http://inst.eecs.berkeley.edu/~ee290-2/sp21/

Hardware for Machine Learning

Lecture 19: Accelerator-Level Parallelism

Sophia Shao

Instruction-Level Parallelism

"It was a wonderful team of people there. For example, Bob Tomasulo developed "Tomasulo's algorithm", which he called— "the common data bus". But I called it the Tomasulo algorithm in a paper that was written contemporaneously with this [machine development] and that name stuck. This was an early version of data flow. Other ideas: the speculation on branch, the out of order execution of instructions, all were— they were very early developments in computing, incorporated in this machine."

Oral History of Michael J. "Mike" Flynn

https://archive.computerhistory.org/resources/access/text/2013/06/102746610-05-01-acc.pdf

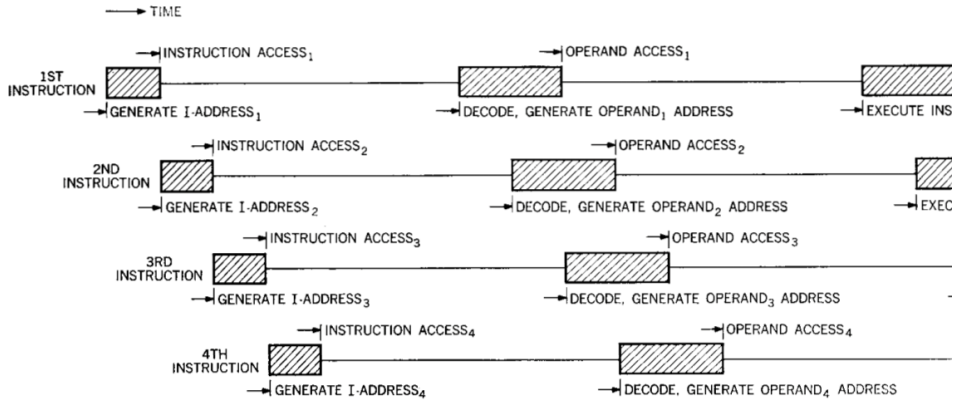
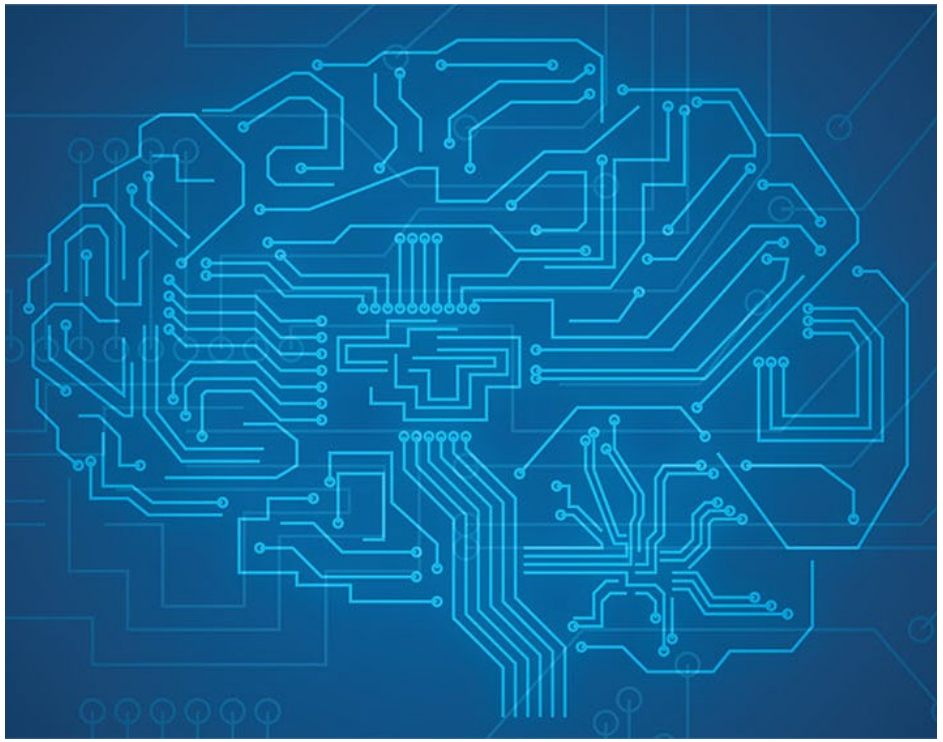


Figure 2 Illustration of concurrency among successive instructions.

The IBM System/360 Model 91: Machine Philosophy and Instruction – Handling, Anderson, Sparacio, Tomasulo

Review

- Core computation in DNN
- Execution order of the core computation
- Hardware realization of the core computation
- Mapping DNNs to hardware
- Data transfer mechanisms across storage hierarchy
- Sparsity in DNNs
- Codesign example
- Other Operators and Near-Data Processing
- Training Kernels



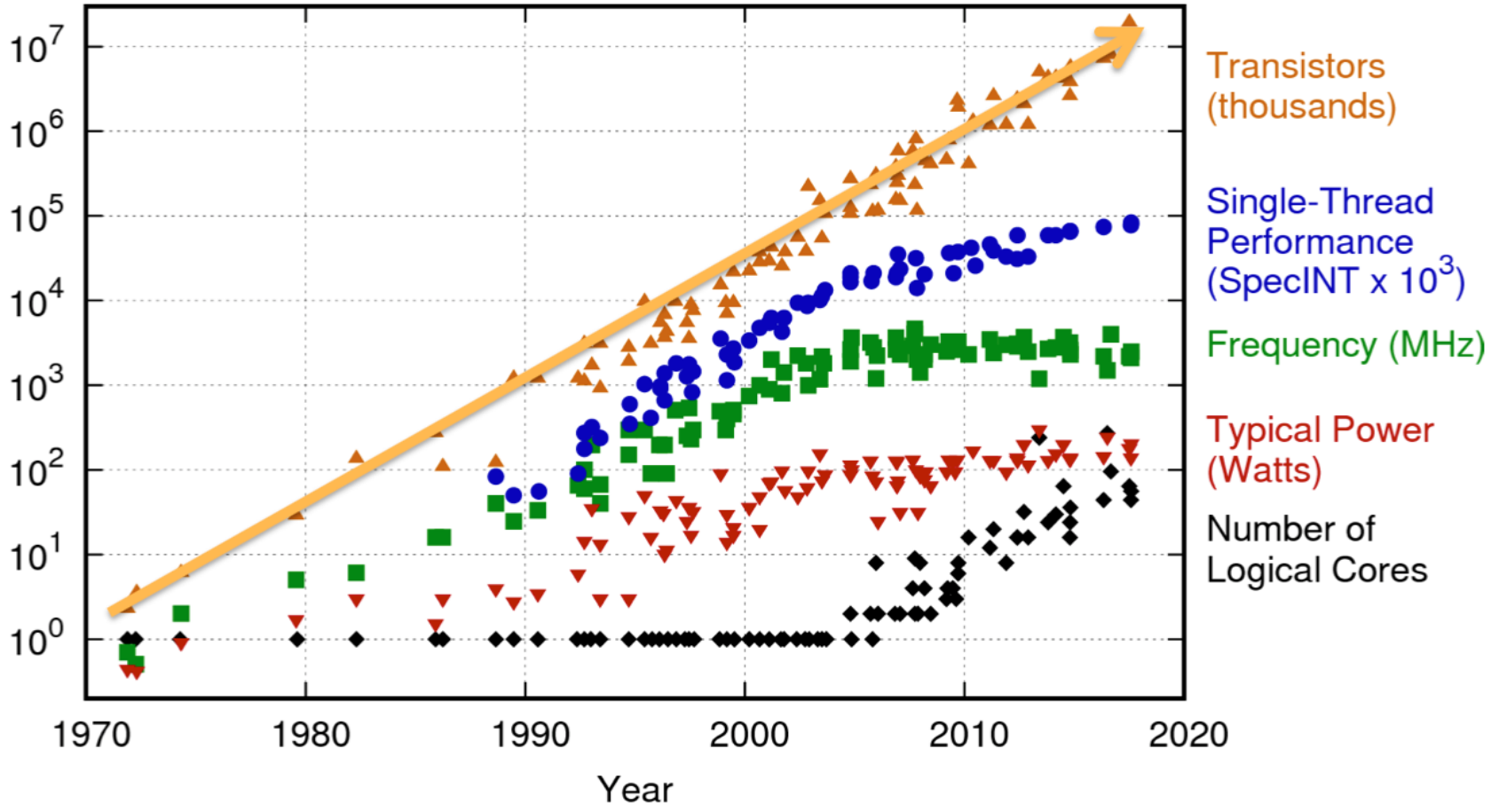
Accelerator-Level Parallelism (ALP)

