


Scaling Performance and Energy Efficiency Through Domain- Specific Accelerators

Tsung Tai Yeh

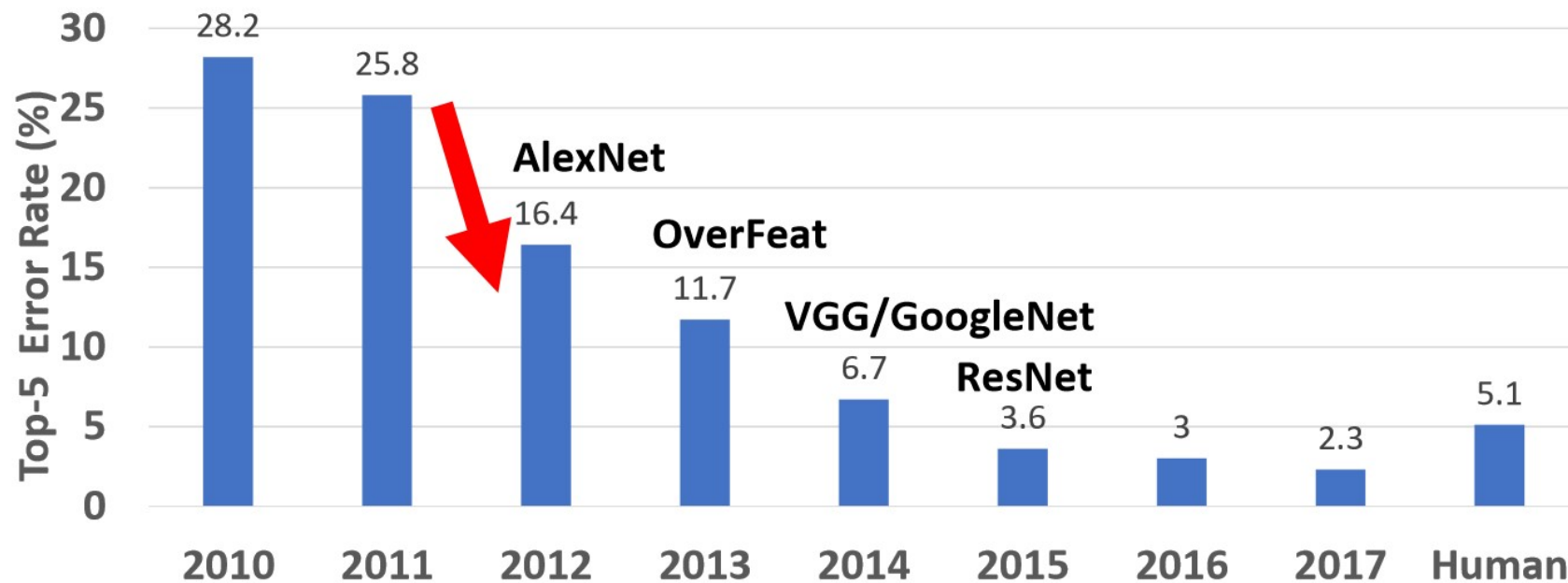
Computer Science Department of
National Chiao Tung University,
Taiwan

Overview

- Machine Learning & Deep Neural Network
- Golden Age of Microprocessor Design
- Domain Specific Accelerator
- Chiplet-based System
- Multi-tasking Computing

Why Deep Neural Network become popular?

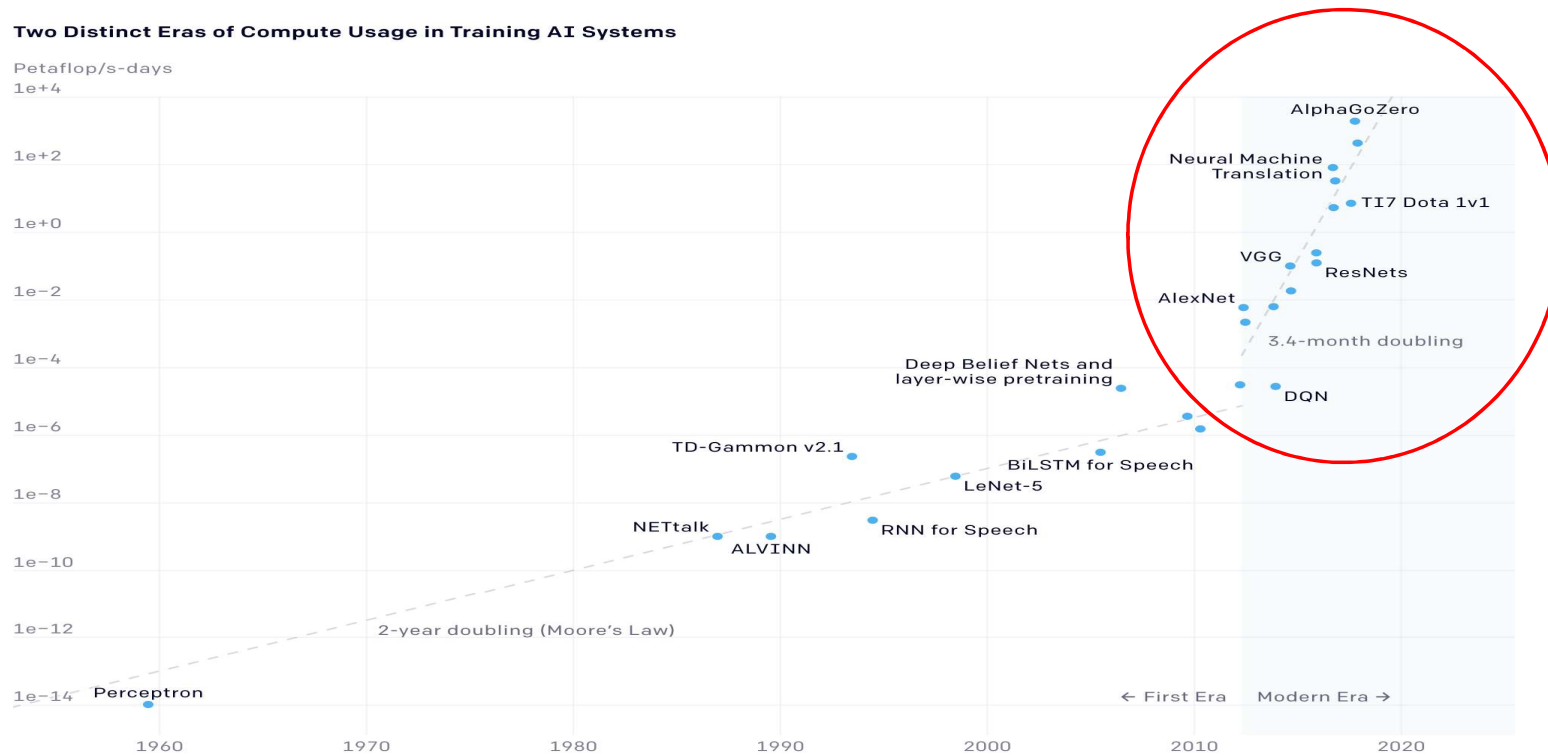
- DNN model outperforms human-being on the ImageNet Challenge



<https://arxiv.org/ftp/arxiv/papers/1911/1911.05289.pdf>

No free lunch on DNN computation

- AlexNet to AlphaGo Zero: A 300,000 x Increase in Compute



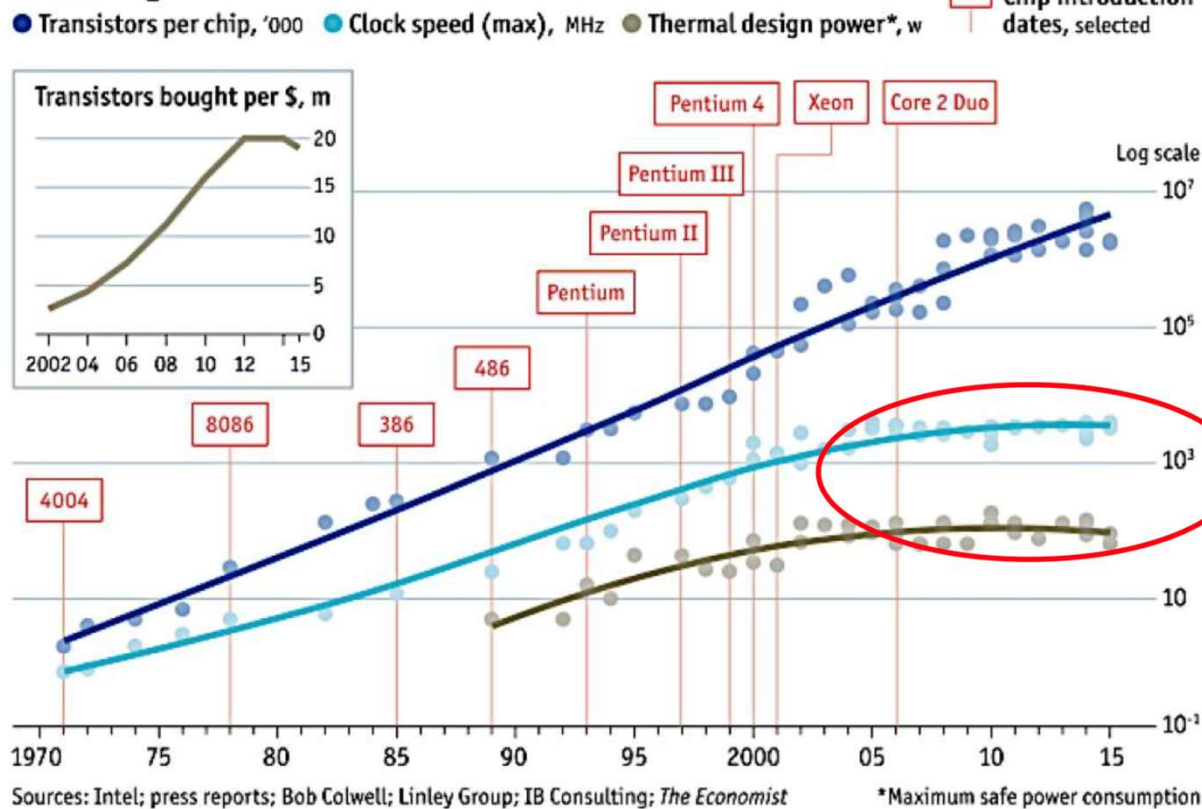
<https://arxiv.org/ftp/arxiv/papers/1911/1911.05289.pdf>

