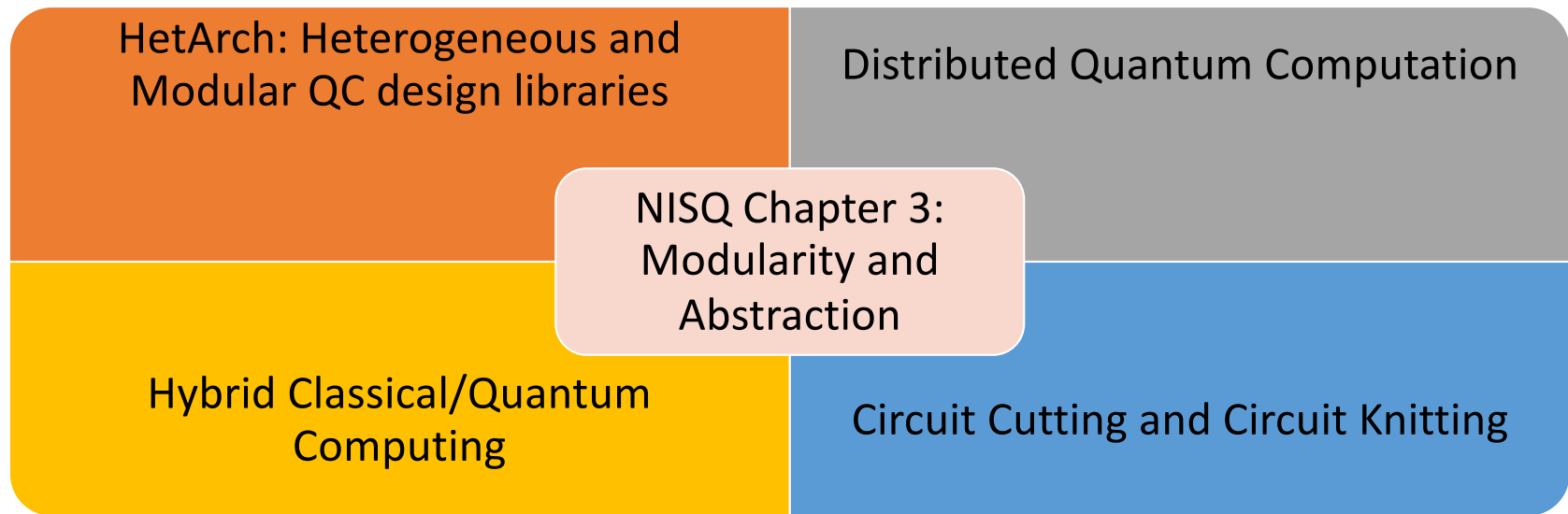
*(QCE), Bellevue, WA, USA, 2023*

# This talk: Some examples and opportunities from each “NISQ Chapter”



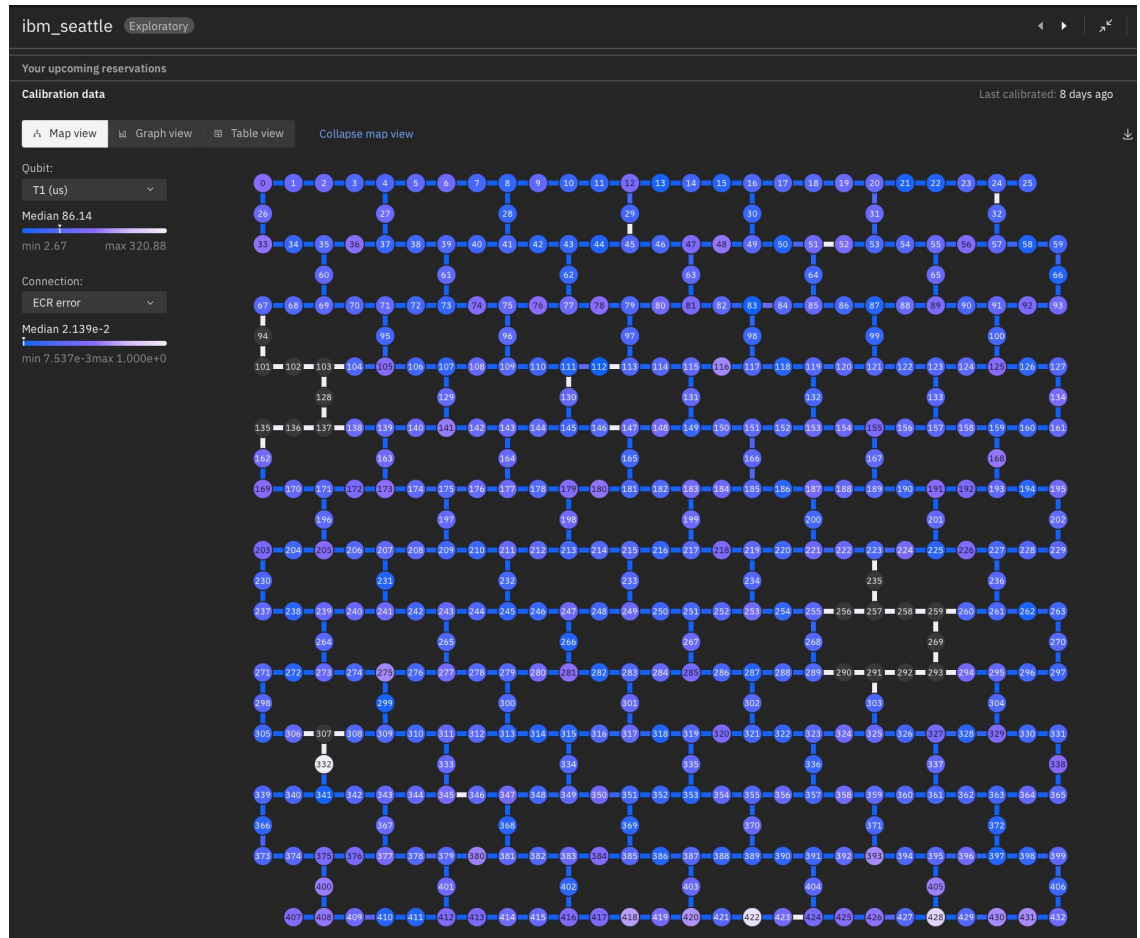


# NISQ Chapter 3: Towards Scale

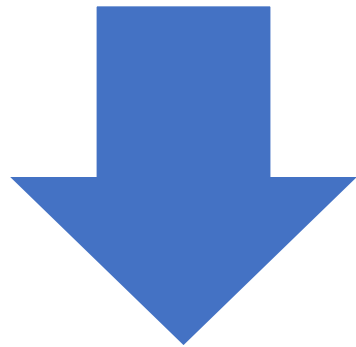


Sam Stein et al. MICRO '23 <https://dl.acm.org/doi/abs/10.1145/3613424.3614300>  
