# Seismic Shift: What's next in the Post-Moore, Post-ISA era?

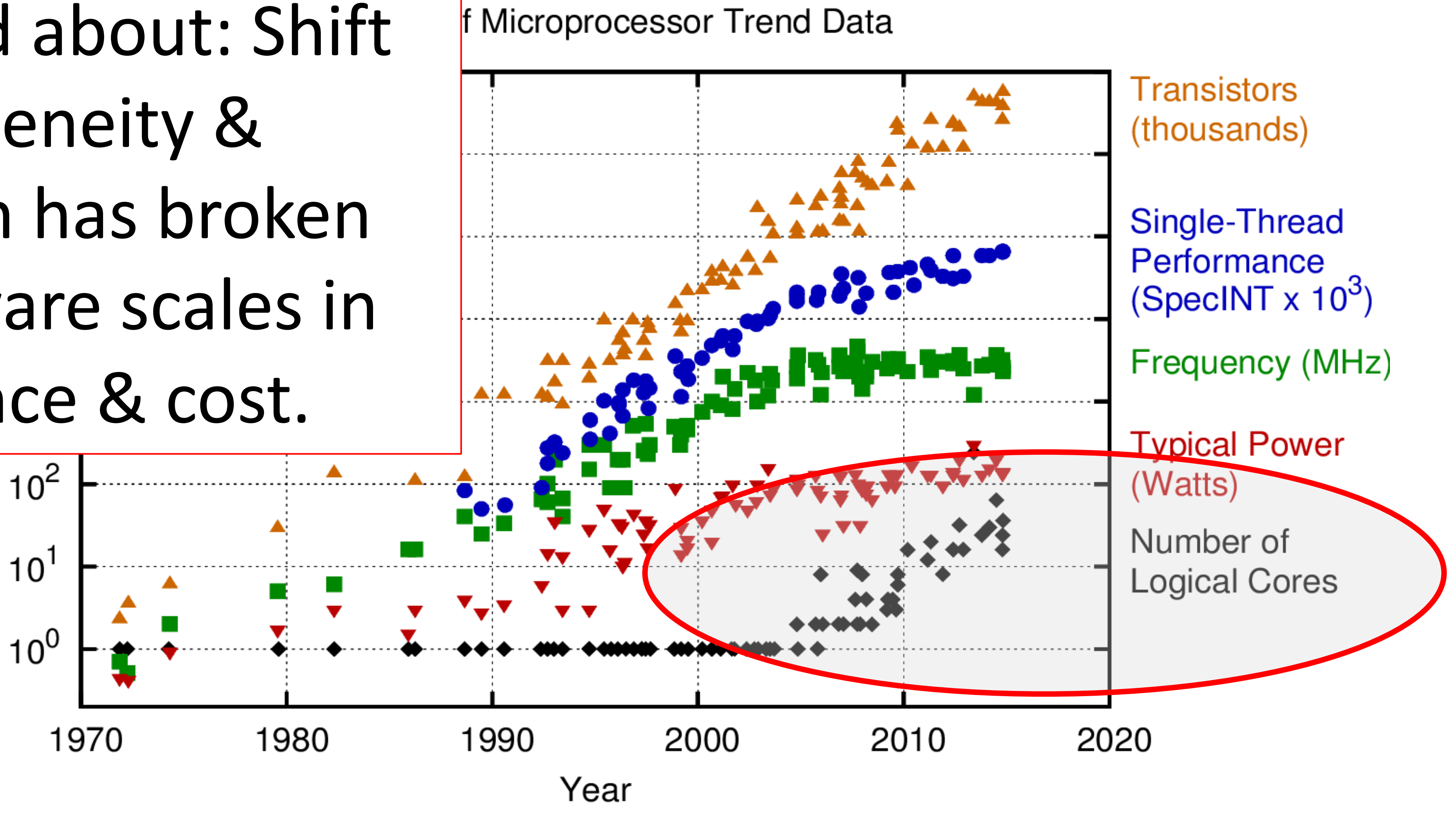
Prof. Margaret Martonosi

H. T. Adams '35 Prof. of Computer Science

Princeton University

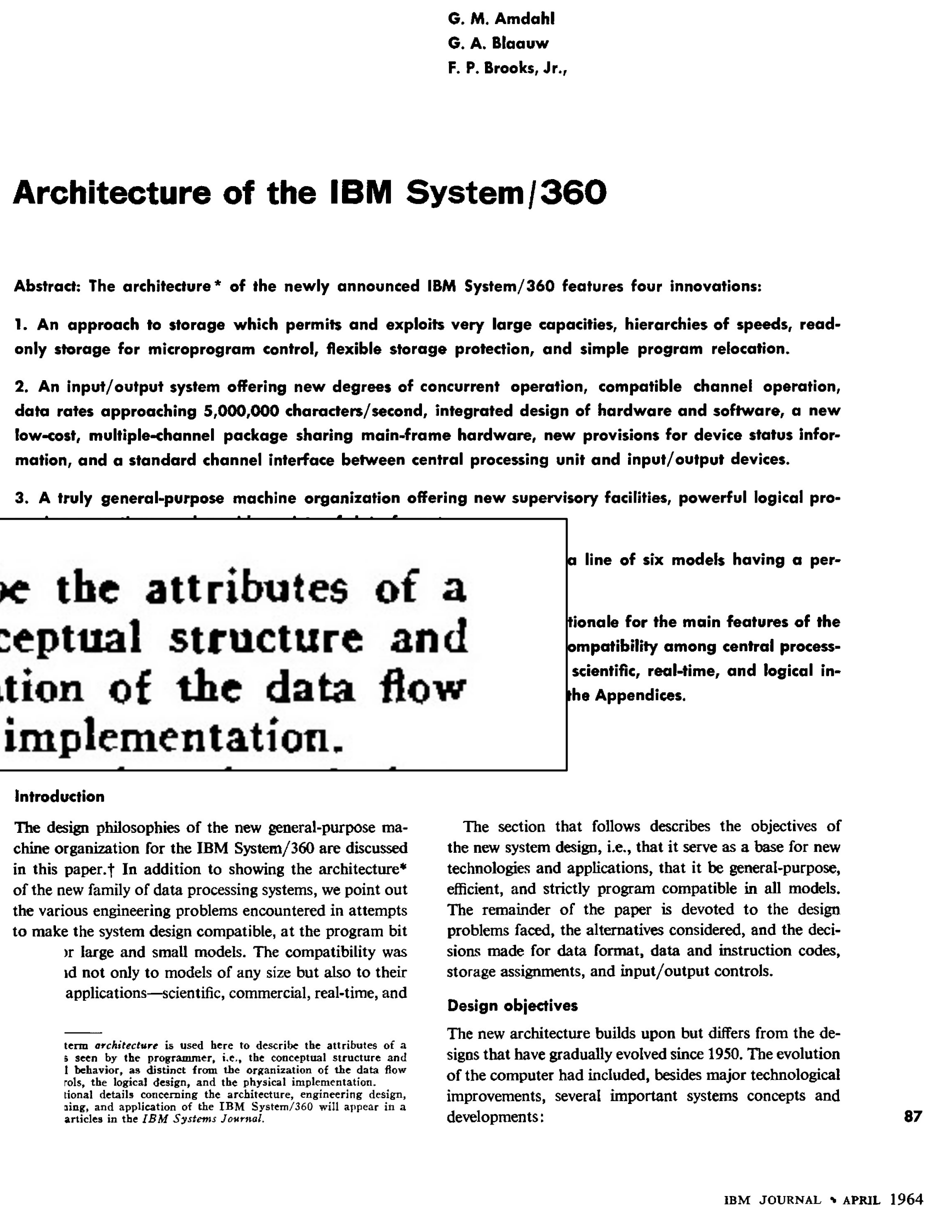
# Not News: End of Decades of Moore's Law scaling

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…



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- the Appendices.**

**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 ɔr large and small models. The compatibility was ıd 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.

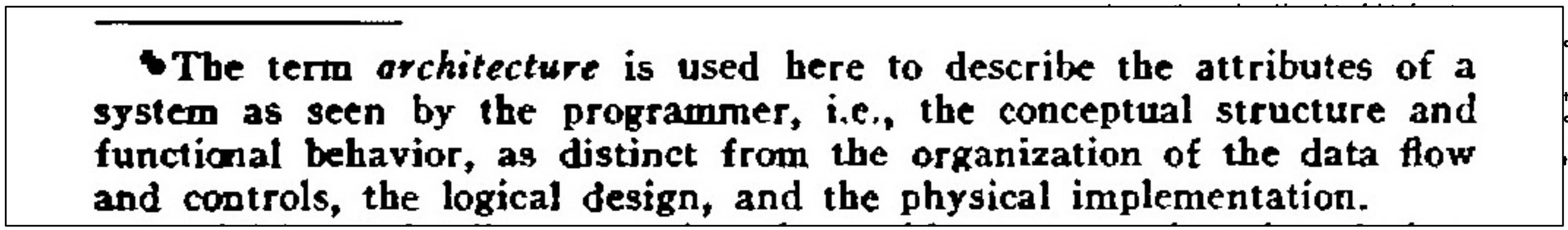
**Design objectives**

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, ıing, and application of the IBM System/360 will appear in a articles in the *IBM Systems Journal*.

87

IBM JOURNAL • APRIL 1964



*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.

