EECS 570
Lecture 17
Accelerators - Intro

Winter 2025

Prof. Satish Narayanasamy

http://www.eecs.umich.edu/co4urses/eecs570/

Slides adapted from instructional material from Joel Emer and Vivienne Sze (MIT)

What is Moore’s Law?

CPU Performance

CPU performance will double every two years, following the pattern predicted by Moore's Law.

Transistor Size

Transistors will shrink to half their size every two years, allowing for more compact and efficient designs.

Chip Performance

Chip performance is expected to double every two years, highlighting ongoing enhancements in microchip technology.

Gate Width

Gate width will shrink every two years, enabling more transistors to fit on a chip.

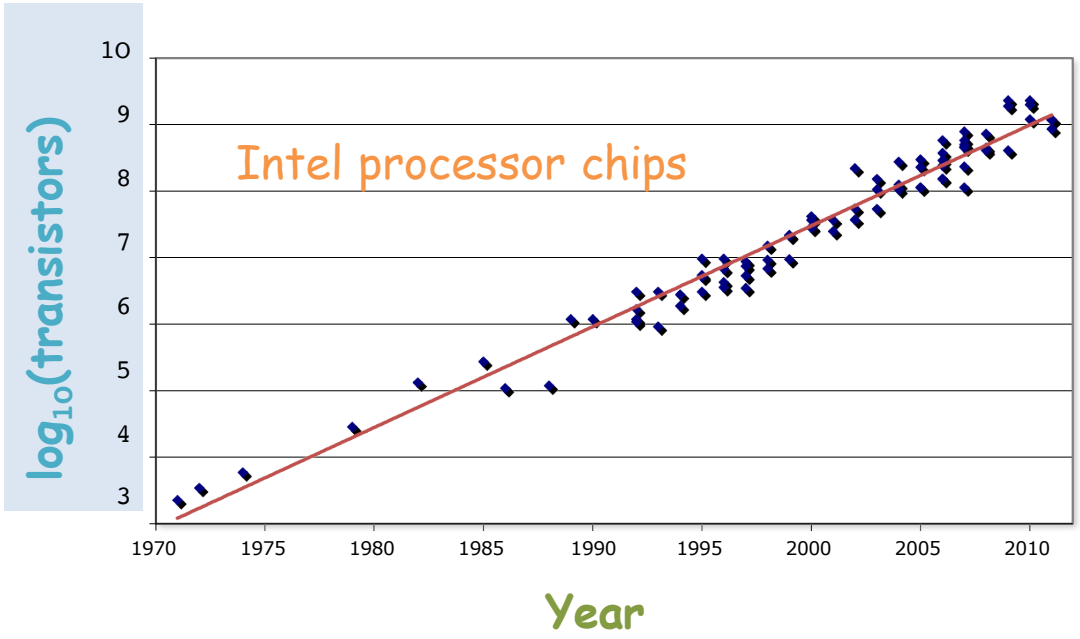
Transistor Speed

The speed of transistors will double every two years, contributing to faster processing capabilities.

Transistors Per Die

The number of transistors per die will double every two years, boosting computational power.