Wei Tang, et al. ASPLOS 2021. <https://dl.acm.org/doi/abs/10.1145/3445814.3446758>  
Wei Tang et al. ArXiv 2022. <https://arxiv.org/abs/2207.00933>

From Sea-of-qubit architectures to differentiated functionality



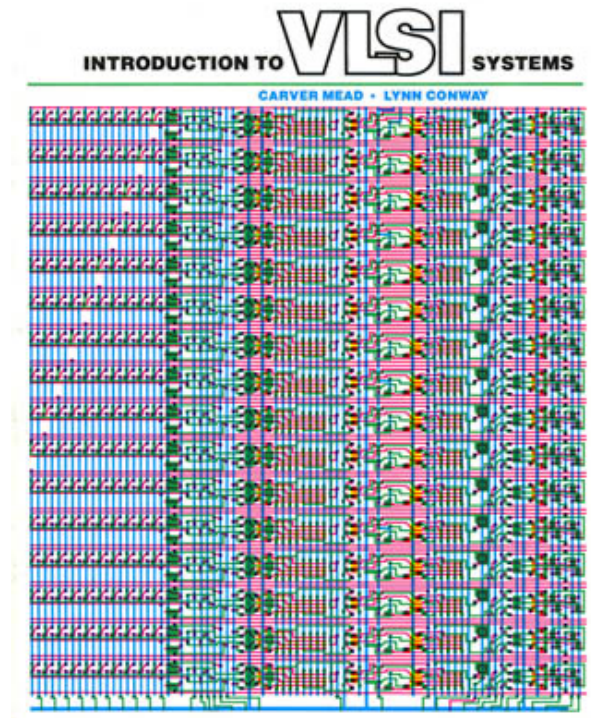
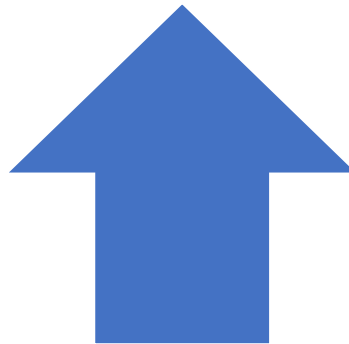
# Heterogeneity and Modularity -> Scale



Specialized modules  
can streamline  
mapping from  
application to  
hardware, but...