= The value of HW/SW abstraction…

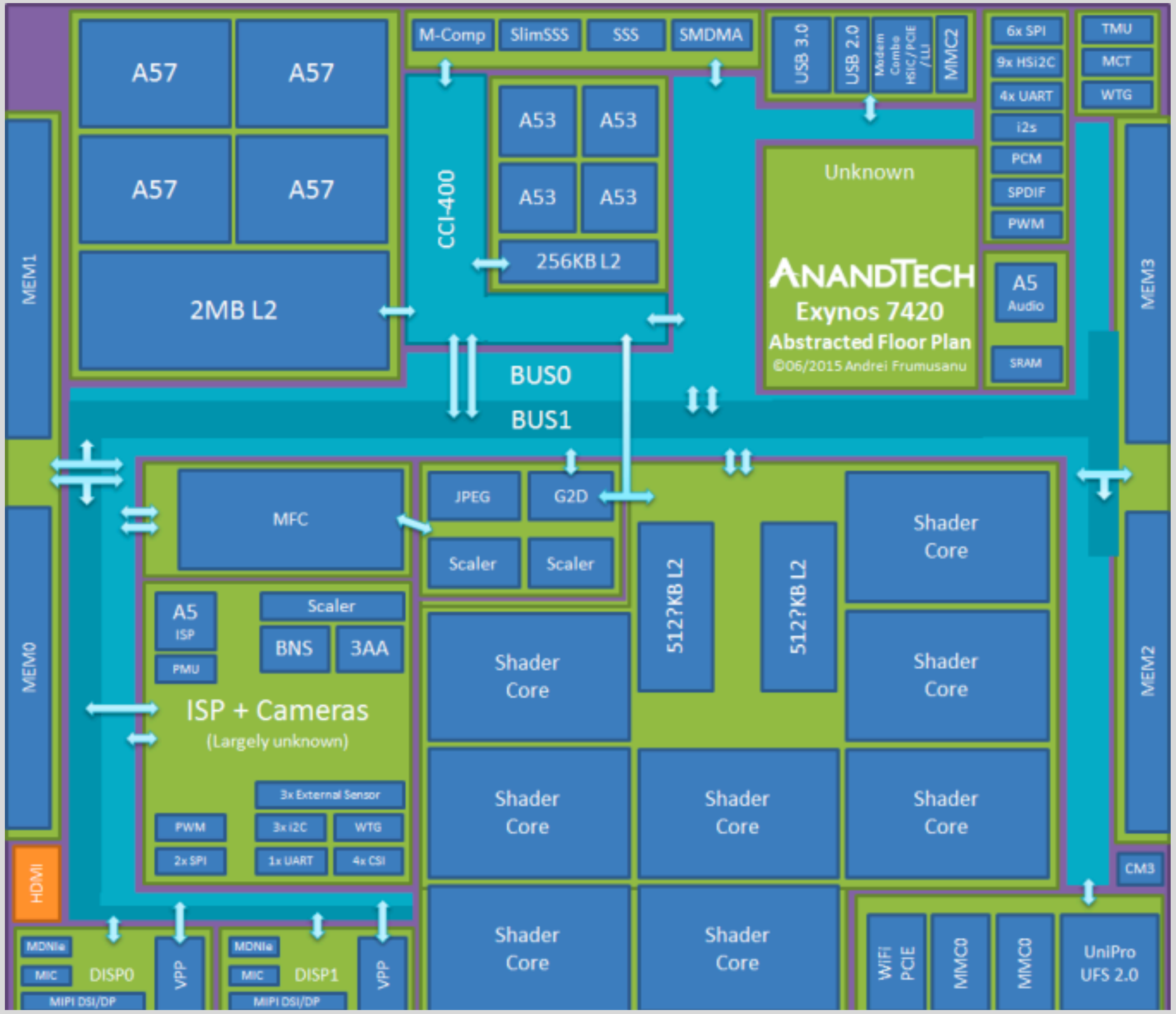
## 2019:
## We are here

Heterogeneity "In the Large":

- 1000's of distinct Android devices
- IoTs even more diverse

Heterogeneity "In the Small"

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



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?
- How to verify them?
- And what technologies come next?

## 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?
- How to verify them?
- And what technologies come next?

## 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?
- How to verify them?
- And what technologies come next?

# DECADES: A VERTICALLY-INTEGRATED APPROACH

## Language and Compiler Support

- Enhance data locality
- Optimize spatial mapping of threads
- Enable in-memory computing

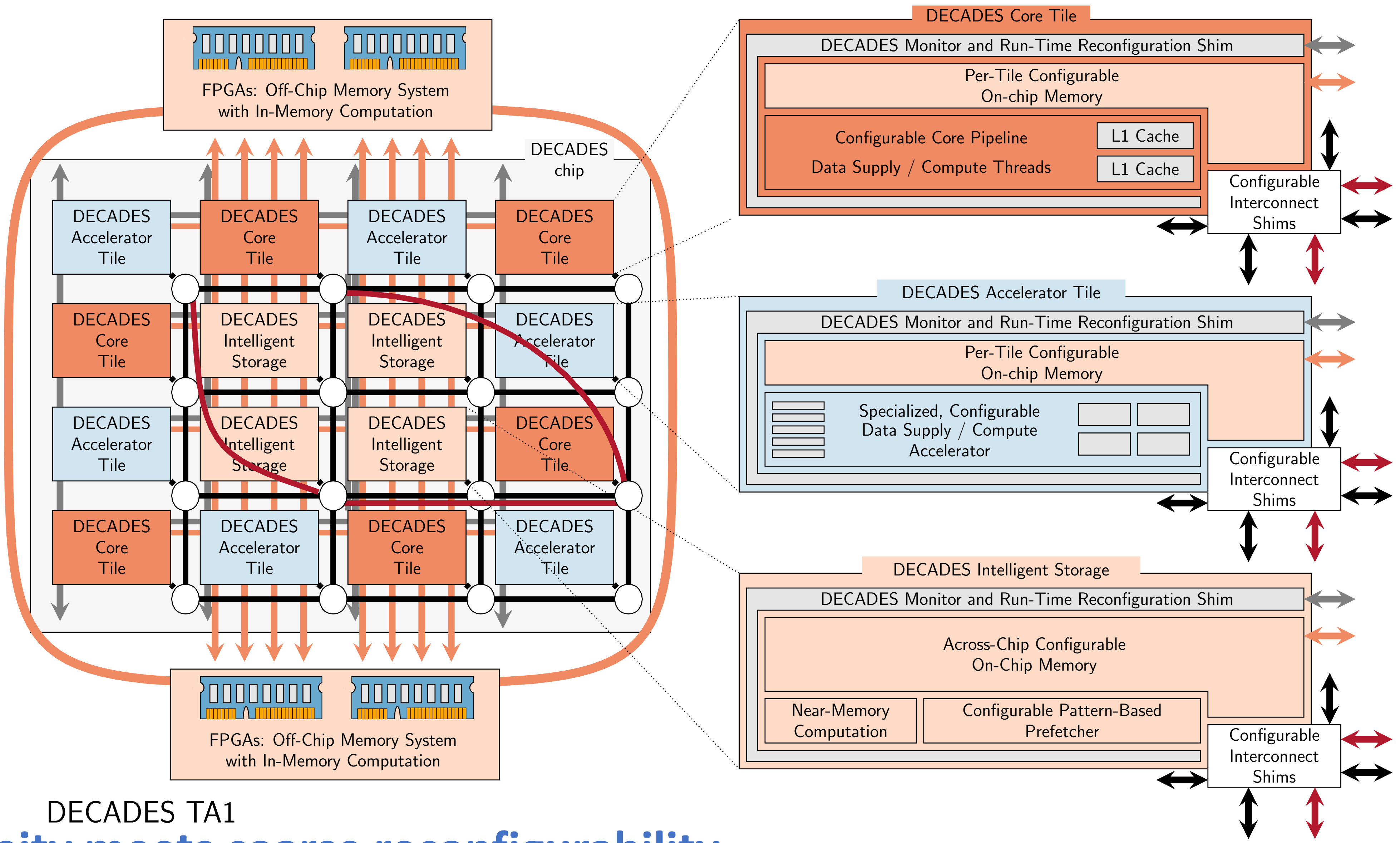
## Very Coarse-Grained Reconfigurable Tile-Based Architecture

- Coarser than CGRA → VCGRTA
- 3 classes of reconfigurable tiles
- Reconfigurable interconnection network
- Reconfigurable in-memory computing

## Multi-Tiered Demonstration Strategy

- Scalable full-system simulation
- Multi-FPGA emulation infrastructure
- 225-tile DECADES chip prototype

# DECADES PLATFORM ARCHITECTURE



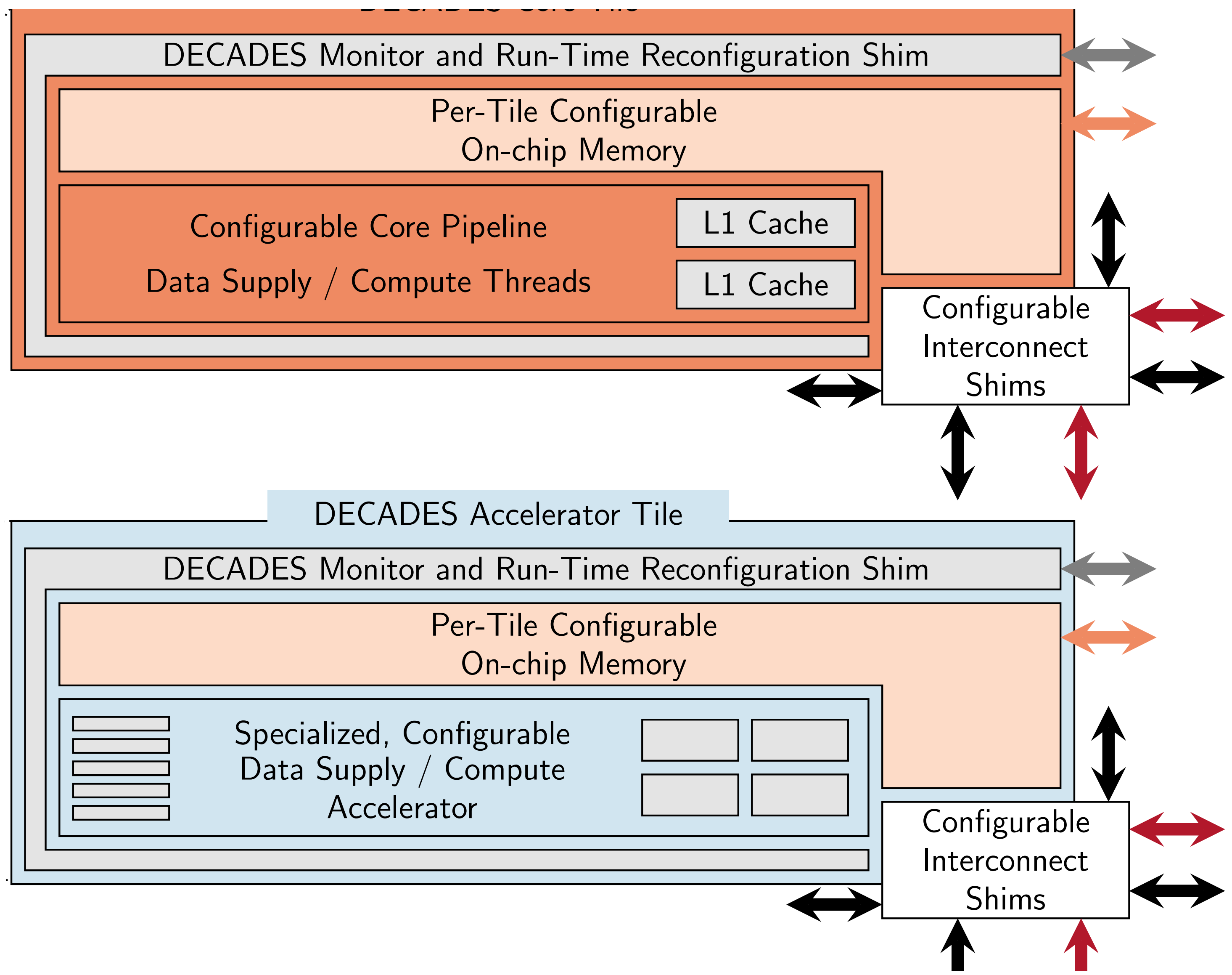
FPGAs: Off-Chip Memory System with In-Memory Computation
DECADES chip
DECADES Accelerator Tile
DECADES Core Tile
DECADES Accelerator Tile
DECADES Core Tile
DECADES Core Tile
DECADES Intelligent Storage
DECADES Intelligent Storage
DECADES Accelerator Tile
DECADES Accelerator Tile
DECADES Intelligent Storage
DECADES Intelligent Storage
DECADES Core Tile
DECADES Core Tile
DECADES Accelerator Tile
DECADES Core Tile
DECADES Accelerator Tile
FPGAs: Off-Chip Memory System with In-Memory Computation
DECADES Core Tile
DECADES Monitor and Run-Time Reconfiguration Shim
Per-Tile Configurable On-chip Memory
Configurable Core Pipeline
Data Supply / Compute Threads
L1 Cache
L1 Cache
Configurable Interconnect Shims
DECADES Accelerator Tile
DECADES Monitor and Run-Time Reconfiguration Shim
Per-Tile Configurable On-chip Memory
Specialized, Configurable Data Supply / Compute Accelerator
Configurable Interconnect Shims
DECADES Intelligent Storage
DECADES Monitor and Run-Time Reconfiguration Shim
Across-Chip Configurable On-Chip Memory
Near-Memory Computation
Configurable Pattern-Based Prefetcher
Configurable Interconnect Shims

