The End of Moore's Law, CPUs (as we know them), and the Rise of Domain Specific Architectures

John Hennessy
Stanford University
Alphabet
September 2019

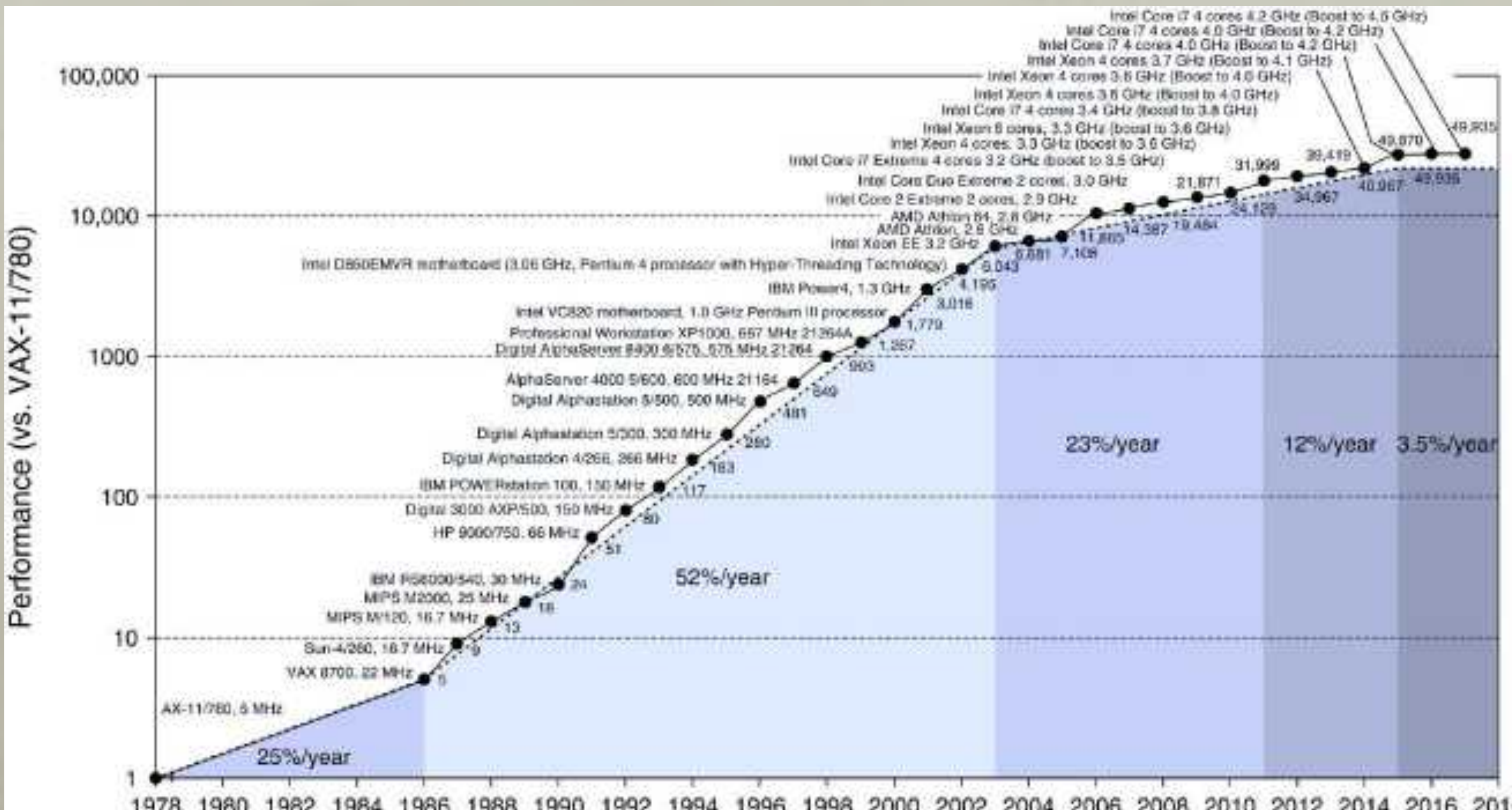
The End of an Era

- 40 years of stunning progress in microprocessor design
  - 1.4x annual performance improvement for 40+ years ≈ 10^6 x faster (throughput)!
- Three architectural innovations:
  - Width: 8->16->64 bit (~4x)
  - Instruction level parallelism:
    - 4-10 *cycles per instruction* to 4+ *instructions per cycle* (~10-20x)
  - Multicore: one processor to 32 cores (~32x)
- Clock rate: 3 MHz to 4 GHz (through technology & architecture)
- Made possible by IC technology:
  - Moore's Law: growth in transistor count
  - Dennard Scaling: power/transistor shrinks as speed & density increase
    - Power = frequency x CV^2
    - Energy expended per computation was reducing

Three Changes Converge

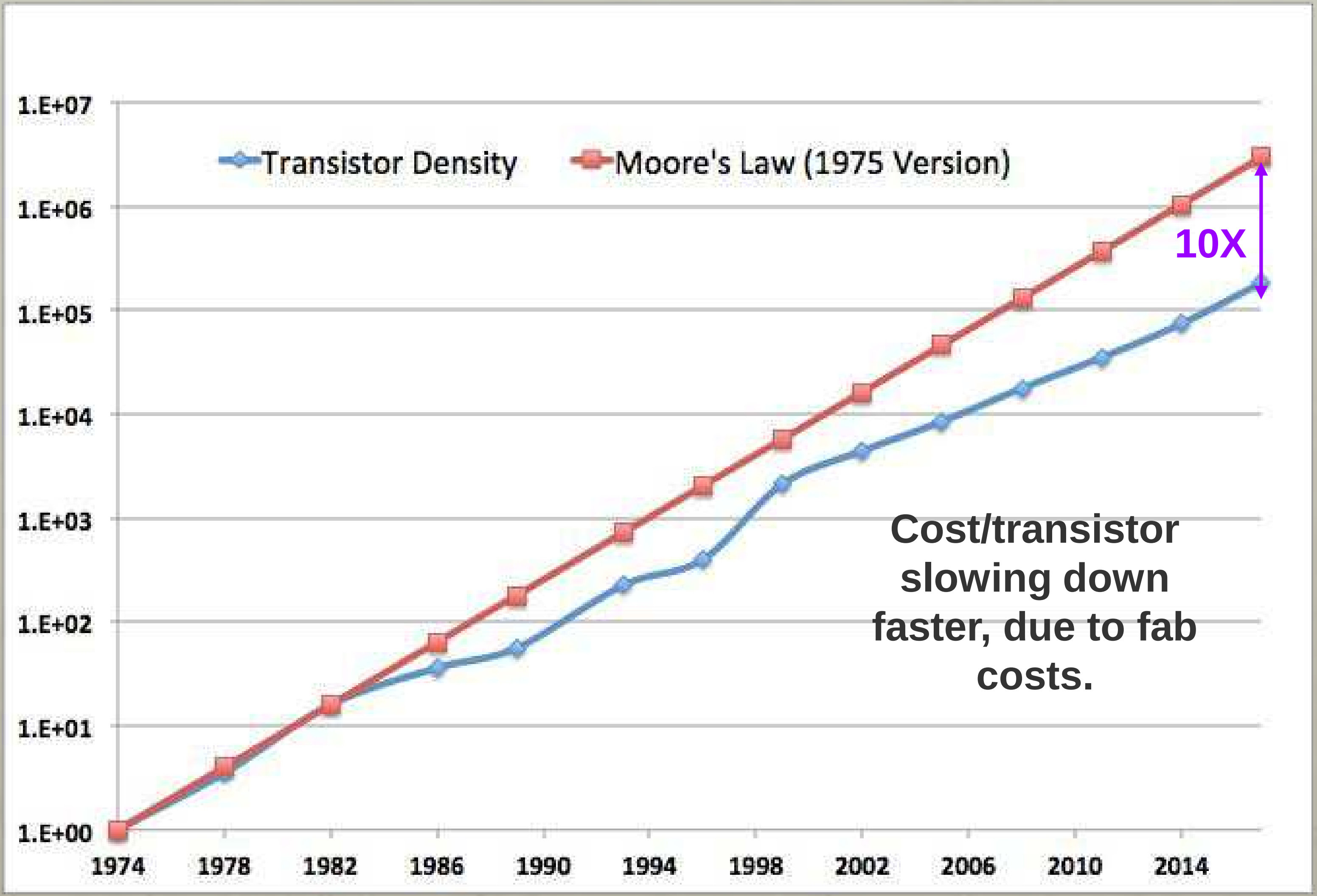
- Technology
  - End of Dennard scaling: power becomes the key constraint
  - Slowdown in Moore's Law: transistors cost (even unused)
- Architectural
  - Limitation and inefficiencies in exploiting instruction level parallelism end the uniprocessor era.
  - Amdahl's Law and its implications end the "easy" multicore era
- Application focus shifts
  - From desktop to individual, mobile devices and ultrascale cloud computing, IoT: new constraints.
  - Machine Learning changes everything!

