ECE 5997
Hardware Accelerator Design & Automation
Fall 2021

Specialized Computing

Cornell University

CSL

Announcements

- Lab 1 due next Wed 10/27
- TA Office Hour on Tuesdays at 4:40pm
  - Same zoom link

Agenda

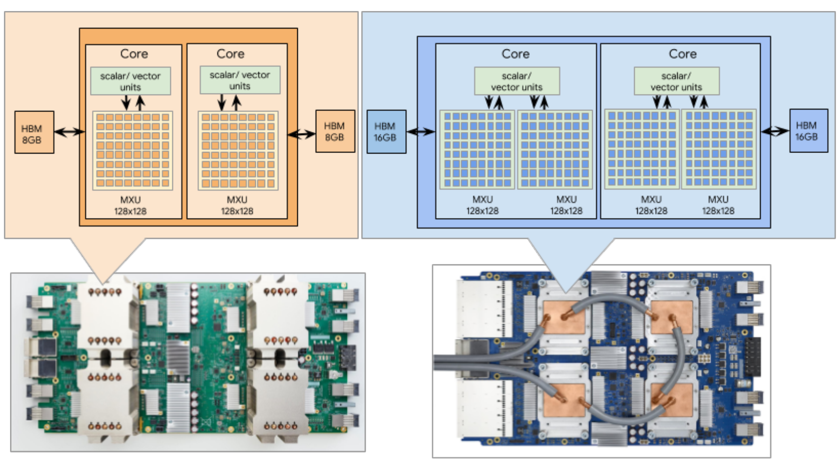
- Motivation for specialized computing
  - Key driving forces from applications and technology
  - Main sources of inefficiency in general-purpose computing
  - Case study on convolution

- FPGA introduction
  - Basic building blocks
  - Classical homogeneous FPGA architectures
  - Modern heterogeneous FPGA architectures

Best of Times for Specialized Computing

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

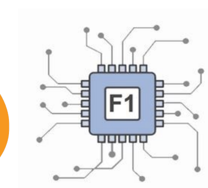
- **Lower barrier** with cloud FPGAs & open-source hardware coming of age



TPU v2 - 4 chips, 2 cores per chip

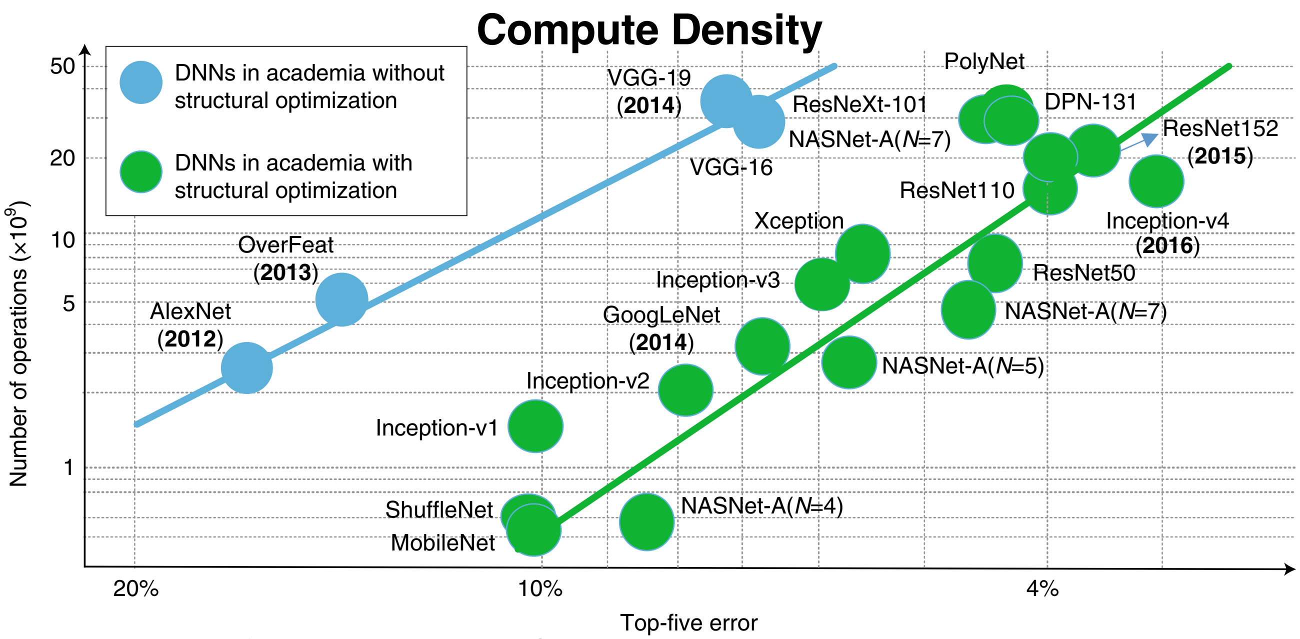
TPU v3 - 4 chips, 2 cores per chip

AWS



RISC-V

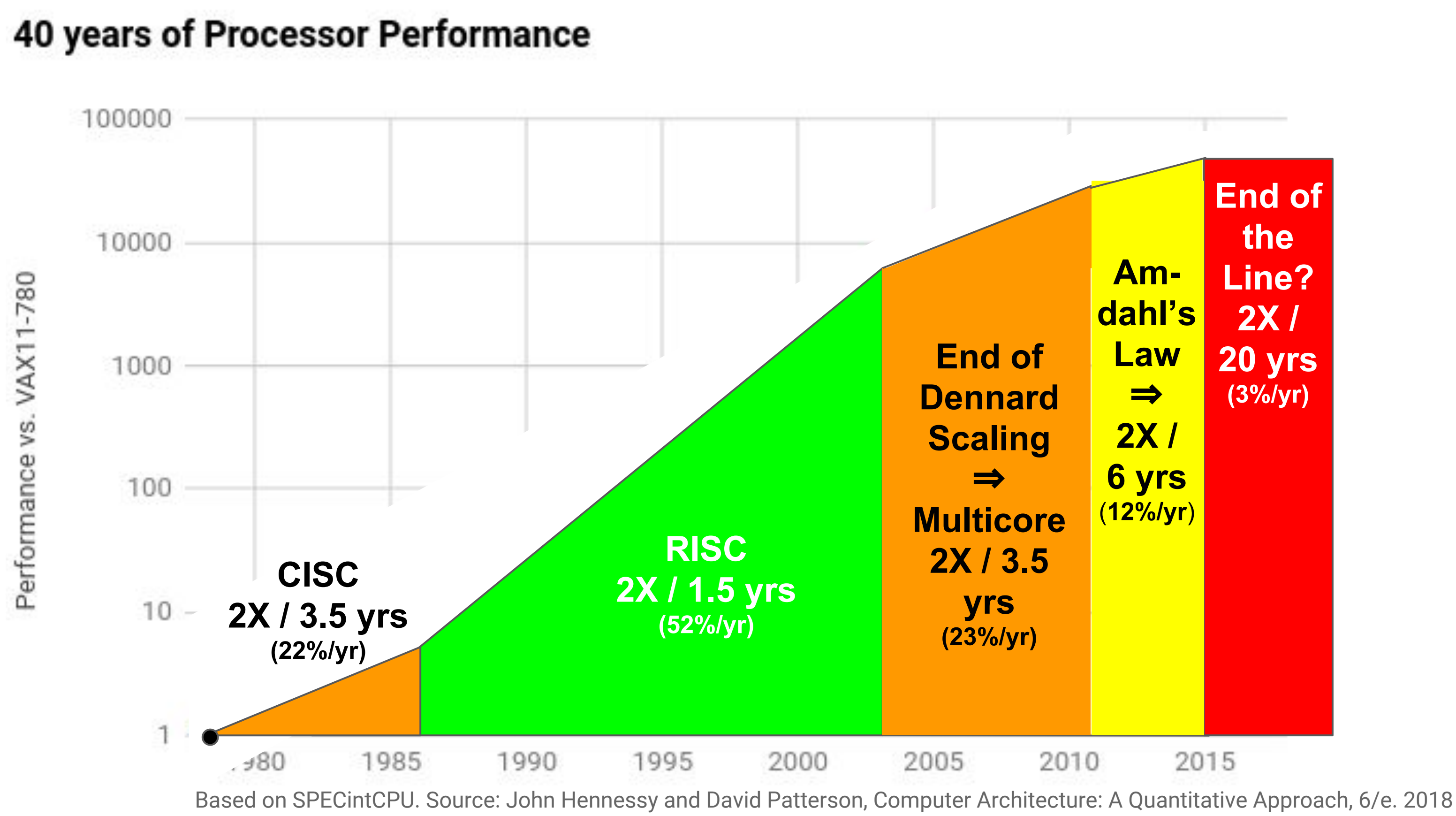
Modern ML Models are Computationally Expensive



X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi. **Scaling for Edge Inference of Neural Networks**. *Nature Electronics*, vol 1, Apr 2018.