DECADES TA1

**Heterogeneity meets coarse reconfigurability**

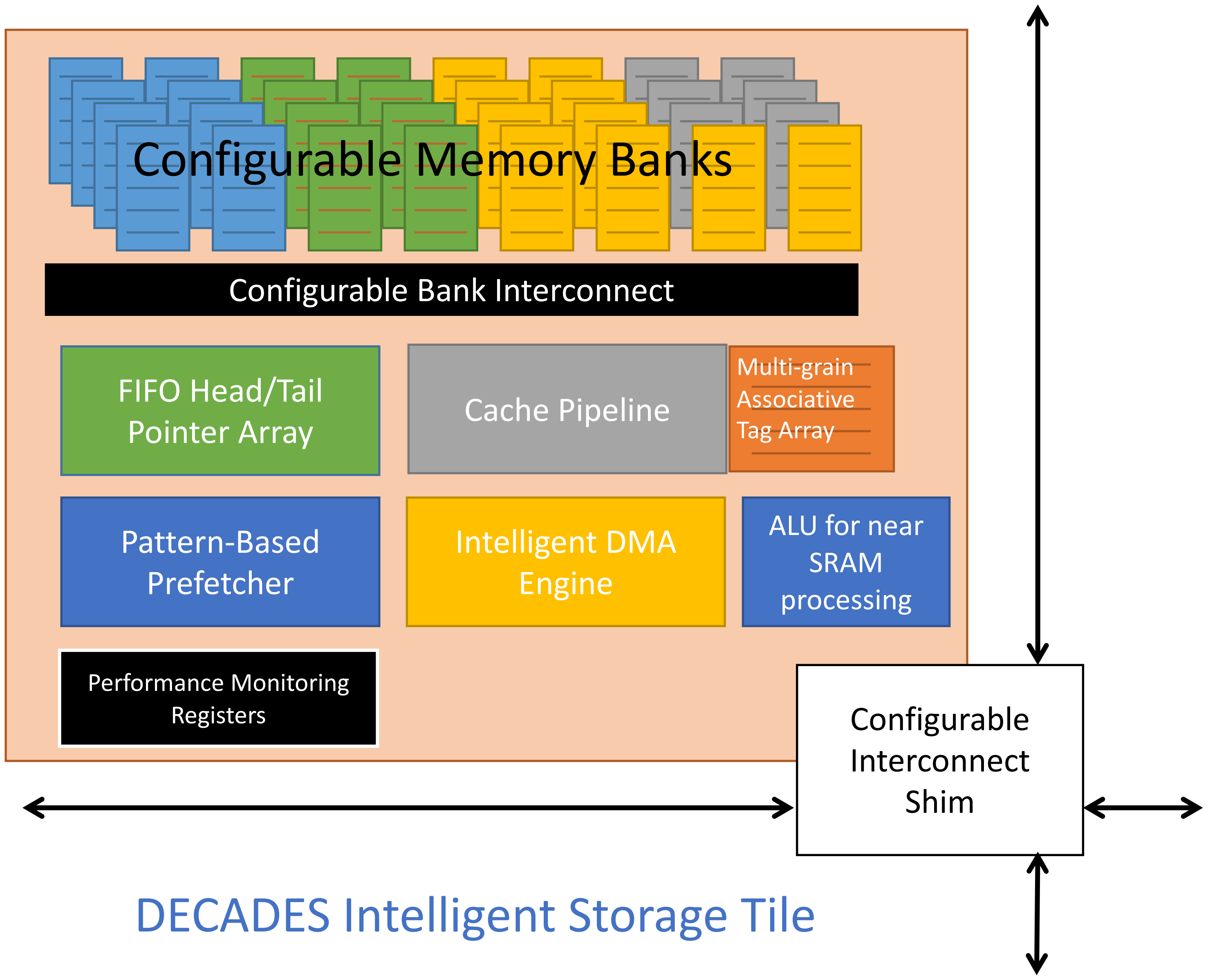
# DECADES Core and Accelerator tiles

- Computations mapped onto core tiles or available accelerator tiles
    - Each tile is wrapped in monitor/reconfiguration shim
- Dynamic reconfiguration of Supply-Compute decoupling, power-performance tradeoffs, and interconnect


DECADES Monitor and Run-Time Reconfiguration Shim
Per-Tile Configurable On-chip Memory
Configurable Core Pipeline
Data Supply / Compute Threads
L1 Cache
L1 Cache
Configurable Interconnect Shims
DECADES Accelerator Tile
DECADES Monitor and Run-Time Reconfiguration Shim
Per-Tile Configurable On-chip Memory
Specialized, Configurable Data Supply / Compute Accelerator
Configurable Interconnect Shims

# DECADES Storage Specialization

- Specialization #1: Map apps onto mix of compute tiles and intelligent storage (IS) tiles
- Specialization #2: Select and configure appropriate storage features within IS
    - Configurable memory banks + address and prefetching features
    - Simple near-SRAM ALU


Configurable Memory Banks
Configurable Bank Interconnect
FIFO Head/Tail Pointer Array
Cache Pipeline
Multi-grain Associative Tag Array
Pattern-Based Prefetcher
Intelligent DMA Engine
ALU for near SRAM processing
Performance Monitoring Registers
Configurable Interconnect Shim
DECADES Intelligent Storage Tile

# Improving Latency-Bound Applications with Decoupled Execution

- Roots in seminal 1982 Smith DAE paper: Separately execute memory accesses (supply) from instructions that compute with them (compute)
- **Then**: Latency tolerance “simpler” than out-of-order execution
- **Now**: Fits well with accelerator-oriented design
  - Separate memory supply from accelerator
  - Orthogonal to DOALL parallelism and bandwidth optimizations
  - Automatically identify and slice at compile time

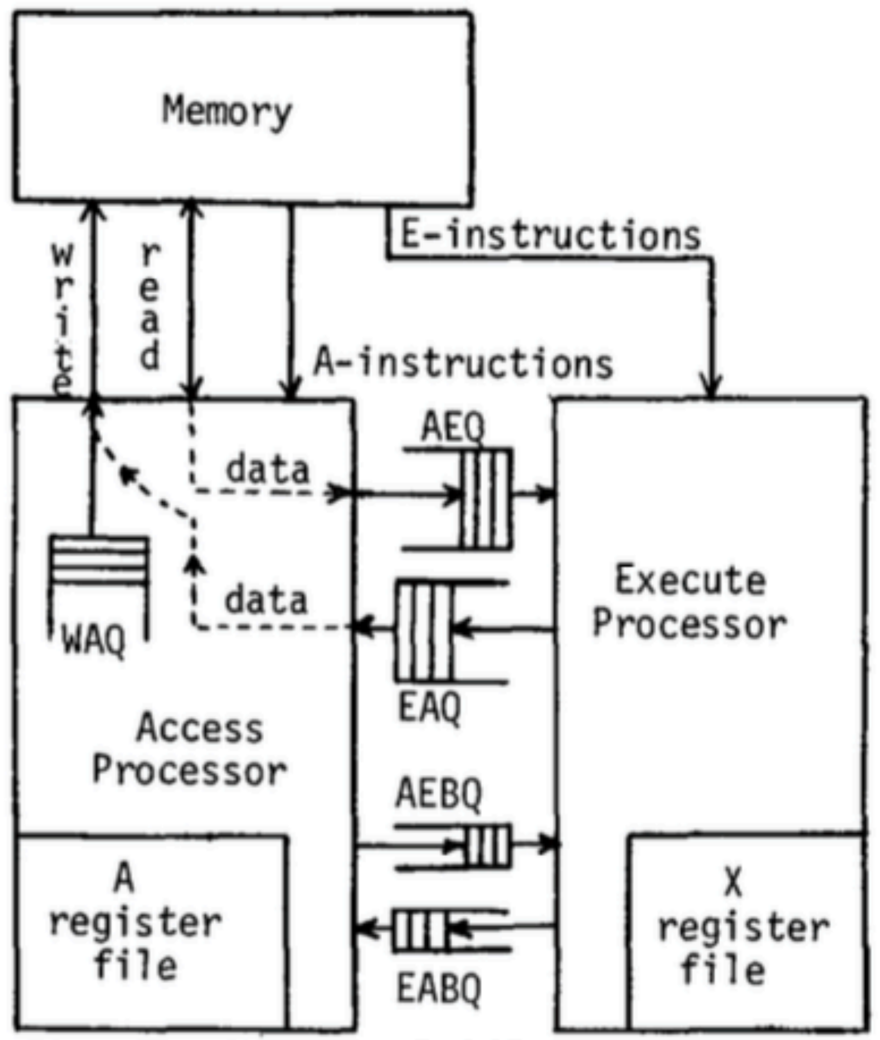
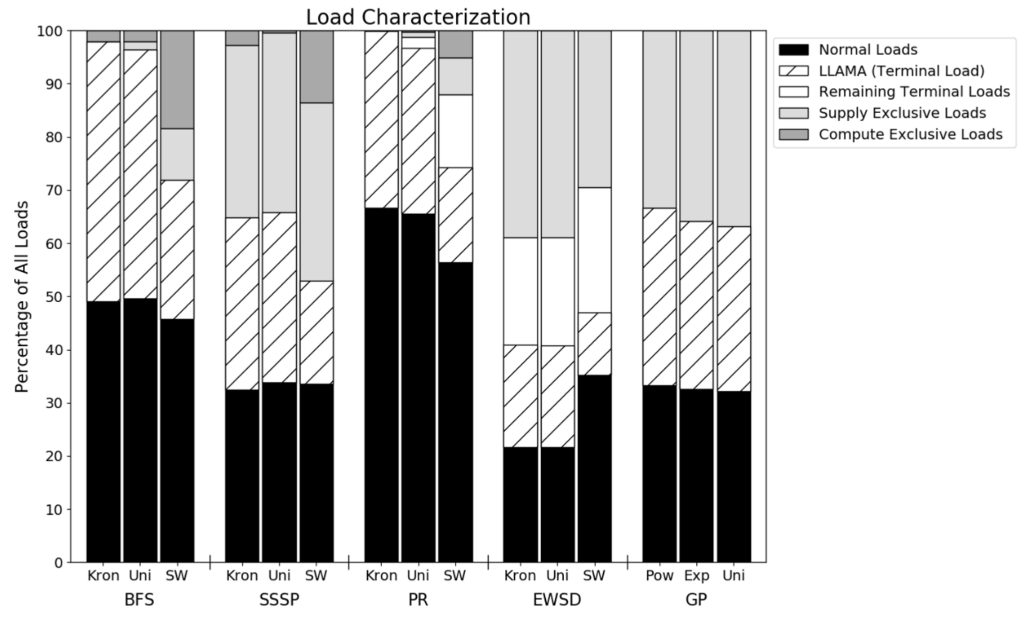