UNIPROCESSOR PERFORMANCE (SINGLE CORE)

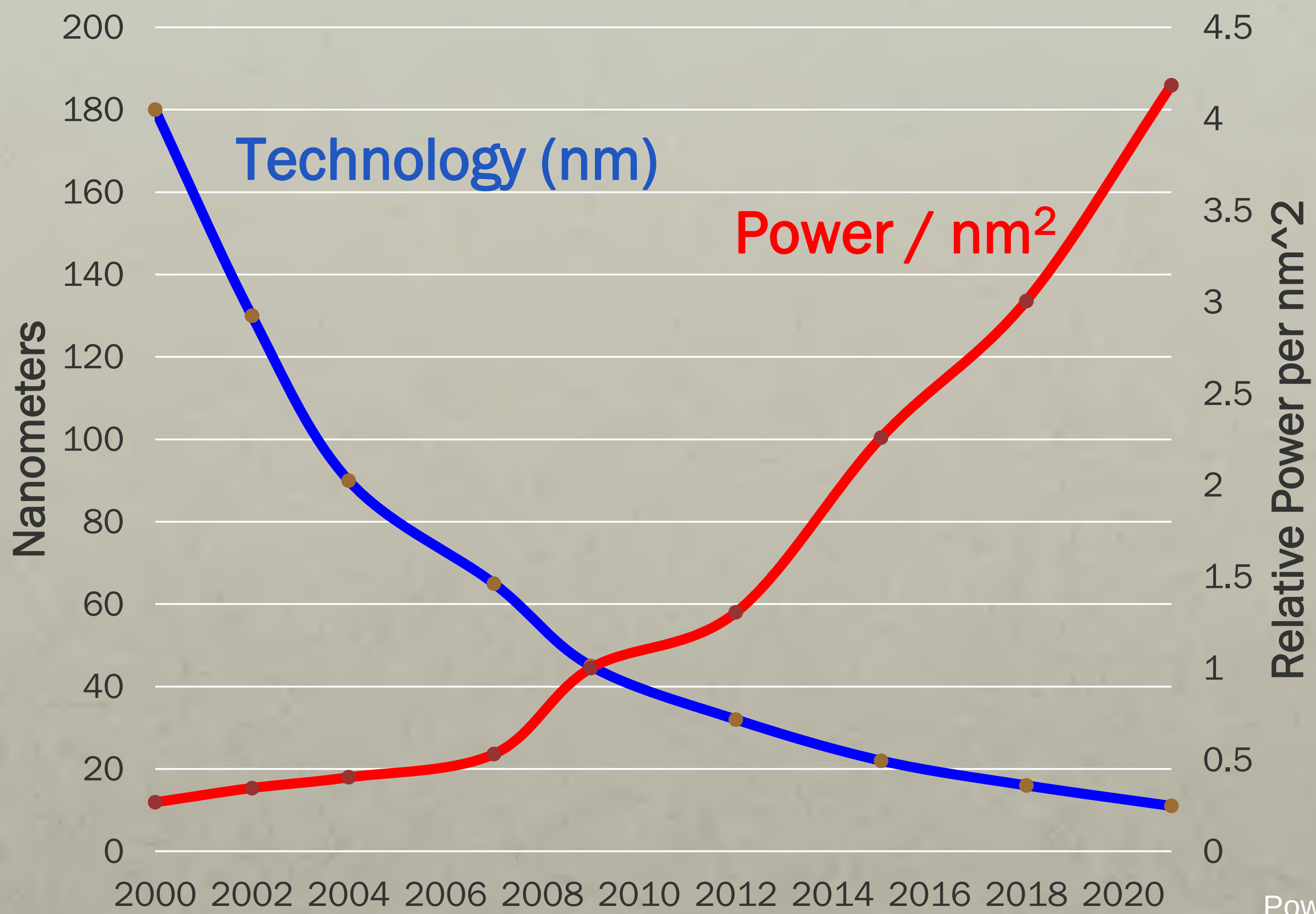


Performance = highest SPECInt by year; from Hennessy & Patterson [2018].

The Technology Shifts
Moore's Law Slowdown in Intel Processors



Technology, Energy, and Dennard Scaling



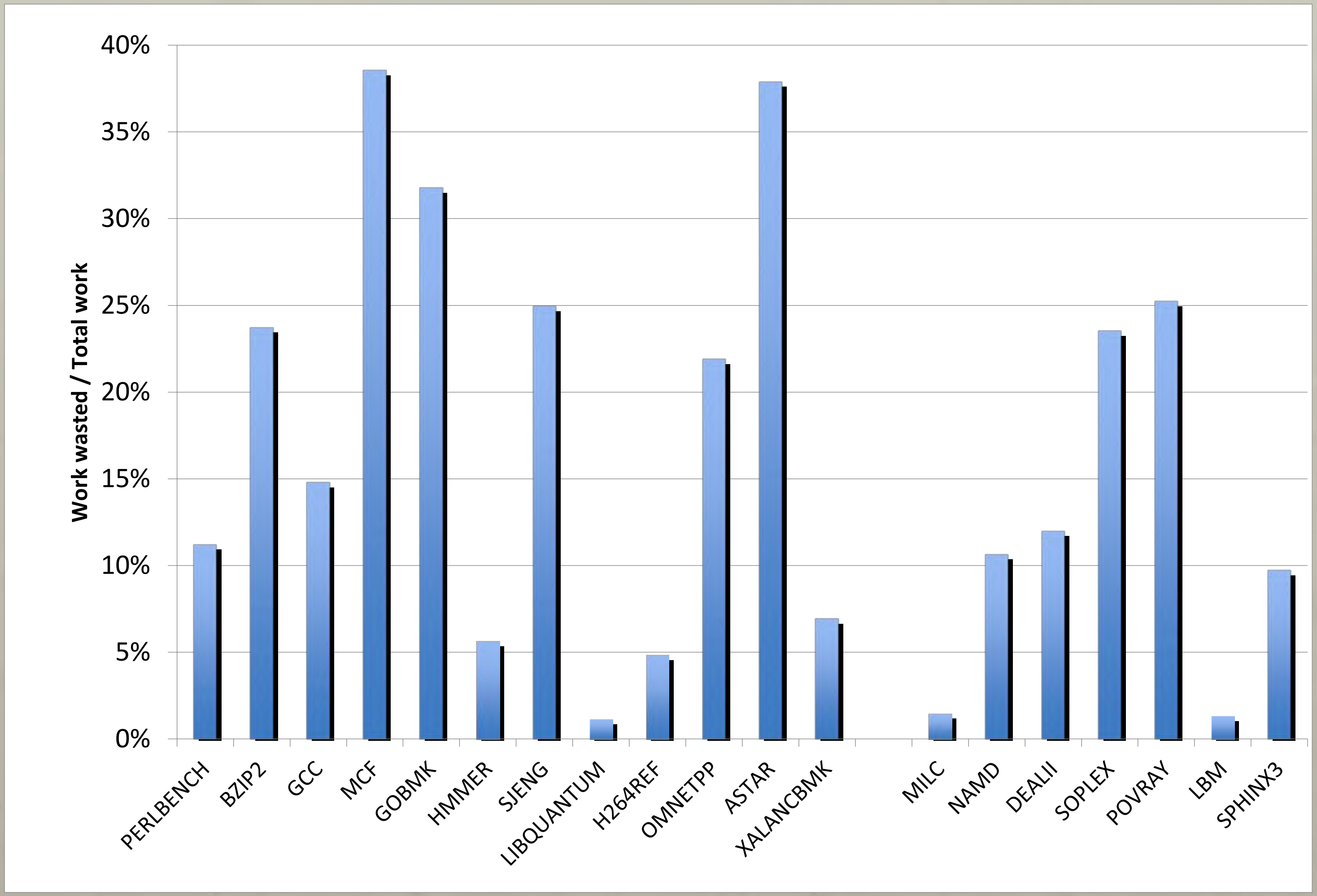
Power consumption based on models in Esmaeilzadeh [2011].