- **Deep neural networks (DNNs) require enormous amount of compute**
  - For example, ResNet50 (70 layers) performs 7.7 billion operations to classify one image

On Crash Course with the End of "Cheap" Technology Scaling



- End of Dennard scaling: power becomes the key constraint
  - Amdahl's Law and dark silicon prevent "easy" multicore scaling

End of Dennard Scaling and Dark Silicon

- Classical scaling
  - Frequency increases at constant power profiles
    - Performance improves “for free"!
- Leakage limited scaling
  - V_{th} virtually stopped scaling due to exponentially increasing leakage power
  - V_{DD} scaling nearly stopped as well to maintain performance
- Dark silicon
  - Power constraints limit how much of the chip can be activated at any one time (not 100% anymore)

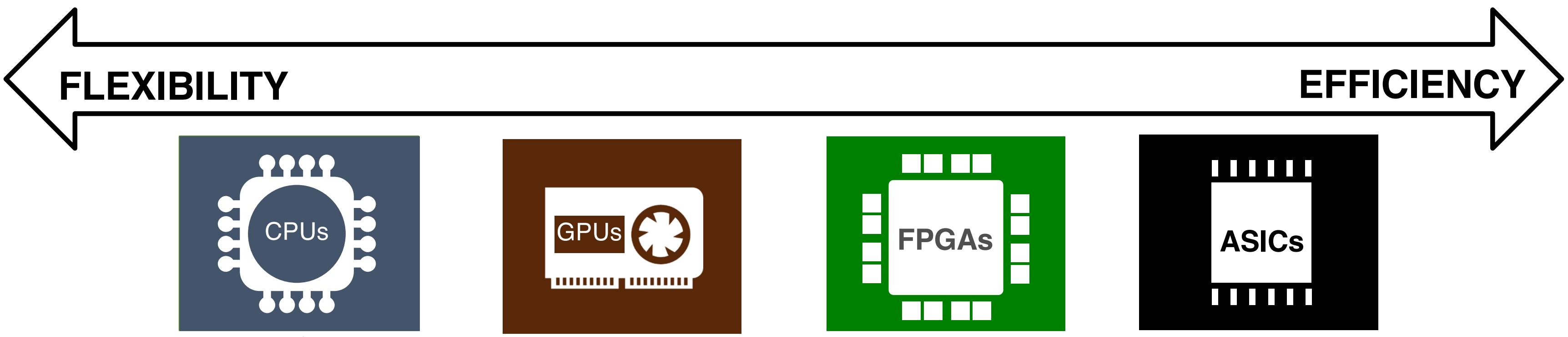
Classical Dennard scaling

Device (transistor) #	S^2
Capacitance / device	1/S
Voltage (V_{dd})	1/S
Frequency	S
Total power	**1**

Leakage limited scaling

Device (transistor) #	S^2
Capacitance / device	1/S
Voltage (V_{dd})	**~1**
Frequency	**~1**
Total power	**S**

Tradeoff between Compute Efficiency and Flexibility



Why is general-purpose CPU less energy efficient?

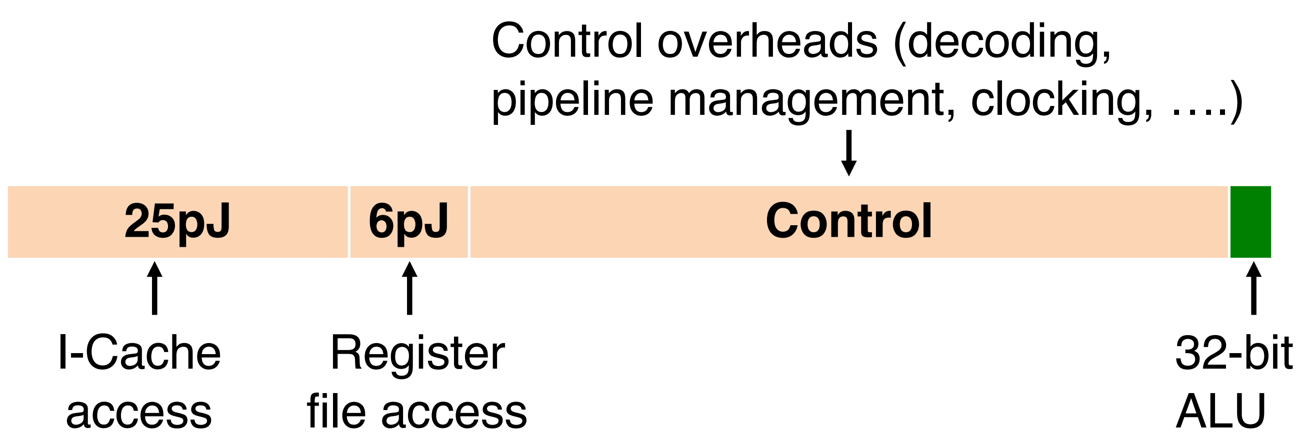
Rough Energy Breakdown for an Instruction

Rough energy costs for various CPU operations (45nm at 0.9V)

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

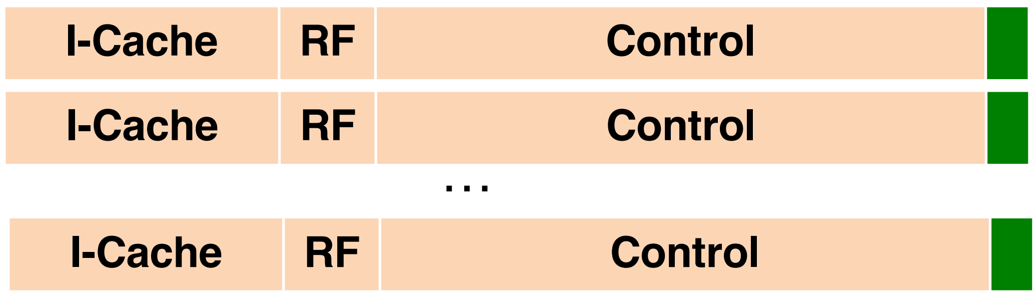
Memory	
Cache	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ



[Source] M. Horowitz, Computing's energy problem (and what we can do about it), ISSCC'2014.

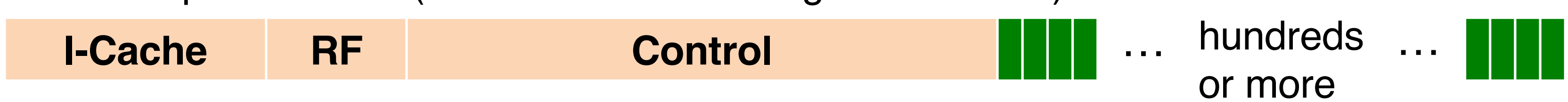
Reducing Compute Energy Overhead

A sequence of energy-inefficient instructions



Single Instruction Multiple Data (SIMD): tens of operations per instruction

Further specialization (what we achieve using accelerators)



[Figure credit] W. Qadder, et al., Convolution Engine: Balancing Efficiency & Flexibility in Specialized Computing, ISCA'2013.