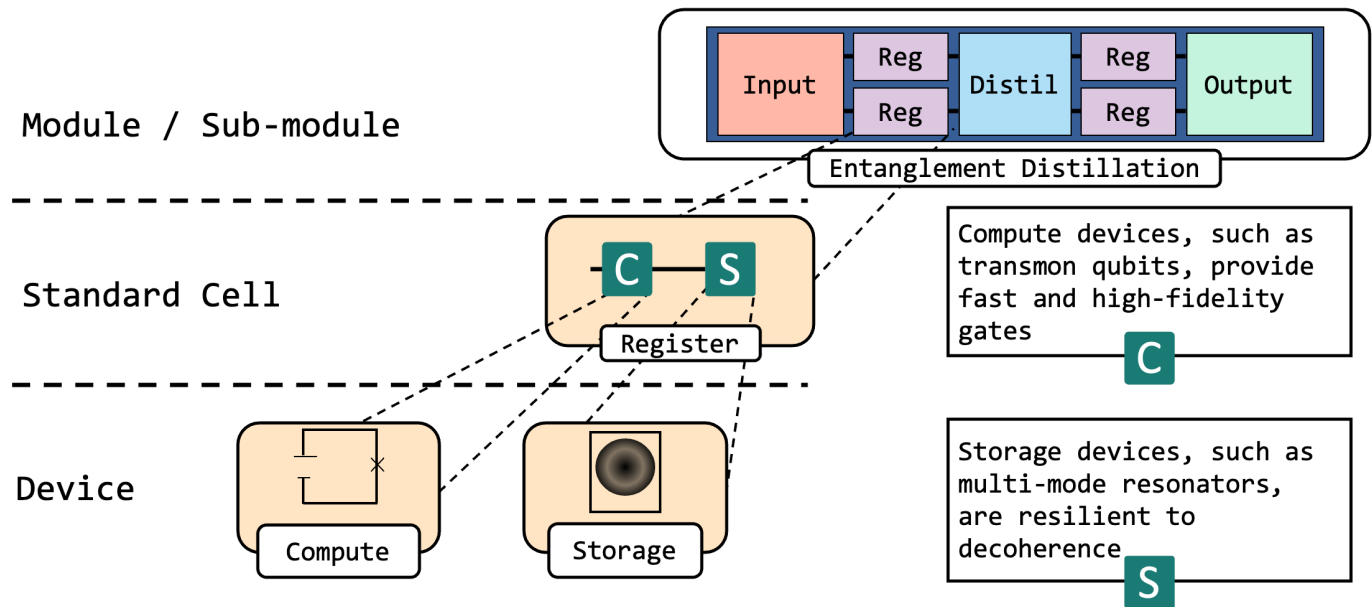


Unconstrained  
specialization  
can be worse  
than sea-of-  
qubits



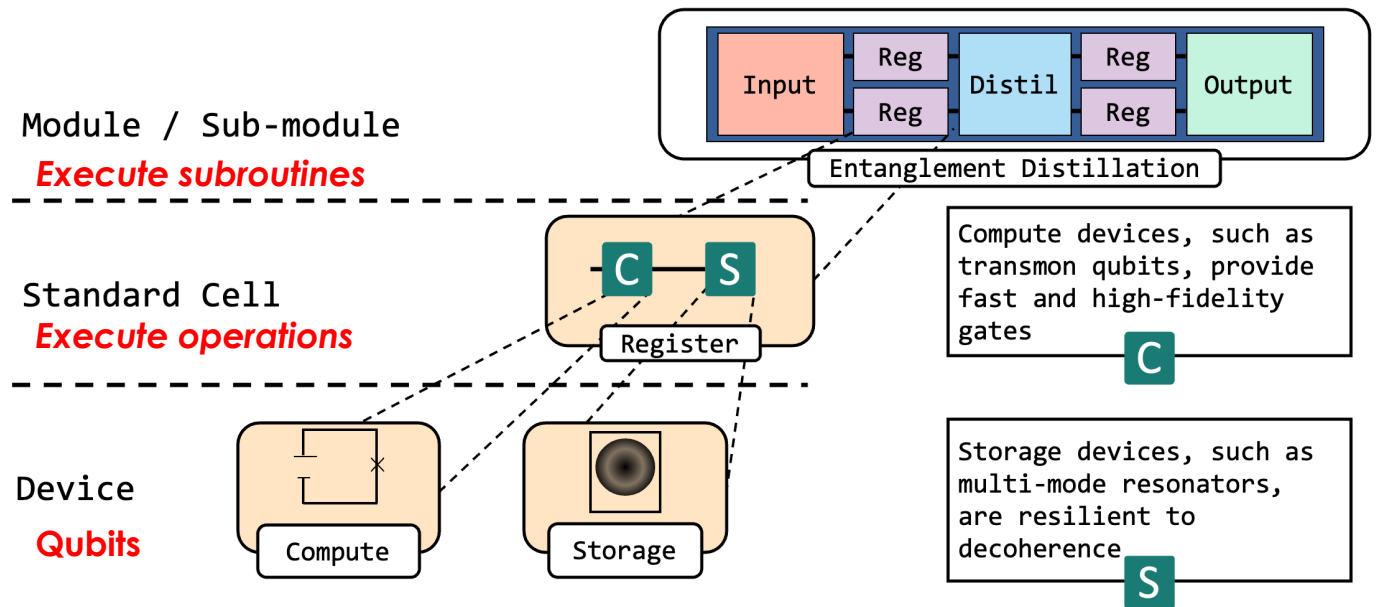
We've faced this before!  
Can we learn from the  
Mead & Conway approach?