Fig. 1. Conceptual DAE Architecture



| Metric | Improvement |
|---|---|
| Performance | 10% |
| Area | 33% |

# DECADES Decoupled Supply-Compute Parallelism:

Automatic Compilation + HW support

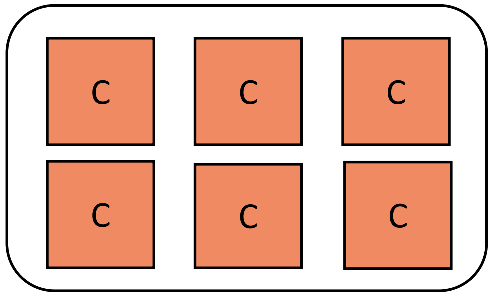
2 in-order CPUs Outperform 1 large out-of-order CPUs at 3X less power, area

# DECADES Reconfigurable Parallelism

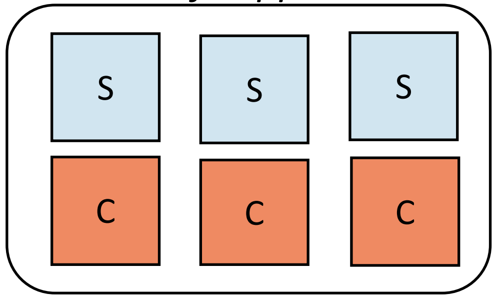
- **Coarse-grained reconfigurability**
- Workflow *Sinkhorn* contains a sequence of two kernel calls:
  - Sparse Elementwise
  - Dense Matrix Multiplication
- Each benefit from **different** system configurations for a finite number of resources
  - Sparse benefits from decoupled supply/compute
  - Dense benefits from traditional parallelism
- Decades architecture can be **reconfigured** to either use
  tiles as suppliers, or all tiles as compute

Example: 6-tiled system configs

*All Compute*



*Half Suppliers*



**Speedups**

| **Application** | **All Compute** | **Half Suppliers** |
|---|---:|---:|
| Sparse | 5x | **7x** |
| Dense | **6x** | 5x |

## 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?
- How to verify them?
- And what technologies come next?

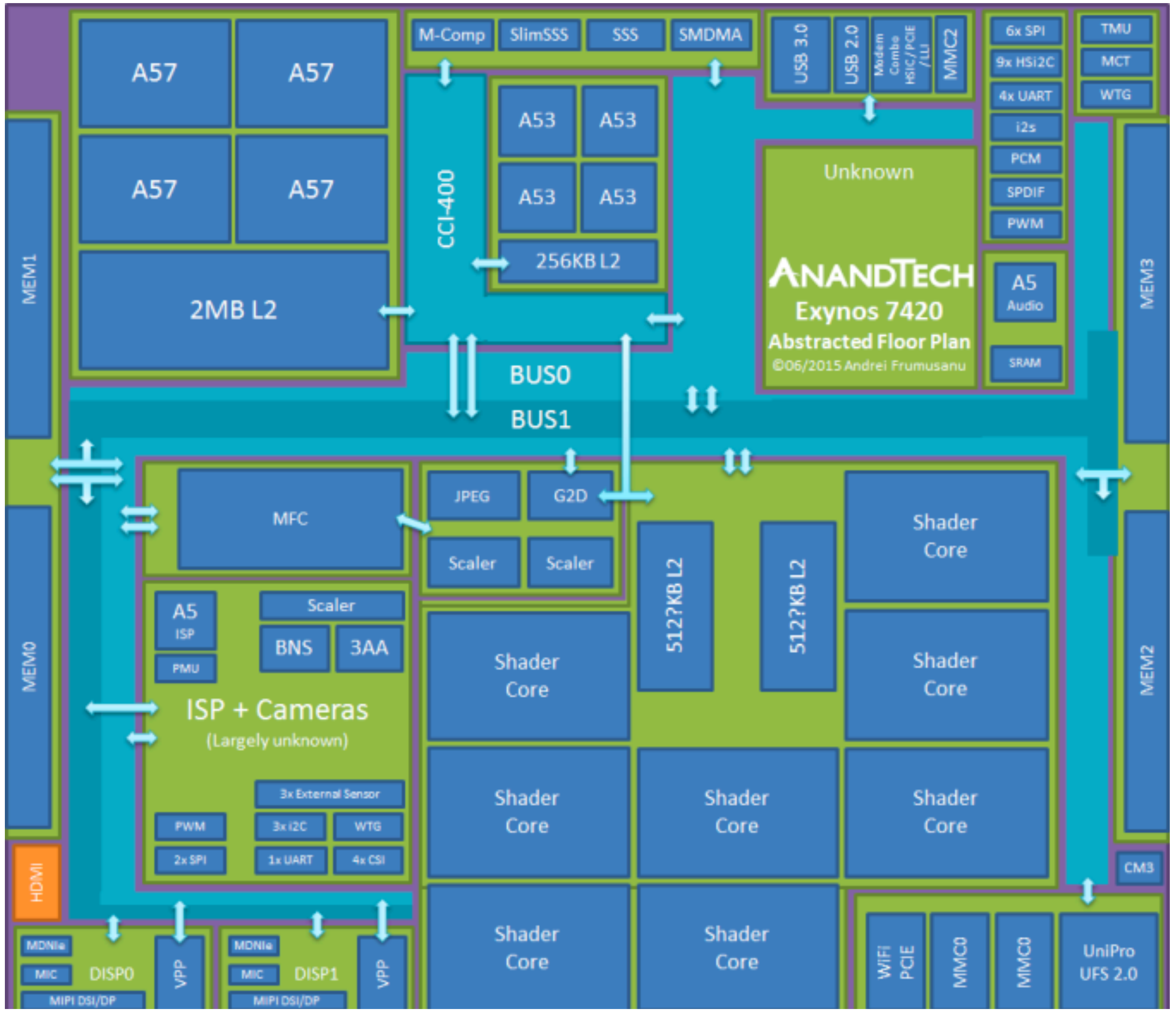
# Post-ISA Hardware Design

SOCs comprised of many CPUs, GPUs, and accelerators from many vendors

-> How to verify that a given block does what it is specified to do?

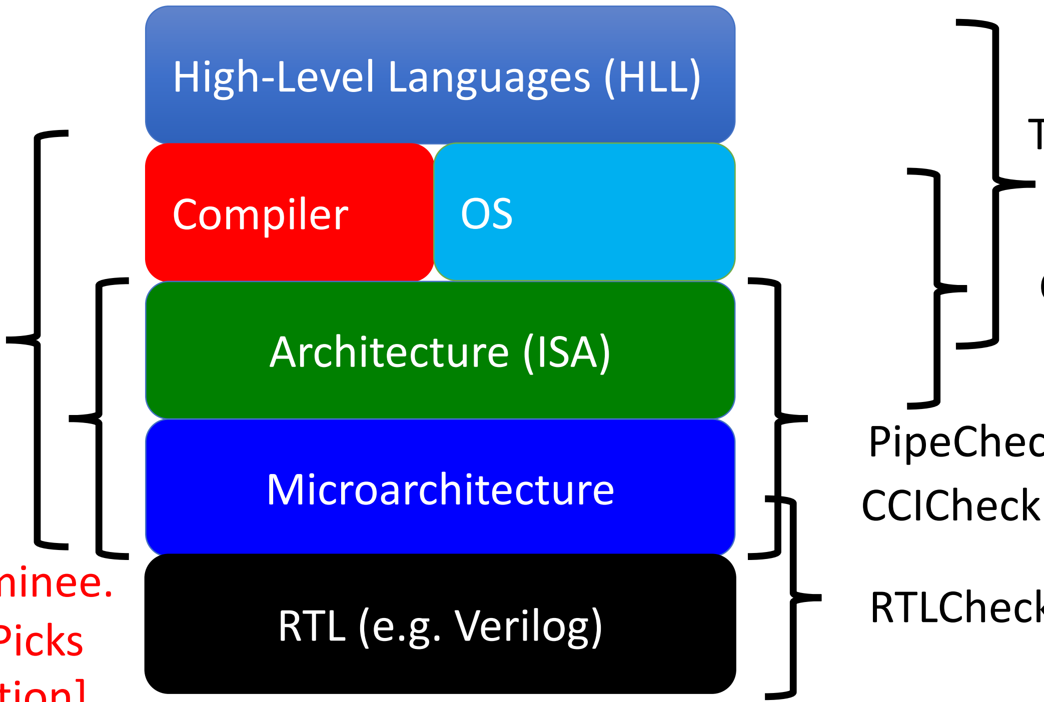
-> How to formally specify what blocks should do?