Additional Energy Savings from Specialization

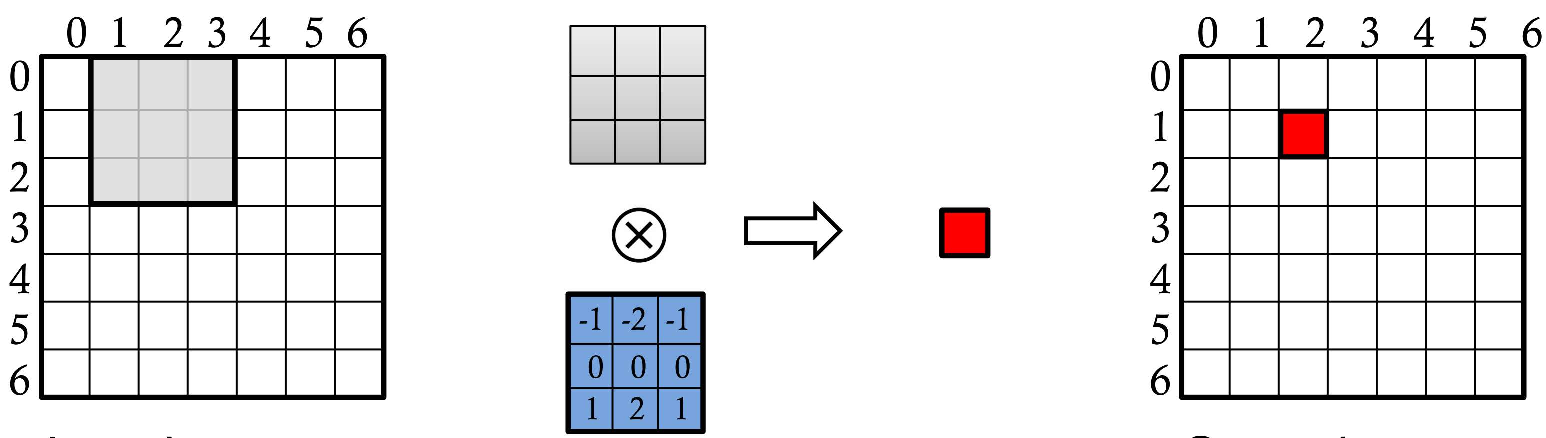
- **Customized data types**
  - Exploit data range information to reduce bitwidth/precision and simply arithmetic operations
- **Customized memory hierarchy**
  - Exploit regular memory access patterns to minimize energy per memory read/write
- **Customized communication architecture**
  - Exploit data 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 GPPs

Customized Memory Hierarchy:
A Case Study on Convolution

- The main computation of image/video processing is performed over overlapping stencils, termed as <u>convolution</u>

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



Input image frame

KxK convolution (K=3 here)

Output image frame

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 $G=(G_X, G_Y)$

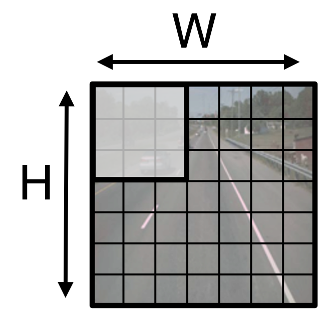
$$G_X=\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

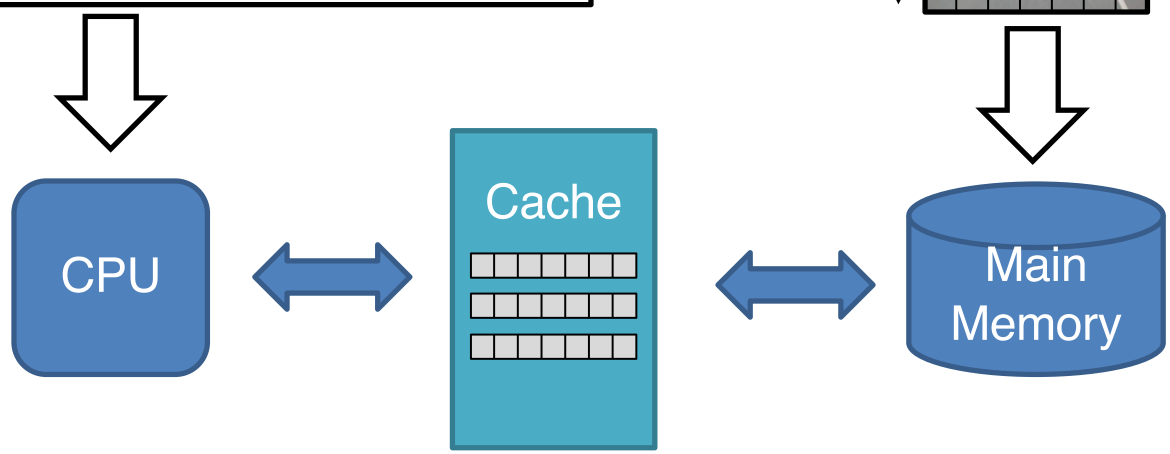
$$G_Y=\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Figures: Pilho Kim, GaTech

CPU Implementation of a 3x3 Convolution

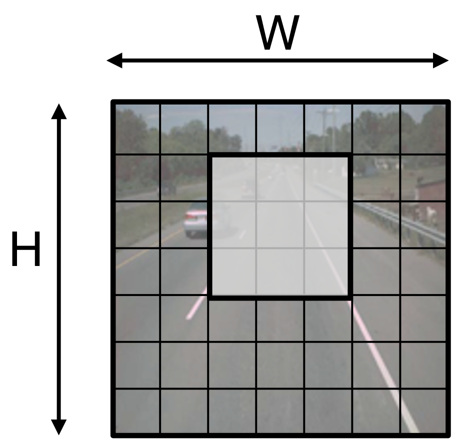
```
for (r = 1; r < H; r++)
   for (c = 1; c < W; c++)
     for (i = 0; i < K; i++)
       for (j = 0; j < K; j++)
         out[r][c] += img[r+i-1][c+j-1] * f[i][j];
```