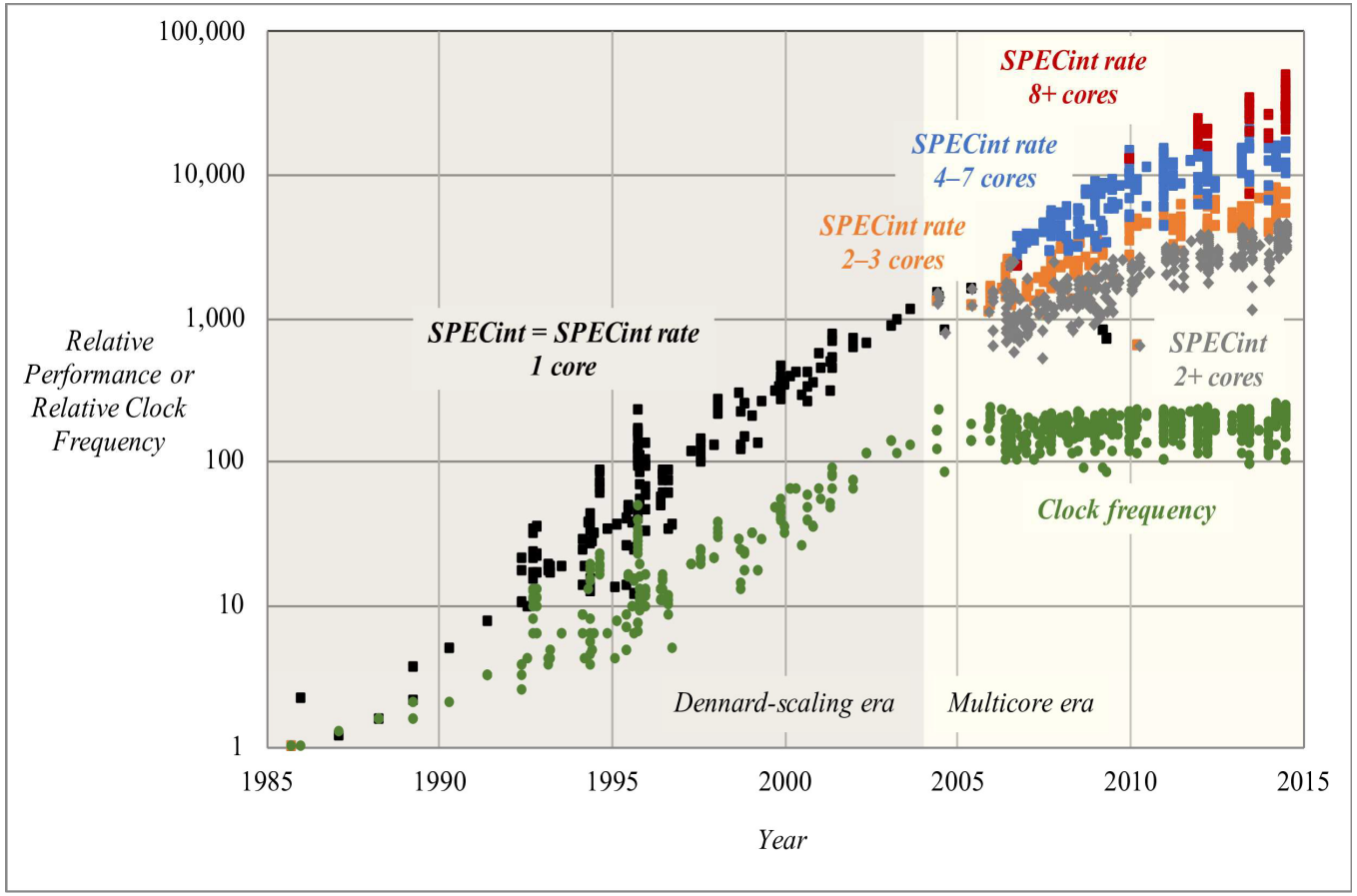
Moore's (Transistor) Law



[Moore, Progress in digital integrated electronics, IEDM 1975]