- X-Level Parallelism
- ALP in Mobile SoC
- Open Challenges

Acknowledgement

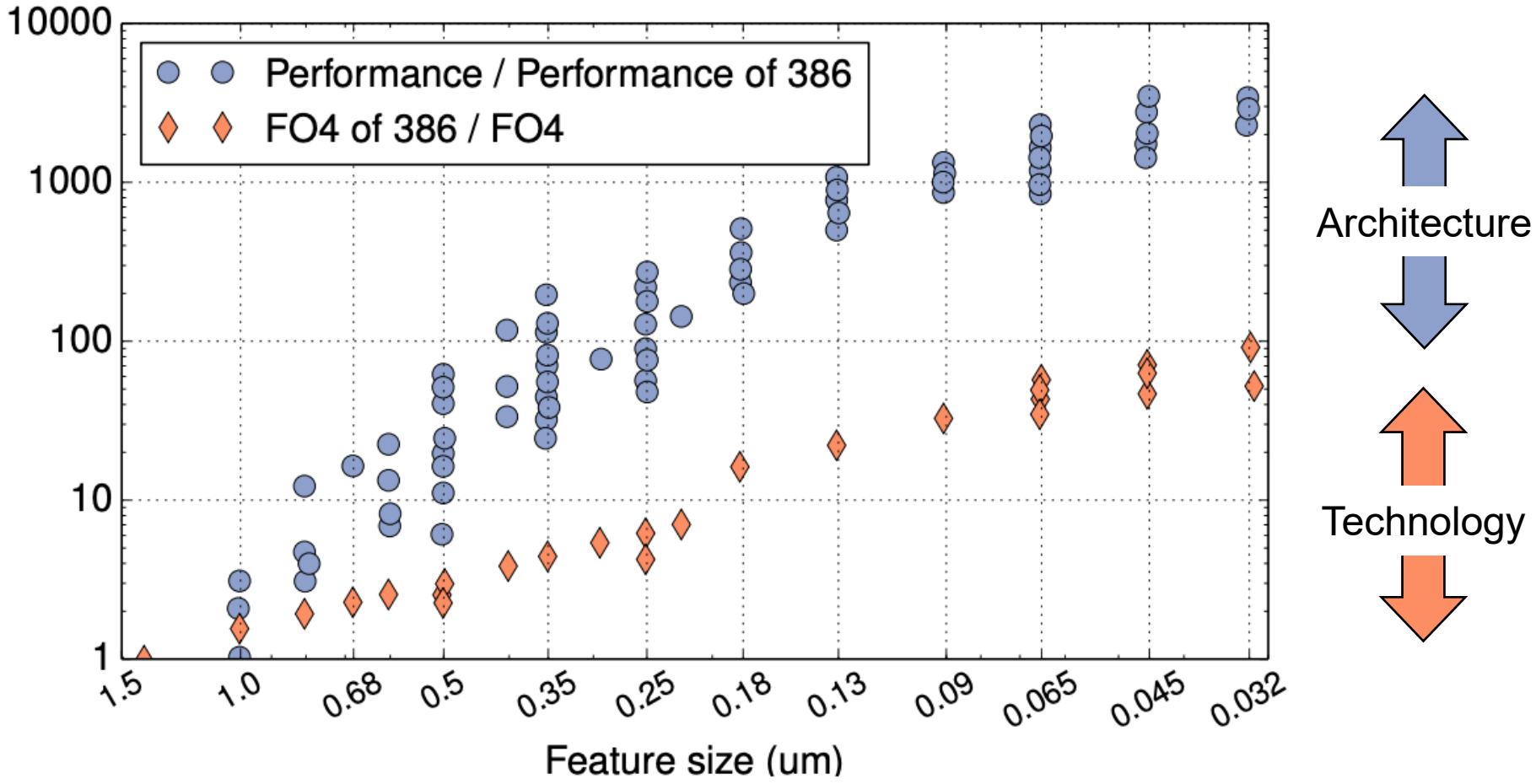
- “Accelerator-Level Parallelism” was coined by Mark Hill and Vijay Janapa Reddi. More details can be found:
  - Accelerator-level Parallelism, arXiv, 2019, https://arxiv.org/abs/1907.02064
  - Accelerator-level Parallelism, SIGARCH Blog, 2019 https://www.sigarch.org/accelerator-level-parallelism/
  - Accelerator-level Parallelism, Presentation, 2020 http://www.cs.wisc.edu/multifacet/papers/ALP_microsoft2020_02.pptx

Advances in Processor Performance



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten. New plot and data for 2010-2017 by K. Rupp from https://www.karlrupp.net/wp-content/uploads/2018/02/42-years-processor-trend.png

Enablers: Technology and Architecture

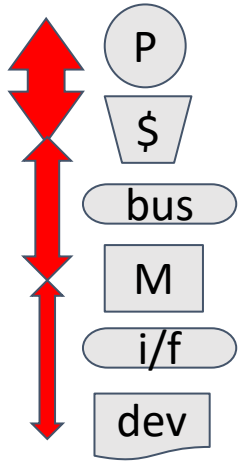


CPU DB: Recording Microprocessor History, Danowitz, Kelley, Mao, Stevenson, Horowitz, Communications of the ACM, 2012

Great Ideas in Computer Architecture

- Abstractions
- Moore's Law
- Principles of Locality/Memory Hierarchy
- **Parallelism**
- Performance Measurement & Improvement
- Dependability via Redundancy

X-Level Parallelism in Computer Architecture



1 CPU

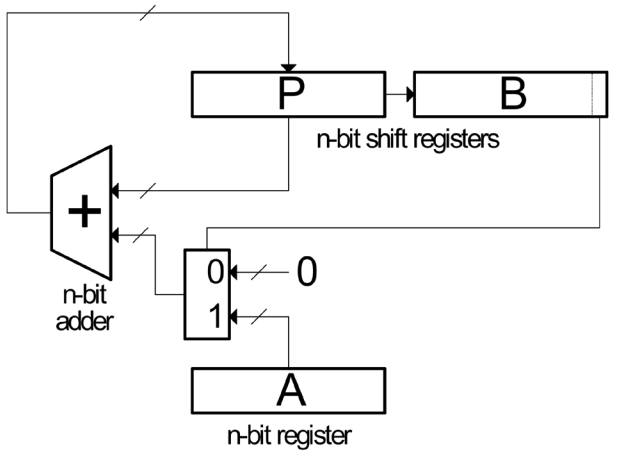
BLP+ILP

Bit/Instrn-Level Parallelism

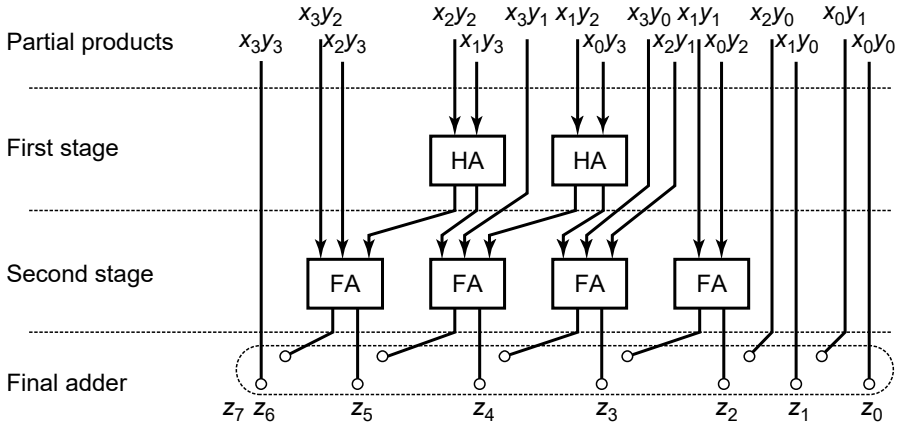
Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

Bit-Level Parallelism (BLP)

- Early computers with few transistors
  - Compute a result in many steps
  - E.g., 1 multiplication partial sum per cycle for “shift-and-add” multiplier
- Bit-Level Parallelism
  - More more bits in parallel
  - Larger words improve BLP:
    - 8b->16b->32b->64b
  - Recent trends:
    - Smaller word size preferred for better efficiency, e.g., DNN inference HW.