**-> Formal Interface Specifications are the new ISA!**



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

# The Check Suite: An Ecosystem of Tools For Verifying Memory Orderings and their Security Implications



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

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

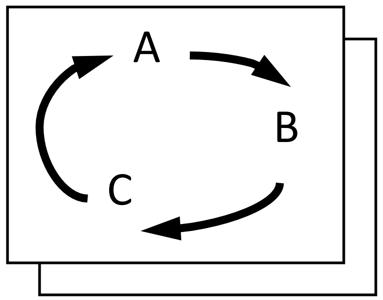
TriCheck [ASPLOS '17] [IEEE MICRO Top Picks]

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

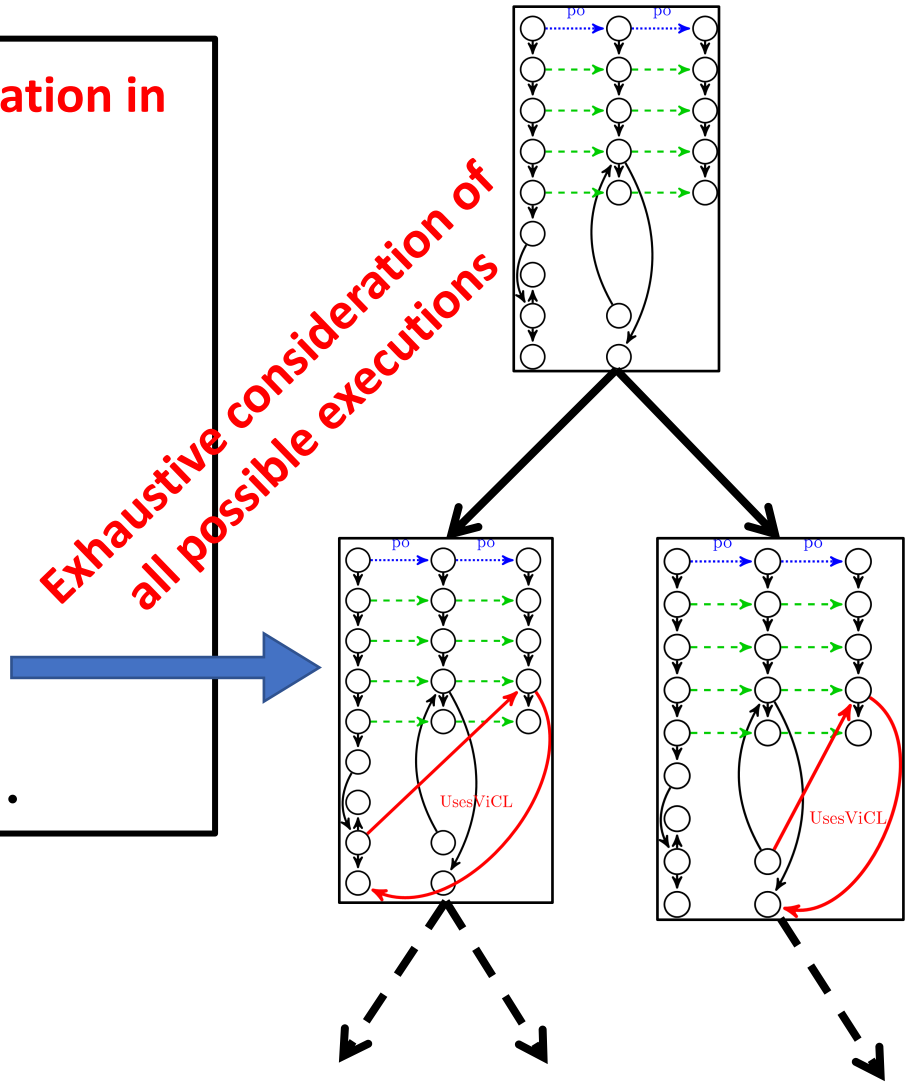
- Axiomatic specifications -> Happens-before graphs
- Check Happens-Before Graphs via Efficient SMT solvers
  - Cyclic => A->B->C->A… Can't happen
  - Acyclic => Scenario is observable

For more info: check.cs.Princeton.edu

# Check: Formal, Axiomatic Models and Interfaces

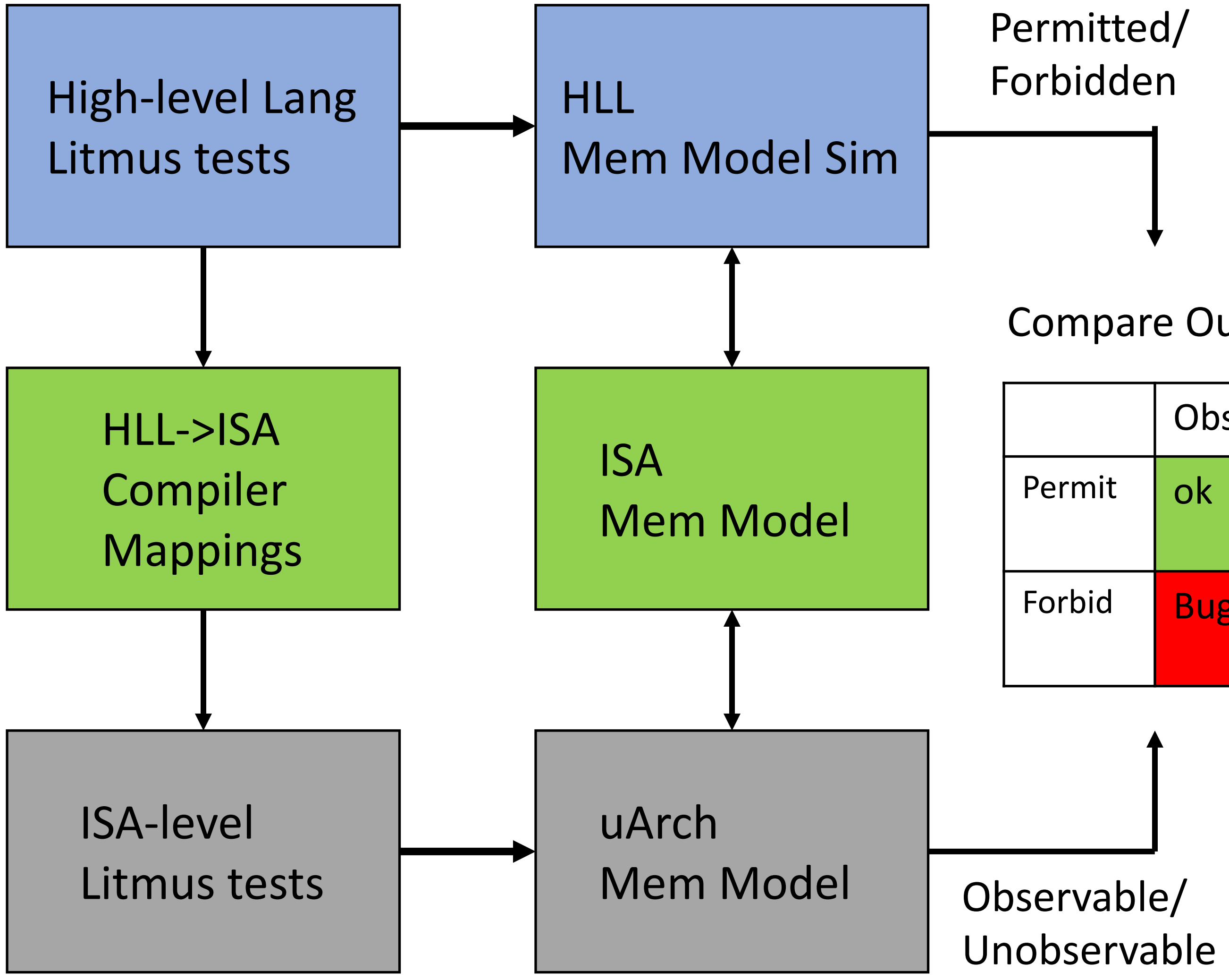
```
Axiom "PO_Fetch":
forall microops "i1",
forall microops "i2",
SameCore i1 i2 /\ ProgramOrder i1 i2 =>
  AddEdge ((i1, Fetch), (i2, Fetch), "PO").

Axiom "Execute_stage_is_in_order":
forall microops "i1",
forall microops "i2",
SameCore i1 i2 /\
 EdgeExists ((i1, Fetch), (i2, Fetch)) =>
   AddEdge ((i1, Execute), (i2, Execute), "PPO").
```

**Microarchitecture Specification in *μSpec* DSL**



**Microarchitectural happens-before (μhb) graphs**

# TriCheck Framework: Verifying Memory Event Ordering from Languages to Hardware



Compare Outcomes

| | Obs. | Not obs |
|---|---|---|
| Permit | ok | Over strict |
| Forbid | Bug | ok |

# RISC-V Case Study

- Apply TriCheck to 7 legal RISC-V implementations:
  - All abide by current RISC-V spec
  - Vary in preserved program order and store atomicity

- Results:
  - Impossible to compile C11 for RISC-V as specified.
  - Insufficiently strong fence instructions, load-load reordering, …
  - Out of 1,701 tested C11 programs:
  - RISC-V-Base-compliant design allows 144 buggy outcomes
  - RISC-V-Base+A-compliant design allows 221 buggy outcomes

**Takeaway: Draft RISC-V spec could not serve as a legal C11 compiler target**

**Next Steps: RISC-V Memory Model Working Group formed to address these issues. Ratified a new and formally specified RISC-V memory model that supports C11, Linux, etc.**

## From Memory Consistency Models to Security: What we would like…

| | |
|---|---|
| 1. Specify system to study | Formal interface and specification of given system implementation |
| 2. Specify attack pattern | E.g. Subtle event sequences during program’s execution |
| 3. Synthesis | Either output synthesized attacks. Or determine that none are possible |

# The CheckMate Tool: Automated Attack Discovery & Synthesis

1. Specify system to study

2. Specify attack pattern

3. Synthesis

- **What we did**: Developed a tool to do this, based on the uHB graphs from previous sections.
- **Results**: Automatically synthesized Spectre and Meltdown, as well as two new distinct exploits and many variants.

[Trippel, Lustig, Martonosi. https://arxiv.org/abs/1802.03802]
[Trippel, Lustig, Martonosi. MICRO 2018. October, 2018] http://check.cs.princeton.edu/papers/ctrippel_MICRO51.pdf

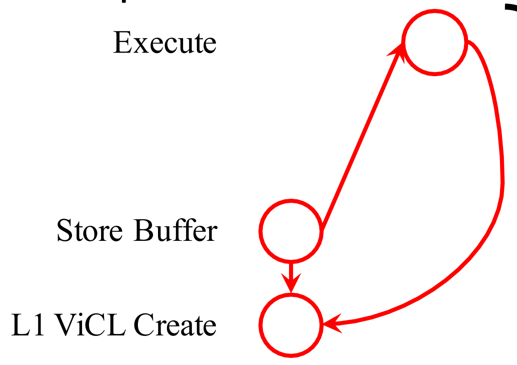
# Microarchitecture-Aware Security Verification

**Microarchitecture**

```
Axiom "PO_Fetch":
forall microops "i1",
forall microops "i2",
SameCore i1 i2 /\ ProgramOrder i1 i2 =>
  AddEdge ((i1, Fetch), (i2, Fetch), "PO").

Axiom "Execute_stage_is_in_order":
forall microops "i1",
forall microops "i2",
SameCore i1 i2 /\
 EdgeExists ((i1, Fetch), (i2, Fetch)) =>
   AddEdge ((i1, Execute), (i2, Execute), "PPO").
```

