The End of Moore's Law & Faster General Purpose Computing, and a Road Forward

John Hennessy
Stanford University
March 2019

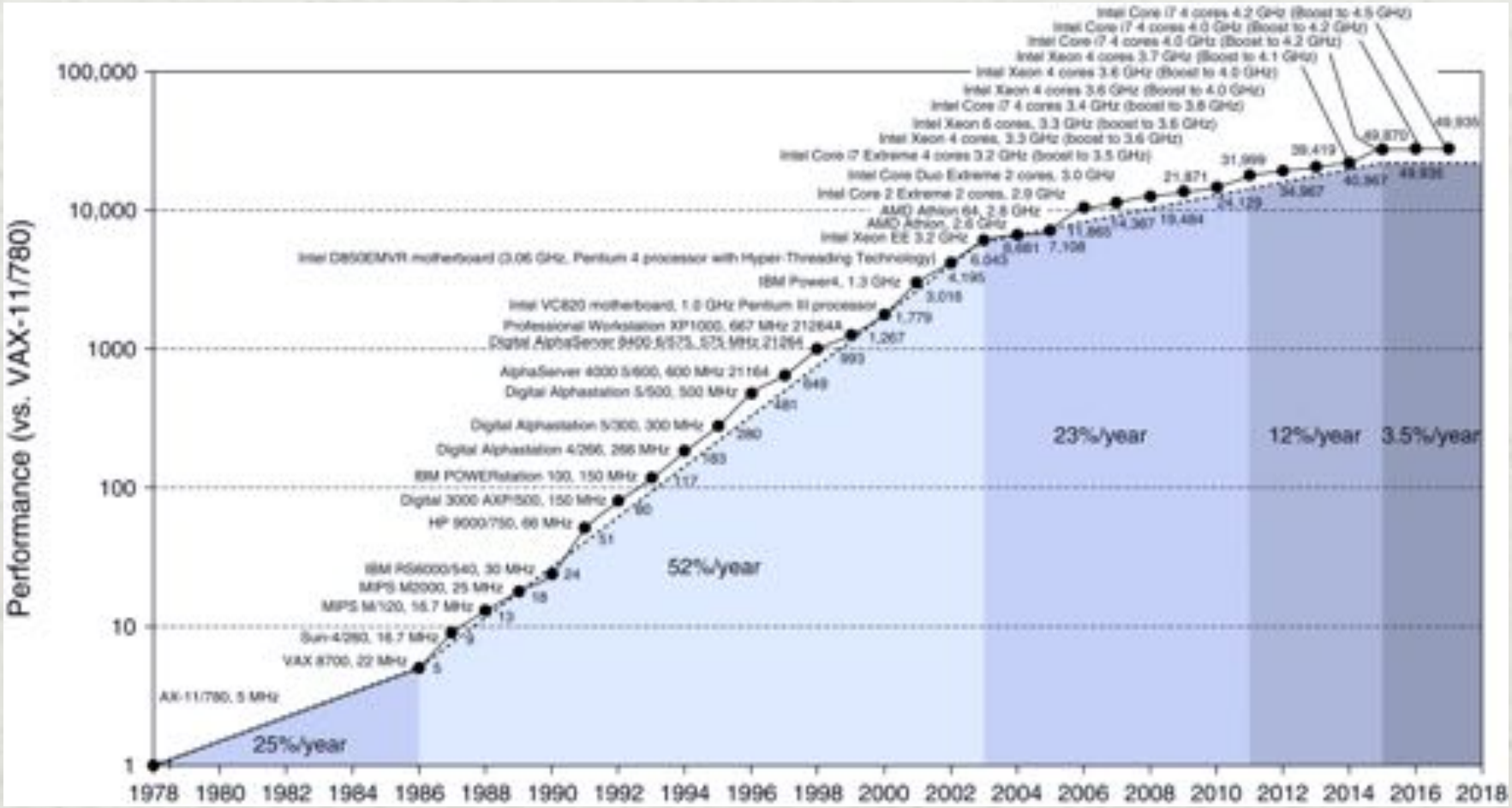
The End of an Era

- 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

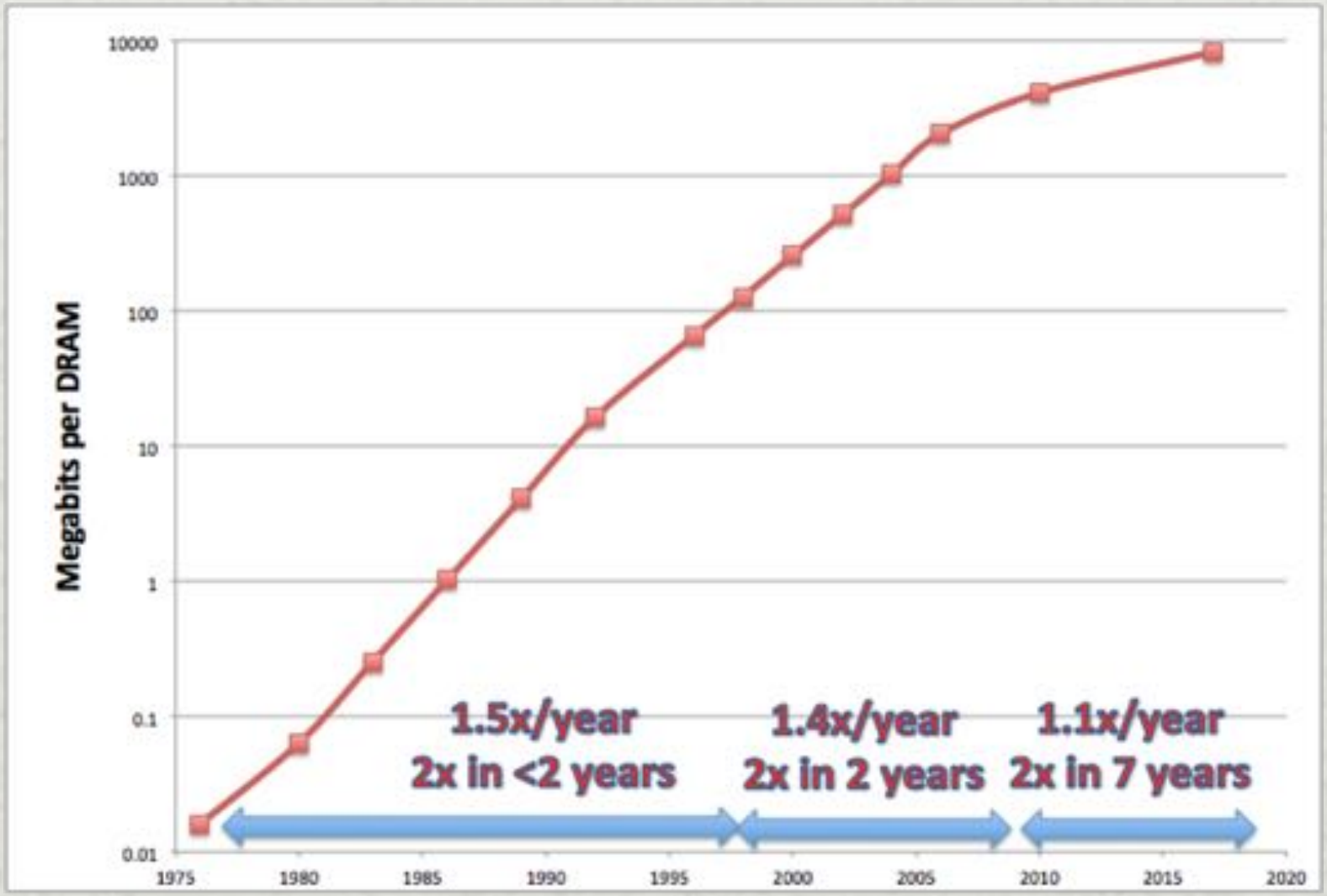
- Technology
  - End of Dennard scaling: power becomes the key constraint
  - Slowdown in Moore's Law: transistors cost (even unused)
- Architectural
  - Limitation and inefficiencies in exploiting instruction level parallelism end the uniprocessor era.
  - Amdahl's Law and its implications end the "easy"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

Uniprocessor Performance (single core)

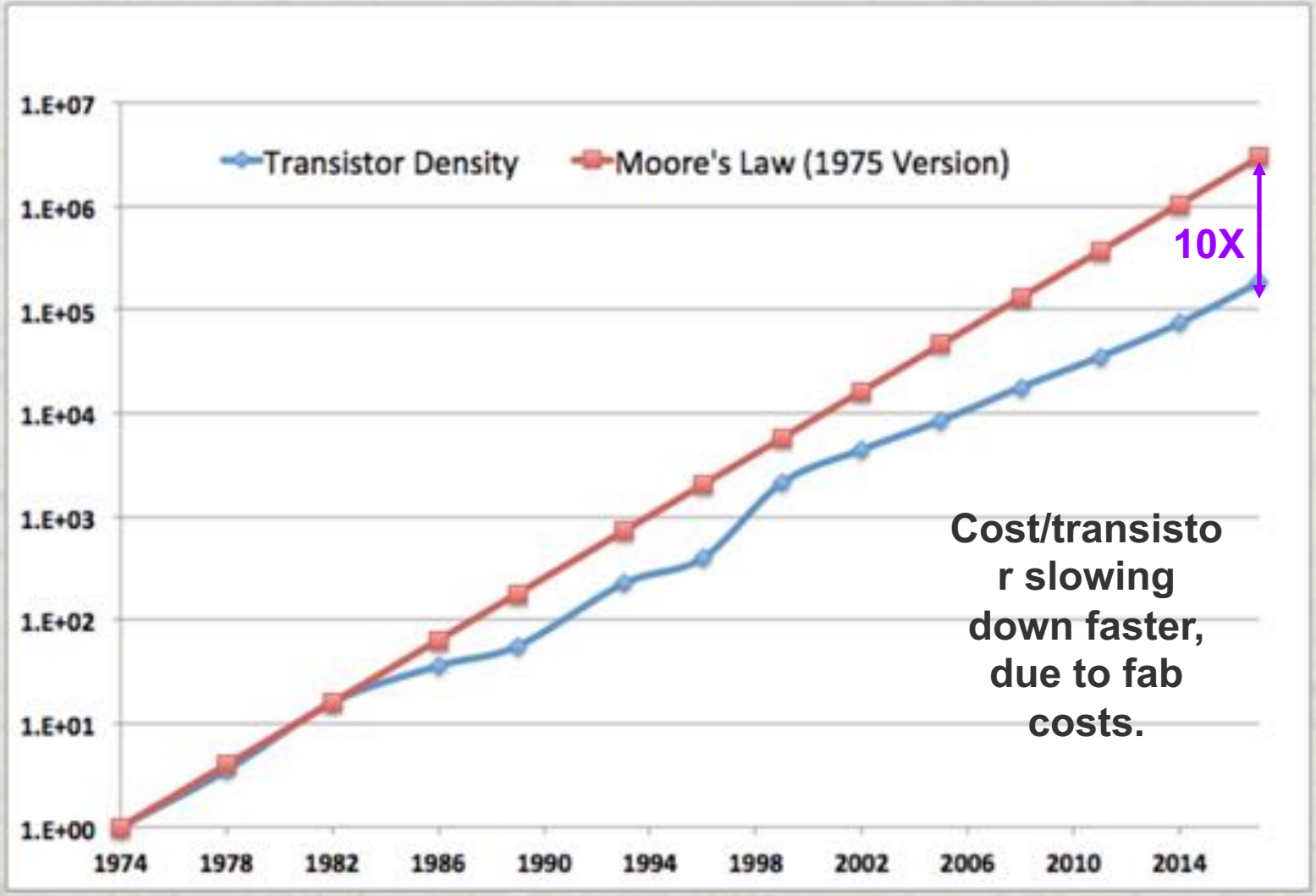


Performance = highest SPECInt by year; from Hennessy & Patterson [2018].

