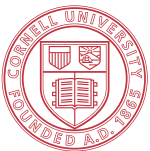




ECE 6775
High-Level Digital Design Automation
Fall 2023

Hardware Specialization



Cornell University



Announcements

- ▶ First reading assignment
 - A. Boutros and V. Betz, “[FPGA Architecture: Principles and Progression](#)”, IEEE CAS-M 2021
 - **Complete reading before Thursday 9/7**
- ▶ Hands-on tutorial on HLS next Tuesday
 - Bring your laptop

Agenda

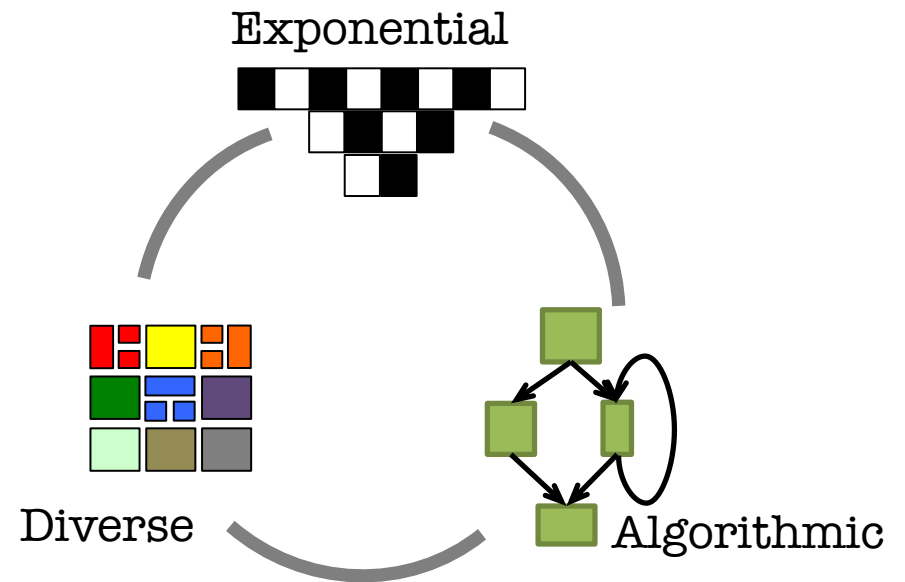
- ▶ Motivation for hardware specialization
 - Key driving forces from applications and technology
 - Main sources of inefficiency in general-purpose computing
- ▶ Essential specialization techniques
- ▶ Roofline-based performance modeling

Recap: Our Interpretation of E-D-A

Exponential
in complexity (or **Extreme** scale)

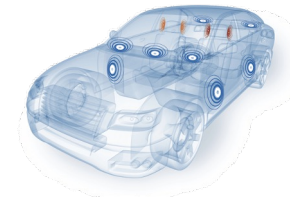
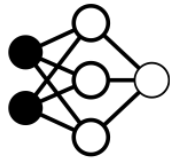
Diverse
increasing system heterogeneity

Algorithmic
intrinsically computational

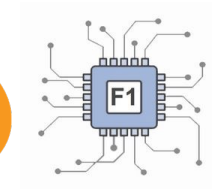
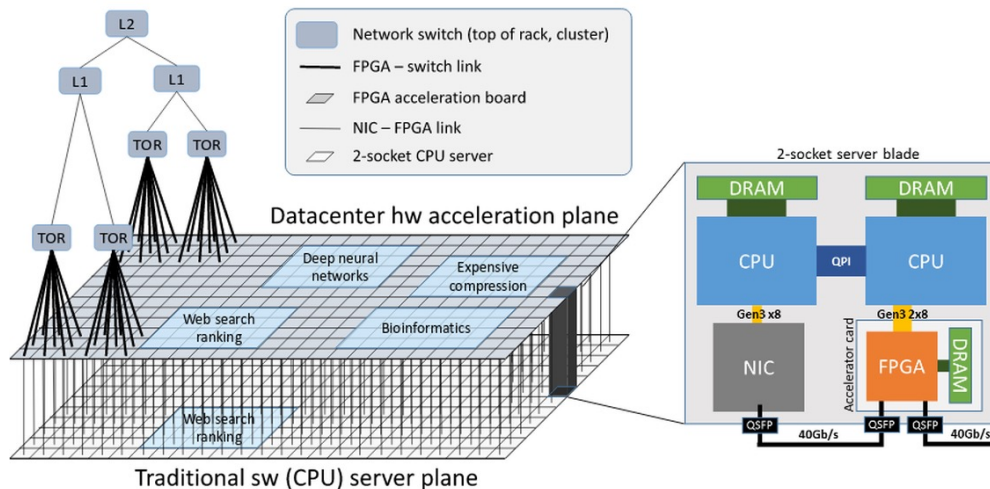


A Golden Age of Hardware Specialization

- ▶ **Higher demand** on efficient compute acceleration, esp. for machine learning (ML) workloads

