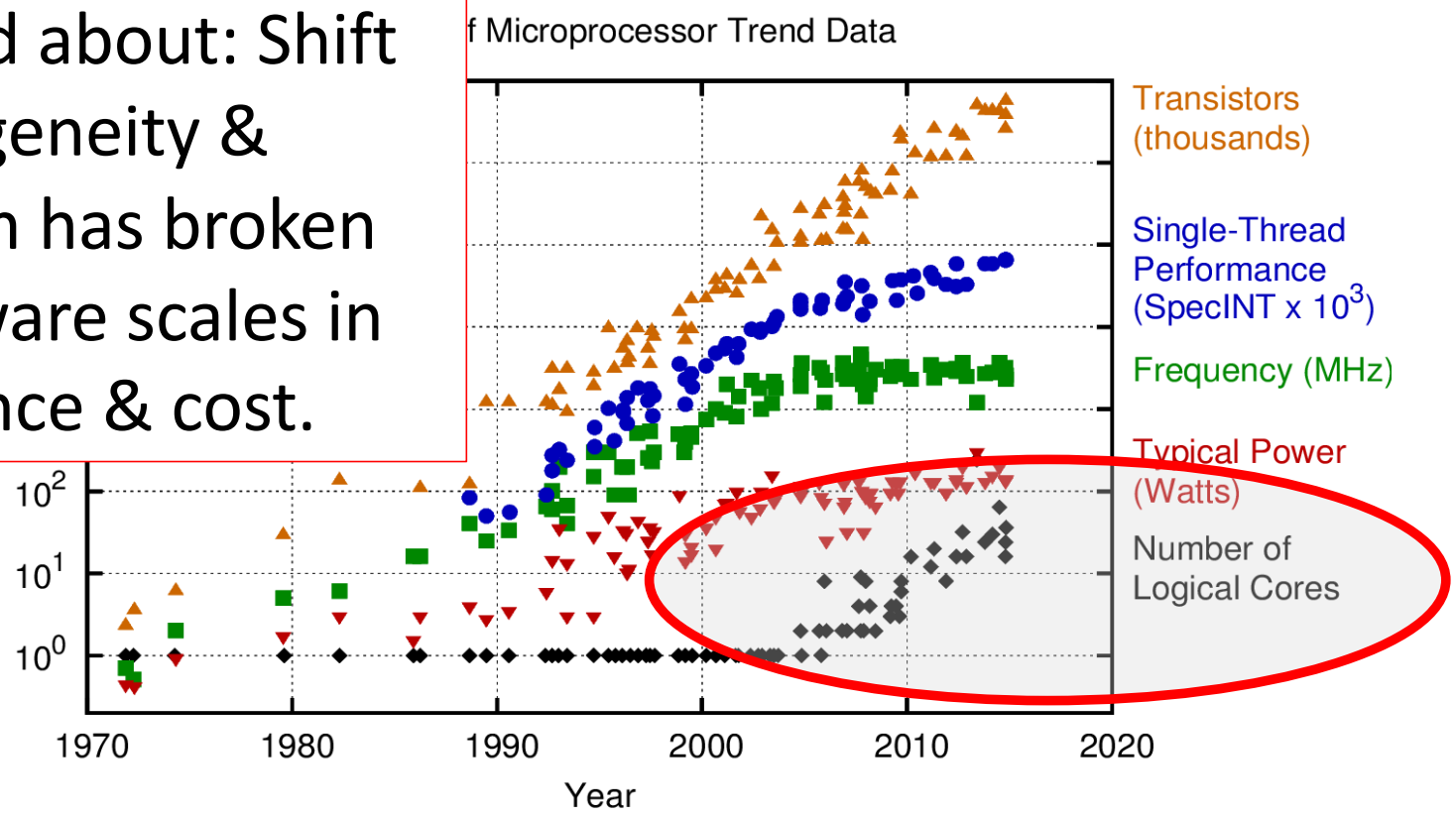
Seismic Shifts: Challenges and Opportunities in the 'Post-ISA' Era of Computer Systems Design

Prof. Margaret Martonosi
H. T. Adams '35 Prof. of Computer Science
Princeton University

End of Decades of Moore's Law scaling

Less talked about: Shift to heterogeneity & parallelism has broken how software scales in performance & cost.



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

1964:
While Moore's Law was being sketched,
"Computer Architecture" was coined…

= The value of HW/SW abstraction…

G. M. Amdahl
G. A. Blaauw
F. P. Brooks, Jr.,

Architecture of the IBM System/360

Abstract: The architecture* of the newly announced IBM System/360 features four innovations:

1. An approach to storage which permits and exploits very large capacities, hierarchies of speeds, read-only storage for microprogram control, flexible storage protection, and simple program relocation.

2. An input/output system offering new degrees of concurrent operation, compatible channel operation, data rates approaching 5,000,000 characters/second, integrated design of hardware and software, a new low-cost, multiple-channel package sharing main-frame hardware, new provisions for device status information, and a standard channel interface between central processing unit and input/output devices.

3. A truly general-purpose machine organization offering new supervisory facilities, powerful logical pro-

a line of six models having a per-

tionale for the main features of the
ompatibility among central process-
scientific, real-time, and logical in-
he Appendices.

*The term *architecture* is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior, as distinct from the organization of the data flow and controls, the logical design, and the physical implementation.

Introduction

The design philosophies of the new general-purpose machine organization for the IBM System/360 are discussed in this paper.† In addition to showing the architecture* of the new family of data processing systems, we point out the various engineering problems encountered in attempts to make the system design compatible, at the program bit
r large and small models. The compatibility was
d not only to models of any size but also to their
applications—scientific, commercial, real-time, and

The section that follows describes the objectives of the new system design, i.e., that it serve as a base for new technologies and applications, that it be general-purpose, efficient, and strictly program compatible in all models. The remainder of the paper is devoted to the design problems faced, the alternatives considered, and the decisions made for data format, data and instruction codes, storage assignments, and input/output controls.

Design objectives

The new architecture builds upon but differs from the designs that have gradually evolved since 1950. The evolution of the computer had included, besides major technological improvements, several important systems concepts and developments:

term *architecture* is used here to describe the attributes of a
s seen by the programmer, i.e., the conceptual structure and
l behavior, as distinct from the organization of the data flow
rols, the logical design, and the physical implementation.
tional details concerning the architecture, engineering design,
sing, and application of the IBM System/360 will appear in a
articles in the *IBM Systems Journal*.

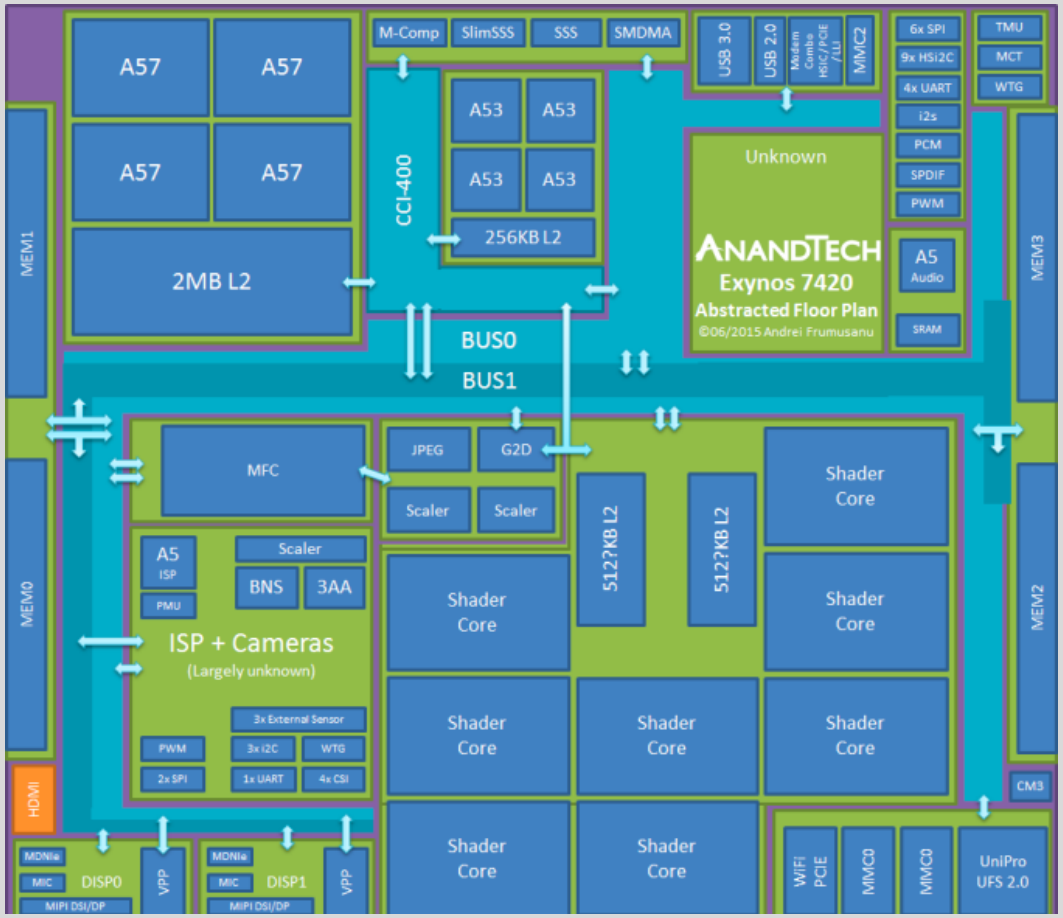
2022:
We are here

Massive Heterogeneity:

- Dozens of ISAs on chip + Accelerators
- Memory, Data Heterogeneity
- Memory Consistency Models

Systems heterogeneity too:

- Thousands of distinct Android, IoT devices…



https://www.anandtech.com/show/9330/exynos-7420-deep-dive/2

Seismic Shift: Entering a Post-ISA World

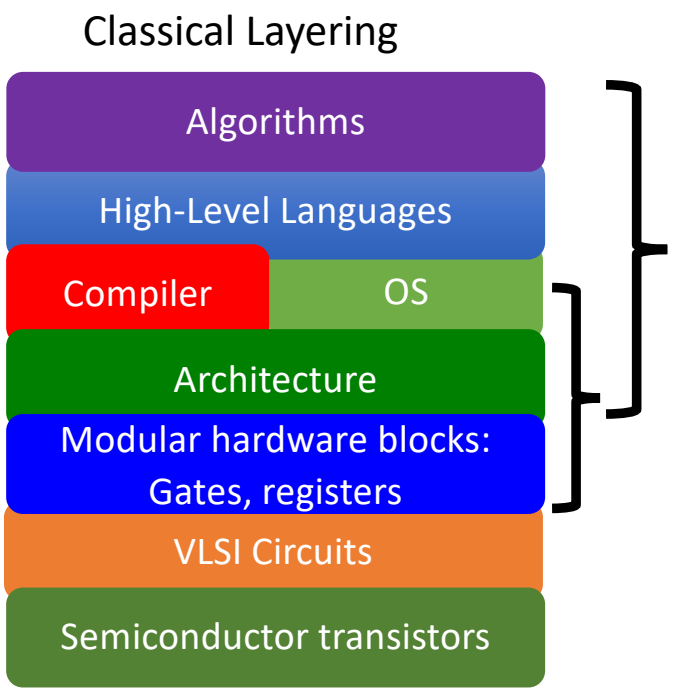
ISAs still useful, but little/no relevance as abstraction layer

- Apple A8 and beyond: >50% of chip area is accelerators that have no ISA.
- NVIDIA PTX vs. SASS: ISA hidden under other layers.