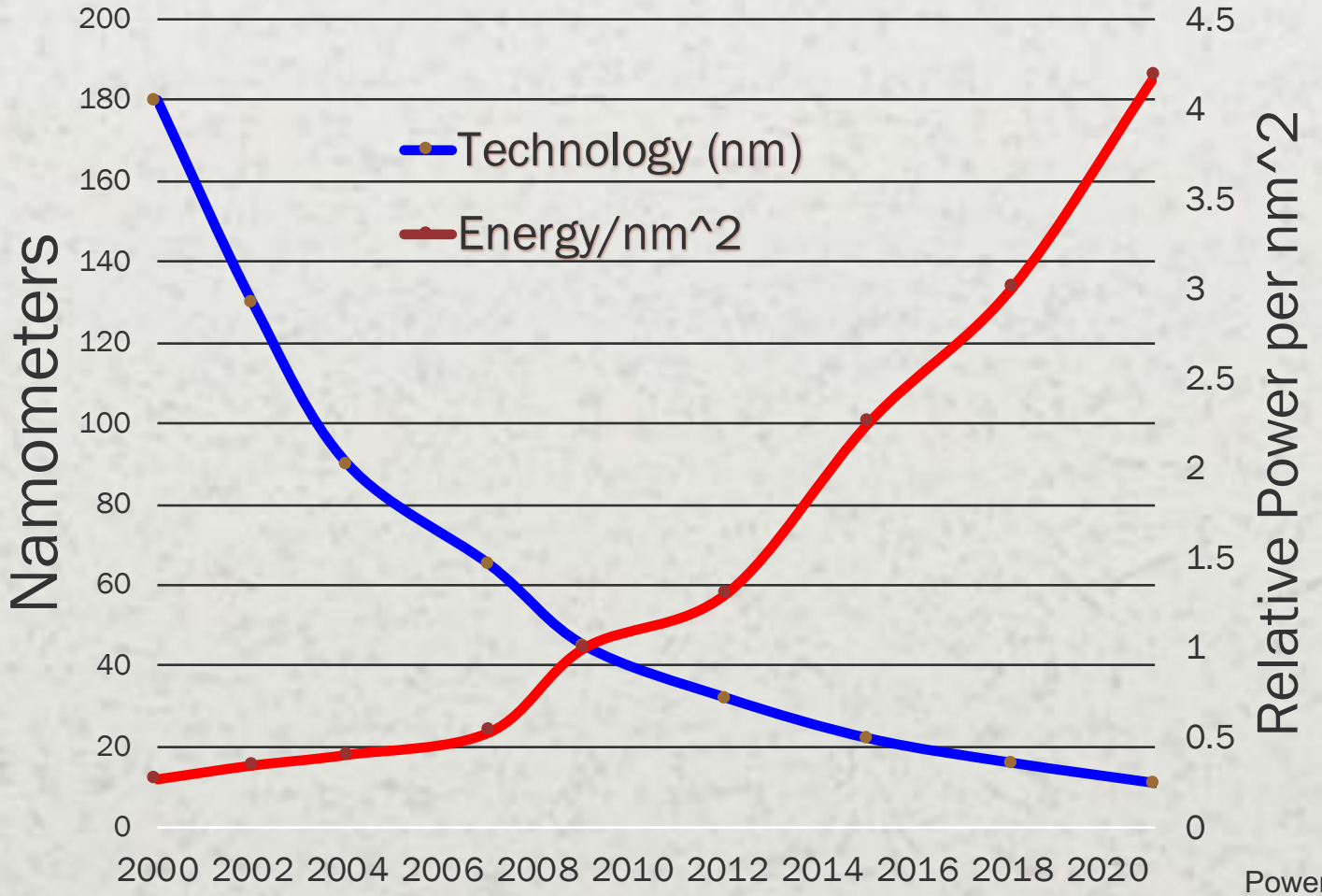
MOORE'S LAW IN DRAMS



The Technology Shifts
Moore's Law Slowdown in Intel Processors



Technology, Power, and Dennard Scaling

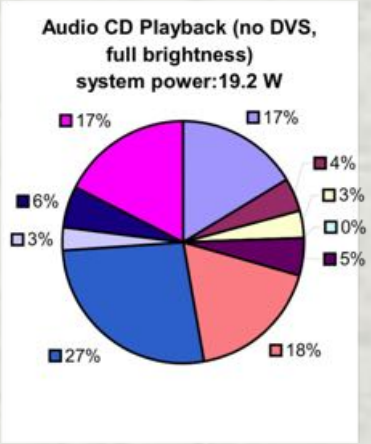


Energy scaling for fixed task is better, since more & faster xistors.

Power consumption based on models in Esmaeilzadeh [2011].

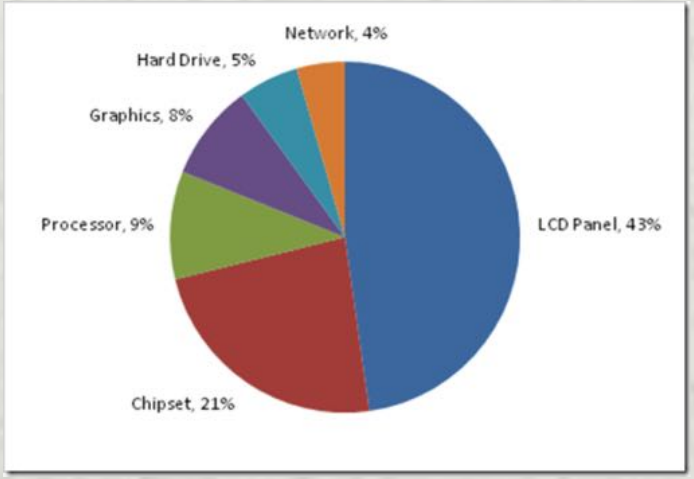
ENERGY EFFICIENCY IS THE NEW METRIC

Battery lifetime determines effectiveness!

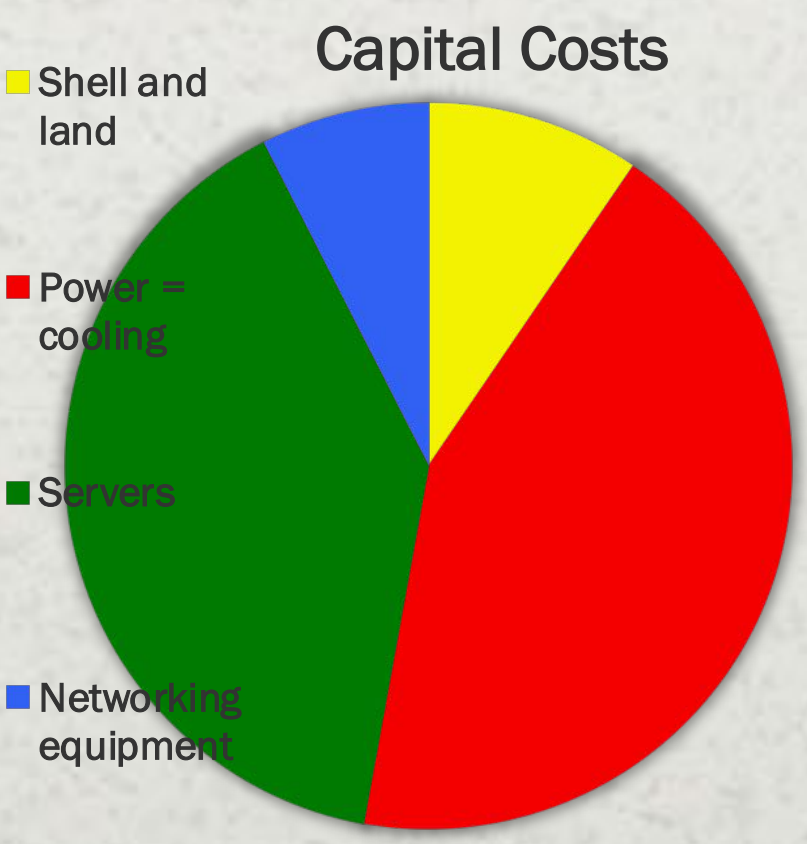


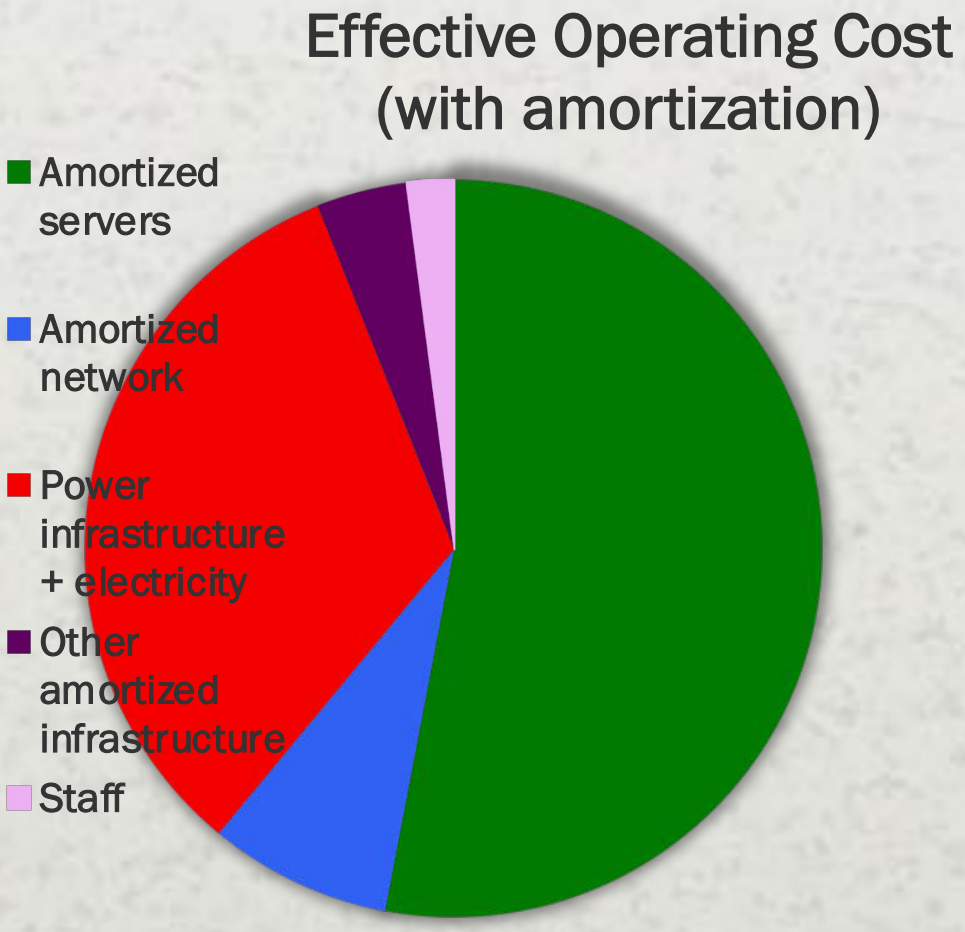
LCD is biggest; CPU close behind.

"Always on" assistants likely to increase CPU demand.



AND IN THE CLOUD





End of Dennard Scaling is a Crisis

- Energy consumption has become more important to users
  - For mobile, IoT, and for large clouds
- 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 the 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).

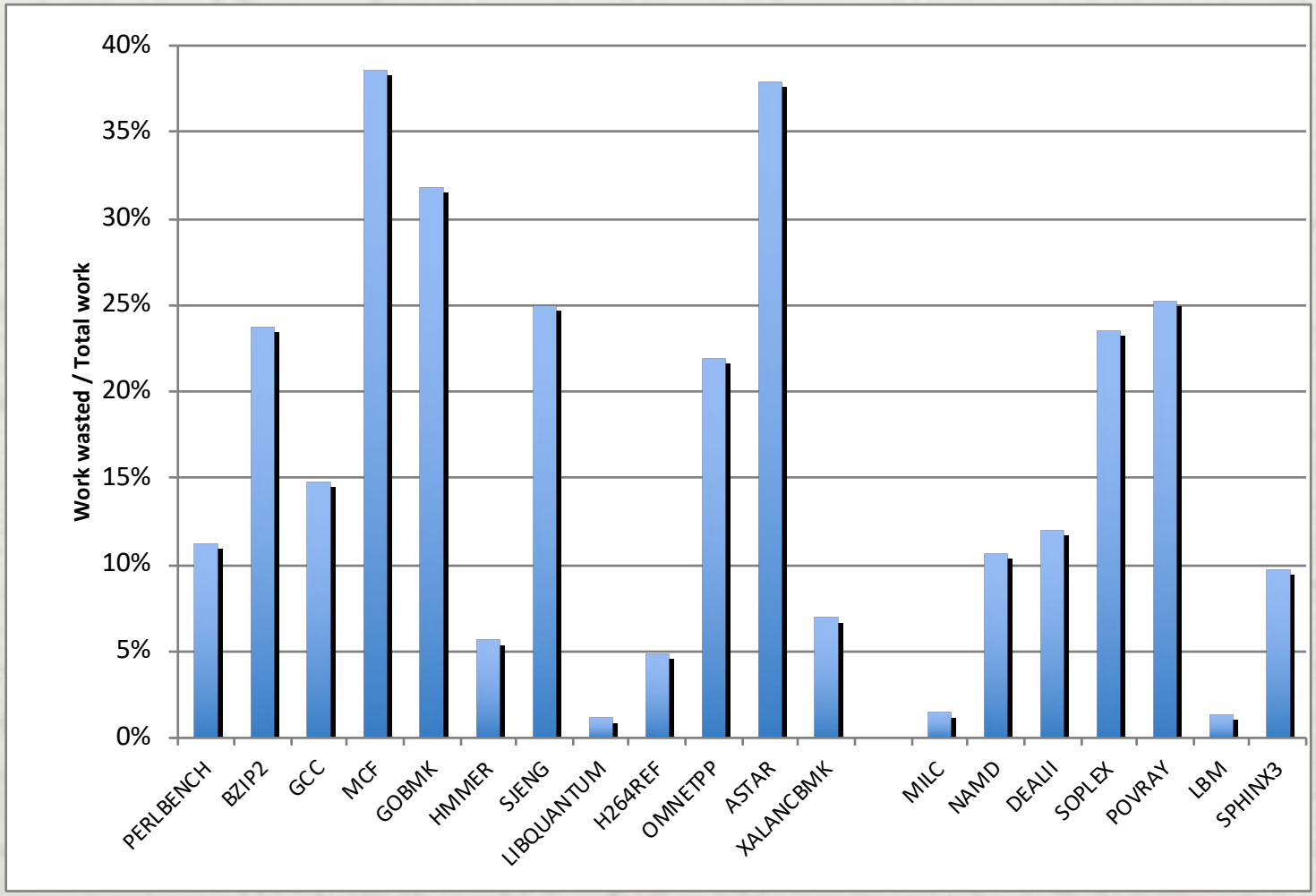
Instruction Level Parallelism Era 1982-2005