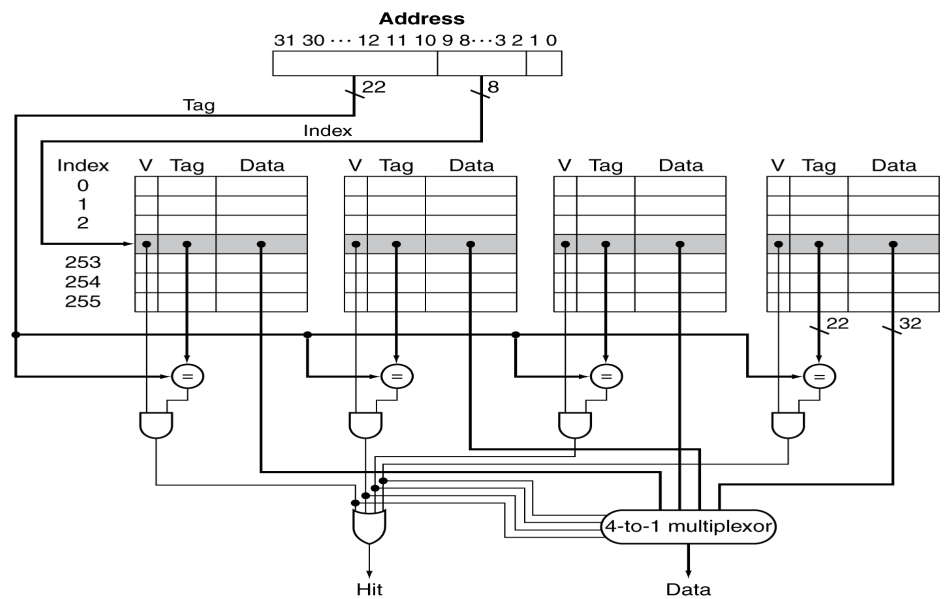


General-Purpose Cache for Convolution

- Minimizes main memory accesses to improve performance



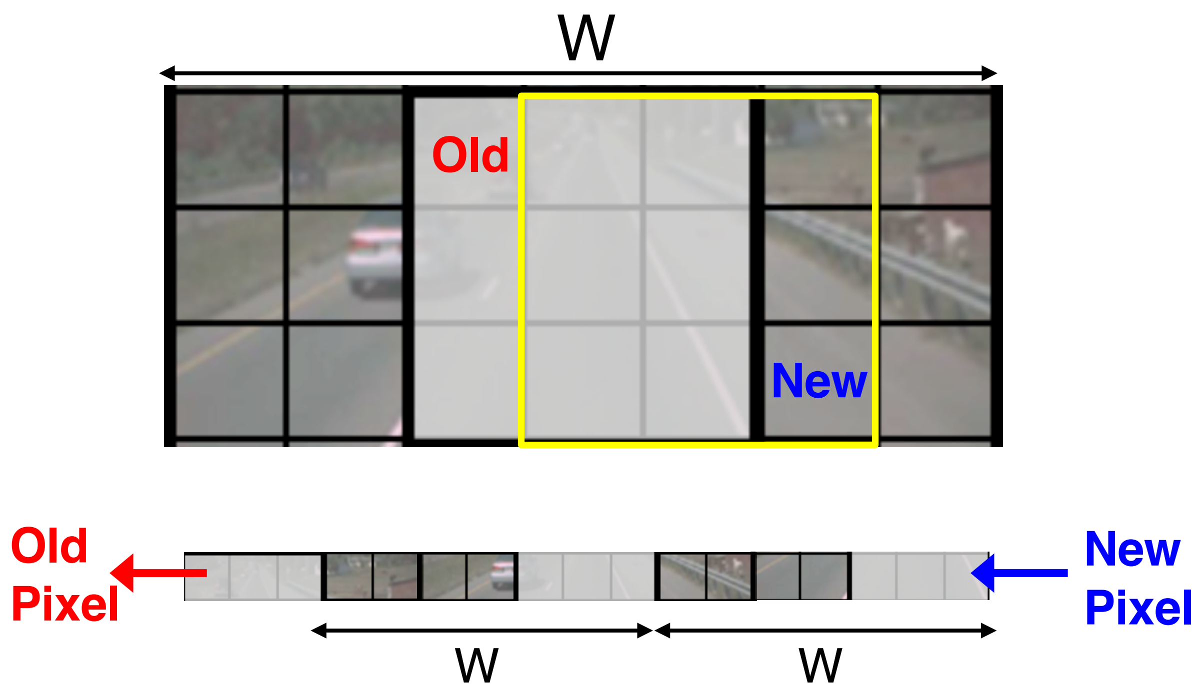
Input picture
(W pixels wide)



- A general-purpose cache is expensive in cost and incurs nontrivial energy overhead

Specializing Cache for Convolution

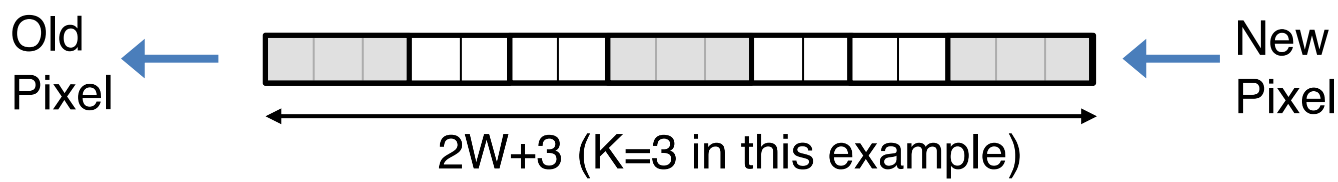
- Rearrange the rows as a 1D array of pixels
- Each time we move the window to right and push in the new pixel to the "cache"



Remove the edge pixels that are not needed for computation

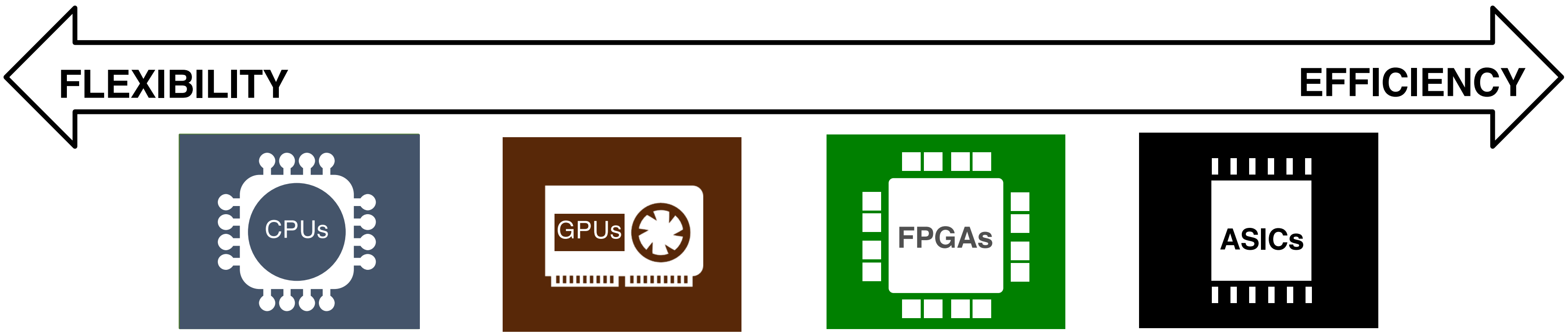
A Specialized "Cache": Line Buffer

- Line buffer: a fixed-width "cache" with (K-1)*W+K pixels in flight
  - Fixed addressing and simple replacement policy
  - Low area/power and high performance



- In customized FPGA implementation, line buffers can be efficiently implemented with on-chip BRAMs

Tradeoff between Compute Efficiency and Flexibility

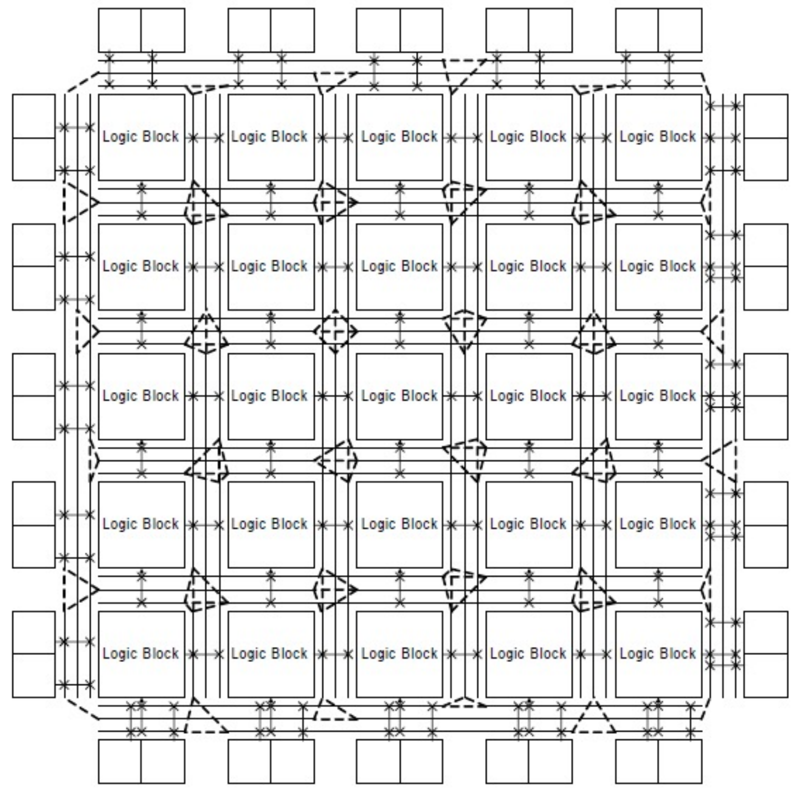


What makes FPGA an interesting compute substrate?

What is an FPGA?

