

A New Golden Age for Computer Architecture:

Domain-Specific Hardware/Software Co-Design,
Enhanced Security, Open Instruction Sets,
and Agile Chip Development



IEEE-CNSV

Consultants' Network
of Silicon Valley

John Hennessy and David Patterson
Stanford and UC Berkeley

13 June 2018

<https://www.youtube.com/watch?v=3LVeEjsn8Ts>

Outline

Part I: History of
Architecture -
Mainframes,
Minicomputers,
Microprocessors,
RISC vs CISC, VLIW

Part II: Current
Architecture Challenges -
Ending of Dennard Scaling
and Moore's Law, Security

Part III: Future Architecture Opportunities -
Domain Specific Languages and Architecture,
Open Architectures, Agile Hardware Development

IBM Compatibility Problem in Early 1960s

By early 1960's, *IBM had 4 incompatible lines of computers!*

701 → 7094

650 → 7074

702 → 7080

1401 → 7010

Each system had its own:

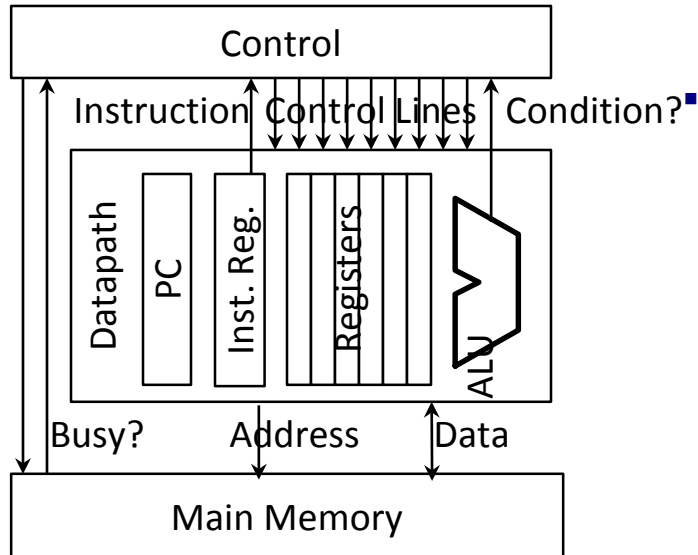
- Instruction set architecture (ISA)
- I/O system and Secondary Storage:
magnetic tapes, drums and disks
- Assemblers, compilers, libraries,...
- Market niche: business, scientific, real time, ...



IBM System/360 – one ISA to rule them all

Control versus Datapath

- Processor designs split between *datapath*, where numbers are stored and arithmetic operations computed, and *control*, which sequences operations on datapath
- Biggest challenge for computer designers was getting control correct



▪ **Maurice Wilkes** invented the idea of *microprogramming* to design the control unit of a processor*



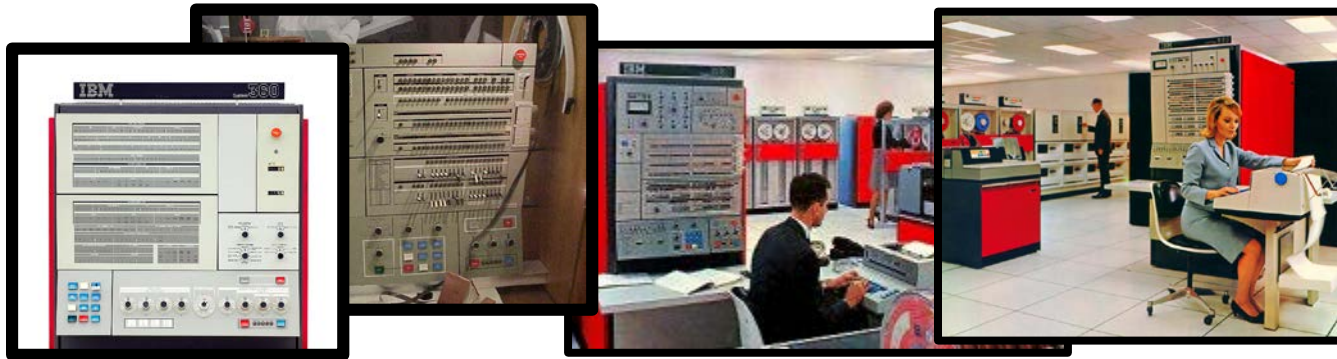
- Logic expensive vs. ROM or RAM
- ROM cheaper than RAM
- ROM much faster than RAM

* "[Micro-programming and the design of the control circuits in an electronic digital computer.](#)"

M. Wilkes, and J. Stringer. *Mathematical Proc. of the Cambridge Philosophical Society*, Vol. 49, 1953.

Microprogramming in IBM 360

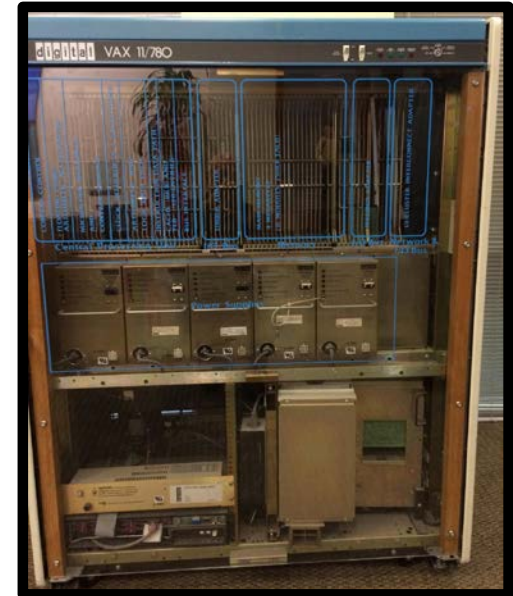
Model	M30	M40	M50	M65
Datapath width	8 bits	16 bits	32 bits	64 bits
Microcode size	4k x 50	4k x 52	2.75k x 85	2.75k x 87
Clock cycle time (ROM)	750 ns	625 ns	500 ns	200 ns
Main memory cycle time	1500 ns	2500 ns	2000 ns	750 ns
Price (1964 \$)	\$192,000	\$216,000	\$460,000	\$1,080,000
Price (2018 \$)	\$1,560,000	\$1,760,000	\$3,720,000	\$8,720,000



Fred Brooks, Jr.

IC Technology, Microcode, and CISC

- Logic, RAM, ROM all implemented using same transistors
- Semiconductor RAM \approx same speed as ROM
- With Moore's Law, memory for control store could grow
- Since RAM, easier to fix microcode bugs
- Allowed more complicated ISAs (CISC)
- Minicomputer (TTL server) example:
 - Digital Equipment Corp. (DEC)
 - VAX ISA in 1977
- 5K x 96b microcode



Writable Control Store

- If Control Store is RAM, then could tailor “*firmware*” to application: “*Writable Control Store*”
- Microprogramming became popular in academia
 - Patterson PhD thesis*
 - SIGMICRO was for microprogramming**
- Xerox Alto (Bit Slice TTL) in 1973
 - 1st computer with Graphical User Interface & Ethernet
 - BitBlt and Ethernet controller in microcode

* *Verification of microprograms*, David Patterson, UCLA, 1976

** “[The design of a system for the synthesis of correct microprograms,](#)”

David Patterson, *Proc. 8th Annual Workshop of Microprogramming*, 1975



Chuck Thacker

Microprocessor Evolution

- Rapid progress in 1970s, fueled by advances in MOS technology, imitated minicomputers and mainframe ISAs
- “Microprocessor Wars”: compete by adding instructions (easy for microcode), justified given assembly language programming
- Intel iAPX 432: Most ambitious 1970s micro, started in 1975
 - 32-bit capability-based object-oriented architecture, custom OS written in Ada
 - Severe performance, complexity (multiple chips), and usability problems; announced 1981
- Intel 8086 (1978, 8MHz, 29,000 transistors)
 - “Stopgap” 16-bit processor, 52 weeks to new chip
 - ISA architected in 3 weeks (10 person weeks) assembly-compatible with 8 bit 8080
- IBM PC 1981 picks Intel 8088 for 8-bit bus (and Motorola 68000 was late)
- Estimated PC sales: 250,000
- Actual PC sales: 100,000,000 \Rightarrow 8086 “overnight” success
- Binary compatibility of PC software \Rightarrow bright future for 8086



Analyzing Microcoded Machines 1980s

- World changed to HLL programming from assembly
 - Compilers now source of measurements
- John Cocke group at IBM
 - Worked on a simple pipelined processor, 801 minicomputer (ECL server), and advanced compilers inside IBM
 - Ported their compiler to IBM 370, only used simple register-register and load/store instructions (similar to 801)
 - Up to 3X faster than existing compilers that used full 370 ISA!
- Emer and Clark at DEC in early 1980s*
 - Found VAX 11/180 average clock cycles per instruction (CPI) = 10!
 - Found 20% of VAX ISA \Rightarrow 60% of microcode, but only 0.2% of execution time!
- Patterson after '79 DEC sabbatical: repair microcode bugs in microprocessors?**
 - What's magic about ISA interpreter in Writable Control Store? Why not other programs?



John Cocke

* "[A Characterization of Processor Performance in the VAX-11/780](#)," J. Emer and D. Clark, *ISCA*, 1984.