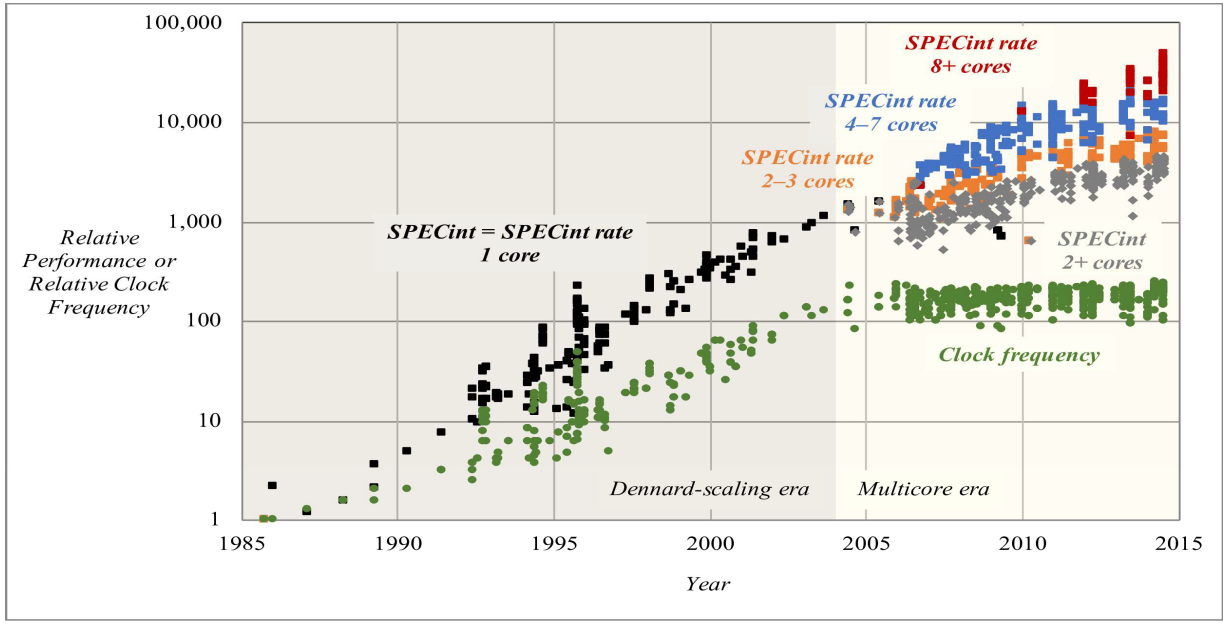
Number of transistors has been doubling

Moore's (Performance) Law



[Leiserson et al., There's Plenty of Room at the Top, Science]

The End of Historic Scaling



[Leiserson et al., There's Plenty of Room at the Top, Science]

Voltage scaling slowed down → Power density increasing!

During the Moore + Dennard's Law Era

- Instruction-level parallelism (ILP) was largely mined out by early 2000s
- Voltage (Dennard) scaling ended in 2005
- Hit the power limit wall in 2005
- Performance is coming from parallelism using more transistors since ~2007
- But….

Moore's Law in DRAMs

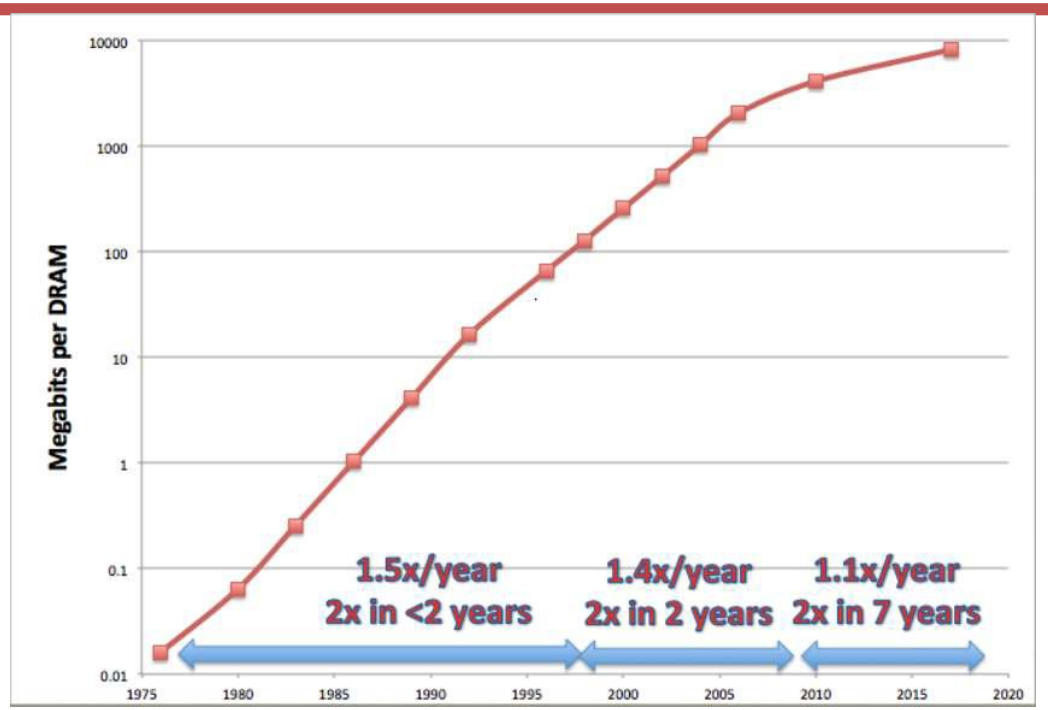
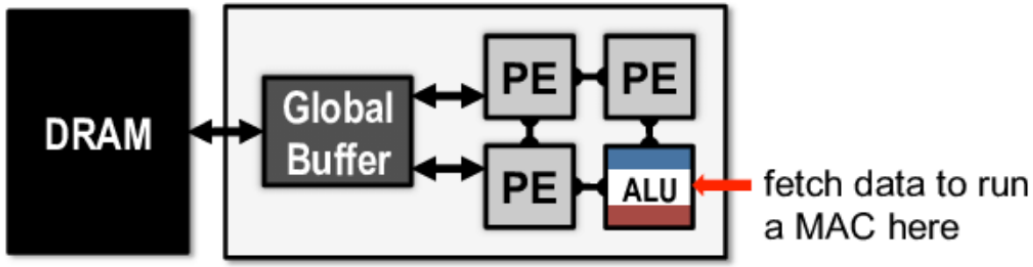