+

**μhb Pattern**



**Load being sourced from the store buffer**

+

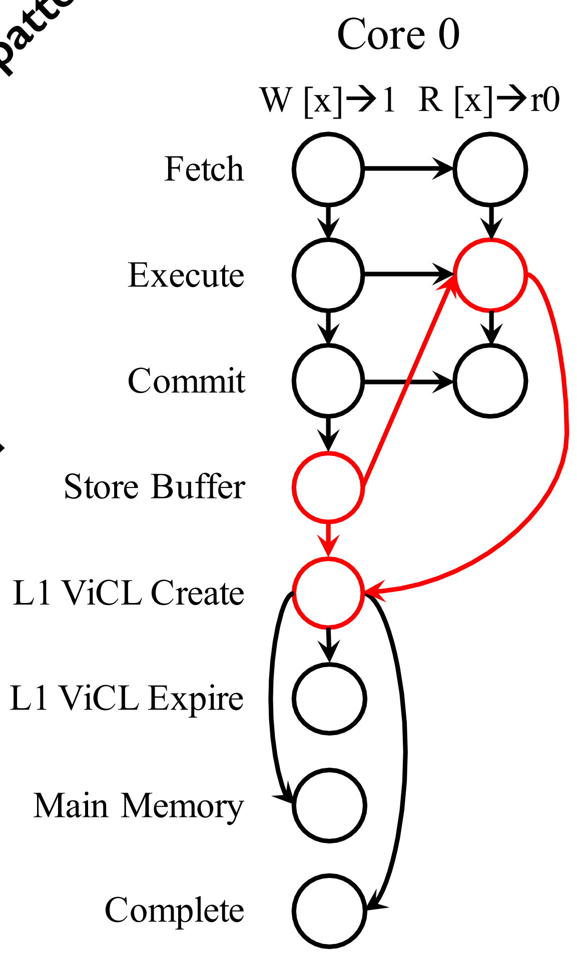
**Execution Constraints**

#cores = 1
#threads = 1
#instr ≤ 2

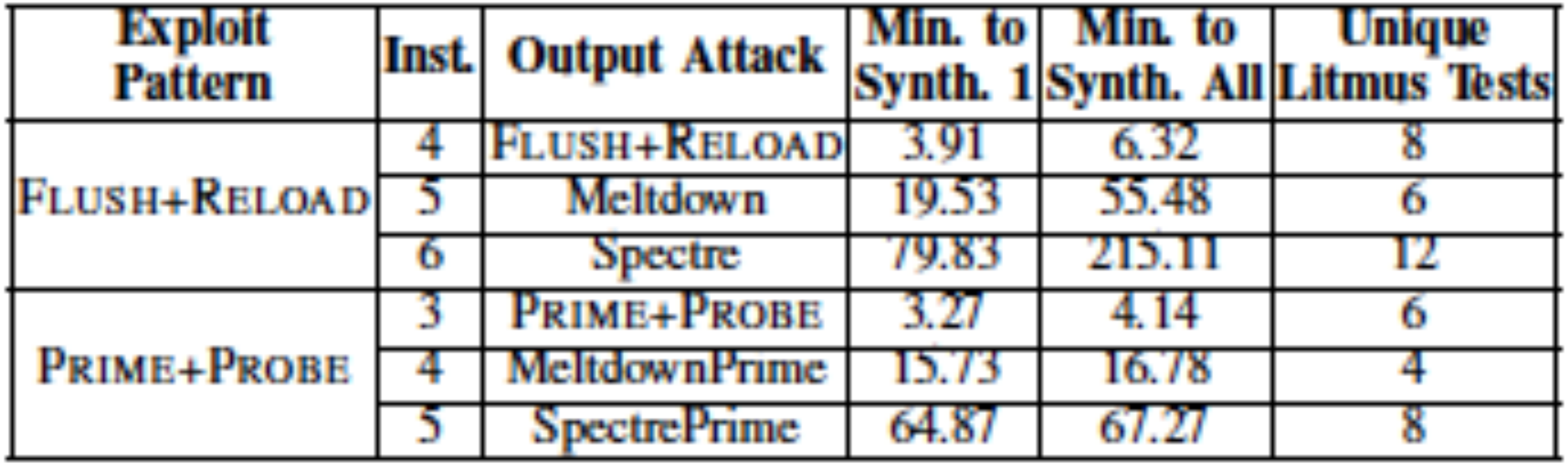
**Check Mate**

**Enumerate all possible execution graphs with pattern**

**μhb Graph**



## Overall Results: What exploits get synthesized? And how long does it take?



| Exploit Pattern | Inst. | Output Attack | Min. to Synth. 1 | Min. to Synth. All | Unique Litmus Tests |
|---|---|---|---|---|---|
| FLUSH+RELOAD | 4 | FLUSH+RELOAD | 3.91 | 6.32 | 8 |
| | 5 | Meltdown | 19.53 | 55.48 | 6 |
| | 6 | Spectre | 79.83 | 215.11 | 12 |
| PRIME+PROBE | 3 | PRIME+PROBE | 3.27 | 4.14 | 6 |
| | 4 | MeltdownPrime | 15.73 | 16.78 | 4 |
| | 5 | SpectrePrime | 64.87 | 67.27 | 8 |

## The Check Suite: Bug Finds and Other Key Takeaways

So far, tools have found bugs in:

- Widely-used Research simulator
- Cache coherence paper
- In-design commercial processors
- RISC-V ISA specification
- Compiler mapping proofs
- IBM XL C++ compiler (fixed in v13.1.5)
- C++ 11 mem model

+ Spectre Prime, MeltdownPrime, and other Vulnerabilities automatically synthesized

### Key Takeaways

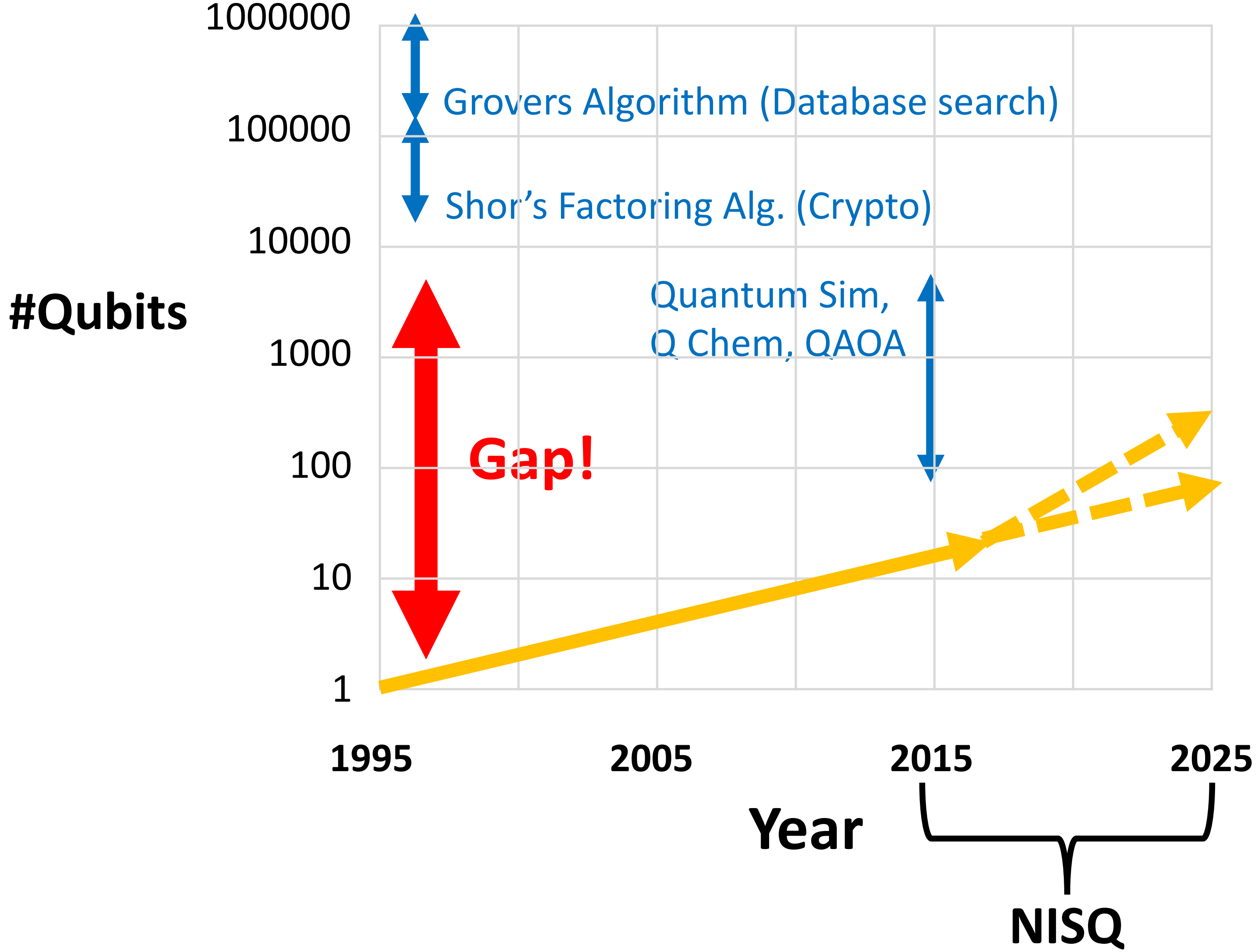
- Need formal, well-specified interfaces
- From well-specified interfaces-> Interaction models and analysis
- From well-specified interfaces -> Reliability and performance metrics

## This Talk: A whirlwind tour of…

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

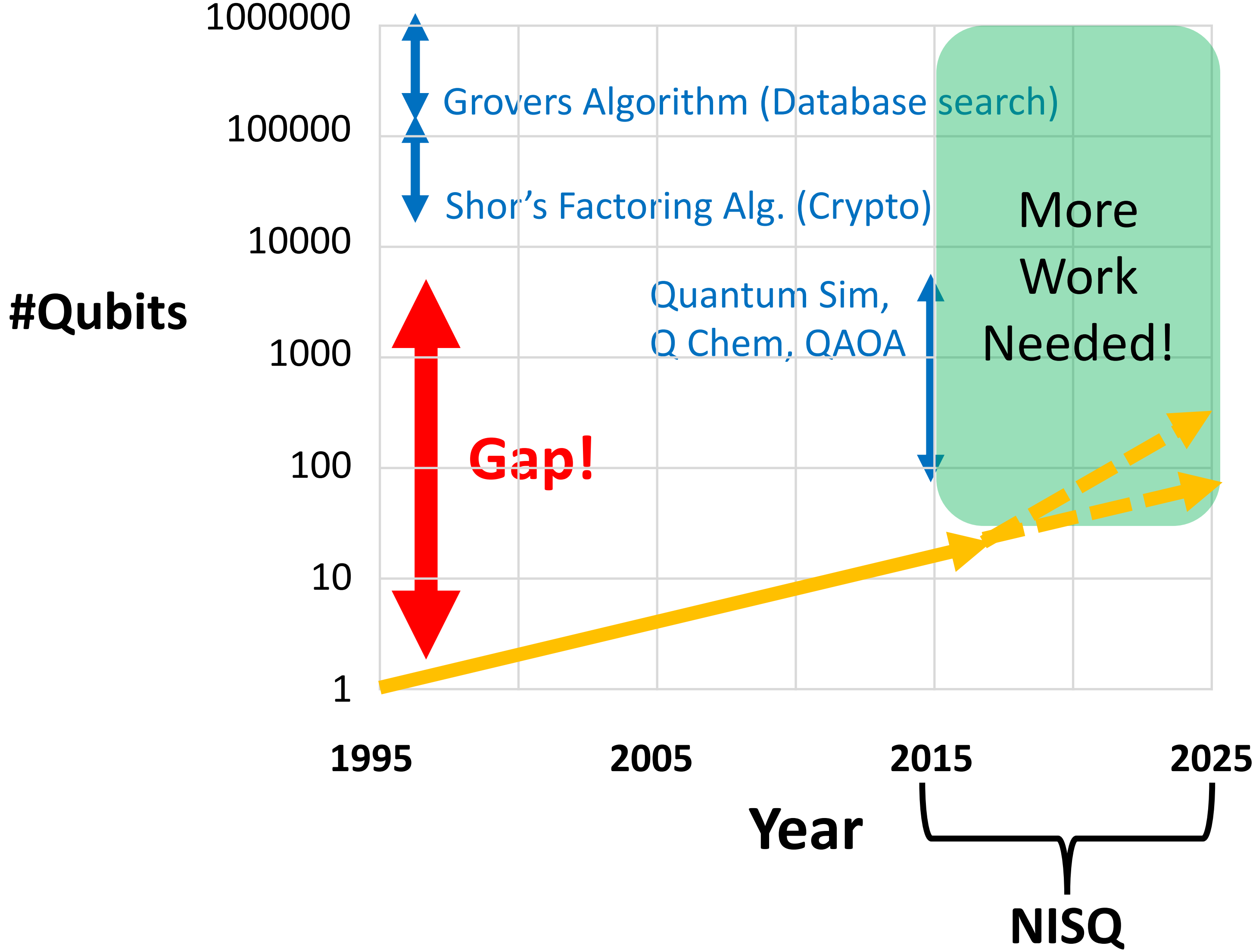
- How to program these highly heterogeneous systems? How to manage the complexity of fast-changing hardware and software?
- How to verify them?
- And what technologies come next?