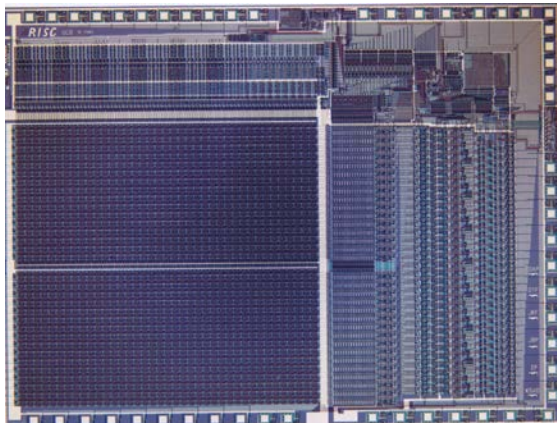
** "[RISCy History](#)," David Patterson, May 30, 2018, Computer Architecture Today Blog

From CISC to RISC

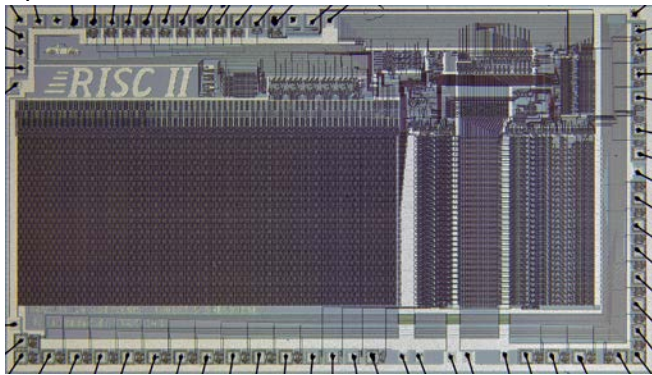
- Use SRAM for instruction *cache* of user-visible instructions
 - Contents of fast instruction memory change to what application needs now vs. ISA interpreter
- Use simple ISA
 - Instructions as simple as microinstructions, but not as wide
 - Compiled code only used a few CISC instructions anyways
 - Enable pipelined implementations
- Further benefit with chip integration
 - In early '80s, could finally fit 32-bit datapath + small caches on a single chip
- Chaitin's register allocation scheme* benefits load-store ISAs

*Chaitin, Gregory J., et al. "[Register allocation via coloring](#)." *Computer languages* 6.1 (1981), 47-57.

Berkeley & Stanford RISC Chips



RISC-I (1982) Contains 44,420 transistors, fabbed in 5 μm NMOS, with a die area of 77 mm^2 , ran at 1 MHz

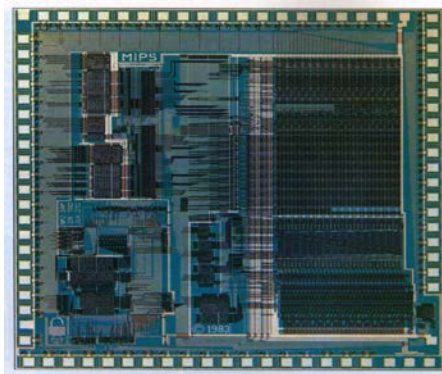


RISC-II (1983) contains 40,760 transistors, was fabbed in 3 μm NMOS, ran at 3 MHz, and the size is 60 mm^2



Fitzpatrick, Daniel, John Foderaro, Manolis Katevenis, Howard Landman, David Patterson, James Peek, Zvi Peshkess, Carlo Séquin, Robert Sherburne, and Korbin Van Dyke. "[A RISCy approach to VLSI.](#)" *ACM SIGARCH Computer Architecture News* 10, no. 1 (1982):

Hennessy, John, Norman Jouppi, Steven Przybylski, Christopher Rowen, Thomas Gross, Forest Baskett, and John Gill. "[MIPS: A microprocessor architecture.](#)" In *ACM SIGMICRO Newsletter*, vol. 13, no. 4, (1982).



Stanford MIPS (1983) contains 25,000 transistors, was fabbed in 3 μm & 4 μm NMOS, ran at 4 MHz (3 μm), and size is 50 mm^2 (4 μm) (Microprocessor without Interlocked Pipeline Stages)



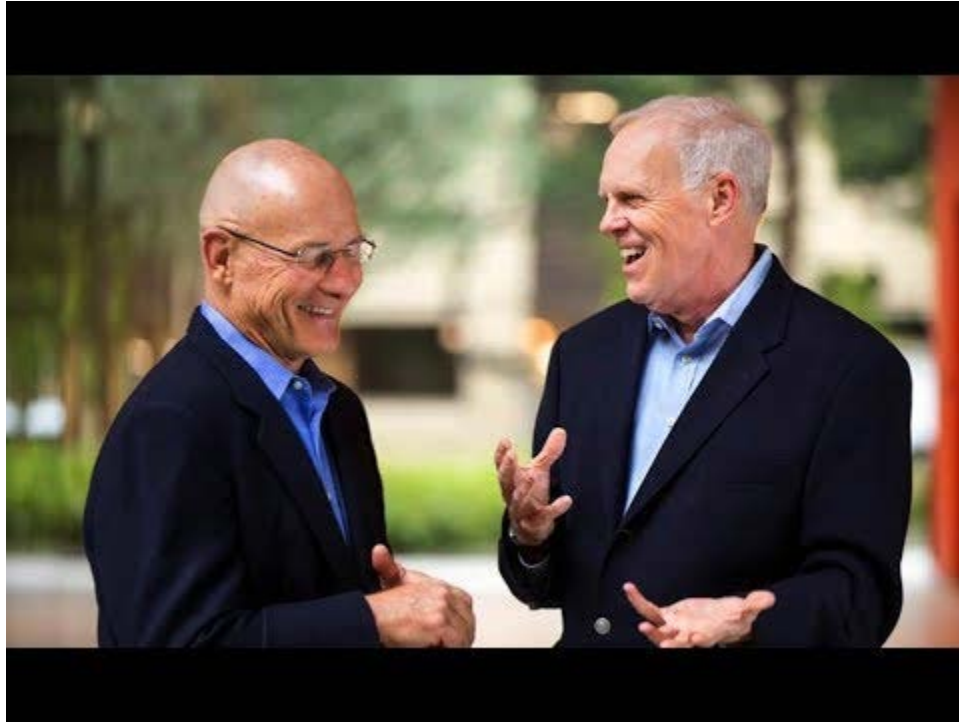
“Iron Law” of Processor Performance: How RISC can win

$$\frac{\text{Time}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Clock cycles}}{\text{Instruction}} * \frac{\text{Time}}{\text{Clock cycle}}$$

- CISC executes fewer instructions per program ($\approx 3/4X$ instructions),
but many more clock cycles per instruction ($\approx 6X$ CPI)
 \Rightarrow RISC $\approx 4X$ faster than CISC

[“Performance from architecture: comparing a RISC and a CISC with similar hardware organization,”](#) Dileep Bhandarkar and Douglas Clark, *Proc. Symposium, ASPLOS*, 1991.

Video of RISC History*



*Full ACM video is at <http://bit.ly/2KKItJ5>

CISC vs. RISC Today

PC Era

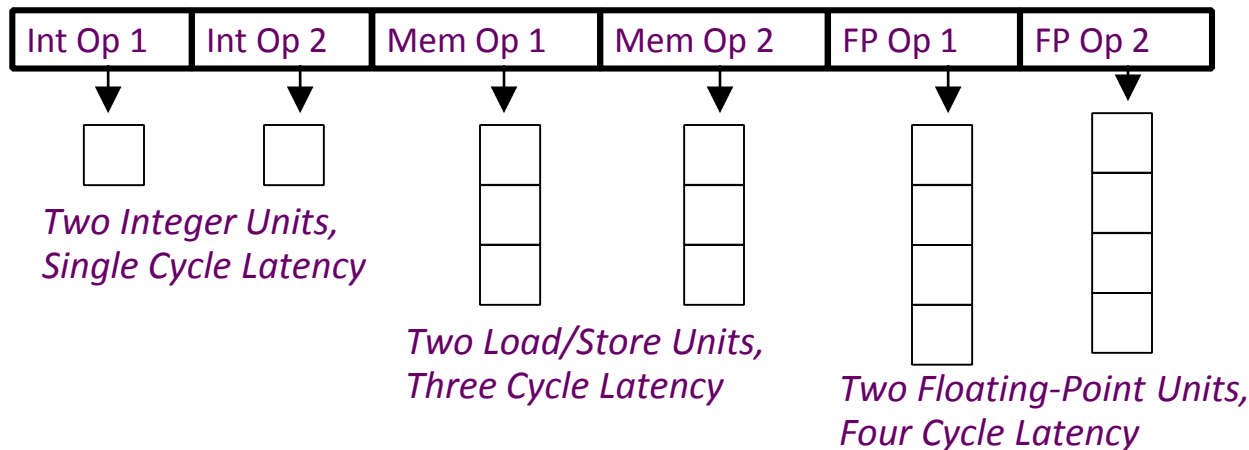
- Hardware translates x86 instructions into internal RISC instructions
- Then use any RISC technique inside MPU
- > 350M / year !
- x86 ISA eventually dominates servers as well as desktops

PostPC Era: Client/Cloud

- IP in SoC vs. MPU
- Value die area, energy as much as performance
- > 20B total / year in 2017
 - x86 in PCs peaks in 2011, now decline ~8% / year (2016 < 2007)
 - x86 servers \Rightarrow Cloud ~10M servers total* (0.05% of 20B)
- 99% Processors today are RISC

*[“A Decade of Mobile Computing”](#), Vijay Reddi, 7/21/17, *Computer Architecture Today*

VLIW: Very Long Instruction Word (Josh Fisher)



- Multiple operations packed into one instruction (like a wide microinstruction)
- Each operation slot is for a fixed function
- Constant operation latencies are specified
- Architecture requires guarantee of:
 - Parallelism within an instruction \Rightarrow no cross-operation RAW check
 - No data use before data ready \Rightarrow no data interlocks

From RISC to Intel/HP Itanium, EPIC IA-64

- EPIC is Intel's name for their VLIW architecture
 - “Explicitly Parallel Instruction Computing”
 - A binary object-code-compatible VLIW
 - Developed jointly with HP starting 1994
- IA-64 was Intel's chosen 64b ISA successor to 32b x86
 - IA-64 = Intel Architecture 64-bit
 - AMD wouldn't be able to make IA-64, unlike x86, so had to make 64-bit x86
- First chip late (2001 vs 1997), but eventually delivered (2002)
- Many companies gave up RISC for Itanium since it was widely believed to be inevitable (Microsoft, SGI, Hitachi, Bull, ...)