- Instruction level parallelism achieves significant performance advantages
- Pipelining: 5 stages to 15+ stages to allow faster clock rates (energy neutralized by Dennard scaling)
- Multiple issue: <1 instruction/clock to 4+ instructions/clock
  - Significant increase in transistors to increase issue rate
- Why did it end?
  - Diminishing returns in efficiency

Getting More ILP

- Branches and memory aliasing are a major limit:
  - 4 instructions/clock x 15 deep pipeline➔ need more than 60 instructions “in flight”
- Speculation was introduced to allow this
- Speculation involves predicting program behavior
  - Predict branches & predict matching memory addresses
  - If prediction is accurate can proceed
  - If the prediction is inaccurate, undo the work and restart
- How good must branch prediction be—*very good*!
  - 15-deep pipeline: ~4 branches 94% correct = 98.7%
  - 60-instructions in flight: ~15 branches 90% = 99%

WASTED WORK ON THE INTEL CORE I7



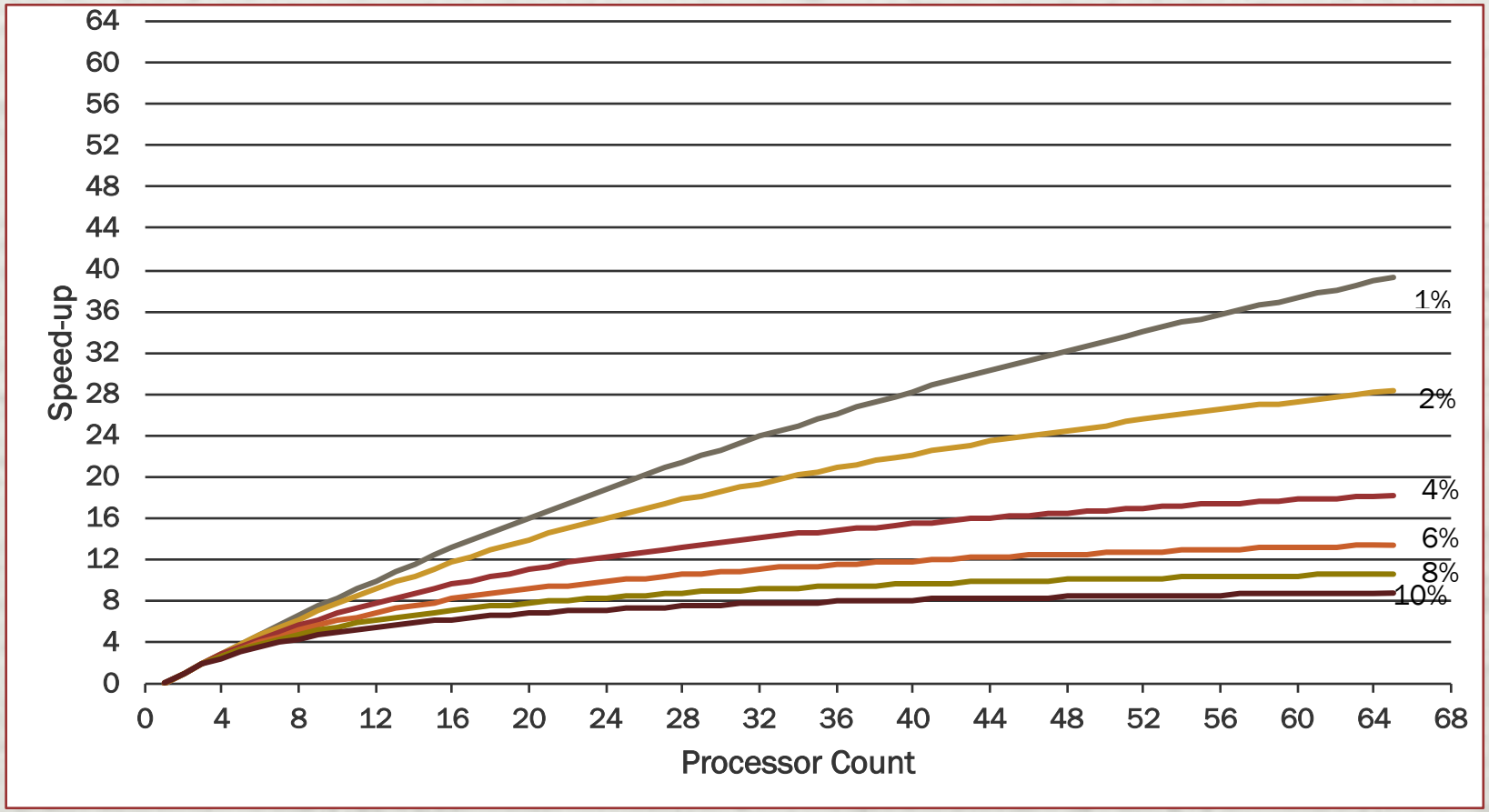
Data collected by Professor Lu Peng and student Ying Zhang at LSU.

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

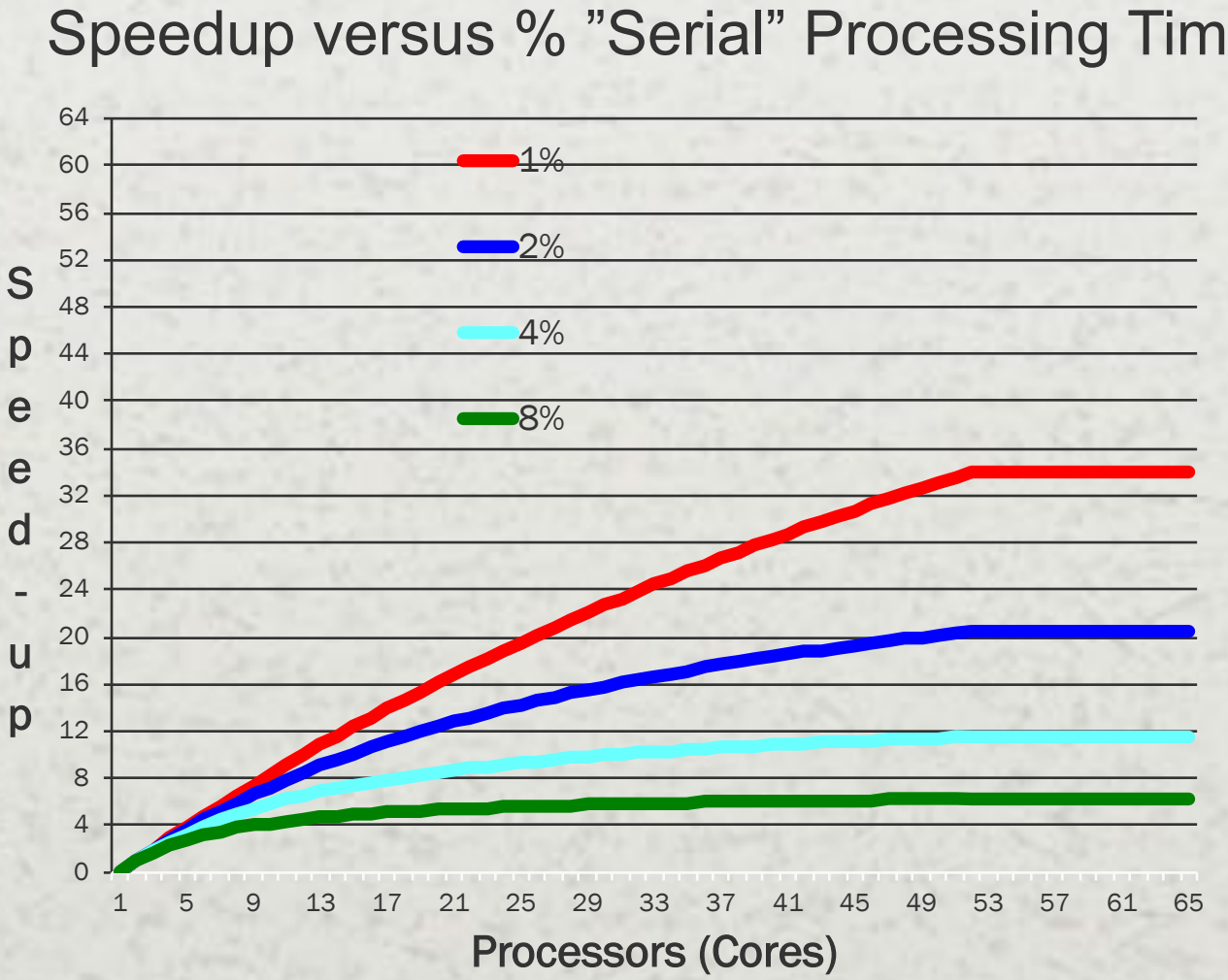


Second Challenge to Higher Performance from Multicore: End of Dennard Scaling

- End of Dennard scaling means multicore scaling ends
  - Full scaling will mean “dark silicon,” with cores OFF.
- Example
  - Today: 14 nm process, largest Intel multicore
    - Intel E7-8890: 24-core, 2.2 GHz, TDP = 165W (power limited)
      - Turbo (one core): 3.4 GHz. All cores @ 3.4 GHz = 255 W.
  - A 7 nm process could yield (estimates)
    - 64 cores; power unconstrained: 6 GHz & 365 W.
    - 64 cores; power constrained: 4 GHz & 250 W.