# QC Algorithms to Machines Gap: Next Ten Years = 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

# An Architect’s Comparative Study of NISQ Machines

**Architectural choices and design questions**

- Gate sets, Connectivity, Noise properties of qubits

**In-depth exploration**

- Real-systems runs across 7 NISQ systems (3 vendors)
- Enabled by our multi-vendor compiler toolflow: fair and accurate comparisons avoiding inefficiencies from vendor compilers

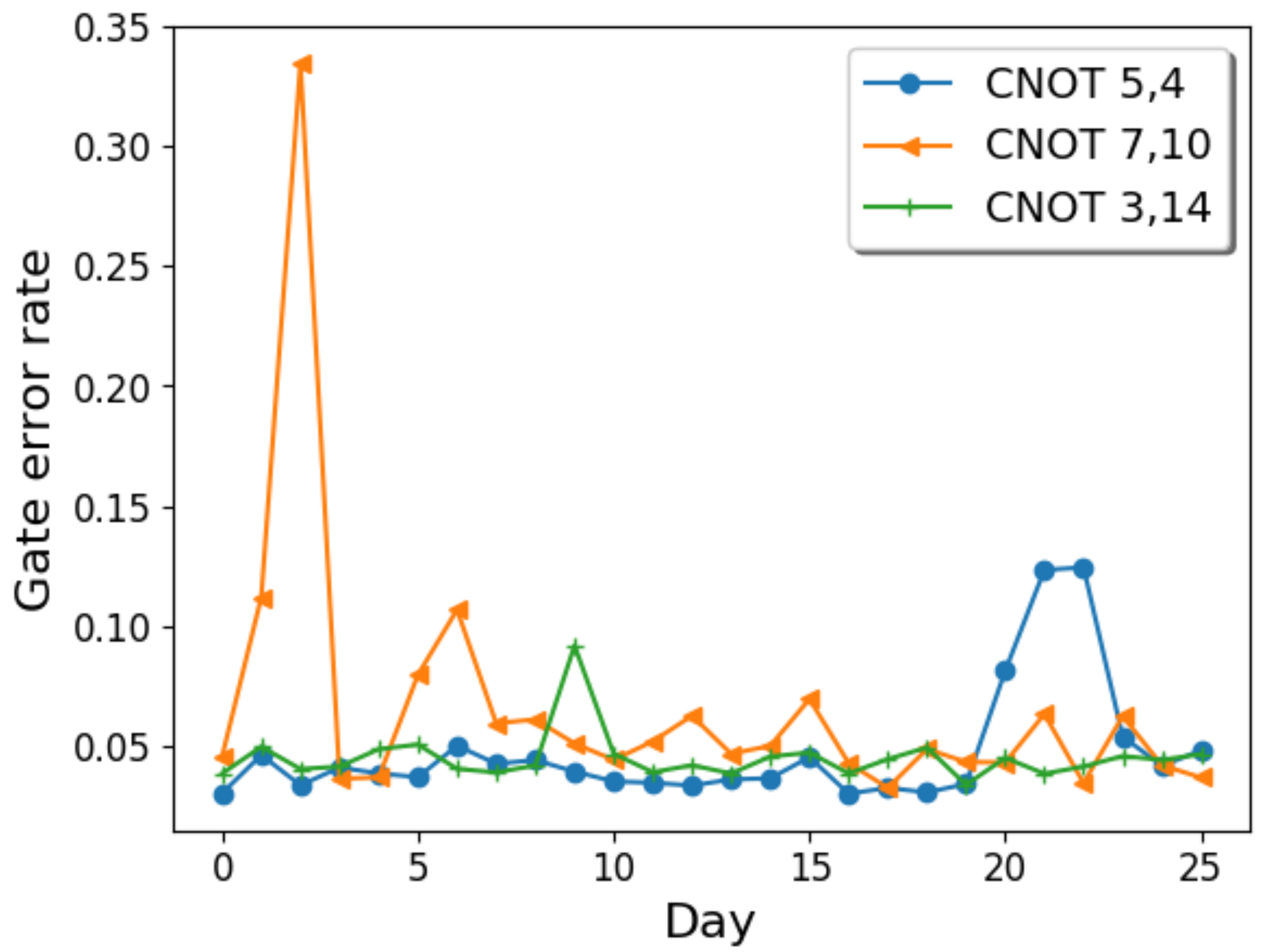
**Design insights from our study**

- Gate set choices
- Connectivity choices
- NISQ execution stack

**[Murali et al. International Symposium on Computer Architecture. June, 2019. [https://arxiv.org/abs/1905.11349](https://arxiv.org/abs/1905.11349) ]**

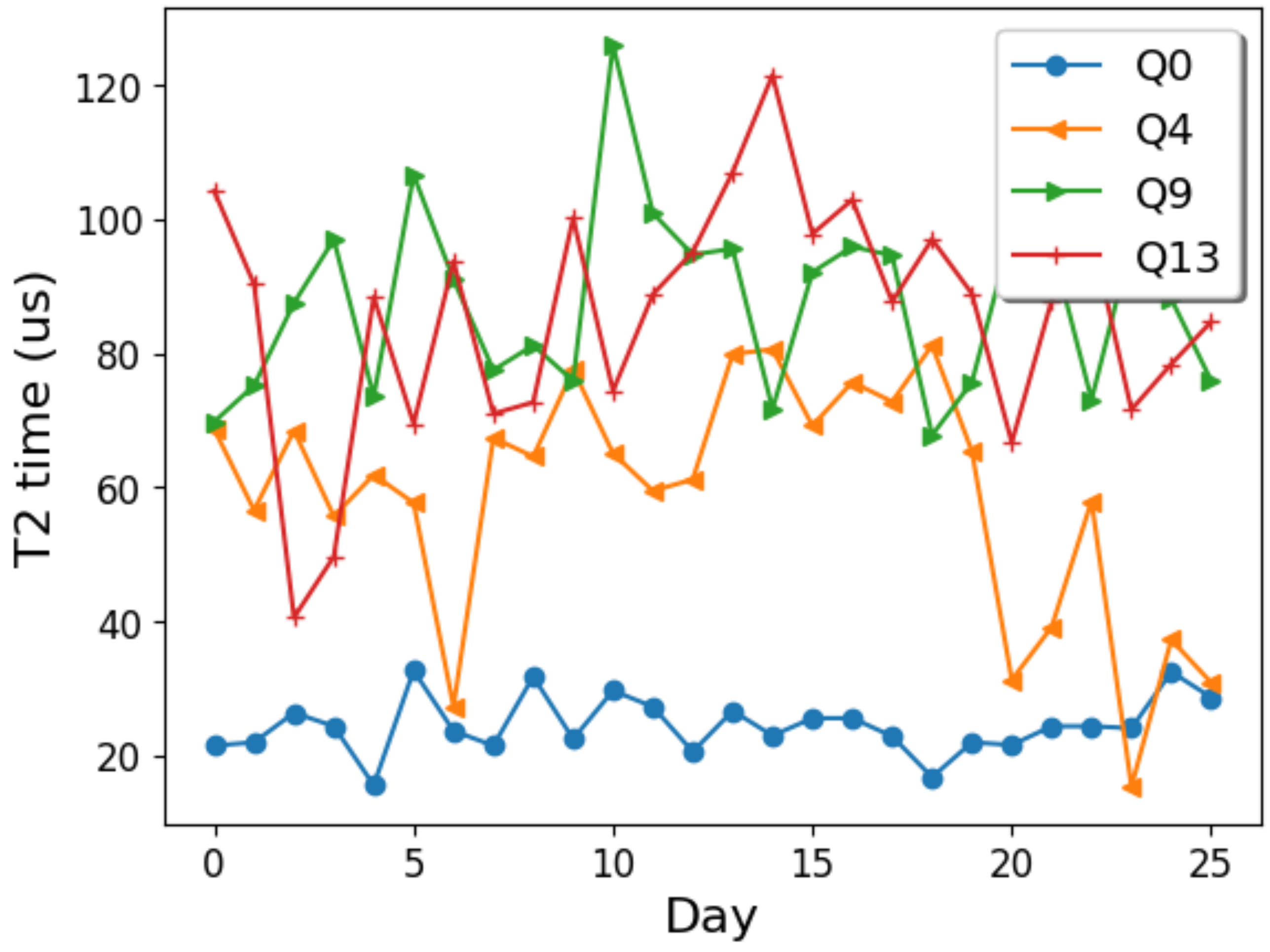
## Large Variations in Machine Characteristics

- Optimized compilation for qubit communication is not sufficient!
- Large spatial and temporal variations in gate error rates!