A Golden Age in Microprocessor Design

- A great leap in microprocessor speed $\sim 10^6$ X faster over 40 years
- Architectural innovations
 - Width: 8- \rightarrow 16- \rightarrow 32- \rightarrow 64 bits (~ 8 X)
 - Instruction level parallelism (ILP)
 - Multicore: 1 processor to 16 cores
 - Clock rate: 3 – 4000 MHz (~ 1000 X through technology & architecture)
- IC technology makes it possible
 - **Moore's Law**: growth in transistor count (2X every 1.5 years)
 - **Dennard Scaling**: power/transistor shrinks at the same rate as transistors are added

Increasing transistors is not getting efficient

Stuttering



General purpose processor is not getting faster and power-efficient because of **Slowdown of Moore's Law and Dennard Scaling**

Need **Specialized/Domain-specific accelerators** to improve computing speed and energy

Moore's Law

- The number of transistors per chip **doubles** every 18-24 months
- That has not been true for years
- It is getting to be increasingly difficult to maintain this exponential improvement !! Why?

Dennard Scaling

- As the size of the transistor becomes **small**
 - The voltage is reduced
 - Circuits can be operated at higher frequency at the same power

Related to
transistor
size



$$\mathbf{Power} = \text{alpha} \times \mathbf{C} \mathbf{F} \mathbf{V}^2$$

alpha: percent time switched

C: capacitance

F: Frequency

V: Voltage

What's wrong on
Dennard Scaling?

Dennard Scaling ignores "leakage current" , "threshold voltage"

So, as transistors get small, power density increases !!

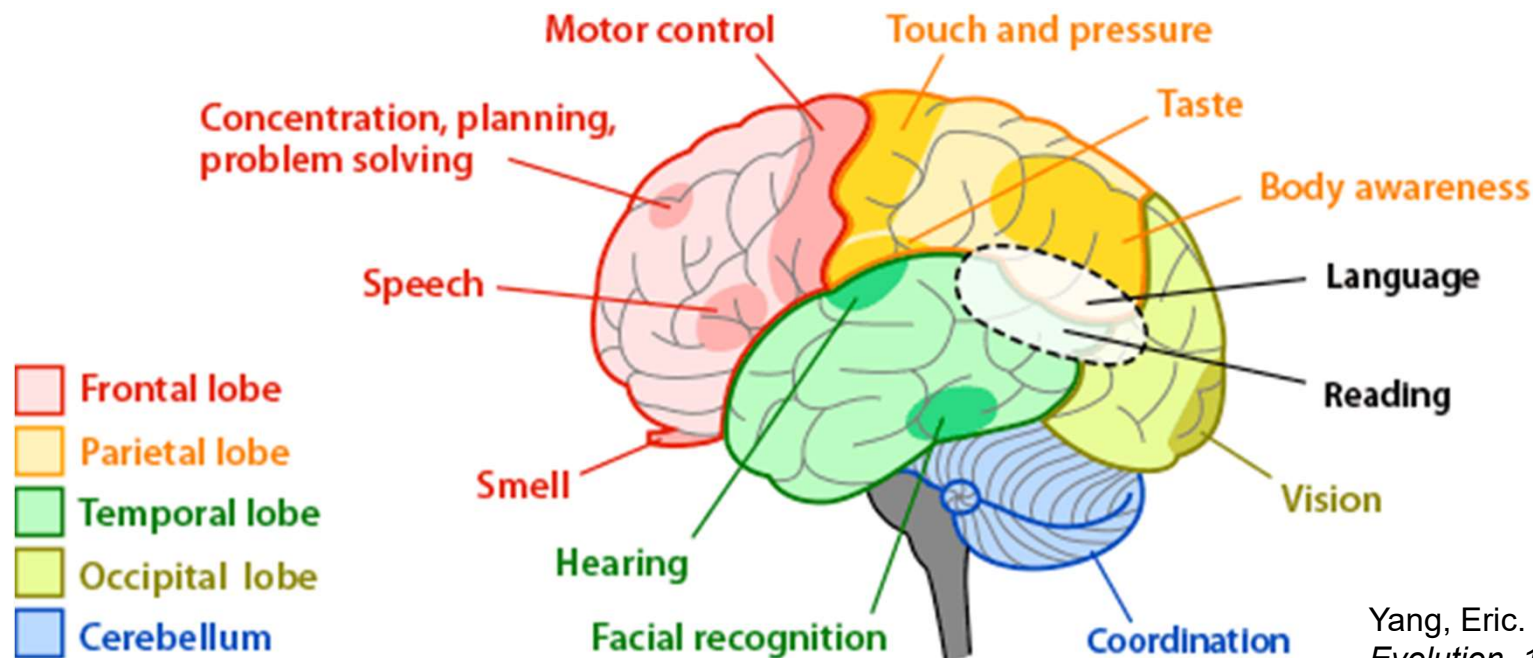
What's Left ?

- Transistors not getting much better
- Power budget not getting much higher
- One inefficient processor/chip to N efficient processors/chip
- One of paths left is **Domain Specific Architectures**
 - Just do a few tasks, but extremely well

Uncover Your Brain

2400 kcal/24 hr = 100 kcal/hr = 27.8 cal/
sec = 116.38 J/s = 116 W
20% x 116 W = 23.3 W

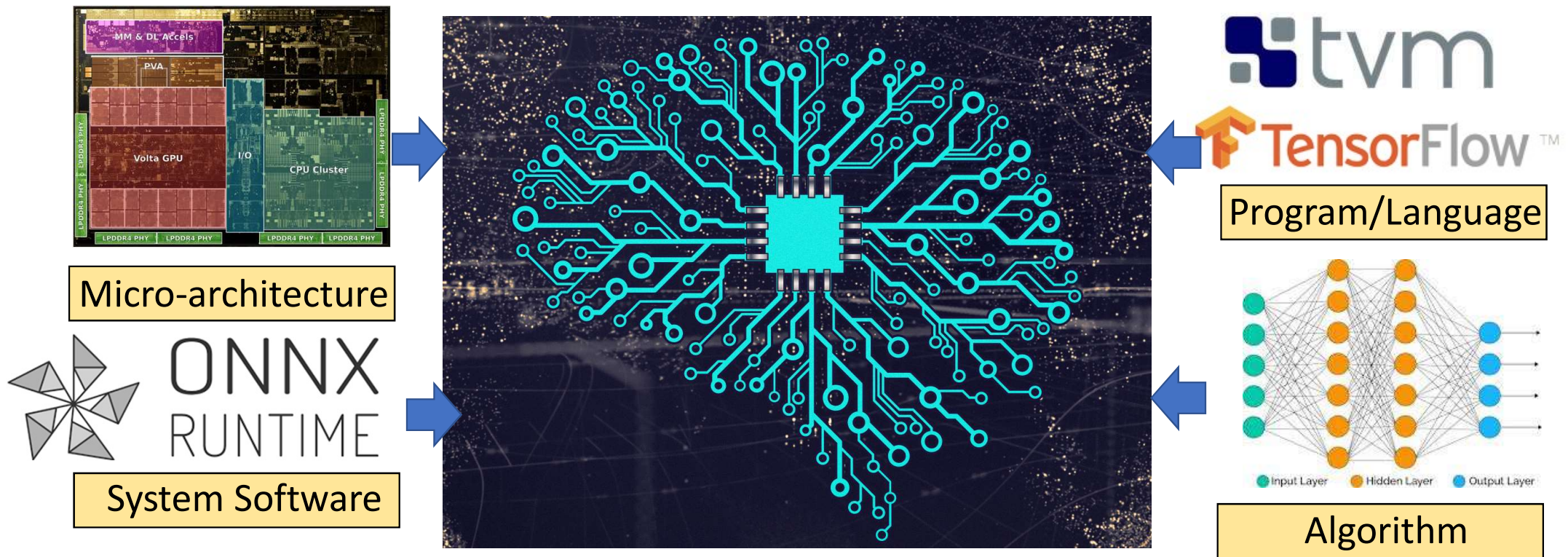
- The human-being brain comprises different areas (accelerators)
- An adult brain only consumes about 23 W a day !! (Yang)