Power Limit	Active Cores
180 W	46/64
200 W	51/64
220 W	56/64

Putting the Challenges Together
Dennard Scaling + Amdahl's Law



Sidebar: Instruction Set Efficiency

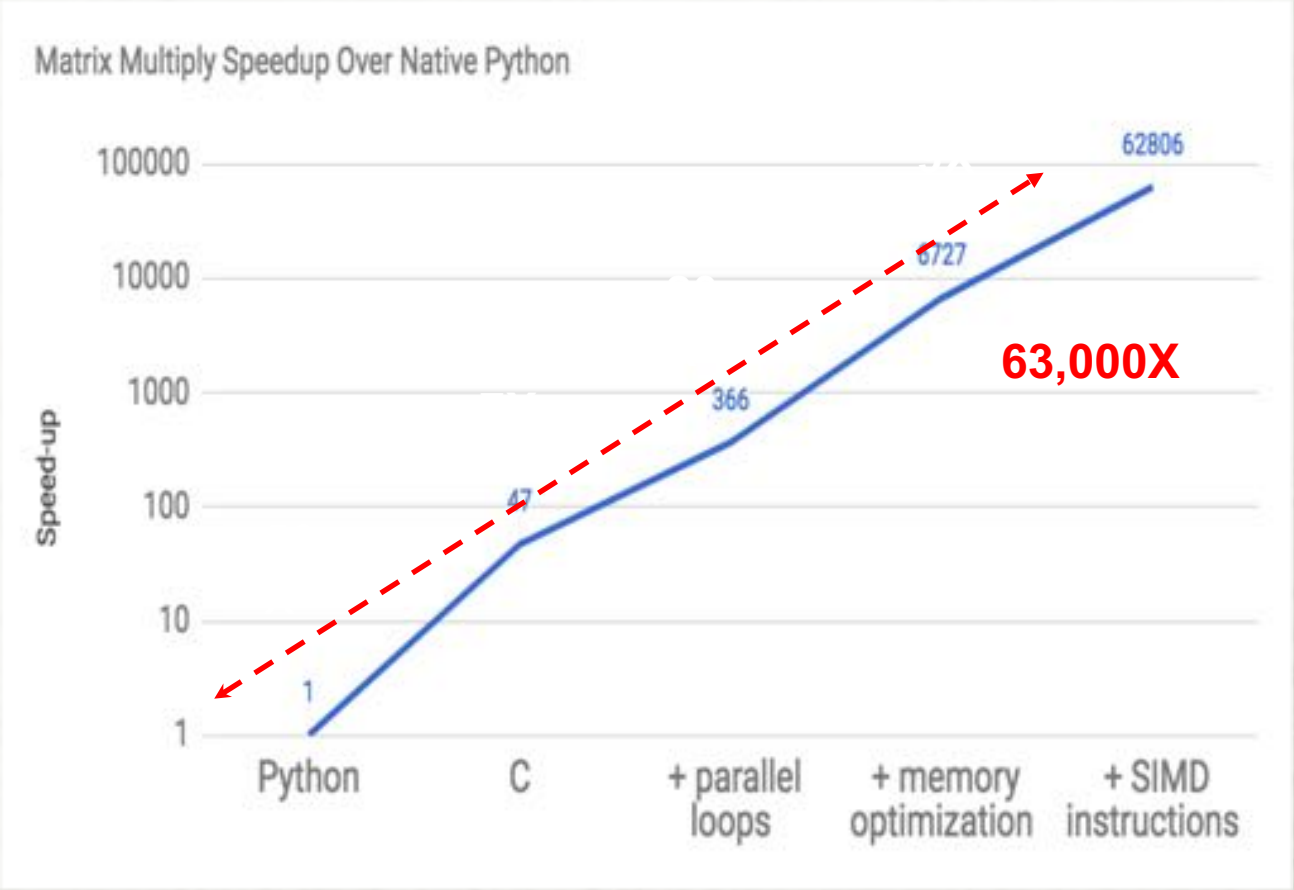
- RISC ideas were about improving efficiency:
  - 1980s: efficiency in use of transistors
    - Less significant in CPUs in 1990s: processors dominated by other things & Moore/Dennard in full operation
- RISC comeback: driven by mobile world in 2000s:
  - Energy efficiency crucial: small batteries, all day operation
  - Si efficiency important for cost!
- 2020 and on: Add Design Efficiency
  - With growing spectrum of designs targeted to specific applications, efficiency of design/verification increasing.

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 the architecture to characteristics of the domain
  - Not one application, but a domain of applications
    - Different from strict ASIC
  - Requires more domain-specific knowledge then GP 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
- Good news: demand for performance focused on such domains

Where does the energy go in GP CPUs? Can DSAs do Better?

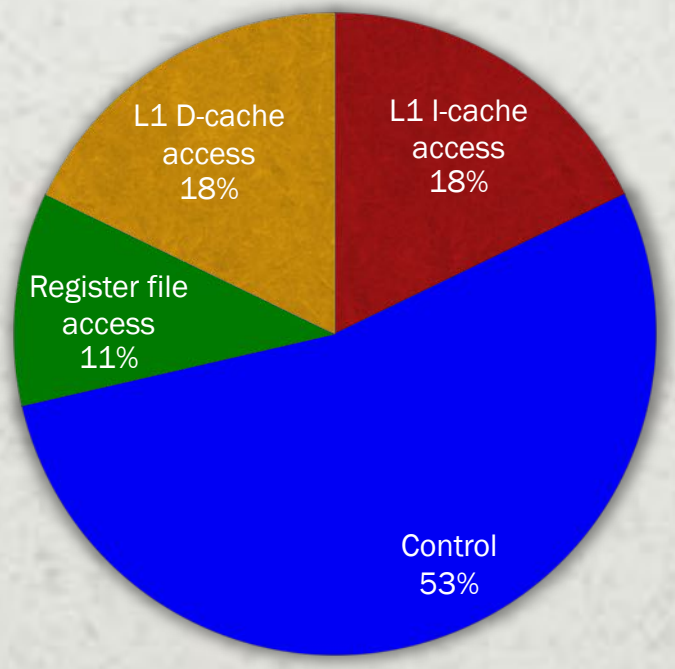
Function	Energy in Picojoules
8-bit add	0.03
32-bit add	0.1
FP Multiply 16-bit	1.1
FP Multiply 32-bit	3.7
Register file access*	6
Control (per instruction, superscalar)	20-40
L1 cache access	10
L2 cache access	20
L3 cache access	100
Off-chip DRAM access	1,300-2,600

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

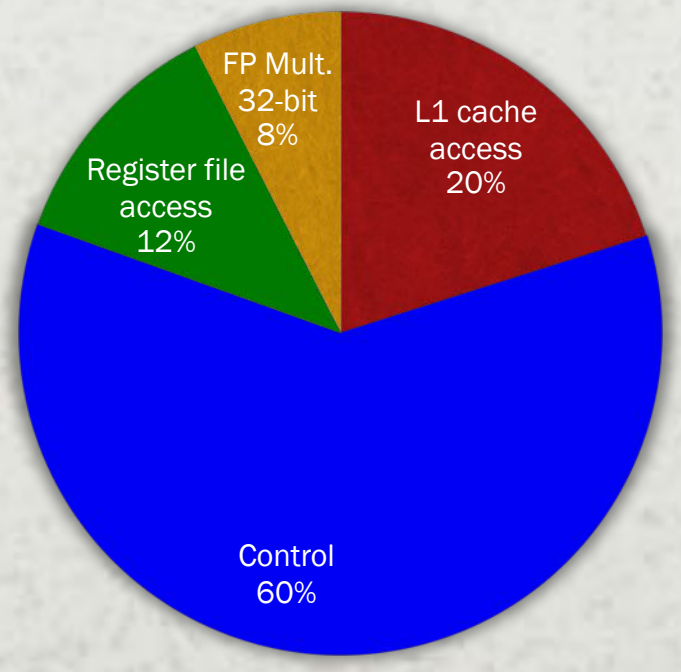
From Horowitz [2016].

Instruction Energy Breakdown

Load Register (from L1 Cache)



FP Multiply (32-bit) from registers



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

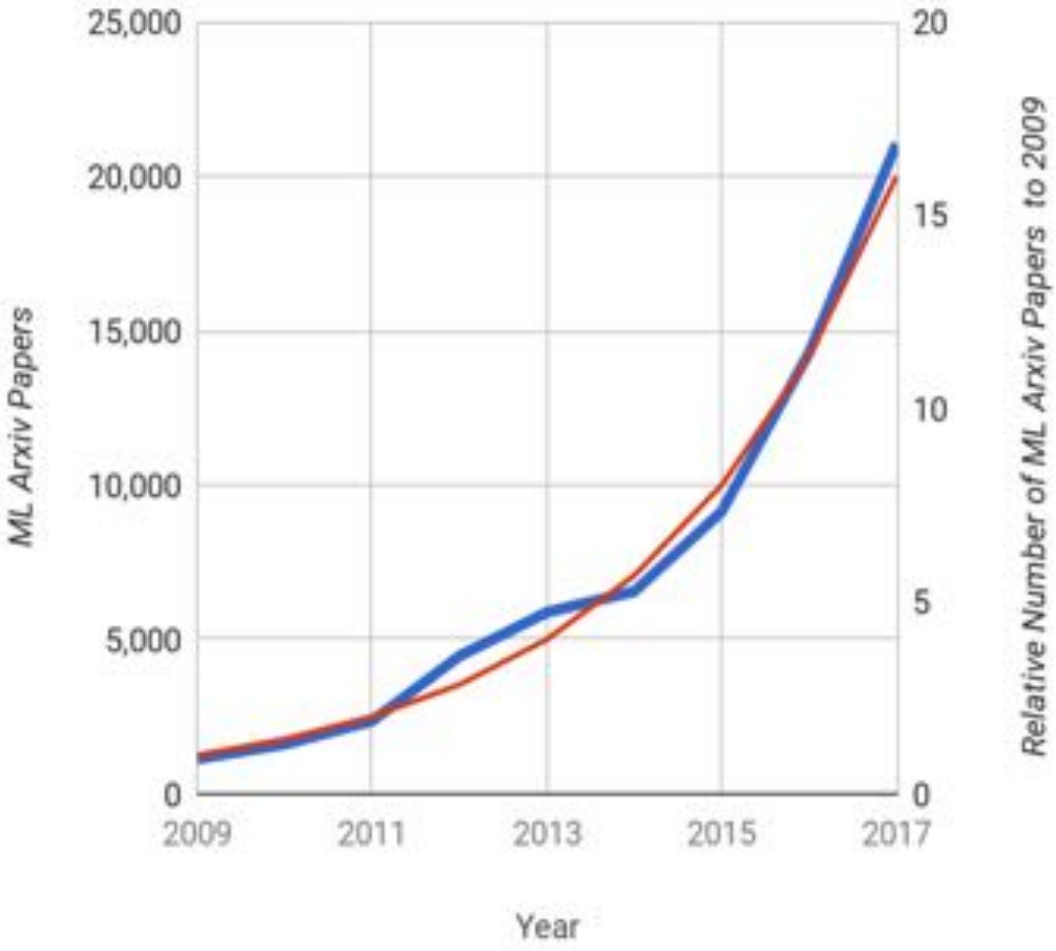
Domain Specific Languages

DSAs require targeting high level operations to 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

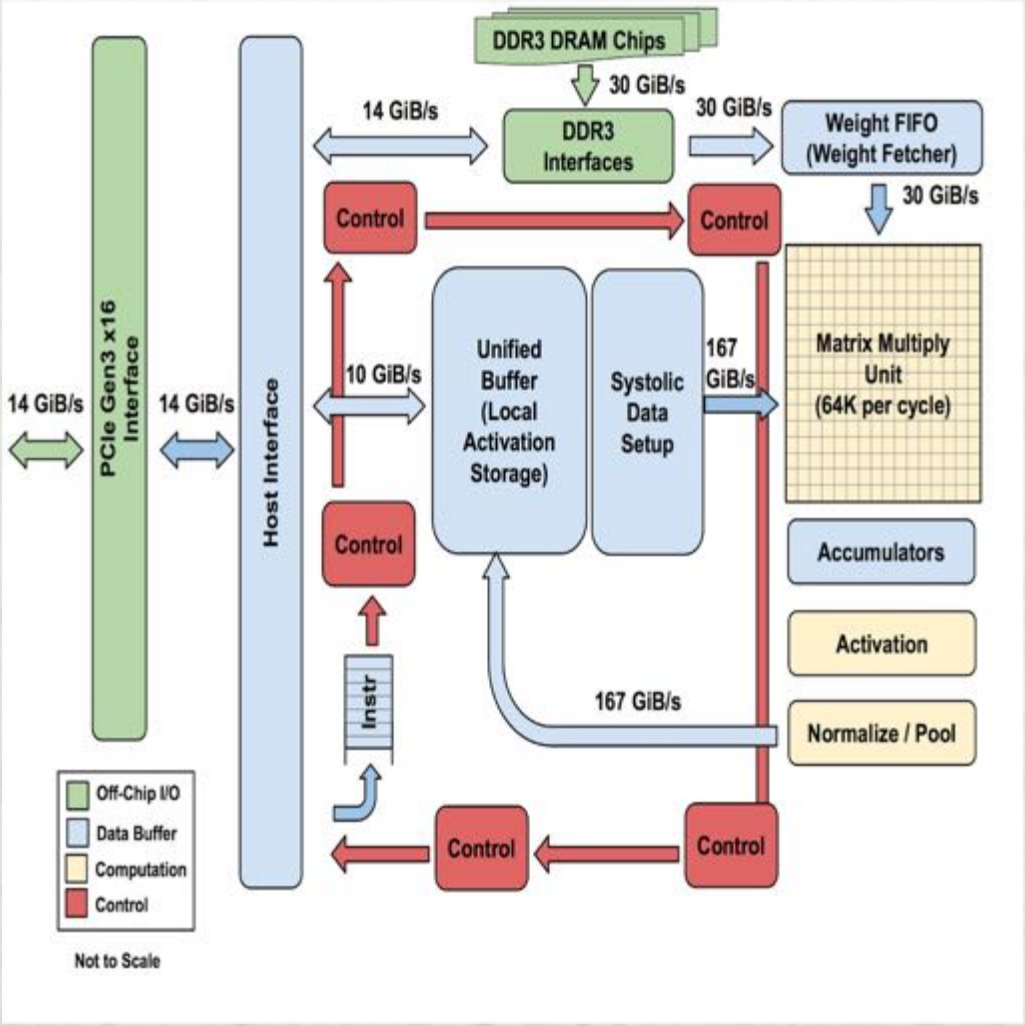
Deep learning is causing a machine learning revolution