Energy scaling for fixed task is better, since more & faster xistors.

End of Dennard Scaling is a Crisis

- Energy consumption has become more important to users
  - For mobile, IoT, and for large clouds (second largest cost factor!)
- Processors have reached their power limit
  - Thermal dissipation is maxed out (chips turn off to avoid overheating!)
  - Even with better packaging: heat and battery are limits.
- Architectural advances must *increase* energy efficiency
  - Reduce power or improve performance for same power
- *But*, most architectural techniques have reached limits in energy efficiency!
  - 1982-2005: Instruction level parallelism
    - Compiler and processor find parallelism
  - 2005-2017: Multicore
    - Programmer identifies parallelism
  - Caches: diminishing returns (small incremental improvements).

WASTED WORK ON THE INTEL CORE I7



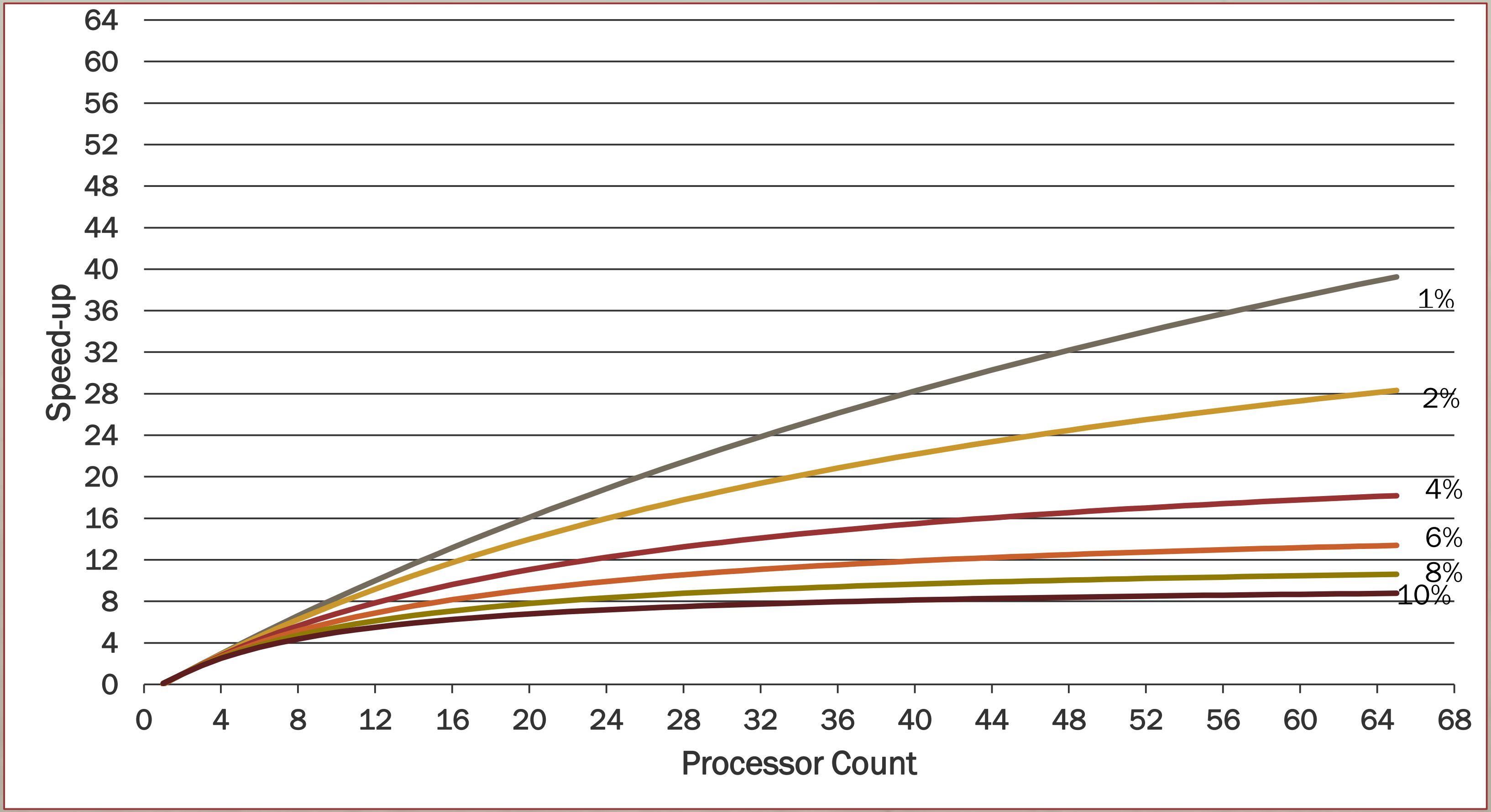
Data collected by Professor Lu Peng and student Ying Zhang at LSU.

The Multicore Era
2005-2017

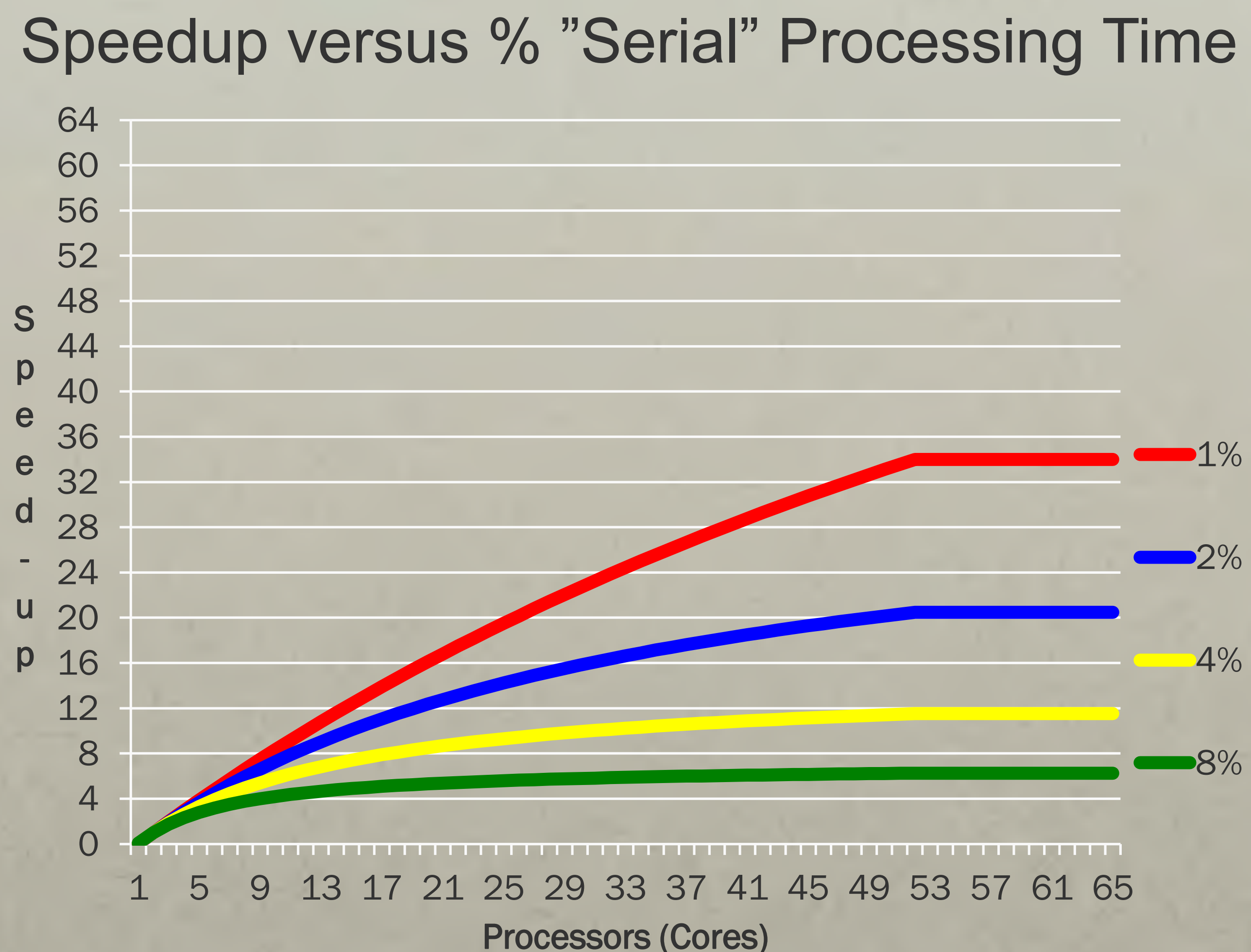
- Make the programmer responsible for identifying parallelism via threads
- Exploit the threads on multiple cores
- Increase cores if more transistors: easy scaling!
- Energy ≈ Transistor count ≈ Active cores
- So, we need Performance ≈ Active cores
- But, Amdahl's Law says that this is highly unlikely