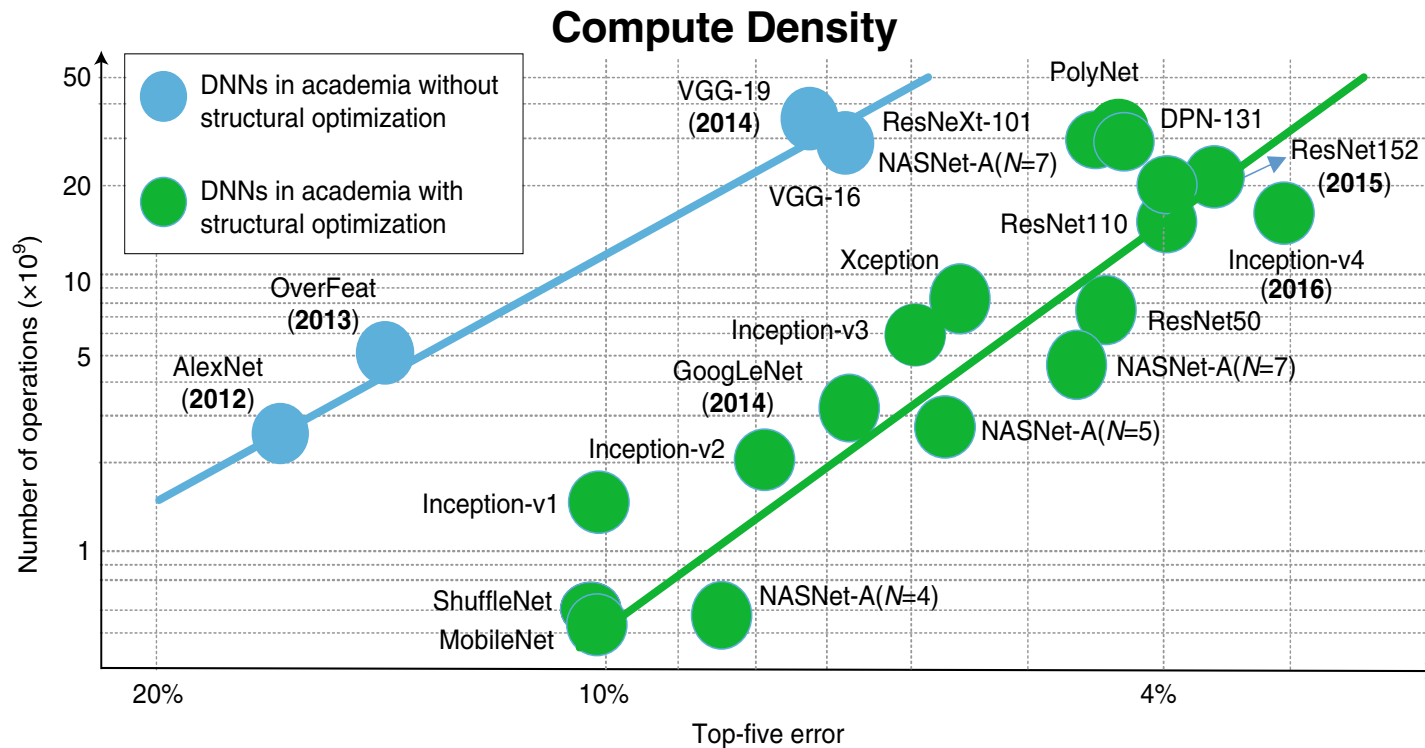


- ▶ **Lower barrier** with open-source hardware & cloud FPGAs coming of age



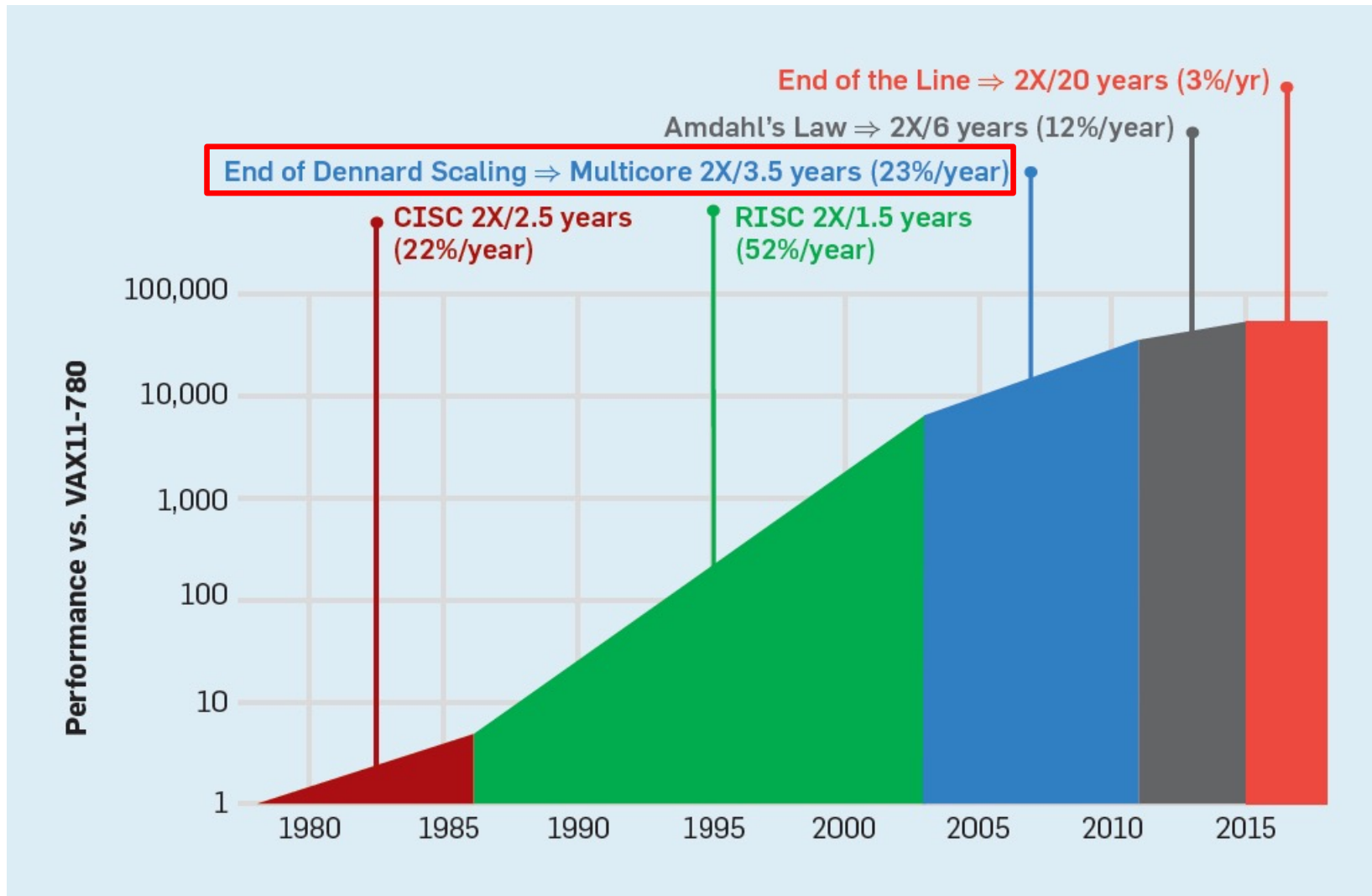
Modern ML Models are Computationally Demanding

- ▶ Deep neural networks (DNNs) require enormous amount of compute
 - Consider ResNet50, a 70-layer model that performs 7.7 billion operations to classify an image (a relatively small model by today's standards)



X. Xu, et al. **Scaling for Edge Inference of Neural Networks**. *Nature Electronics*, 2018.

On Crash Course with the End of “Cheap” Technology Scaling



John Hennessy, David Patterson, “A New Golden Age for Computer Architecture”, CACM 2019.

End of Dennard Scaling and its Implications

- ▶ Classical Dennard scaling
 - Frequency increases at constant power profiles
 - Performance improves “for free”!

- ▶ Leakage limited scaling
 - V_{th} scaling halted due to exponentially increasing leakage power
 - V_{DD} scaling nearly stopped as well to maintain performance

- ▶ Implication: “Dark silicon”?
 - Power limits restrict how much of the chip can be activated simultaneously
 - No longer 100% without more power

Dennard scaling

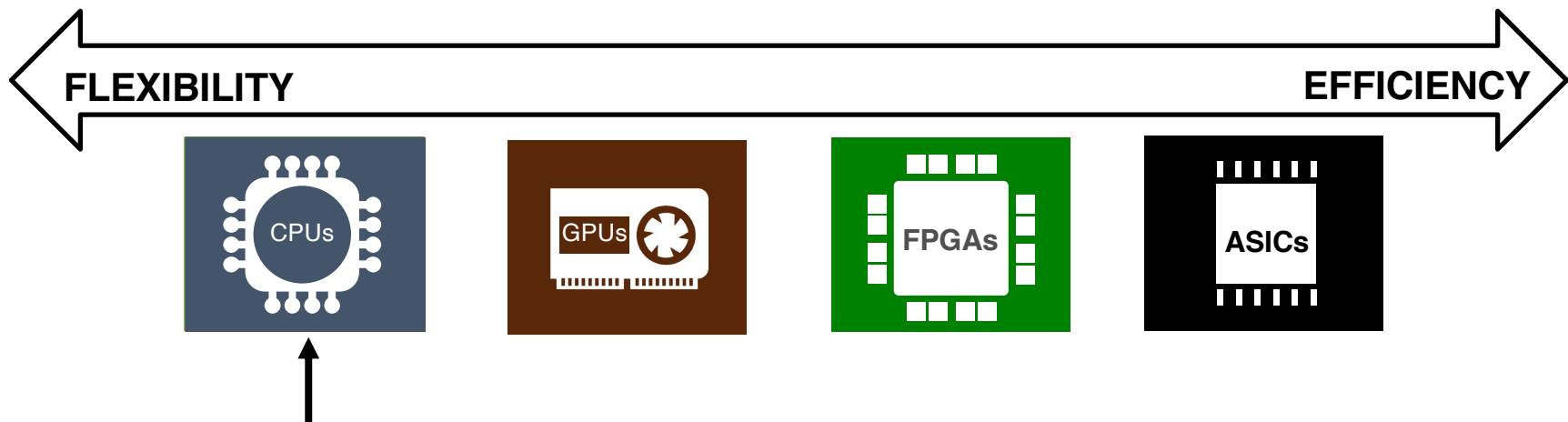
Transistor (trans.) #	S^2
Capacitance / trans.	$1/S$
Voltage (V_{dd})	$1/S$
Frequency	S
Total power	1

Leakage limited scaling

Transistor (trans.) #	S^2
Capacitance / trans.	$1/S$
Voltage (V_{dd})	~ 1
Frequency	~ 1
Total power	S

Note: dynamic power = CV^2F
 (assuming switching activity factor is 1)

Trade-off Between Compute Efficiency and Flexibility



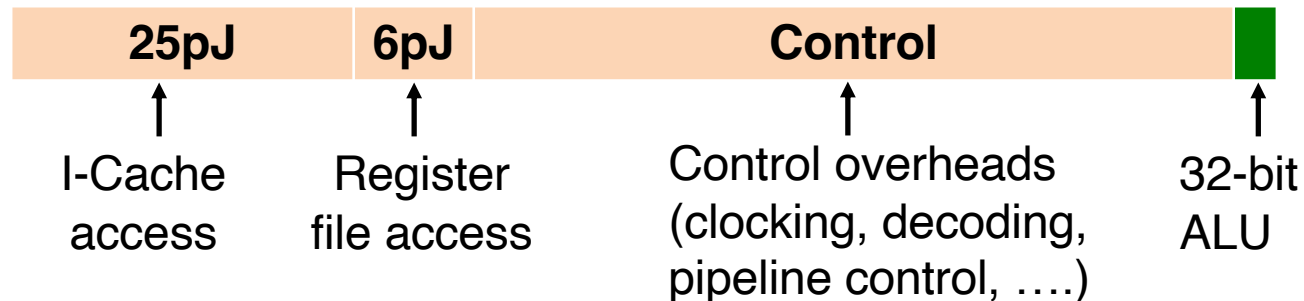
General-purpose CPUs
are less energy efficient

WHY?