VLIW Issues and an “EPIC Failure”

- Compiler couldn't handle complex dependencies in integer code (pointers)
- Code size explosion
- Unpredictable branches
- Variable memory latency (unpredictable cache misses)
 - Out of Order techniques dealt with cache latencies
- Out of Order subsumed VLIW benefits
- *“The Itanium approach...was supposed to be so terrific –until it turned out that the wished-for compilers were basically impossible to write.”*
 - Donald Knuth, Stanford
- Pundits noted delays and under performance of Itanium product ridiculed by the chip industry



Itanium \Rightarrow “Itanic” (like infamous ship *Titanic*)

Summary Part I: Consensus on ISAs Today



- Not CISC: no new general-purpose CISC ISA in 30 years
- Not VLIW: no new general-purpose VLIW ISA in 15 years.
VLIW has failed in general-purpose computing arena
 - Complex VLIW architectures close to in-order superscalar in complexity, no real advantage on large complex apps
 - Although VLIWs successful in embedded DSP market
(Simpler VLIWs, easier branches, no caches, smaller programs)
- RISC! Widely agreed (still) that RISC principles are best for general purpose ISA!

Outline

Part I: History of
Architecture -
*Mainframes,
Minicomputers,
Microprocessors,
RISC vs CISC, VLIW*

Part II: Current
Architecture Challenges -
*Ending of Dennard Scaling
and Moore's Law, Security*

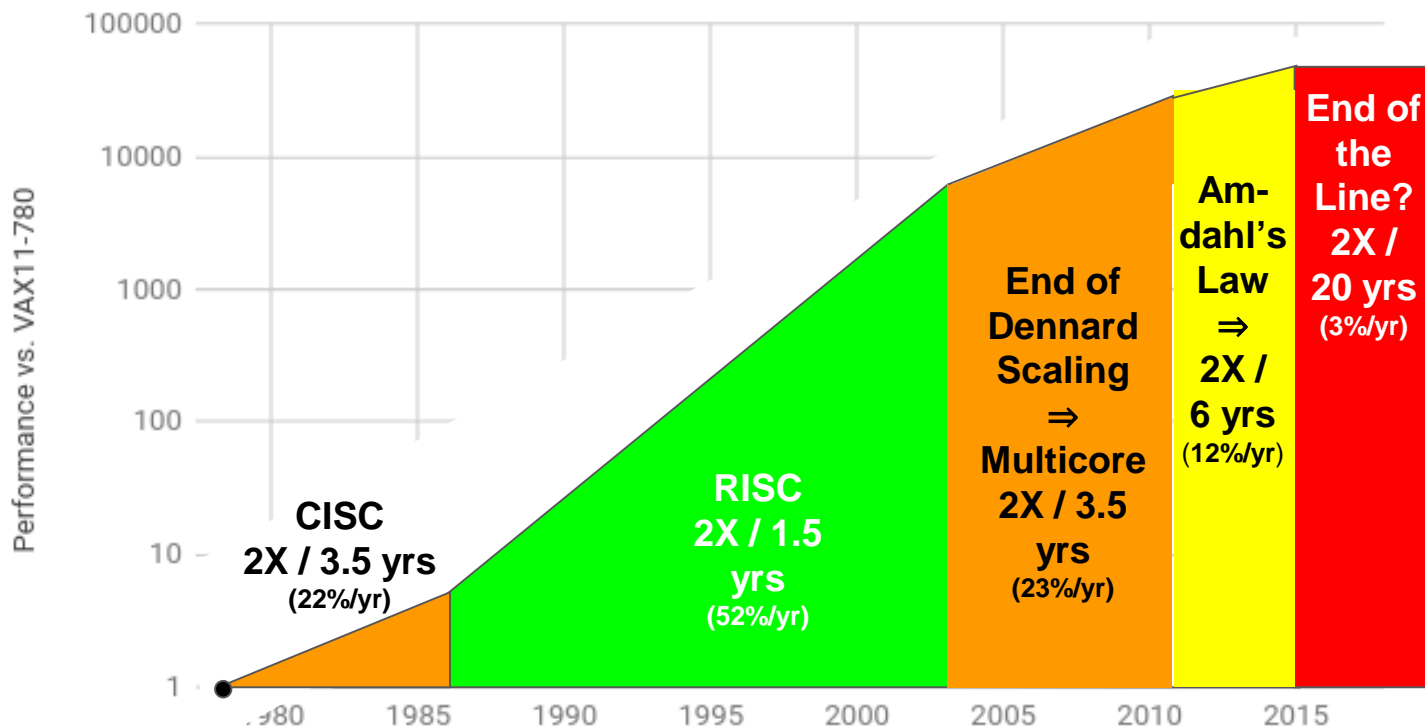
Part III: Future Architecture Opportunities -
*Domain Specific Languages and Architecture,
Open Architectures, Agile Hardware Development*

Fundamental Changes in Technology

- Technology
 - End of Dennard scaling: power becomes the key constraint
 - Ending of Moore's Law: transistors improvement slows
- Architectural
 - Limitation and inefficiencies in exploiting instruction level parallelism end the uniprocessor era in 2004
 - Amdahl's Law and its implications end “easy” multicore era
- Products
 - PC/Server \Rightarrow IoT, Mobile/Cloud

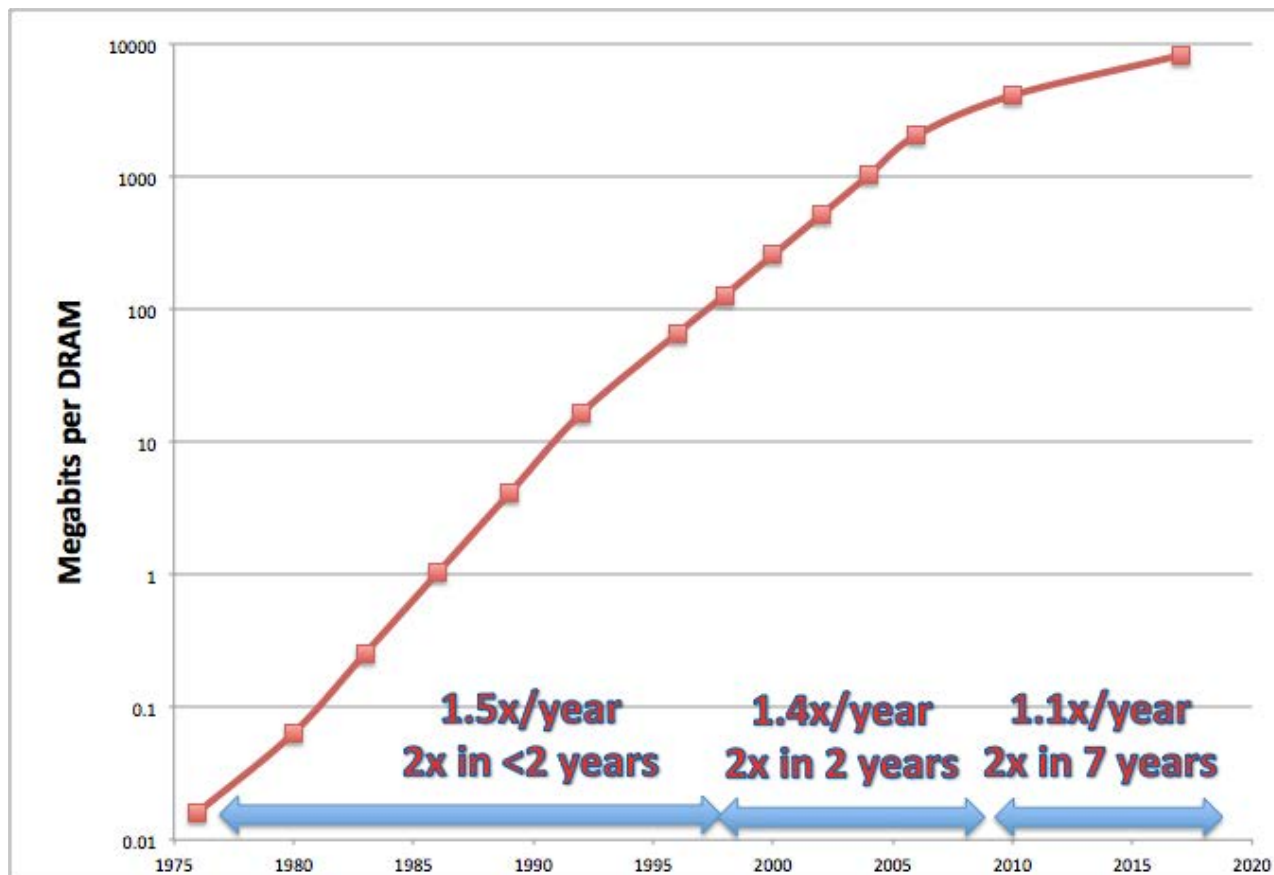
End of Growth of Single Program Speed?