Image source: John Hennessy

After multi-core, specialization (i.e., accelerators) seems to be
the most attractive architectural option to cope with the end of Moore's Law

The High Cost of Data Movement

Fetching operands more expensive than computing on them



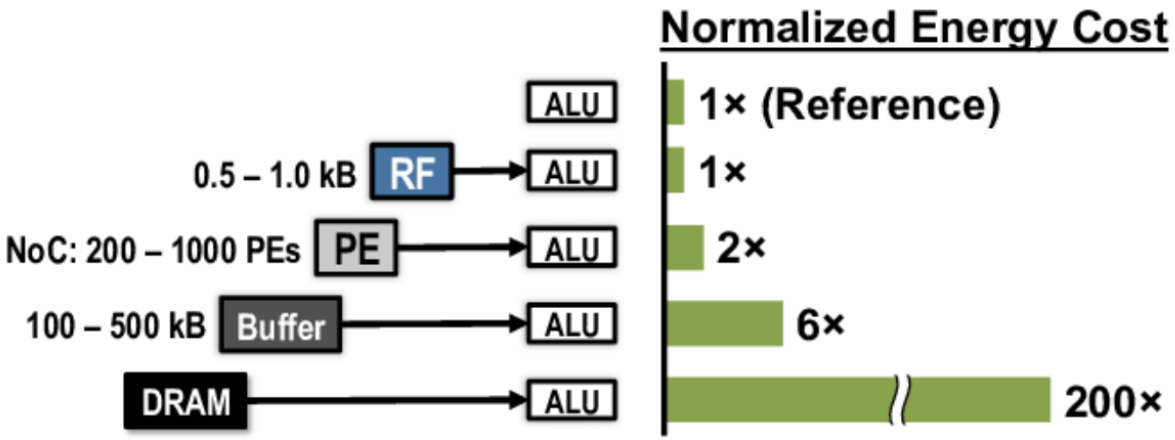


Image source: Bill Dally
Data movement energy comparison among memory hierarchies [14] | Download Scientific Diagram

Now the key is how we use our transistors most effectively.