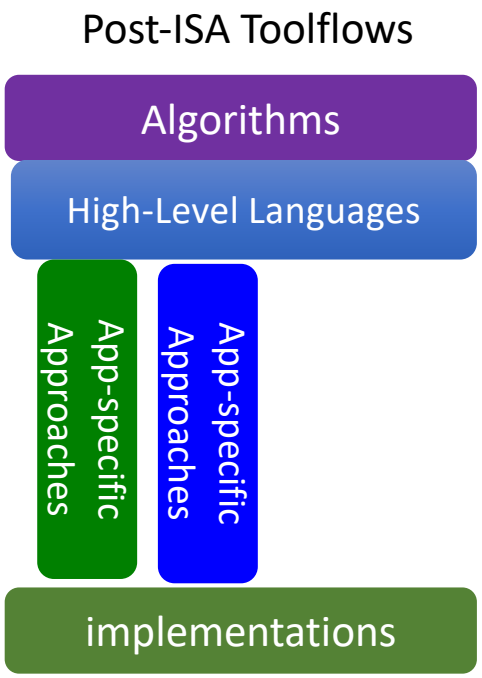
Questions for the Post-Moore/Post-ISA Future:

- How to program these highly heterogeneous systems? How to manage the complexity of fast-changing hardware and software?
- And what technologies come next?

End-of-Moore Systems: Rethinking Full-Stack Approaches



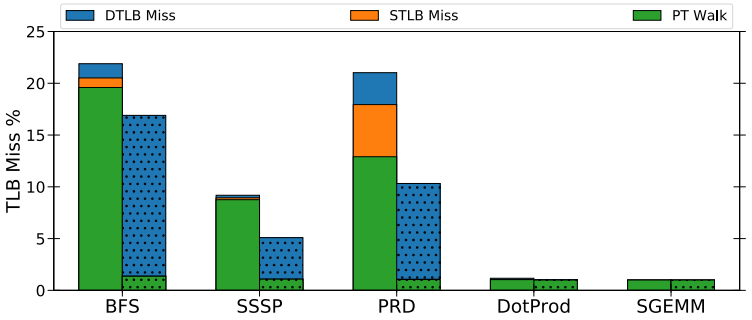
Rethinking layers, interfaces, and abstractions



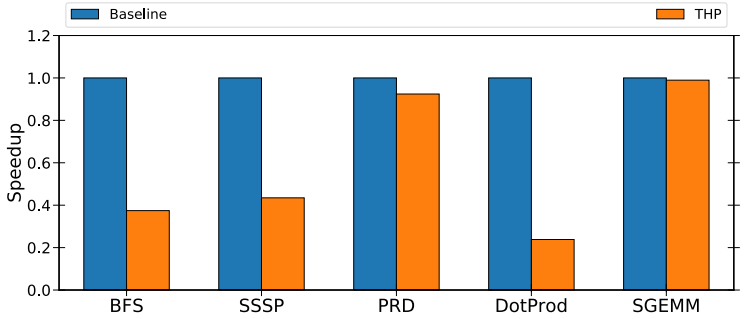
Tailoring for particular application domains?

Example 1: OS Page Size Management Tailored For Graph Analytics

- Graph analytics have high TLB miss rates that cause address translation overheads
- Huge pages (2MB on x86) can alleviate such overheads with increased TLB reach
- Modern OS policies greedily (over)allocate huge pages due to lack of app knowledge
- Need: OS techniques to intelligently manage huge pages tailored for graph analytics



TLB miss rates without (left) and with (right) THPs; graph analytics have high miss rates compared to dense apps



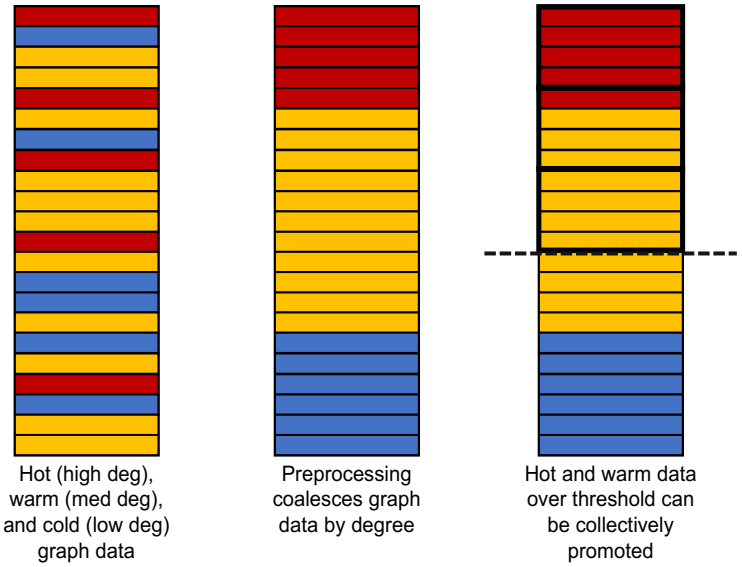
Runtime speedups without (left) and with (right) THPs; Linux THP causes slowdown when memory is constrained

Intelligent Page Size Management

- Objective: utilize huge pages in an intelligent, application-aware manner where they will bring most benefit (lower TLB miss rate)

  - Graph-tailored huge page management:
    - **Preprocess** dataset [1] to coalesce hot pages worth of (high-degree vertex) data
    - Dynamically **promote** hot data based on amount of memory fragmentation

  - Promote irregularly accessed data that has highest access frequency



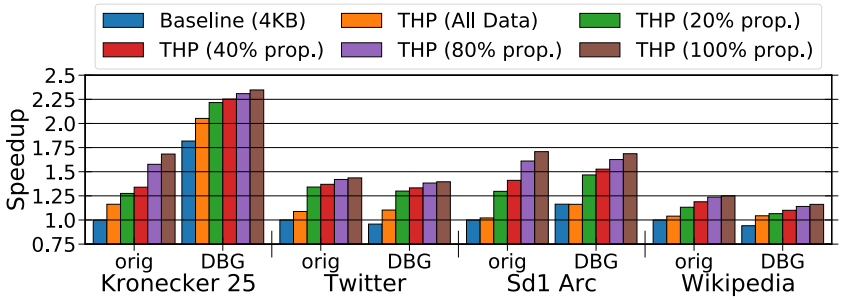
[1] Faldu et al. *IISWC '19*

Results and Takeaways

- Leveraging application knowledge for huge page allocation and placement best optimizes performance improvements from huge pages in real systems

- For graph analytics, utilize huge pages selectively for hottest percentage of **property array** (frequently and irregularly accessed data)

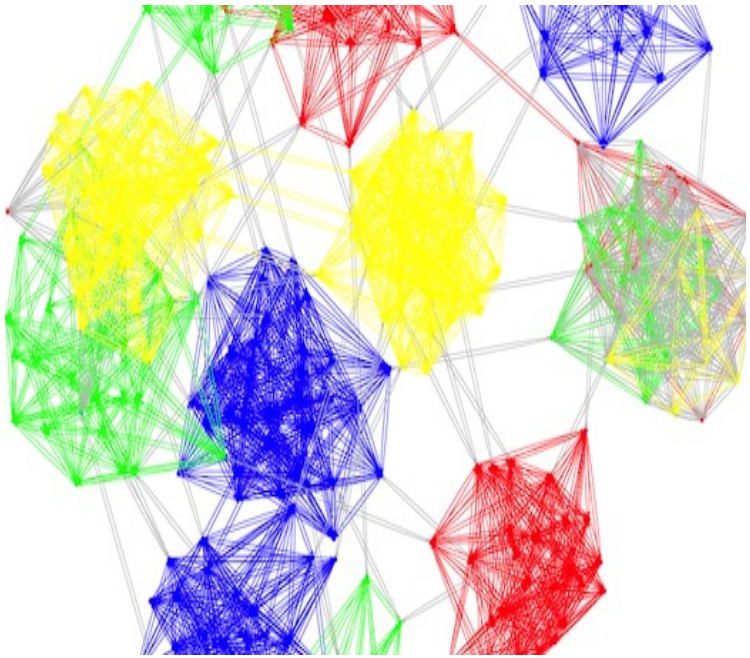
  - **1.26-1.57x** speedup over 4KB pages
    - **77.3-96.3%** of ideal THP performance
    - Requires only **0.58-2.52%** of application footprint to be backed by huge pages



Runtime speedups comparing THPs applied system-wide vs. selectively to percentage of preprocessed TLB-sensitive prop. array

Example 2: Hardware and Programming Models for Sparse/Graph Applications

- Graph analytics and memory bottlenecks
- Challenges:
  - Little compute per loaded cache line
  - Little data reuse
  - >50% of accesses go to main memory
  - >95% of total energy spent on memory operations
- Prior work mitigates the memory latency, but bandwidth and synchronization remains a problem when scaling to high core counts



Dalorex: A Data-Local Program Execution and Architecture for Memory-bound Applications

- **Data local program execution model**:
  - Data arrays are distributed in equal chunks across tiles
  - Only one core has access to a given data (no copies)