Rough Energy Breakdown for an Instruction

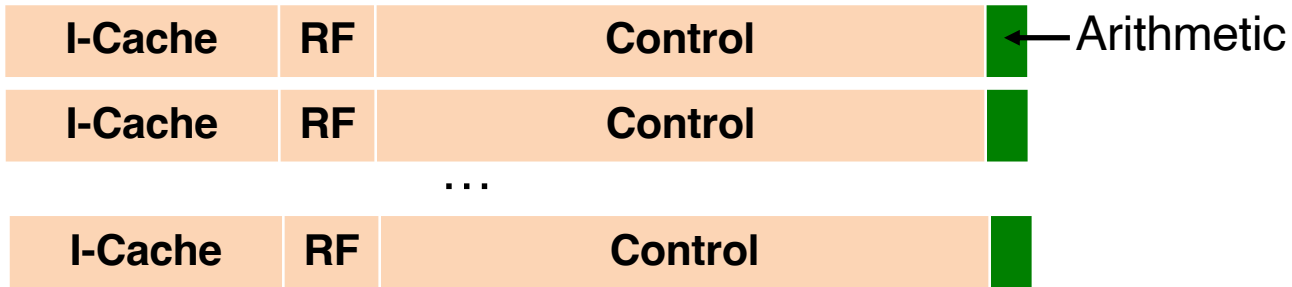
Estimated energy costs for various integer, floating-point (FP), and memory operations on CPU with a 45nm target node at 0.9V

Integer		FP		Memory	
Add		FAdd		Cache	(64bit)
8 bit	0.03pJ	16 bit	0.4pJ	8KB	10pJ
32 bit	0.1pJ	32 bit	0.9pJ	32KB	20pJ
Mult		FMult		1MB	100pJ
8 bit	0.2pJ	16 bit	1.1pJ	DRAM	1.3-2.6nJ
32 bit	3.1pJ	32 bit	3.7pJ		



Reducing Compute Energy Overhead

A sequence of energy-inefficient instructions



Single instruction multiple Data (SIMD): tens of operations per instruction



Further specialization (what we achieve using accelerators)



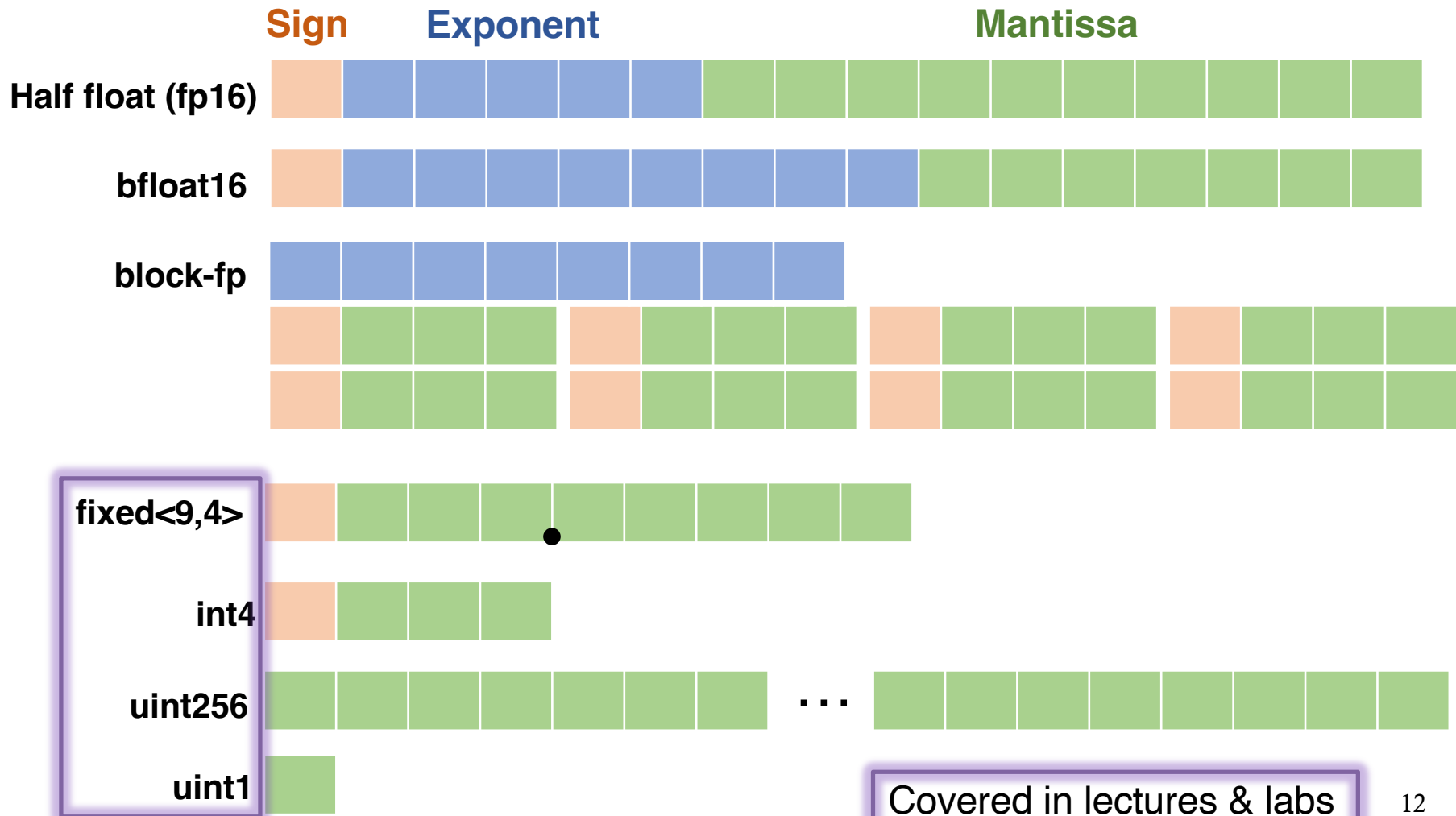
Additional Energy Savings from Specialization

- ▶ **Customized data types**
 - Exploit accuracy-efficiency trade-off to simplify arithmetic operations and reduce memory accesses
- ▶ **Customized memory hierarchy**
 - Exploit regular memory access patterns to minimize energy per memory read/write
- ▶ **Customized communication architecture**
 - Exploit data movement patterns to optimize the structure/topology of on-chip interconnection network

These techniques combined can lead to another 10-100X energy efficiency improvement over general-purpose processors

Customized Data Types

- Using custom numeric types tailored for a given application/domain improves performance & efficiency



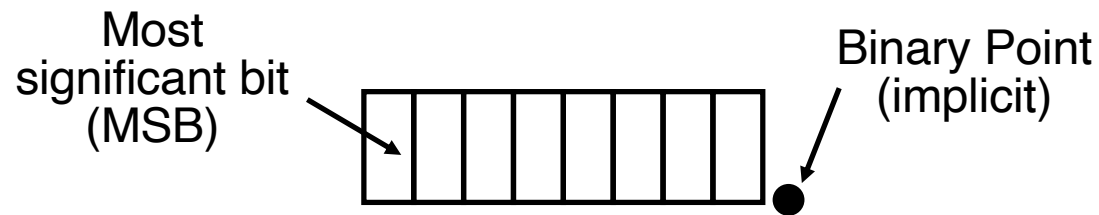
Binary Representation – Positional Encoding