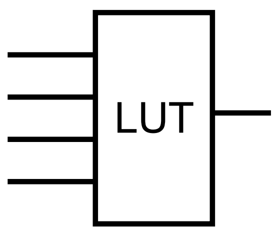
- FPGA: Field-Programmable Gate Array
  - An integrated circuit designed to be configured by a customer or a designer after manufacturing (wikipedia)

- Components in an FPGA Chip
  - Programmable logic blocks
  - Programmable interconnects
  - Programmable I/Os

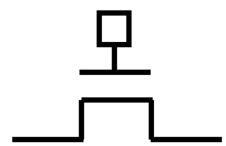


Three Important Pieces

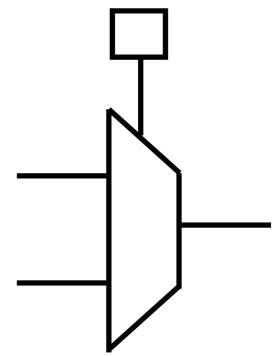
- SRAM-based implementation is popular
  - Non-standard technology means older technology generation



Lookup table (LUT, formed by SRAM bits)



Pass transistor (controlled by an SRAM bit)

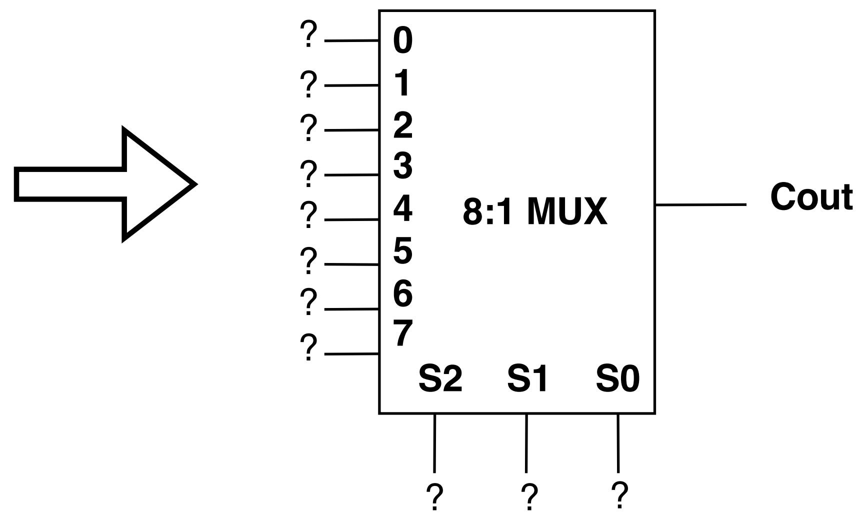


Multiplexer (controlled by SRAM bits)

Multiplexer as a Universal Gate

- Any function of k variables can be implemented with a 2^k:1 multiplexer

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



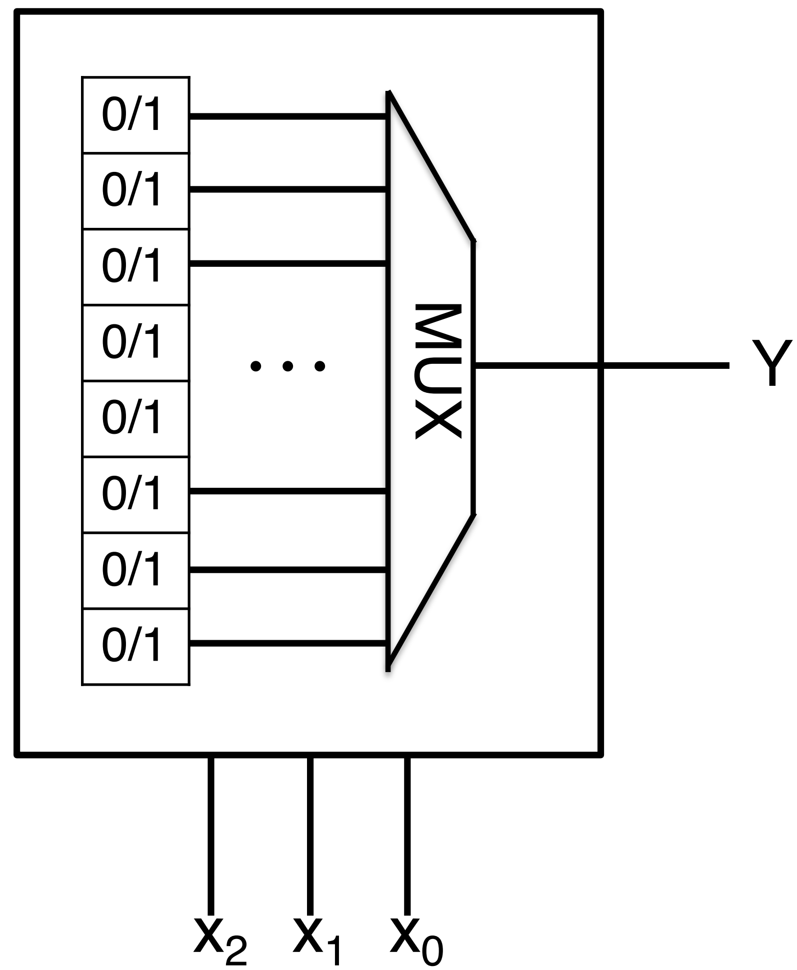
How Many Functions?

- How many distinct 3-input 1-output Boolean functions exist?

- What about K inputs?

Look-Up Table (LUT)

- A k-input LUT (k-LUT) can be configured to implement any k-input 1-output combinational logic
  - 2^k SRAM bits
  - Delay is independent of logic function



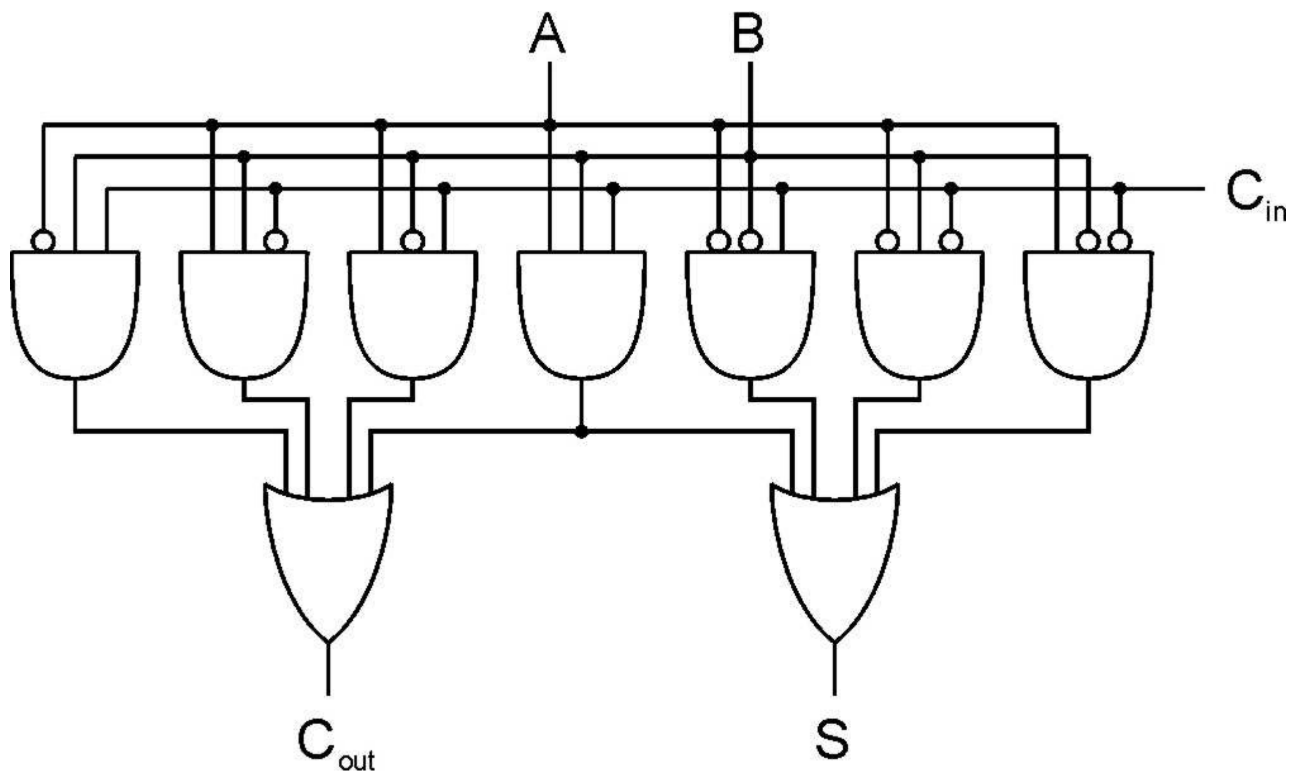
A 3-input LUT

Exercise: How Many LUTs? (2 mins)

(1) How many 3-input LUTs are needed to implement the following full adder?