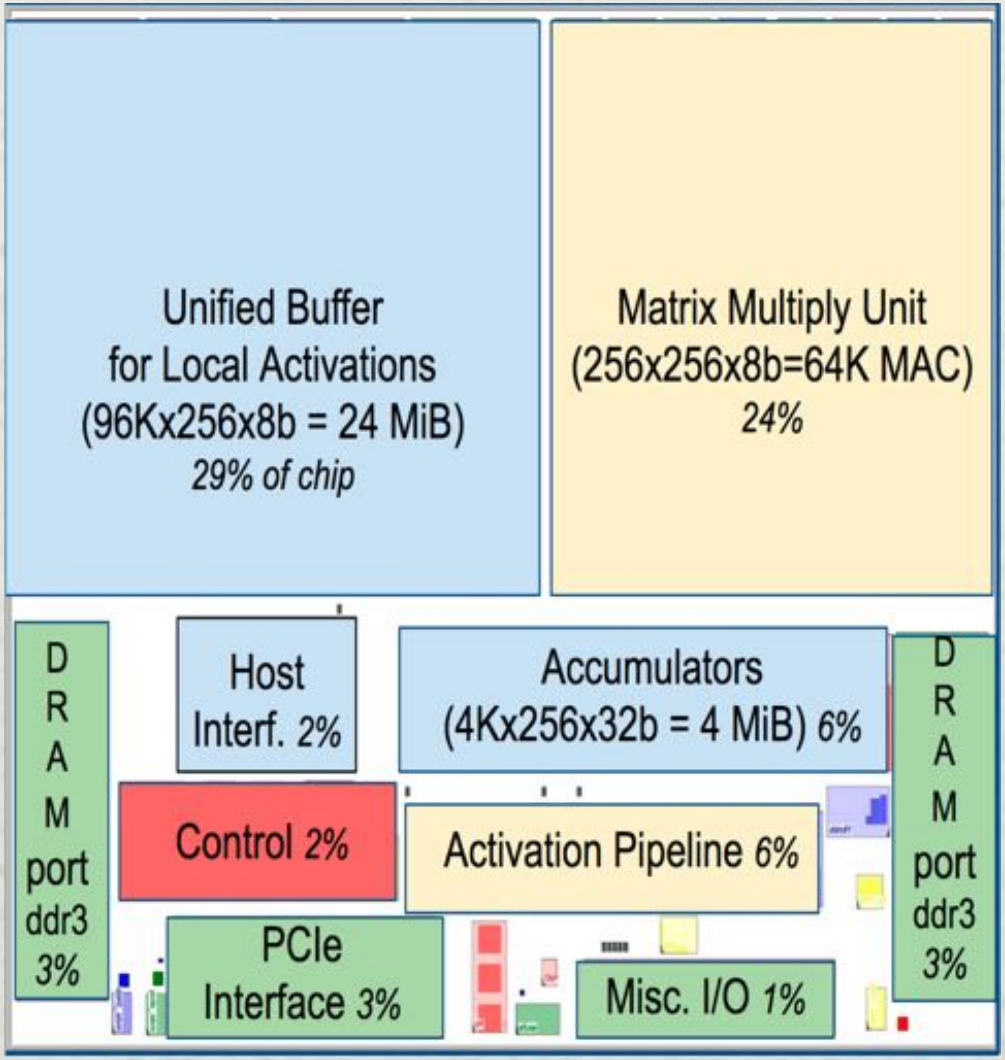
From "A New Golden Age in Computer Architecture: Empowering the Machine-Learning Revolution." Dean, J., Patterson, D., & Young, C. (2018). IEEE Micro, 38(2), 21-29.

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 Accumulator memory
- 24 MiB of on-chip Unified Buffer (activation memory)
- 3.5X as much on-chip memory vs. GPU



How is Silicon Used: TPU-1 vs. CPU?

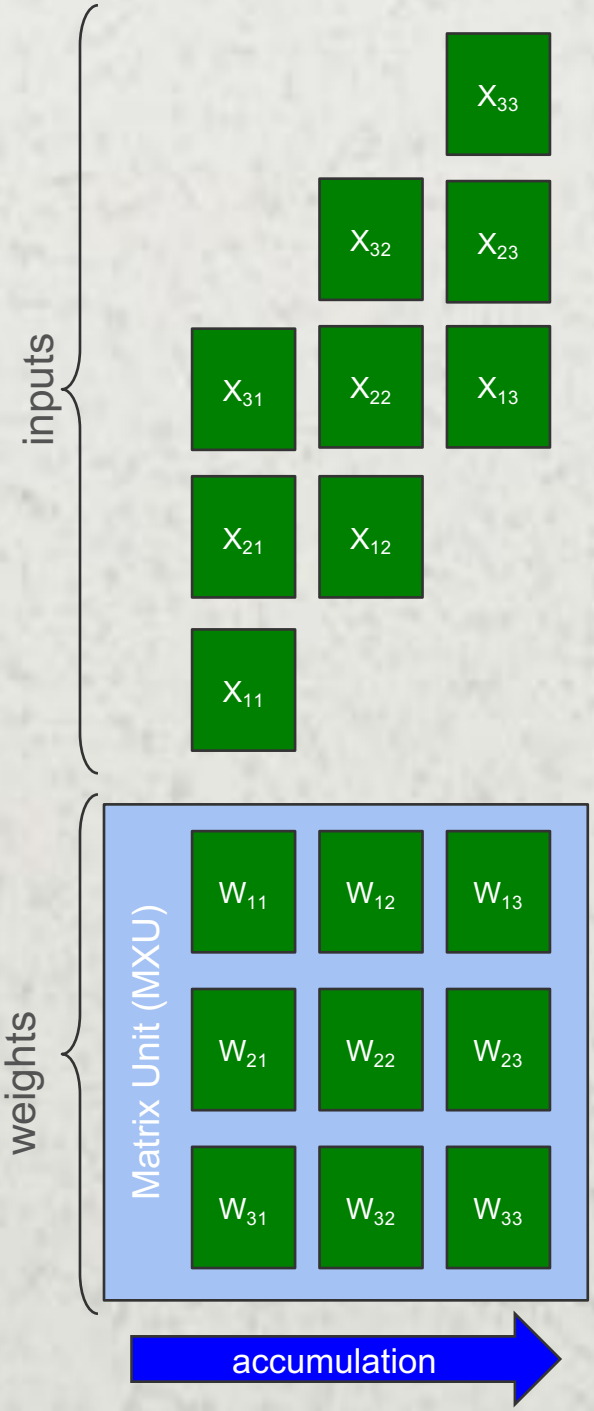


TPU-1 (–pads)

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

CPU (Skylake core)

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



Matrix Unit Systolic Array

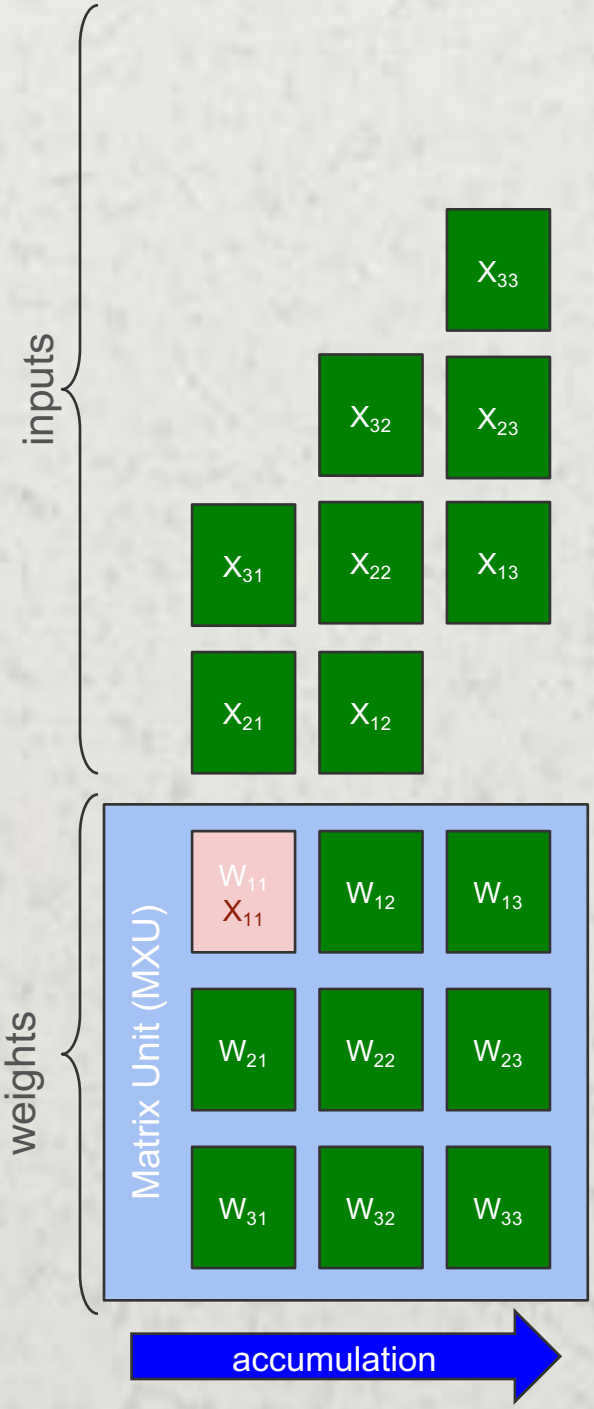
(Kung & Leiserson)

Computing Y = WX

3x3 systolic array
W = 3x3 matrix

Systolic slides courtesy of Cliff Young @ Google.