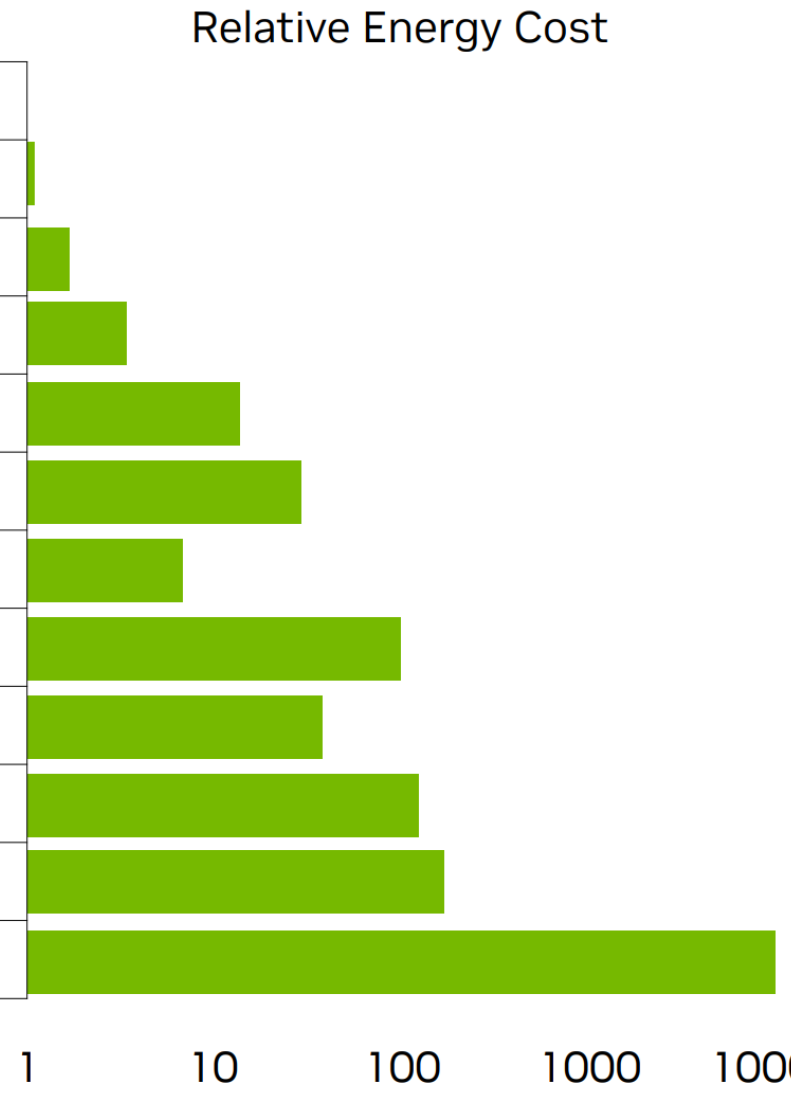
Accelerator Design Attributes

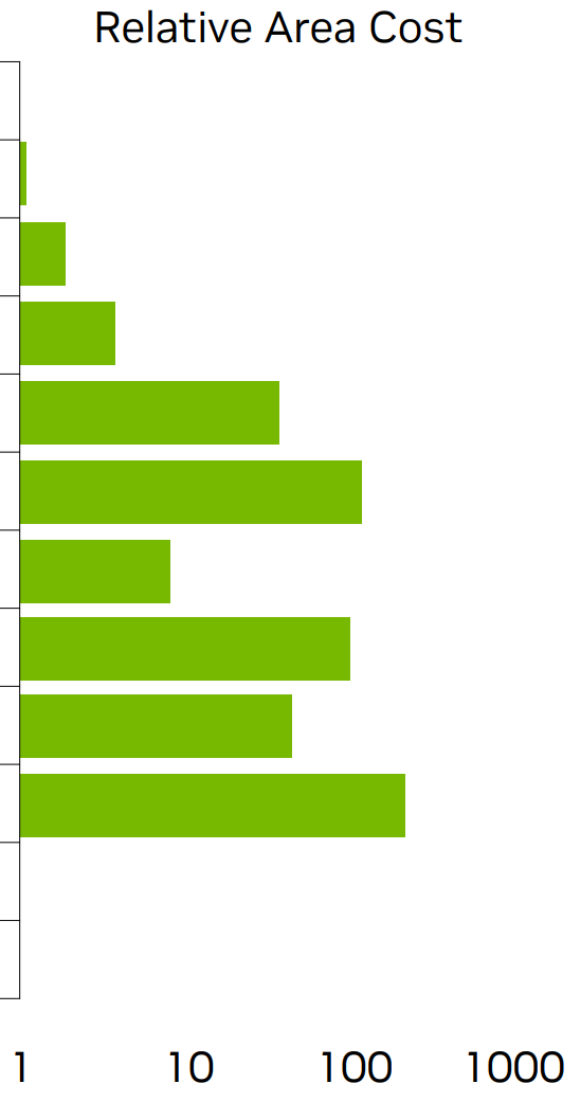
- Integration into system
  - How is the accelerator integrated into the system?
- Operation specification/sequencing
  - How is activity managed within the accelerator?
- Data management
  - How is data movement orchestrated in the accelerator?