- “Shift-and-add” multiplier
- Performance: N cycles for N-bit additions



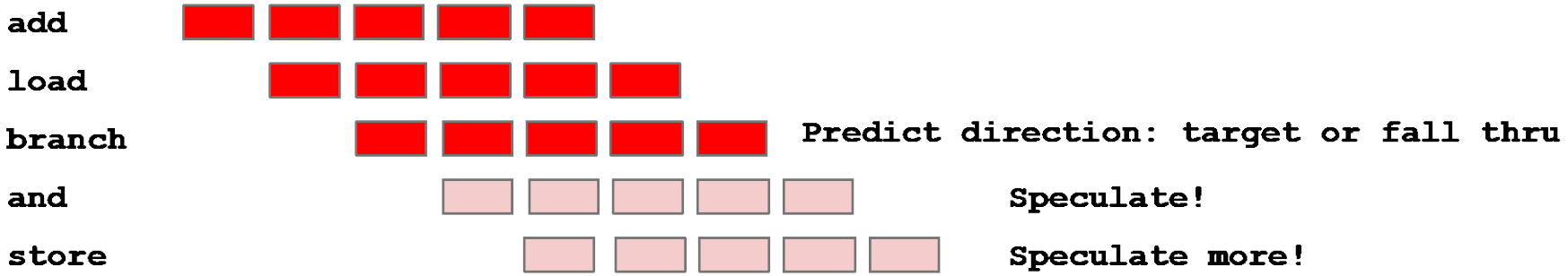
- Parallel Wallace Tree Multiplier

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

Instruction-Level Parallelism (ILP)

- Processors logically do instructions sequentially (time →)

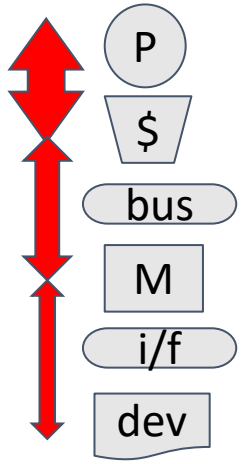
- Though they actually execute instructions in parallel, i.e., instruction-level parallelism.

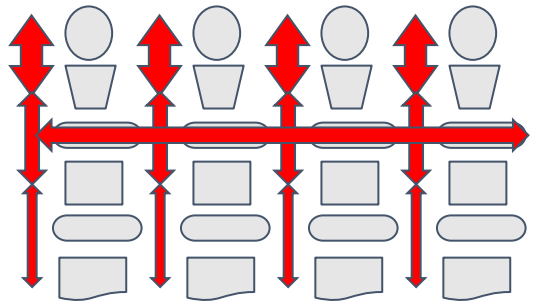


- Completely hides hardware complexity from software.

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

X-Level Parallelism in Computer Architecture





1 CPU

BLP+ILP

Bit/Instrn-Level Parallelism

Multiprocessor

+ TLP

Thread-Level Parallelism

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

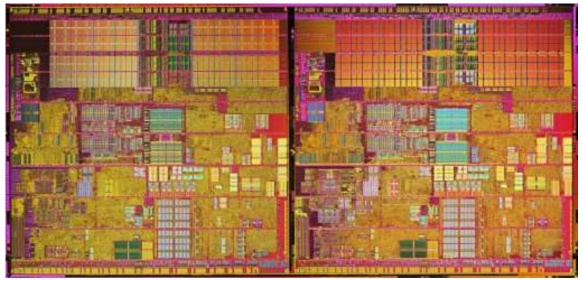
Thread-Level Parallelism (TLP)

- HW: Multiple sequential processors
- SW: Each runs asynchronous thread

- SW must partition work, synchronize, & manage communication
  - E.g., pThreads, OpenMP, MPI

- Less easy for software but:
  - Bifurcation: experts program TLP; others use it.

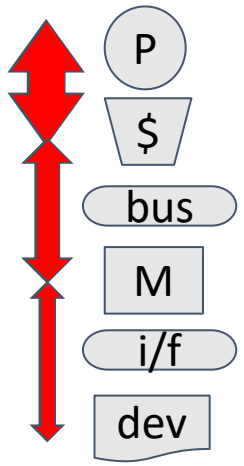
CDC 6600, 1964,
(TLP via multithreaded processor)

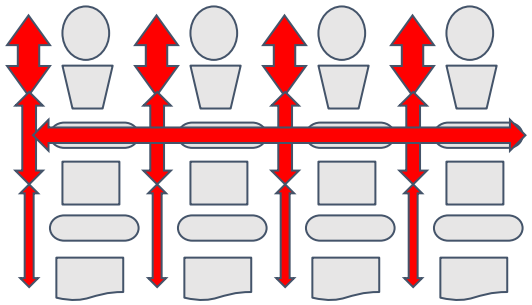


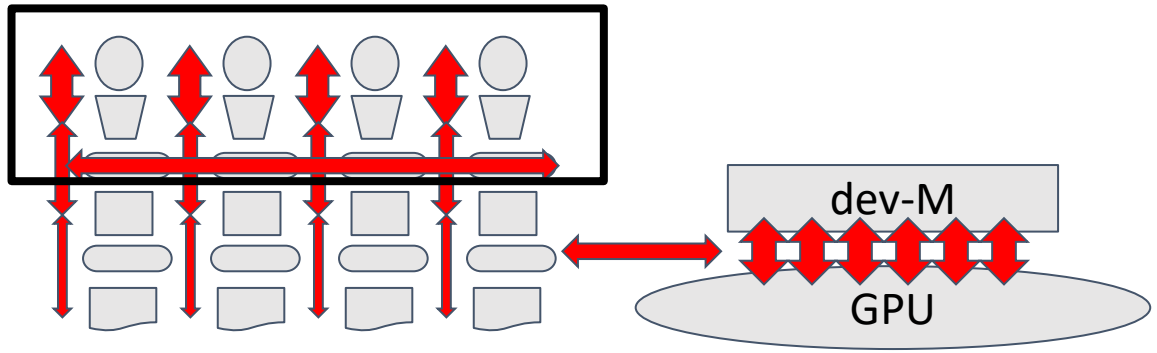
Intel Pentium Pro Extreme Edition,
early 2000s

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

X-Level Parallelism in Computer Architecture



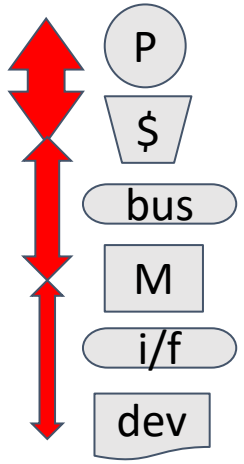


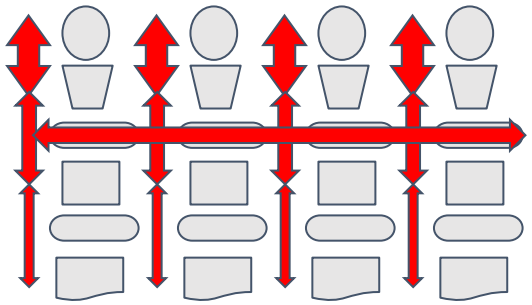


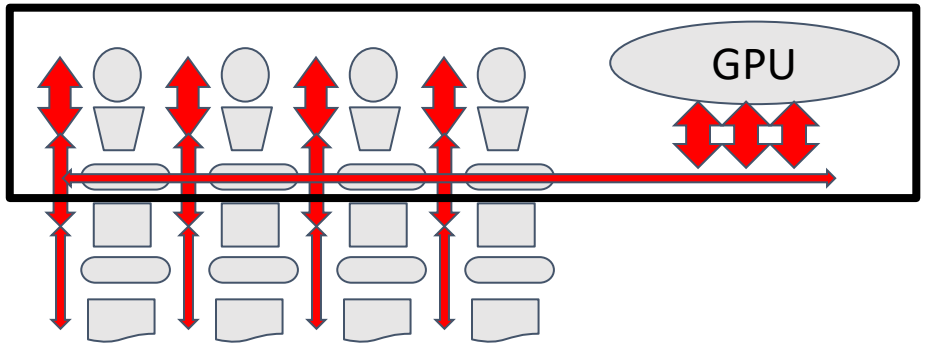
1 CPU	**Multiprocessor**	**Multicore**	+ **Discrete GPU**
BLP+ILP	+ TLP	+ TLP	+ DLP
Bit/Instrn-Level Parallelism	Thread-Level Parallelism	Thread-Level Parallelism	Data-Level Parallelism