Edge-sized array tuple: chunked among all tiles

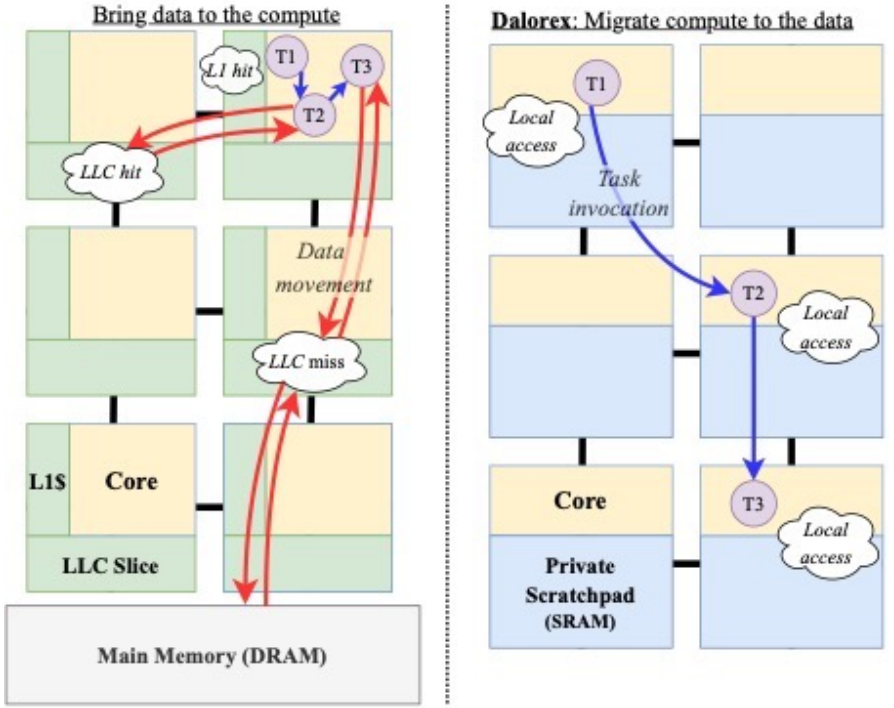
Tile 1	Tile 2	Tile 3	Tile 4

Vertex-sized array tuple

Tile 1	Tile 2	Tile 3	Tile 4

- **Program is sliced at each pointer indirection** resulting in multiple program slices (tasks)
  - All tiles are homogeneous, they can perform any task
  - A task is performed in the core where data is local
  - Tasks can invoke other tasks by placing the tasks parameters in the on-chip network.
  - The first parameter is an index to the distributed array

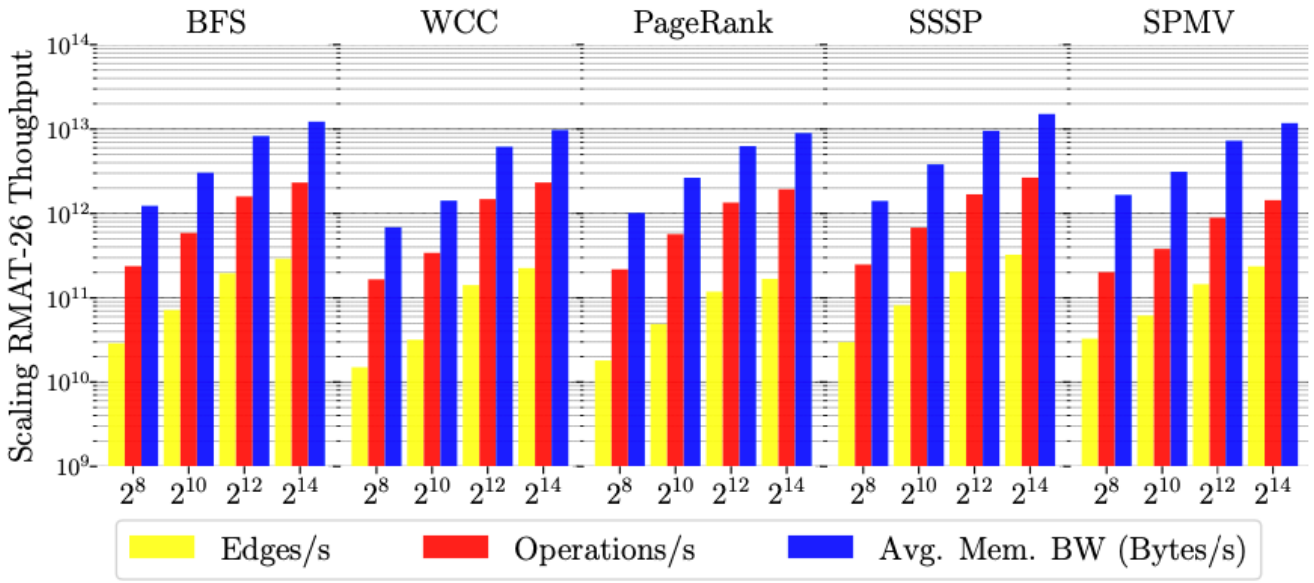
- **Dalorex** provides a new programming model and architecture to support task invocations natively
  - Plus optimizations in task scheduling and work-balance!



A **tile** in Dalorex is composed of a local SRAM memory, a stripdown sw-programmable core (no cache) and a route

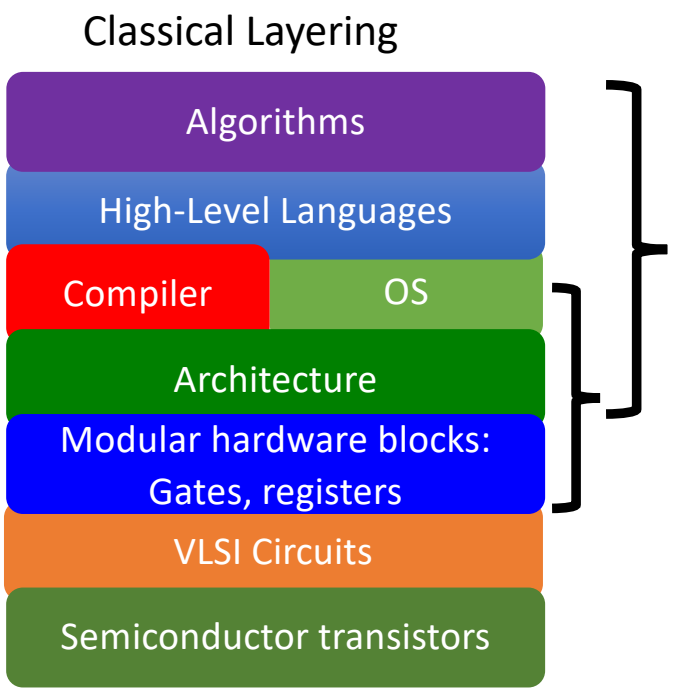
Orenes-Vera, Tureci, Wentzlaff, Martonosi. Dalorex: A Data-Local Program Execution and Architecture for Memory-Bound Applications". ArXiv July 2022

Scalability achieved via full-stack tailoring and Data-local compute

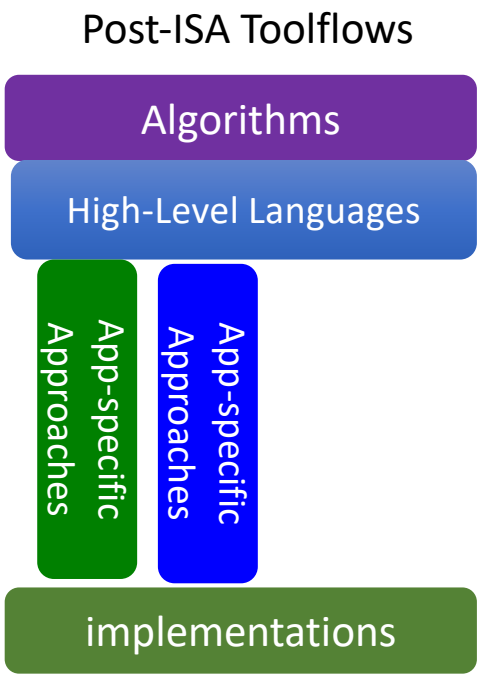


- Scaling up to 16.000 processing elements (128 × 128 tiles) reaches over 2 tera-ops/s, using around 10 terabytes/s of memory BW
- 2 orders of magnitude performance/energy improvements over prior work
- Sparse apps reach limit of compute and network resources
- Parallelism exposed splitting iterations at each pointer indirection

End-of-Moore Systems: Rethinking Full-Stack Approaches

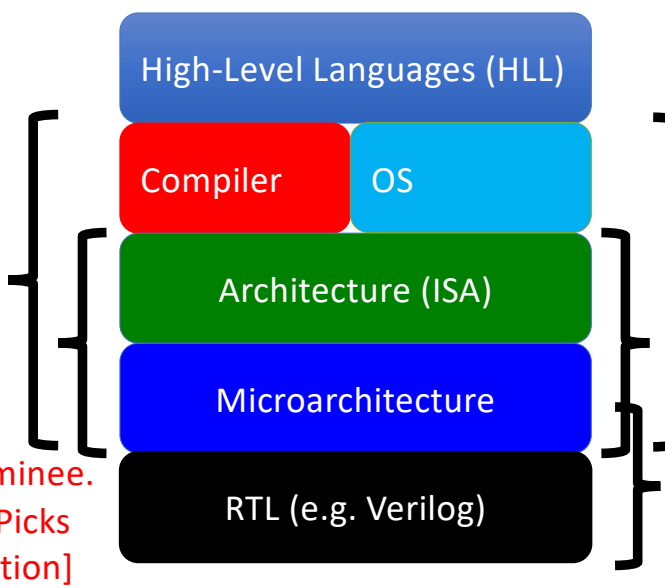


Rethinking layers, interfaces, and abstractions



Tailoring for particular application domains?

