Lecture 11:

Hardware Specialization

Parallel Computing
Stanford CS149, Fall 2024

Energy-constrained computing

Energy (Power x Time)-constrained computing

Supercomputers are energy constrained

- Due to shear scale of machine
- Overall cost to operate (power for machine and for cooling)

Datacenters are energy constrained

- Reduce cost of cooling
- Reduce physical space requirements

Mobile devices are energy constrained

- Limited battery life
- Heat dissipation without fan

Performance and Power

Performance

Energy efficiency

$$Power = \frac{Ops}{second} \times \frac{Joules}{Op}$$

FIXED

What is the magnitude of improvement from specialization?

Specialization (fixed function) ⇒ better energy efficiency

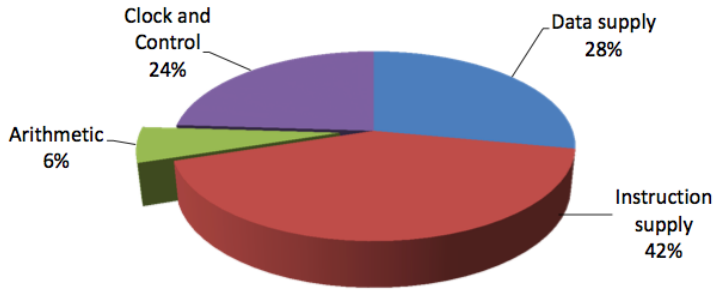
Pursuing highly efficient processing…
(specializing hardware beyond just parallel CPUs and GPUs)

Why is a "general-purpose processor" so inefficient?

Wait… this entire class we've been talking about making efficient use out of multi-core CPUs and GPUs…
and now you're telling me these platforms are "inefficient"?

Consider the complexity of executing an instruction on a modern processor…

Read instruction — Address translation, communicate with icache, access icache, etc.
Decode instruction — Translate op to uops, access uop cache, etc.
Check for dependencies/pipeline hazards
Identify available execution resource
Use decoded operands to control register file SRAM (retrieve data)
Move data from register file to selected execution resource
Perform arithmetic operation
Move data from execution resource to register file
Use decoded operands to control write to register file SRAM



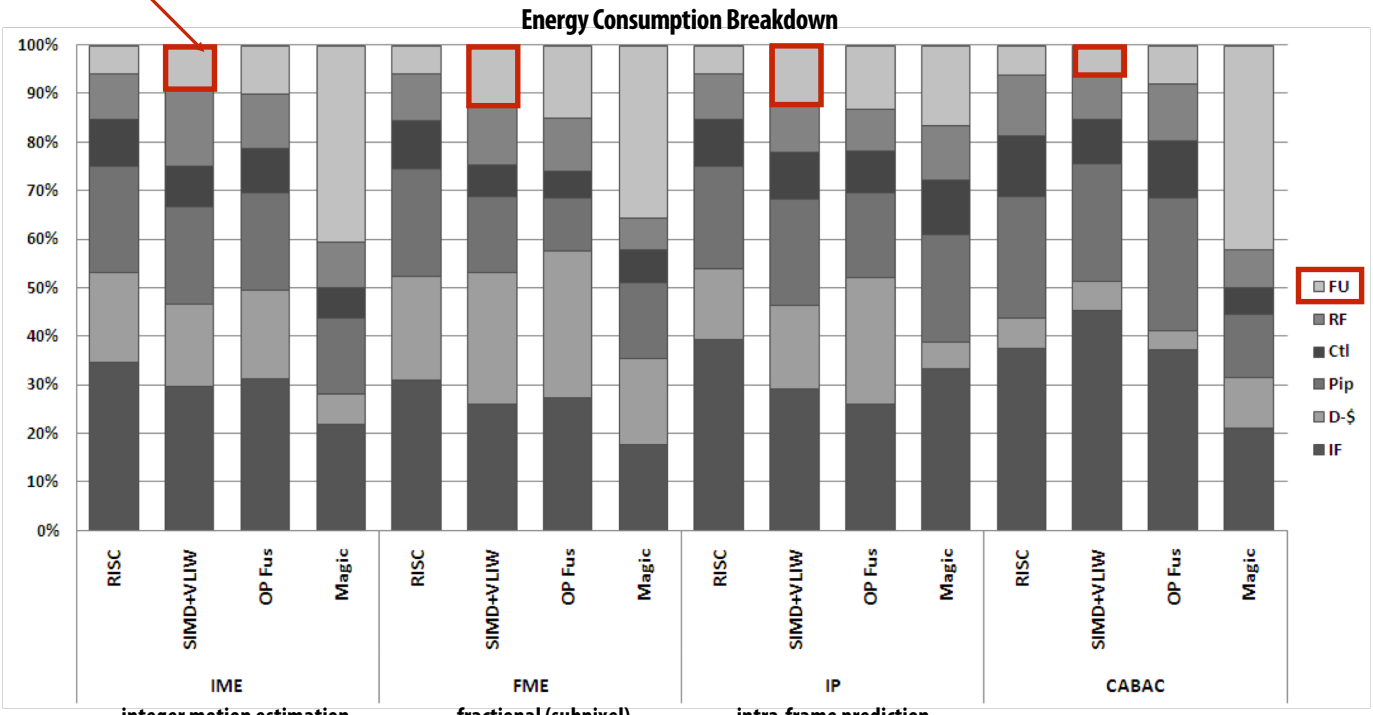
Efficient Embedded Computing [Dally et al. 08]
[Figure credit Eric Chung]

Review question:
How does SIMD execution reduce overhead of certain types of computations?
What properties must these computations have?

H.264 video encoding: fraction of energy consumed by functional units is small (even when using SIMD)

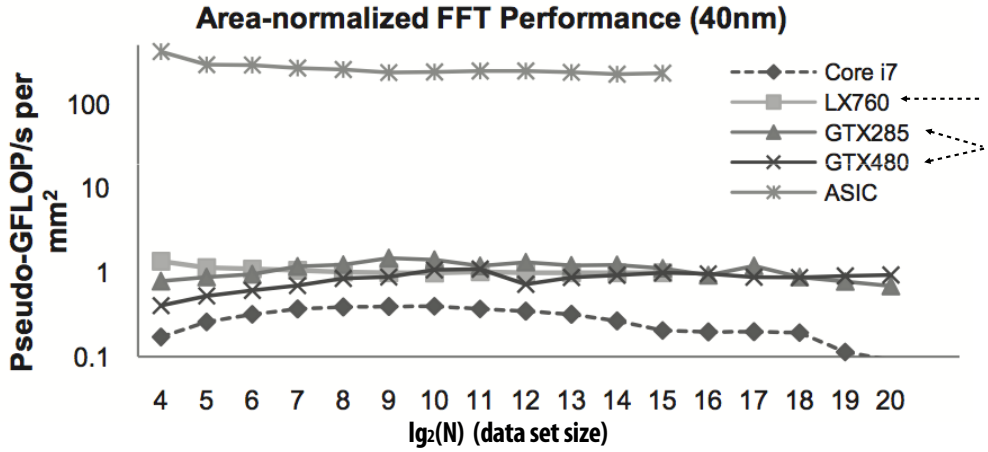
Even after encoding implemented with SIMD instruction

[Hameed et al. ISCA 2010]



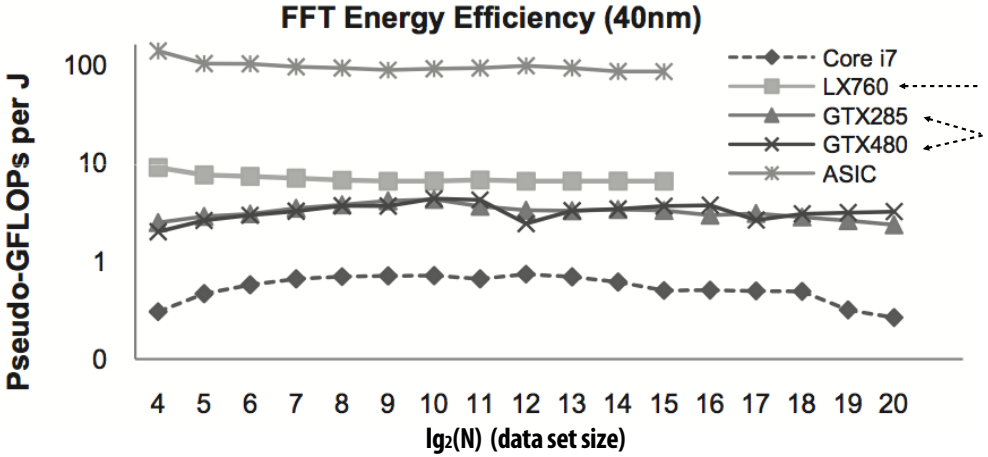
FU = functional units	Pip = pipeline registers (interstage)
RF = register fetch	D-$ = data cache
Ctrl = misc pipeline control	IF = instruction fetch + instruction cache

Fast Fourier transform (FFT): throughput and energy benefits of specialization



ASIC delivers same performance as one CPU core with ~ 1/1000th the chip area.

GPU cores: ~ 5-7 times more area efficient than CPU cores.



ASIC delivers same performance as one CPU core using only ~ 1/100th the power

[Chung et al. MICRO 2010]

Digital signal processors (DSPs)

Programmable processors, but simpler instruction stream control paths

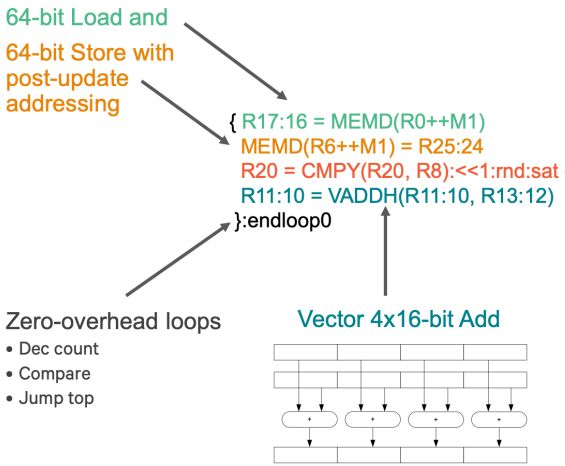
Complex instructions (e.g., SIMD/VLIW): perform many operations per instruction (amortize cost of control)

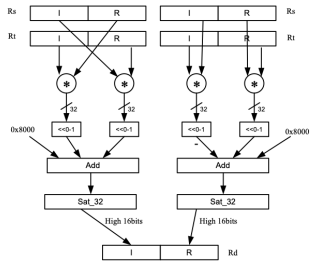
Example: Qualcomm Hexagon DSP

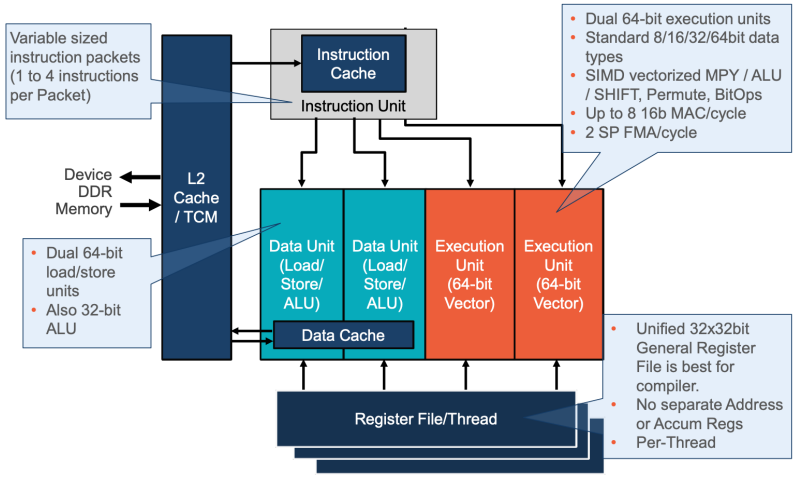
Used for modem, audio, and (increasingly) image processing on Qualcomm Snapdragon SoC processors

VLIW: "very-long instruction word"
Single instruction specifies multiple different operations to do at once (contrast to SIMD)

Below: innermost loop of FFT
Hexagon DSP performs 29 "RISC" ops per cycle







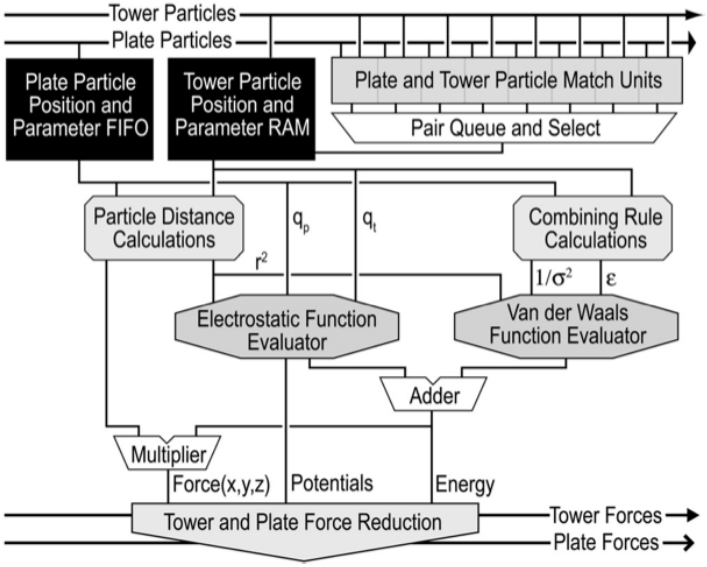
Hexagon DSP is in Google Pixel phone

Anton supercomputer for molecular dynamics

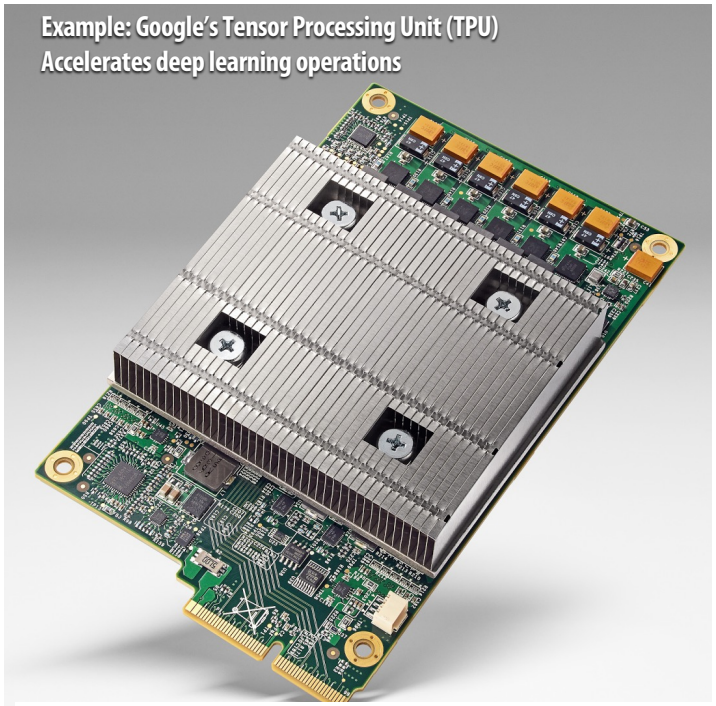
[Developed by DE Shaw Research]

- Simulates time evolution of proteins
- ASIC for computing particle-particle interactions (512 of them in machine)
- Throughput-oriented subsystem for efficient fast-fourier transforms

Custom, low-latency communication network designed for communication patterns of N-body simulations



Specialized processors for evaluating deep networks



AI & Machine Learning

Google supercharges machine learning tasks with TPU custom chip

May 18, 2016

Norm Jouppi
Google Fellow, Google

Countless papers followed at top computer architecture research conferences on the topic of ASICs or accelerators for deep learning or evaluating deep networks…

- **Cambricon: an instruction set architecture for neural networks**, Liu et al. ISCA 2016
- **EIE: Efficient Inference Engine on Compressed Deep Neural Network**, Han et al. ISCA 2016
- **Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing**, Albericio et al. ISCA 2016
- **Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators**, Reagen et al. ISCA 2016
- **vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design**, Rhu et al. MICRO 2016
- **Fused-Layer CNN Architectures**, Alwani et al. MICRO 2016
- **Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Network**, Chen et al. ISCA 2016
- **PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory**, Chi et al. ISCA 2016
- **DNNWEAVER: From High-Level Deep Network Models to FPGA Acceleration**, Sharma et al. MICRO 2016

FPGAs (Field Programmable Gate Arrays)

- Middle ground between an ASIC and a processor
- FPGA chip provides array of logic blocks, connected by interconnect
- Programmer-defined logic implemented directly by FGPA

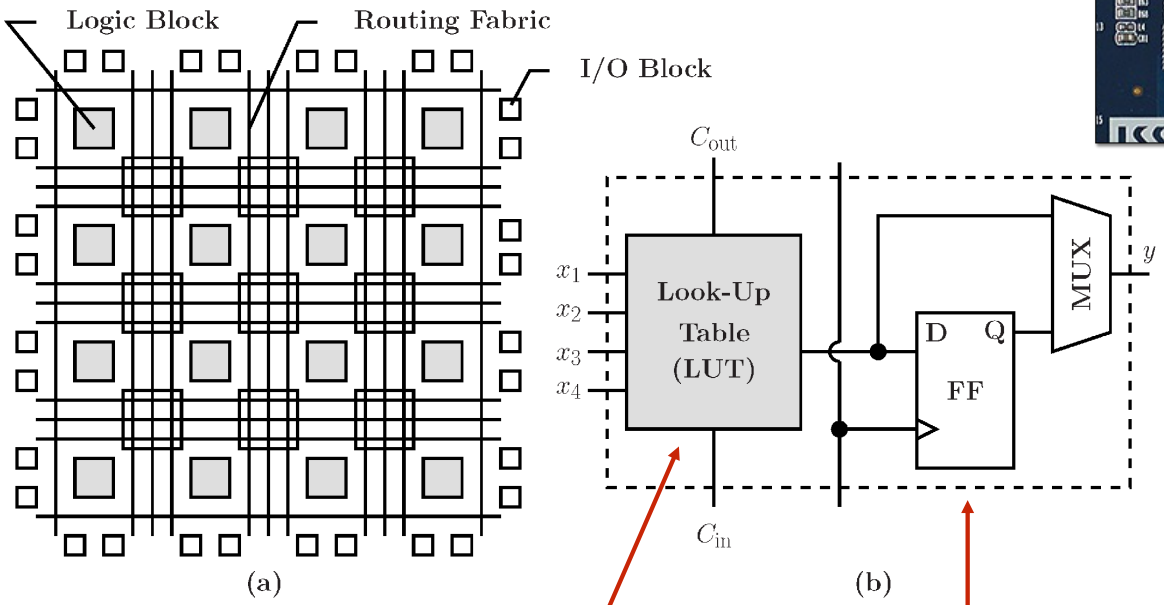
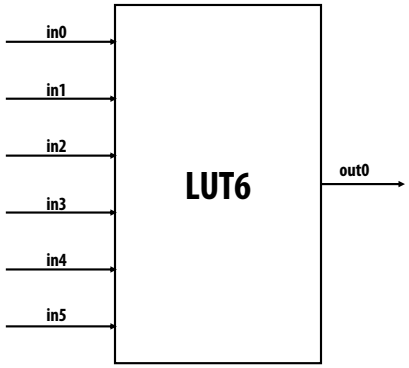


Image credit: Bai et al. 2014

Specifying combinational logic as a LUT

- Example: 6-input, 1 output LUT in Xilinx Virtex-7 FPGAs
  - Think of a LUT6 as a 64 element table



Example:
6-input AND

In	Out
0	0
1	0
2	0
3	0
⋮	⋮
63	1

40-input AND constructed by chaining outputs of eight LUT6's (delay = 3)