Yang, Eric. *Think Dinner. Mac Evolution, 1998*

Learn from Human Being's Brain

- Designing “**Accelerators**” to boost up **Machine Learning**



Domain Specific Architecture (DSAs)

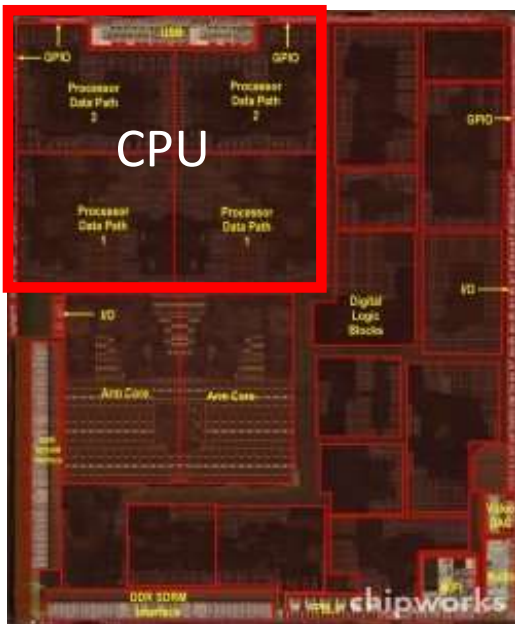
- Achieving higher performance by tailoring characteristics of domain applications to the architecture
 - Need domain-specific knowledge to work out good DSAs
 - Domain Specific Languages (DSLs) + DSAs (not strict ASIC)
 - Specialize to a **domain of many applications**
- Examples
 - GPU for computer 3D graphics, virtual reality
 - Neural processing unit (NPU) for machine learning
 - Visual processing unit (VPU) for image processing

Domain Specific Languages (DSL)

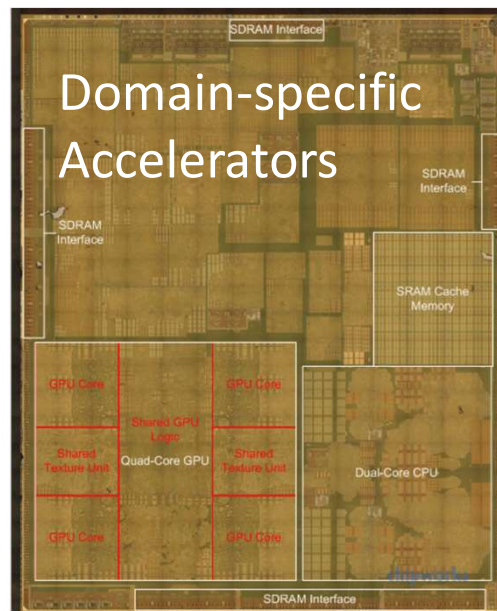
- DSLs target specific operations on a domain of applications
- Need vector, matrix or sparse matrix operations
- DSLs tailors for these operations
 - OpenGL, TensorFlow, Halide
- Compilers are important if DSLs are architecture-independent
 - Translate, schedule, map ISAs to right DSAs

Where is Domain-Specific Accelerators ?

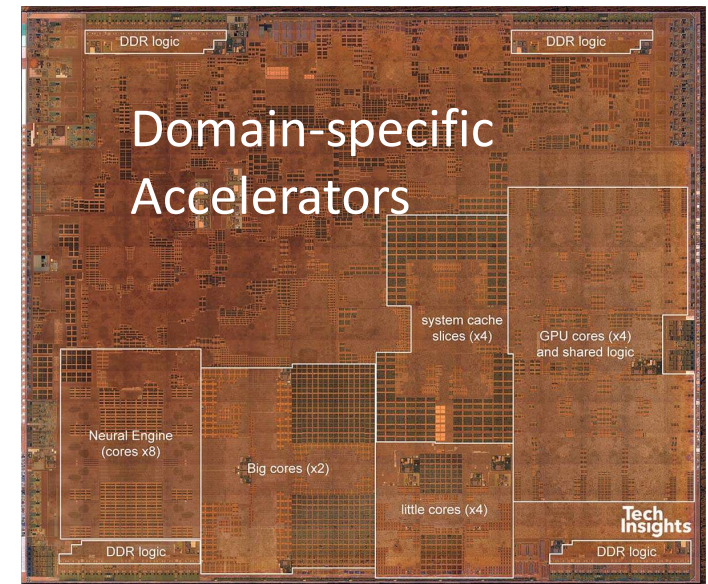
- Domain-Specific Accelerators are everywhere



2010 Apple A4
65 nm TSMC 53 mm²
4 accelerators



2014 Apple A8
20 nm TSMC 89 mm²
28 accelerators



2019 Apple A12
7 nm TSMC 83 mm²
42 accelerators

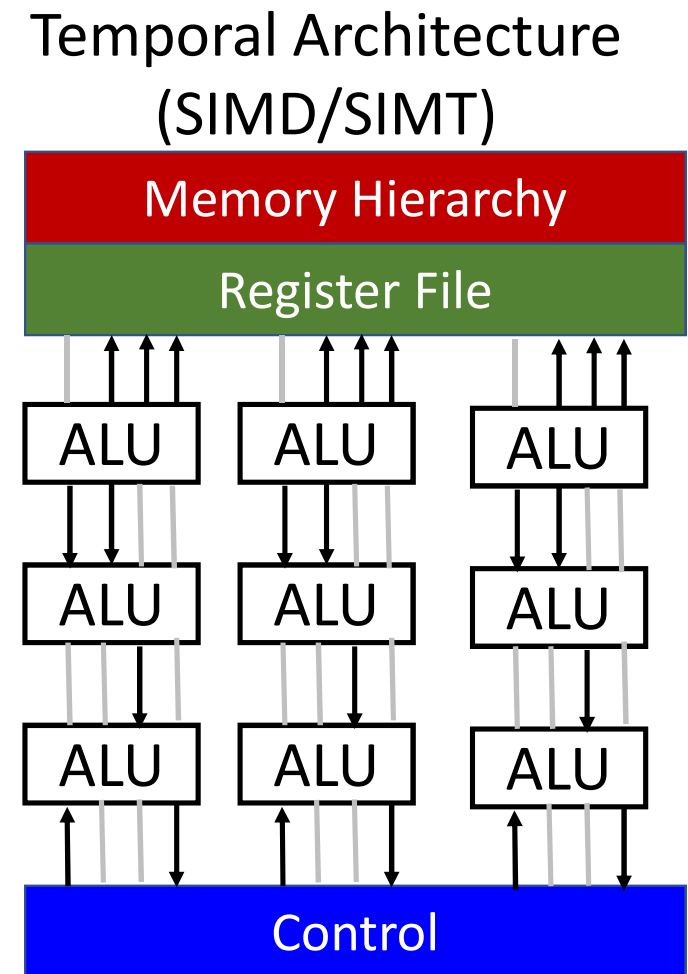
<https://edge.seas.harvard.edu/files/edge/files/alp.pdf>

Why DSAs can win ?