Example 3: The Check Suite: An Ecosystem of Tools For Early-Stage Verification and Example Synthesis



CheckMate [Micro '18] [IEEE Micro Top Picks]

PipeProof [Micro '18] [Best Paper Nominee. IEEE Micro Top Picks Honorable Mention]

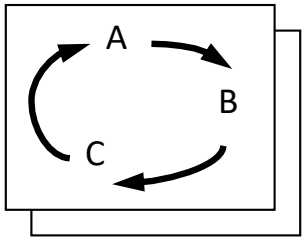
TriCheck [ASPLOS '17] [IEEE MICRO Top Picks]

COATCheck [ASPLOS '16] [IEEE MICRO Top Picks]

PipeCheck [Micro '14] [IEEE MICRO Top Picks]

CCICheck [Micro '15] [Nominated for Best Paper Award]

RTLCheck [Micro '17] [IEEE MICRO Top Picks Honorable Mention]



Our Approach

- Axiomatic specifications -> Happens-before graphs
- Check Happens-Before Graphs via Efficient SMT solvers
  - Cyclic => A->B->C->A… Can't happen
  - Acyclic => Scenario is observable

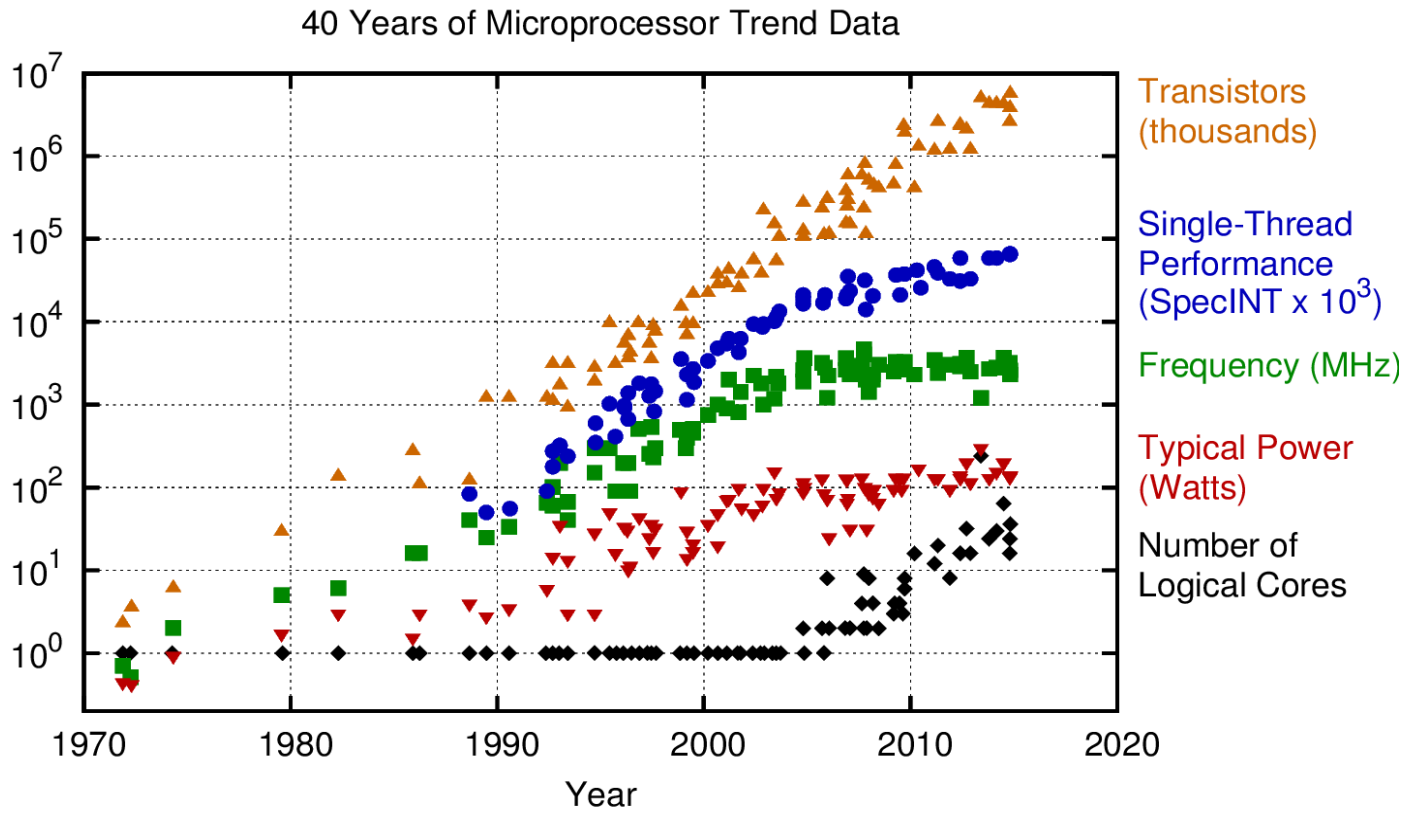
For more info: check.cs.Princeton.edu

This Talk: A whirlwind tour of…

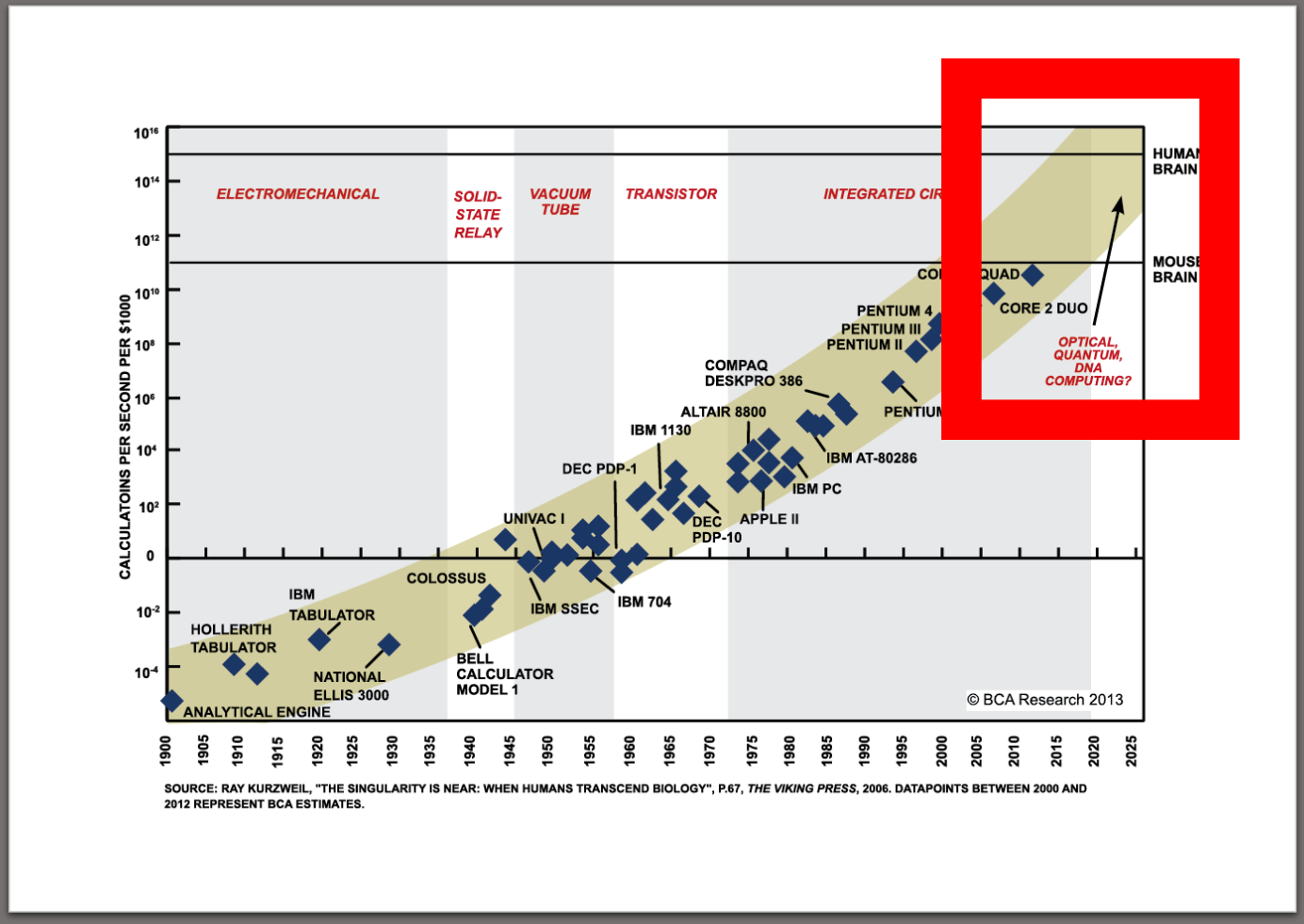
Questions for the Post-Moore/Post-ISA Future:

- How to program these highly heterogeneous systems? How to manage the complexity of fast-changing hardware and software?
- And what technologies come next?

Decades of Moore's Law scaling



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Performance Scaling over the Decades

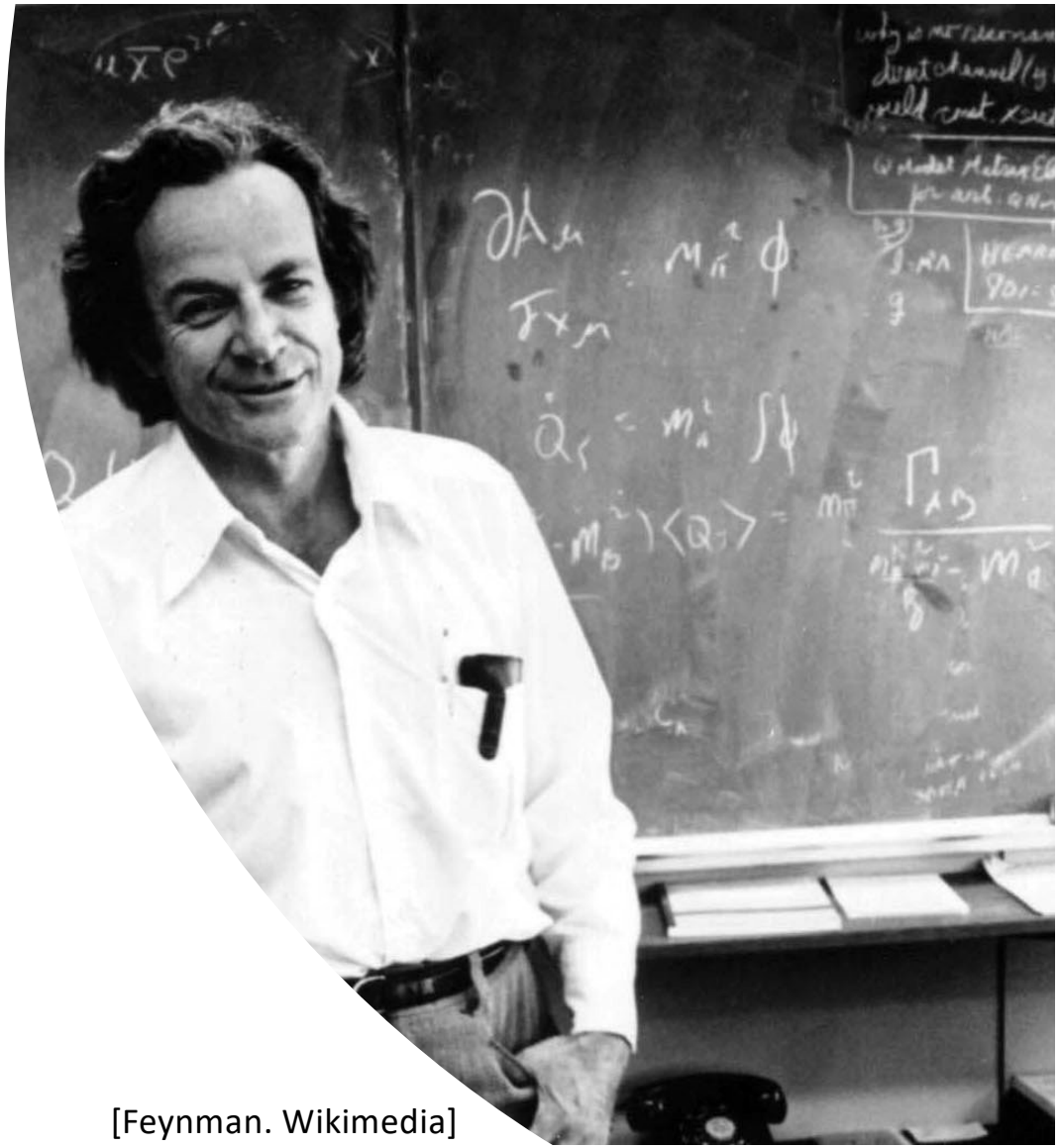
Disruptive Moment: What is next?