## Large Variations in Machine Characteristics Part 2

- Large spatial and temporal variations in Qubit Coherence Times
  - How long they hold their state

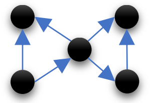
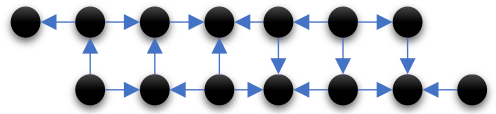
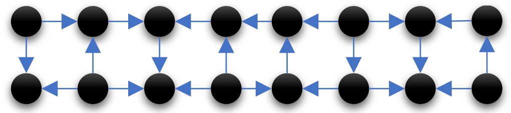
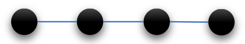
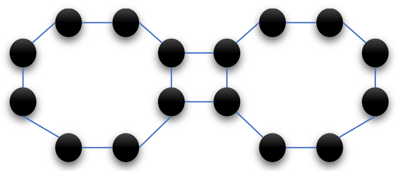
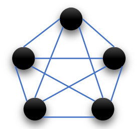


## Programs

| Name | Qubits | Gates |
|---|---|---|
| BV4 | 4 | 12 |
| BV6 | 6 | 12 |
| BV8 | 8 | 18 |
| HS2 | 2 | 16 |
| HS4 | 4 | 28 |
| HS6 | 6 | 42 |
| Toffoli | 3 | 18 |
| Fredkin | 3 | 19 |
| Or | 3 | 17 |
| Peres | 3 | 16 |
| QFT | 2 | 13 |
| Adder | 4 | 23 |

TriQ Compiler

## Machines

| Machine | Qubit Topology |
|---|---|
| IBM Q5 Tenerife |  |
| IBM Q14 Melbourne |  |
| IBM Q16 Rüschlikon |  |
| Rigetti Agave |  |
| Rigetti Aspen1 |  |
| Rigetti Aspen3 | |
| UMD Trapped Ion (UMDTI) |  |

# Putting it all together…

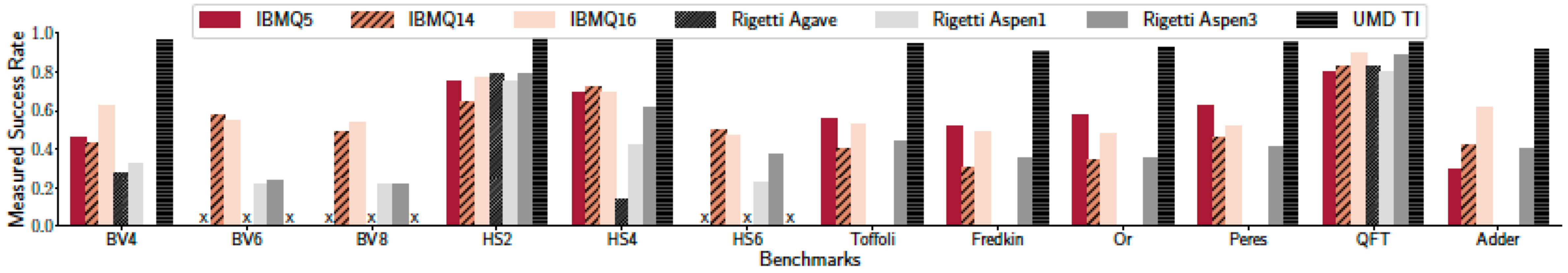
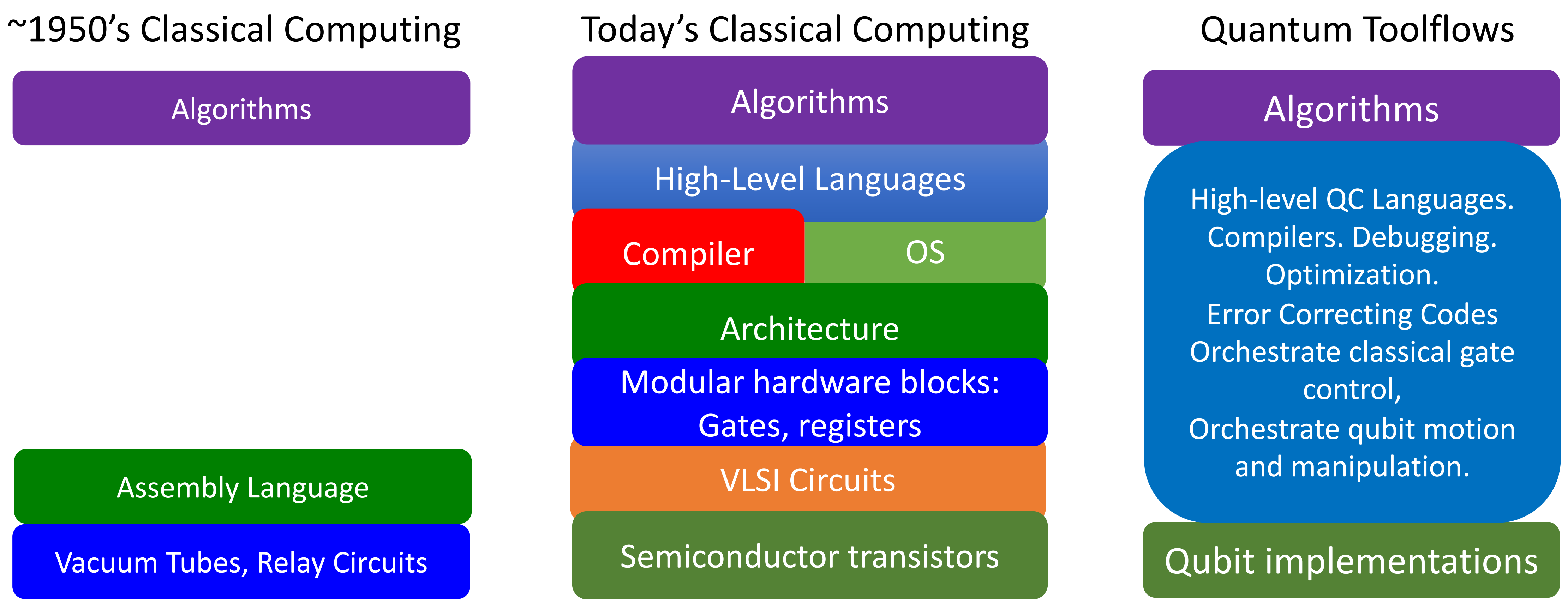
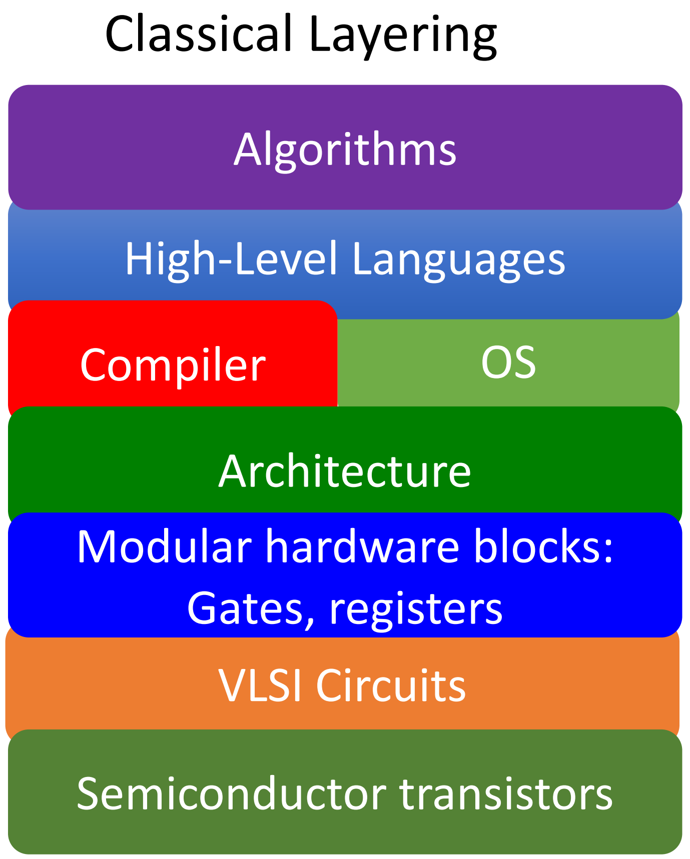


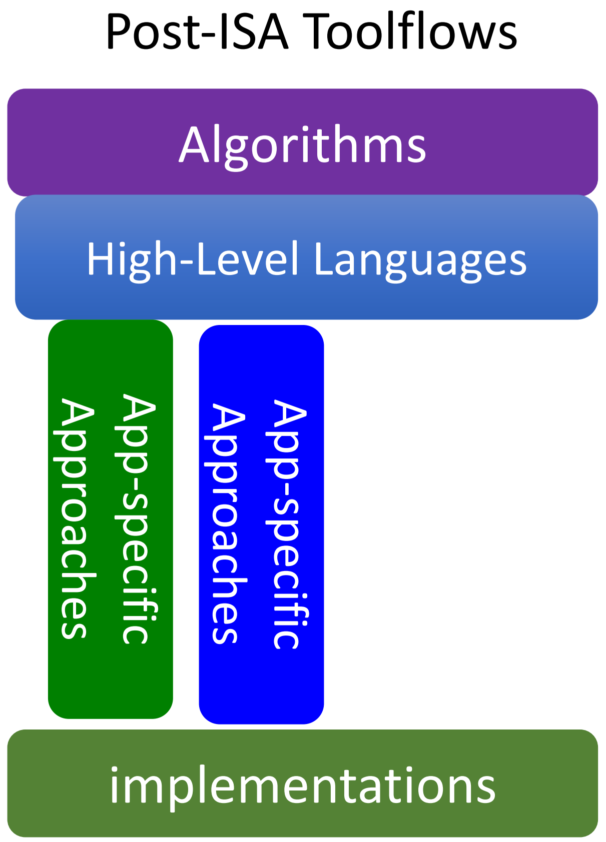
**Figure 12: Success rate for 12 benchmarks on seven systems. Success rates varies drastically across systems and is influenced by error rates, qubit connectivity, and application-machine topology match. Benchmarks that are too large to be mapped onto a machine are marked "X". Runs with zero height bars correspond to failed runs where the correct answer did not dominate in the output distribution.** ***This comparison is intended to understand the impact of architectural design choices such as gate set and connectivity on benchmark performance and is not intended to pick a winning technology, vendor or implementation. Individual benchmark performance numbers may change over time — these measurements represent a snapshot of the performance of these systems when we performed the experiments.***

# Quantum Systems Today: An Analogy


~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 Toolflows
Algorithms
High-level QC Languages. Compilers. Debugging. Optimization.
Error Correcting Codes
Orchestrate classical gate control,
Orchestrate qubit motion and manipulation.
Qubit implementations

# Post-ISA Systems: Automated full-Stack compilation via APIs and formally-spec'd interfaces





Examples:
QC Toolflows

TensorFlow and TPUs

…

Potential for a Seismic Shift in Computer Systems Design

## Thanks to:

**Students and Co-Authors**

**Funding**: NSF, DARPA, SRC CFAR, SRC ADA.

## For more info:

http://check.cs.Princeton.edu

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

The Networking and Information Technology Research and Development (NITRD) Program

**Mailing Address:** NCO/NITRD, 2415 Eisenhower Avenue, Alexandria, VA 22314

**Physical Address:** 490 L'Enfant Plaza SW, Suite 8001, Washington, DC 20024, USA Tel: 202-459-9674, Fax: 202-459-9673, Email: nco@nitrd.gov, Website: https://www.nitrd.gov