40 years of Processor Performance

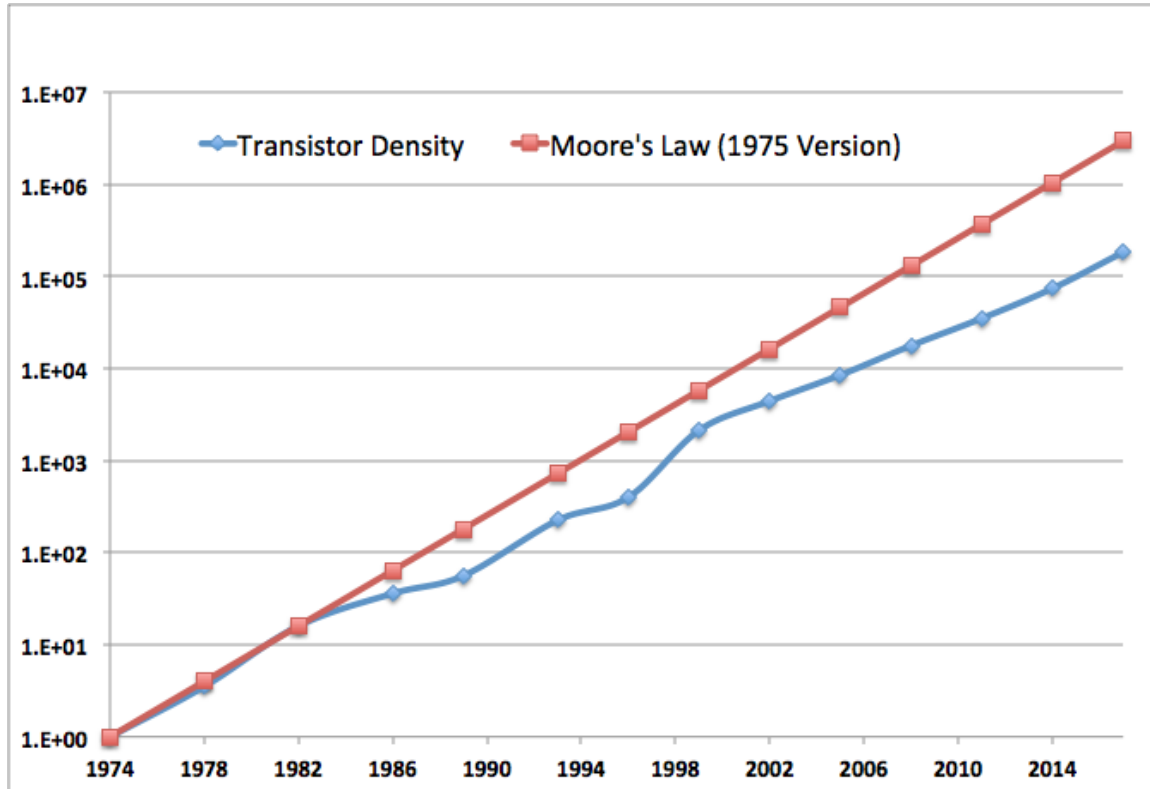


Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

Moore's Law in DRAMs

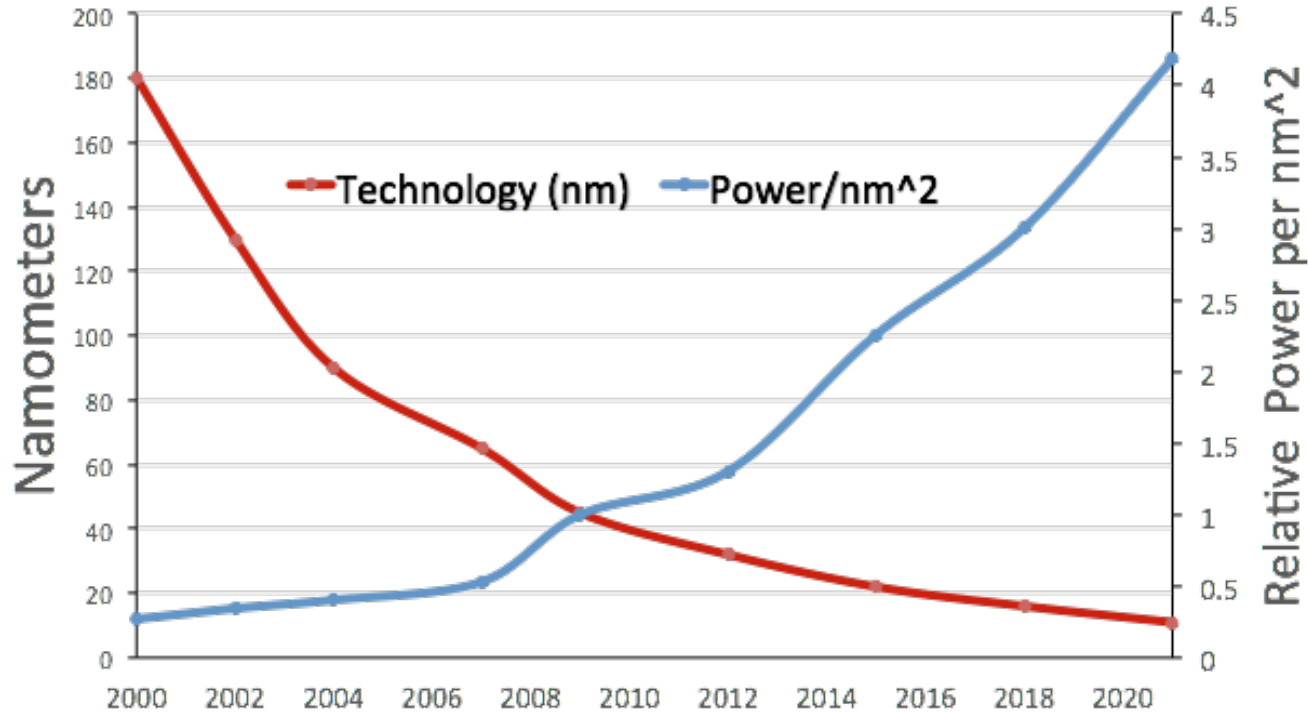


Moore's Law Slowdown in Intel Processors



Cost/transistor
slowing down
faster, due to
fab costs.

Technology & Power: Dennard Scaling



Power consumption based on models in "[Dark Silicon and the End of Multicore Scaling](#)," Hadi Esmaeilzadeh, *ISCA*, 2011

Energy scaling for fixed task is better, since more and faster transistors

Sorry State of Security

- Many protection mechanisms earlier
 - Domains, rings, even capabilities
- Not well used \Rightarrow disappeared
 - Didn't seem to help, and lots of overhead
- Early hope: SW would eliminate attack vectors
 - Perhaps through verification: too hard
 - Kernels and microkernels: explosion in size
- Clearly, not case for almost all software
 - Must build secure systems despite SW bugs!
- Hardware must help with security!

Example of Current State of the Art: x86

- 40+ years of interfaces leading to attack vectors
 - e.g., *Intel Management Engine (ME) processor*
 - Runs firmware management system more privileged than system SW
 - *“Sadly, and most depressing, there is no option for us users to opt-out from having this on our computing devices, whether we want it or not. The author considers this as probably the biggest mistake the PC industry has got itself into she has ever witnessed.”**
 - e.g., Fuzz testing of x86 potential opcodes**
 - Unknown instruction: freeze processor despite being in user mode

* [“Intel x86 considered harmful,”](#) Joanna Rutkowska, 2015

** [“Breaking the x86 ISA,”](#) Christopher Domas, 2016

Spectre & Computer Architecture

- Definition of instruction set architecture
 - “*What the machine language programmer must know to properly write a correct but timing-independent program.*”
- Spectre: speculation \Rightarrow timing attacks that leak ≥ 10 kb/s
- More microarchitecture attacks on the way*
- Security via resource Isolation? Turn off multithreading
- Spectre is bug in computer architecture definition vs chip
- Need Computer Architecture 2.0 to prevent timing leaks**

* “[A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware](#),” Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser, *Journal of Cryptographic Engineering*, April, 2018

** “[A Primer on the Meltdown & Spectre Hardware Security Design Flaws and their Important Implications](#)”, Mark Hill, 2/15/18, *Computer Architecture Today*

Part II: Challenges Summary

- Performance improvements are at a standstill
 - Slowing Moore's Law
 - No more Dennard Scaling
 - Microarchitecture techniques: ILP, multicore, etc. are inefficient, hence burn energy
- State of computer security is embarrassing for all of us in the computing field
 - Seems unlikely systems will ever become secure using software only solutions

Outline

Part I: History of
Architecture -
*Mainframes,
Minicomputers,
Microprocessors,
RISC vs CISC, VLIW*

Part II: Current
Architecture Challenges -
*Ending of Dennard Scaling
and Moore's Law, Security*