Feynman: Simulating the Physical World

”The full description of quantum mechanics for a large system with R particles … has too many variables, it cannot be simulated with a normal computer with a number of elements proportional to R or proportional to N...

And therefore, the problem is, how can we simulate the quantum mechanics? …. We can give up on our rule about what the computer was, we can say:

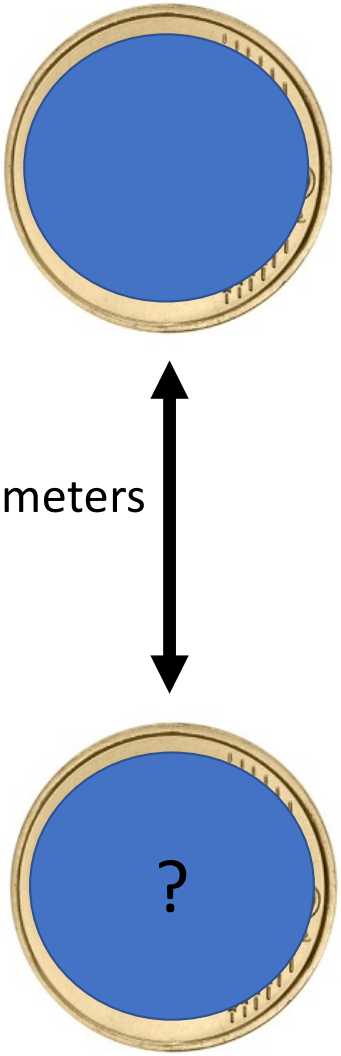
Let the computer itself be built of quantum mechanical elements which obey quantum mechanical laws. “

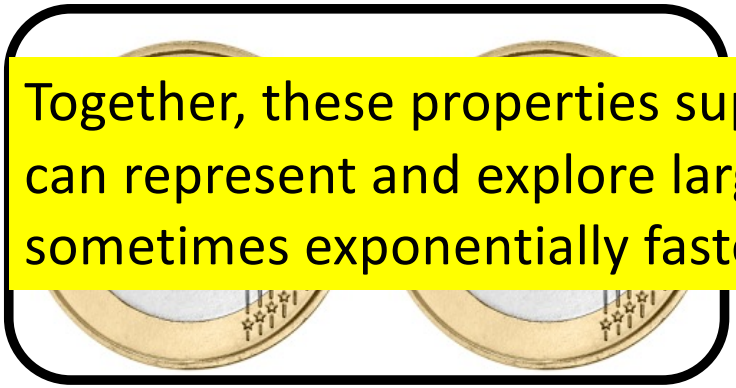


[Feynman. Wikimedia]

Key Enablers of Quantum Speedups

- Superposition of states within a quantum bit (qubit)
  - Large and probabilistic representation of possibilities
- Entanglement of states between qubits
  - Correlations between qubit states, once entangled.
  - Einstein: ”Spooky action at a distance”
  - Entangled correlations can be maintained even if qubits are physically distant (1400 kms distance record set last year)

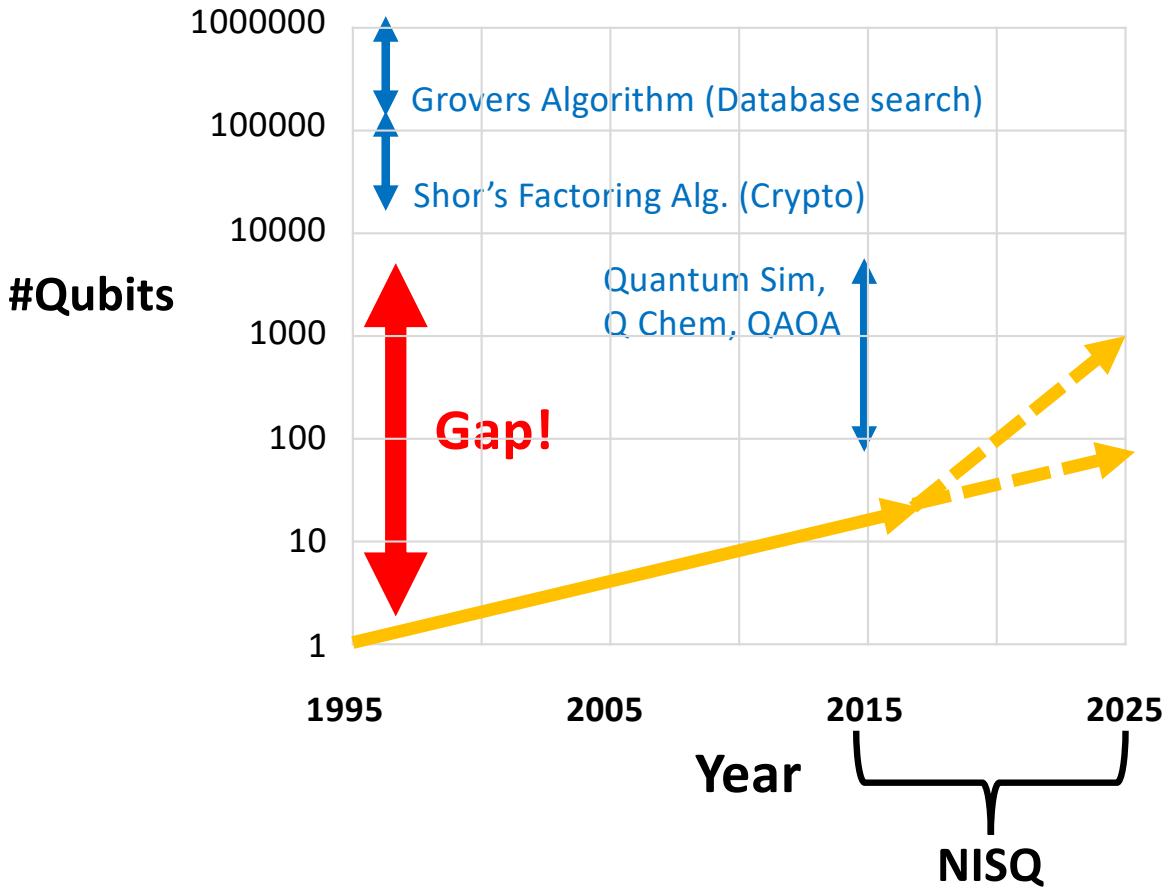




Together, these properties support novel algorithms that can represent and explore large state spaces faster—sometimes exponentially faster—than classical computing.

Entangle. (~ QC Gate operation)

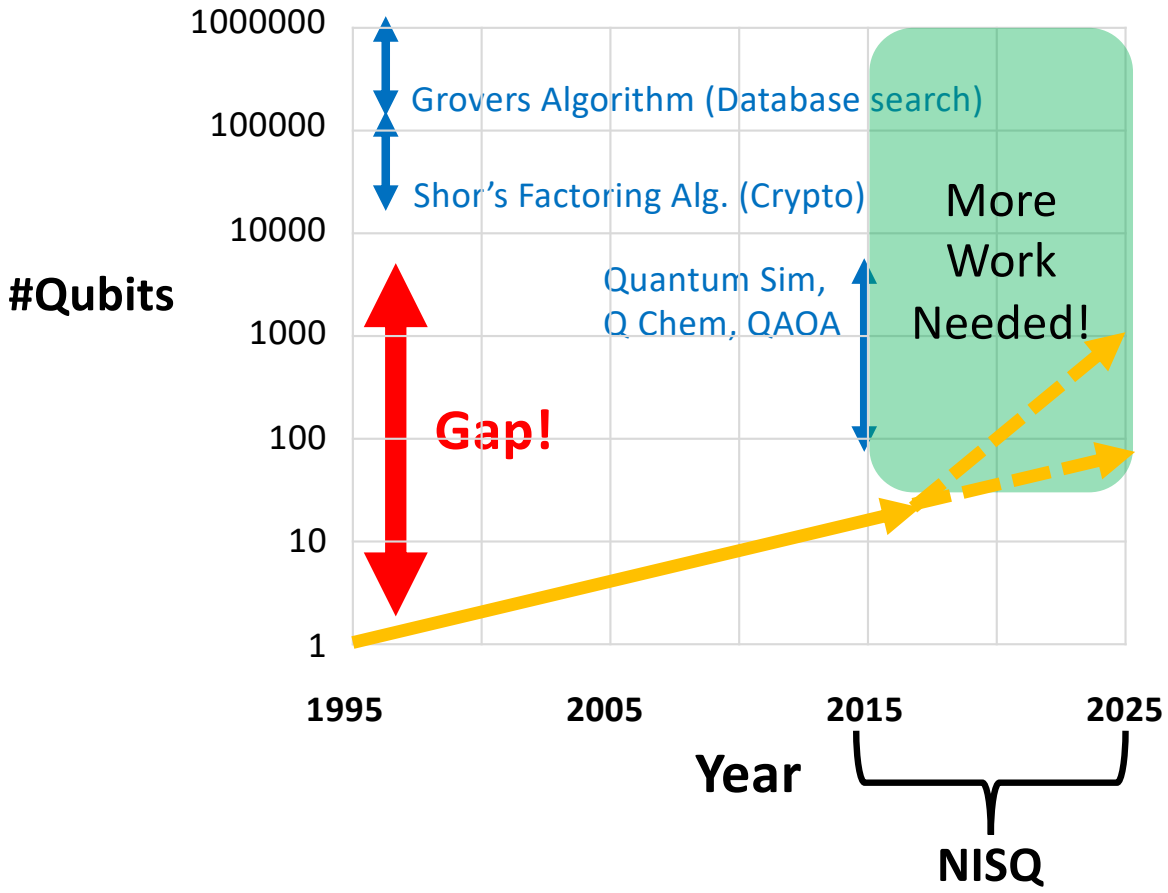
QC Algorithms to Machines Gap: The NISQ Era



- Noisy Intermediate-Scale Quantum (NISQ)
  - Preskill, Jan 2018
  - 10-1000 qubits
- Too small for known algorithms with exponential speedup
- Too small for ECC

- Large enough to support interesting experiments!