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

X-Level Parallelism in Computer Architecture







1 CPU

BLP+ILP

Bit/Instrn-Level Parallelism

Multiprocessor

+ TLP

Thread-Level Parallelism

Multicore + **Integrated GPU**

+ TLP

Thread-Level Parallelism

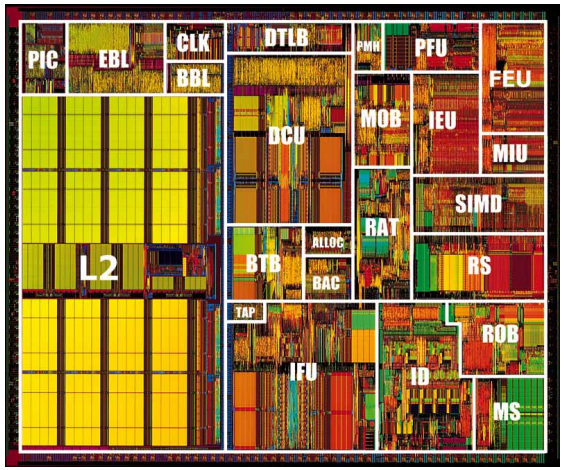
+ DLP

Data-Level Parallelism

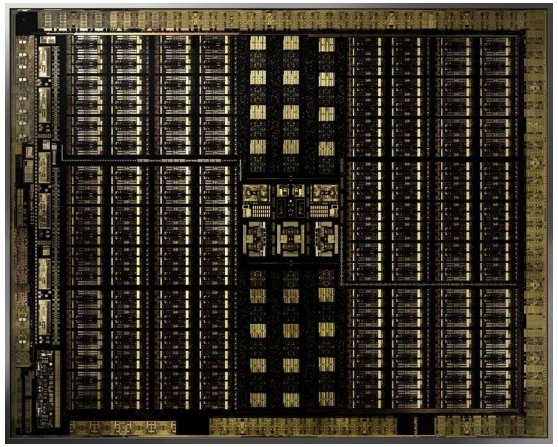
Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

Data-Level Parallelism

- Perform the same operation on different data elements
  - Single-Instruction-Multiple-Data (SIMD)
- DLP success: General-Purpose GPUs
  - Single-Instruction-Multiple-Threads (SIMT)
  - SW (CUDA) & Libraries (math & ML)
  - Bifurcation again:
    - Experts program SIMT (TLP + DLP);
    - Others use it.



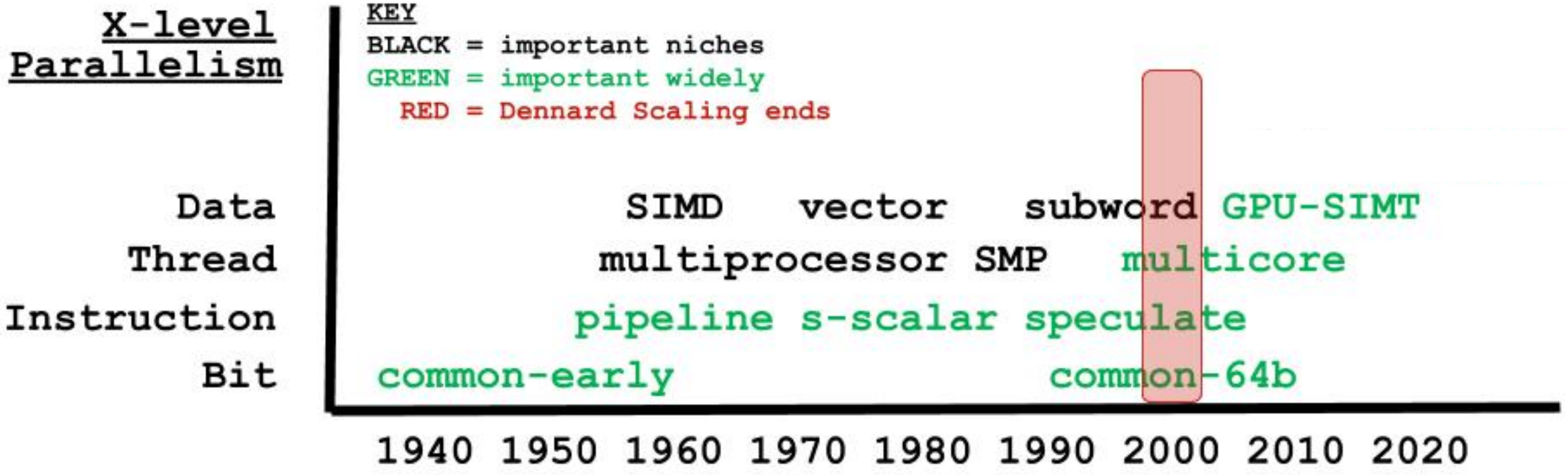
Intel Pentium III



NVIDIA Turing

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

X-Level Parallelism in Computer Architecture



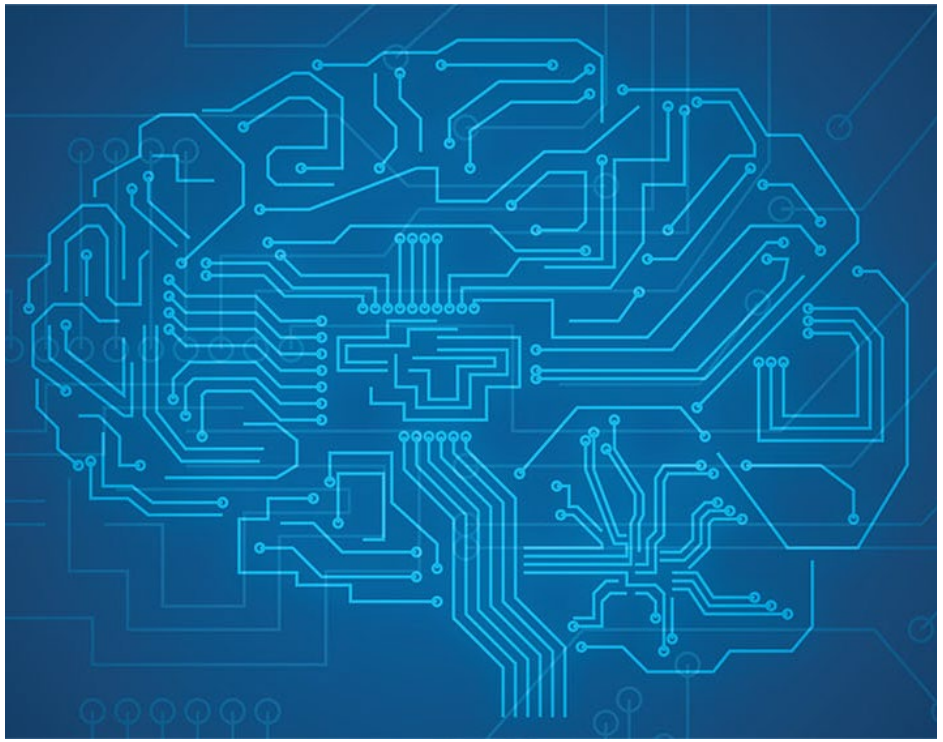
Mark Hill and Vijay Janapa Reddi, Accelerator-Level Parallelism, arXiv'2019

Administrivia

- Course Survey
  - We VALUE your feedback!
  - Tell us your experience!
  - Tell us what worked and what could be improved!

Administrivia

- Project (50%):
  - Proposal (10%)
  - Checkpoint * 2 (5% * 2. Required.)
  - Presentation (10%)
  - Final report (15%)
  - Results (5%)
- Project Checkpoint 1:
  - This week!
  - Please prepare 2 or 3 slides showing your current progress.
  - The slides should include:
    - A diagram representing your hardware/software design, and showing how it fits into your full system.
    - An account of the progress you've made, and whether you've reached your planned Checkpoint 2 goals.
    - A description of any changes you may have made to your plans or your project.