Amdahl's Law Limits Performance Gains from Parallel Processing

Speedup versus % "Serial" Processing Time



Putting the Challenges Together
Dennard Scaling + Amdahl's Law

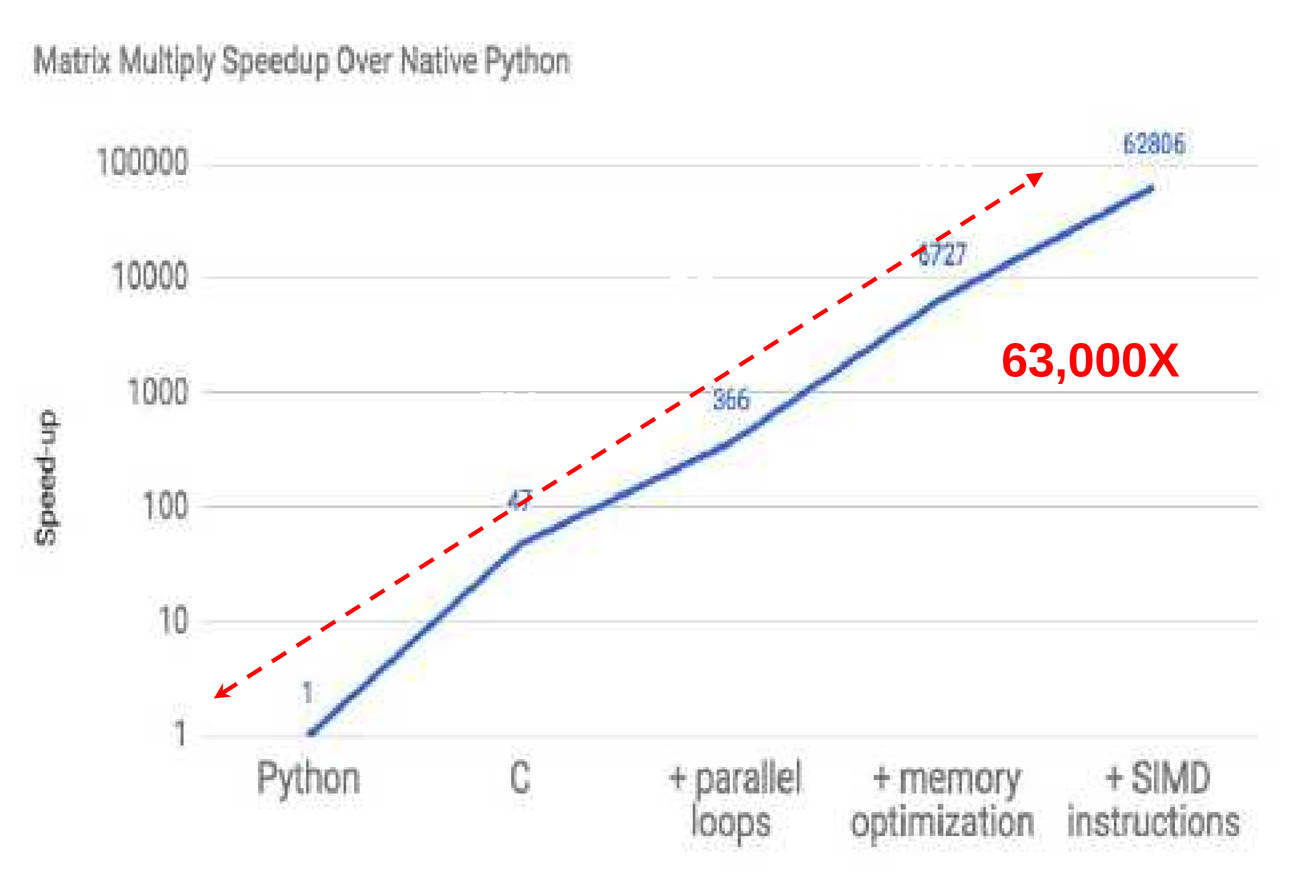


What Opportunities Left?

- 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)



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

Domain Specific Architectures (DSAs)

- Achieve higher efficiency by tailoring architecture to characteristics of domain
  - Not one application, but domain (different from strict ASIC)
  - Requires more domain-specific knowledge then general purpose processors need
  - Design DSAs and processors for targeted environments
    - More variability than in GP processors
- Examples:
  - Neural network processors for machine learning
  - GPUs for graphics, virtual reality
- Some good news: demand for higher performance focused on such domains
- Caveat: most attempts to “beat” general purpose CPUs in past have failed
  - This time is different: but do your HW!

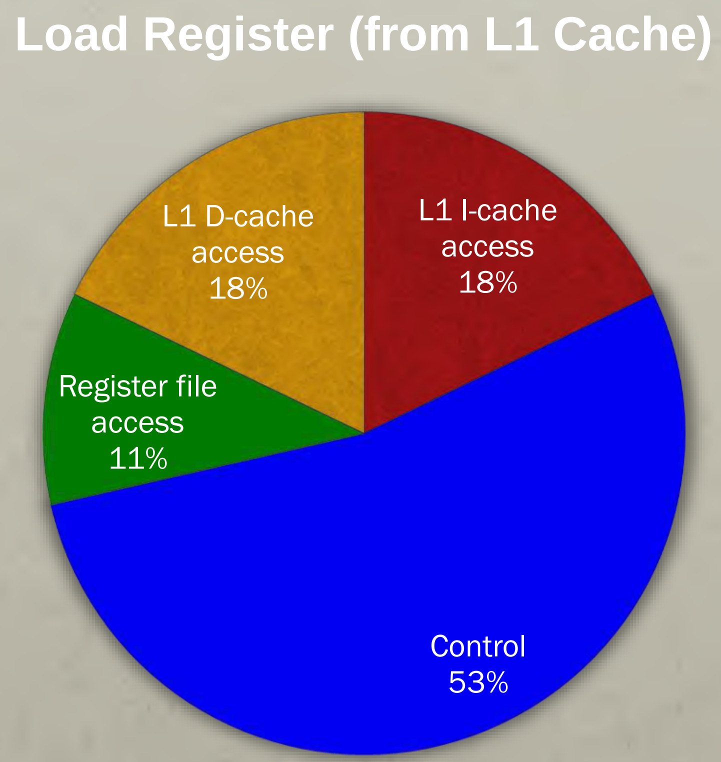
Where does the energy go? Can DSAs do Better?