Unsigned number

- ▶ MSB has a place value (weight) of 2^{n-1}

Two's complement

- ▶ MSB weight = -2^{n-1}

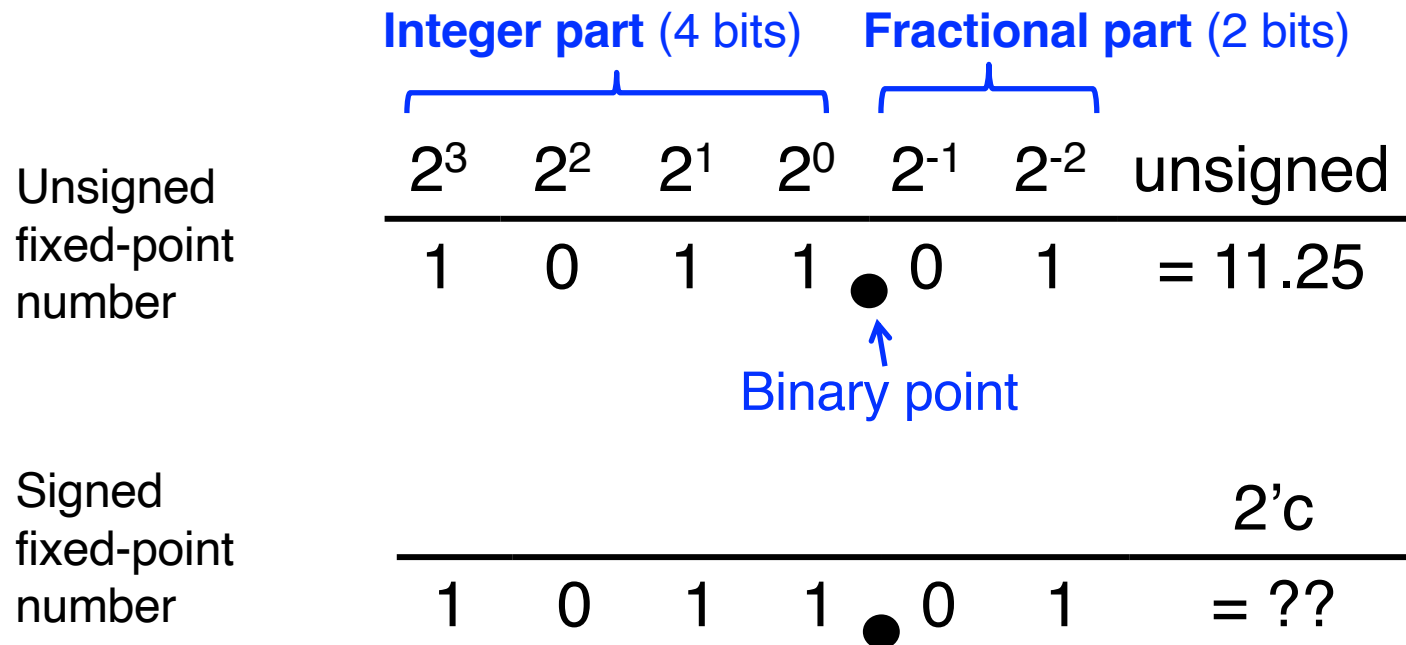


2^3	2^2	2^1	2^0	unsigned
1	0	1	1	= 11

-2^3	2^2	2^1	2^0	2'c
1	0	1	1	= -5

Fixed-Point Representation of Fractional Numbers

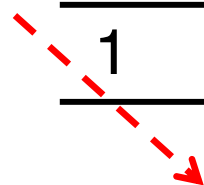
- ▶ The positional binary encoding can also represent fractional values, by using a **fixed** position of the binary point and place values with negative exponents
 - (-) Less convenient to use in software, compared to floating point
 - (+) Much more efficient in hardware



Overflow and Underflow

- ▶ **Overflow** occurs when a number is larger than the largest number that can be represented using a given number of bits

2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	unsigned
1	0	1	1	0	1	= 11.25
0	1	1	0	1		= 3.25



Overflow with 3 integer bits

- ▶ **Underflow** occurs when a number is smaller than the smallest number that can be represented

Handling Overflow

- ▶ One common & efficient way of handling overflow is to drop the MSB(s) of the original number
 - This is commonly called **wrapping**

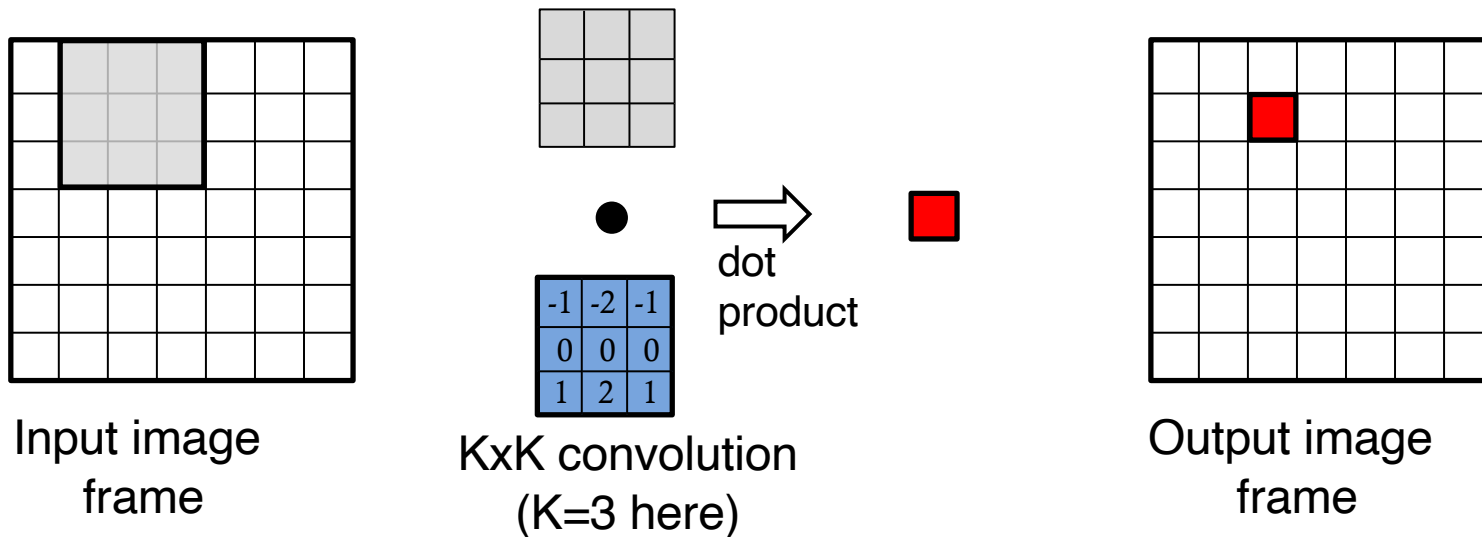
-2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^c
1	0	1	1	0	1	= -4.75
Dropping MSB when integer width is reduced						
						2^c
0	1	1	0	1		= ??

Wrapping can cause a negative number to become positive, or a positive to negative

Custom Memory Hierarchy: A Case Study on Convolution

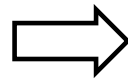
- **Convolution** is pervasive in image/video processing and ML – performed over overlapping windows (aka stencils)

$$(Img \otimes f)_{\left[n+\frac{k-1}{2}, m+\frac{k-1}{2}\right]} = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} Img_{[n+i][m+j]} \cdot f_{[i,j]}$$



Example Application: Edge Detection

- ▶ Identifies discontinuities in an image where brightness (or image intensity) changes sharply
 - Very useful for feature extractions in computer vision