- More effective parallelism for a specific domain
 - SIMD vs. MIMD
 - VLIW vs. Speculative, out-of-order
- More effective use of memory bandwidth
 - User controlled vs. caches
- Eliminate unneeded accuracy (Quantization)
 - Lower FP/INT data precision (32 bit integers -> 8 bit integers)
- Increase the hardware utilization
 - Reduce the idle time on pipelining and LD/ST

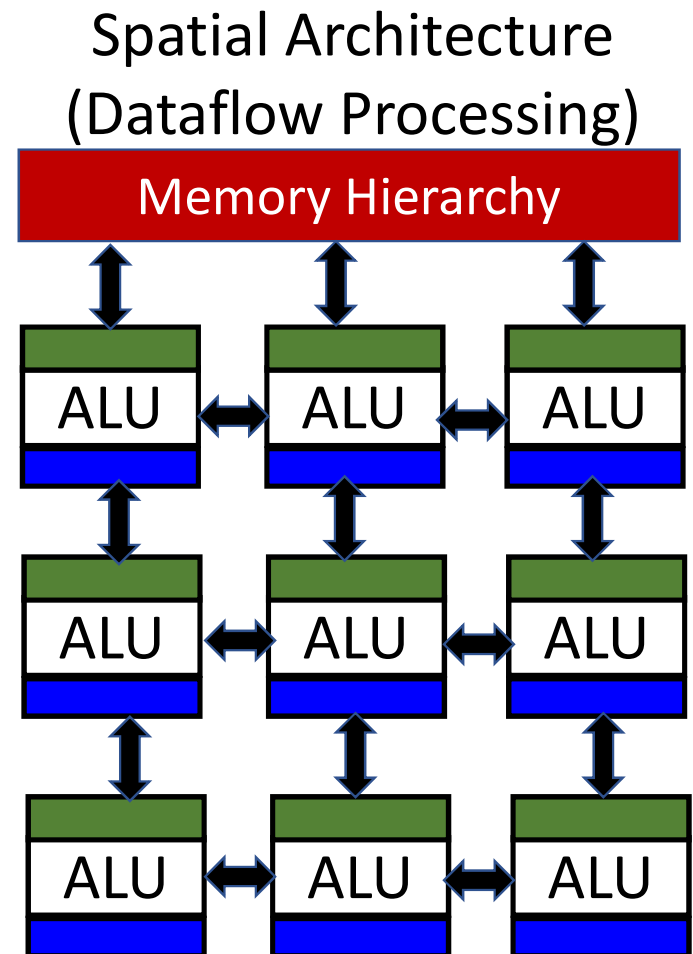
Design Aspects of Temporal Accelerator (TA)

- Centralized control for ALUs
- ALUs can only fetch data from the memory hierarchy
- ALUs “cannot” communicate directly with each other
- Why TA becomes popular? Parallelism
- Design aspects for DNN workloads
 - **Reduce # of multiplication** -> increase throughput
 - **Ordered computation (tiling)** -> improve memory subsystem



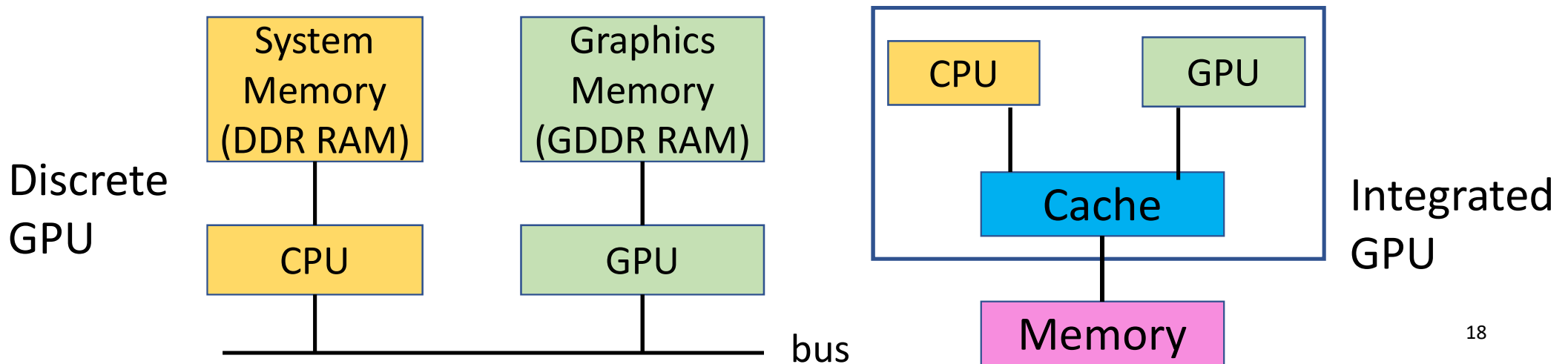
Design Aspects of Spatial Accelerator (SA)

- ALUs can pass data from one to another directly
- ALU can have its own control logics and local memory (registers)
- Dataflow processing
 - Programmable -> dynamic vs static graphs
 - Dynamic Mapping -> increase data reuse -> energy-efficiency
- Why SA are popular on DNN workloads?
 - Consume lower power & high throughput
 - Why? Data reuse -> reduce data movement



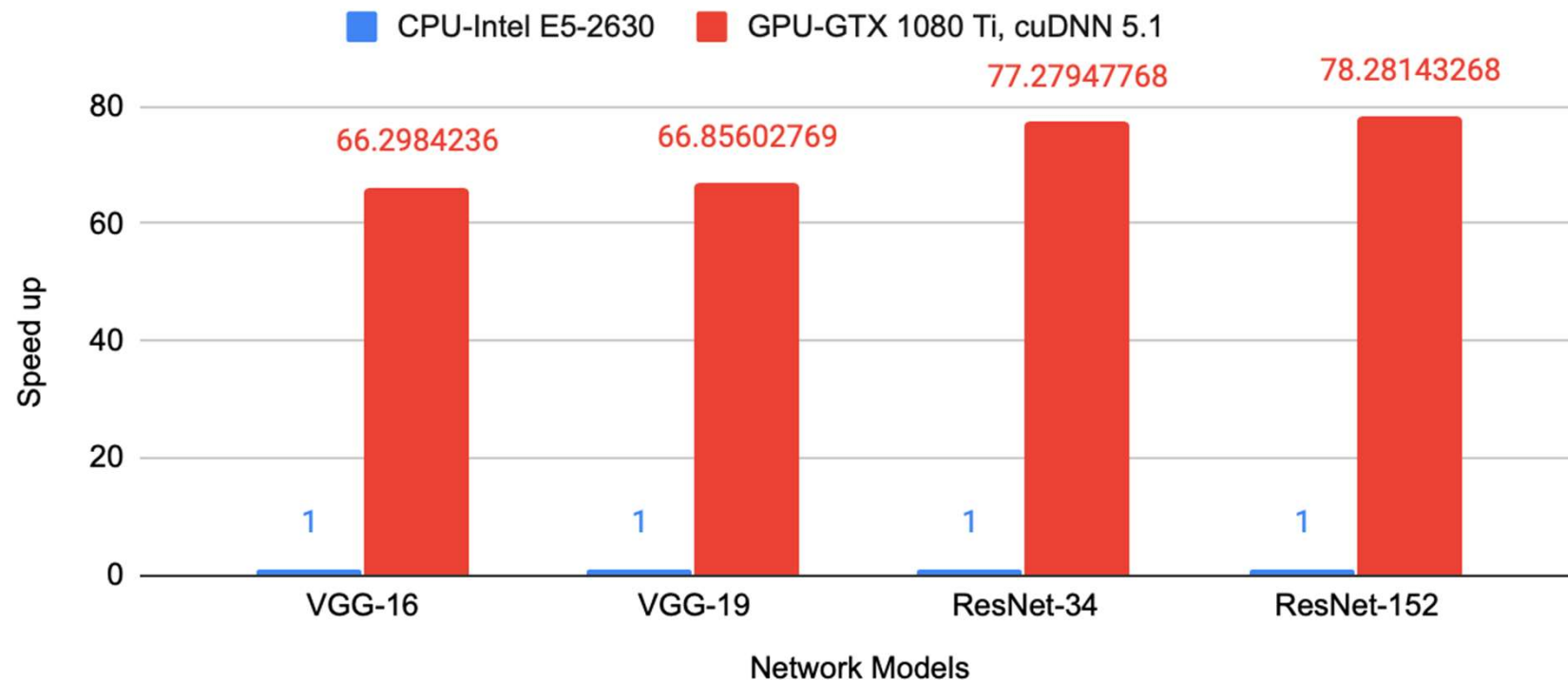
What is GPU?

- GPU = Graphics Processing Units
- Accelerate computer graphics rendering and rasterization
- Highly programmable (OpenGL, OpenCL, CUDA, HIP etc..)
- Why does GPU use GDDR memory?
 - DDR RAM -> low latency access, GDDR RAM -> high bandwidth




CPU vs GPU Training Time Comparison

- Normalized Training time on CPU and GPU (CPU has 16 cores, 32 threads)
- Why the model training on GPUs is much faster than on the CPU?