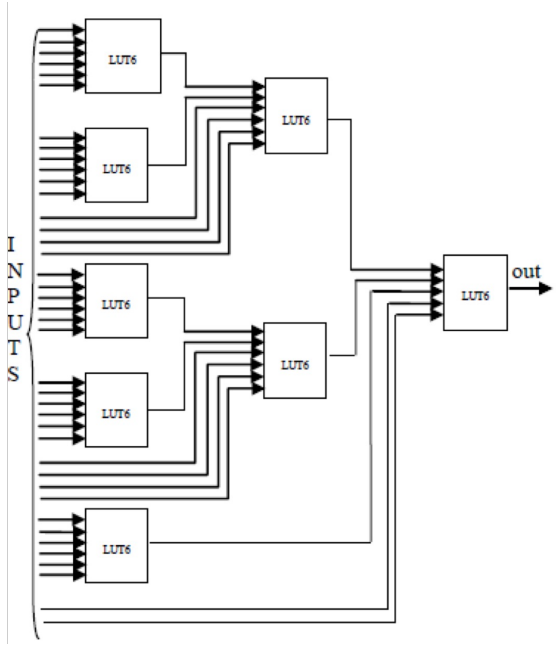
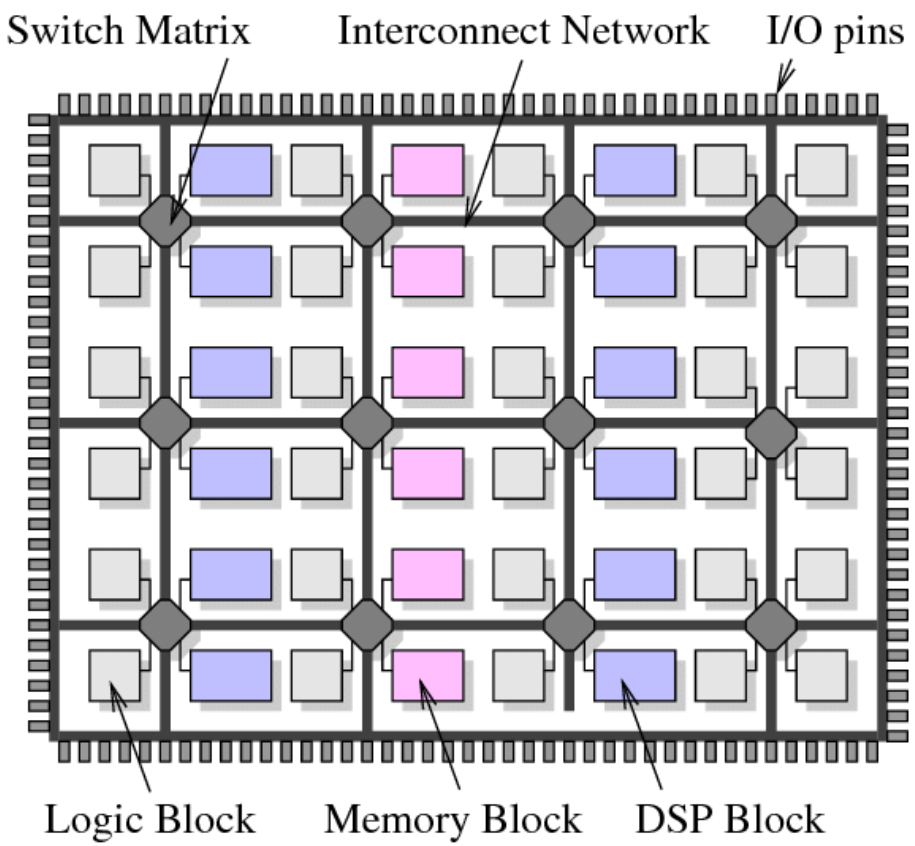


Image credit: [Zia 2013]

Modern FPGAs



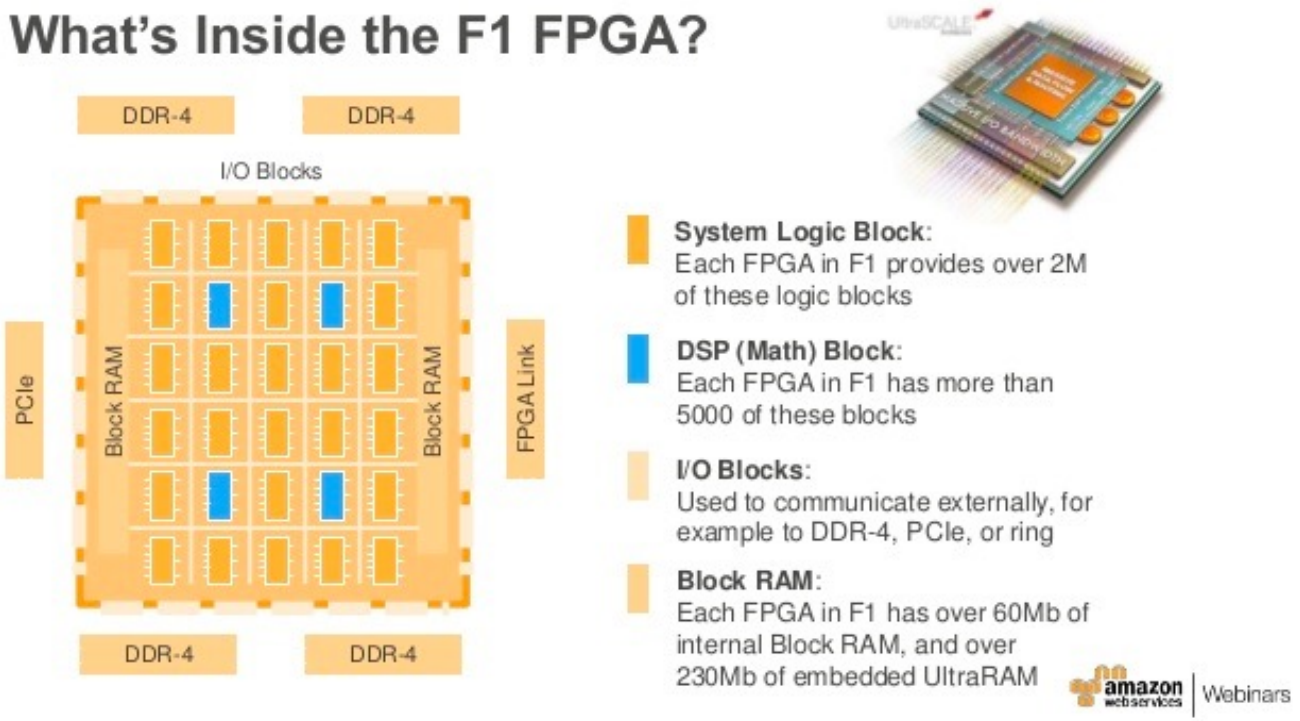
A lot of area devoted to hard gates

- **Memory blocks (SRAM)**
- **DSP blocks (multiplier)**

Program with a hardware description language (e.g. Verilog, EE108)

Amazon EC2 F1

- FPGA's are now available on Amazon cloud services



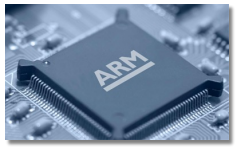
Efficiency benefits of compute specialization

- Rules of thumb: compared to high-quality C code on CPU...
- Throughput-maximized processor architectures: e.g., GPU cores
  - Approximately 10x improvement in perf / watt
  - Assuming code maps well to wide data-parallel execution and is compute bound

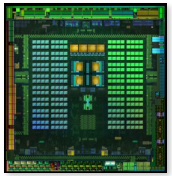
- Fixed-function ASIC ("application-specific integrated circuit")
  - Can approach 100-1000x or greater improvement in perf/watt
  - Assuming code is compute bound and is not floating-point math

Choosing the right tool for the job

Energy-optimized CPU



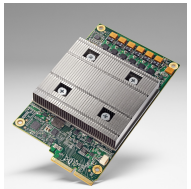
Throughput-oriented processor (GPU)



Programmable DSP

HEXAGON™

Domain Specific Accelerator



Google TPU

FPGA/ reconfigurable logic

ASIC

Video encode/decode, Audio playback, Camera RAW processing, neural nets (future?)

	~10X more efficient	~20X	~50X??? (jury still out)	~100-1000X more efficient
Easiest to program		Limited domain of programmability with DSLs (e.g. DNN)	Difficult to program (making it easier is active area of research)	Not programmable + costs 10-100's millions of dollars to design / verify / create

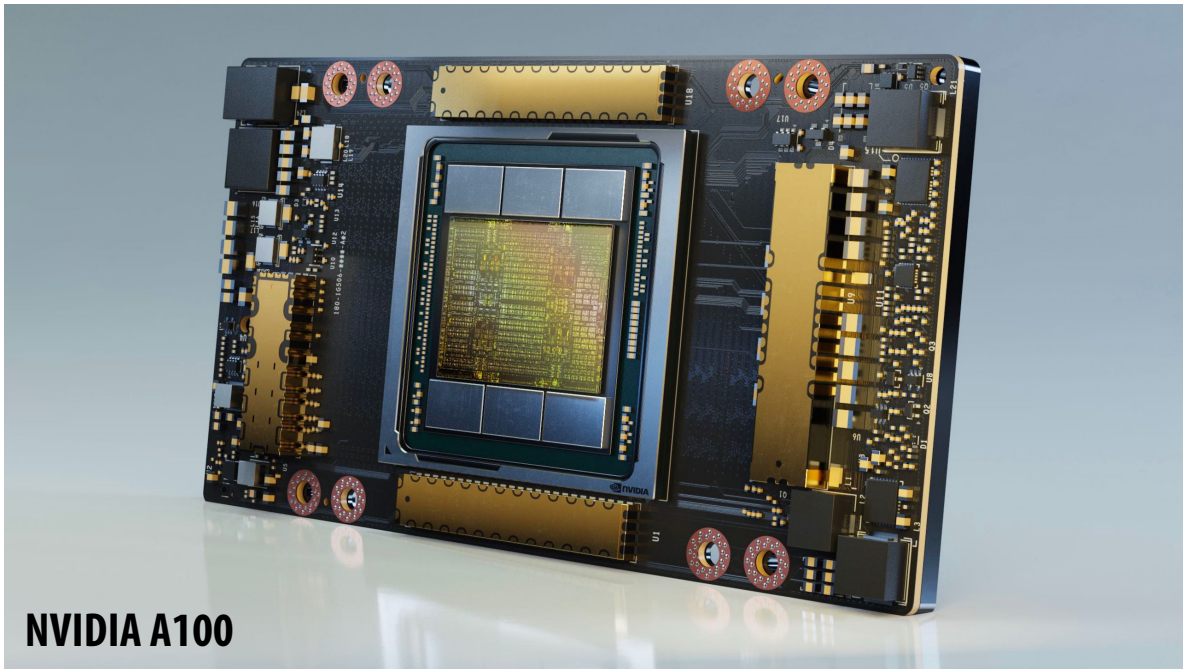
Credit: Pat Hanrahan for this slide design

Why might a GPU be a good platform for DNN evaluation?

consider:
arithmetic intensity, SIMD, data-parallelism,
memory bandwidth requirements

Deep neural networks on GPUs

- **Many high-performance DNN implementations target GPUs**
  - **High arithmetic intensity computations (computational characteristics similar to dense matrix-matrix multiplication)**
  - **Benefit from flop-rich GPU architectures**
  - **Highly-optimized library of kernels exist for GPUs (cuDNN)**



NVIDIA A100

Why might a GPU be a sub-optimal platform for DNN evaluation?

(Hint: is a general purpose processor needed?)

Special instruction support

Recall: compute specialization = energy efficiency

Rules of thumb: compared to high-quality C code on CPU…

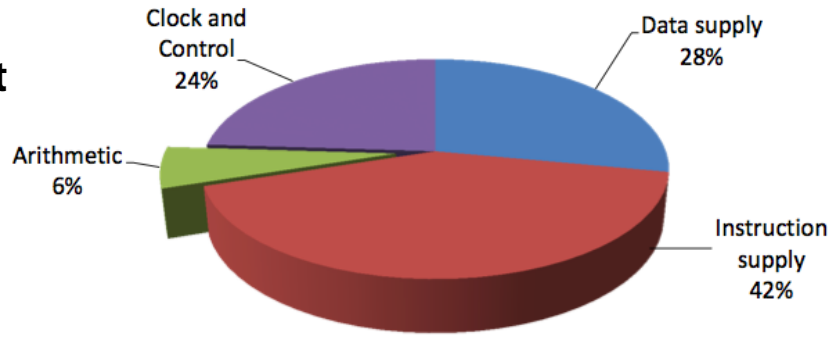
Throughput-maximized processor architectures: e.g., GPU cores

- **Approximately 10x improvement in perf / watt**
- **Assuming code maps well to wide data-parallel execution and is compute bound**

Fixed-function ASIC ("application-specific integrated circuit")

- **Can approach 100-1000x or greater improvement in perf/watt**
- **Assuming code is compute bound and**

  and is not floating-point math



Efficient Embedded Computing [Dally et al. 08]

[Figure credit Eric Chung]

[Source: Chung et al. 2010 , Dally 08]

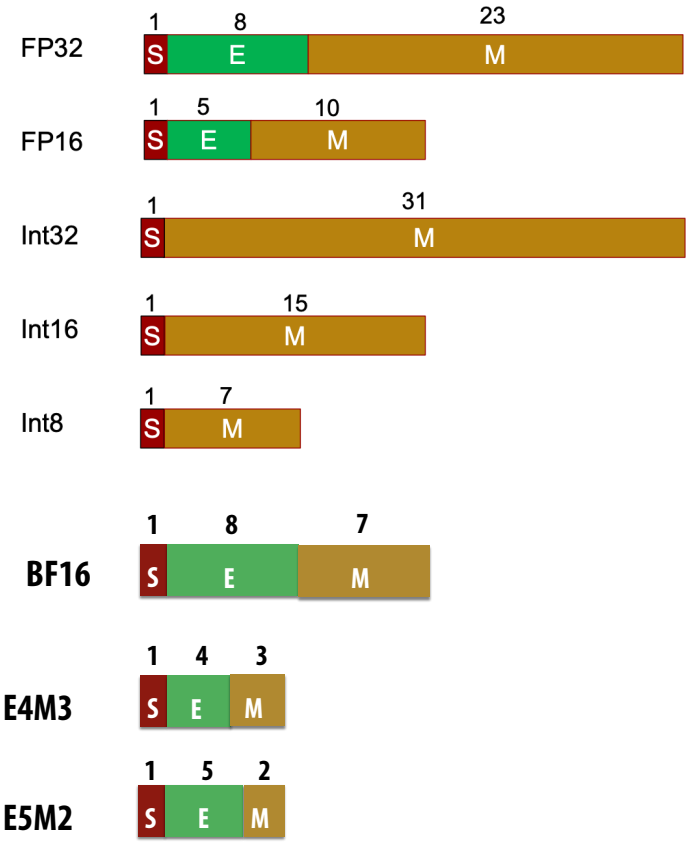
Amortize overhead of instruction stream control using more complex instructions

Estimated overhead of programmability (instruction stream, control, etc.)

- **Half-precision FMA (fused multiply-add)** **2000%**
- **Half-precision DP4 (vec4 dot product)** **500%**
- **Half-precision 4x4 MMA (matrix-matrix multiply + accumulate)** **27%**

Key principle: amortize cost of instruction stream processing across many operations of a single complex instruction

Numerical data formats



Reminder:

$$-1^{S} \times (1 + (M \times 2^{-23})) \times 2^{(E-127)}$$

BF16: Same range as FP32, but lower accuracy

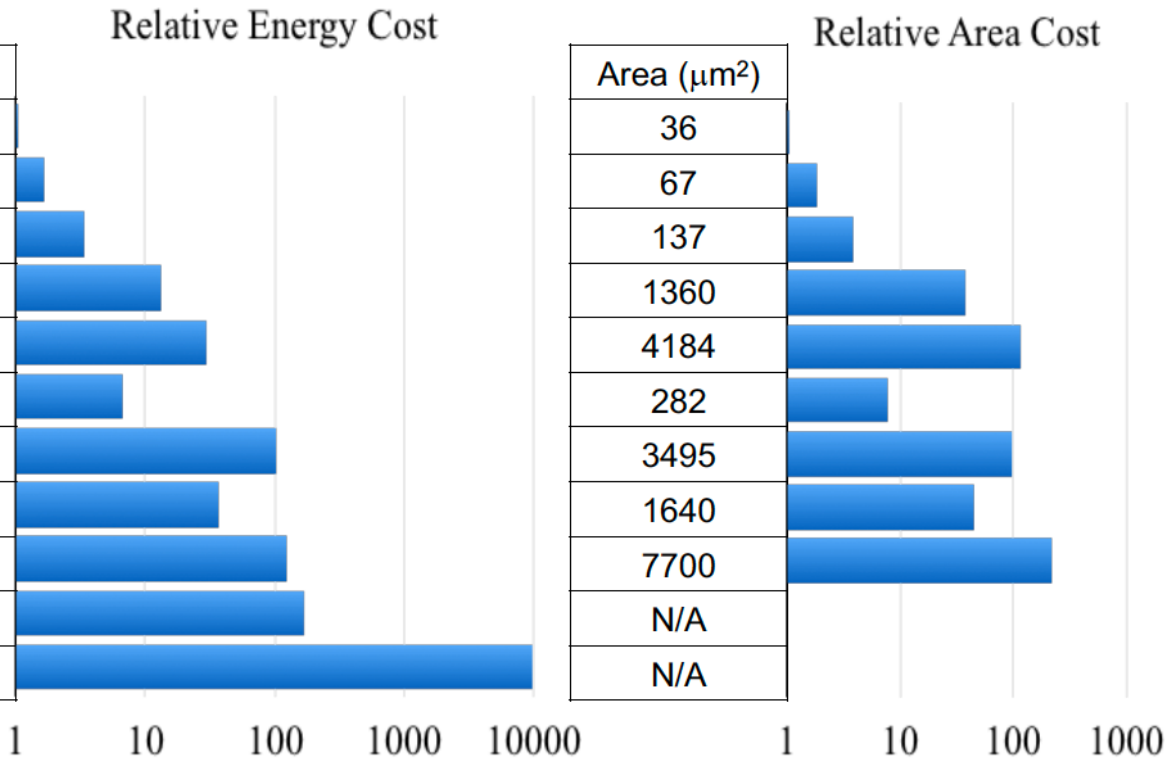
0 - 448

0 - 57344

Slide credit: Bill Dally

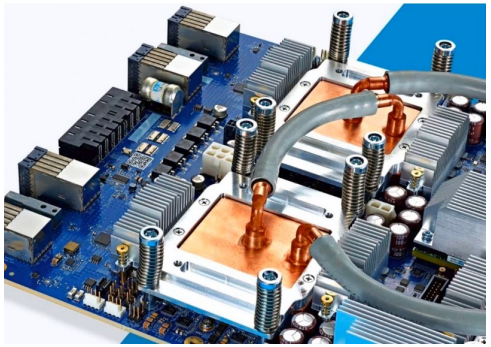
Energy and Area Cost of Compute

Operation:	Energy (pJ)	Area (μm²)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A



Energy numbers are from Mark Horowitz "Computing's Energy Problem (and what we can do about it)", ISSCC 2014
Area numbers are from synthesized result using Design Compiler under TSMC 45nm tech node. FP units used DesignWare Library.

Hardware acceleration of DNN inference/training



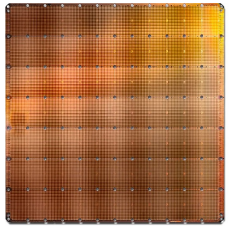
Google TPU3

AWS Trainium 2

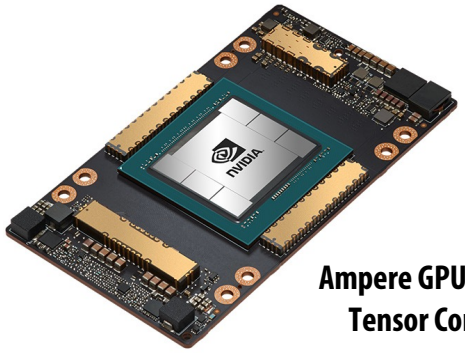
Apple Neural Engine

Intel Deep Learning
Inference Accelerator

SambaNova
Cardinal SN10



Cerebras Wafer Scale Engine



Ampere GPU with
Tensor Cores

Investment in AI hardware

SambaNova Systems Raises $676M in Series D, Surpasses $5B Valuation and Becomes World's Best-Funded AI Startup

SoftBank Vision Fund 2 leads round backing breakthrough platform that delivers unprecedented AI capability and accessibility to customers worldwide

April 13, 2021 09:00 AM Eastern Daylight Time

PALO ALTO, Calif.--(BUSINESS WIRE)--SambaNova Systems, the company building the industry's most advanced software, hardware and services to run AI applications, today announced a $676 million Series D funding round led by SoftBank Vision Fund 2*. The round includes additional new investors Temasek and GIC, plus existing backers including funds and accounts managed by BlackRock, Intel Capital, GV (formerly Google V

"We're here to revolutionize the AI market, and this round greatly accelerates that mission"

Tweet this

"We're here to revolutionize the AI market, and this round gre
founder and CEO. "Traditional CPU and GPU architectures hi
to solve humanity's greatest technology challenges, a new ap
to see a wealth of prudent investors validate that."