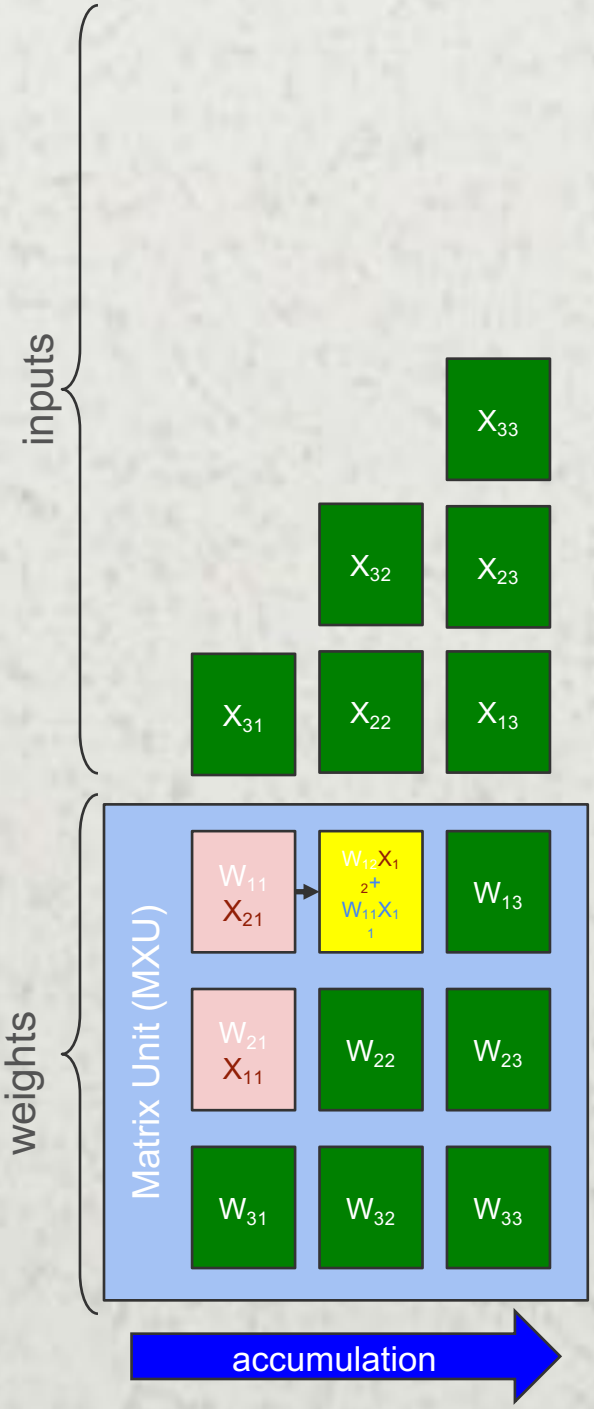
Matrix Unit Systolic Array



Matrix Unit Systolic Array

Computing Y = WX

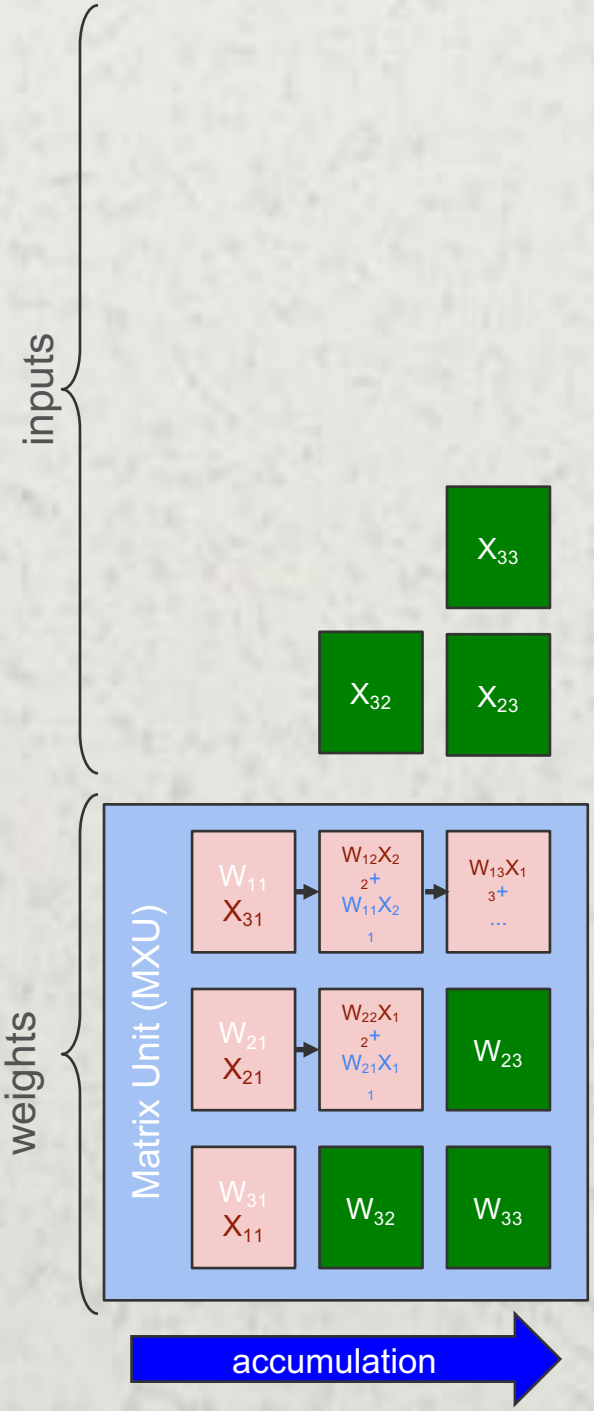
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

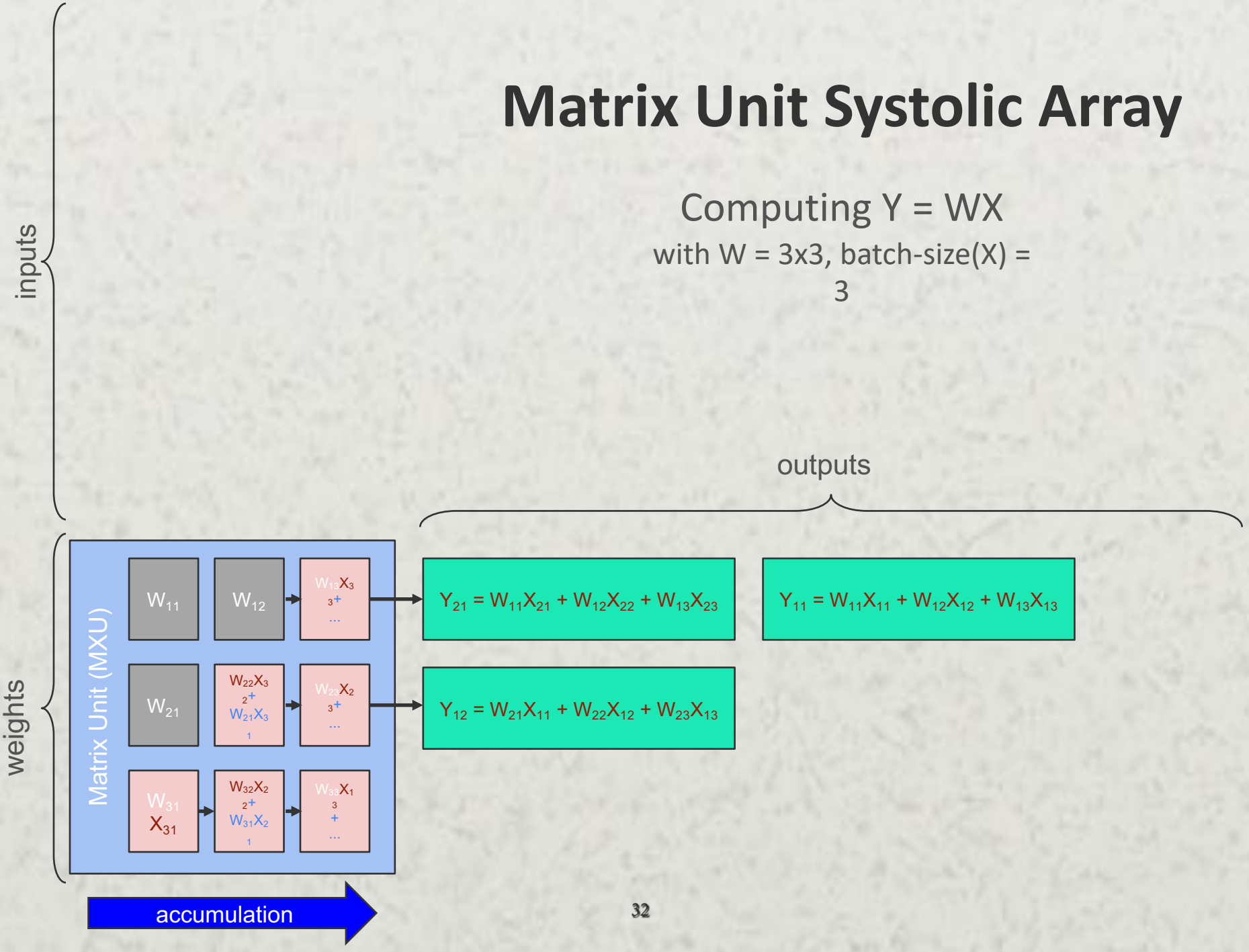
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

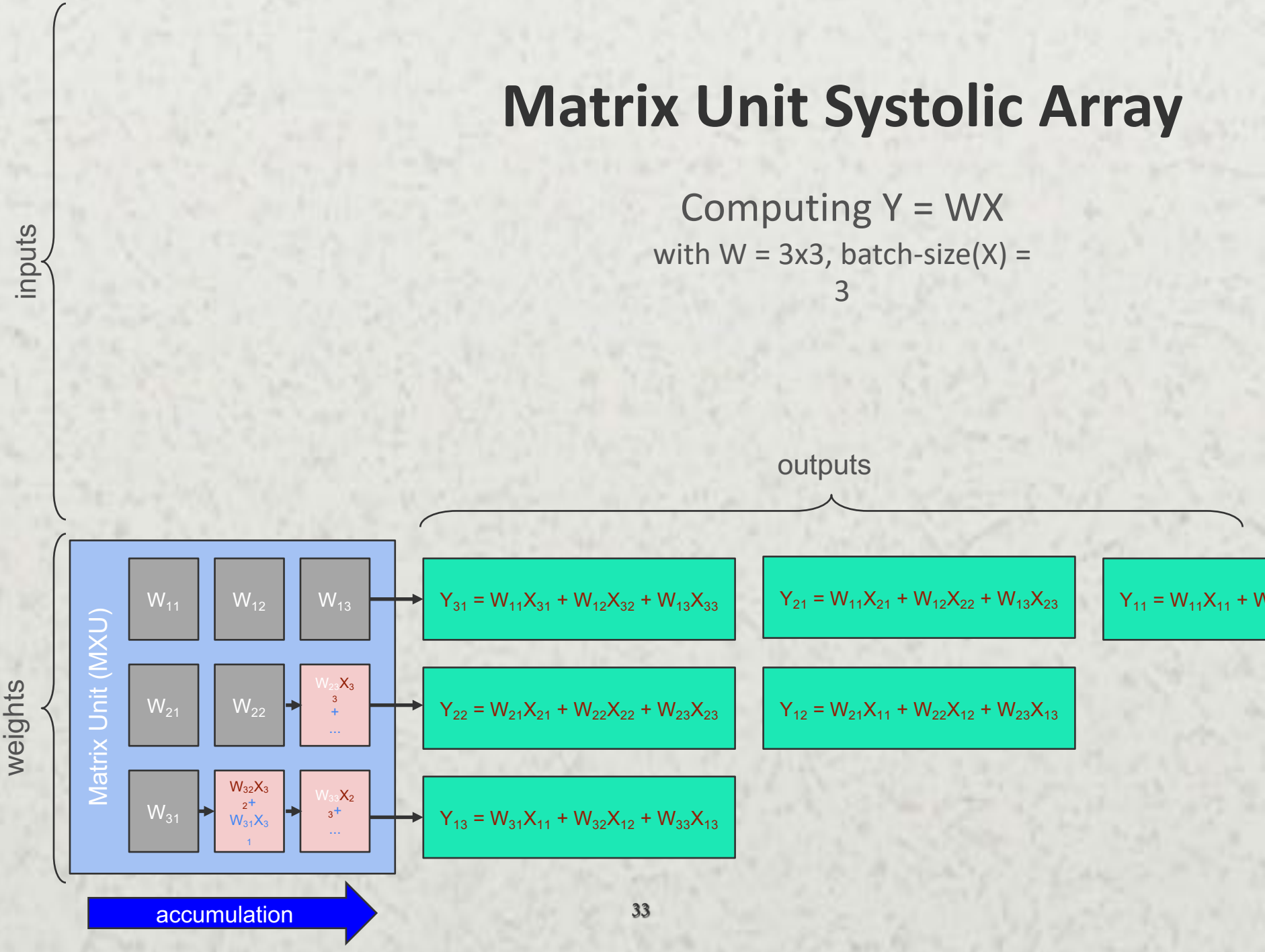
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