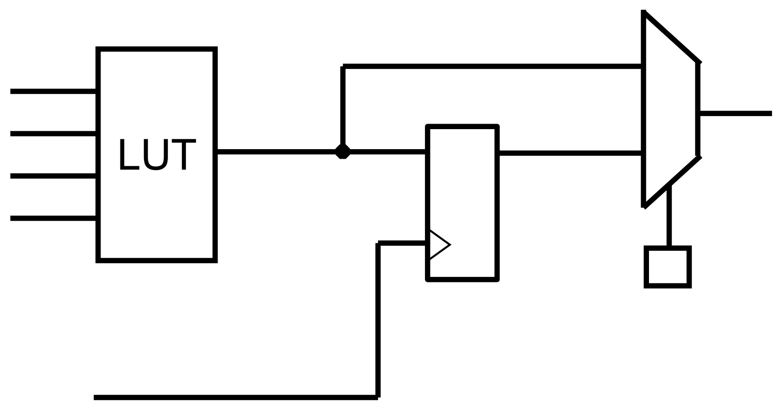
(2) How about using 4-input LUTs?

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



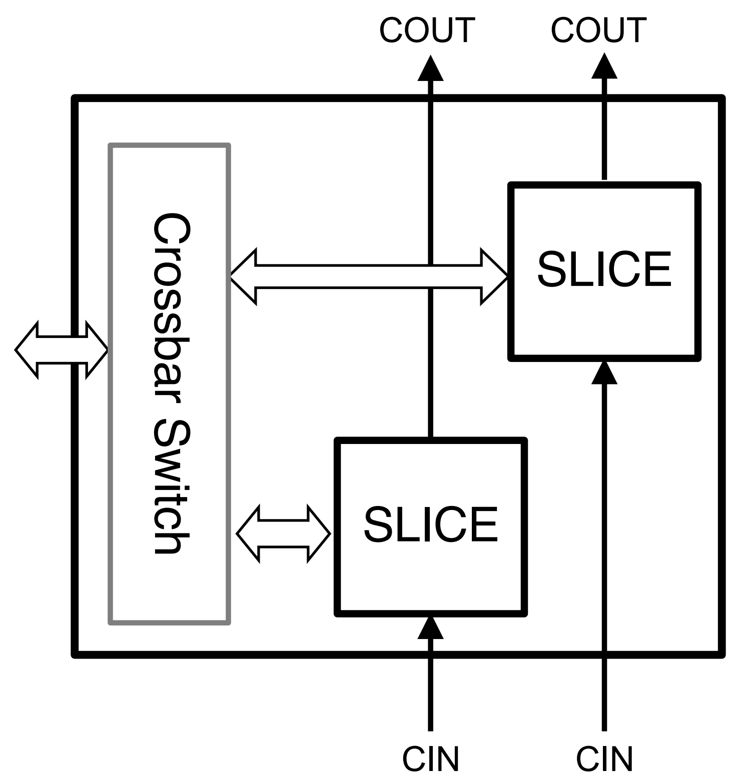
A Logic Element

- A k-input LUT is usually followed by a flip-flop (FF) that can be bypassed
- The LUT and FF combined form a logic element

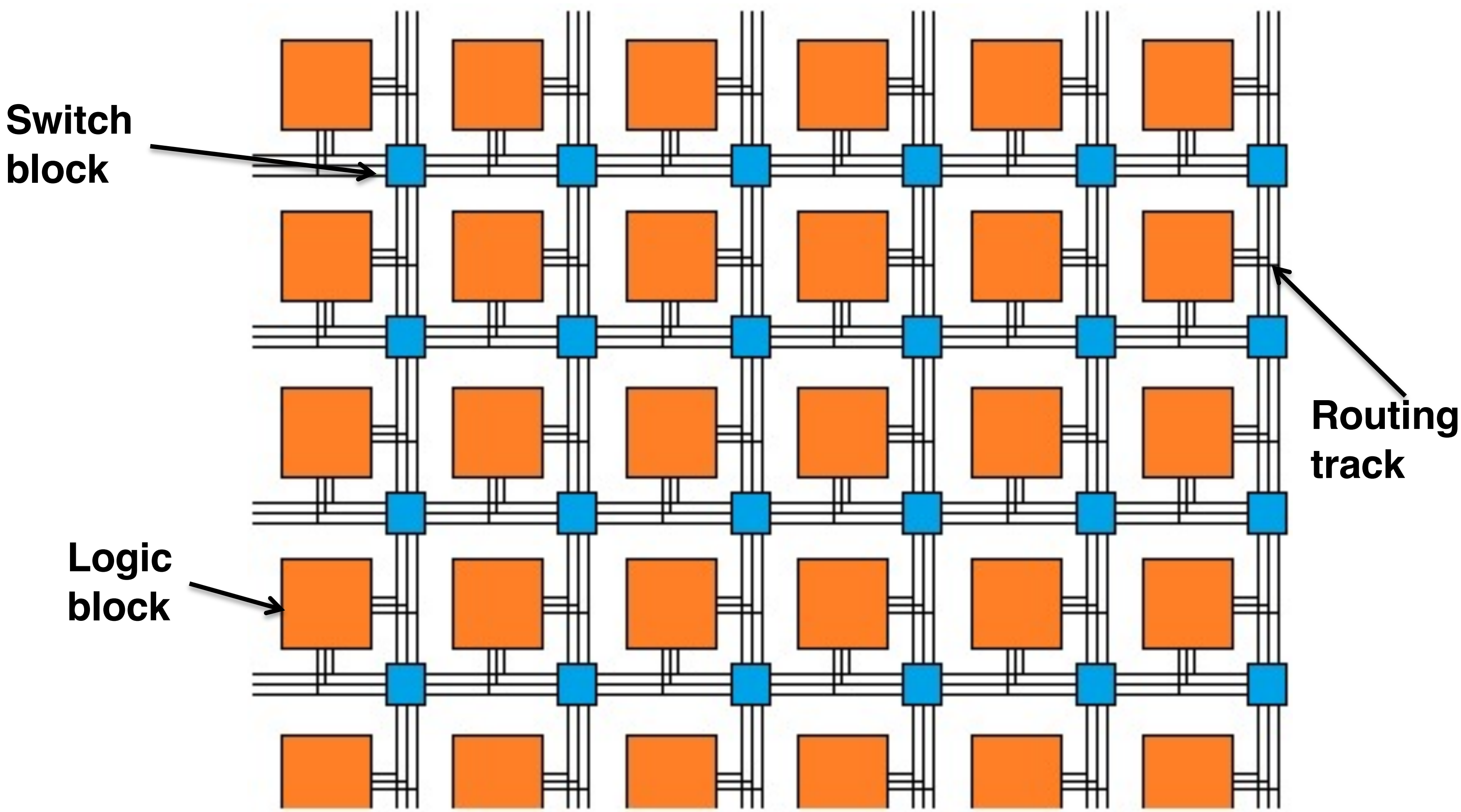


A Logic Block

- A logic block clusters multiple logic elements
- Example: In Xilinx 7-series FPGAs, each configurable logic block (CLB) has two slices
  - Two independent carry chains per CLB for implementing adders
  - Each slice contains four LUTs