CPU vs GPU

	Cores	Clock Speed	Memory	Price	Throughput
CPU (Intel Core i7-7700k)	4	4.2 GHz	DDR4 RAM	\$385	~540 GFLOPs F32
GPU (Nvidia RTX 3090 Ti)	10496	1.7 GHz	DDR6 24 GB	\$1499	36 TFLOPs F32

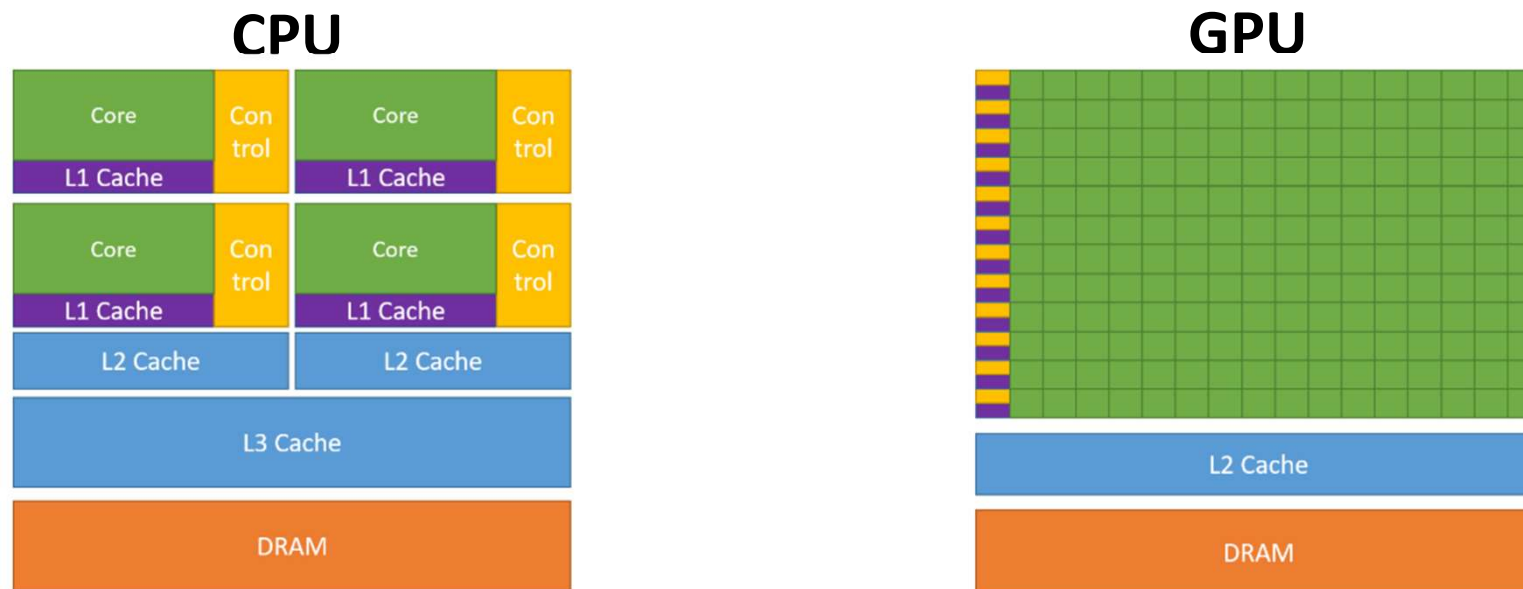


CPU: A **small** number of **complex** cores, the clock speed of each core is high, great for sequential tasks

GPU: A **large** number of **simple** cores, the clock speed of each core is low, great for parallel tasks

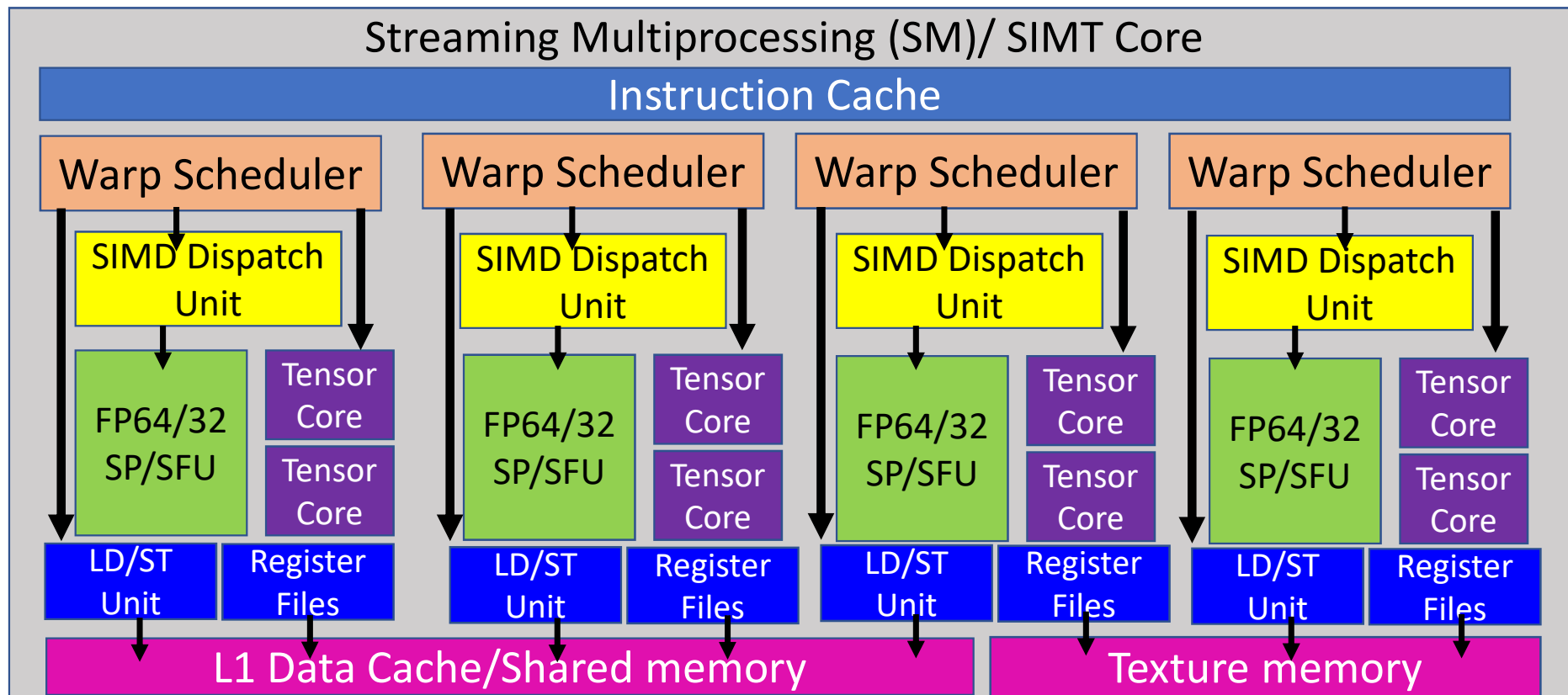
Why do we use GPU for computing ?

- What is difference between CPU and GPU?
 - GPU uses a large portion of silicon on the computation against CPU
 - GPU (2nJ/op) is more energy-efficient than CPU (200 pJ/op) at peak performance
 - Need to map applications on the GPU carefully (Programmers' duties)



What is Tensor Core on GPU?

- Execute $4 \times 4 \times 4$ matrix multiplication and addition in one cycle ($D = A \times B + C$)

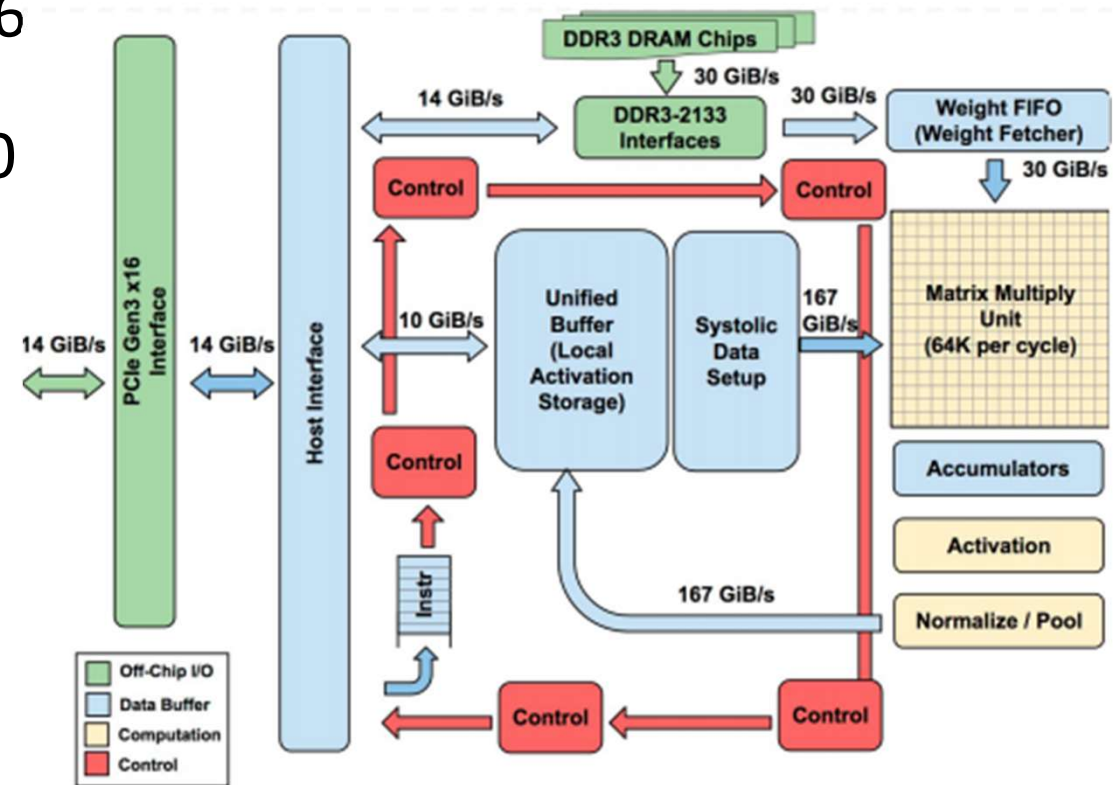


Why do we need Tensor Core on GPUs ?