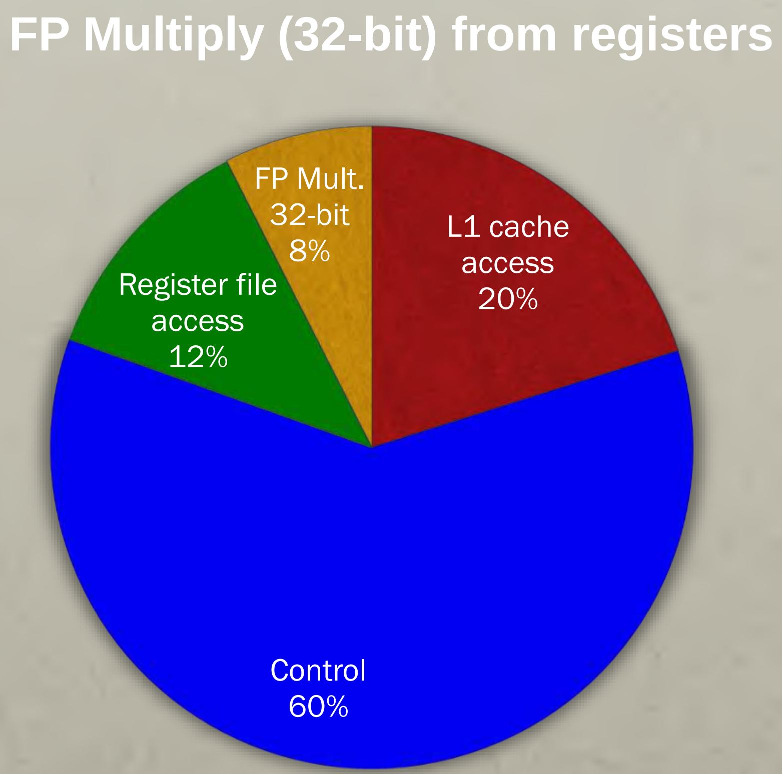
Function	Energy in Picojoules
8-bit add	0.03
32-bit add	0.1
FP Multiply 16-bit	1.1
FP Multiply 32-bit	3.7
Register file access*	6
Control (per instruction, superscalar)	20-40
L1 cache access	10
L2 cache access	20
L3 cache access	100
Off-chip DRAM access	1,300-2,600

* Increasing the size or number of ports, increases energy roughly proportionally.

From Horowitz [2016].

INSTRUCTION ENERGY BREAKDOWN

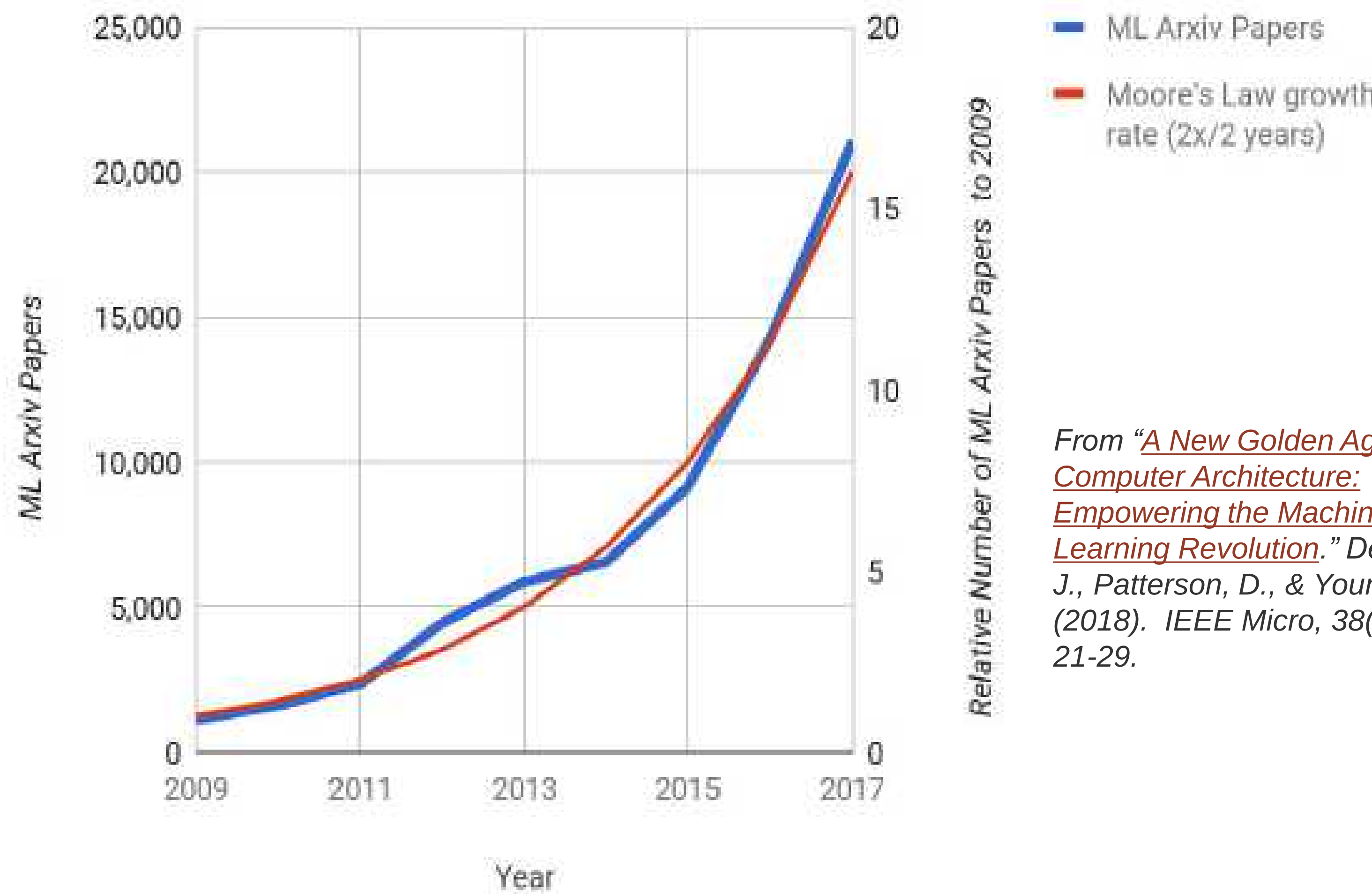




Why DSAs Can Win (no magic)
Tailor the Architecture to the Domain

- Simpler parallelism for a specific domain (less control HW):
  - SIMD vs. MIMD
  - VLIW vs. Speculative, out-of-order
- More effective use of memory bandwidth (on/off chip)
  - User controlled versus caches
  - Processor + memory structures versus traditional
  - Program prefetching to off-chip memory when needed
- Eliminate unneeded accuracy
  - IEEE replaced by lower precision FP
  - 32-bit,64-bit integers to 8-16 bits
- Domain specific programming model matches application to the processor architecture

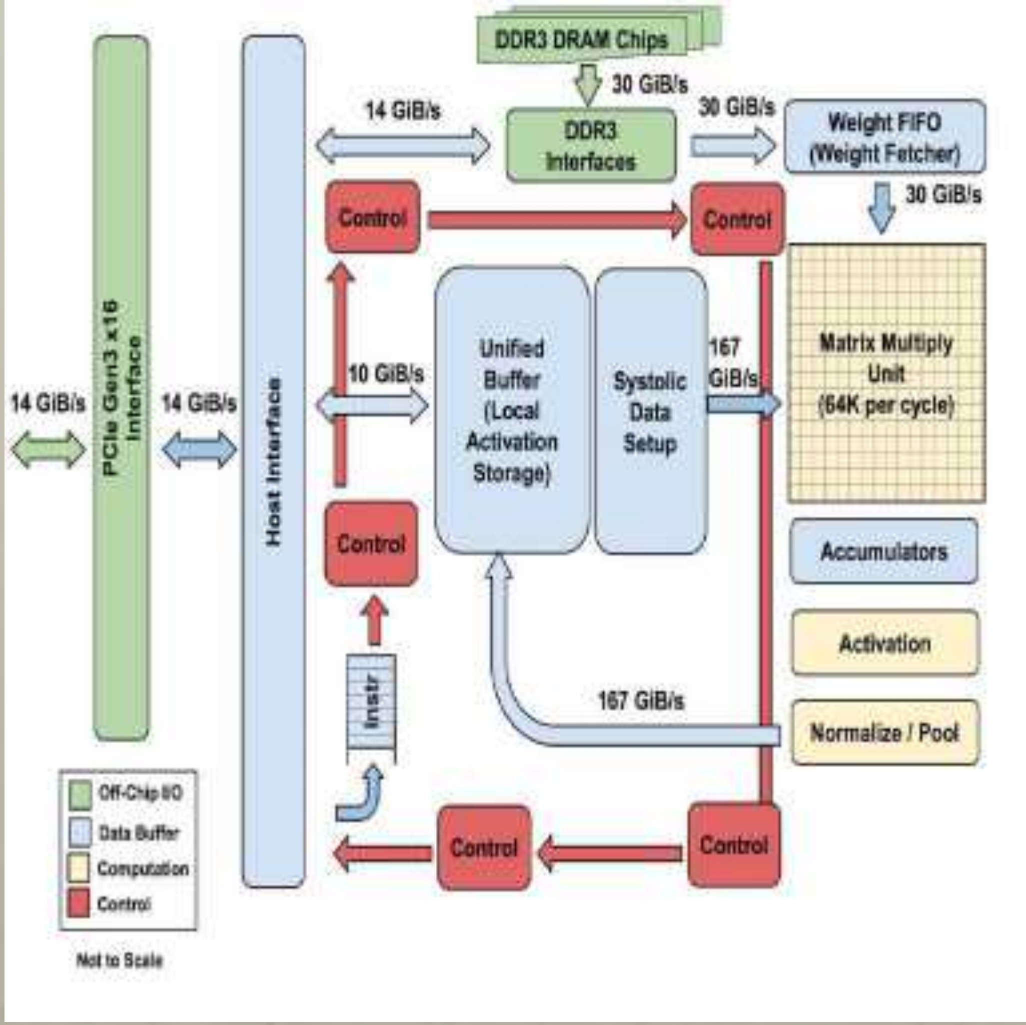
Deep learning is causing a machine learning revolution