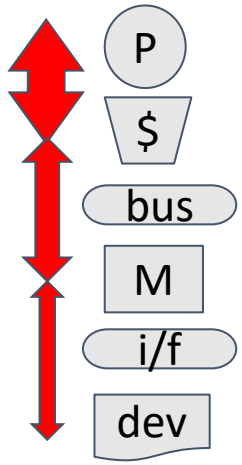


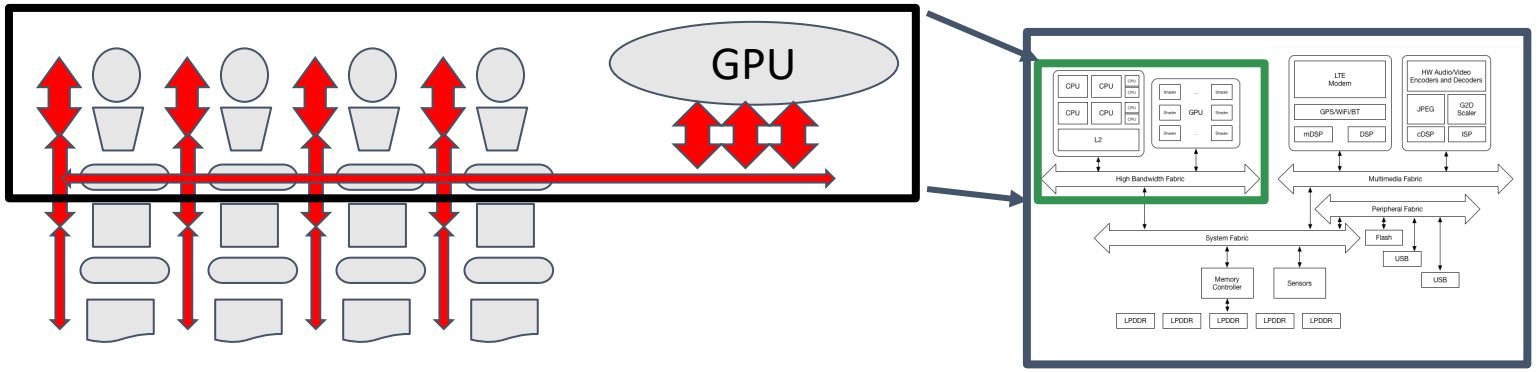
Accelerator-Level Parallelism (ALP)

- X-Level Parallelism
- ALP in Mobile SoC
- Open Challenges

X-Level Parallelism in Computer Architecture

- Accelerator-level parallelism (ALP): the parallelism of different workload components concurrently executing on multiple accelerators.





1 CPU

BLP+ILP

Bit/Instrn-Level Parallelism

Multicore + **Integrated GPU**

+ TLP

Thread-Level Parallelism

+ DLP

Data-Level Parallelism

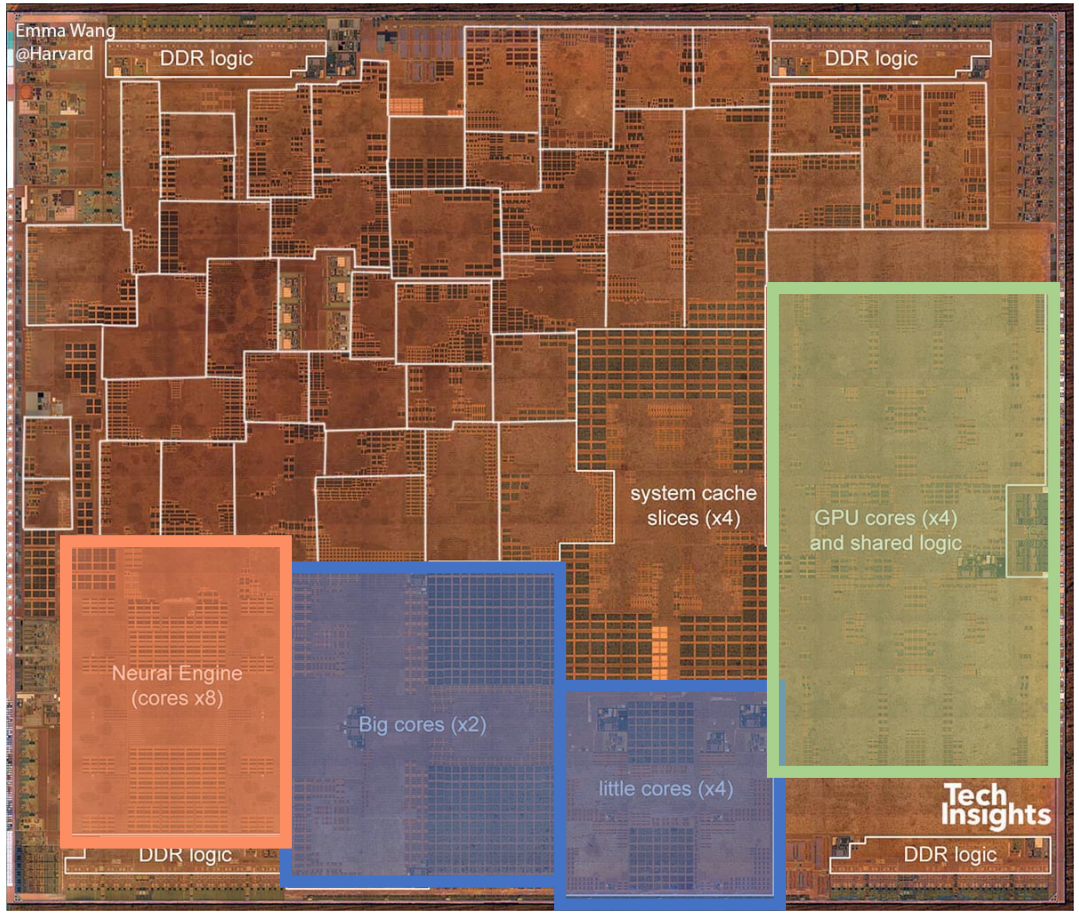
System on a Chip (SoC)