Part III: Future Architecture Opportunities -
*Domain Specific Languages and Architecture,
Open Architectures, Agile Hardware Development*

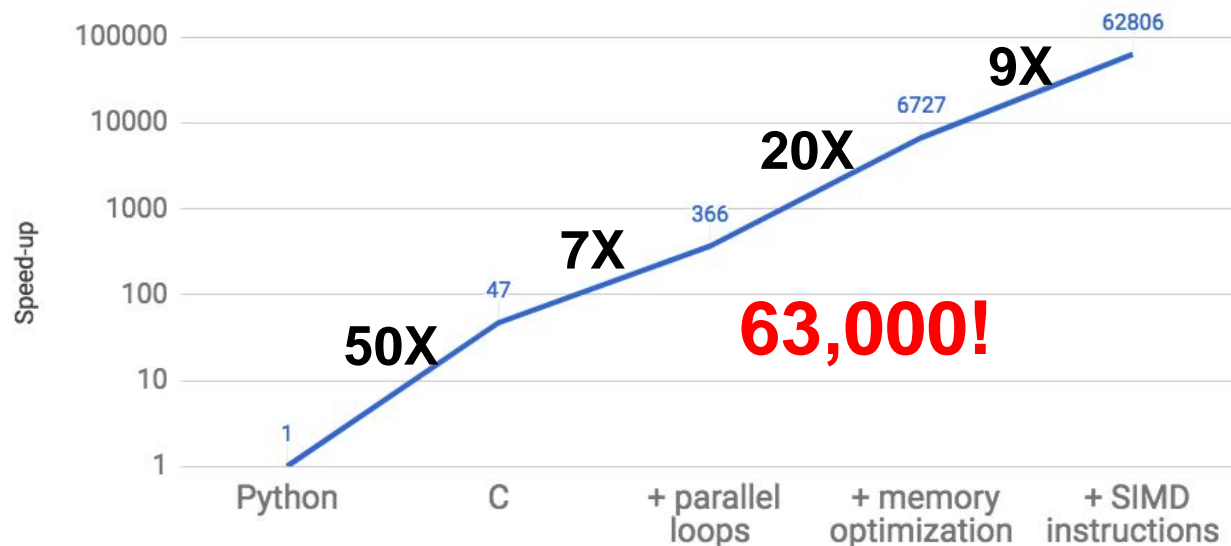
What Opportunities Left?

- SW-centric
 - Modern scripting languages are interpreted, dynamically-typed and encourage reuse
 - Efficient for programmers but not for execution
- HW-centric
 - Only path left is *Domain Specific Architectures*
 - Just do a few tasks, but extremely well
- Combination
 - Domain Specific Languages & Architectures

What's the Opportunity?

Matrix Multiply: relative speedup to a Python version (18 core Intel)

Matrix Multiply Speedup Over Native Python



from: "There's Plenty of Room at the Top," Leiserson, et. al., to appear.

Domain Specific Architectures (DSAs)

- Achieve higher efficiency by tailoring the architecture to characteristics of the domain
 - Not one application, but a domain of applications
 - Different from strict ASIC
 - Requires more domain-specific knowledge than general purpose processors need
- Examples:
 - Neural network processors for machine learning
 - GPUs for graphics, virtual reality
 - Programmable network switches and interfaces

Why DSAs Can Win (no magic)

Tailor the Architecture to the Domain

- More effective parallelism for a specific domain:
 - SIMD vs. MIMD
 - VLIW vs. Speculative, out-of-order
- More effective use of memory bandwidth
 - User controlled versus caches
- Eliminate unneeded accuracy
 - IEEE replaced by lower precision FP
 - 32-64 bit integers to 8-16 bit integers
- Domain specific programming language

Domain Specific Languages

DSAs require targeting of high level operations to the architecture

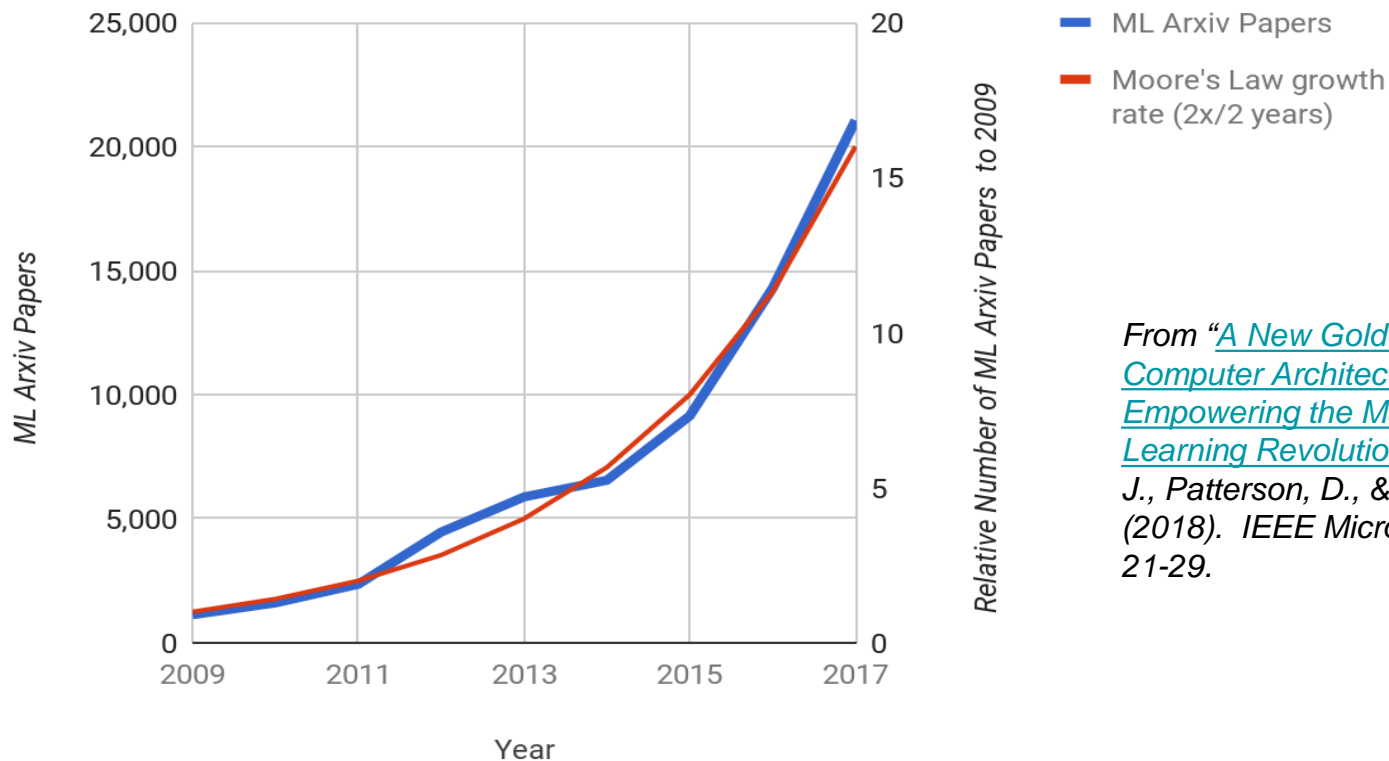
- Hard to start with C or Python-like language and recover structure
- Need matrix, vector, or sparse matrix operations
- Domain Specific Languages specify these operations:
 - OpenGL, TensorFlow, P4
- If DSL programs retain architecture-independence, interesting compiler challenges will exist
 - XLA

[“XLA - TensorFlow, Compiled”](#), XLA Team, March 6, 2017

Research Opportunities

- General-purpose applications:
 - Make Python run like C with compiler + HW
 - Deja vu: make HLLs fast on RISC
- Domain-specific applications (bigger opportunity?)
 - What are the right DSLs for important applications?
 - Codesign of new DSLs and DSAs
 - Advanced compilation techniques for optimizing the matching:
 - New territory: not extraction of high level structure from C/Fortran but matching/optimization
- Challenge: not to compromise DSLs with short-term ISA-specific or microarchitectural-specific compromises

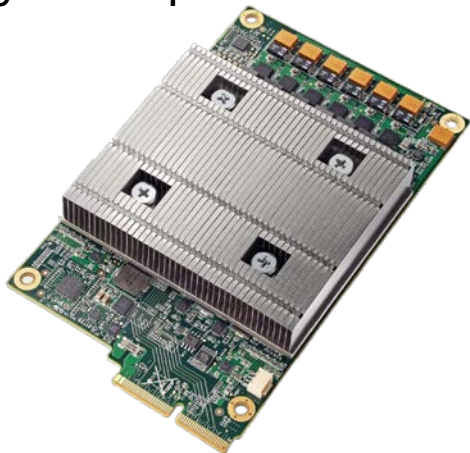
Deep learning is causing a machine learning revolution



From [*"A New Golden Age in Computer Architecture: Empowering the Machine-Learning Revolution."*](#) Dean, J., Patterson, D., & Young, C. (2018). *IEEE Micro*, 38(2), 21-29.