- Higher throughput for GEMM ?
 - A CUDA (SIMT) core offers 1 single precision multiply-and-accumulate operation per GPU cycle
 - Tensor core can multiply two 4 x 4 F16 matrices and add the multiplication product F32 matrix per GPU cycle
 - Tensor core can achieve **125 Tflops/s** vs **15.7 Tflops/s** for the single precision operation
 - Domain-specific Accelerator within the GPU

Details in TPU v1

- **The Matrix Unit:** 64K (256 x 256) 8 bit INT multiply-accumulate
- Peak: 92T ops = 65536 x 2 x 700 MHz clock rate
- 4 MiB of 32-bit **Accumulator** collects 16 bit products
- Hardware **activation logics**
- 2.4 MiB **on-chip Unified Buffer** (Intermediate results)
- 3.5 X as much on-chip memory vs GPU
- 8 GiB **off-chip weight DRAM**

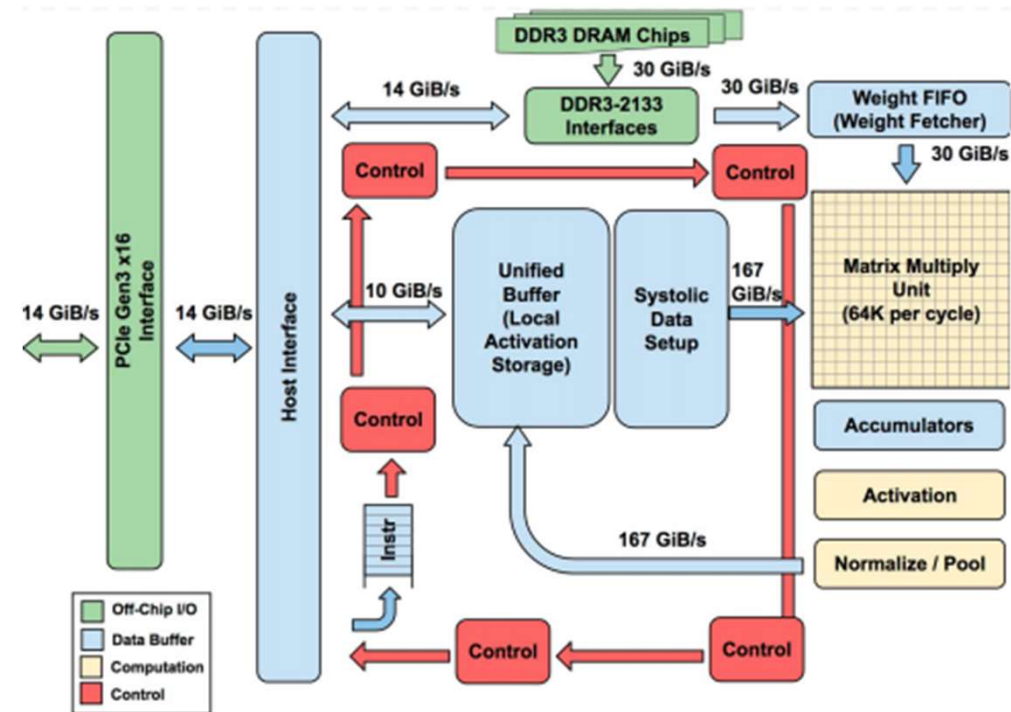


TPU Instruction Set Architectures

- TPU instruction follows the **CISC** fashion
- Average clock cycles per instructions > 10
- **No** program counter and branch instruction
- In-order issue
- SW controls buffer, pipeline synchronization
- A dozen instructions overall, five key ones
 - Read_Host_Memory
 - Read_Weights
 - MatrixMultiply/Convolve
 - Activate
 - Write_Host_Memory

TPU Microarchitecture

- **4-stage overlapped execution**, 1 instruction type/ stage
- Execute other instructions while MM is busy
- Read_Weight doesn't wait for weights fetched from DRAM
- The MM unit uses **not-ready** signal to indicate data aren't available in unified and Weight FIFO buffer



Performance Comparison

Processor	mm ²	Clock(MHz)	TDP (Watts)	Memory (GB/sec)	Peak TOPS/chip	
					8 b INT	32b FP
CPU: Haswell (18 core)	662	2300	145	51	2.6	1.3
GPU: Nvidia K80	561	560	150	160	--	2.8
TPU	<331	700	75	34	91.8	--

K80 and TPU in 28 nm process; Haswell fabbed in Intel 22nm process

Why TPU can Win ?

- Large matrix multiply unit
- Substantial software-controlled on-chip memory
- Data Quantization (8-bit INT)
- Parallelism on the hardware instead of Thread-level parallelism on GPUs
- What else ?

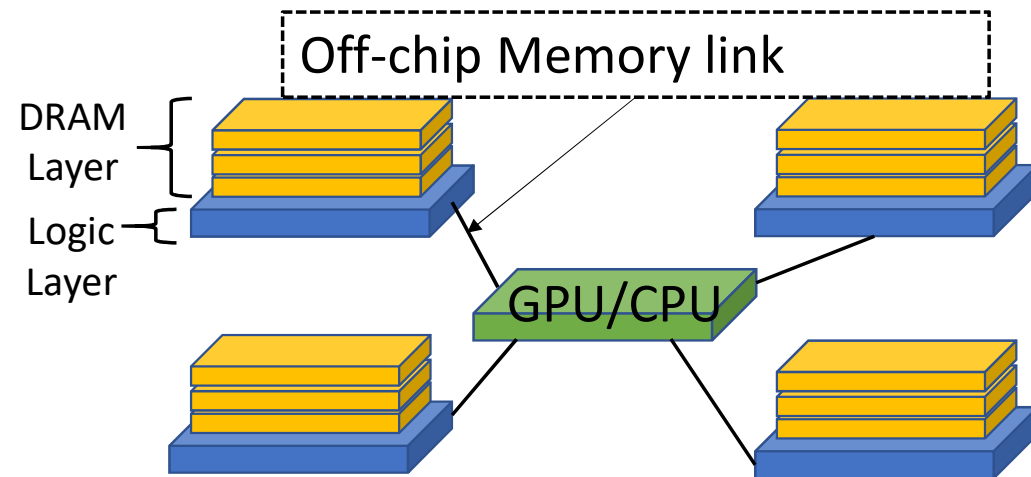
Chiplet-Based System

- Motivation

- Difficult to pack more functionality on a single chip
- High cost on the large chip
 - Verification cost is high
 - Manufacturing defects in densely packed logic can reduce the wafer yield

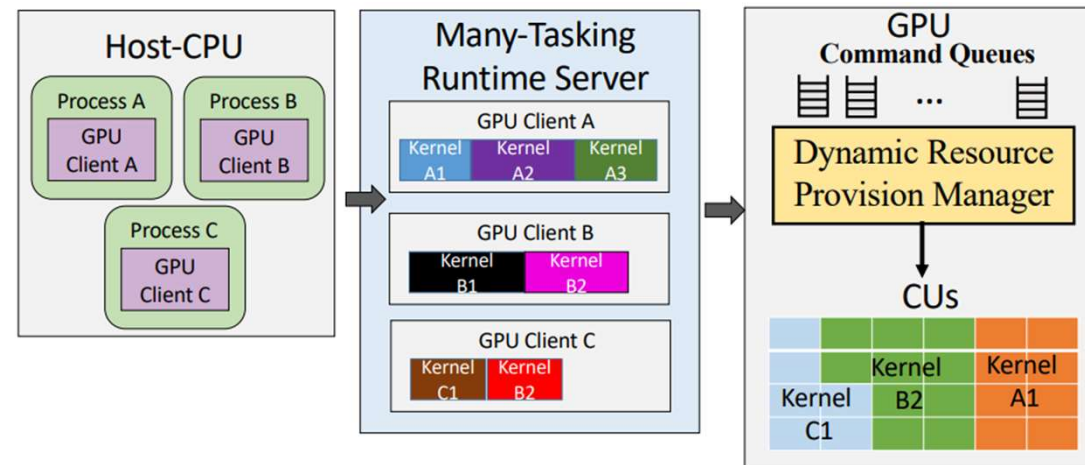
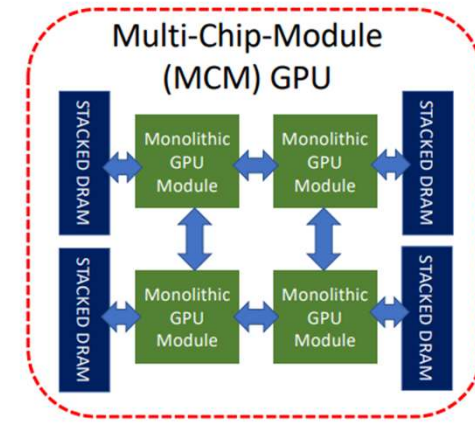
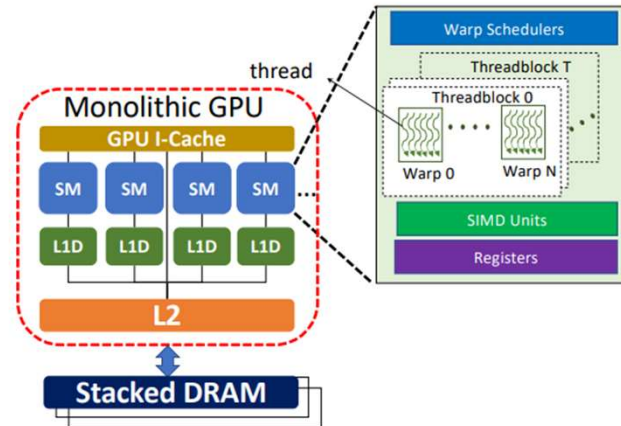
- Chiplet-based system