SambaNova's flagship offering is Dataflow-as-a-Service (Daa
to jump-start enterprise-level AI initiatives, augmenting organ
centers, allowing the organization to focus on its business objectives instead of infrastructure.

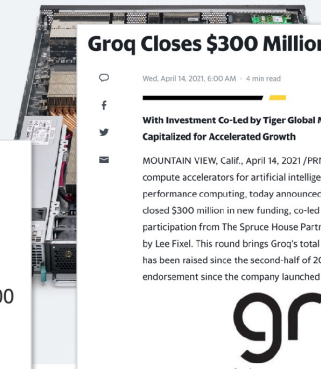
Artificial intelligence chip startup Cerebras Systems claims it has the "world's fastest AI supercomputer," thanks to its large Wafer Scale Engine processor that comes with 400,000 compute cores.

The Los Altos, Calif.-based startup introduced its CS-1 system at the Supercomputing conference in Denver last week after raising more than $200 million in funding from investors, most recently with an $88 million Series D round that was raised in November 2018, according to Andrew Feldman, the founder and CEO of Cerebras who was previously an executive at AMD.

AI chipmaker Graphcore raises $222M at a $2.77B valuation and puts an IPO in its sights

Ingrid Lunden @ingridlunden / 10:59 PM PST • December 28, 2020

Comment



Groq Closes $300 Million Fundraise

Wed, April 14, 2021, 6:00 AM · 4 min read

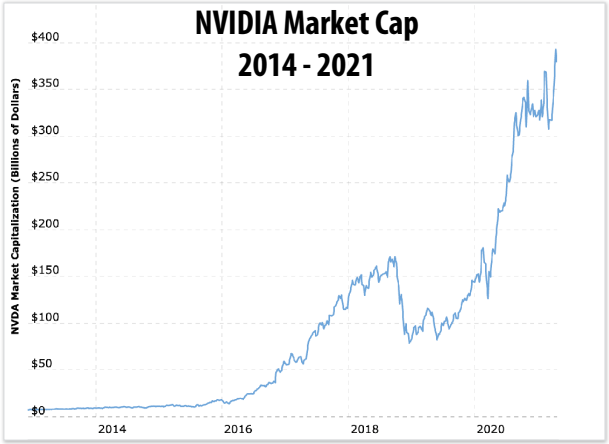
With Investment Co-Led by Tiger Global Management and D1 Capital, Groq Is Well Capitalized for Accelerated Growth

MOUNTAIN VIEW, Calif., April 14, 2021 /PRNewswire/ -- Groq Inc., a leading innovator in compute accelerators for artificial intelligence (AI), machine learning (ML) and high performance computing, today announced that it has closed its Series C fundraising. Groq closed $300 million in new funding, co-led by Tiger Global Management and D1 Capital, with participation from The Spruce House Partnership and Addition, the venture firm founded by Lee Fixel. This round brings Groq's total funding to $367 million, of which $300 million has been raised since the second-half of 2020, a direct result of strong customer endorsement since the company launched its first product.

groq™

Groq logo

Applications based on artificial intelligence — whether they are systems running autonomous services, platforms being used in drug development or to predict the spread of a virus, traffic management for 5G networks or something else altogether — require an unprecedented amount of computing power to run. And today, one of the big names in the world of designing and



Intel Acquires Artificial Intelligence Chipmaker Habana Labs

Combination Advances Intel's AI Strategy, Strengthens Portfolio of AI Accelerators for the Data Center

SANTA CLARA Calif., Dec. 16, 2019 – Intel Corporation today announced that it has acquired Habana Labs, an Israel-based developer of programmable deep learning accelerators for the data center for approximately $2 billion. The combination strengthens Intel's artificial intelligence (AI) portfolio and accelerates its efforts in the nascent, fast-growing AI silicon market, which Intel expects to be greater than $25 billion by 2024[1].

"This acquisition advances our AI strategy, which is to provide customers with solutions to fit every performance need – from the intelligent edge to the data center," said Navin Shenoy, executive vice president and general manager of the Data Platforms Group at Intel. "More specifically, Habana turbo-charges our AI offerings for the data center with a high-performance training processor family and a standards-based programming environment to address evolving AI workloads."

Google's TPU (v1)

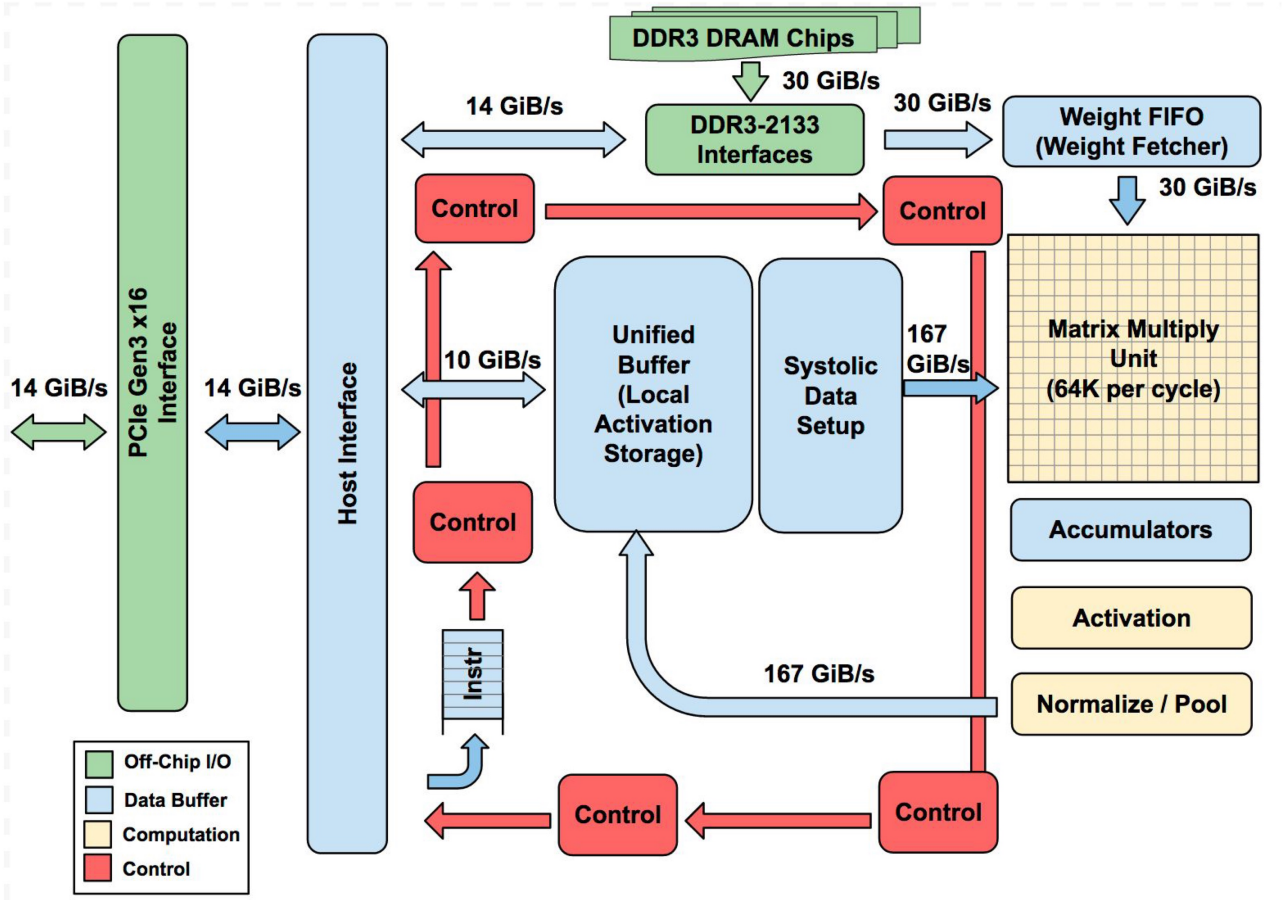
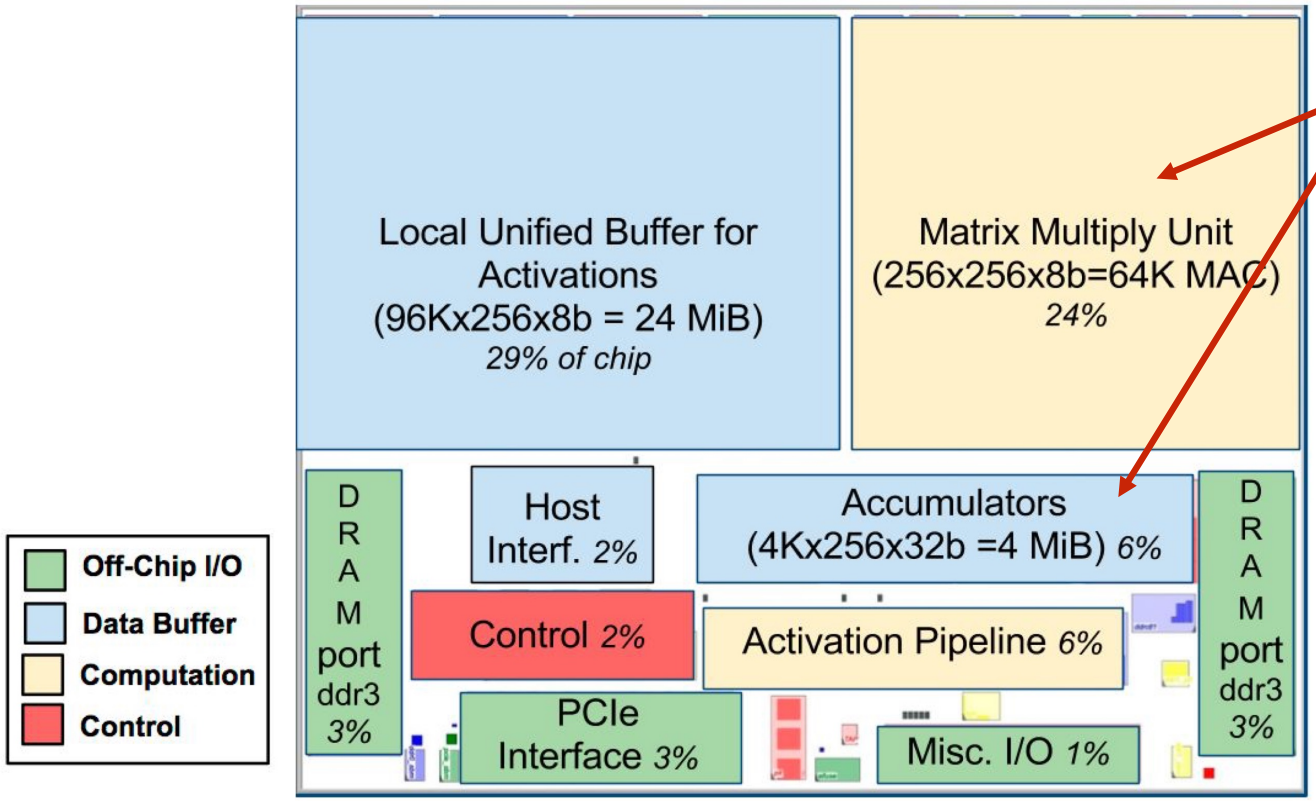


Figure credit: Jouppi et al. 2017

TPU area proportionality



Arithmetic units ~ 30% of chip
Note low area footprint of control

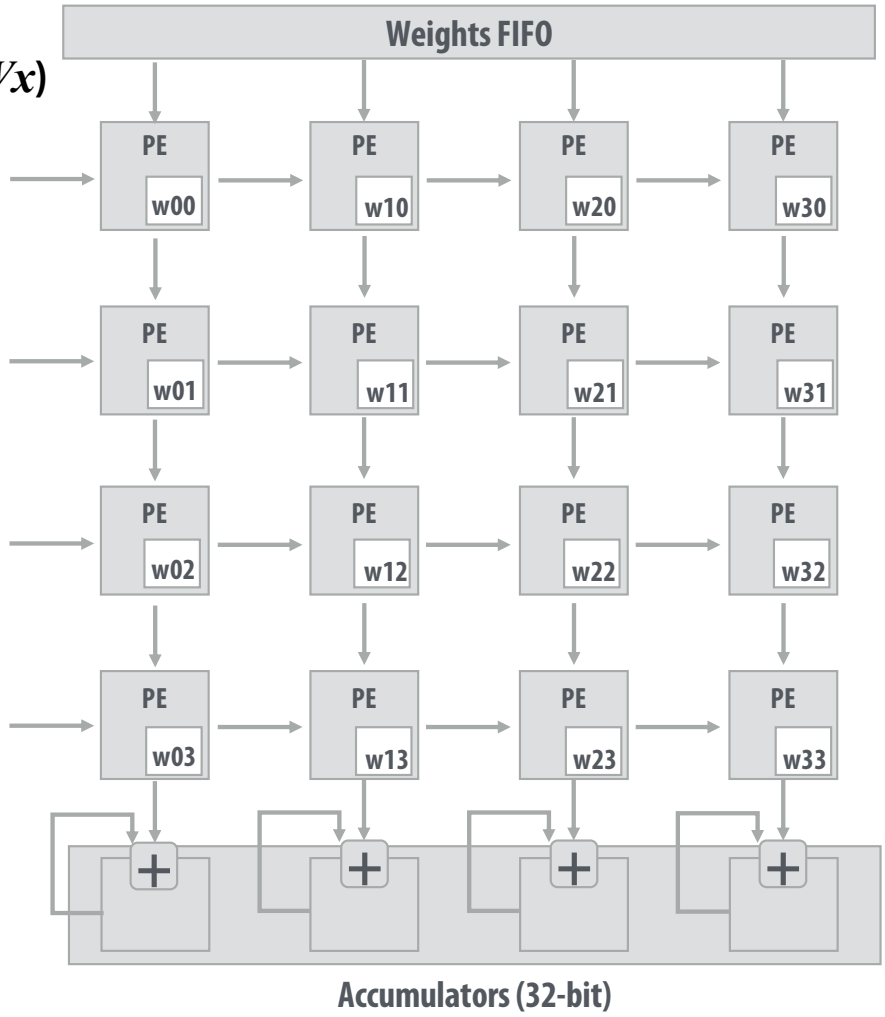
Key instructions:

- **read host memory**
- **write host memory**
- **read weights**
- **matrix_multiply / convolve**
- **activate**

Figure credit: Jouppi et al. 2017

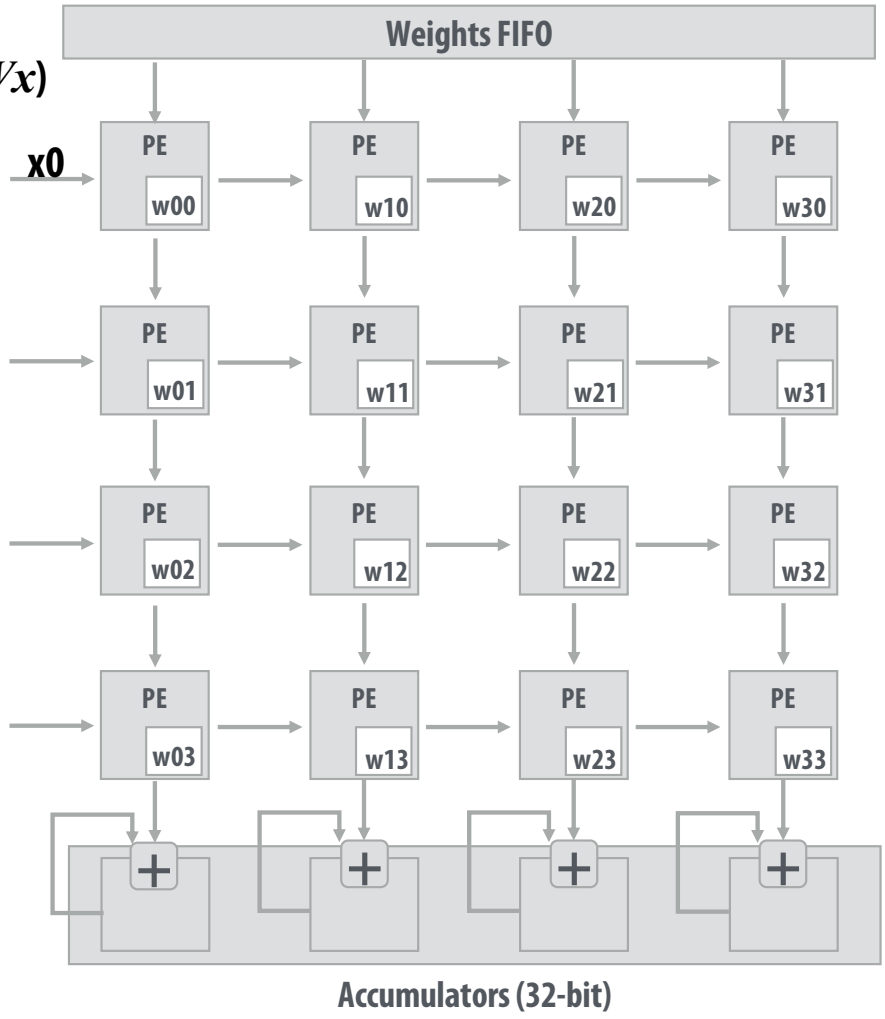
Systolic array

(matrix vector multiplication example: $y=Wx$)



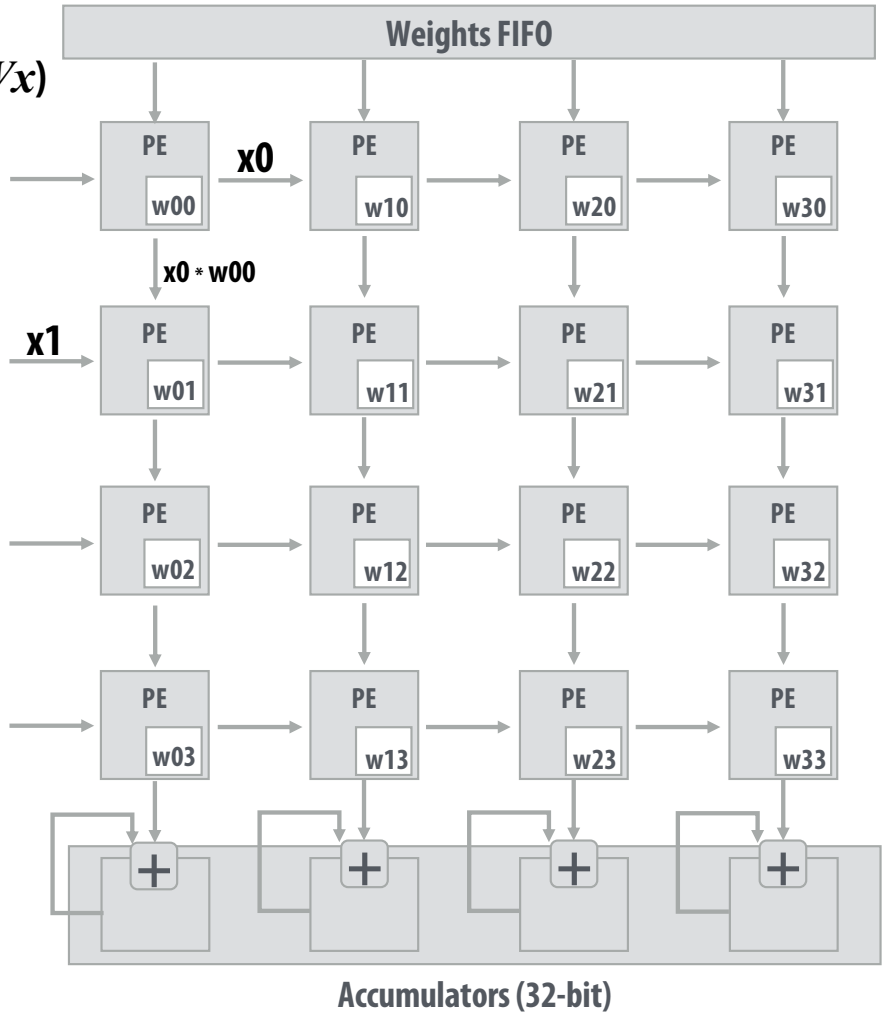
Systolic array

(matrix vector multiplication example: $y=Wx$)