Tensor Processing Unit v1

Google-designed chip for neural net **inference**



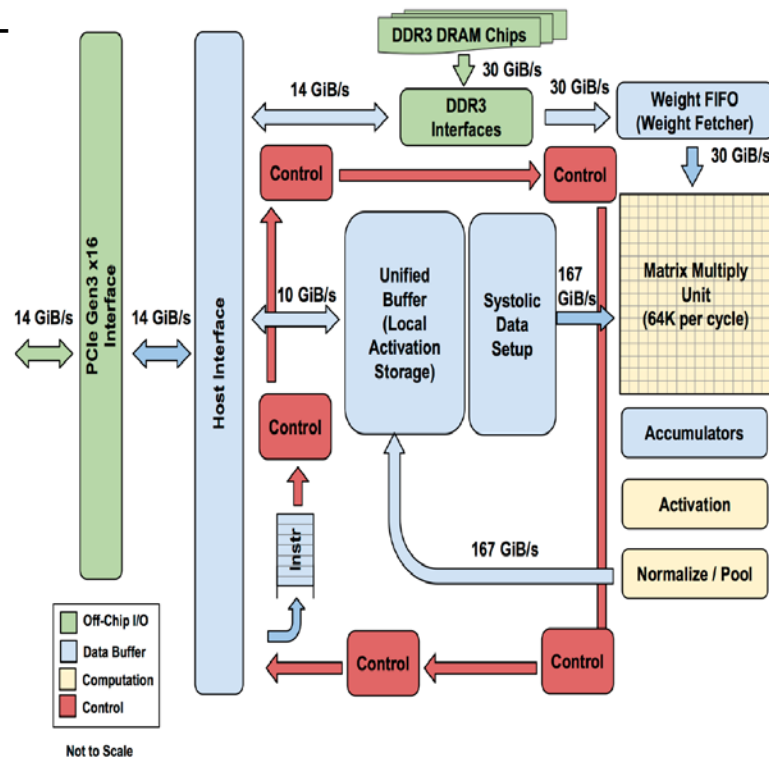
In production use for 36 months: used by billions on search queries, for neural machine translation, for AlphaGo match, ...

[In-Datacenter Performance Analysis of a Tensor Processing Unit](#), Jouppi, Young, Patil, Patterson et al., ISCA 2017,

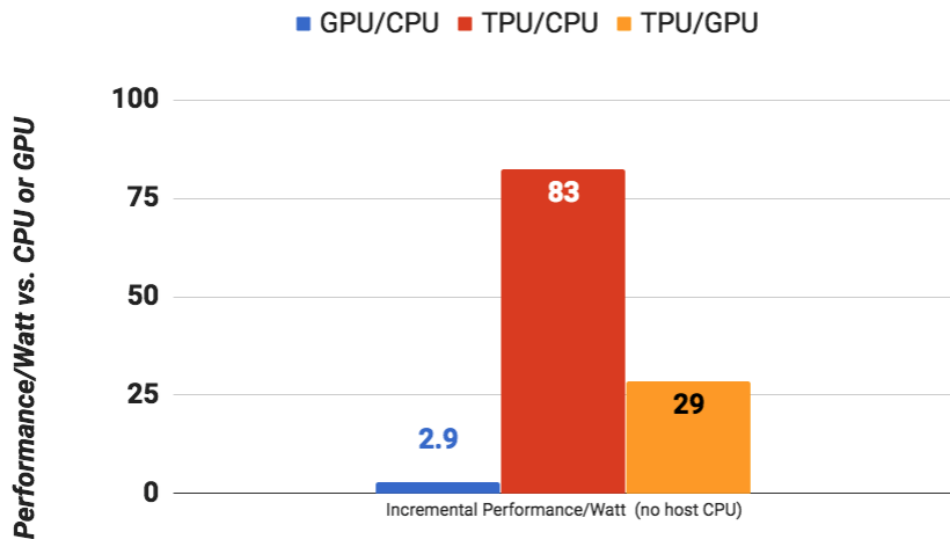


TPU: High-level Chip Architecture

- The Matrix Unit: 65,536 (256x256) 8-bit multiply-accumulate units
- 700 MHz clock rate
- Peak: 92T operations/second
 - $65,536 * 2 * 700M$
- >25X as many MACs vs GPU
- >100X as many MACs vs CPU
- 4 MiB of on-chip Accumulator memory
- 24 MiB of on-chip Unified Buffer (activation memory)
- 3.5X as much on-chip memory vs GPU
- Two 2133MHz DDR3 DRAM channels
- 8 GiB of off-chip weight DRAM memory



Perf/Watt TPU vs CPU & GPU



Measure performance of Machine Learning?

See MLPerf.org (“SPEC for ML”)

- Benchmark suite being developed by
 - ≥ 7 companies and ≥ 5 universities
 - To be released 7/1/18

Part III: DSL/DSA Summary

- Lots of opportunities
- But, new approach to computer architecture is needed.
- The Renaissance computer architecture team is vertically integrated. Understands:
 - Applications
 - DSLs and related compiler technology
 - Principles of architecture
 - Implementation technology
- Everything old is new again!

Part III: Open Architectures

- Why open source compilers and operating systems but not ISAs?
- *What if there were free and open ISAs we could use for everything?*

RISC-V Origin Story

- UC Berkeley Research using x86 & ARM?
 - Impossible – too complex *and* IP issues
- 2010 started “3-month project” to develop own clean-slate ISA
 - Krste Asanovic, Andrew Waterman, Yunsup Lee, Dave Patterson
- 4 years later, released frozen base user spec

Why are outsiders complaining about changes of RISC-V in Berkeley classes?

What's Different About RISC-V?

- Simple
 - Far smaller than proprietary ISAs
 - 2500 pages for x86, ARMv8 manual vs 200 for RISC-V manual
- Clean-slate design
 - 25 years later, so can learn from mistakes of predecessors
 - Avoids μ architecture or technology-dependent features
- Modular
 - Small standard base ISA
 - Multiple standard extensions
- Supports specialization
 - Vast opcode space reserved
- Community designed
 - Base and standard extensions finished
 - Grow via optional extensions vs. incremental required features
- RISC-V Foundation extends ISA for technical reasons
 - vs. private corporation for internal (marketing) reasons

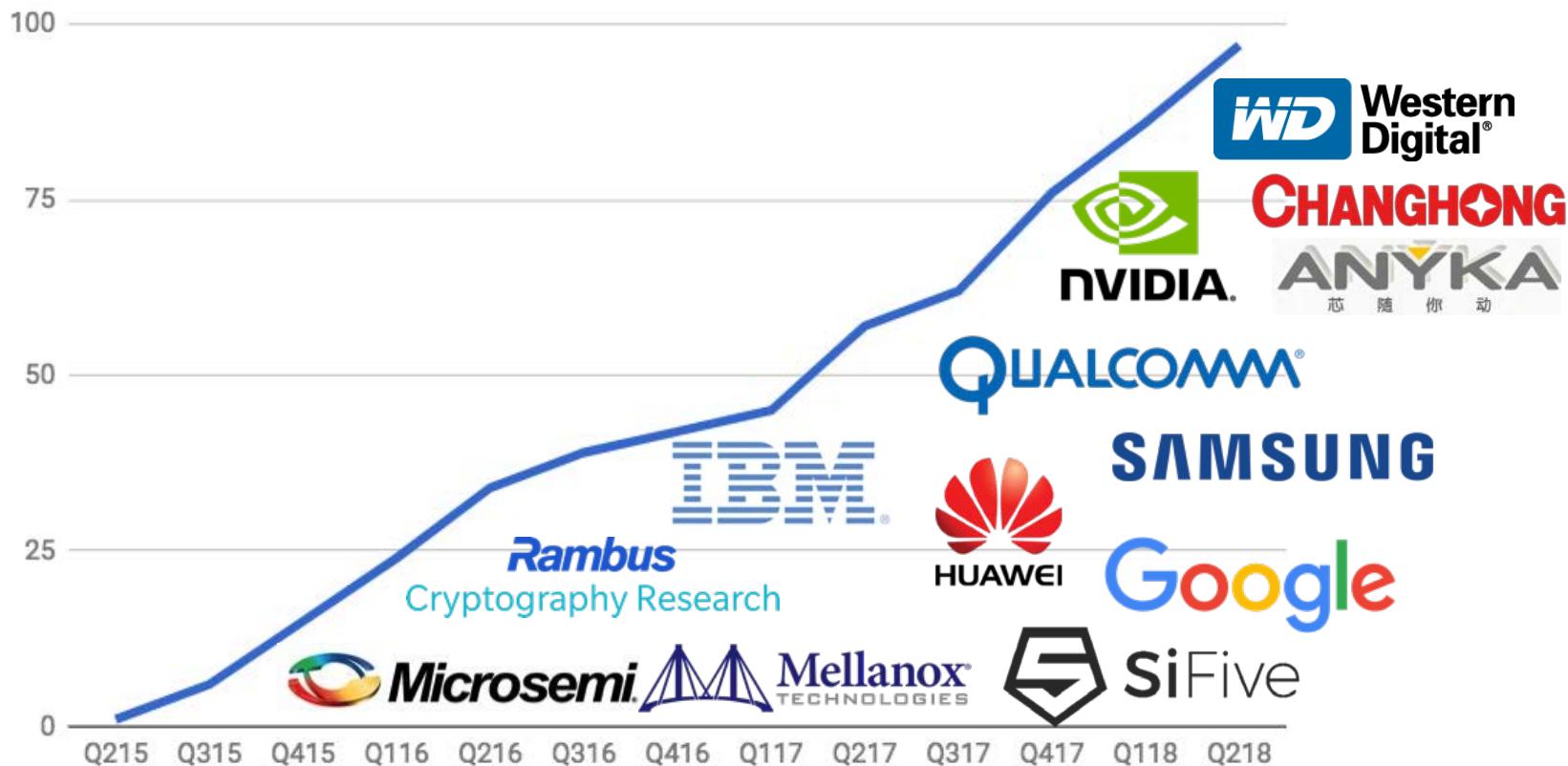
RISC-V Base Plus Standard Extensions

- A few base integer ISAs
 - RV32E, RV32I, RV64I
 - RV32E is 16-reg subset of RV32I
 - <50 hardware instructions in base (Similar to RISC-I!*)
- Standard extensions
 - M: Integer multiply/divide
 - A: Atomic memory operations
 - F/D: Single/Double-precision FI-point
 - C: Compressed Instructions (<x86)
 - V: Vector Extension for DLP (>SIMD**)
- Standard RISC encoding in fixed 32-bit instruction format
- Supported forever by RISC-V Foundation