# Our Solution: HetArch



- A **methodology and toolbox** for the systematic design and simulation of modular, heterogeneous quantum microarchitectures
  - **Hierarchical hardware synthesis method** to decompose high-level subroutines
  - **Design rules** to systematically design quantum standard cells
  - **Design space exploration framework** that efficiently simulates performance

# Our Solution: HetArch



- A **methodology and toolbox** for the systematic design and simulation of modular, heterogeneous quantum microarchitectures
  - **Hierarchical hardware synthesis method** to decompose high-level subroutines
  - **Design rules** to systematically design quantum standard cells
  - **Design space exploration framework** that efficiently simulates performance

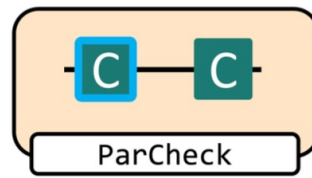
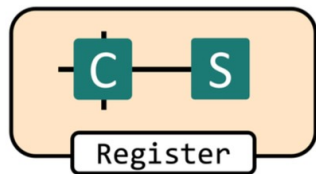
# HetArch Devices

Device	$T_1/T_2$	Readout time	Gate set	Gate error (time)	Connectivity	Control Overhead	Footprint	Notes
Fixed frequency qubit [15, 82]	$300\mu s / 550\mu s$	$1\mu s$	Arb. 1Q/2Q	$1e-3$ (100ns)	4	1 charge 1 readout	2 mm x 2 mm	e.g. Transmon
Flux tunable qubit [83, 84]	$800\mu s / 200\mu s$	$1\mu s$	Arb. 1Q/2Q	$1e-3$ (100ns)	4	1 charge 1 flux 1 readout	2 mm x 2 mm	e.g. Fluxonium
3D quantum memory [85, 86]	$25ms / 30ms$	N/A	SWAP	$1e-2$ ( $1\mu s$ )	1	N/A	50 mm x 0.5 mm x 1 mm	Requires 2D/3D integration
3D multimode resonator (10 modes) [19]	$2ms / 2.5ms$	N/A	SWAP	$1e-2$ (400ns)	1	N/A	100 mm x 100 mm x 10 mm	Requires 2D/3D integration
Future on-chip multimode resonator [19, 20, 87]	$1ms / 1ms$	N/A	SWAP	$1e-2$ (100ns)	1	N/A	5 mm x 5 mm	No demonstration

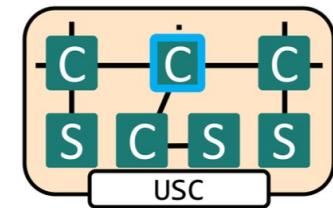
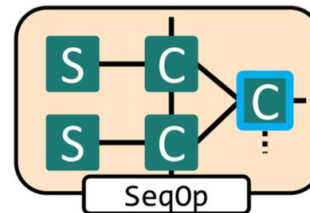
- Compute desires high connectivity and fast, high-fidelity gates
  - Planar transmon
- Storage desires long coherence times and multi-qubit capacity
  - Multimode resonator
- HetArch's Devices Abstraction:
  - Abstracts away actual implementation technology
  - But retains basic, simplifying distinction between “compute” and “storage” functionality
  - Aimed at central tradeoff between long coherence time and high connectivity

# Standard Cells & Design Rules

- HetArch **Standard Cells** = Functional units composed of devices, optimized for particular operation(s), complying by **architectural design rules independent of device implementation technology**
- Example Design Rules
  - Compute devices should be connected to at most 4 other devices.
  - Storage devices should be connected to exactly 1 compute device to maximize coherence.
  - ...
- Standard Cells are pre-characterized by detailed density matrix simulation → time, fidelity of operation



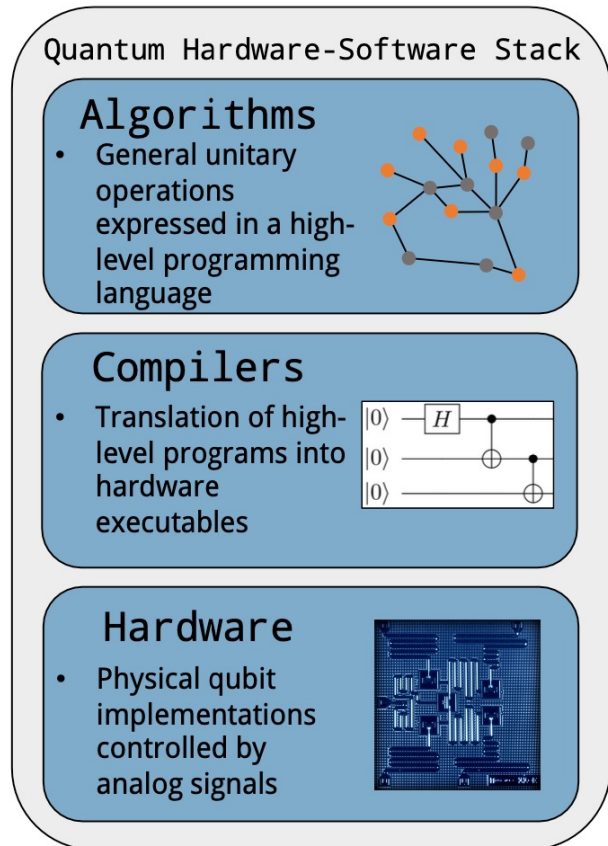
"Parity Check Cell"