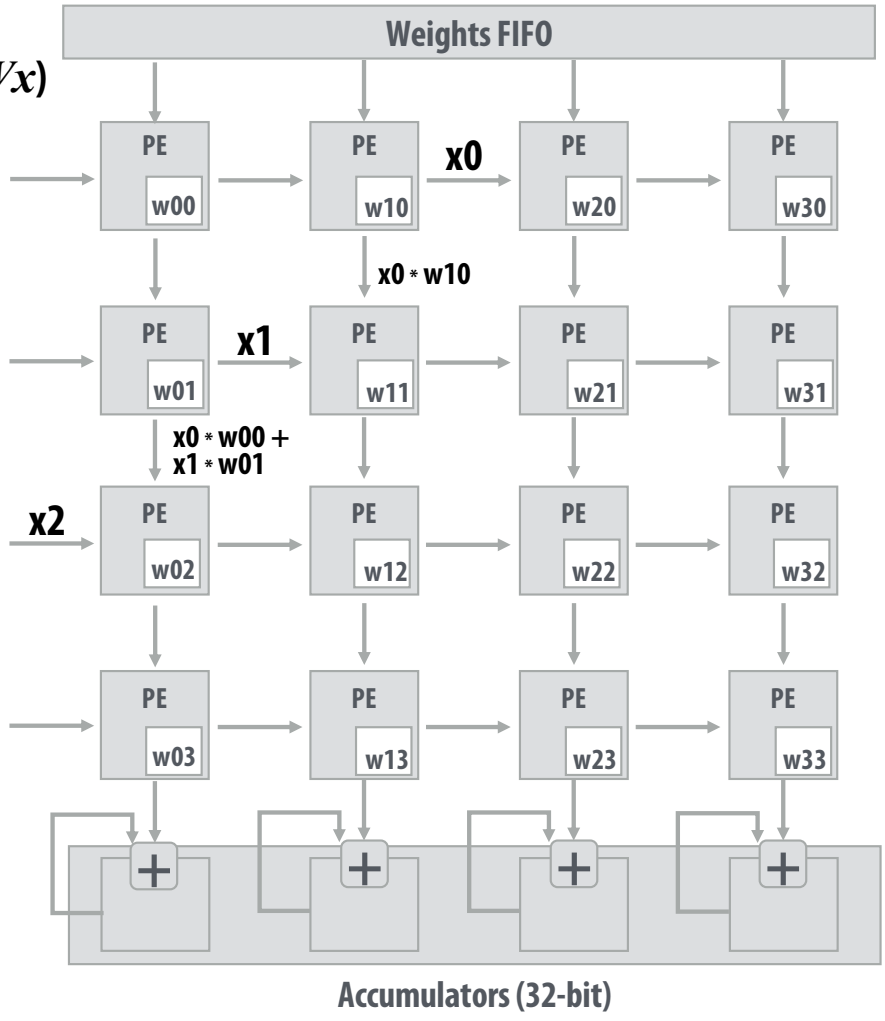
Systolic array

(matrix vector multiplication example: $y=Wx$)



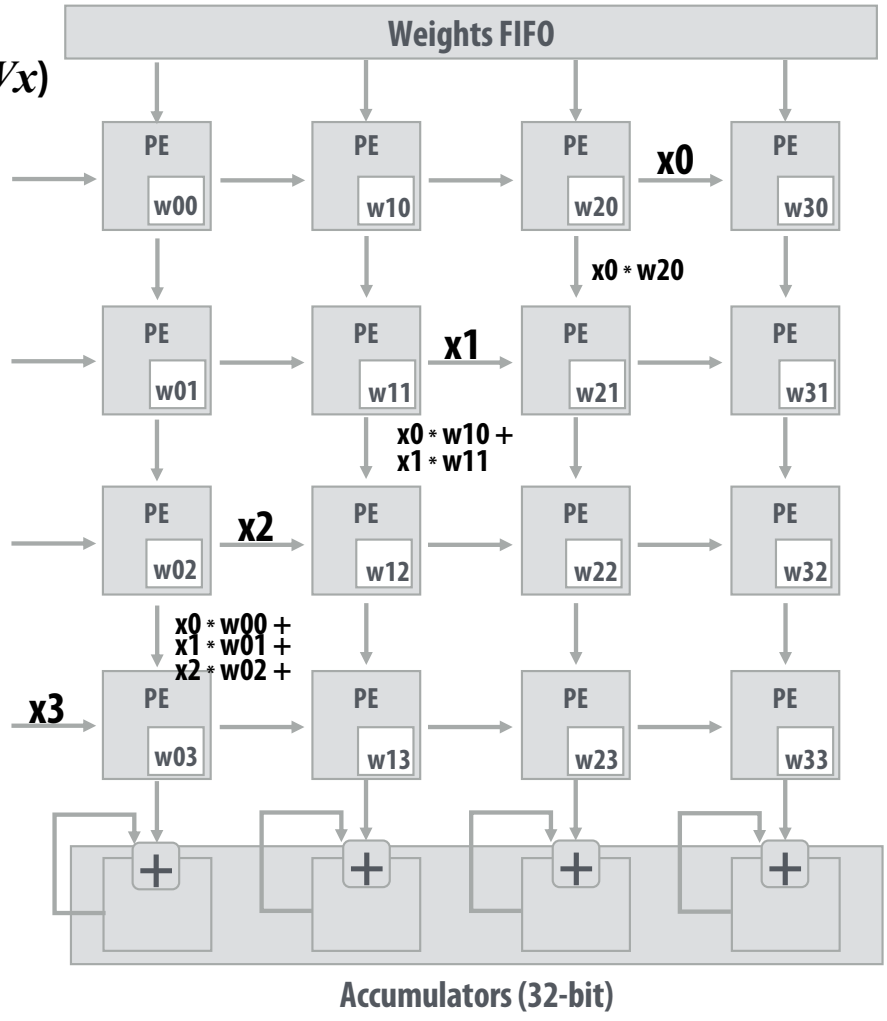
Systolic array

(matrix vector multiplication example: $y=Wx$)



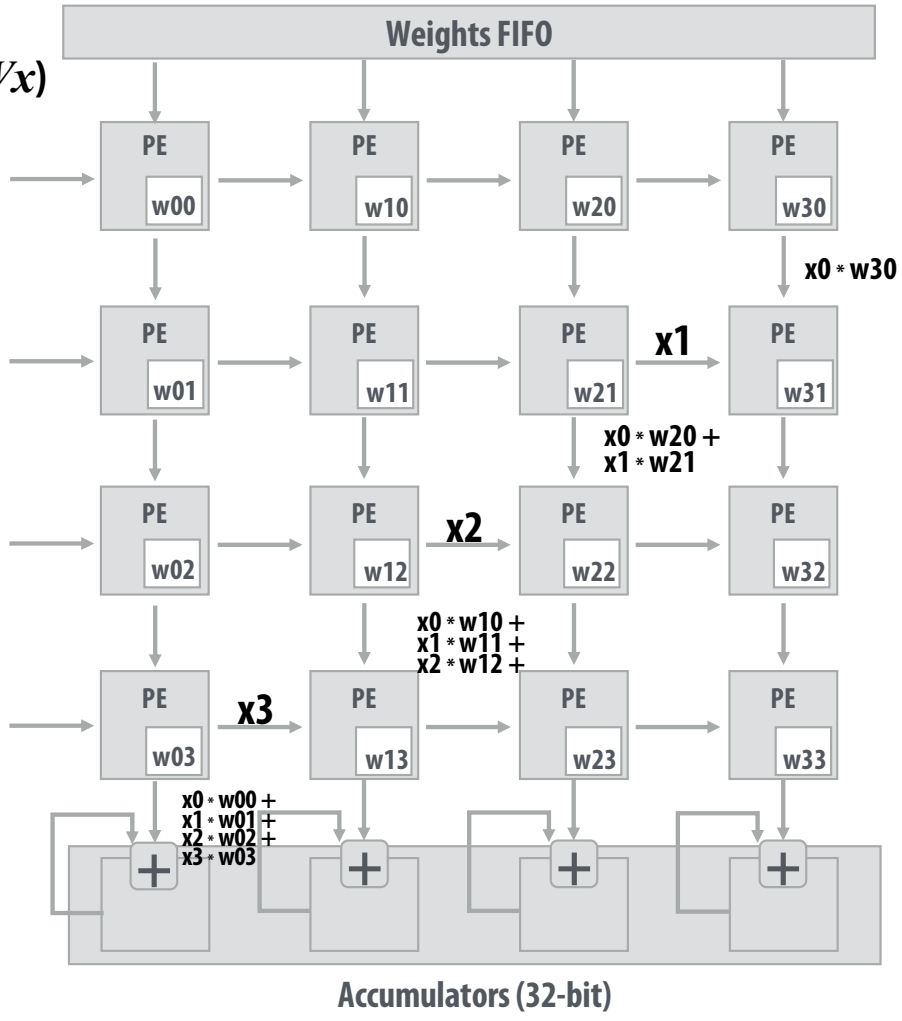
Systolic array

(matrix vector multiplication example: $y=Wx$)



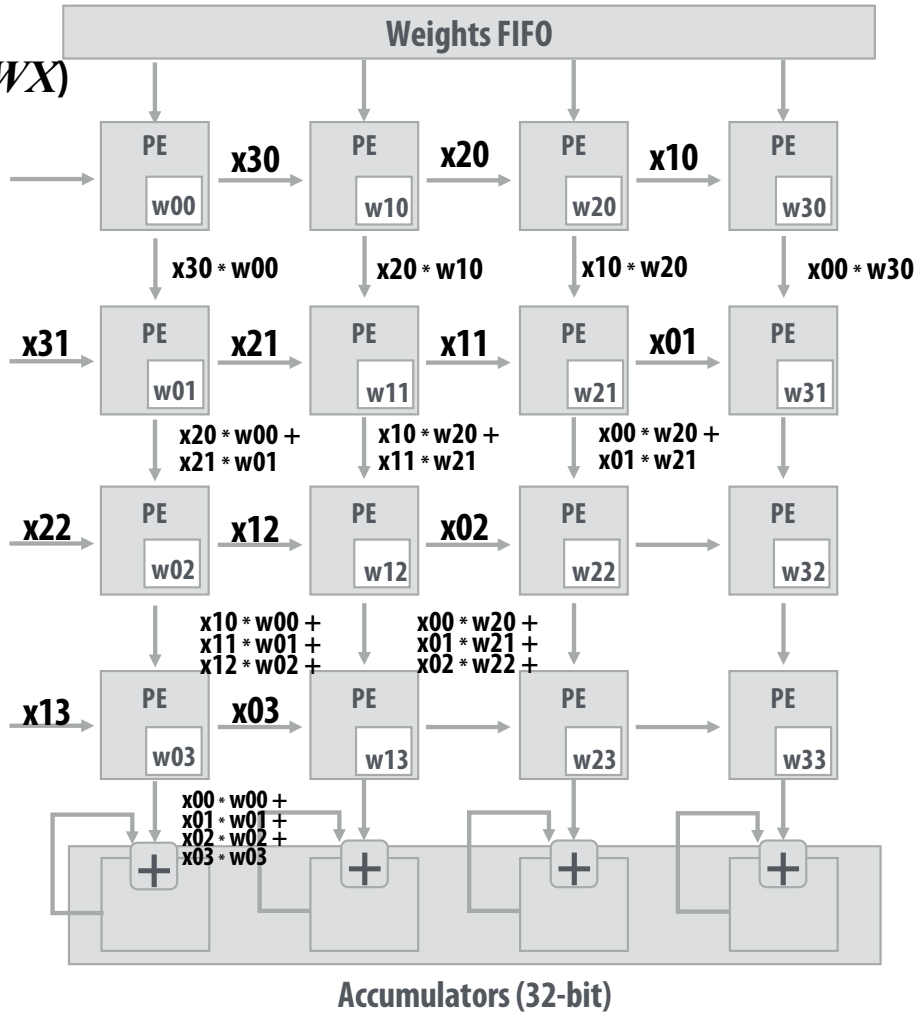
Systolic array

(matrix vector multiplication example: $y=Wx$)



Systolic array

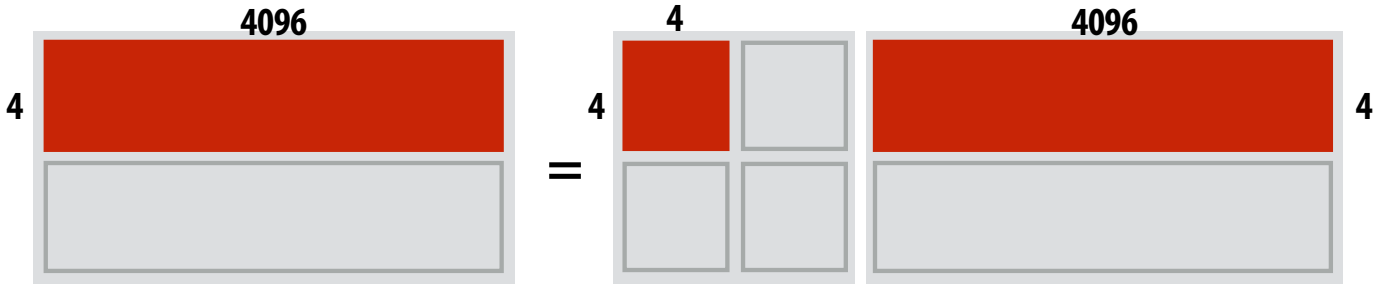
(matrix matrix multiplication example: $Y=WX$)



Notice: need multiple 4x32bit accumulators to hold output columns

Building larger matrix-matrix multiplies

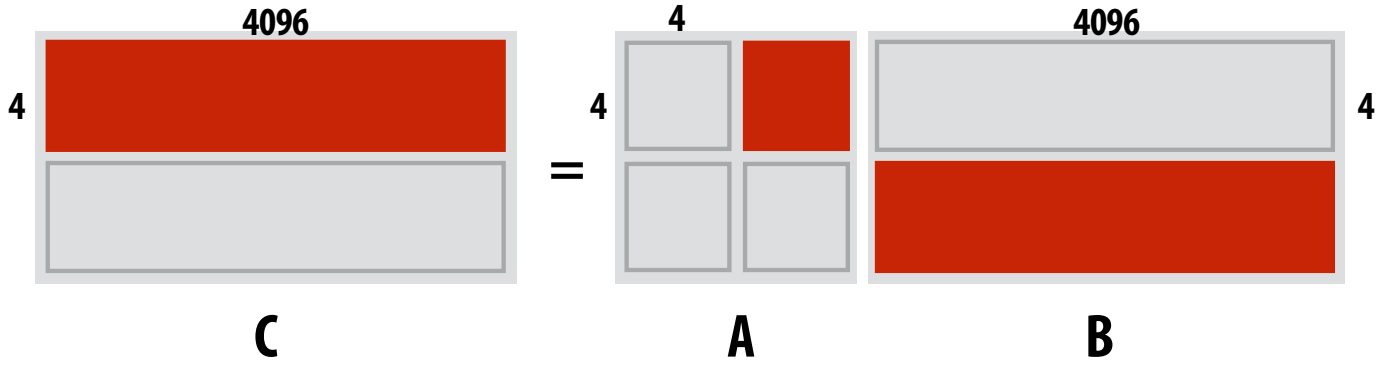
Example: A = 8x8, B= 8x4096, C=8x4096



Assume 4096 accumulators

Building larger matrix-matrix multiplies

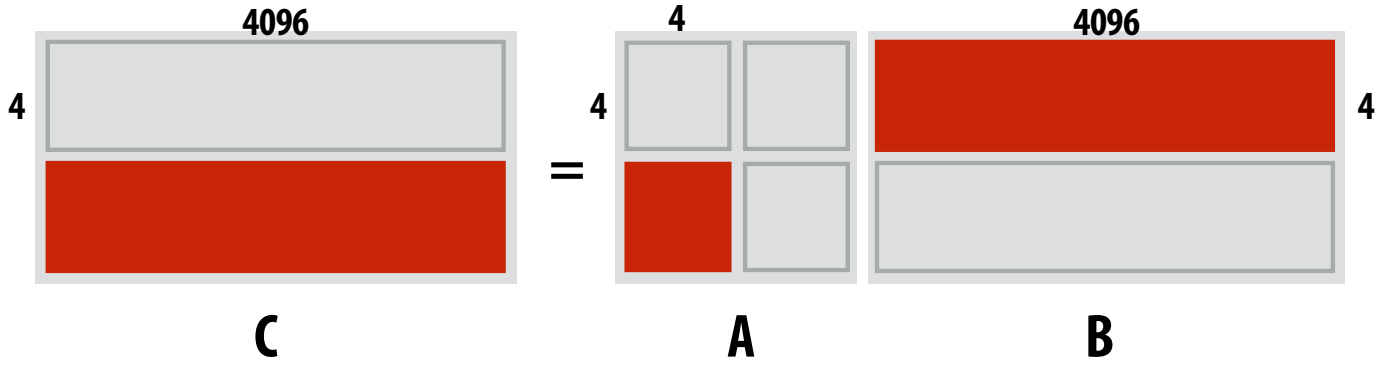
Example: A = 8x8, B= 8x4096, C=8x4096



Assume 4096 accumulators

Building larger matrix-matrix multiplies

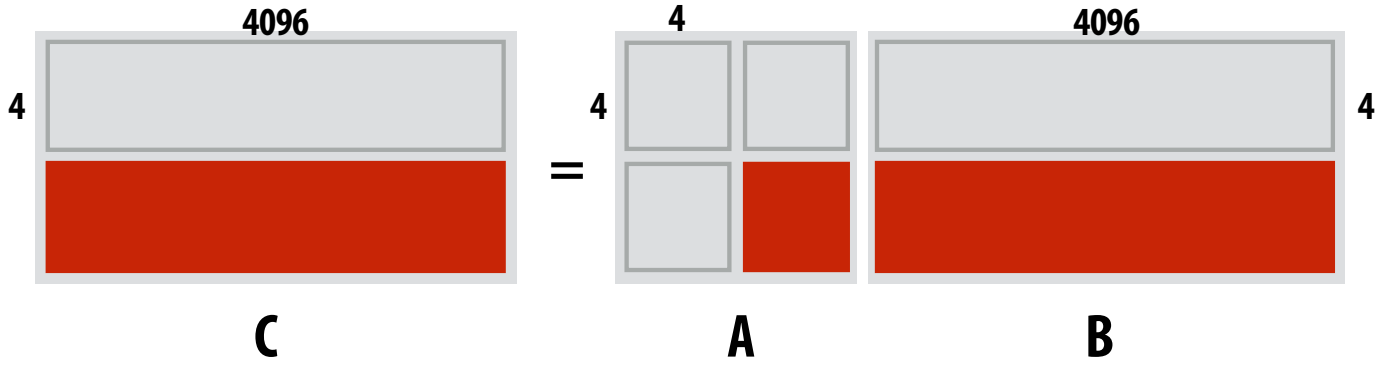
Example: A = 8x8, B= 8x4096, C=8x4096



Assume 4096 accumulators

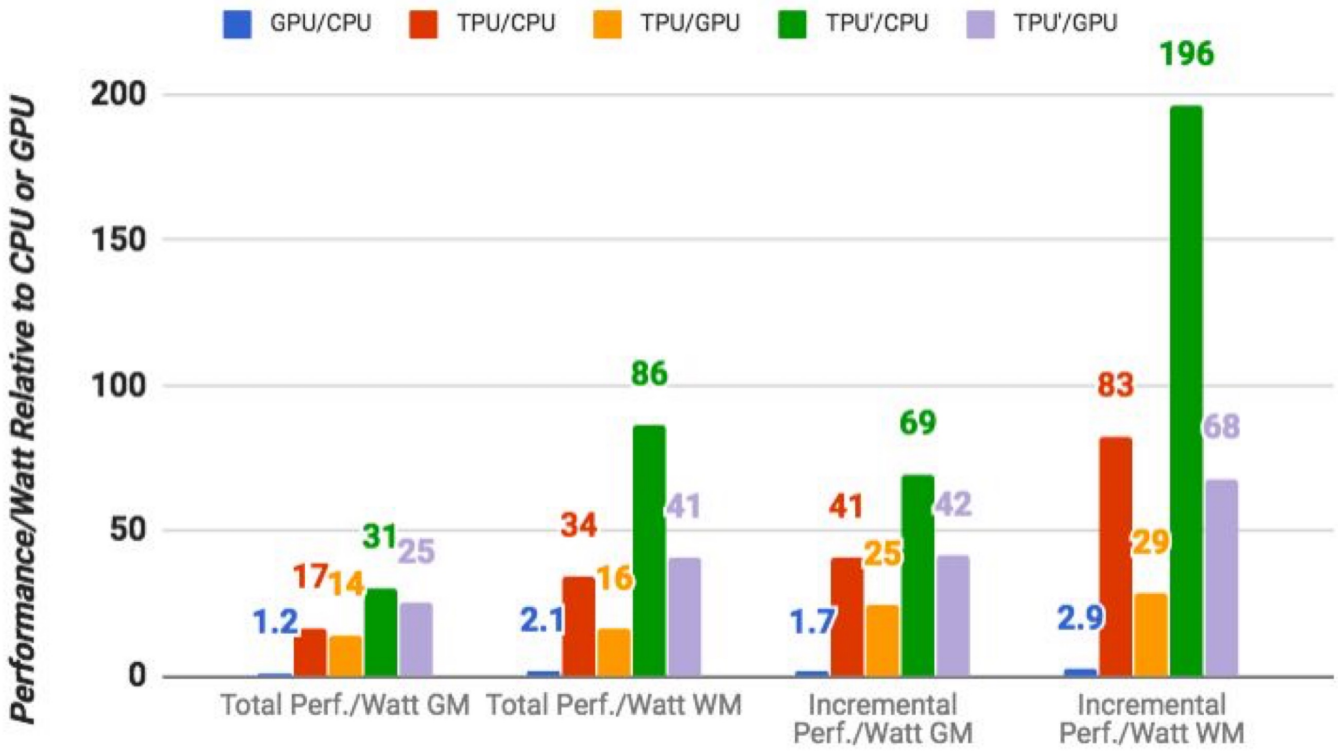
Building larger matrix-matrix multiplies