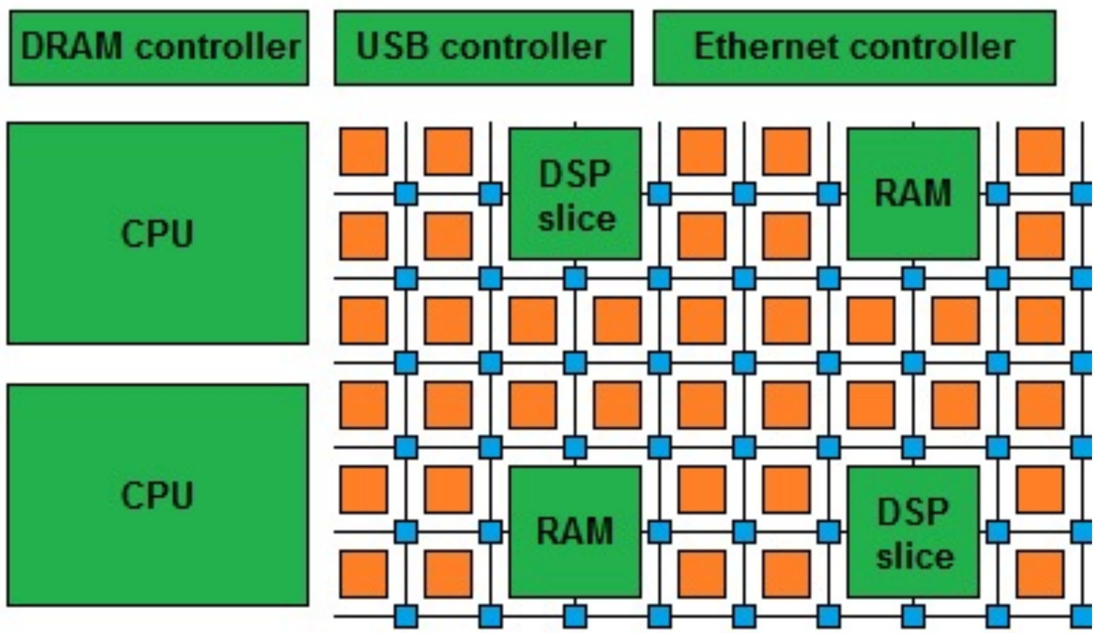


Traditional Homogeneous FPGA Architecture



Modern Heterogeneous Field-Programmable System-on-Chip

- Island-style configurable mesh routing
- Lots of dedicated components
  - Memories/multipliers, I/Os, processors
  - Specialization leads to higher performance and lower power

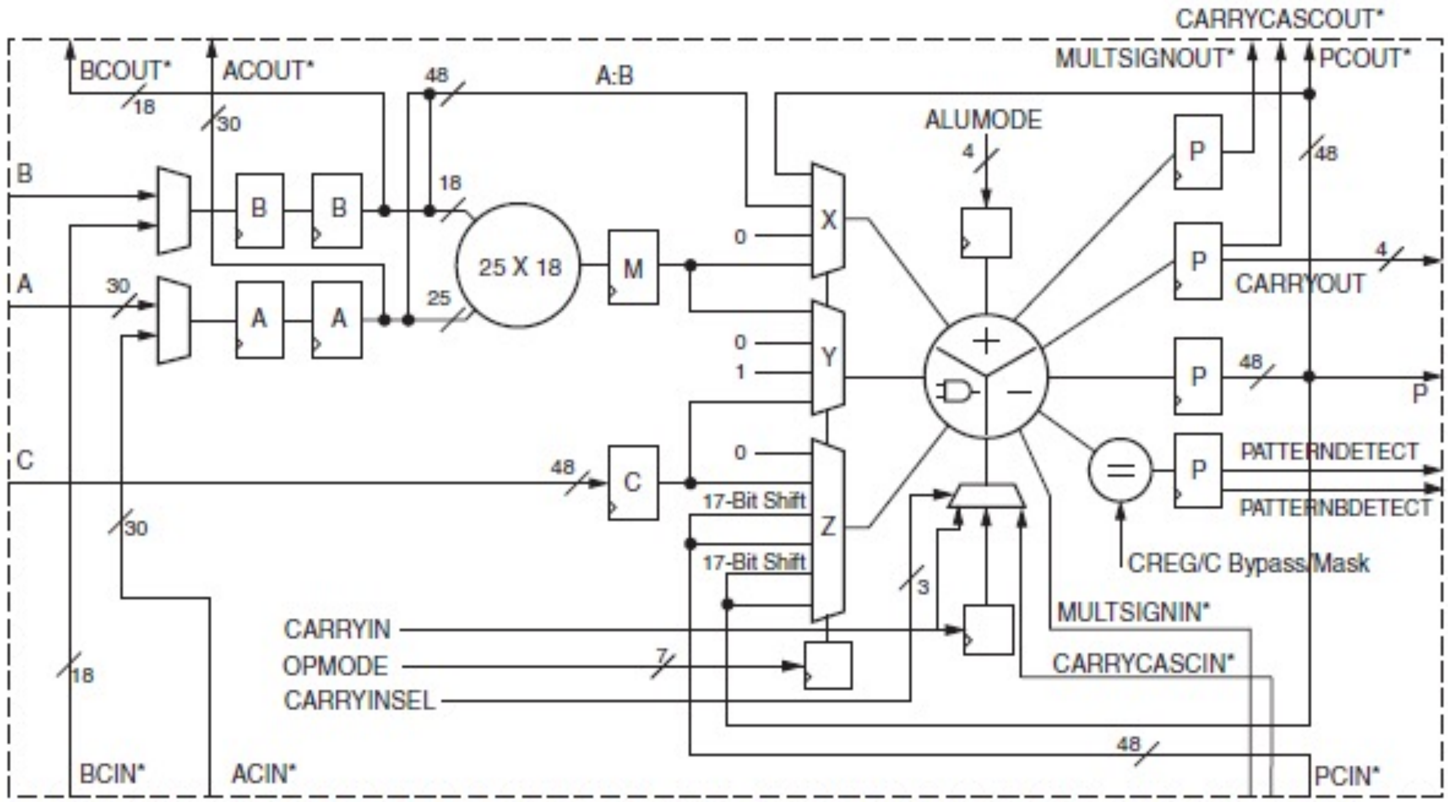


[Figure credit: embeddedrelated.com]

Dedicated DSP Blocks

- Built-in components for fast arithmetic operation optimized for DSP applications
  - Essentially a multiply-accumulate core with many other features
  - Fixed logic and connections, functionality may be configured using control signals at run time
  - Much faster than LUT-based implementation (ASIC vs. LUT)

Example: Xilinx DSP48E Slice



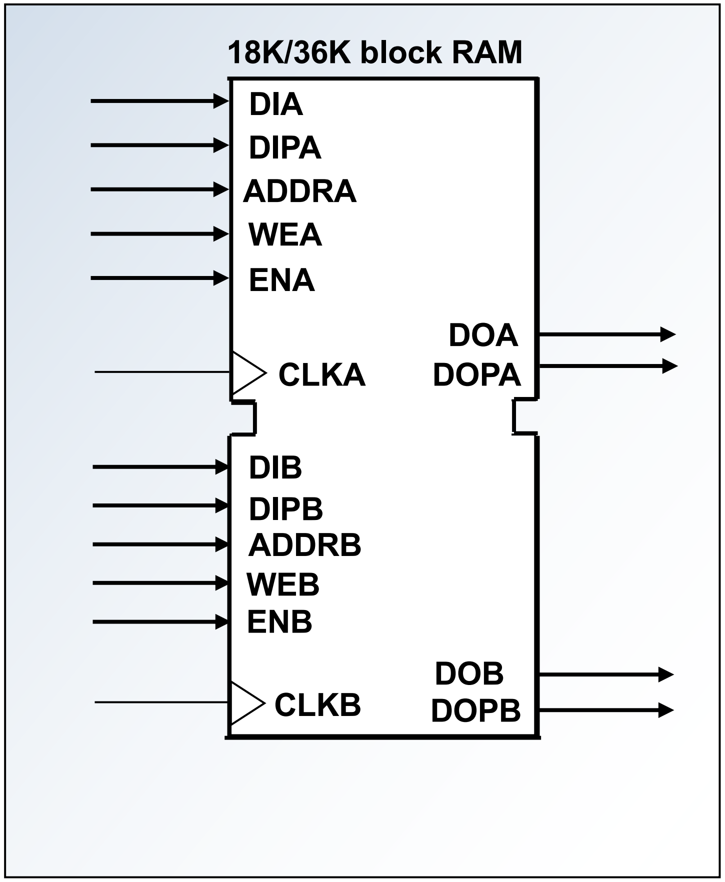
*These signals are dedicated routing paths internal to the DSP48E column. They a

- 25x18 signed multiplier
- 48-bit add/subtract/accumulate
- 48-bit logic operations
- SIMD operations (12/24 bit)
- Pipeline registers for high speed

[source: Xilinx Inc.]