"Universal Stabilizer Cell"

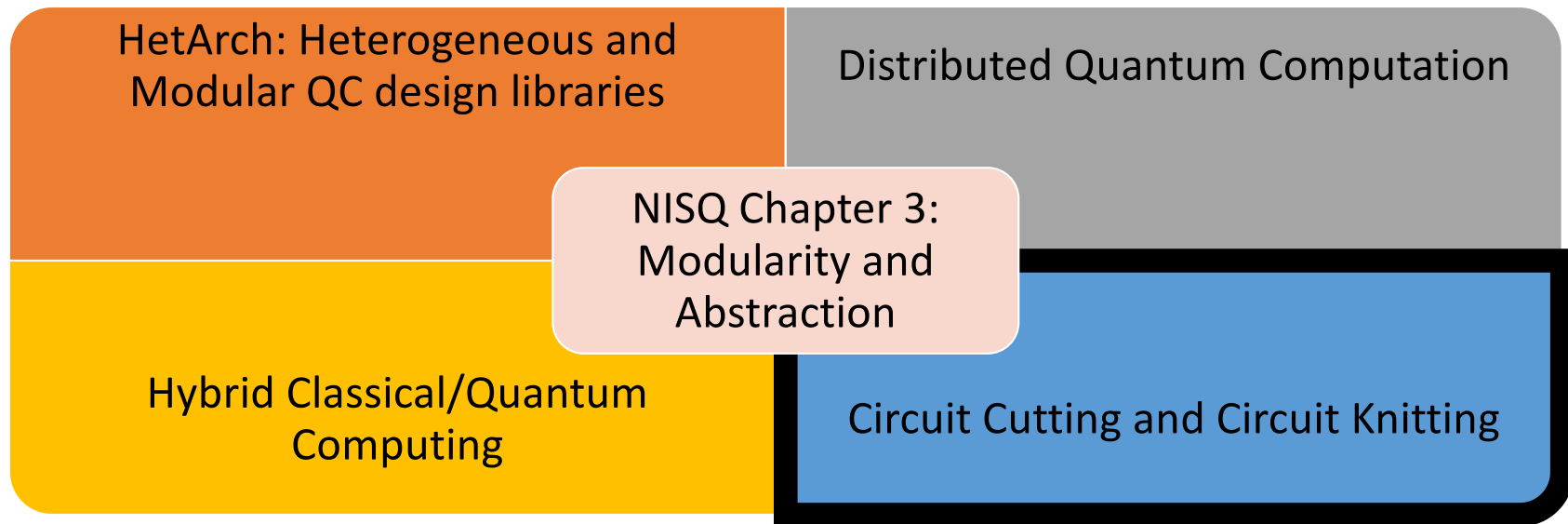
# More in the Paper

- Use multi-level modular abstractions to project and optimize behavior of complex units without requiring full device-level simulation
- Demonstrate on Code Teleportation and other examples