Example: A = 8x8, B= 8x4096, C=8x4096



Assume 4096 accumulators

TPU Performance/Watt



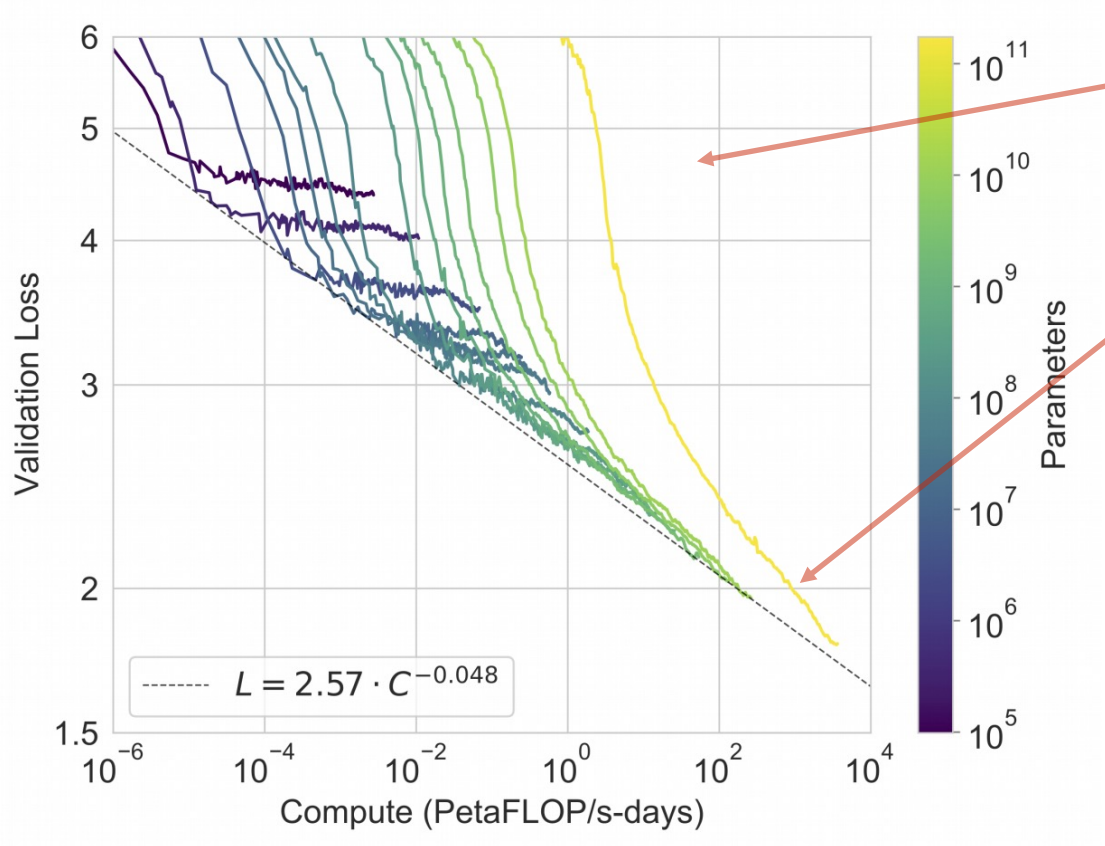
GM = geometric mean over all apps
WM = weighted mean over all apps

total = cost of host machine + CPU
incremental = only cost of TPU

Figure credit: Jouppi et al. 2017

Scaling up (for training big models)

Example: GPT-3 language model

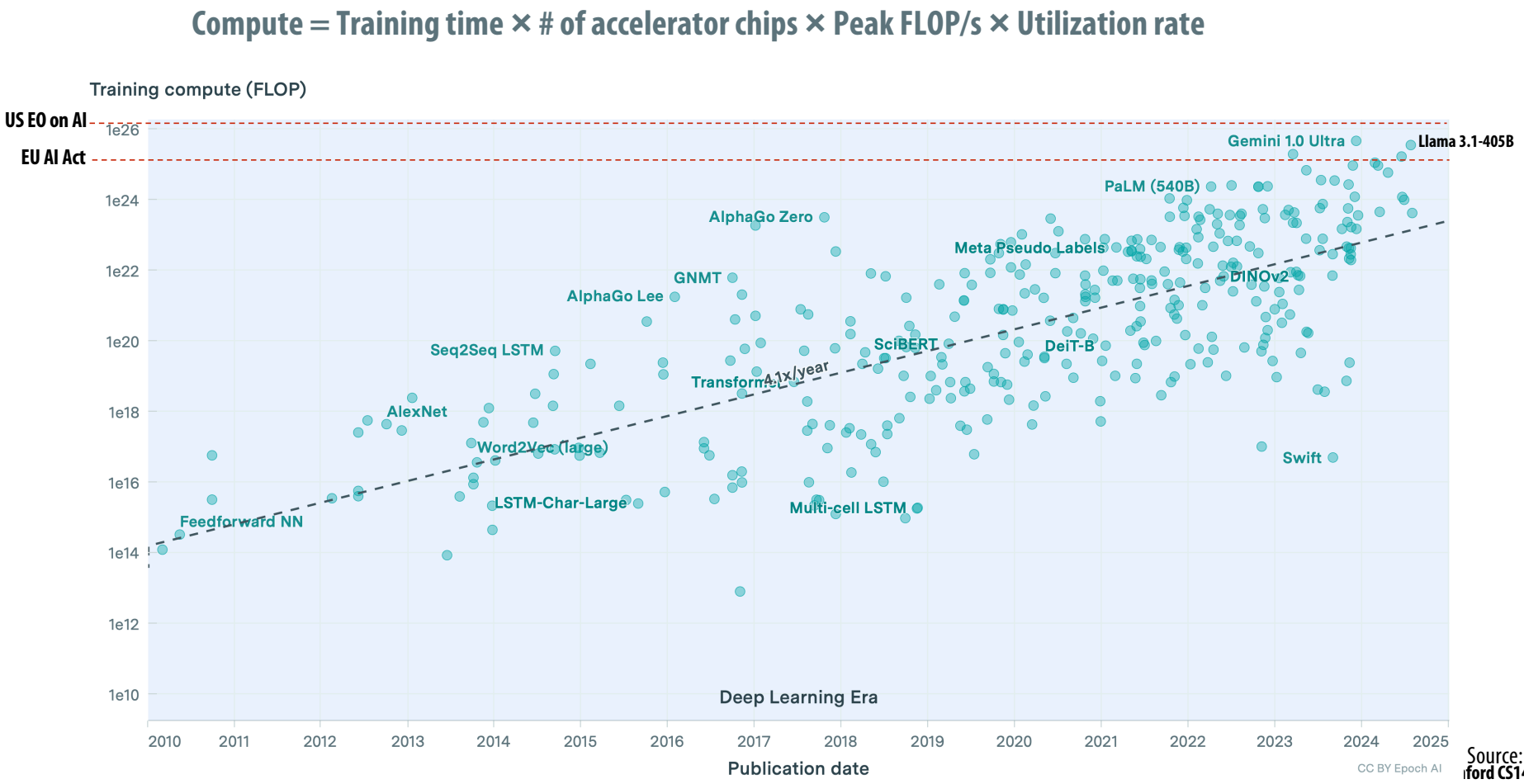


(Amount of training — note this is log scale)

Very big models +
More training
=
Better accuracy

Power law effect:
exponentially more compute to take constant step in accuracy

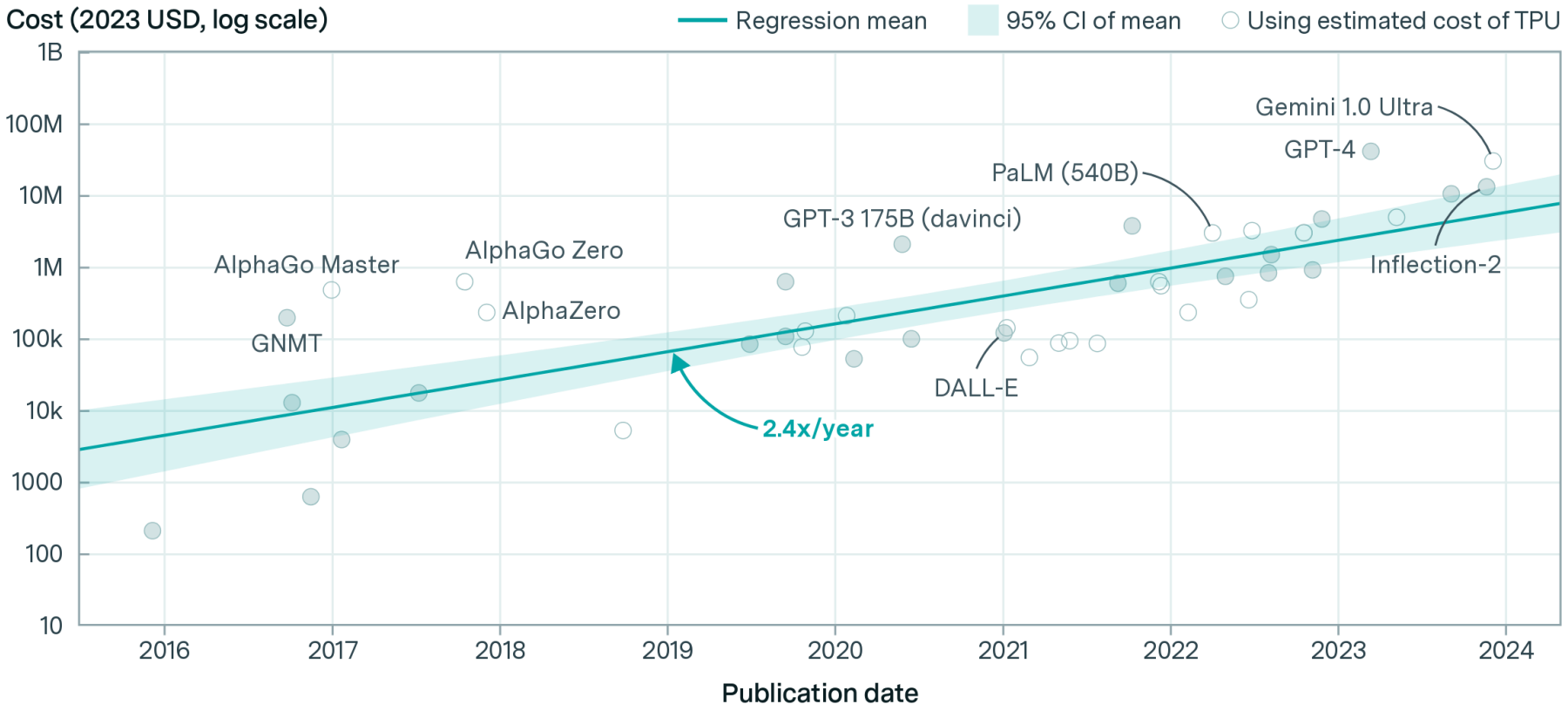
Large Model Training Compute



Source: EPOCH AI

ford CS149, Fall 2024

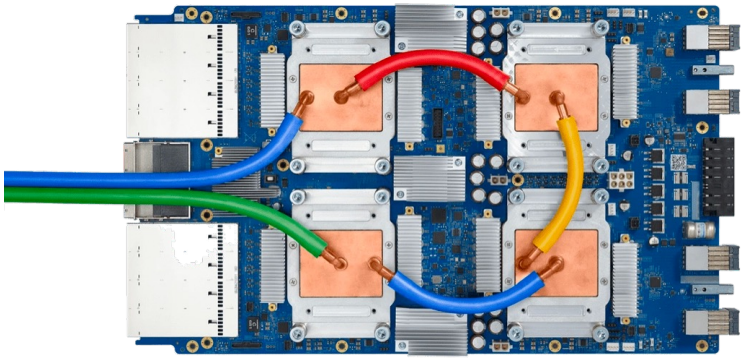
Hardware and Energy Costs of Training



Source: EPOCH AI

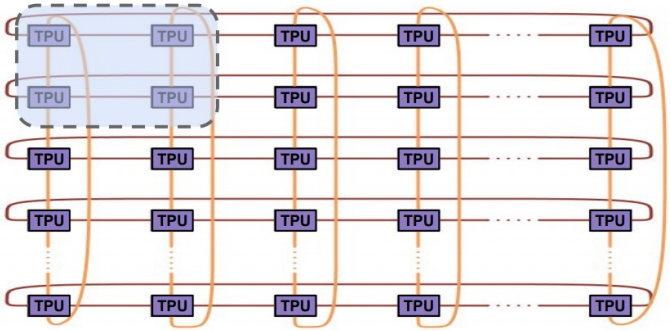
TPU v3 supercomputer

TPU v3 board
4 TPU3 chips

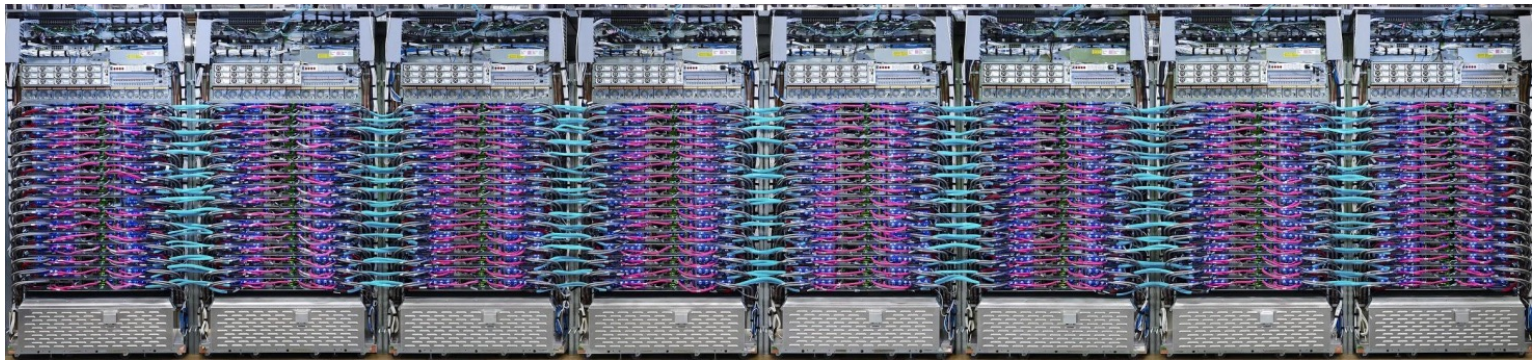


One TPU v3 board

TPUs connected by
2D Torus interconnect

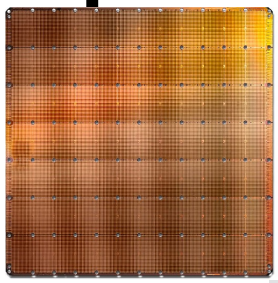


TPU supercomputer (1024 TPU v3 chips)

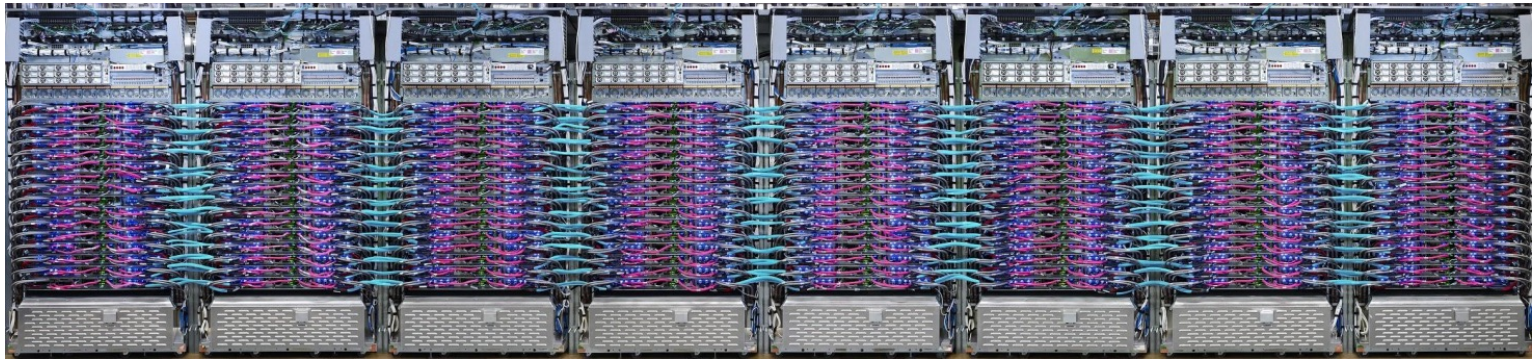


Summary: specialized hardware for DNN processing

- Specialized hardware for executing key DNN computations efficiently
- Feature many arithmetic units
- Customized/configurable datapaths to directly move intermediate data values between processing units (schedule computation by laying it out spatially on the chip)
- Large amounts of on-chip storage for fast access to intermediates



	Cerebras WSE
Chip size	46,225 mm^2
Cores	400,000
On chip memory	18 Gigabytes
Memory bandwidth	9 Petabytes/S
Fabric bandwidth	100 Petabits/S



TPU supercomputer (1024 TPU v3 chips)

Reducing energy consumption idea 1: use specialized processing

(use the right processor for the job)

Reducing energy consumption idea 2: move less data

Data movement has high energy cost

- Rule of thumb in mobile system design: always seek to reduce amount of data transferred from memory
  - Earlier in class we discussed minimizing communication to reduce stalls (poor performance). Now, we wish to reduce communication to reduce energy consumption
- "Ballpark" numbers [Sources: Bill Dally (NVIDIA), Tom Olson (ARM)]
  - Integer op: ~ 1 pJ *
  - Floating point op: ~20 pJ *
  - Reading 64 bits from small local SRAM (1mm away on chip): ~ 26 pJ
  - Reading 64 bits from low power mobile DRAM (LPDDR): ~1200 pJ

Suggests that recomputing values, rather than storing and reloading them, is a better answer when optimizing code for energy efficiency!

- Implications
  - Reading 10 GB/sec from memory: ~1.6 watts
  - Entire power budget for mobile GPU: ~1 watt (remember phone is also running CPU, display, radios, etc.)
  - iPhone 16 battery: ~14 watt-hours (note: my Macbook Pro laptop: 99 watt-hour battery)
  - Exploiting locality matters!!!

* Cost to just perform the logical operation, not counting overhead of instruction decode, load data from registers, etc.

Moving data is costly!