* [“How close is RISC-V to RISC-I?”](#) David Patterson, 9/19/17, ASPIRE Blog

** [“SIMD Instructions Considered Harmful,”](#) David Patterson and Andrew Waterman, 9/18/17

Foundation Members since 2015

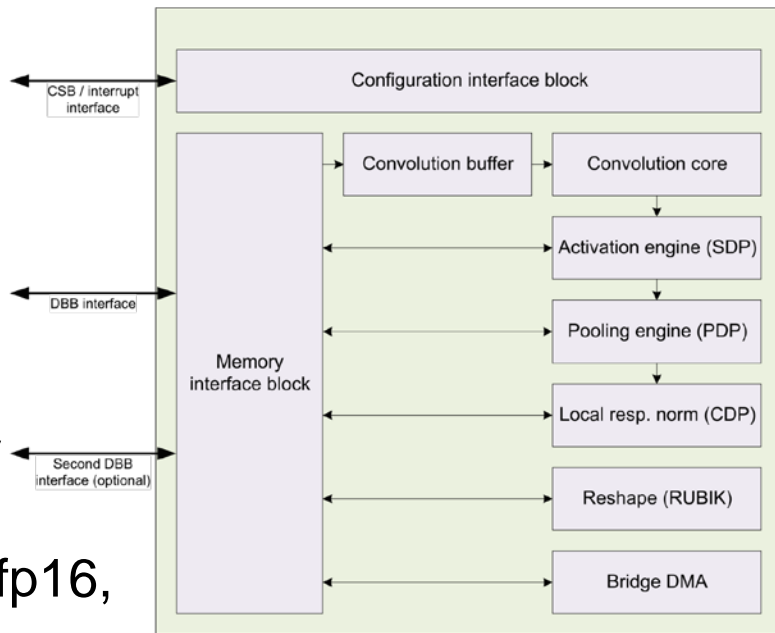


Foundation Working Groups (partial list)



NVDLA: An Open DSA and Implementation

- NVDLA: NVIDIA Deep Learning Accelerator for DNN Inference
- Free & Open: All SW, HW, and documentation on GitHub
- Scalable, configurable design
 - Each block operates independently or in pipeline to bypass memory
 - Data type configurable: int8, int16, fp16,
 - 2D MAC array configurable:
 - 8 to 64 x 4 to 64
 - Size scales 6X (0.5 - 3mm²), power scales 15X (20 - 300 mW)
- RISC-V core as host (optional)



Security and Open Architecture

- Security community likes verifiable (no trap doors*), alterable, free and open architecture and implementations
- Equally important is number of people and organizations performing architecture experiments
 - Want all the best minds to work on security
- Plasticity of FPGAs + open source RISC-V implementations and SW \Rightarrow novel architectures can be deployed online, evaluated, & iterated in weeks vs years (even 100 MHz OK)
- RISC-V may become security exemplar via HW/SW codesign by architects and security experts

* Sturton, C., Hicks, M., Wagner, D., & King, S. T. (2011). "[Defeating UCI: Building stealthy and malicious hardware](#)," *IEEE Symp. on Security and Privacy*.

Part III: Open Architecture Summary

- Free/open architectures \Rightarrow no proprietary lock-in,
no contracts before start, anyone can use/buy/sell
- Typically simpler as not marketing driven (helps verification,
security) > area/power/performance at low end and = at high end
- Readily and freely extensible (support DSAs)
- More organizations designing processors (open source)
 \Rightarrow Faster innovation, more competitive marketplace
- Will become primary experimental vehicle of security experts?

Open Architecture Goal

Create industry-standard open ISAs for all computing devices
“Linux for processors”

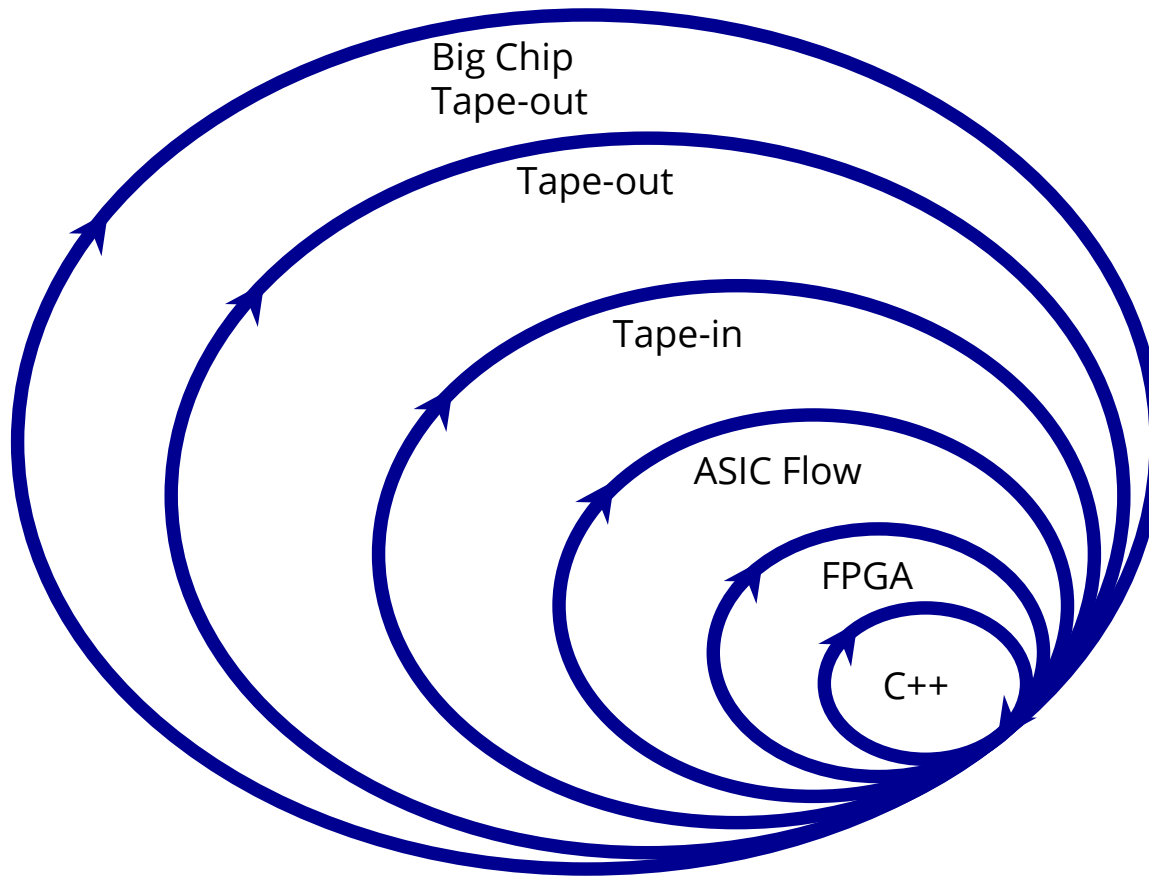
Agile Hardware Development

- Agile: small teams do short development between working but incomplete prototypes and get customer feedback per step
- Scrum team organization
 - 5 - 10 person team size
 - 2 - 4 week sprints for next prototype iteration
- New CAD enables SW Dev techniques to make small teams productive via abstraction & reuse

Reuse: Shared Lines of RTL Code (Chisel)

RISC-V Core	Z-scale	Rocket	BOOM
Description	32-bit 3-stage pipeline in-order 1-instruction issue L1 caches (≈ ARM Cortex-M0)	64-bit, FPU, MMU 5-stage pipeline in-order 1-instruction issue L1 & L2 caches (≈ ARM Cortex-A5)	64-bit, FPU, MMU 5-stage pipeline out-of-order 2-, 3-, or 4- instruction issue L1 & L2 caches (≈ ARM Cortex-A9)
Unique LOC	600 (40%)	1,400 (10%)	9,000 (45%)
LOC all 3 share	500 (30%)	500 (5%)	500 (5%)
LOC Z-scale & Rocket share	500 (30%)	500 (5%)	---
LOC Rocket & BOOM share	---	10,000 (80%)	10,000 (50%)
Total LOC	1,600	12,400	19,500