Dedicated Block RAMs (BRAMs)

- Example: Xilinx 18K/36K block RAMs
  - 32k x 1 to 512 x 72 in one 36K block
  - Simple dual-port and true dual-port configurations
  - Built-in FIFO logic
  - 64-bit error correction coding per 36K block

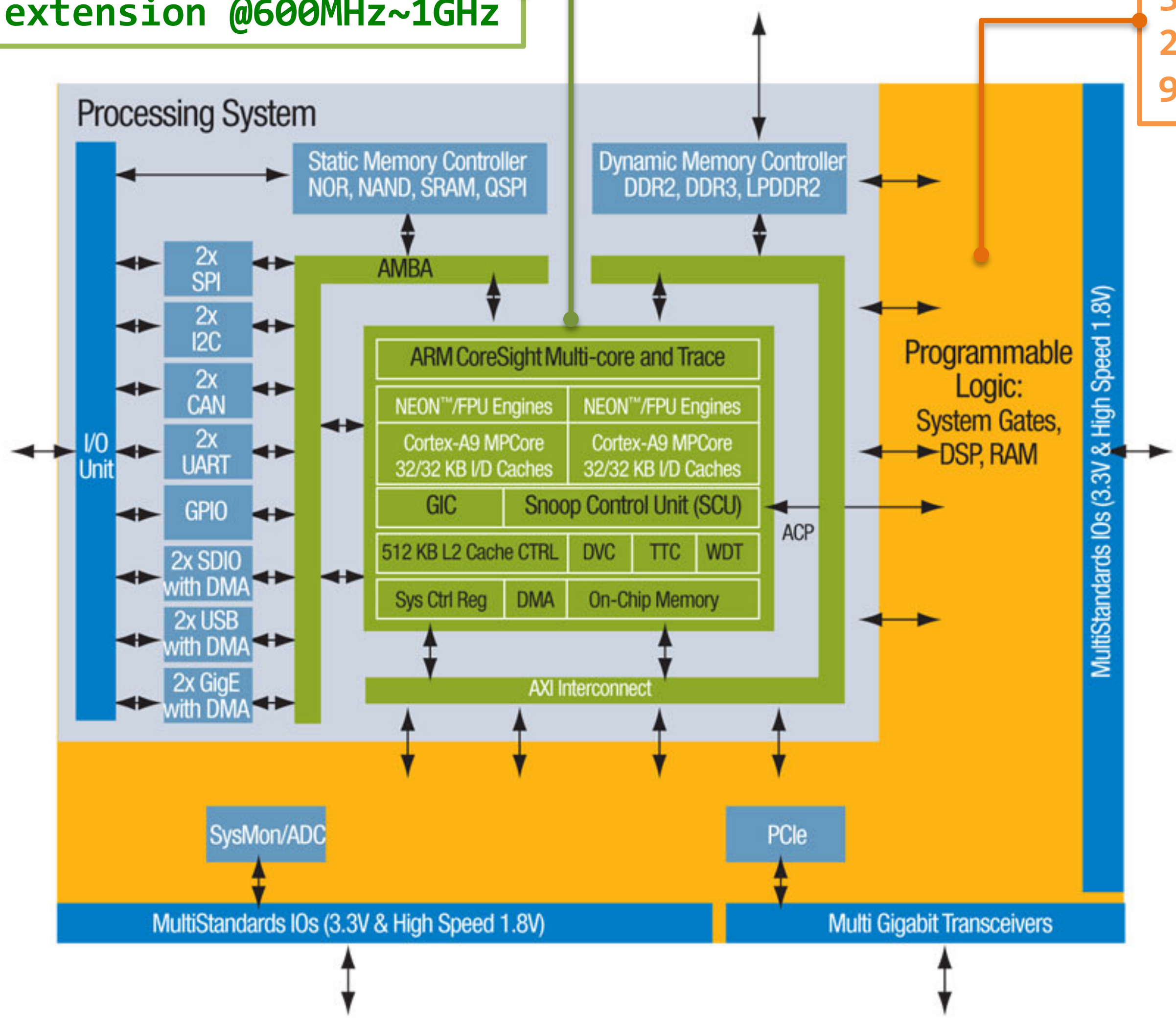


[source: Xilinx Inc.]

Embedded FPGA System-on-Chip

Dual ARM Cortex-A9 + NEON SIMD extension @600MHz~1GHz

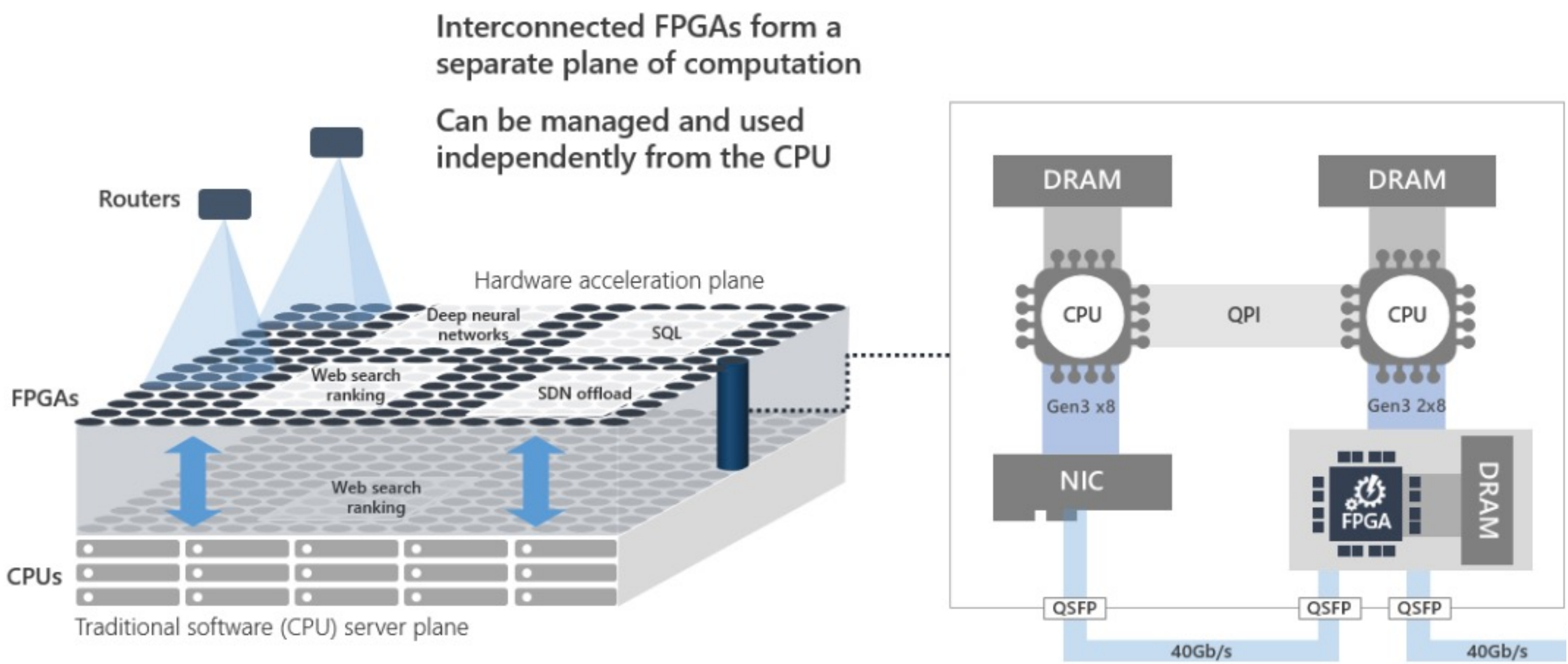
Up to
350K logic cells
2MB Block RAM
900 DSP48s



Xilinx Zynq All Programmable System-on-Chip
[Source: Xilinx Inc.]

FPGA Deployment in Datacenter

- FPGAs deployed in Microsoft datacenters to accelerate various web, database, and AI services
  - e.g., project BrainWave claimed ~40Teraflops on large recurrent neural networks using Intel Stratix 10 FPGAs



[source: Microsoft, BrainWave]

Summary: FPGA as a Programmable Accelerator

- **Massive amount of fine-grained parallelism**
  - Highly parallel and/or deeply pipelined to achieve maximum parallelism
  - Distributed data/control dispatch
- **Silicon configurable to fit algorithm**
  - Compute the exact algorithm at the desired level of numerical accuracy
    - Bit-level sizing and sub-cycle chaining
  - Customized memory hierarchy
- **Performance/watt advantage**
  - Low power consumption compared to CPU and GPGPUs
    - Low clock speed
    - Specialized architecture blocks

Next Class

- Front-end Compilation

Acknowledgements

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