QC Algorithms to Machines Gap: Opportunity



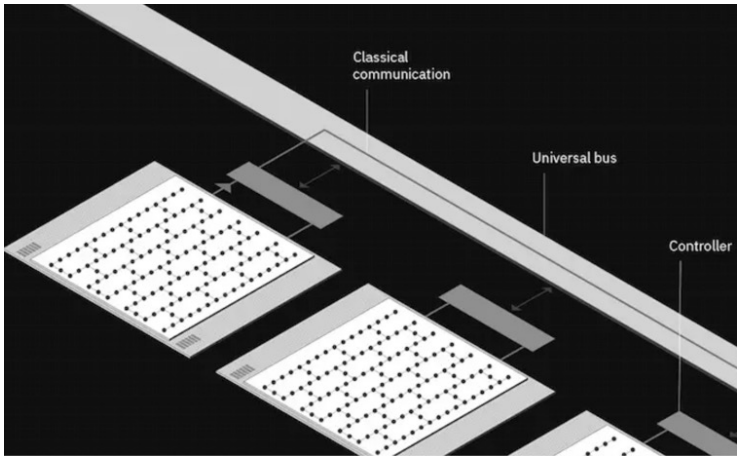
QC programming and design tools that shrink the gap can move the feasibility point years sooner!

- Reduce algorithm qubit requirements
- Improve effectiveness of hardware qubits

Scaling Quantum Systems: Mind the Gap!

- Today: Small NISQ QC Systems available for use
- For quantum advantage, most algorithms require a large and reliable QPU. But, building such monolithic QPUs is challenging.
  - E.g., 27-qubit IBM Kolkata has 2X the “quantum volume” (capability) of 127-qubit IBM Washington, despite many fewer qubits
- Still much easier to build multiple smaller QPUs.
- How do we make use of the multiple small QPUs to run large target applications?

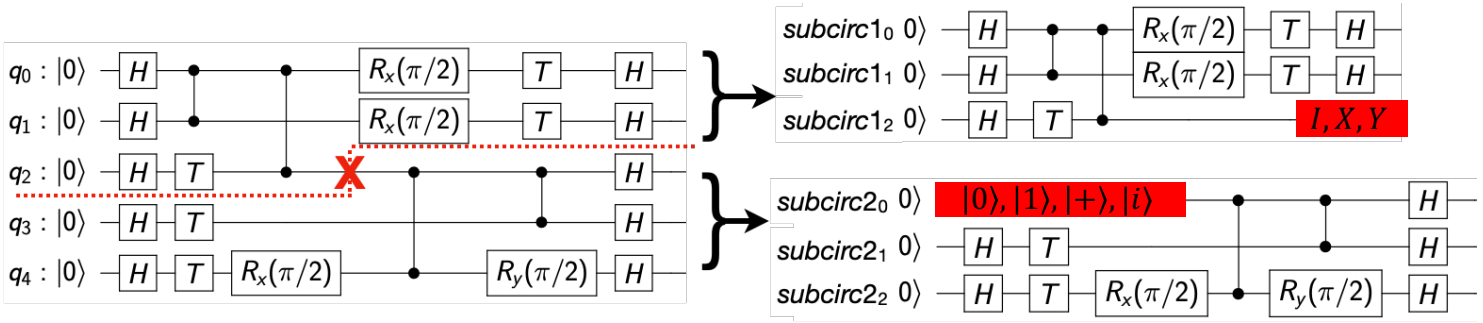
IBM Roadmap



Example 4: CutQC: Combining Classical and Quantum Computation to Run QC algorithms at Larger Scale

- Approach: Cut quantum circuits into smaller subcircuits that fit and reconstruct the results classically afterward.
- Challenge: Classical reconstruction scales exponentially!
- Solution: parallel processing[1] and GPU[2].

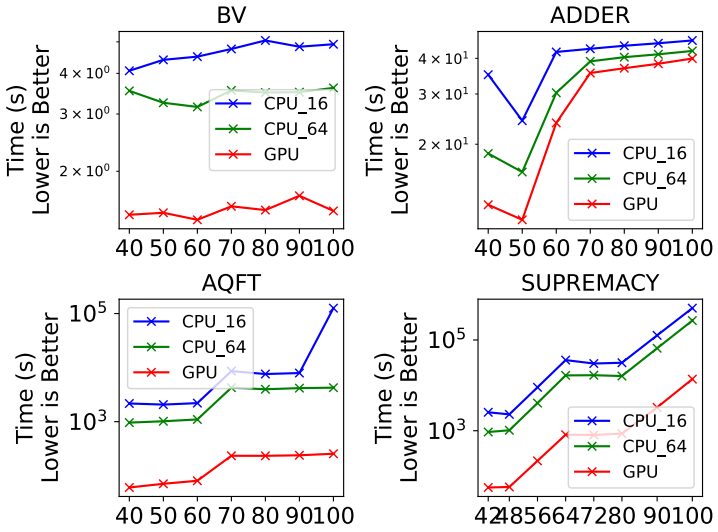
Example: Cut one edge to split a 5-qubit circuit into two smaller (3-qubit each) subcircuits.



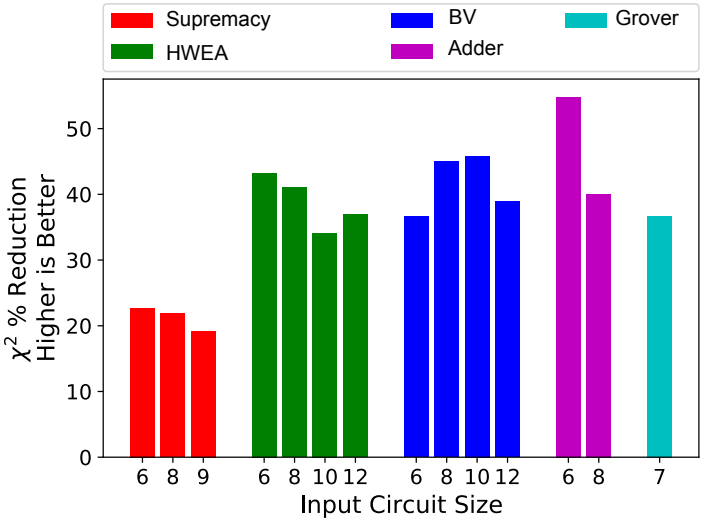
[1]Tang, Wei, Teague Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. "Cutqc: using small quantum computers for large quantum circuit evaluations." In *Proceedings of the 26th ACM International conference on architectural support for programming languages and operating systems*, pp. 473-486. 2021.

[2]Tang, Wei, and Margaret Martonosi. "Cutting Quantum Circuits to Run on Quantum and Classical Platforms." *arXiv preprint arXiv:2205.05836* (2022).

Result: Runtime and Fidelity Improvements



Faster than classical.



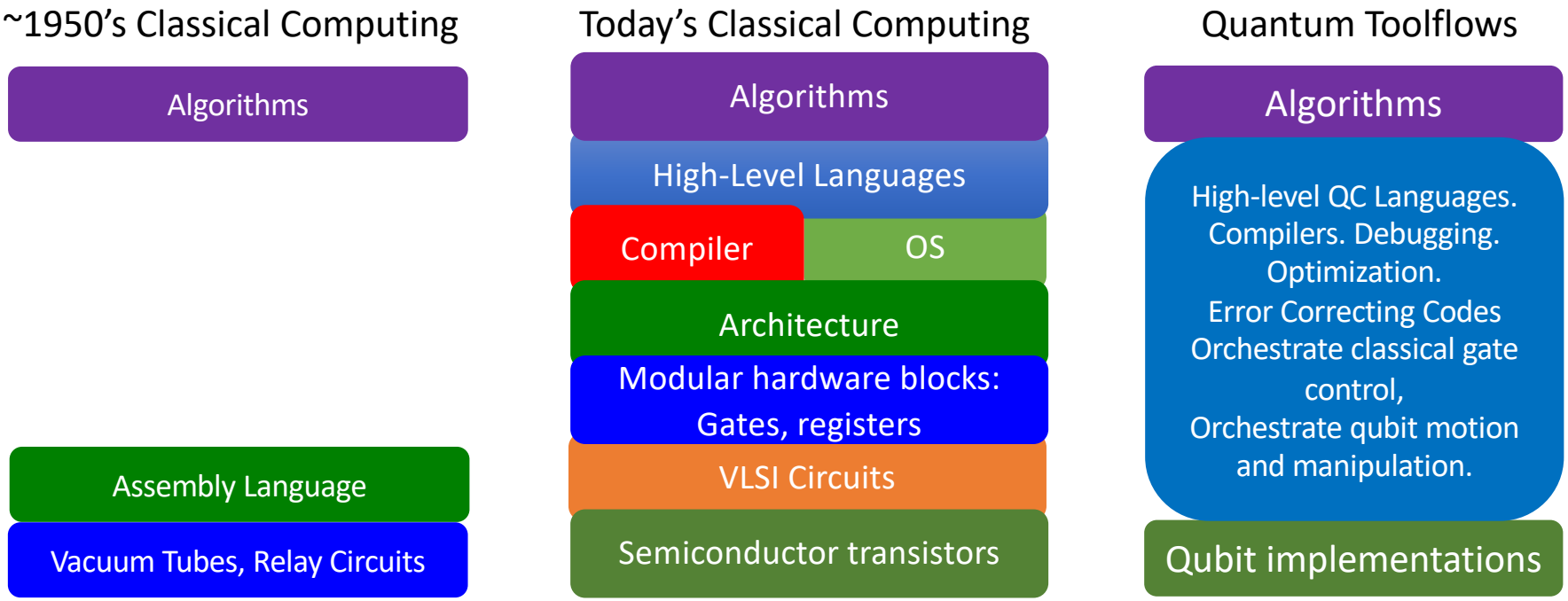
Higher fidelity than large monolithic QPUs.

- Cut and run benchmarks with up to 75% of number of qubits in input circuits.
- Runtime shows the reconstruction of 2^{30} bins. GPU is the fastest backend as expected.
- CutQC achieves an average of 21% to 47% fidelity improvement

CutQC: Takeaways

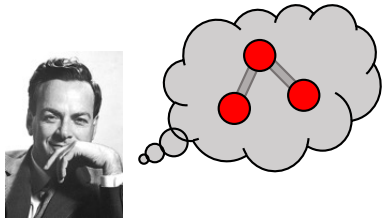
- Leverage both quantum and classical computing platforms
  - Execute quantum algorithms beyond typical QC frontier, eg up to 100 qubits
  - while simultaneously improving the fidelity of the output.
- Results significantly beyond the current reach of quantum or classical methods alone.
- Insight: Small QC + CutQC > Large QC.