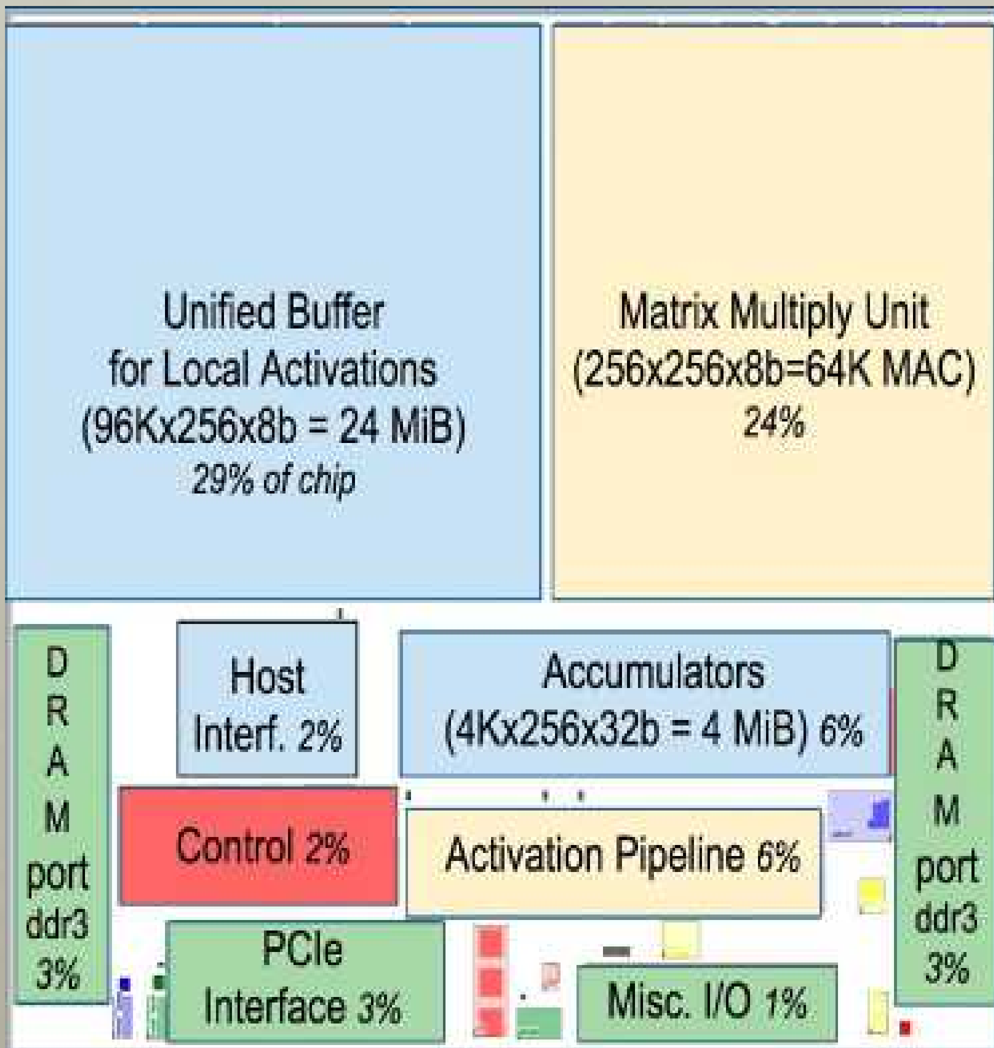
From "A New Golden Age in Computer Architecture: Empowering the Machine-Learning Revolution." Dean, J., Patterson, D., & Young, C. (2018). IEEE Micro, 38(2), 21-29.

TPU 1: High-level Chip Architecture for DNN Inference

- 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
- Accelerator (not a CPU)
- Inference only



How is Silicon Used: DSA vs CPU



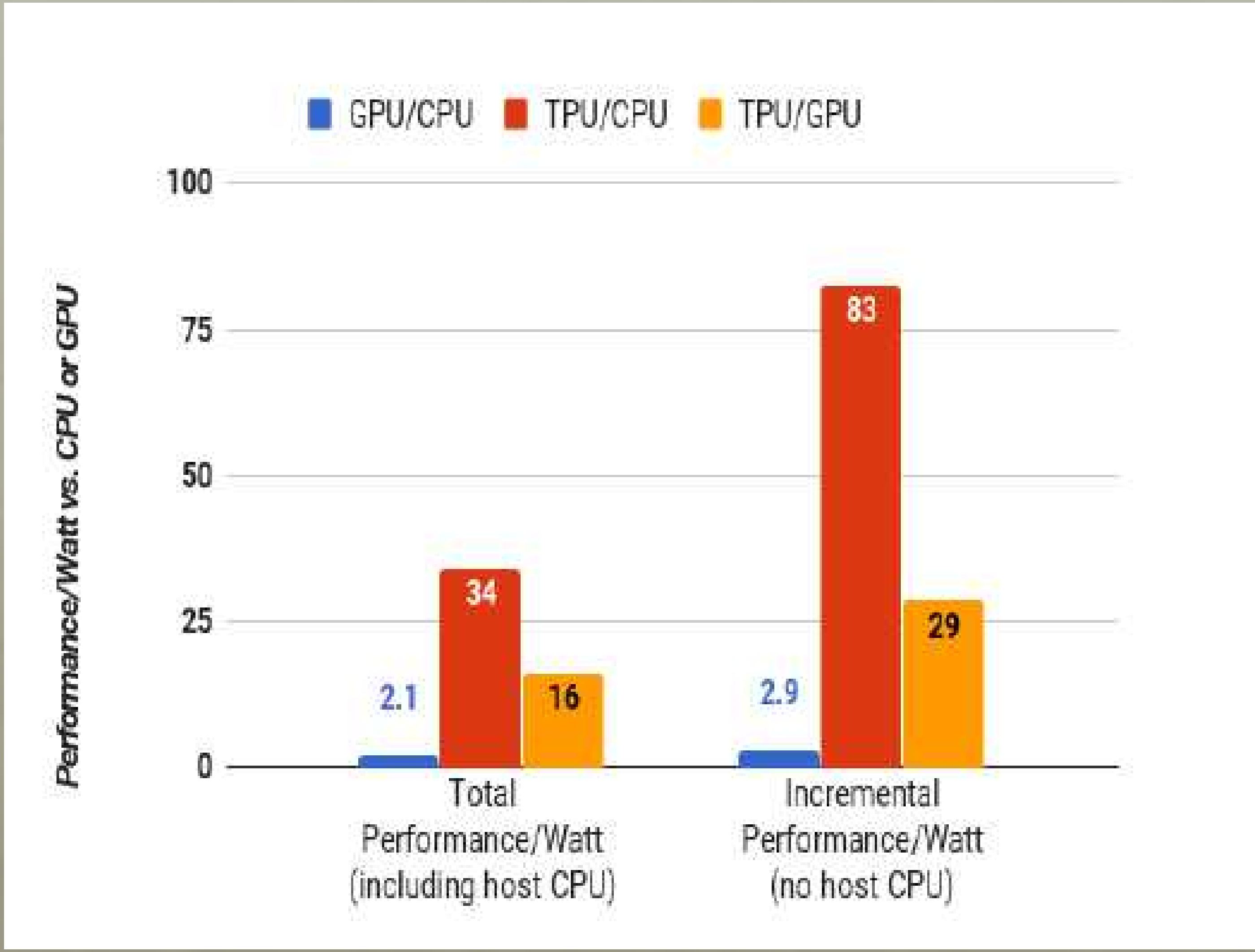
TPU-1 (–pads)