+ ALP

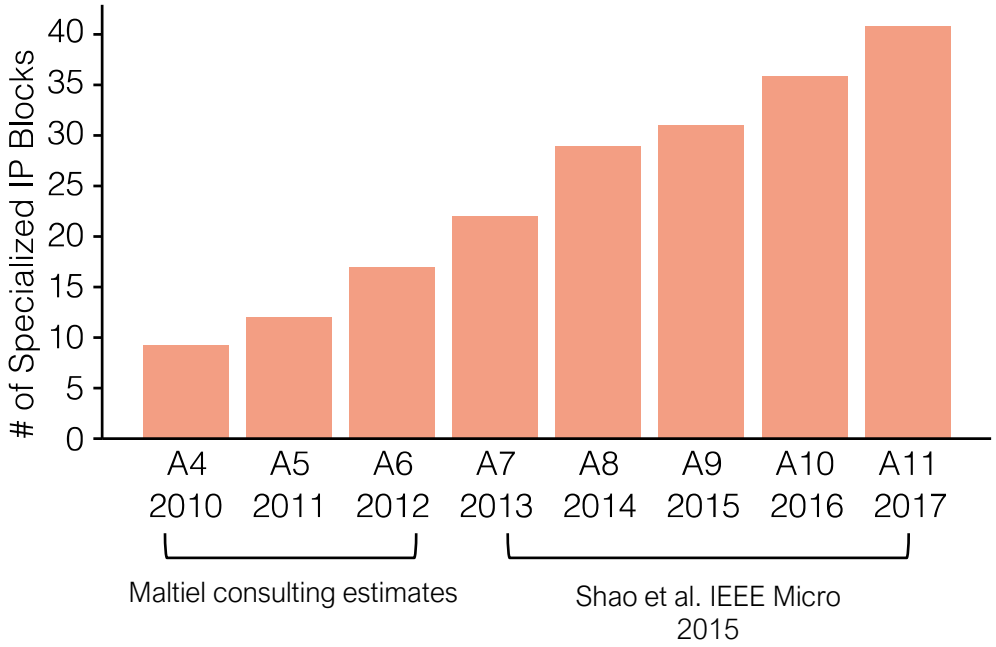
Accelerator-Level Parallelism

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

CPU, GPU, and Accelerators



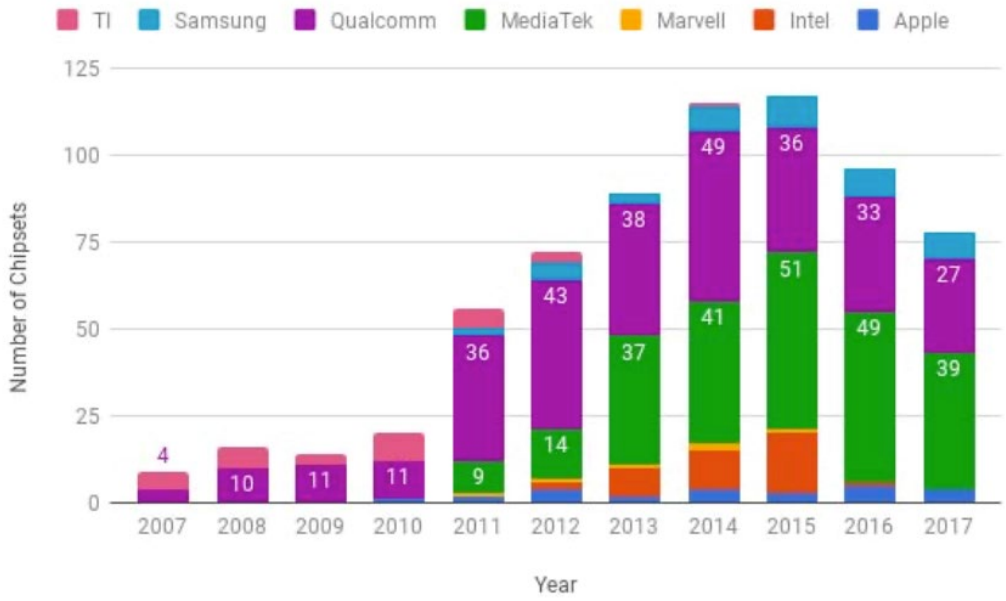
Apple A12 SoC



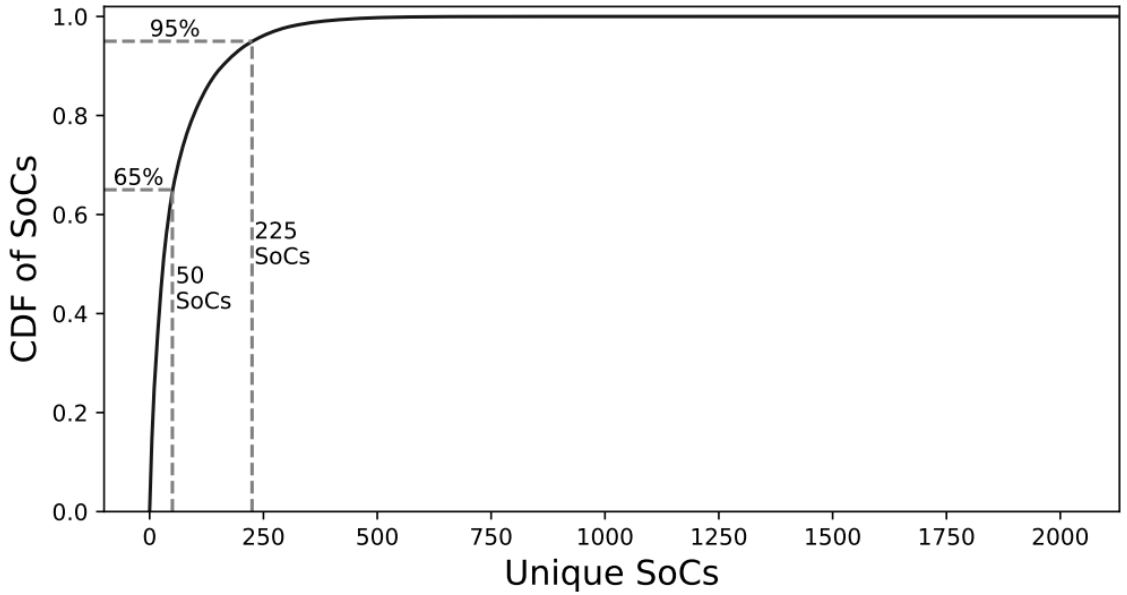
http://vlsiarch.eecs.harvard.edu/research/accelerators/die-photo-analysis/

Mobile SoC Diversity

- 10s-100s of SoCs introduced to the market every years across different SoC vendors.
- No standard mobile SoC to optimize for. Top 50 most common SoCs -> only 65% of the market.



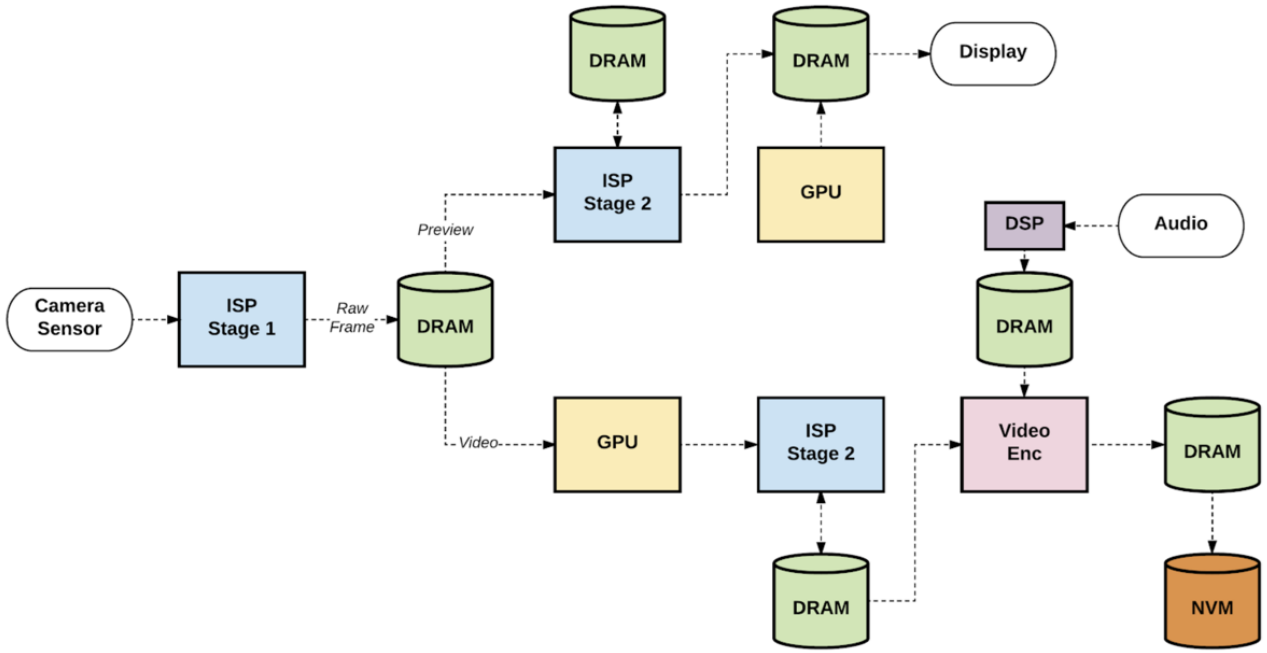
Two Billion Devices and Counting: An Industry Perspective on the State of Mobile Computer Architecture, IEEE Micro'2018



Machine Learning at Facebook: Understanding Inference at the Edge, HPCA'2019

Mobile SoC Usecase

- Mainstream architecture has long focused on general-purpose CPUs and GPUs.
- In an SoC, multiple IP blocks are active at the same time and communicate frequently with each other.
- Example:
  - Recording a 4K video
  - Camera -> ISP
    - “Preview stream” for display
    - “Video stream” for storage
  - DRAM for data sharing



Two Billion Devices and Counting: An Industry Perspective on the State of Mobile Computer Architecture, IEEE Micro’2018

Mobile SoC Usecases

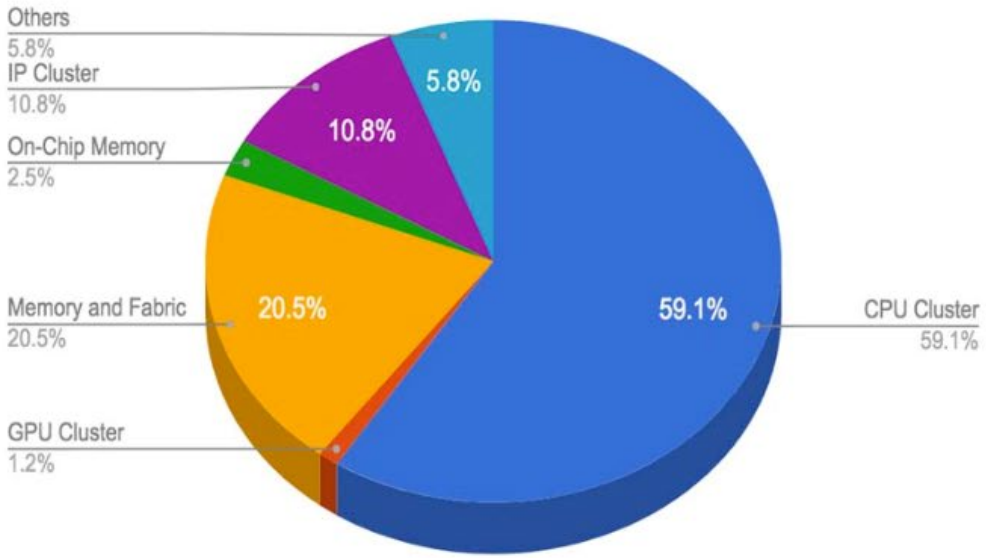
- Multiple accelerators are running concurrently for different usecases.