Sobel operator
 $G = (G_X, G_Y)$

$$G_X = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

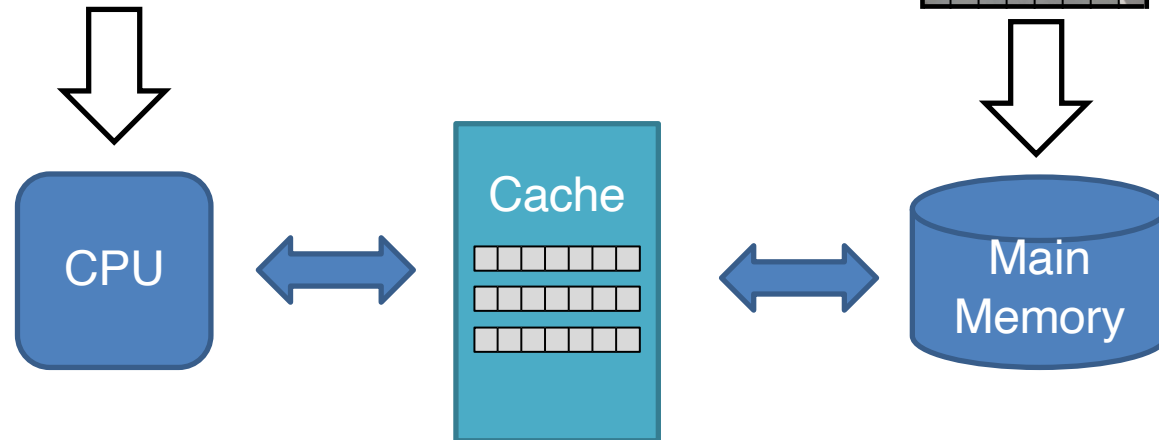
$$G_Y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



Figures: Pilho Kim, GaTech

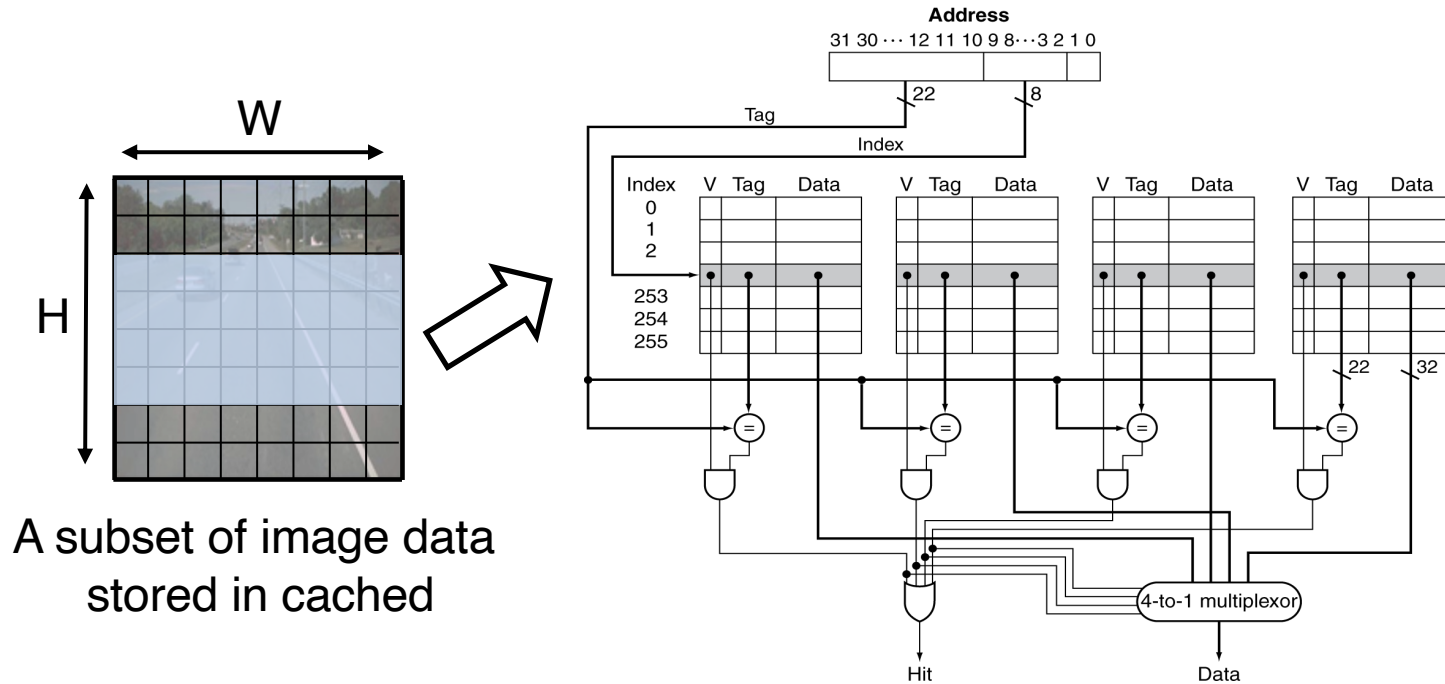
CPU Implementation of a 3x3 Convolution

```
for (r = 1; r < H; r++)  
  for (c = 1; c < W; c++)  
    for (i = 0; i < K; i++)  
      for (j = 0; j < K; j++)  
        out[r][c] += img[r+i-1][c+j-1] * f[i][j];
```



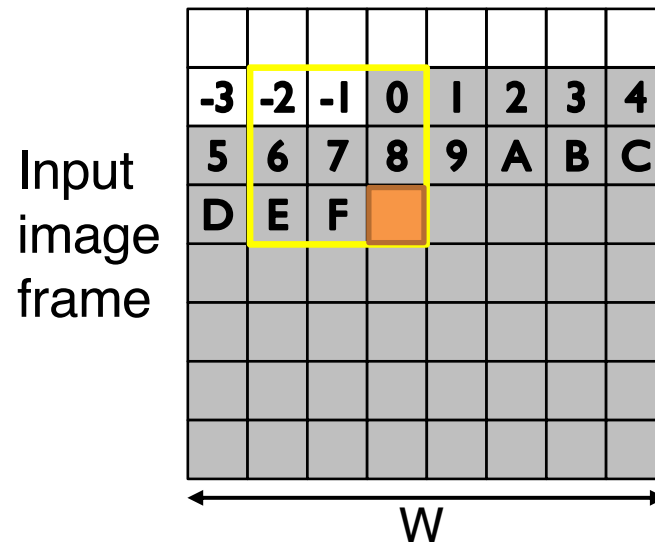
General-Purpose Cache for Convolution

- ▶ A general-purpose cache can effectively reduce external memory accesses
 - but is expensive in cost and incurs nontrivial energy overhead



Line Buffer: Customized “Cache” for Convolution

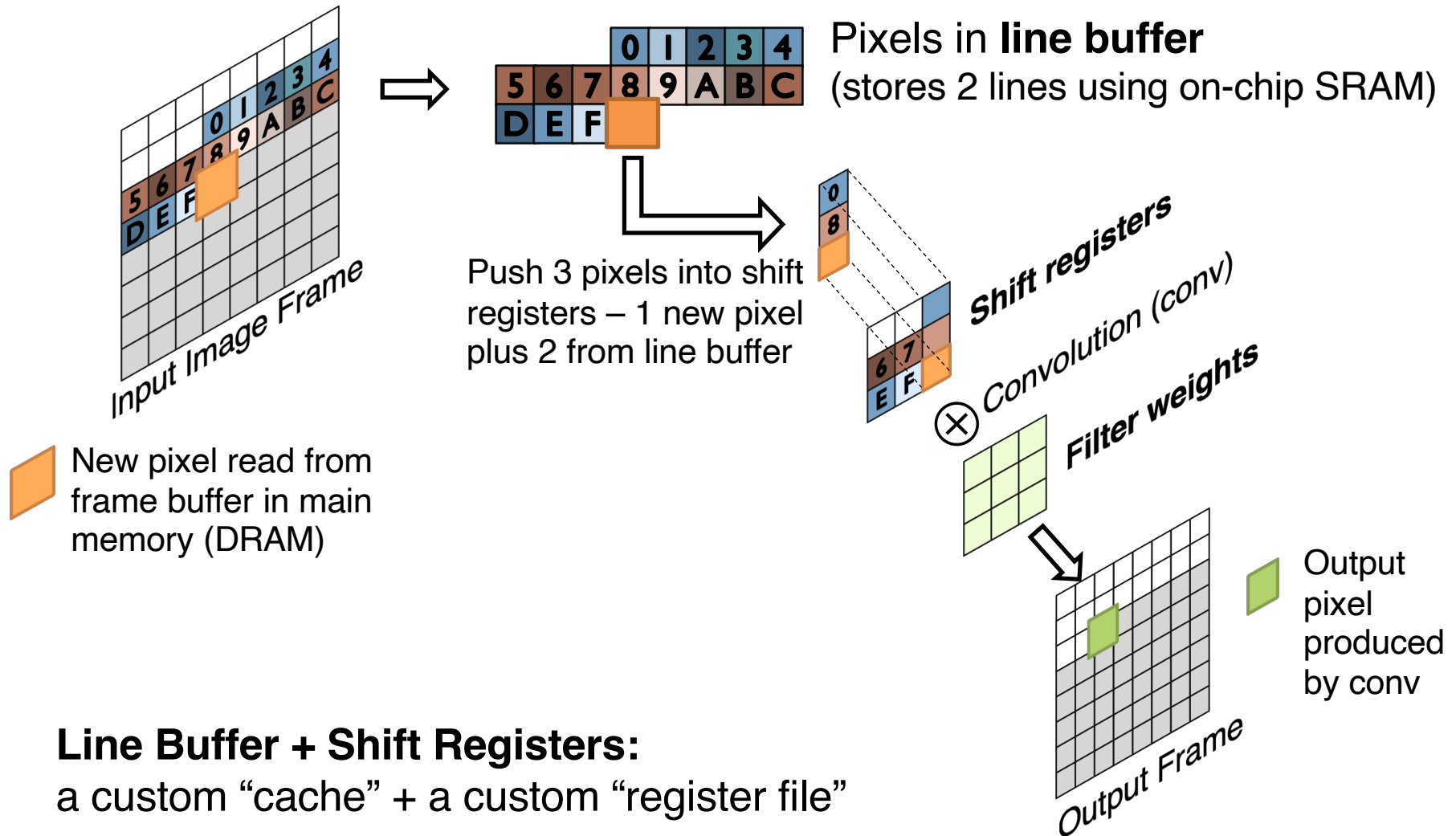
- ▶ “Cache” the input pixels in a line buffer
 - Each time we move the $K \times K$ window (in yellow) to the right and push in a new pixel (in orange) to the specialized “cache”