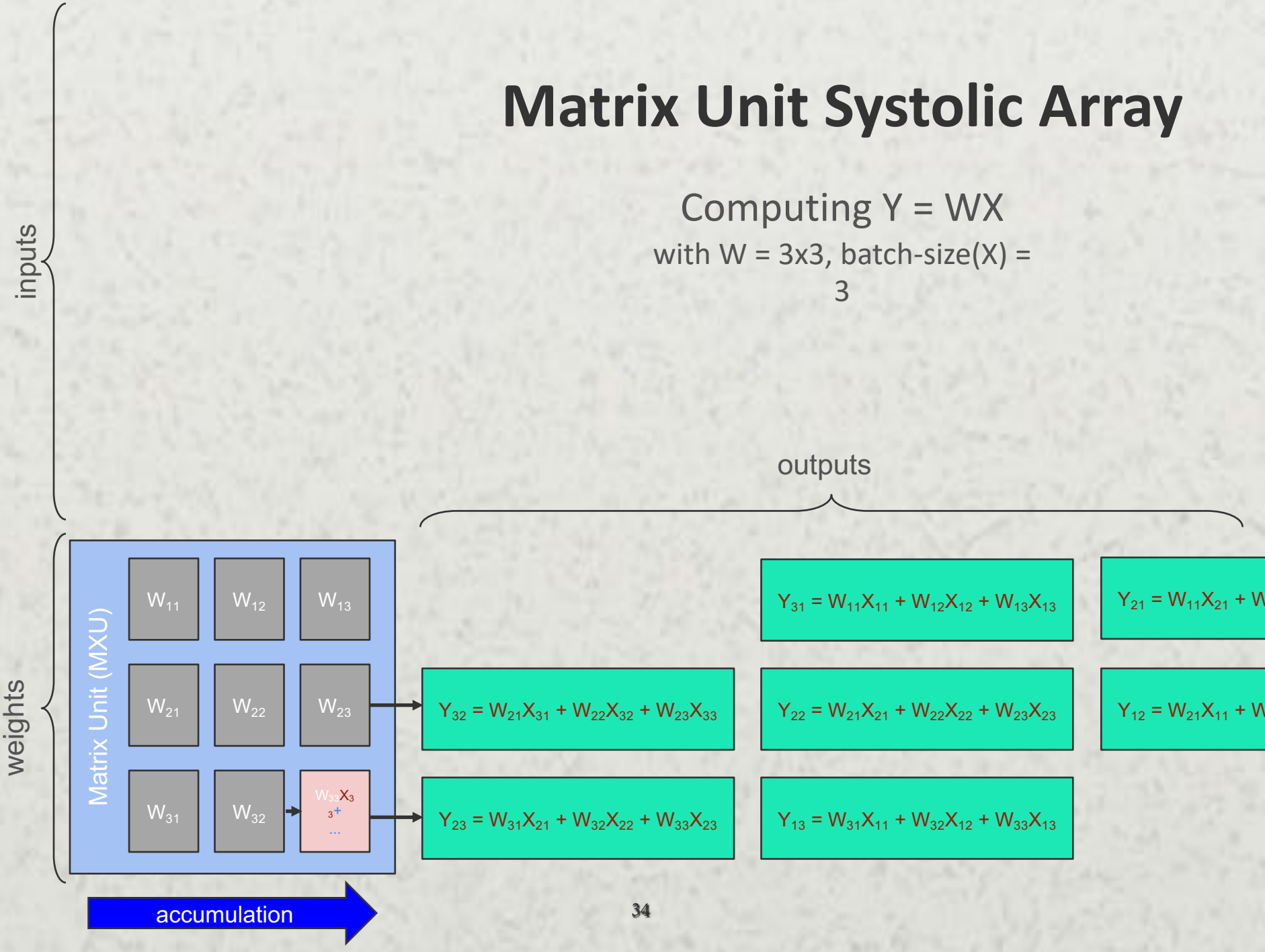
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

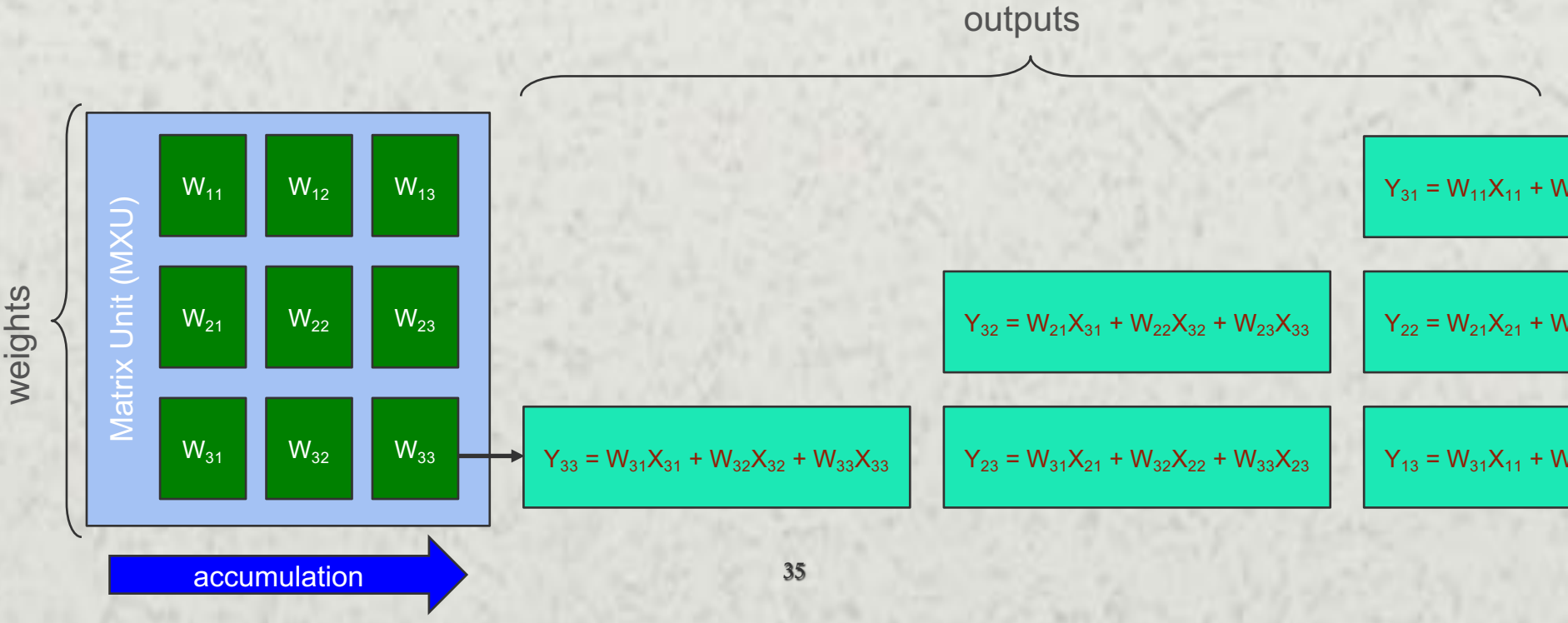
Computing Y = WX

with W = 3x3, batch-size(X) = 3

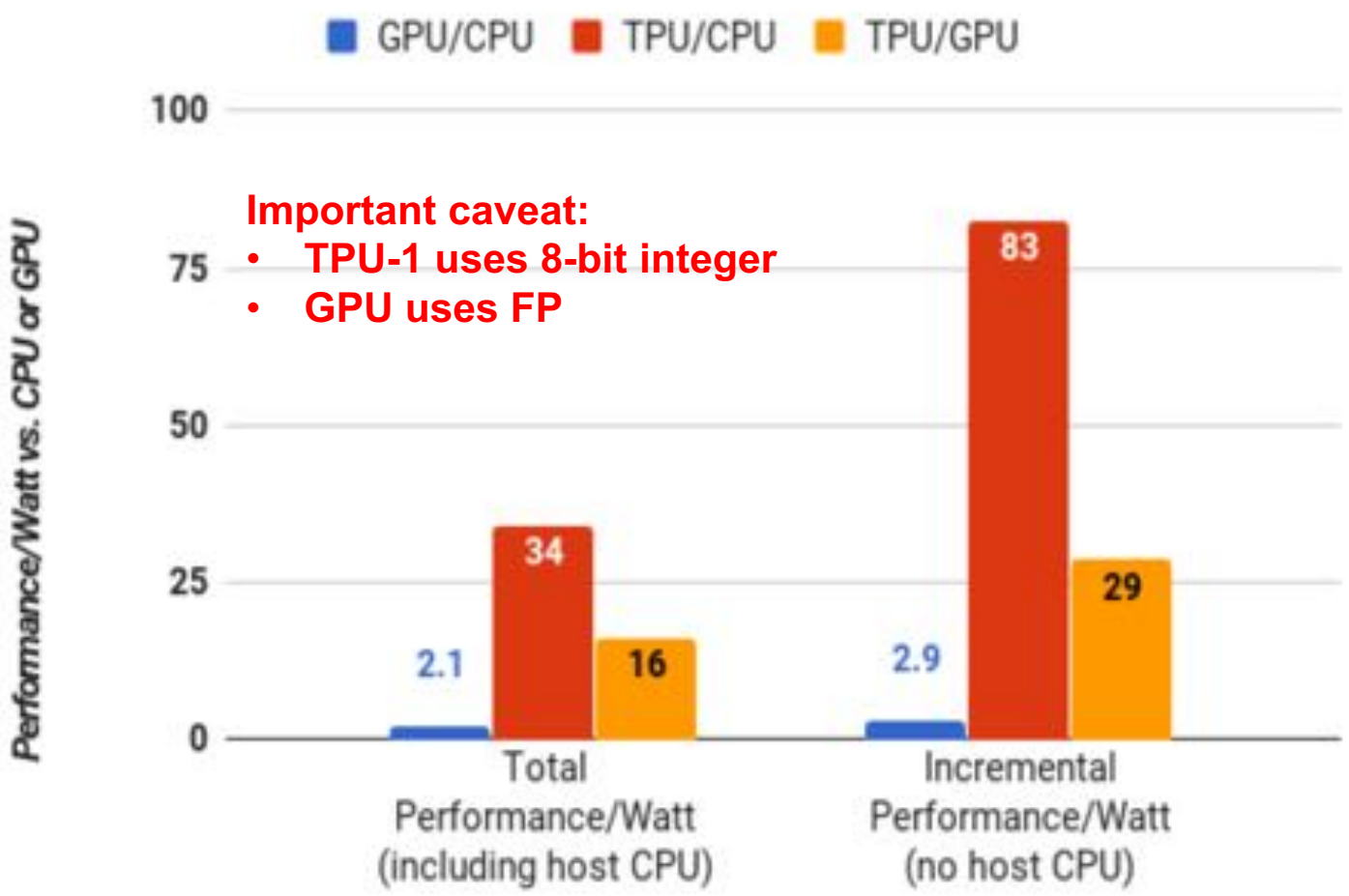


Advantages: Matrix Unit Systolic Array in TPU-1

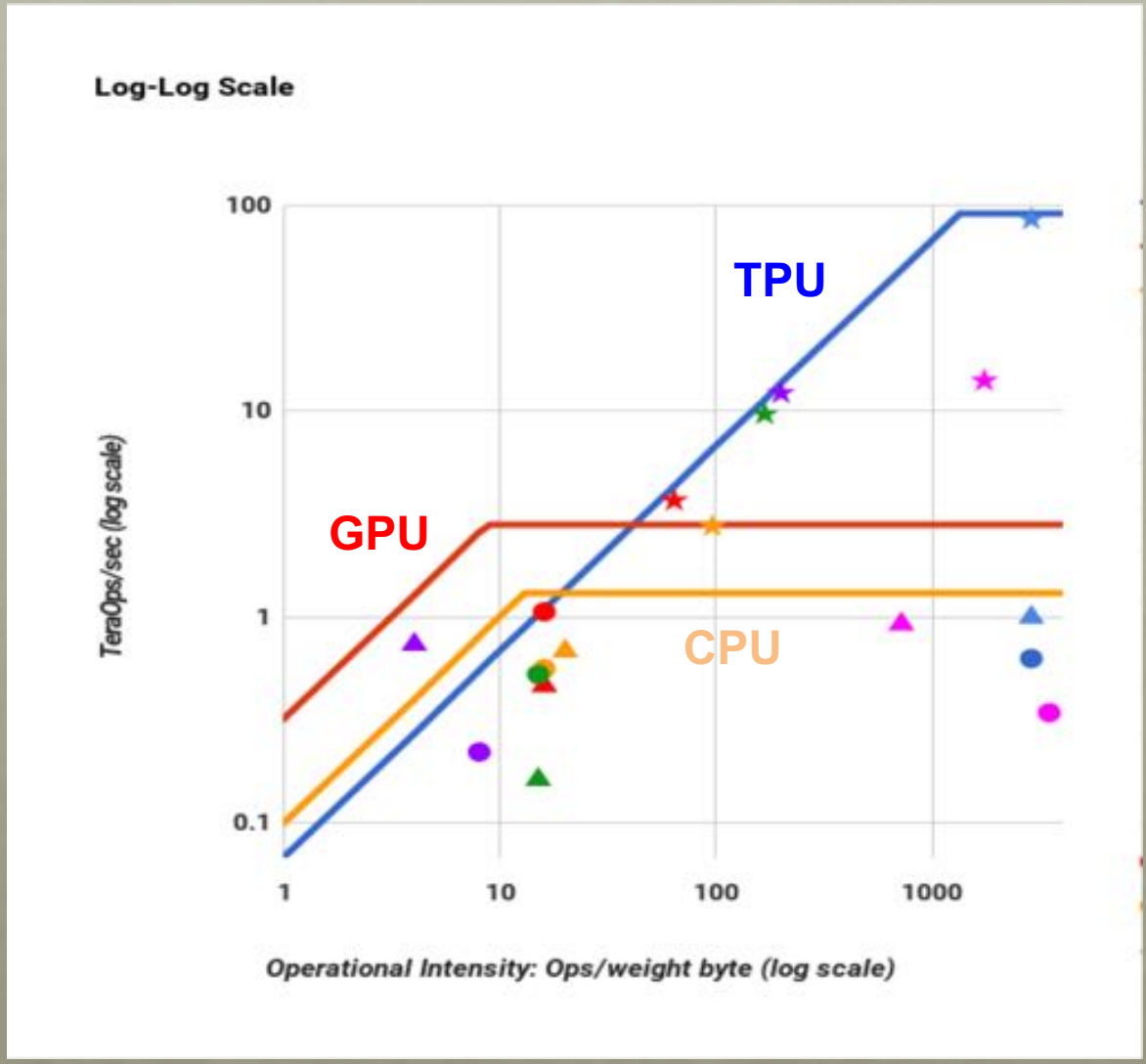
- Each operand is used up to 256 times!
- Nearest-neighbor communication replaces RF access:
  - Eliminate many reads/writes; reduce long wire delays
- For 64K Matrix Unit:
  - Energy from eliminating register access > energy of Matrix Unit!