- Memory: 44%
- Compute: 39%
- Interface: 15%
- Control: 2%

CPU (Skylake core)

- Cache: 33%
- Control: 30%
- Compute: 21%
- Mem Man:12%
- Misc: 4%

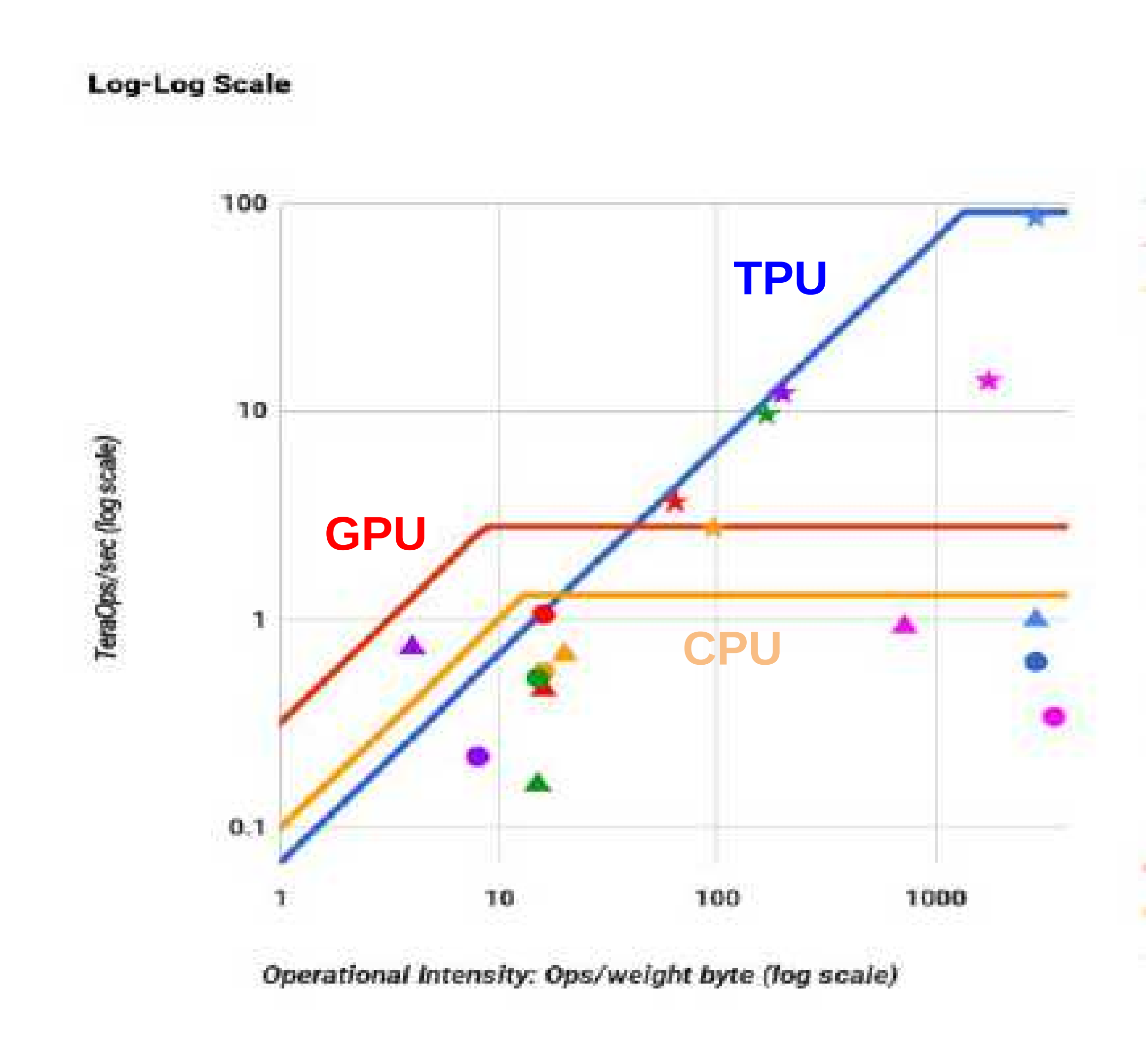
Performance/Watt on Inference TPU-1 vs CPU & GPU



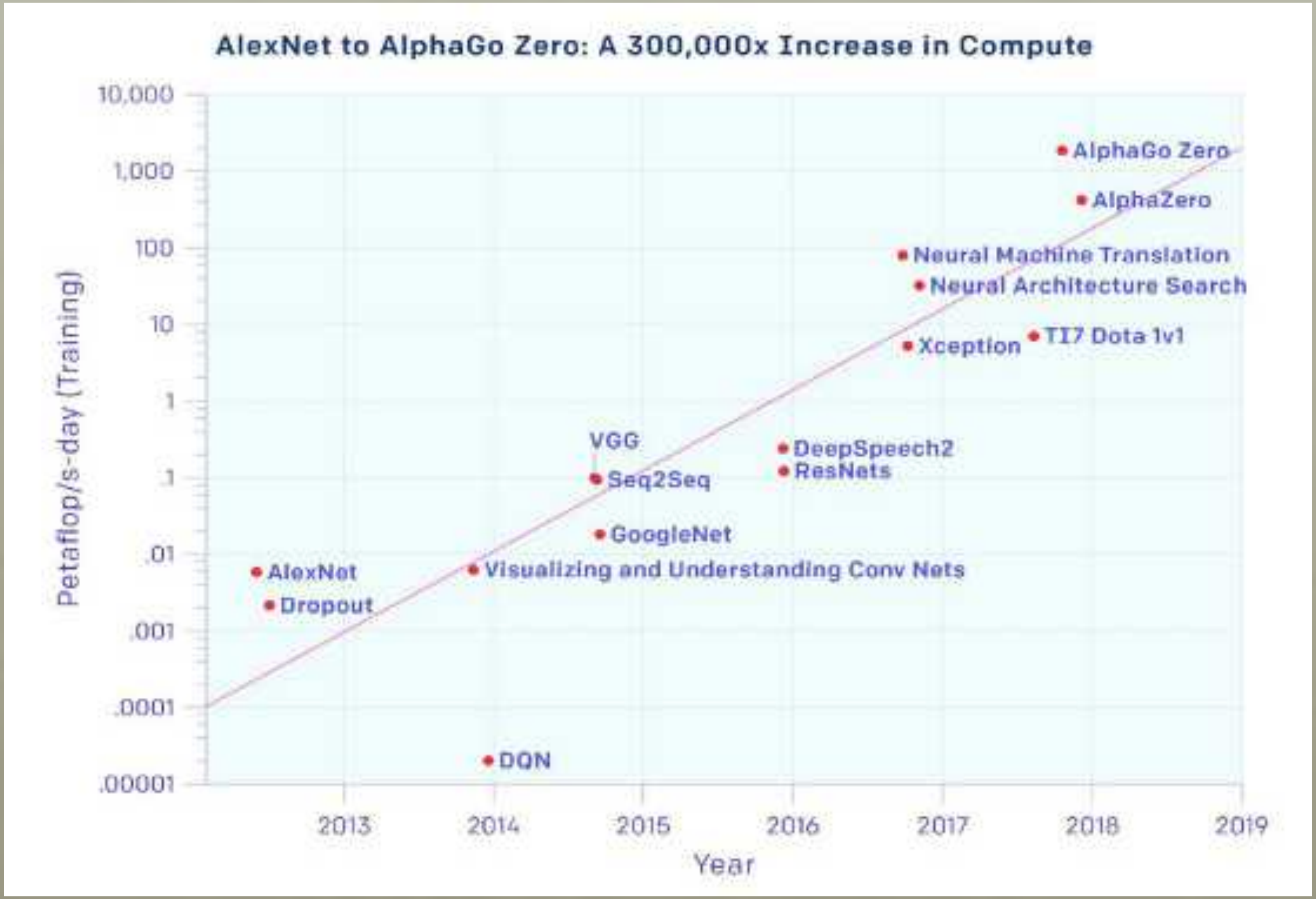
Important caveat:

- **TPU-1 uses 8-bit integer**
- **GPU uses FP**

Log Rooflines for CPU, GPU, TPU-1



Training: A Much More Intensive Problem



Rapid Innovation

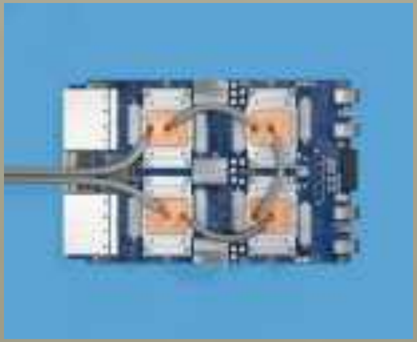
TPU v1
(deployed 2015)

92 teraops
Inference only

Cloud TPU v2
2 Tensor Cores each
with 128x128 MXU
Training and inference

180 teraflops: vector unit
64 GB HBM
VLIW instructions (8-wide)
Transpose, reduce, permute unit

Cloud TPU v3
2 Tensor Cores each with
2x128x128 MXU
Training and inference



420 teraflops: vector unit
128 GB HBM
VLIW instructions (8-wide)
Transpose, reduce, permute unit

Enabled by simpler design, compatibility at DSL level, ease of verification.

Enabling Massive Computing Cycles for Training

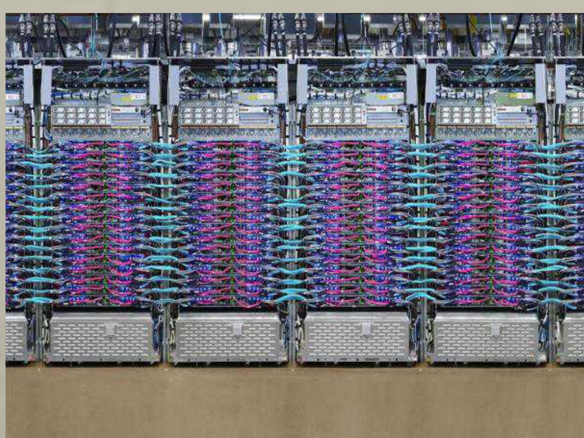
MPU v1 Pod (TPU v2, 2017)

11.5 petaflops

4 TB HBM

Glueless MP:

- 4 chips in a ring



MPU v2 Pod (TPU v3 2018)

> 100 petaflops!

32 TB HBM

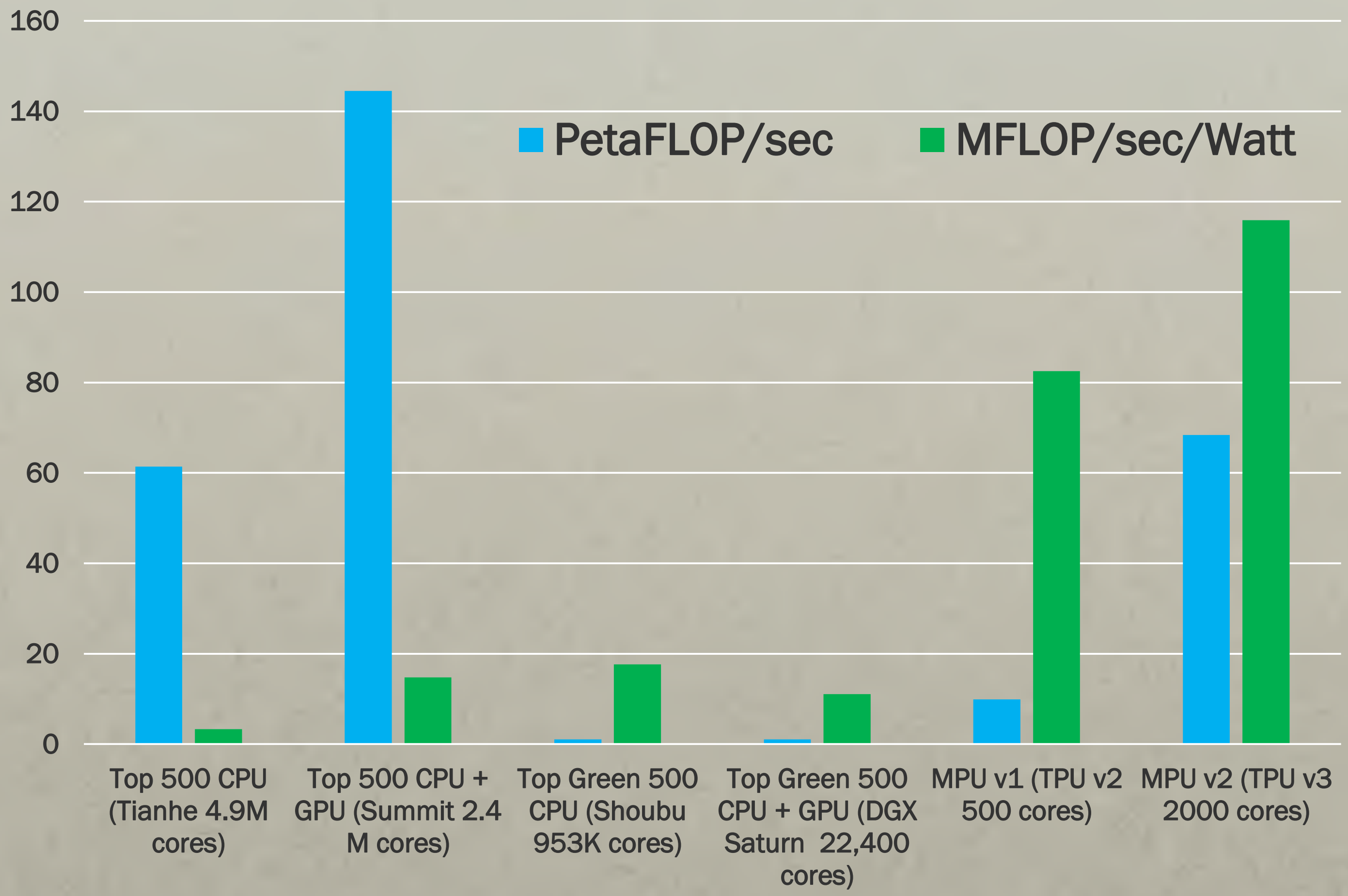
Liquid cooled

New chip architecture + larger-scale system

Glueless MP:

- 4 links @ 650Gbits/s per link
- 2-D toroidal mesh network: 1,024 TPUs!

TOP 500 AND TOP GREEN 500 ON SCALED LINPACK VERSUS MPU V1 AND V2 ON CNN0



Note: Supercomputers use 64/32 FP; MPU uses 32/64.

CHALLENGES AND OPPORTUNITIES

- Design of DSAs and DSLs
  - Optimizing the mapping to a DSA for *both* portability & performance.
  - DSAs & DSLs for new fields (how general is the architecture?)
  - Big open problem: dealing with sparse data
- Make HW development more like software:
  - Prototyping, reuse, abstraction
  - Open HW stacks (ISA to IP libraries)
  - Role of ML in CAD?
- Technology:
  - Silicon: Extend Dennard scaling and Moore's Law
  - Packaging: use optics, enhance cooling
  - Beyond Si: Carbon nanotubes, Quantum?

CONCLUDING THOUGHTS: EVERYTHING OLD IS NEW AGAIN

- **Dave Kuck, software architect for Illiac IV (circa 1975)**

  "What I was really frustrated about was the fact, with Iliac IV, programming the machine was very difficult and the architecture probably was not very well suited to some of the applications we were trying to run. The key idea was that I did not think we had a very good match in Iliac IV between applications and architecture."

- **Achieving cost-performance in this era of DSAs will require matching the applications, languages, architecture, and reducing design cost.**

Dave Kuck, ACM Oral History