Line Buffer: simple addressing, simple replacement policy (first in, first out)



A More Complete Picture: Customized On-Chip Memory Hierarchy



Line Buffer + Shift Registers:
a custom “cache” + a custom “register file”

Custom Communication Architecture

Example: Systolic Arrays

- ▶ An array of processing elements (PEs) that process data in a systolic manner using nearest-neighbor communication

Systolic Arrays (for VLSI)

H. T. Kung[†] and Charles E. Leiserson[†]

*And now I see with eye serene
The very pulse of the machine.
--William Wordsworth*

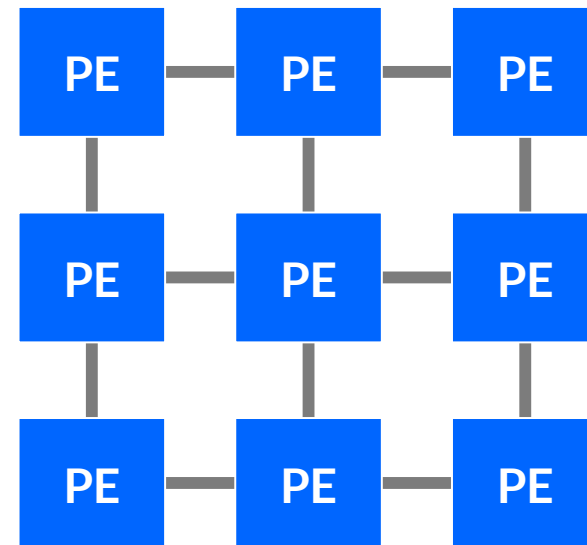
Abstract

A **systolic** system is a network of processors which rhythmically compute and pass data through the system. Physiologists use the word "systole" to refer to the rhythmically recurrent contraction of the heart and arteries which pulses blood through the body. In a **systolic** computing system, the function of a processor is analogous to that of the heart. Every processor regularly pumps data in and out, each time performing some short computation, so that a regular flow of data is kept up in the network.

Many basic matrix computations can be pipelined elegantly and efficiently on **systolic** networks having an array structure. As an example, hexagonally connected processors can optimally perform matrix multiplication. Surprisingly, a similar **systolic** array can compute the LU-decomposition of a matrix. These **systolic** arrays enjoy simple and regular communication paths, and almost all processors used in the networks are identical. As a result, special purpose hardware devices based on **systolic** arrays can be built inexpensively using the **VLSI** technology.

1. Introduction

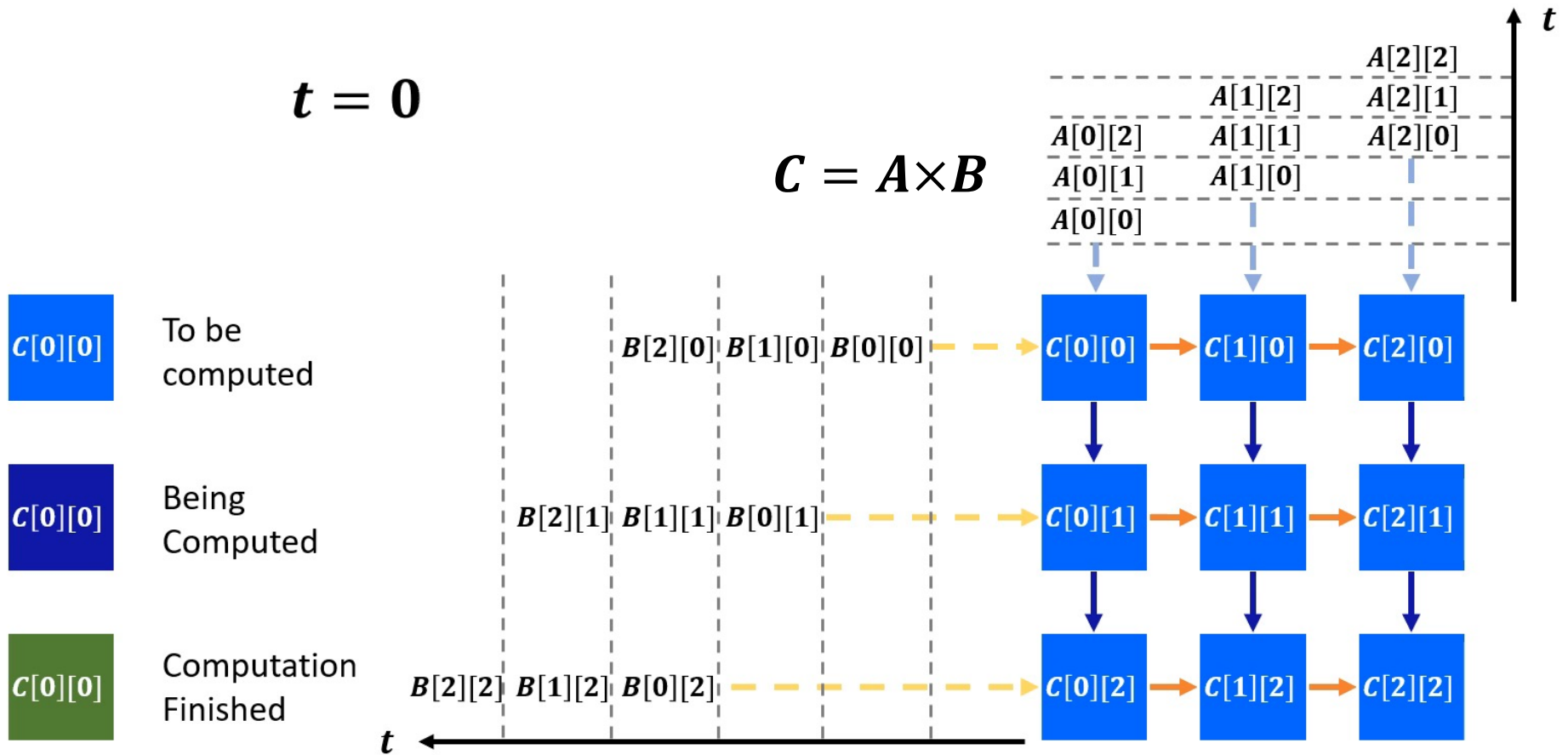
Developments in microelectronics have revolutionized computer design. Integrated circuit technology has increased the number and complexity of components that can fit on a chip or a printed circuit board. Component density has been doubling every one-to-two years and already, a multiplier can fit on a very large scale integrated