Quantum Systems Today: An Analogy

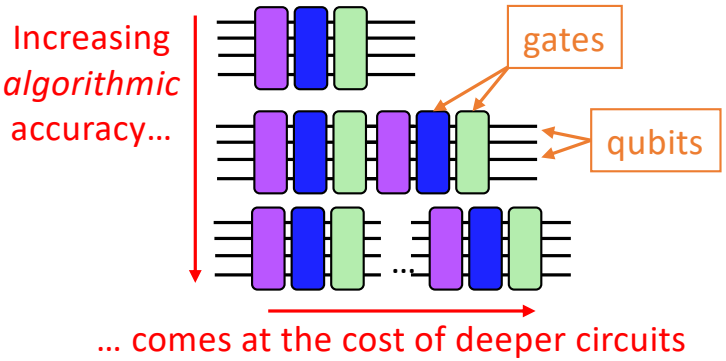


Example 5: Using Codesign to optimize Hamiltonian Simulation

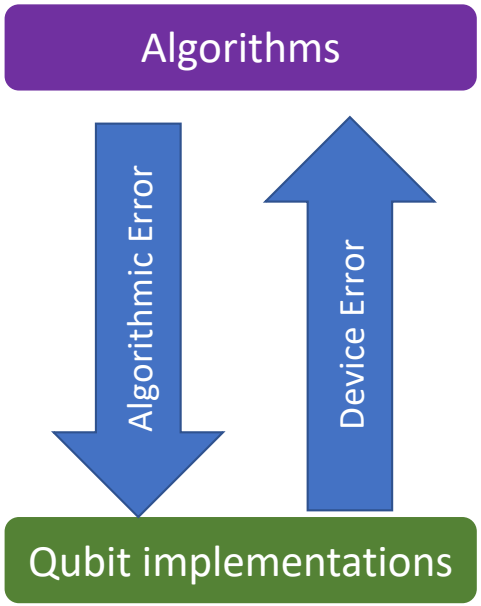
1. Hamiltonian Simulation



Balance tradeoffs when mapping the problem to a QC

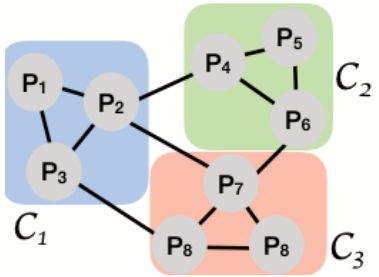


2. Cross-layer Codesign

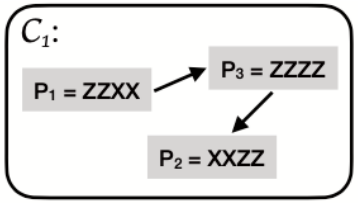


3. Max-commute-tsp

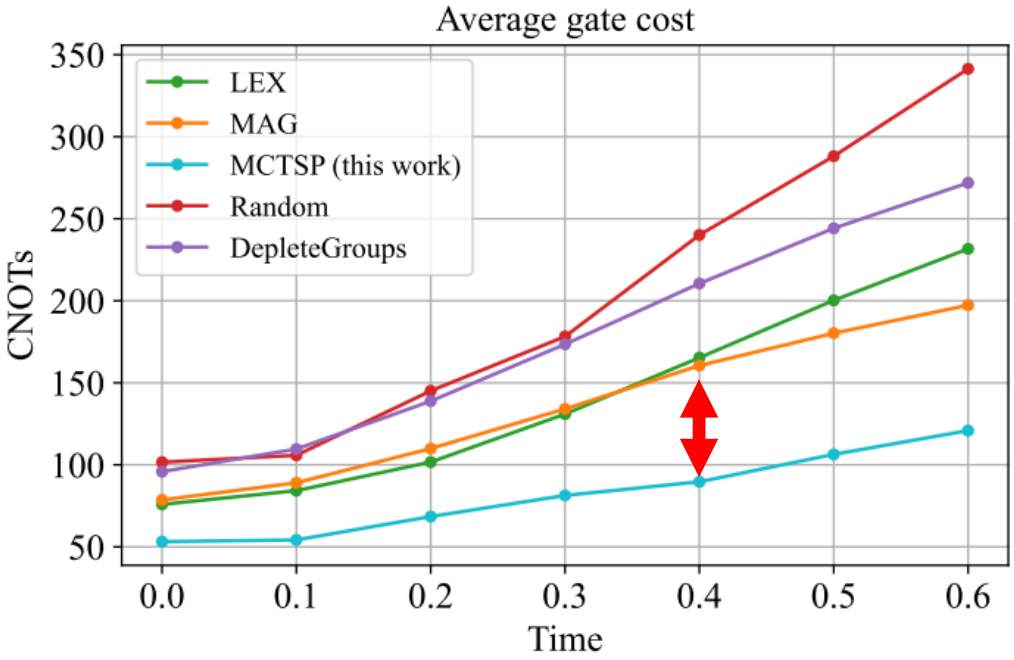
- Mitigate algorithmic errors
  - Group commuting terms together



- Mitigate physical errors
  - Sort terms using TSP



Simultaneous optimization results in 40% fewer CNOT gates in equal accuracy comparisons



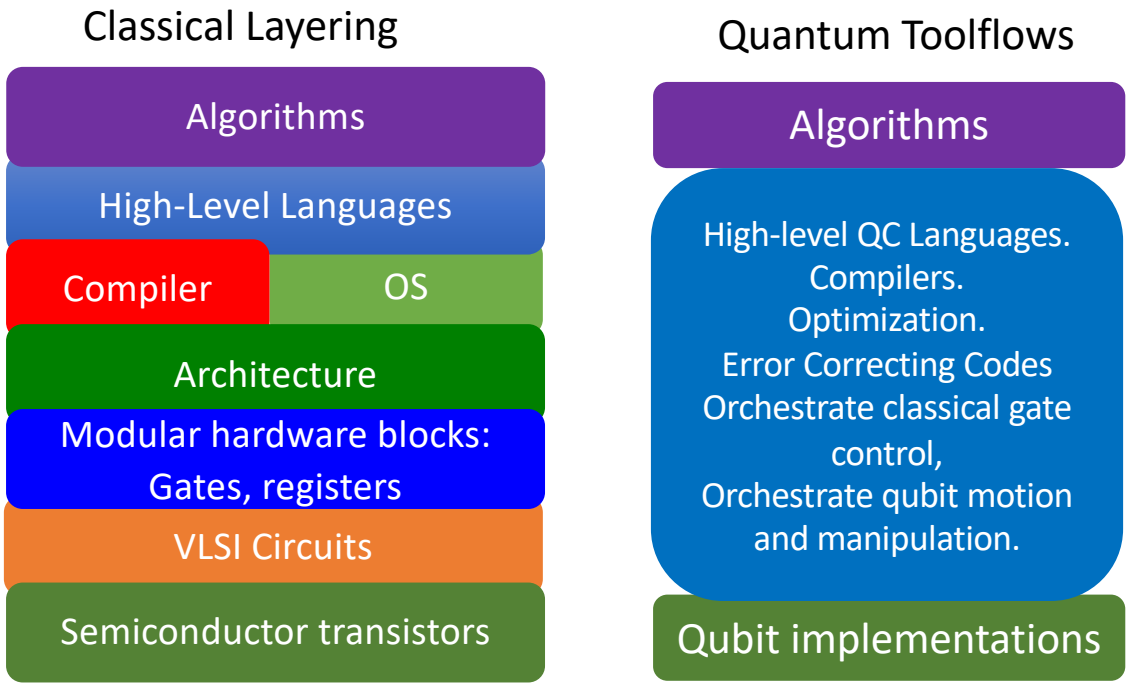
- Simultaneously mitigate *both* algorithmic and physical errors
- Codesign optimizations useful now and into the future when NISQ transitions to fault-tolerant approaches

Tomesh, Gui, Gokhale, Shi, Chong, Martonosi, Suchara.
"Optimized Quantum Program Execution Ordering to Mitigate Errors in Simulations of Quantum Systems."
In *2021 Intl. Conf. on Rebooting Computing (ICRC)* **Best Paper Award**

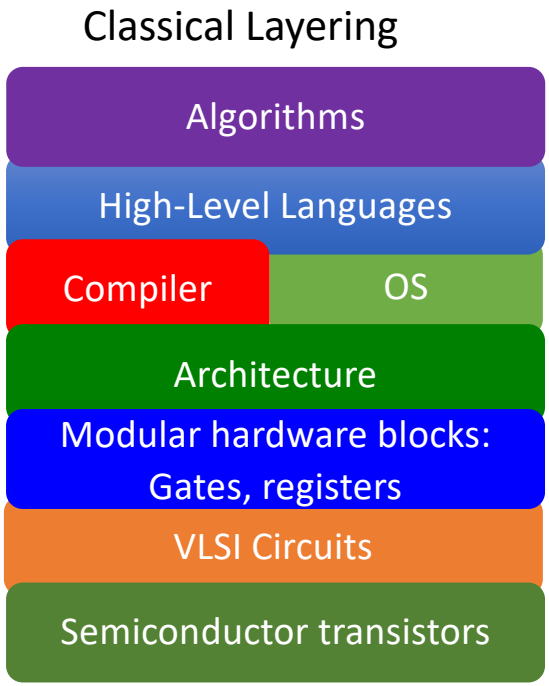
Other QC Examples

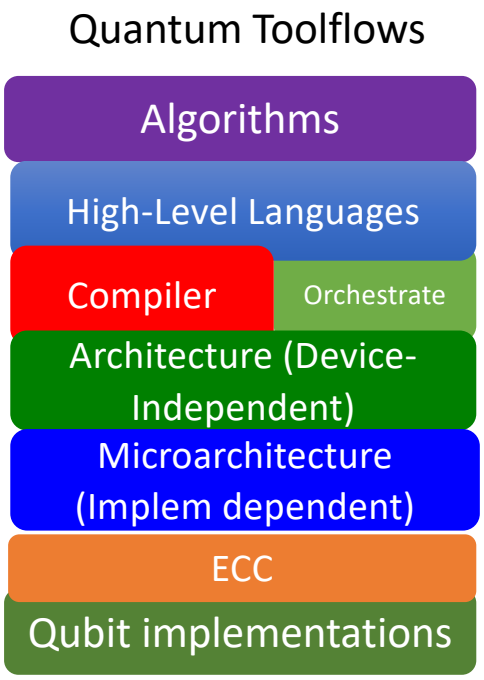
- Tolerating long computation (ie gate) latencies:
  - SIMD operating zones to parallelize many qubit operations [Chi, ISCA 2006]
  - Multi-SIMD approaches allow different gate types to be executed in same cycle [Javadi-Abhari, CF 2014, Best paper]
- Arch and App tradeoffs for ECC: [Javadi-Abhari, MICRO-50]
- Accounting for communication latency
  - Achieving high Multi-SIMD parallelism requires properly accounting for qubit movement times. [Heckey, ASPLOS 2015]
- Scaffold programming language and ScaffCC Compiler [Javadi-Abhari, CF 2014, Best paper]
- Proposing and evaluating QC PL assertions for debuggable QC code [Huang, Plateau, 2018]
- Recurring theme: Full-stack knowledge from Apps to HW characteristics is important, and will be even more so in NISQ devices.