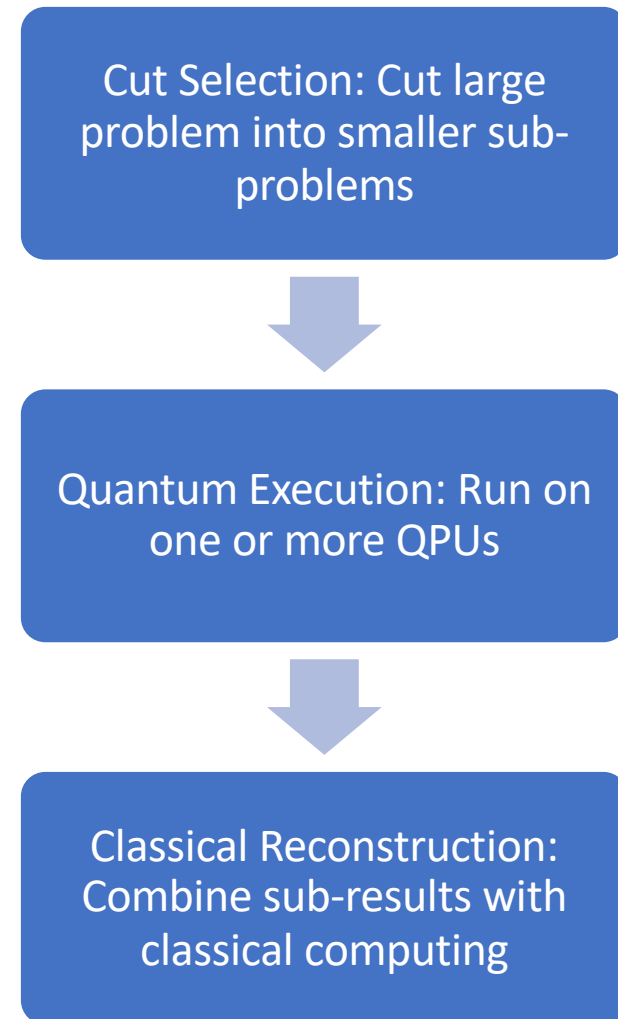
# NISQ Chapter 3: Towards Scale



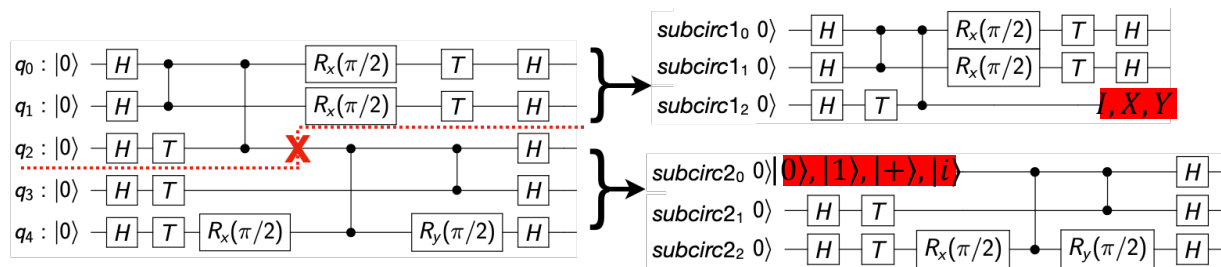
Sam Stein et al. MICRO '23 <https://dl.acm.org/doi/abs/10.1145/3613424.3614300>  
Wei Tang, et al. ASPLOS 2021. <https://dl.acm.org/doi/abs/10.1145/3445814.3446758>  
Wei Tang et al. ArXiv 2022. <https://arxiv.org/abs/2207.00933>

# Modularity and Scale: Cut QC Circuits to fit onto NISQ Platforms

- Goal: Make use of NISQ devices despite the noise and capacity challenges.
- Approach: Cut QC circuits into sub-parts that fit, and sew together results classically afterwards
- Challenge: Classical reconstruction is time and resource-intensive!
- Our Work: First practical circuit cutting toolchain that beats classical simulation in runtime, and NISQ in fidelity and circuit sizes.



# CutQC: Combining Classical and Quantum Computation to run QC algorithms at larger scale



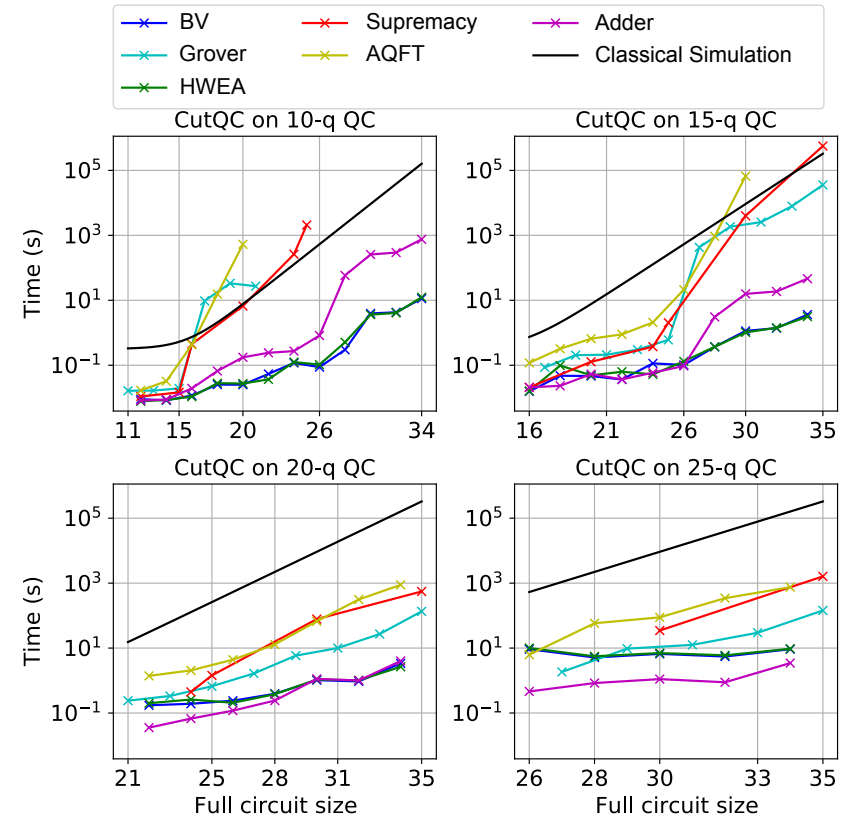
Example: Cut one edge to split a 5-qubit circuit into two smaller (3-qubit each) subcircuits.