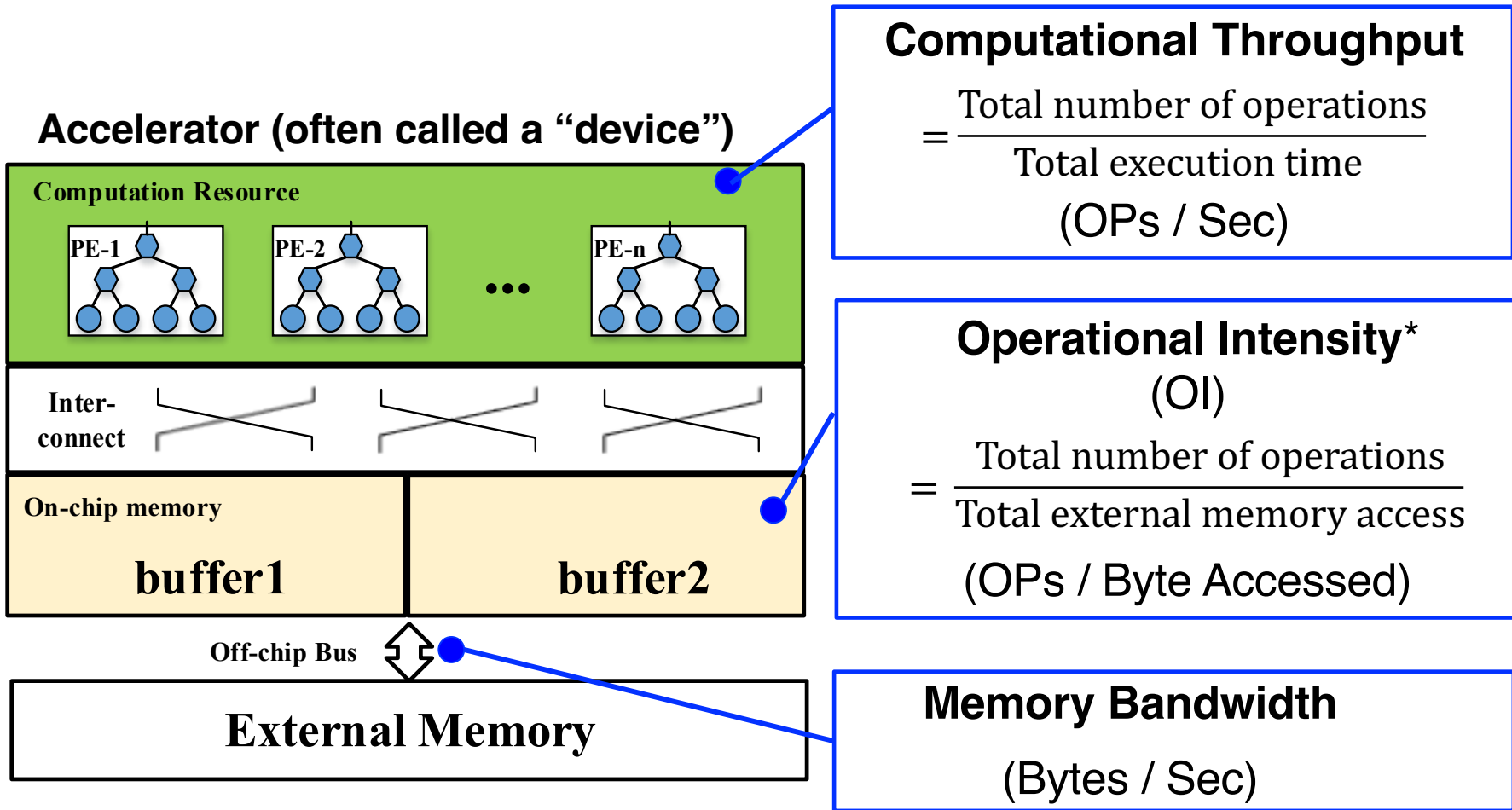
- + Simple & regular design
- + Massive parallelism
- + Short nearest-neighbor interconnection
- + Balancing compute with I/O