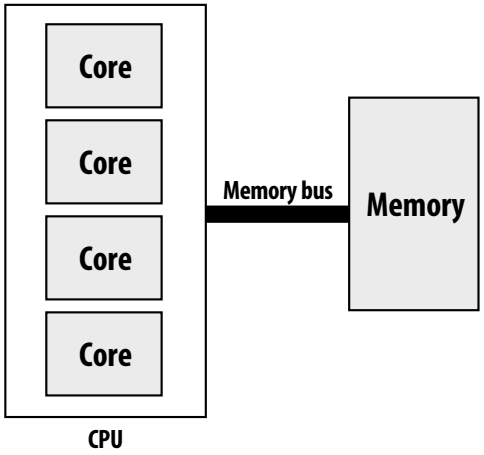
Data movement limits performance

Many processing elements…

= higher overall rate of memory requests

= need for more memory bandwidth

(result: bandwidth-limited execution)

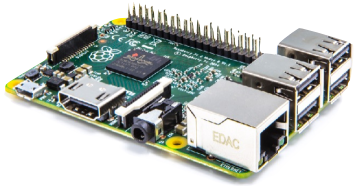


Data movement has high energy cost

~ 0.9 pJ for a 32-bit floating-point math op *

~ 5 pJ for a local SRAM (on chip) data access

~ 640 pJ to load 32 bits from LPDDR memory

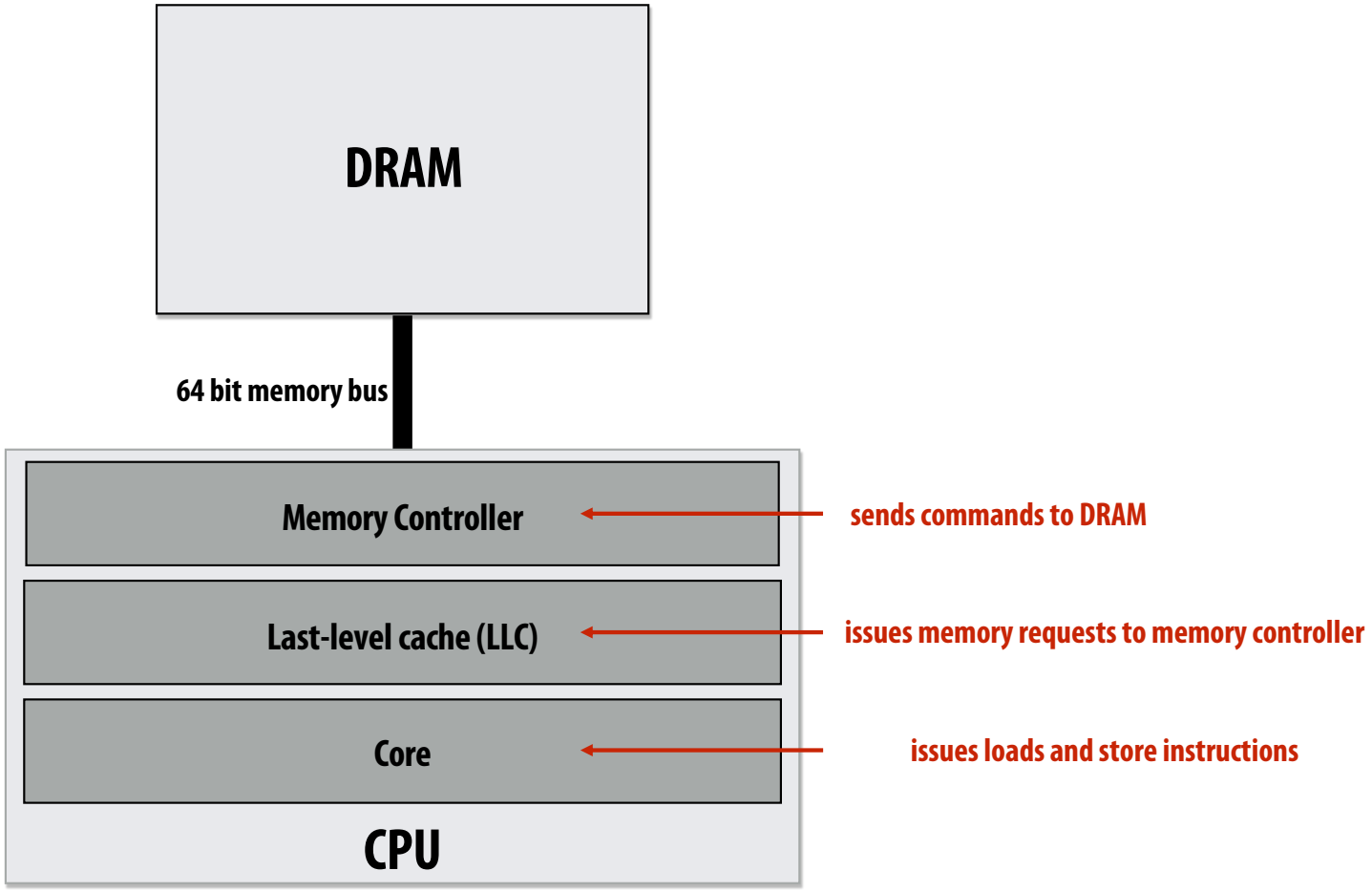


* Source: [Han, ICLR 2016], 45 nm CMOS assumption

Accessing DRAM

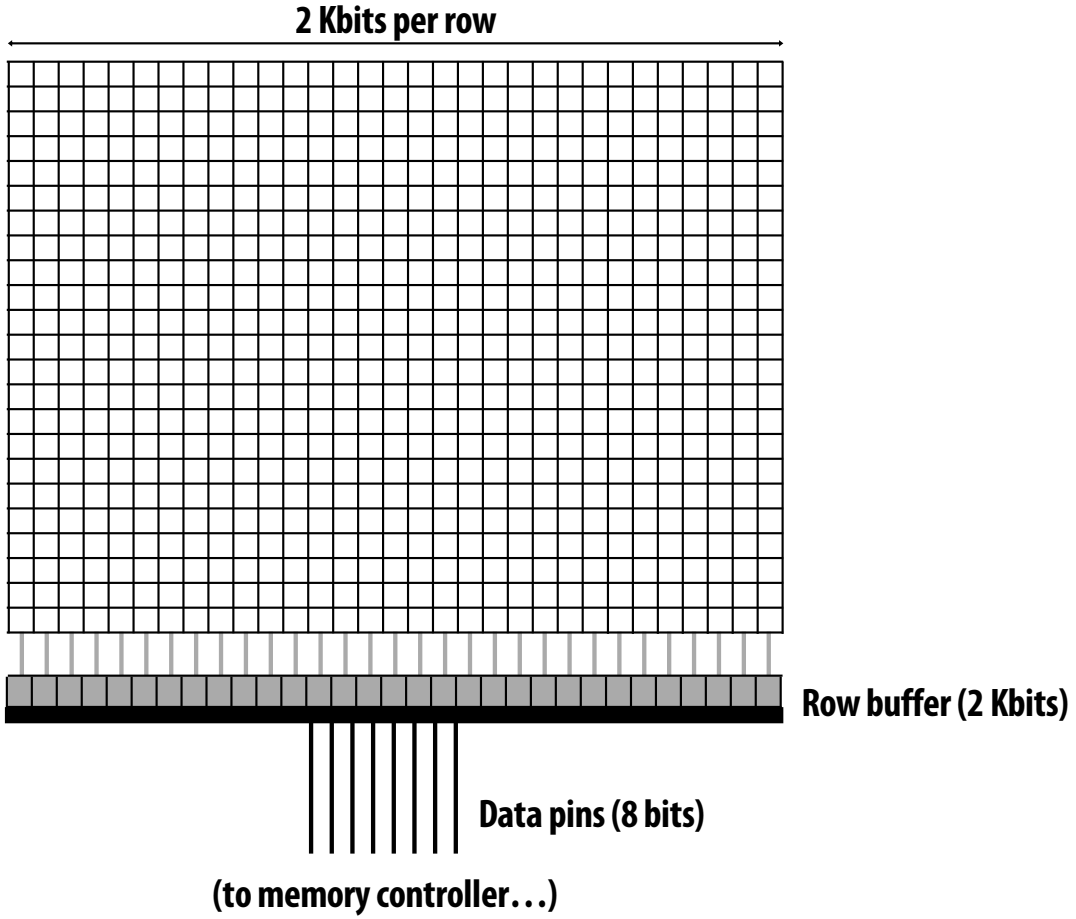
(a basic tutorial on how DRAM works)

The memory system



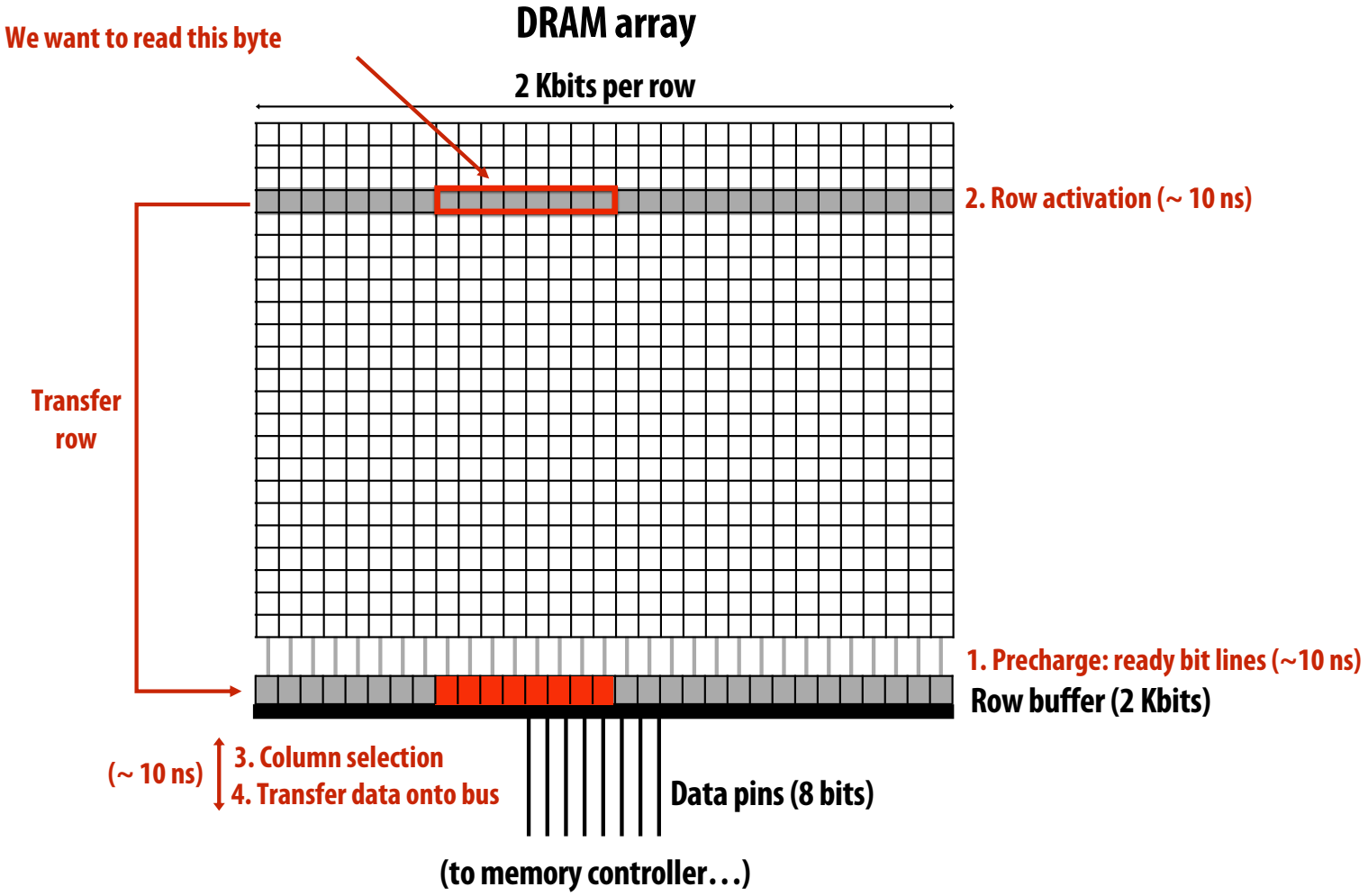
DRAM array

1 transistor + capacitor per "bit" (Recall: a capacitor stores charge)



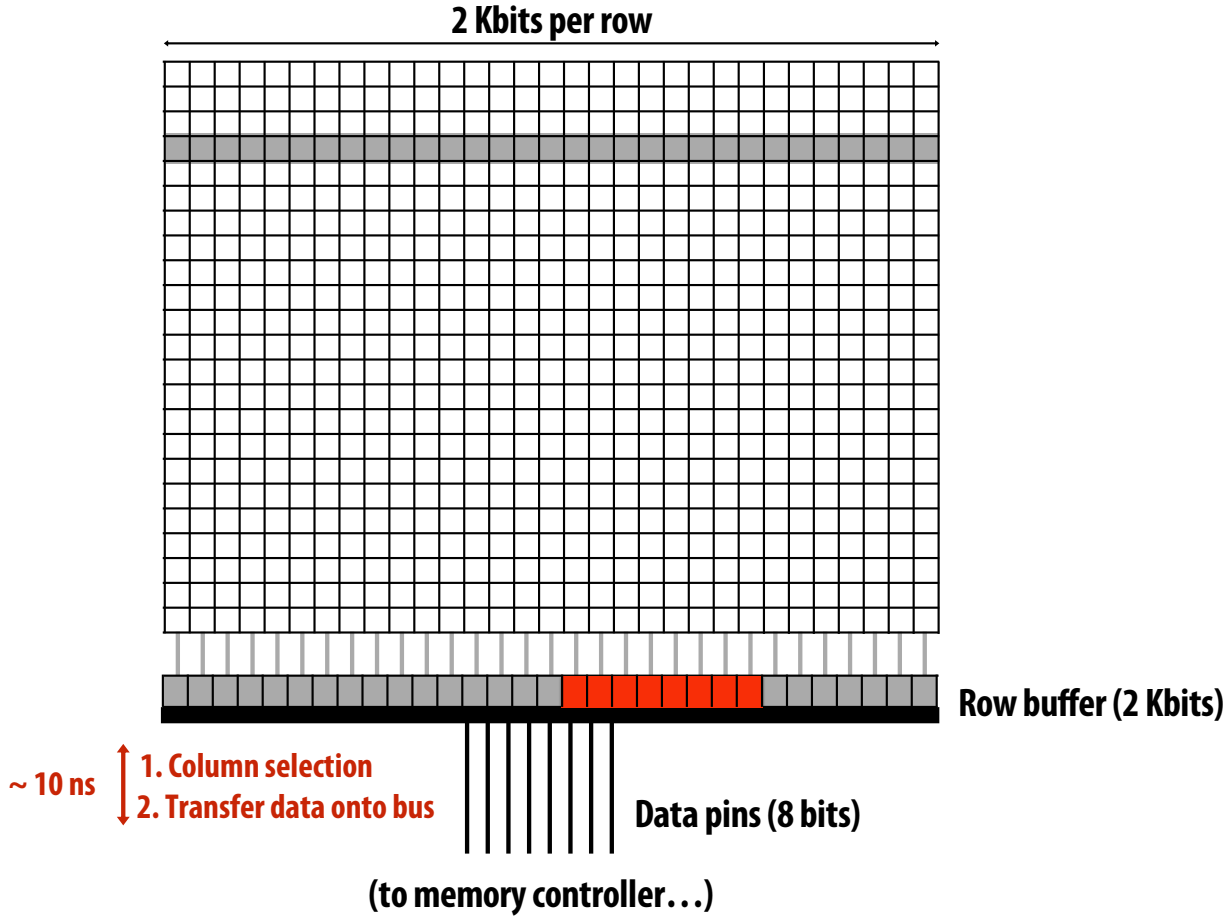
DRAM operation (load one byte)

Estimated latencies are in units of memory clocks: DDR3-1600



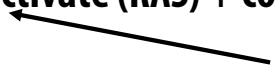
Load next byte from (already active) row

Lower latency operation: can skip precharge and row activation steps



DRAM access latency is not fixed

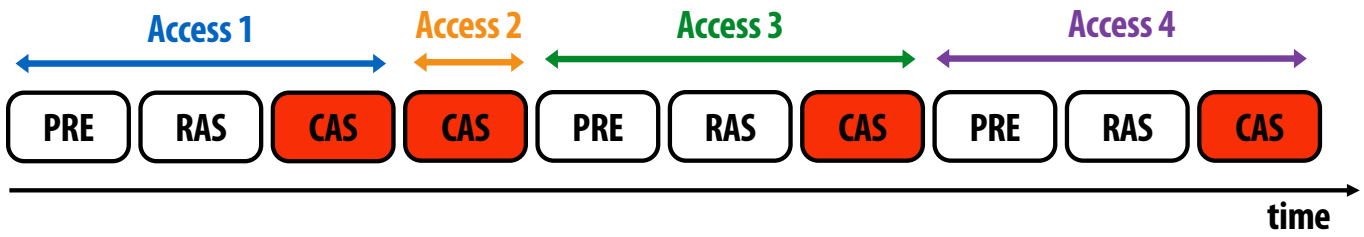
- **Best case latency: read from active row**
  - Column access time (CAS)
- **Worst case latency: bit lines not ready, read from new row**
  - Precharge (PRE) + row activate (RAS) + column access (CAS)



Precharge readies bit lines and writes row buffer contents back into DRAM array (read was destructive)

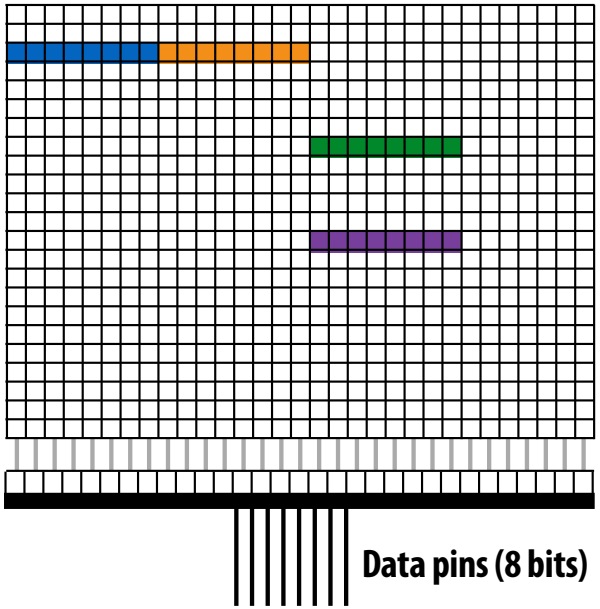
- **Question 1: when to execute precharge?**
  - After each column access?
  - Only when new row is accessed?
- **Question 2: how to handle latency of DRAM access?**

Problem: low pin utilization due to latency of access

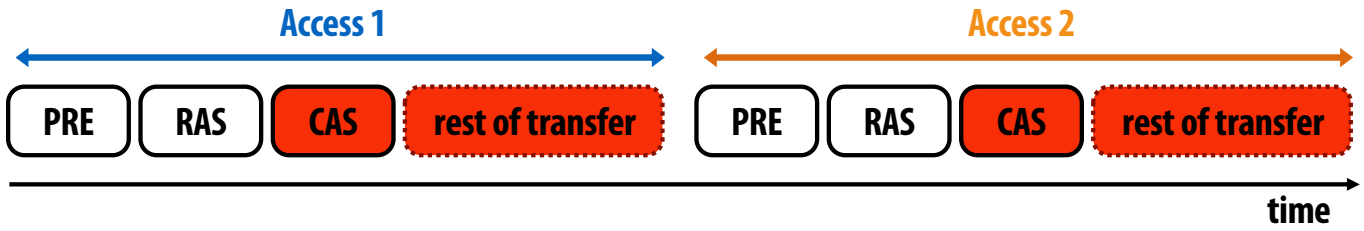


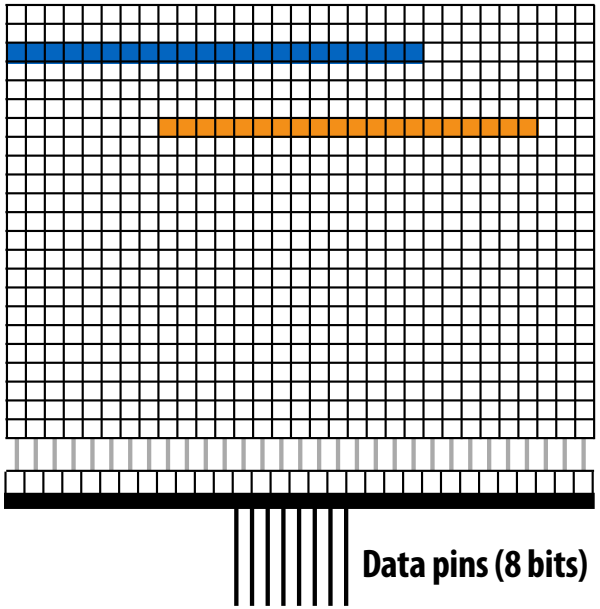
Data pins in use only a small fraction of time
(red = data pins busy)

This is bad since they are the scarcest resource!