Accelerators (IPs) ➔ Usecases (rows)	CPUs (AP)	Display	Media Scaler	GPU	Image Signal Proc.	JPEG	Pixel Visual Core	Video Decoder	Video Encoder	**Dozens More**
Photo Enhancing	X	X		X	X	X	X			
Video Capture	X	X		X	X				X	
Video Capture HDR	X	X		X	X				X	
Video Playback	X	X	X	X				X		
Image Recognition	X	X	X	X						

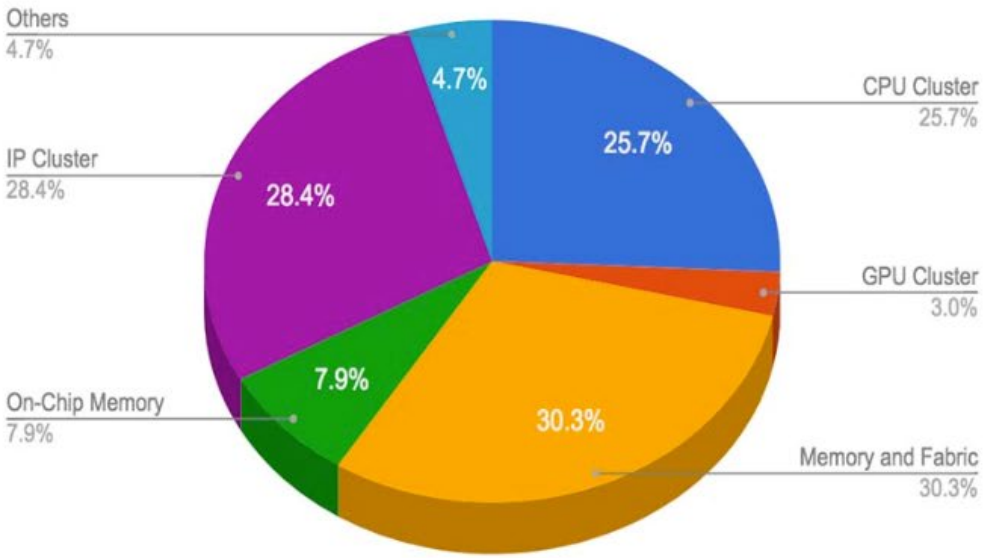
Mark Hill and Vijay Janapa Reddi, Gables: A Roofline Model for Mobile SoCs, HPCA'2019

Mobile SoC Usecases

- Different usecases have different power profiles.

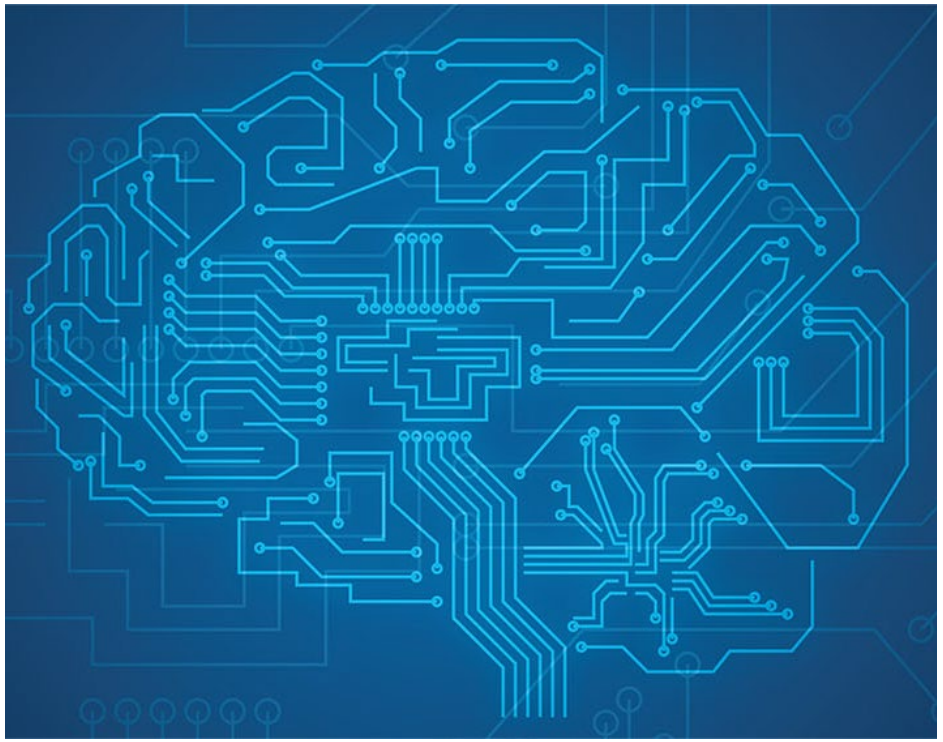


Web browser scrolling
(total 651.77 mW)



4K video recording
(total 3,473.55mW)

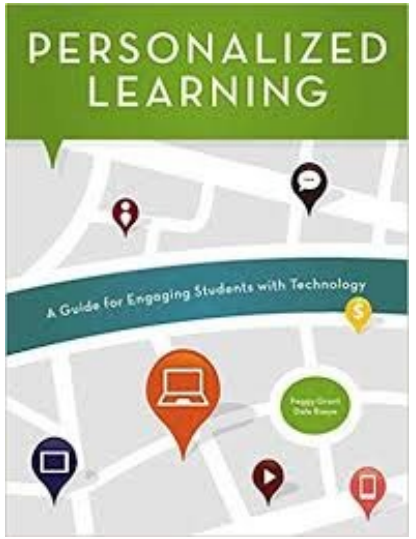
Two Billion Devices and Counting: An Industry Perspective on the State of Mobile Computer Architecture, IEEE Micro'2018

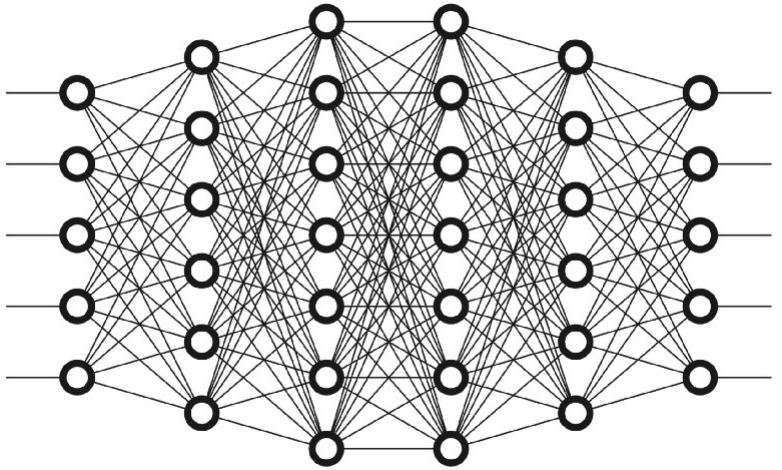


Accelerator-Level Parallelism (ALP)

- X-Level Parallelism
- ALP in Mobile SoC
- Open Challenges

Future Apps Demand Much More Computing

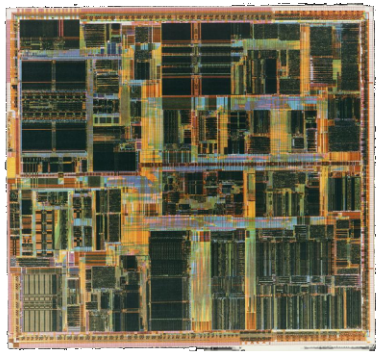




Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

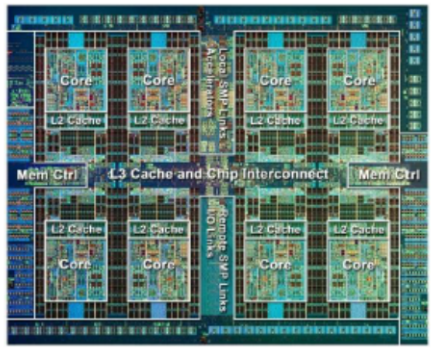
X-level Parallel Hardware + Software (Model)

Single-threaded languages, compilers, runtimes, etc. **hides** parallelism



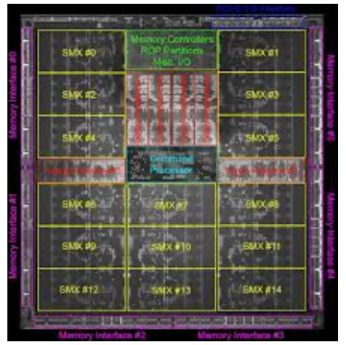
Intel Pentium Pro
BLP+ILP

Software **abstracts** TLP, e.g., pThreads, OpenMP, & MPI. Also **hidden** in cloud, etc.



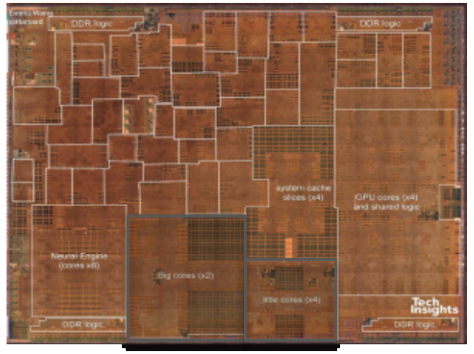
IBM Power 7
BLP+ILP+TLP

Software **abstracts** TLP+DLP, e.g., CUDA, OpenCL, & graphics OpenGL Also **hidden** in cloud, etc.