Quantum Systems: Layering Options



Quantum Systems: Layering Options

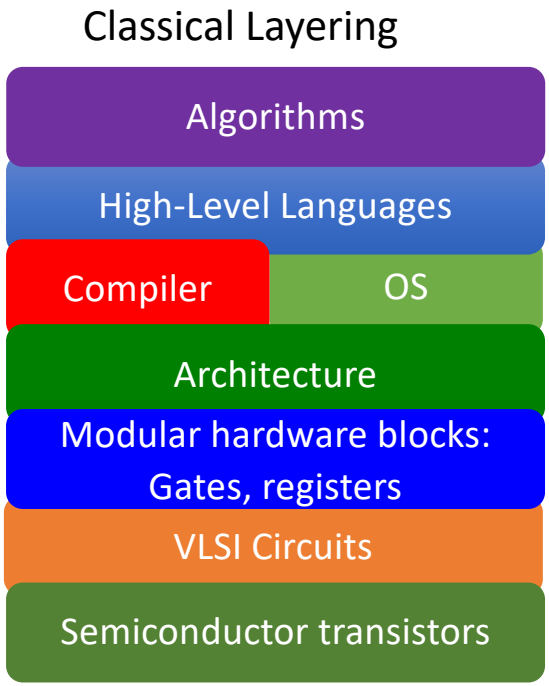


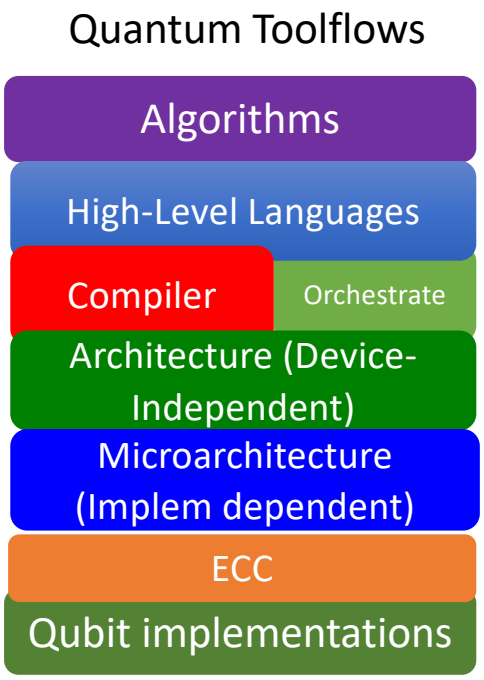


High-level QC Languages.
Compilers.
Optimization.
Error Correcting Codes
Orchestrate classical gate control,
Orchestrate qubit motion and manipulation.

Option 1

Quantum Systems: Layering Options

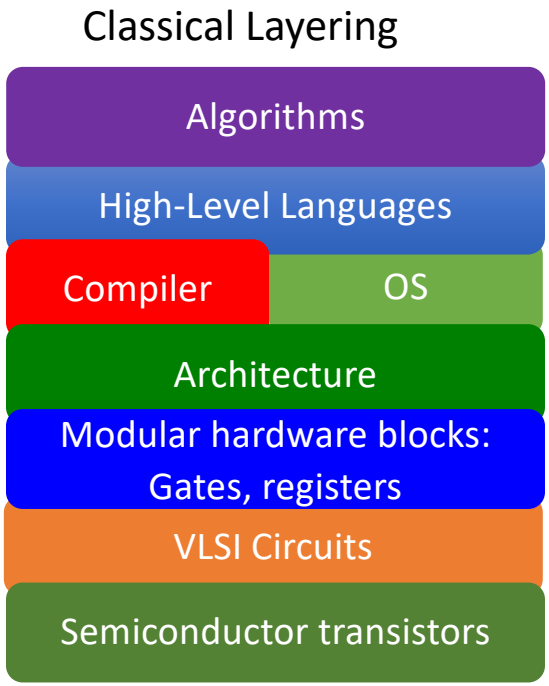


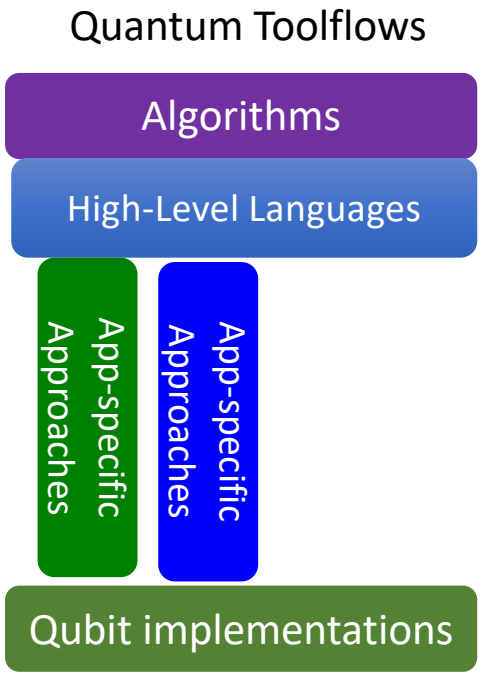


Challenge:
Facing tight resource constraints, need more info flow up and down stack about device and algorithm characteristics.

Option 1

Quantum Systems: Layering Option 2



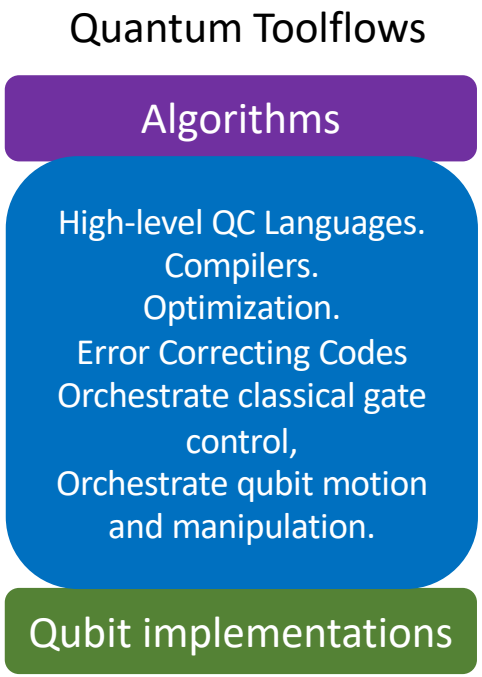


Info flow up/down stack about:

- error characteristics,
- communication latencies
- Connectivity
- Gate decompositions
- App parallelism, resource requirements
- + other characteristics …

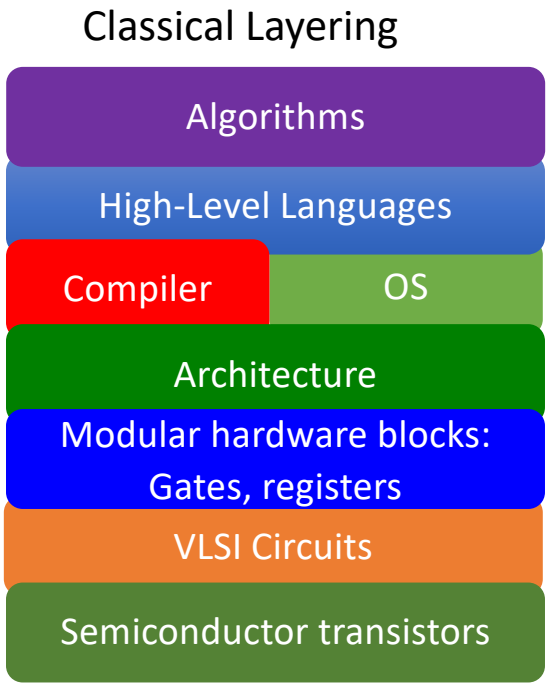
Think of Classical Accelerators…

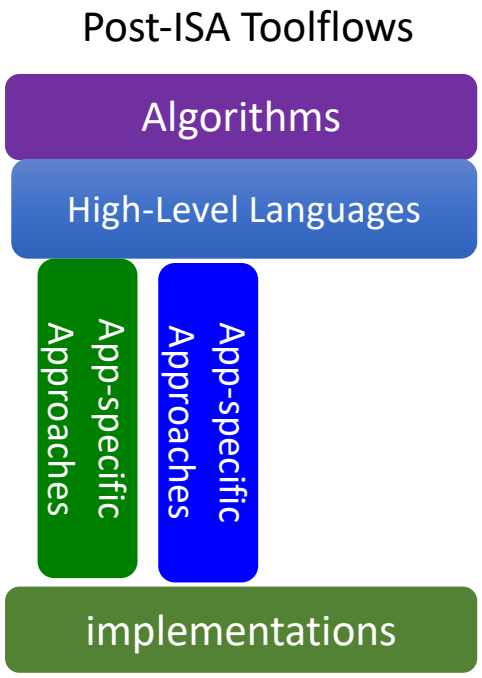
Conclusions & What's next?



- QC is NOT a Moore's Law replacement
  - Unique, special-purpose hardware
  - Focused applications
- But potentially game-changing
  - Make intractable tractable
  - Lessons learned (algs, systems, devices) drive innovation on classical side as well
- Full CS ecosystem needed to shift QC from theoretical to commercial

Post-ISA Systems: Rethinking the Stack in Both Classical and QC Systems





Examples:

- Accelerator-oriented Parallelism
- TPUs, Cerebras, Sambanova, …

- Tailored approaches for sparse, memory-bound applications
- Dalorex
- QC Toolflows

…

Overall: A Seismic Shift in Computer Systems Design!

Thanks to:

Students and Co-Authors

Funding: DARPA, DOE

For more info:

http://mrmgroup.cs.Princeton.edu

Quantum Resources:

CRA CCC Workshop Report "Next Steps in QC: Computer Science's Role"
https://arxiv.org/abs/1903.10541

National Academies Report "Quantum Computing Progress and Prospects"
https://www.nap.edu/catalog/25196/quantum-computing-progress-and-prospects