- CutQC: Cut + Quantum Processing + Classical Processing
  1. **Select Cut Points:** Mixed Integer Programming (MIP) automatically locates efficient cut points in large QC algorithm circuits and splits into small subcircuits runnable on NISQ platforms.
  2. **Run QC subcircuits:** Small NISQ platforms evaluate the subcircuits.
  3. **Reconstruct:** Classical reconstruction approaches sew subcircuits back together, produce the uncut output in full definition (FD) or dynamically adjust the definition (DD).
- Our work is the first practical circuit cutting toolchain that beats classical simulation in runtime, and NISQ in fidelity and circuit sizes.

*Wei Tang, Teague Tomesh, Martin Suchara, Jeffrey Larson, Margaret Martonosi, CutQC: Using Small Quantum Computers for Large Quantum Circuit Evaluations, ASPLOS 2021*

# Result: Runtime Improvement vs. Simulation

- 60X-8600X speedup over the classical simulation baseline (IBM Qiskit)
- Insights:
  - Densely connected circuits are harder to cut.
  - Larger quantum circuits generally require more postprocessing.
  - Having larger quantum computers generally improves the runtime but has diminishing returns. E.g. 5\*7 supremacy admits the same cuts on 20-, and 25-q QC.



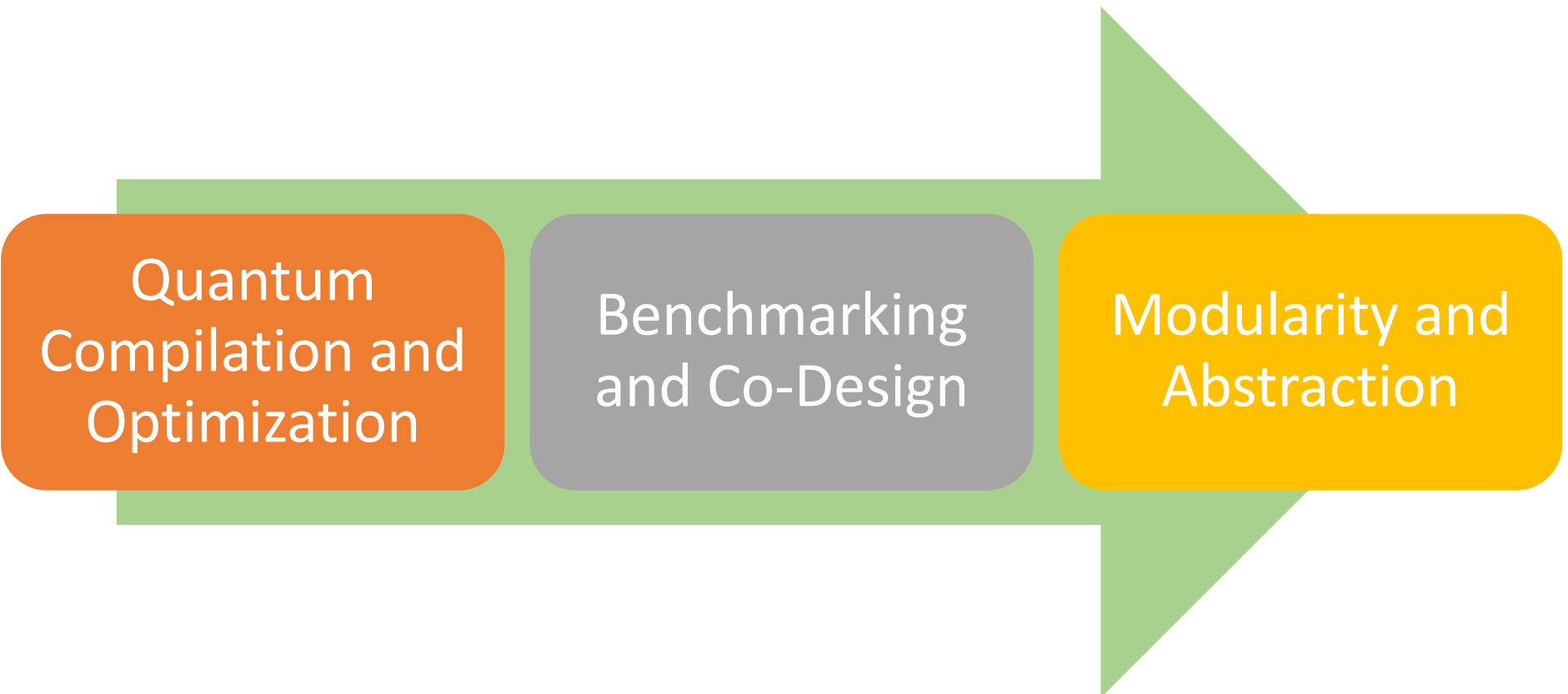
# To Scale Further...