Nvidia GK110
BLP+TLP+DLP

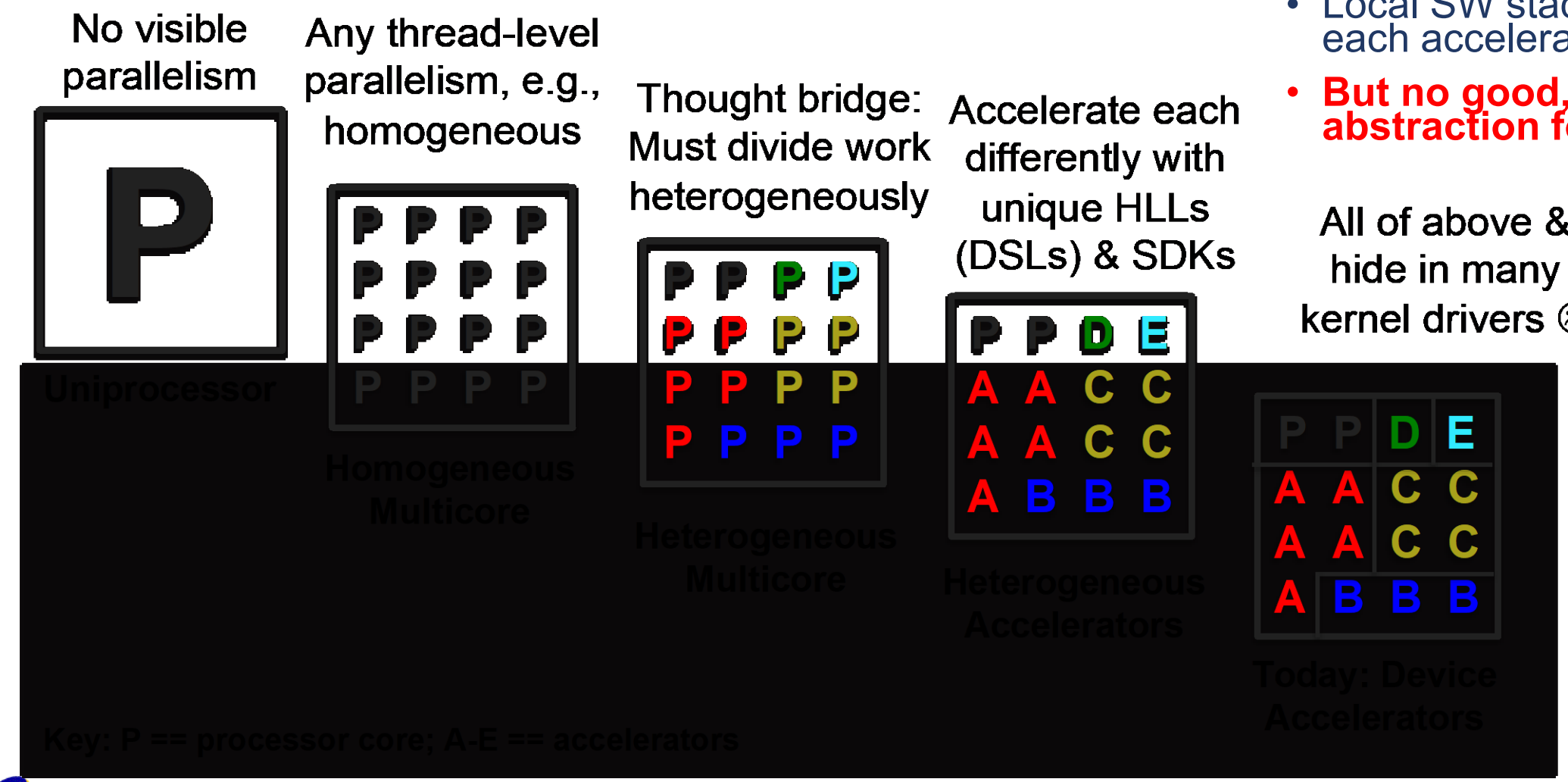
Local software stack **abstracts** each accelerator. **But no good, general software abstraction for SoC ALP!**



Apple A12
BLP+ILP+TLP+DLP+ALP

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

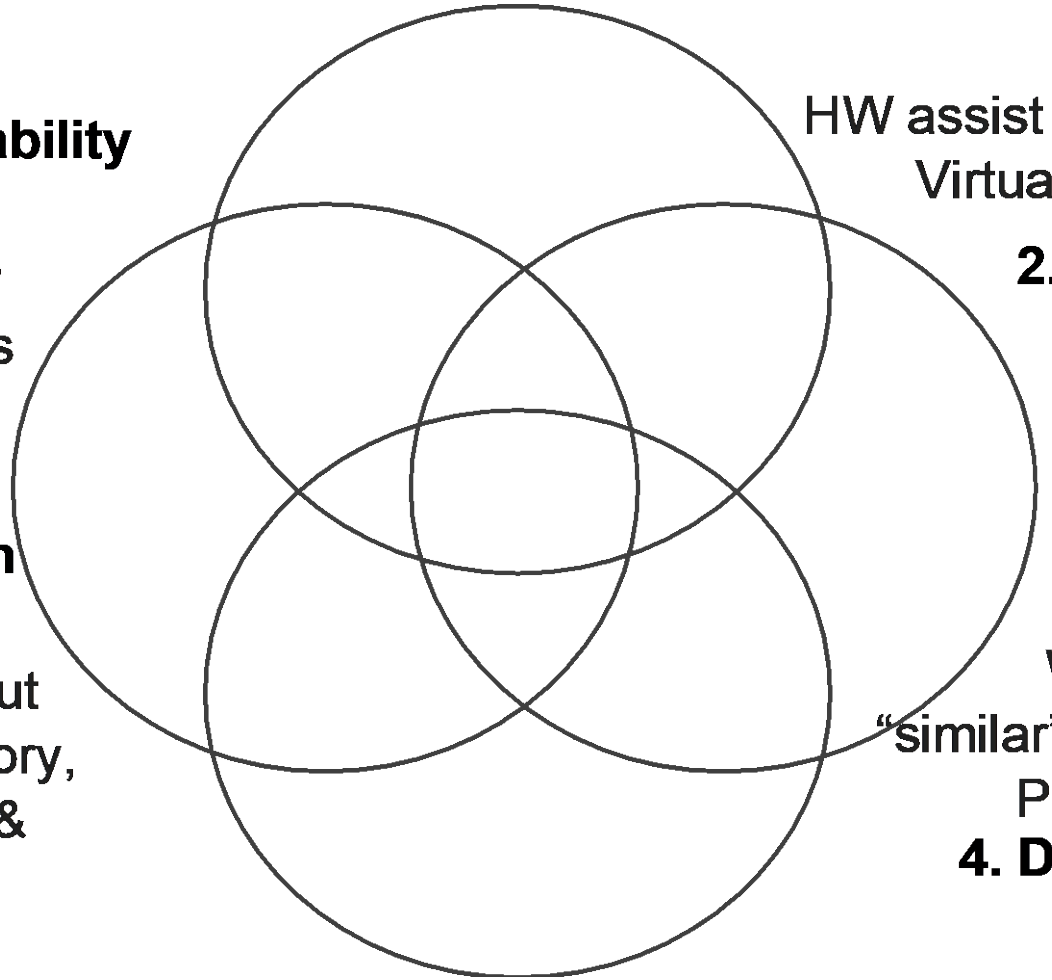
ALP Challenges & Opportunities



- Local SW stack **abstracts** each accelerator.
- **But no good, general SW abstraction for SoC ALP!**

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

ALP Challenges & Opportunities



1. **Programmability**

Whither global model/runtime? DAG of streams for SoCs?

3. Communication

How should SW stack reason about local/global memory, caches, queues, & scratchpads?

HW assist for scheduling? Virtualize & partition?

2. Concurrency

When combine "similar" accelerators? Power vs. area?

4. Design Space

Mark Hill, Accelerator-Level Parallelism, Presentation, 2020

Review

- Core computation in DNN
- Execution order of the core computation
- Hardware realization of the core computation
- Mapping DNNs to hardware
- Data transfer mechanisms across storage hierarchy
- Sparsity in DNNs
- Codesign example
- Other Operators and Near-Data Processing
- Training Kernels
- This lecture: Accelerator-Level Parallelism