Agile Hardware Dev. Methodology

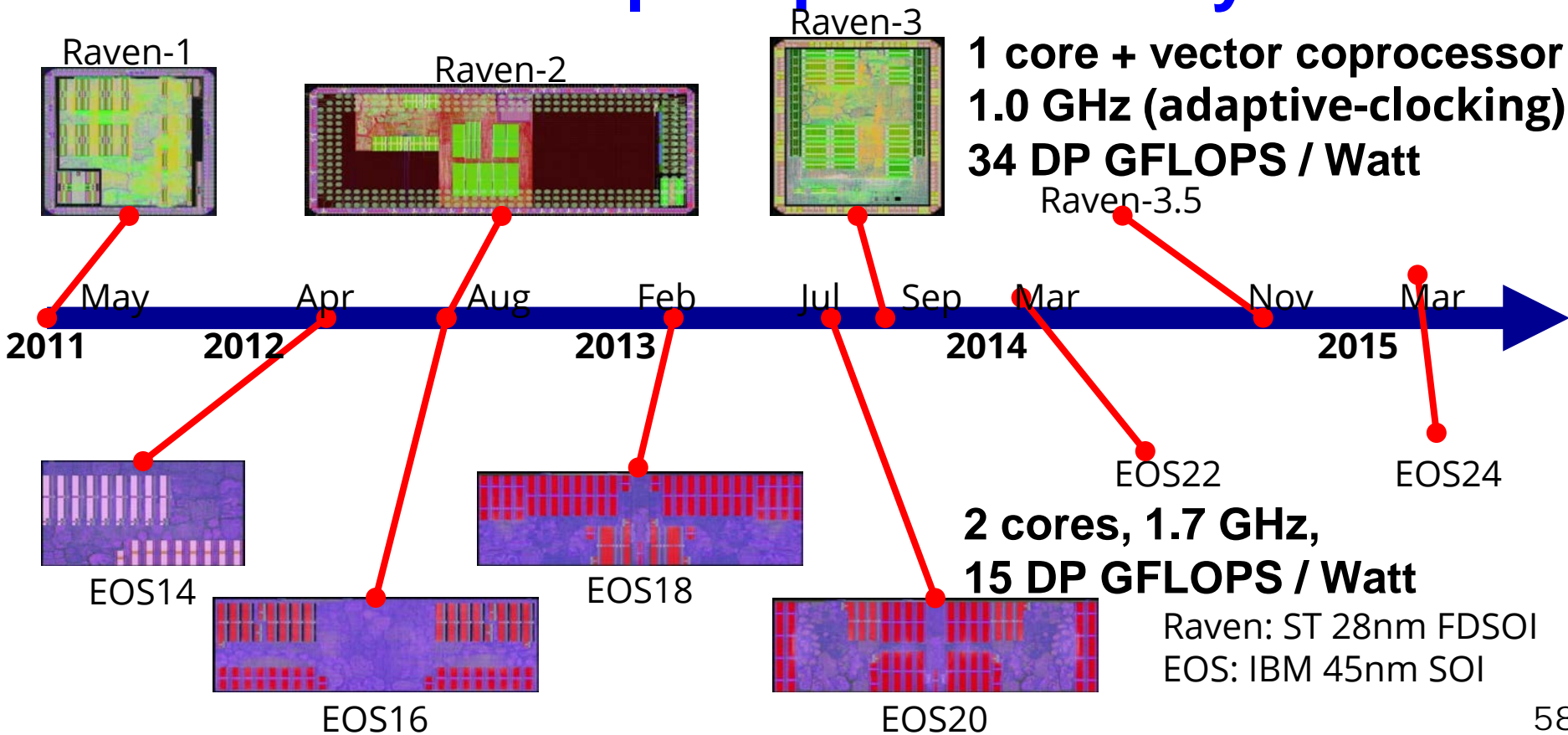


Lee, Y., Waterman, A., Cook, H., Zimmer, B., Keller, B., Puggelli, A., ... & Chiu, P. F. (2016). "[An agile approach to building RISC-V microprocessors.](#)" *IEEE Micro*, 36(2), 8-20.

Small chip
tape-out 100
chips 1x1mm
@ 28nm is
affordable at
\$14,000!

AWS FPGA
F1 instance ⇒
develop new
prototypes
using cloud
(nothing to
buy)

Four 28nm & Six 45nm RISC-V Chips taped out in 5 years



Conclusion: A New Golden Age

- End of Dennard Scaling and Moore's Law
⇒ architecture innovation to improve performance/cost/energy
- Security ⇒ architecture innovation too
- Domain Specific Languages ⇒ Domain Specific Architectures
- Free, open architectures and open source implementations
⇒ everyone can innovate and contribute
- Cloud FPGAs ⇒ all can design and deploy custom “HW”
- Agile HW development ⇒ all can afford to make (small) chips
- Like 1980s, great time for architects in academia & in industry!

Questions?



[“Chip Technology's Friendly Rivals,”](#)

John Markoff,
New York Times,
June 4, 1991

[“RISC Management,”](#)

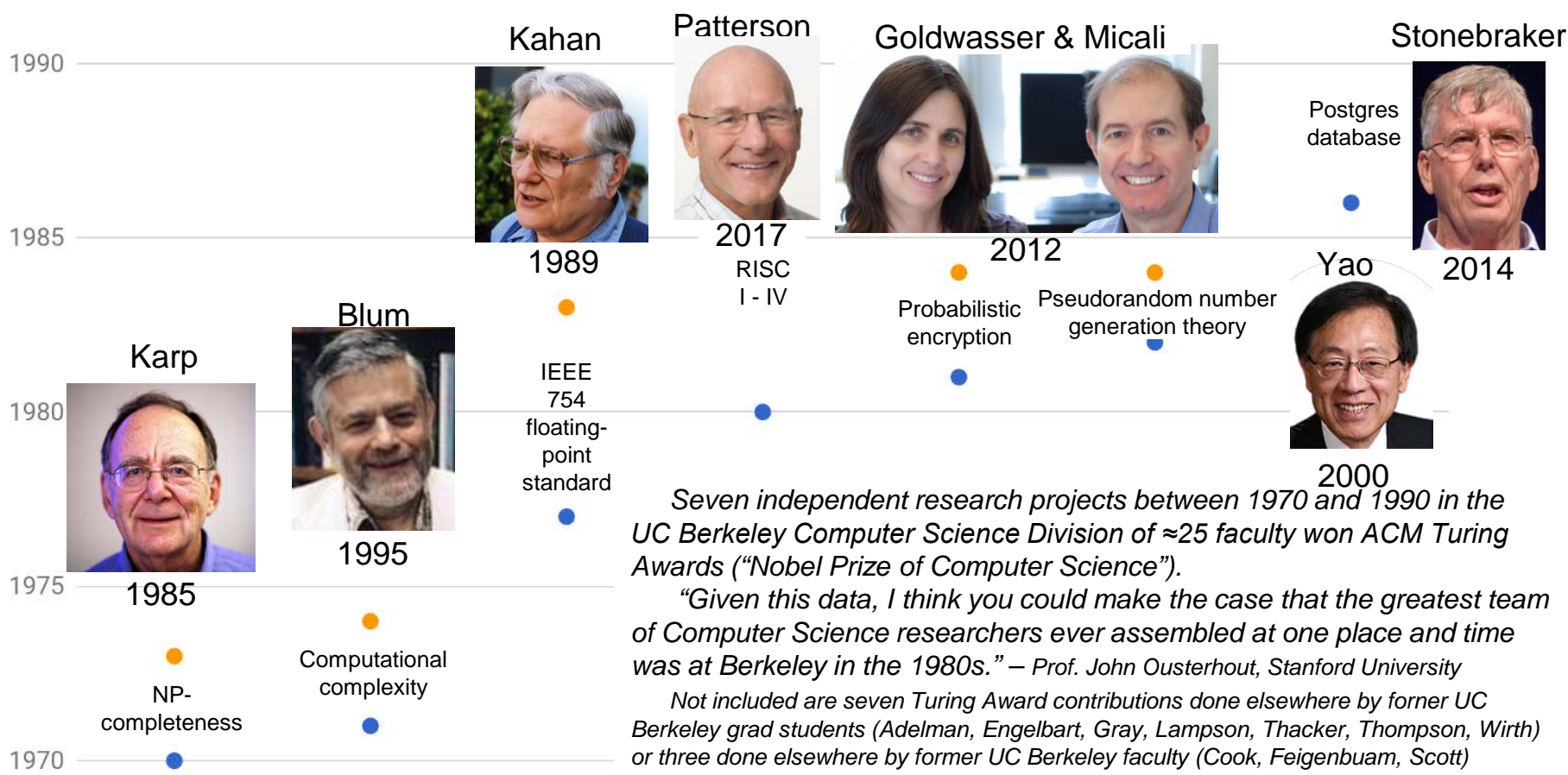
Leah Hoffmann,
CACM, June 2018

[“Rewarded for RISC,”](#)

Neil Savage,
CACM, June 2018

Video: David
Patterson and John
Hennessy, 2017 ACM
A.M. Turing Award
[https://cacm.acm.org/
videos/2017-acm-
turing-award](https://cacm.acm.org/videos/2017-acm-turing-award)

UC Berkeley CS Division Turing Award Projects 1970-90



Seven independent research projects between 1970 and 1990 in the UC Berkeley Computer Science Division of ~25 faculty won ACM Turing Awards (“Nobel Prize of Computer Science”).

“Given this data, I think you could make the case that the greatest team of Computer Science researchers ever assembled at one place and time was at Berkeley in the 1980s.” – Prof. John Ousterhout, Stanford University

Not included are seven Turing Award contributions done elsewhere by former UC Berkeley grad students (Adelman, Engelbart, Gray, Lampson, Thacker, Thompson, Wirth) or three done elsewhere by former UC Berkeley faculty (Cook, Feigenbaum, Scott)