- The integration of multiple discrete chips within the same package
- Multi-chip module & silicon interposer



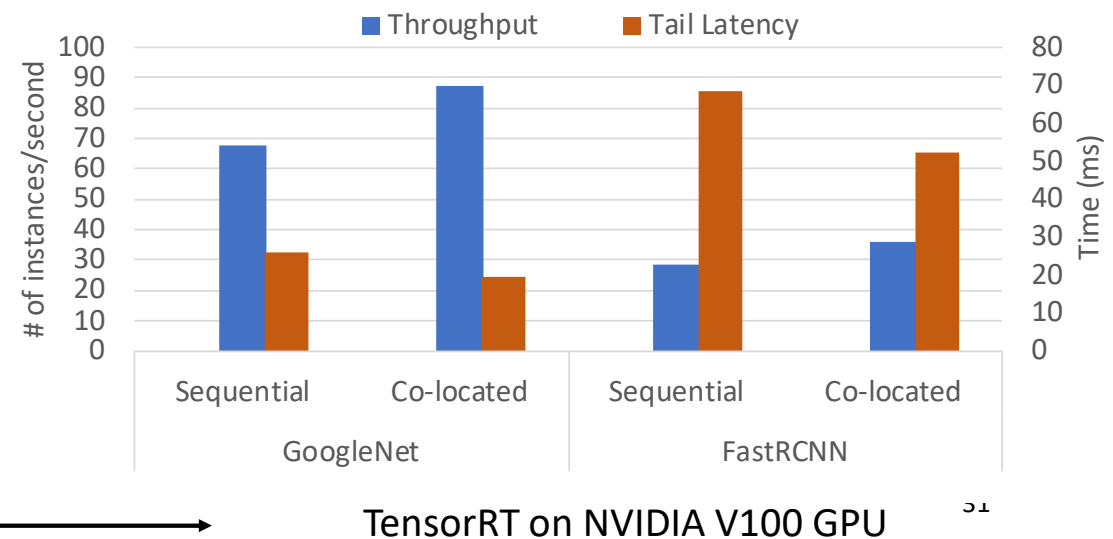
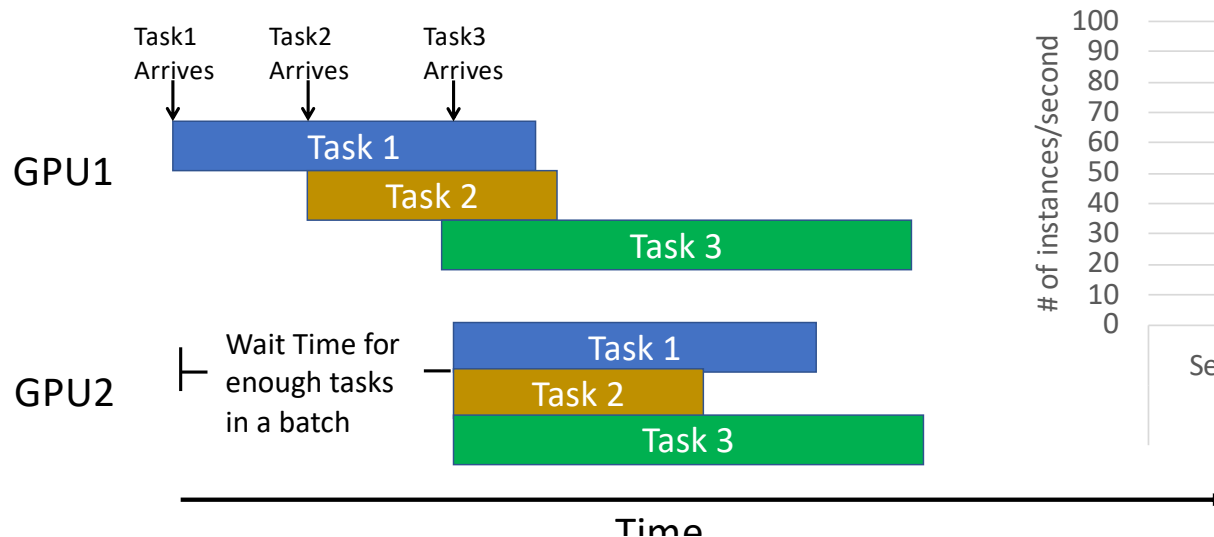
Multi-Tasking Computing

- Multi-tasking is everywhere
 - AI inference serving
 - Fintech (High Frequency Trading)
 - Networking/database
- Goals
 - High throughput
 - Low latency
 - High hardware resource util.
- Designs
 - QoS Scheduling
 - Virtualization



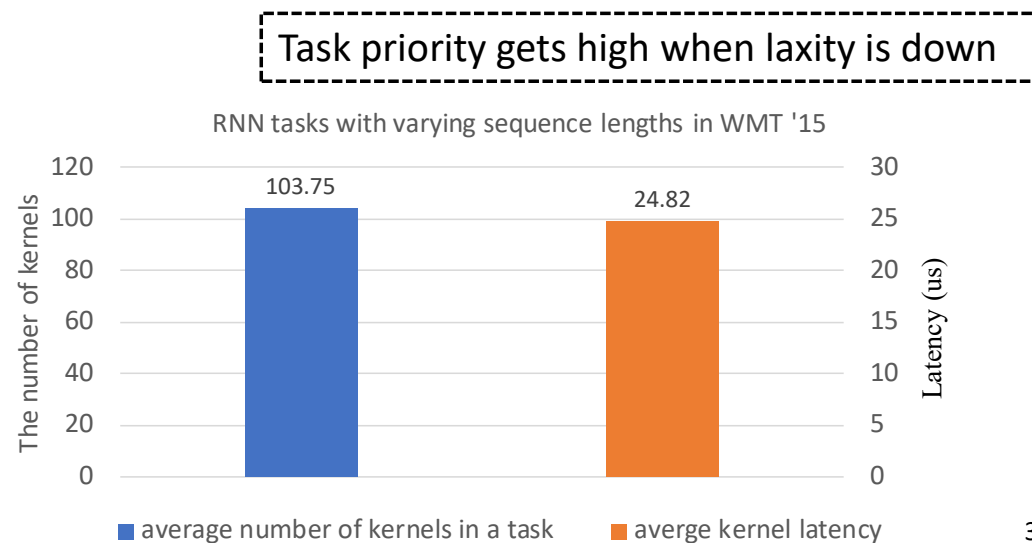
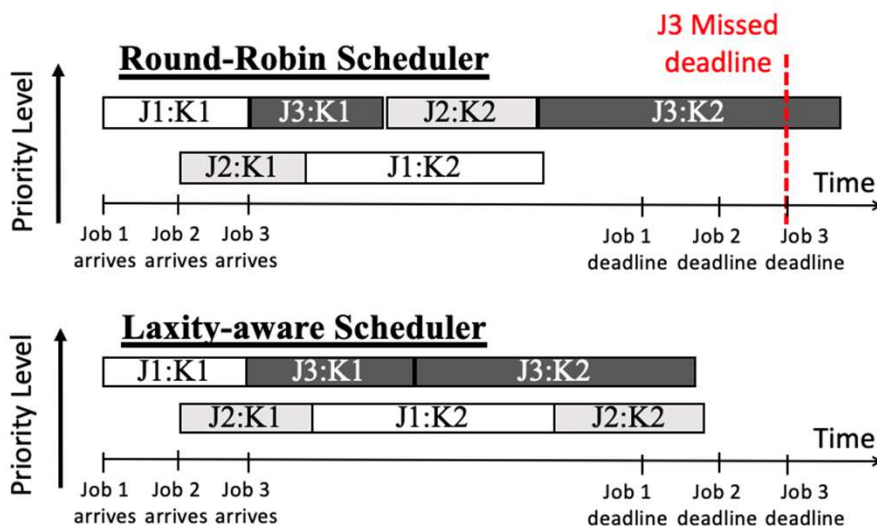
Deep Learning Inferencing Challenges

- GPU Utilization vs Inferencing latency
 - GPU can tackle multiple tasks simultaneously
 - Issue tasks in large batches (Util. is up, but increase task latency)
 - Task arrival time can be varying
 - Tasks have different length (Text sequence in RNN apps.)