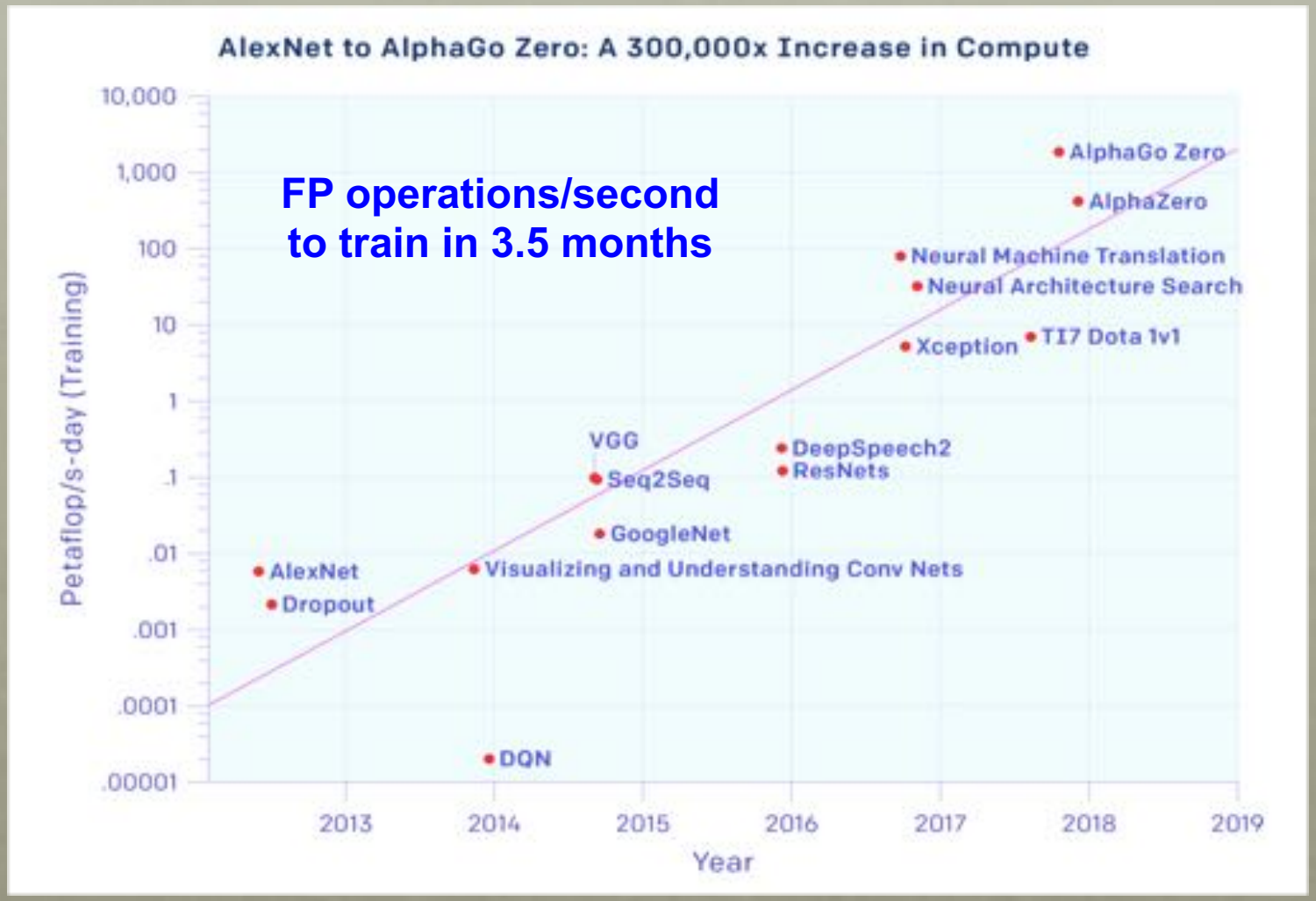
Performance/Watt on Inference TPU-1 vs CPU & GPU



Log Rooflines for CPU, GPU, TPU



Training: A Much More Intensive Problem

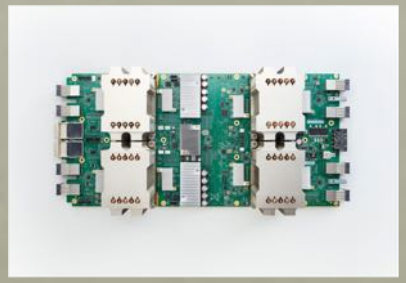


Rapid Innovation

TPU v1
(deployed 2015)

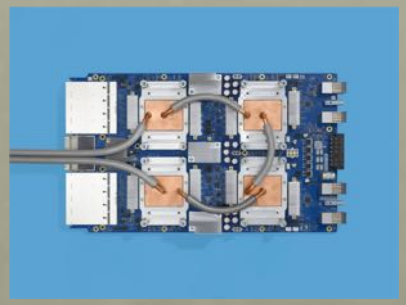
92 teraops
Inference only

Cloud TPU
(v2, Cloud GA 2017,
Pod Alpha 2018)



180 teraflops
64 GB HBM
Training and inference
Generally available (GA)

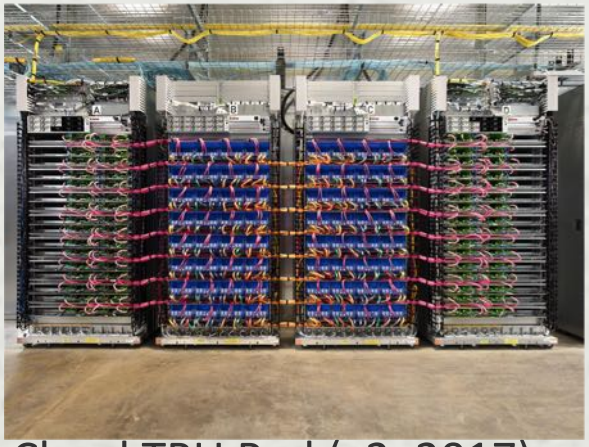
Cloud TPU
(v3, Cloud Beta 2018)



420 teraflops
128 GB HBM
Training and inference
Beta

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

Enabling Massive Computing Cycles for Training



Cloud TPU Pod (v2, 2017)

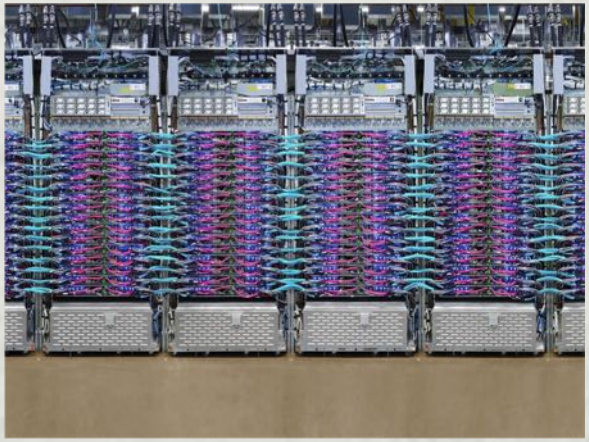
11.5 petaflops (64xTPU v2)

4 TB HBM

2-D toroidal mesh network

Training and inference

Alpha



TPU v3 Pod (2018)

> 100 petaflops! (256xTPU v3)

32 TB HBM

Liquid cooled

New chip architecture + larger-scale system

Challenges and Opportunities

- Design of DSAs and DSLs
  - Optimizing the mapping to a DSA for portability & performance.
  - DSAs & DSLs for new fields
  - 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)

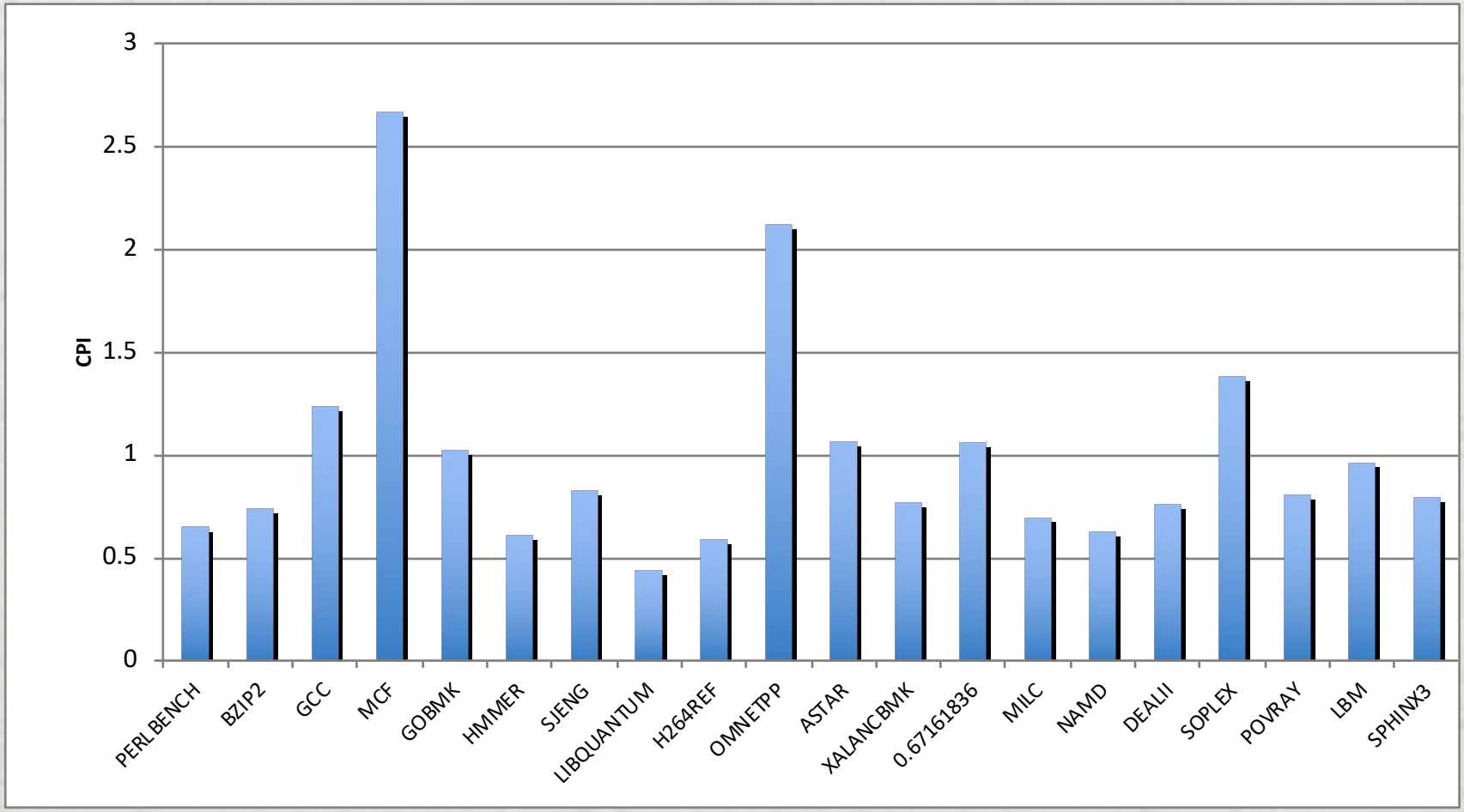
  "What I was really frustrated about was the fact, with Iliac IV, programming the machine was very difficult and the architecture probably was not very well suited to some of the applications we were trying to run. The key idea was that I did not think we had a very good match in Iliac IV between applications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

WHY DRAMS ARE HARD AND CRITICAL!

INTEL CORE I7: Theoretical CPI = 0.25
Achieved CPI



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