DRAM burst mode





Idea: amortize latency over larger transfers

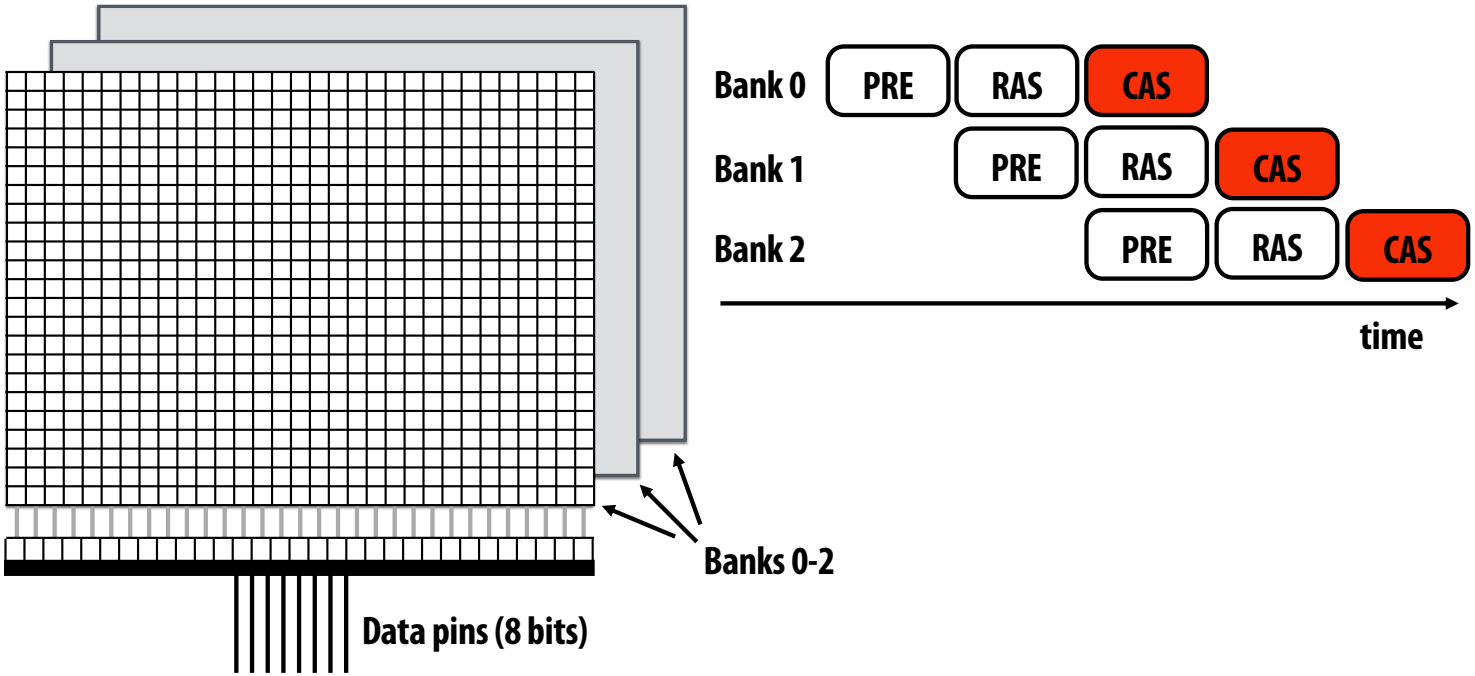
Each DRAM command describes bulk transfer
Bits placed on output pins in consecutive clocks

DRAM chip consists of multiple banks

All banks share same pins (only one transfer at a time)

Banks allow for pipelining of memory requests

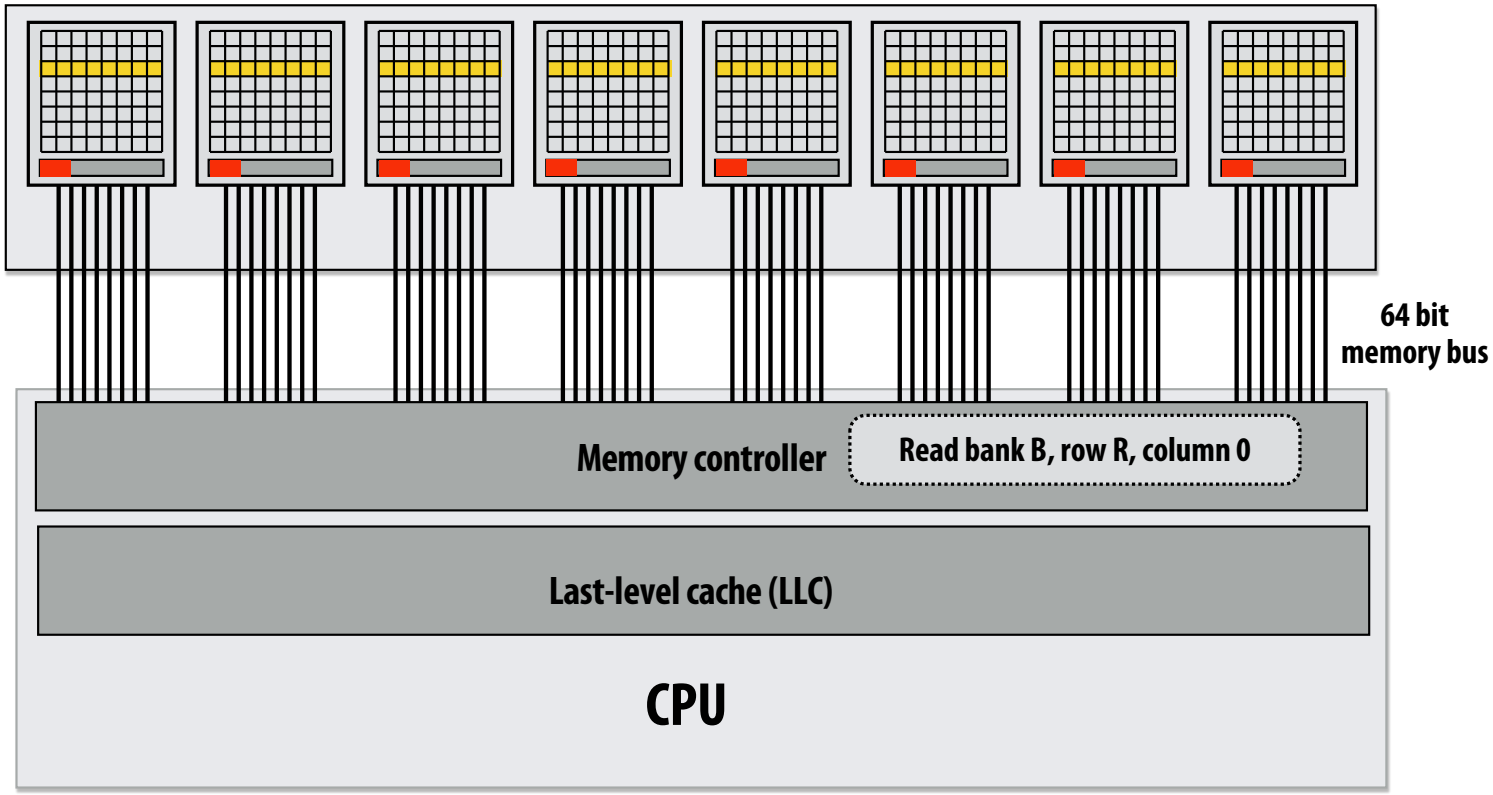
- **Precharge/activate rows/send column address to one bank while transferring data from another**
- **Achieves high data pin utilization**



Organize multiple chips into a DIMM

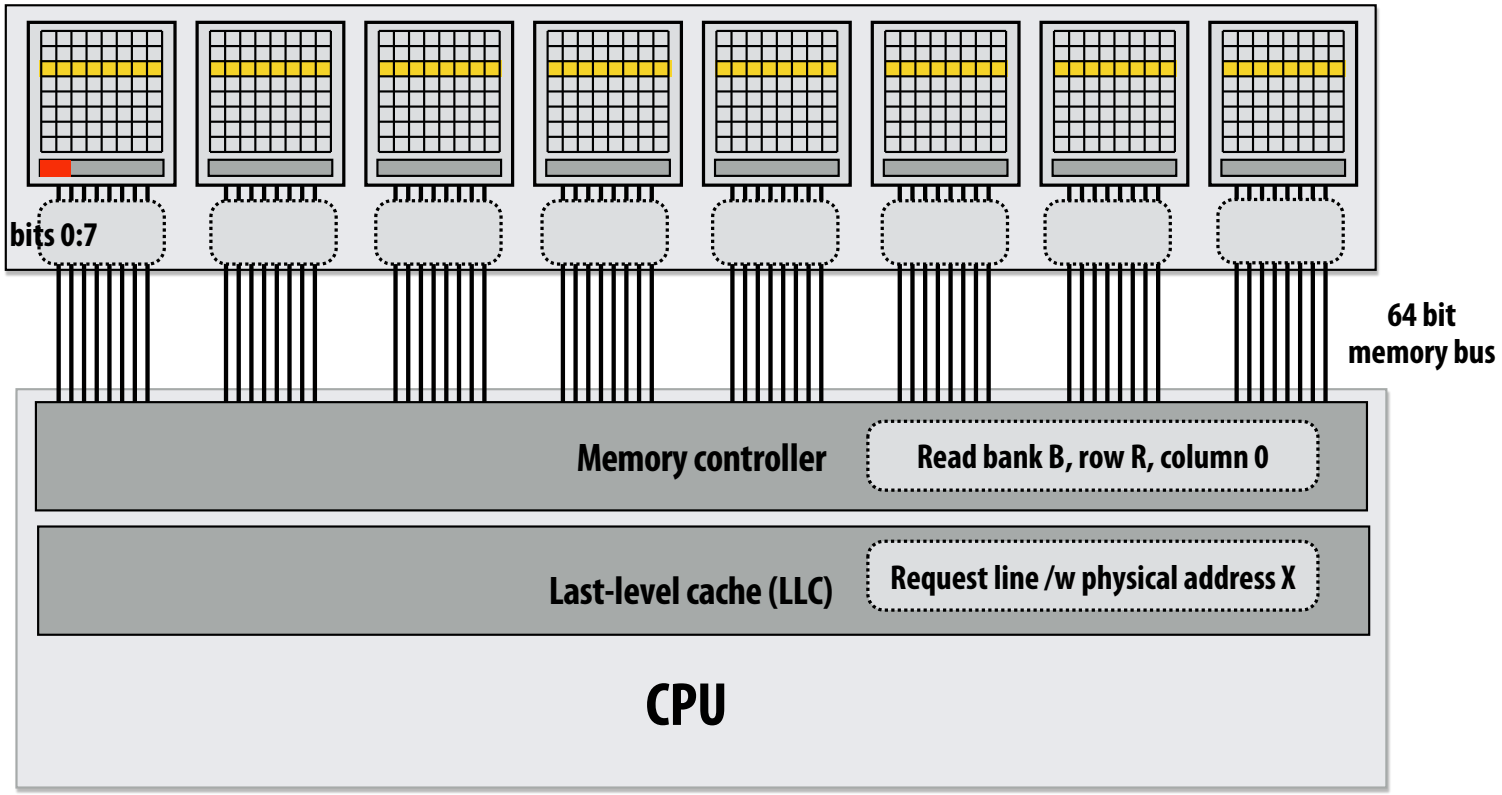
Example: Eight DRAM chips (64-bit memory bus)

Note: DIMM appears as a single, higher capacity, wider interface DRAM module to the memory controller. Higher aggregate bandwidth, but minimum transfer granularity is now 64 bits.



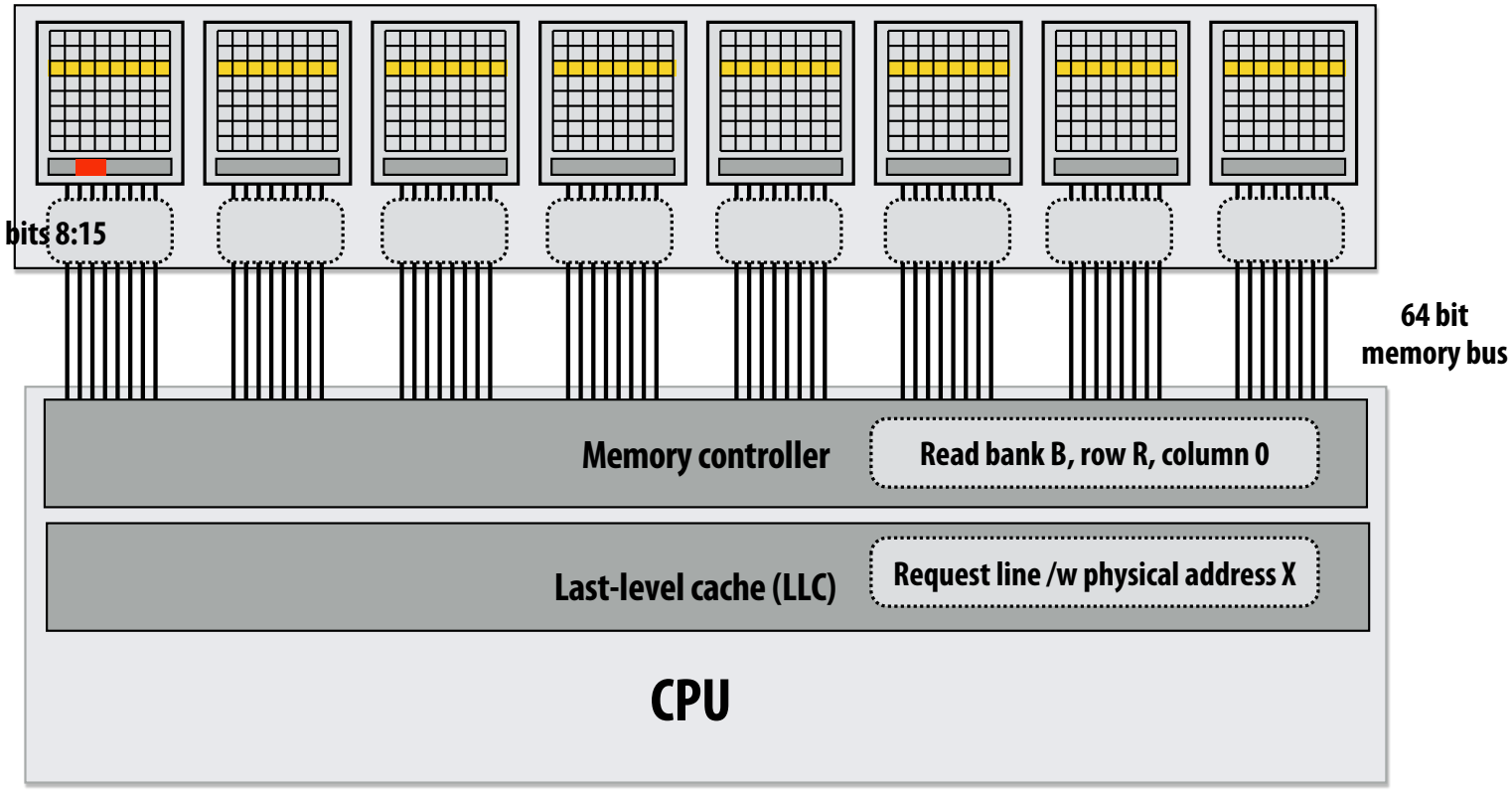
Reading one 64-byte (512 bit) cache line (the wrong way)

Assume: consecutive physical addresses mapped to same row of same chip
Memory controller converts physical address to DRAM bank, row, column



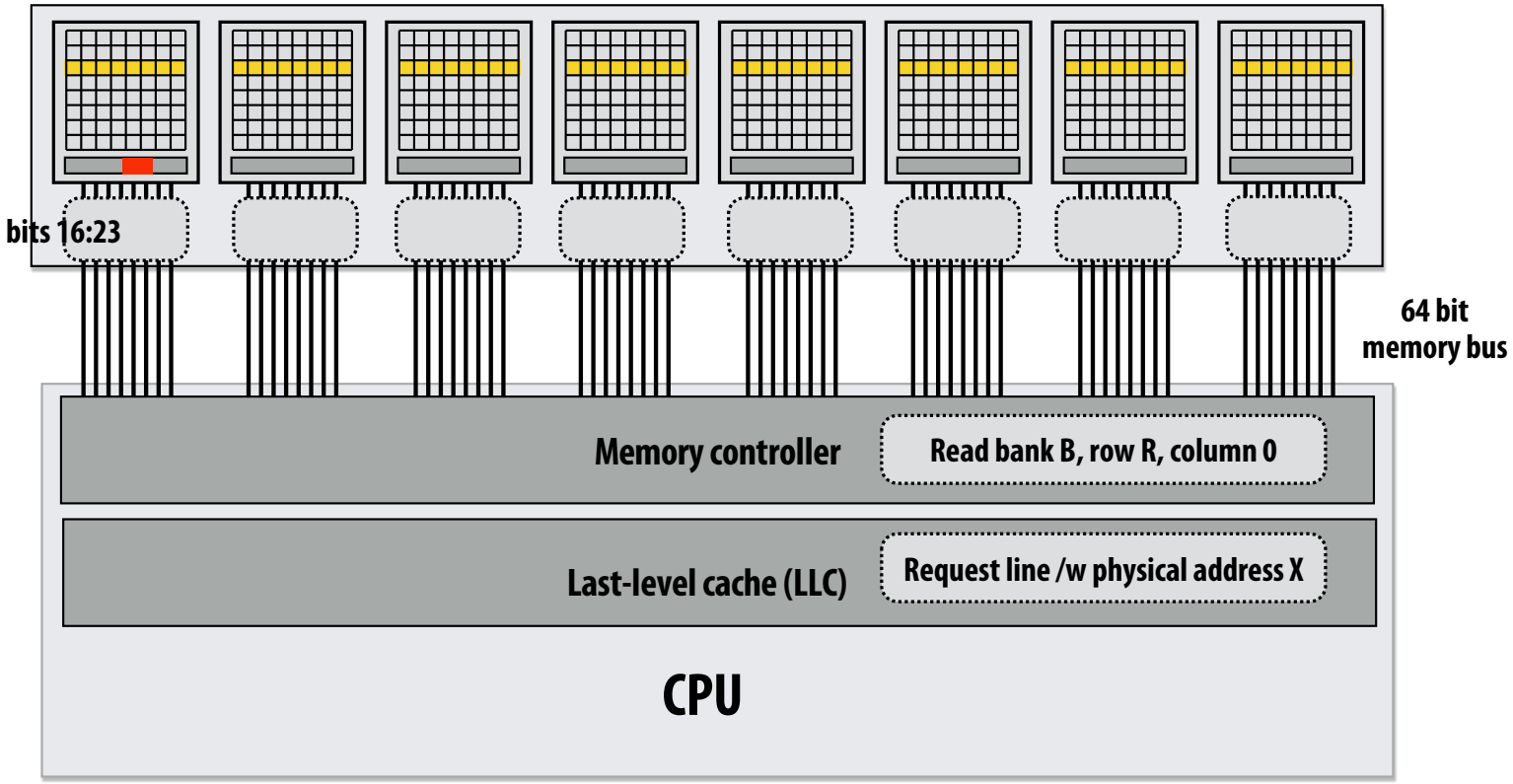
Reading one 64-byte (512 bit) cache line (the wrong way)

All data for cache line serviced by the same chip
Bytes sent consecutively over same pins



Reading one 64-byte (512 bit) cache line (the wrong way)

All data for cache line serviced by the same chip
Bytes sent consecutively over same pins