Matrix Multiplication on a Systolic Array

- ▶ An array of processing elements that process data in a systolic manner

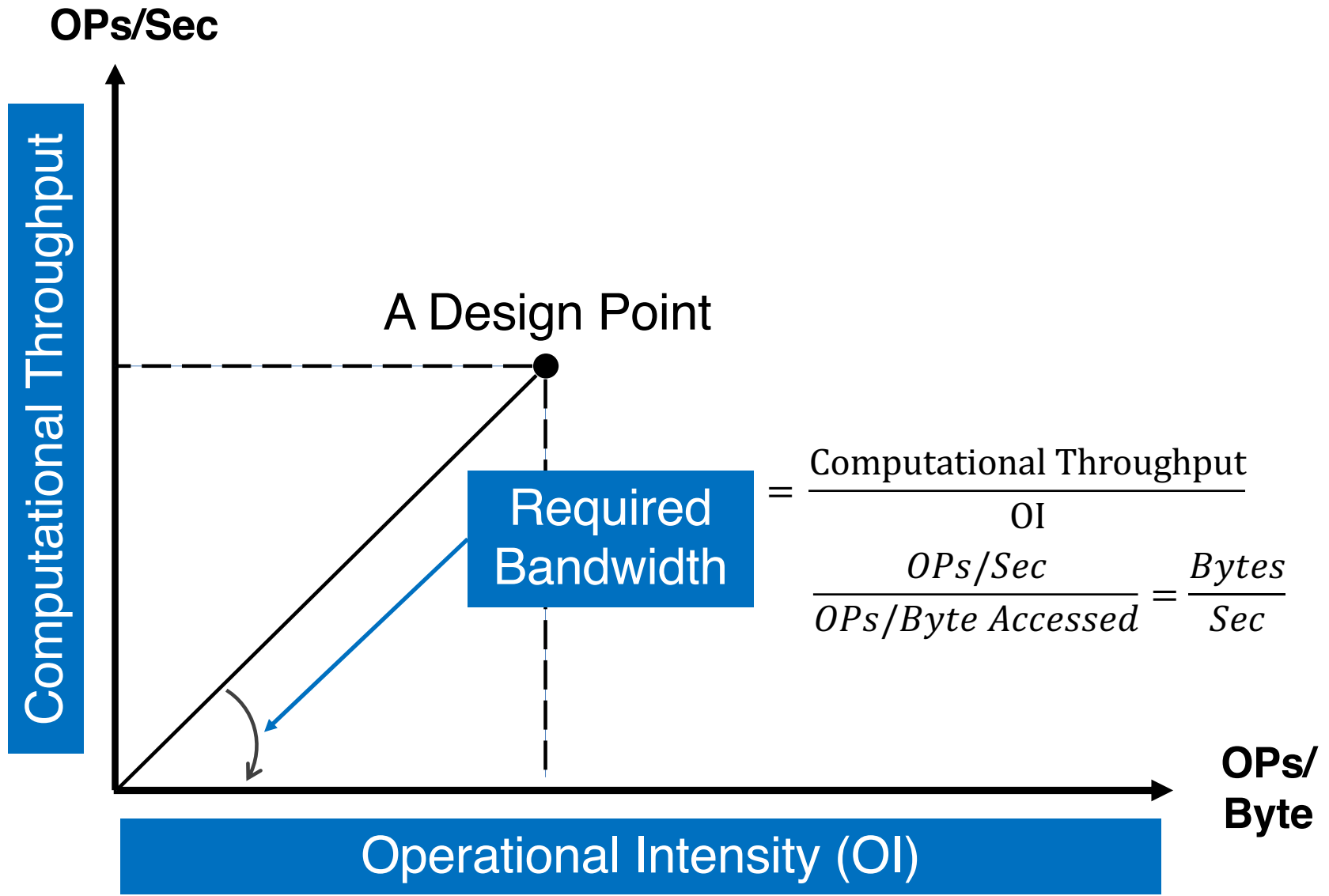


Accelerator Performance Modeling



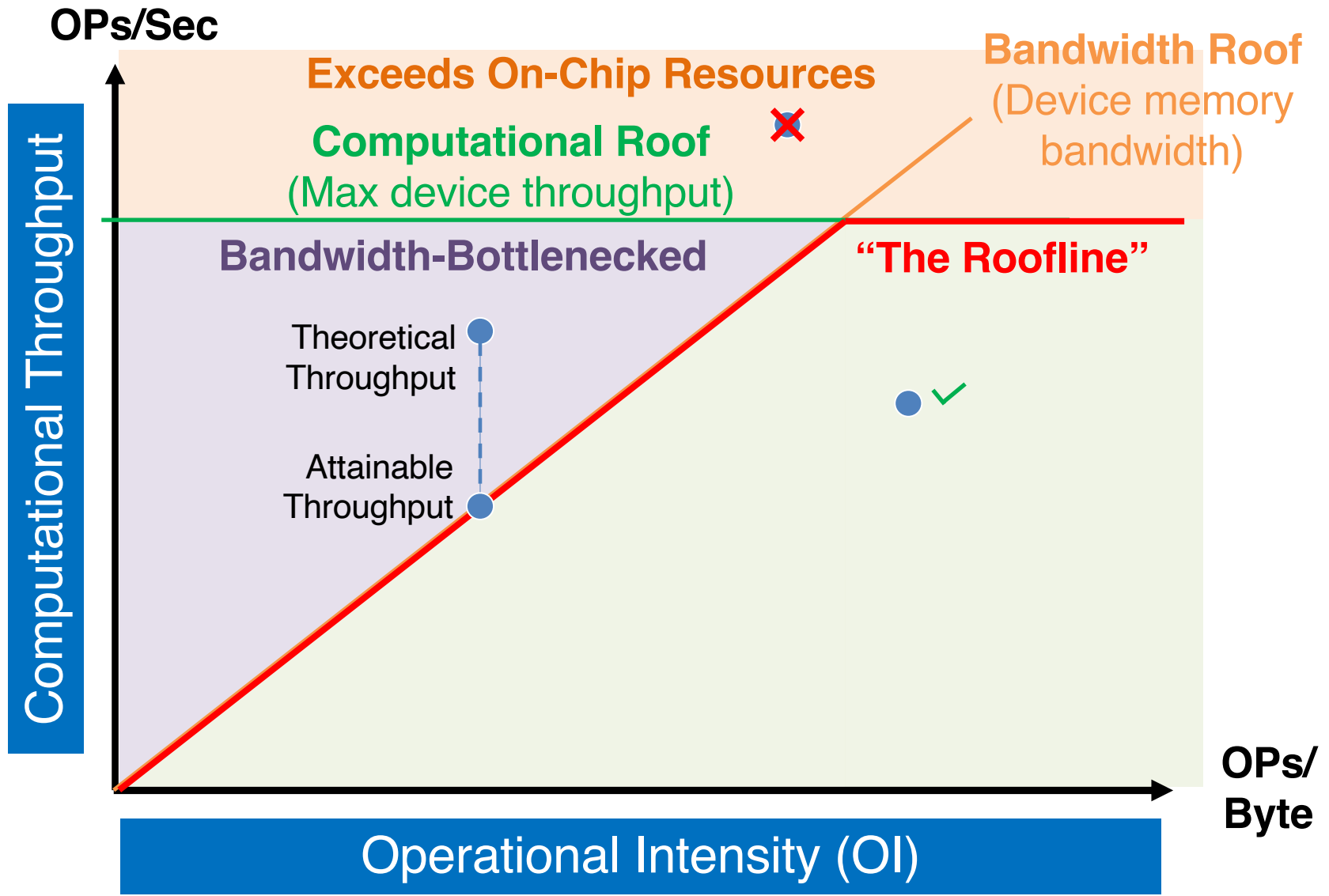
* OI is also known as computation to communication ratio (CTC) or arithmetic intensity (AI)

Roofline Model [1]



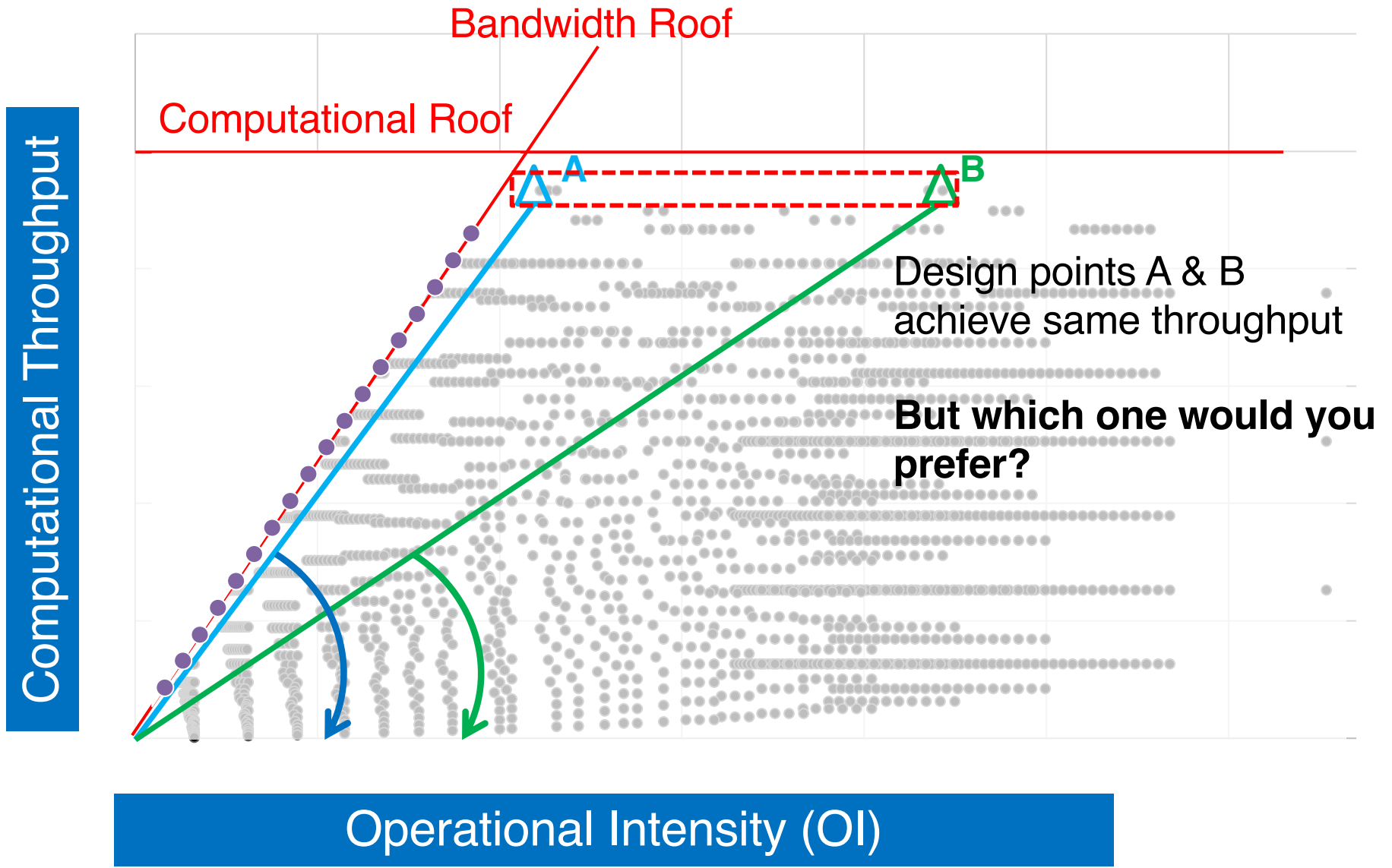
[1] S. Williams, A. Waterman, and D. Patterson, Roofline: an insightful visual performance model for multicore architectures, CACM, 2009.

Roofline Model [1]



[1] S. Williams, A. Waterman, and D. Patterson, Roofline: an insightful visual performance model for multicore architectures, CACM, 2009.

Design Space Exploration with Roofline



Summary

- ▶ End of Dennard scaling leads to increasing **hardware specialization** to sustain improvement in hardware performance and energy efficiency
- ▶ Special-purpose hardware **accelerators** commonly leverage customized (1) processing engines, (2) data types, (3) memory hierarchy, and (4) communication architectures
- ▶ **Roofline modeling** is a useful tool for first-order analysis of the accelerator performance

Reading Assignment

- ▶ Before next Thursday (9/7)
 - A. Boutros and V. Betz, “[FPGA Architecture: Principles and Progression](#)”, IEEE CAS-M 2021

Acknowledgements

- ▶ These slides contain/adapt materials developed by
 - Prof. Jason Cong (UCLA)