LAX GPU Multi-tasking Scheduler

- Google TPU paper indicates 7 ms inference latency constraint
- Round-robin (RR) scheduler ignores QoS of inference apps
- How to run laxity-aware scheduling on the GPU ?
 - Laxity = Deadline – (TimeRemaining + DurationTime)

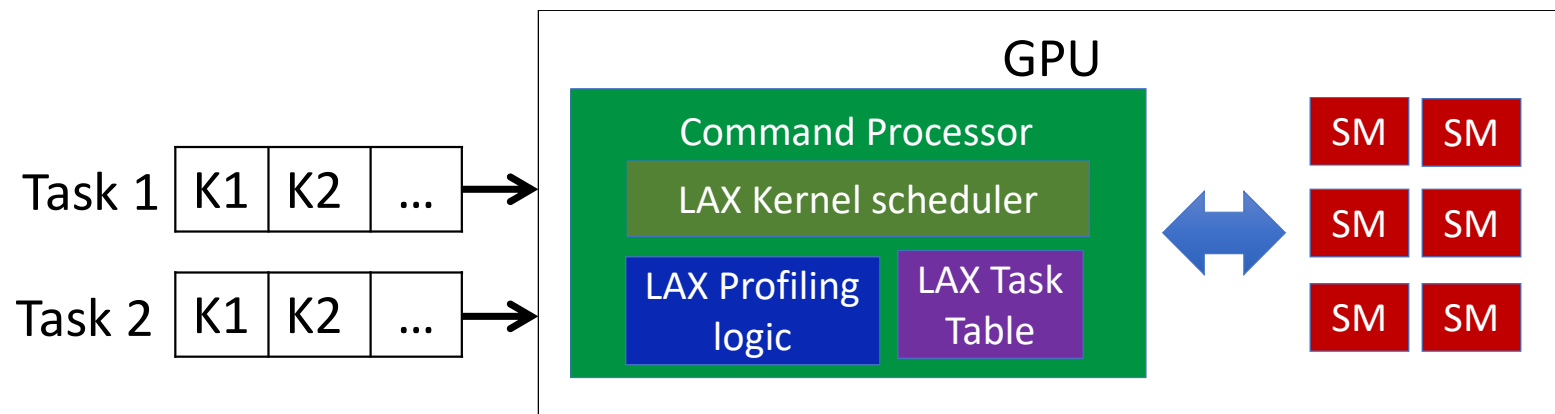


Laxity-base Scheduling Challenges on GPUs

- A DNN inference job often comprises many short-running kernels
- Simultaneous inference job execution incurs contention
- **Estimate the end-to-end inference latency** is difficult
- **Host-device handshaking overhead** is unacceptable
- **High OS-managed CPU scheduling overhead**

LAX Overview

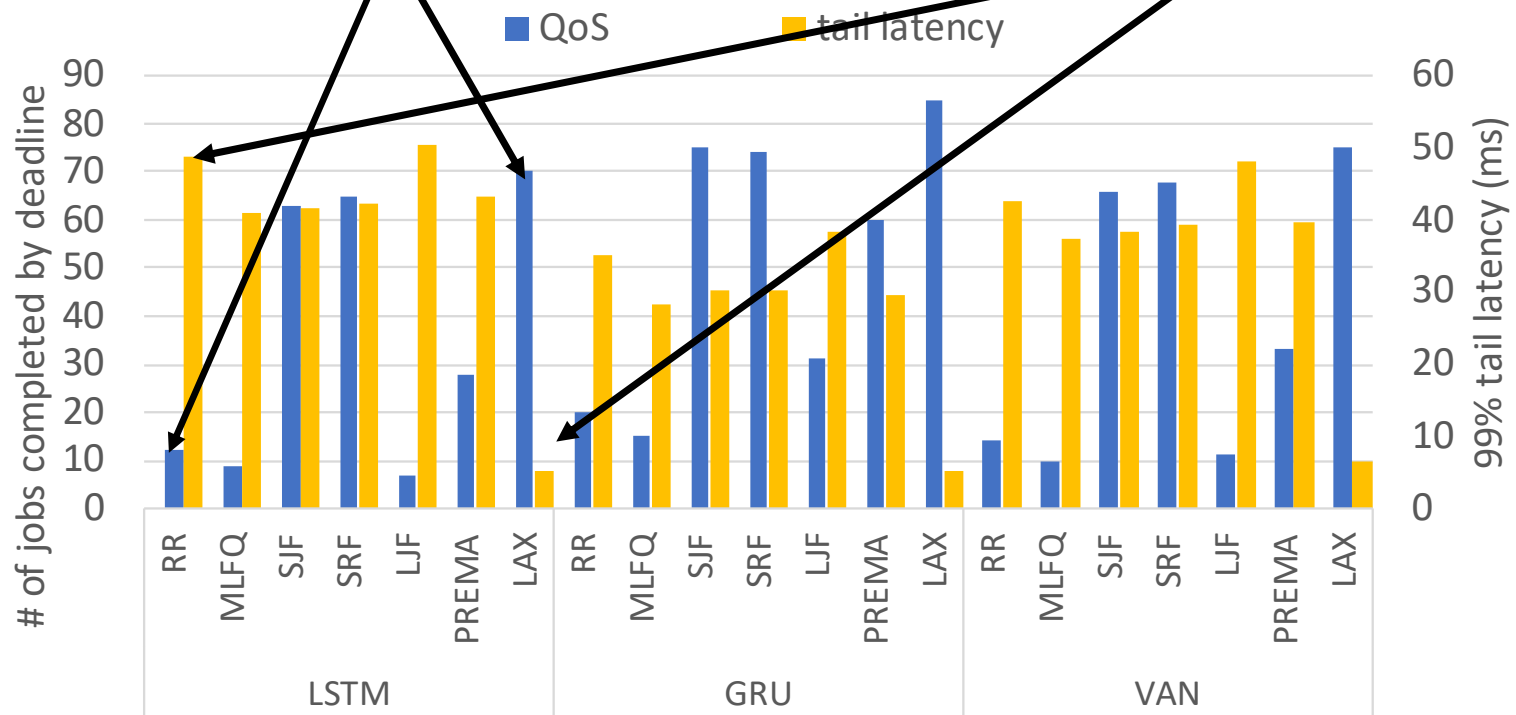
- How to predict the GPU task's remaining time ?
 - Hierarchical DNN models
 - Command Processor parses kernels
 - Workgroup-centric estimation mechanism
- How does LAX work when the task arrival rate is high ?
 - Migration + pull model to determine proper # of active tasks



LAX QoS vs. Tail Latency

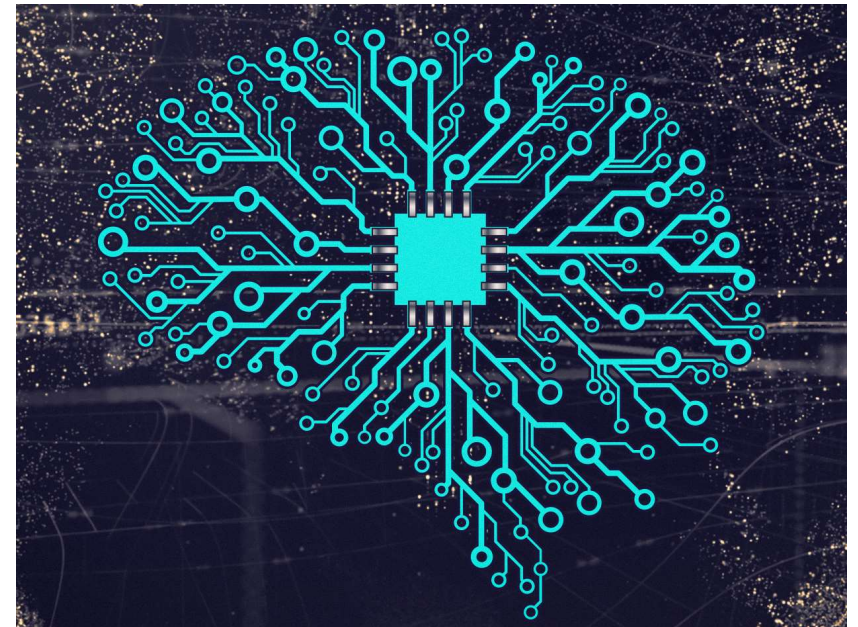
LAX's LSTM job QoS is 5.8 X higher than RR

LAX's LSTM job tail latency is 5.6 X lower than RR



Conclusion

- **Domain-specific Accelerator** is one of paths to continue the increase of performance and energy efficiency of computing hardware
- **Domain-Specific Language + Accelerator**
- Software defined hardware
 - RTL -> parallel high level language



Thank You!!

Q & A