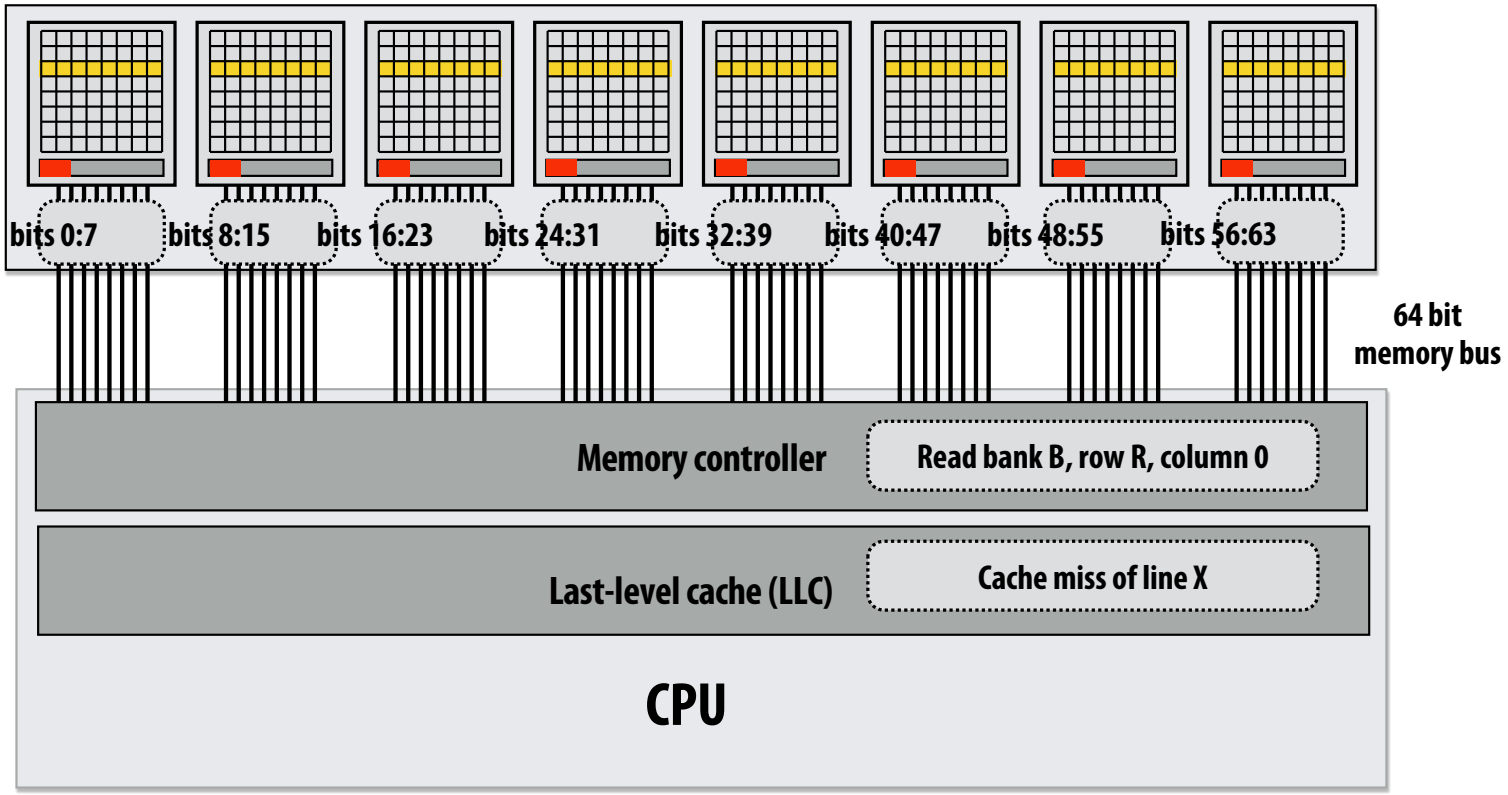


Reading one 64-byte (512 bit) cache line

Memory controller converts physical address to DRAM bank, row, column

Here: physical addresses are interleaved across DRAM chips at byte granularity

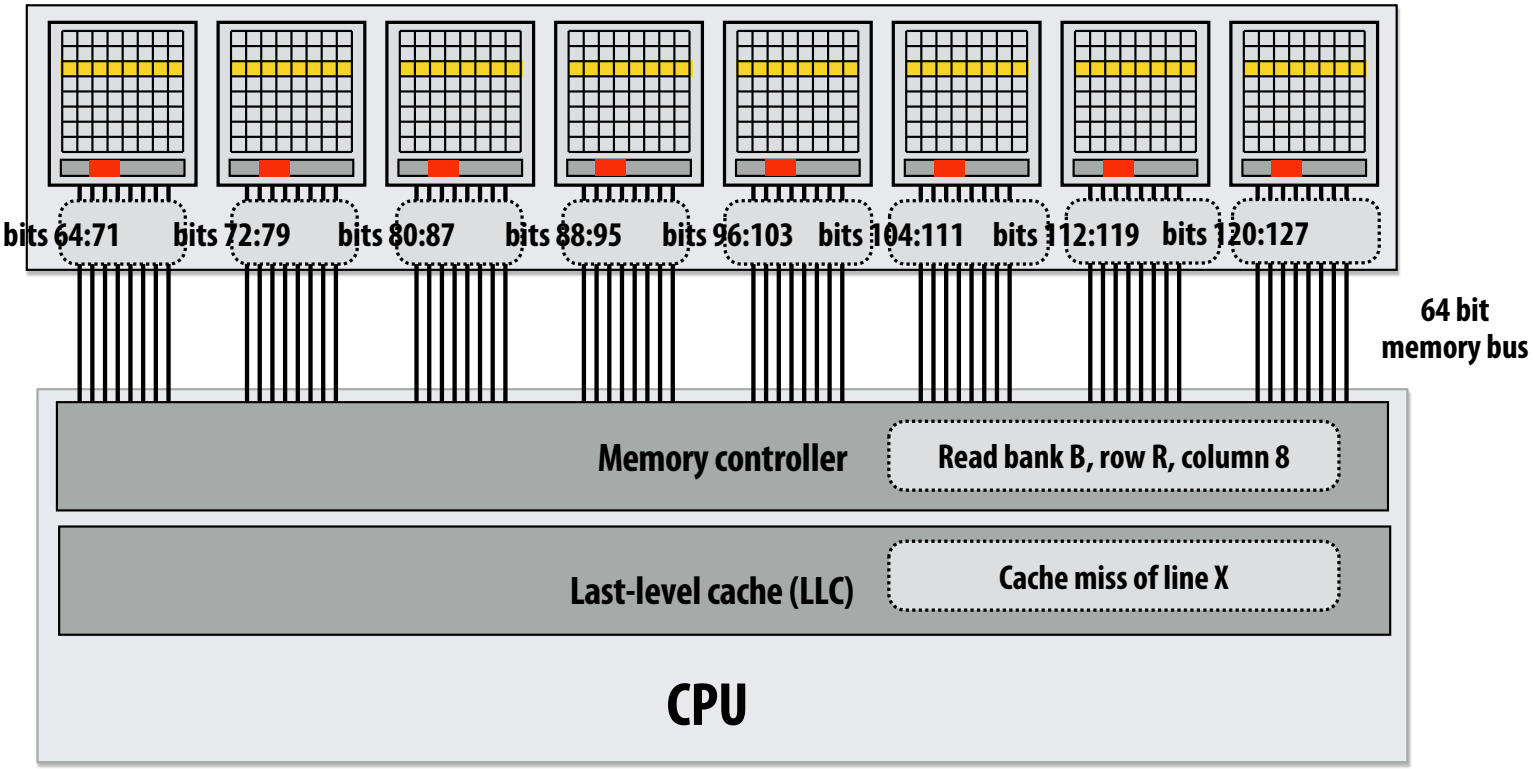
DRAM chips transmit first 64 bits in parallel



Reading one 64-byte (512 bit) cache line

DRAM controller requests data from new column *

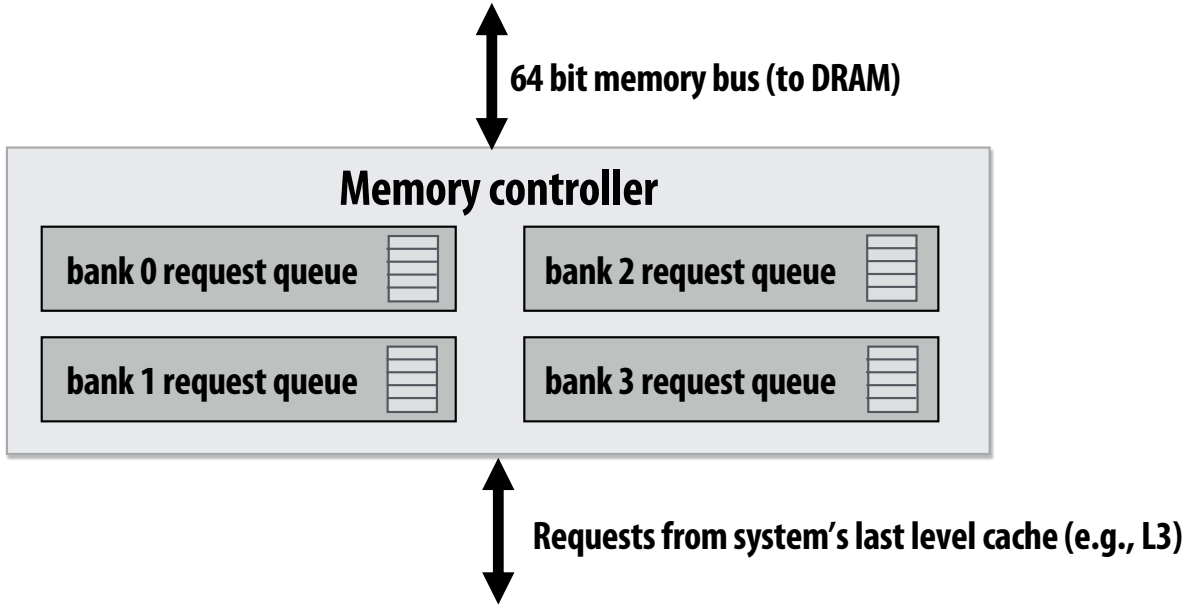
DRAM chips transmit next 64 bits in parallel



* Recall modern DRAM's support burst mode transfer of multiple consecutive columns, which would be used here

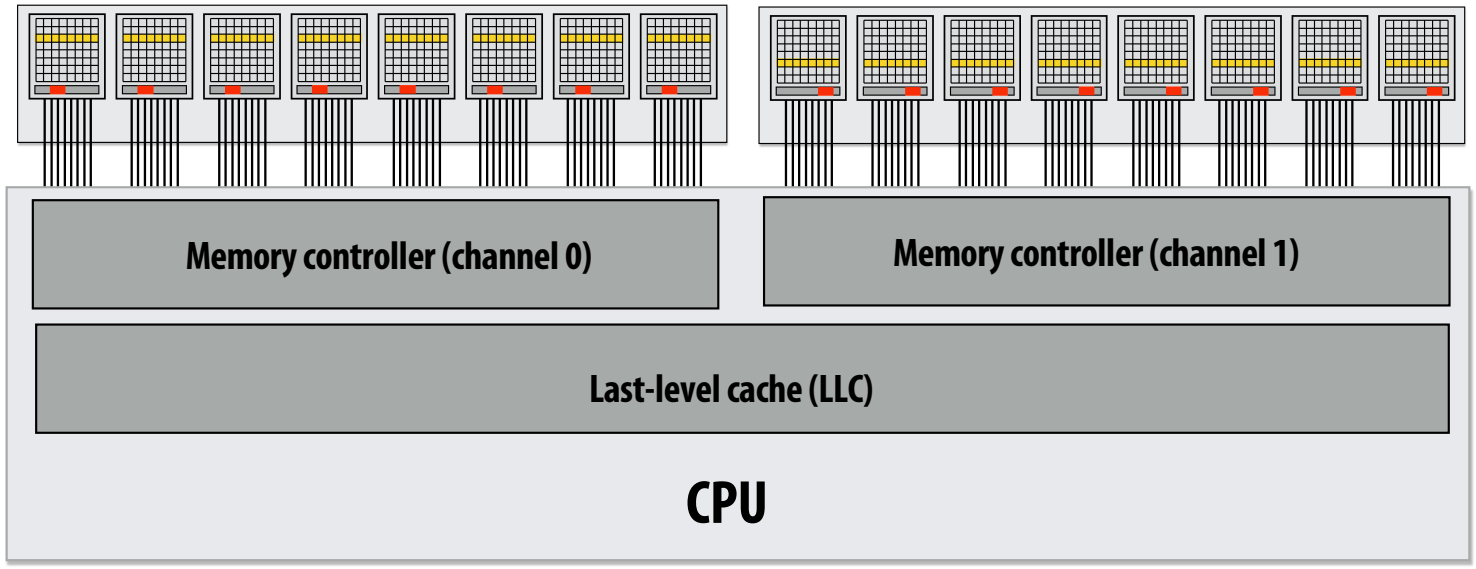
Memory controller is a memory request scheduler

- Receives load/store requests from LLC
- Conflicting scheduling goals
  - Maximize throughput, minimize latency, minimize energy consumption
  - Common scheduling policy: FR-FCFS (first-ready, first-come-first-serve)
    - Service requests to currently open row first (maximize row locality)
    - Service requests to other rows in FIFO order
  - Controller may coalesce multiple small requests into large contiguous requests (to take advantage of DRAM "burst modes")



Dual-channel memory system

- Increase throughput by adding memory channels (effectively widen bus)
- Below: each channel can issue independent commands
  - Different row/column is read in each channel
  - Simpler setup: use single controller to drive same command to multiple channels



Example: DDR4 memory

DDR4 2400

Processor: Intel® Core™ i7-7700K Processor (in Myth cluster)

- 64-bit memory bus x 1.2GHz x 2 transfers per clock* = 19.2GB/s per channel
- 2 channels = 38.4 GB/sec
- ~13 nanosecond CAS

Memory system details from Intel's site:

Memory Specifications

Max Memory Size (dependent on memory type)	64 GB
Memory Types	DDR4-2133/2400, DDR3L-1333/1600 @ 1.35V
Max # of Memory Channels	2
ECC Memory Supported ‡	No

*** DDR stands for "double data rate"**

https://ark.intel.com/content/www/us/en/ark/products/97129/intel-core-i7-7700k-processor-8m-cache-up-to-4-50-ghz.html

DRAM summary

- **DRAM access latency can depend on many low-level factors**
  - Discussed today:
    - State of DRAM chip: row hit/miss? is recharge necessary?
    - Buffering/reordering of requests in memory controller
- **Significant amount of complexity in a modern multi-core processor has moved into the design of memory controller**
  - Responsible for scheduling ten's to hundreds of outstanding memory requests
  - Responsible for mapping physical addresses to the geometry of DRAMs
  - Area of active computer architecture research

Modern architecture challenge:
improving memory performance:

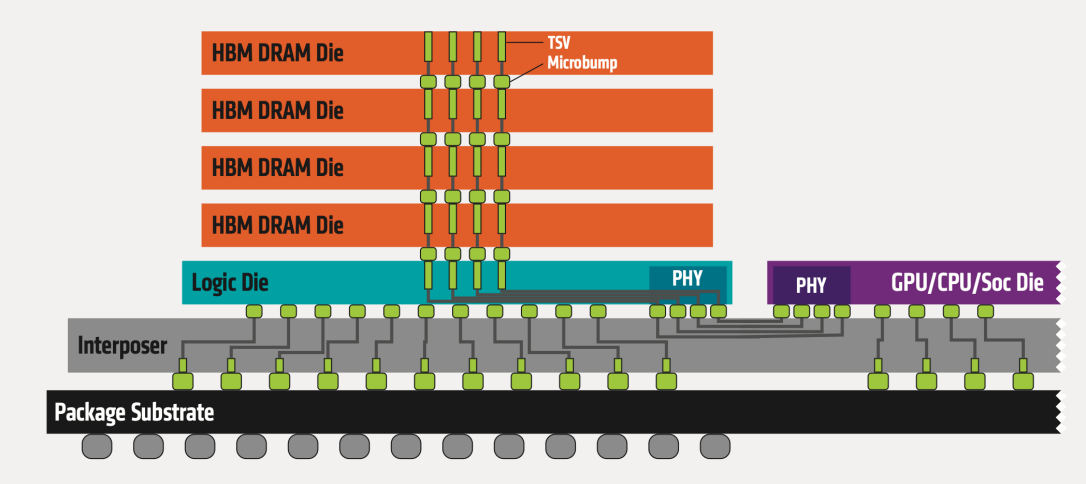
Decrease distance data must move by
locating memory closer to processors

(enables shorter, but wider interfaces)

Increase bandwidth, reduce power by chip stacking

Enabling technology: 3D stacking of DRAM chips

- DRAMs connected via through-silicon-vias (TSVs) that run through the chips
- TSVs provide highly parallel connection between logic layer and DRAMs
- Base layer of stack "logic layer" is memory controller, manages requests from processor
- Silicon "interposer" serves as high-bandwidth interconnect between DRAM stack and processor



Technologies:
Micron/Intel Hybrid Memory Cube (HBC)
High-bandwidth memory (HBM) - 1024 bit interface to stack

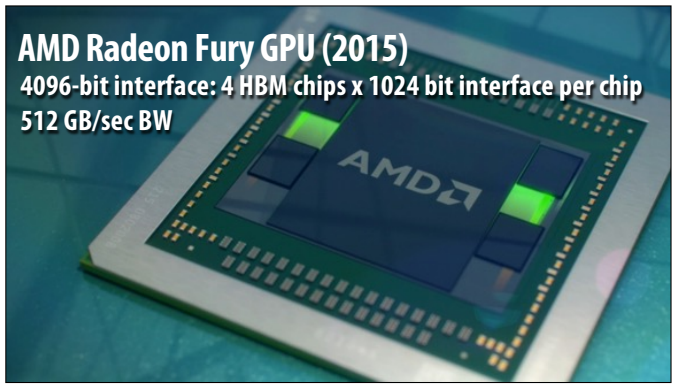
Image credit: AMD

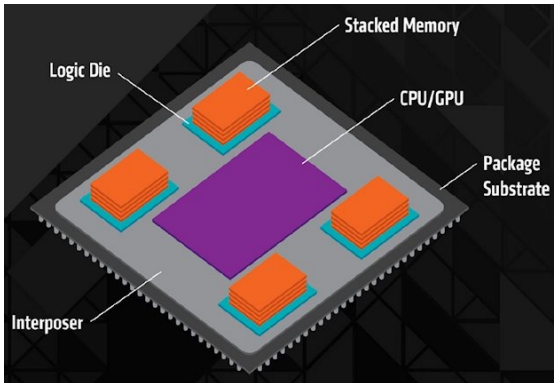
HBM Advantages

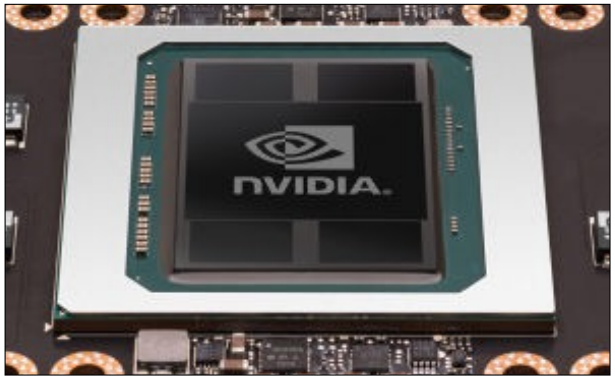
More Bandwidth
High Power Efficiency
Small Form Factor

	DDR4	LPDDR4(X)	GDDR6	HBM2	HBM2E (JEDEC)	HBM3 (TBD)
Data rate	3200Mbps	3200Mbps (up to 4266 Mbps)	14Gbps (up to 16Gb ps)	2.4Gbps	2.8Gbps	>3.2Gbps (TBD)
Pin count	x4/x8/x16	x16/ch (2ch per die)	x16/x32	x1024	x1024	x1024
Bandwidth	5.4GB/s	12.8(17)GB/s	56GB/s	307GB/s	358GB/s	>500GB/s
Density (per package)	4Gb/8Gb	8Gb/16Gb/2 4Gb/32Gb	8Gb/16Gb	4GB/8GB	8GB/16GB	8GB/16GB/ 24GB (TBD)

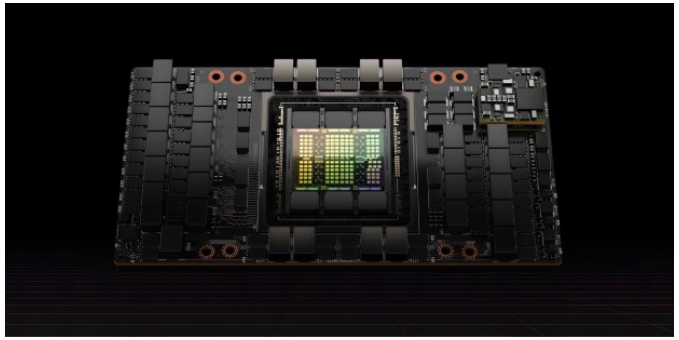
GPUs are adopting HBM technologies







NVIDIA P100 GPU (2016)
4096-bit interface: 4 HBM2 chips x 1024 bit interface per chip
720 GB/sec peak BW
4 x 4 GB = 16 GB capacity



NVIDIA H100 GPU (2022)
6144-bit interface: 6 HBM3 stacks x 1024 bit interface per stack
3.2 TB/sec peak BW
80 GB capacity

Summary: the memory bottleneck is being addressed in many ways

By the application programmer

- Schedule computation to maximize locality (minimize required data movement)

By new hardware architectures

- Intelligent DRAM request scheduling
- Bringing data closer to processor (deep cache hierarchies, 3D stacking)
- Increase bandwidth (wider memory systems)
- Ongoing research in locating limited forms of computation "in" or near memory
- Ongoing research in hardware accelerated compression (not discussed today)

General principles

- Locate data storage near processor
- Move computation to data storage
- Data compression (trade-off extra computation for less data transfer)