- Dynamic definition: Zoom in on states of particular interest, to reduce reconstruction required
- Parallel GPUs: See our poster here!
- Tensor Optimizations: Exploit modern GPUs and tensor contraction optimizations to further reduce reconstruction complexity and time

Find open source Github, full paper, full talk here:

<https://www.wtang.page>

# What's next?



Quantum  
Compilation and  
Optimization

Benchmarking  
and Co-Design

Modularity and  
Abstraction

# Quantum Systems Today

## ~1950's Classical Computing

Algorithms

Assembly Language

Vacuum Tubes, Relay Circuits

## Today's Classical Computing

Algorithms

High-Level Languages

Compiler OS

Architecture

Modular hardware blocks:  
Gates, registers

VLSI Circuits

Semiconductor transistors

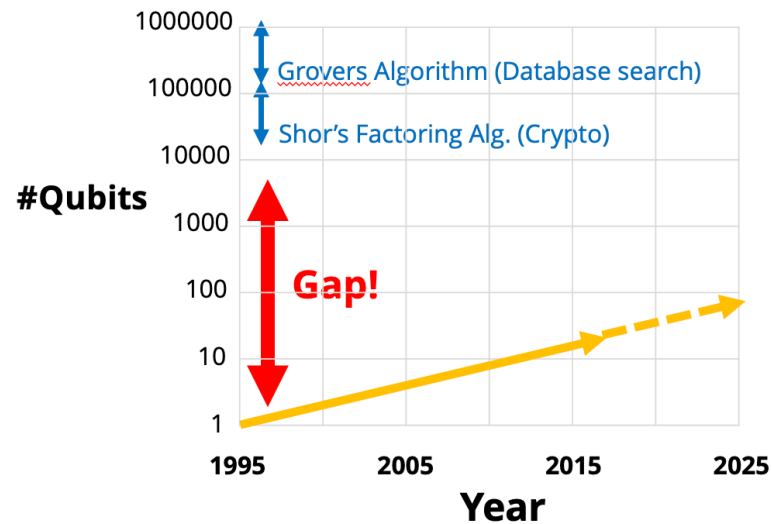
## Quantum Systems

Algorithms

High-level QC Languages.  
Compilers.  
Optimization.  
Error Correcting Codes  
Orchestrate classical gate control,  
Orchestrate qubit motion and manipulation.

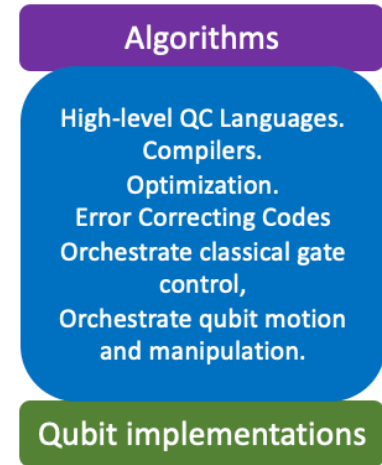
Qubit implementations

# Mind the Gaps!



Capability Gap:  
Algorithms to  
Devices

## Quantum Systems



Topical Gap: Between  
Algorithms and  
Devices is a world of  
QSE research



# With thanks to my amazing co-authors

- **Wei Tang\***
  - Final Year Princeton CS PhD student
  - Circuit cutting and optimizations
  - Impact into IBM and Amazon systems
- Teague Tomesh, now at Infleqtion
  - Princeton CS PhD 2023
  - Quantum algorithms and benchmarking for NISQ advantage
  - Now at Infleqtion
- Prakash Murali, now at University of Cambridge, UK
  - Princeton CS PhD 2021
  - Sequence of papers on gate selection and optimizations across superconducting and trapped ion implementations
  - ACM SIGARCH/IEEE CS TCCA Outstanding Dissertation Award (2022)
- Ali Javadi-Abhari, now at IBM Q
  - Princeton EE PhD 2017
  - Lead on Scaffold QC Compiler/PL work + resource estimation and optimization
- Esin Tureci
  - Associate Research Scholar at Princeton University
- **Chuck Garcia, Summer Undergraduate Research Visitor from University of Texas, Austin\***
- **Ellie Vogel, Summer Undergraduate Research Visitor from Duke University\***

\* See our poster next!