System Integration

Cost of Operations

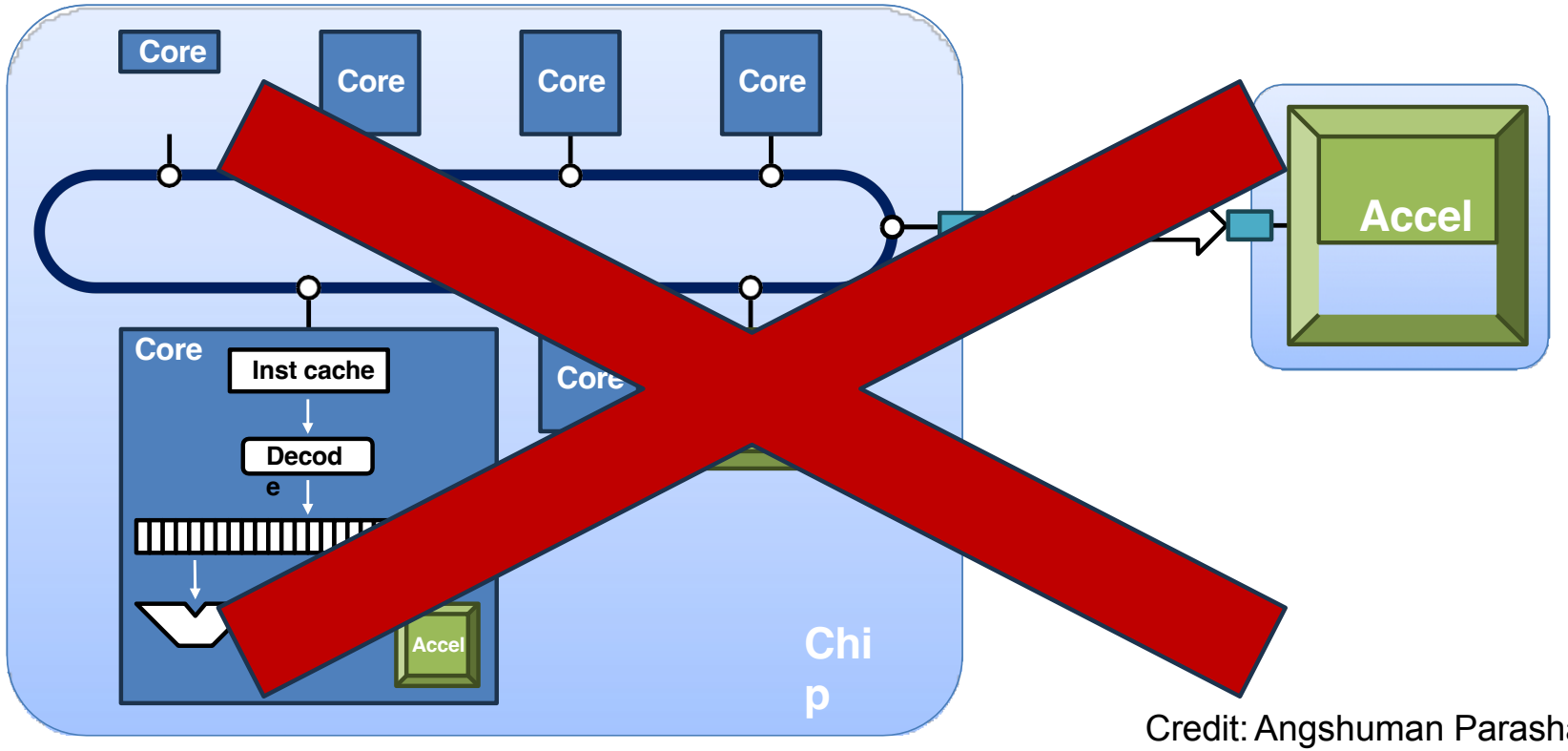
Operation:	Energy (pJ)	Area (μm²)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A





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.

Accelerator Integration Taxonomy



Credit: Angshuman Parashar

Accelerator Architectural Choices

- State
  - Local state – Is there local state? (i.e. context)
  - Global state – e.g., main memory shared with CPU
- Data Operations
  - Custom data operations in the accelerator
- Memory Access Operations
  - Operations to access shared global memory – Do they exist?
- Control Flow Operations
  - How is accelerator sequenced relative to CPU?

How to Sequence Accelerator Logic?

- Synchronous
  - Accelerated operations inside processor pipeline
    - E.g., as a separate function unit
  - Control handled by standard control instructions

- Asynchronous
  - A standalone logical machine
    - Accelerator started by processor that continues running

What factors mitigate for one form or the other?

Latency of operation
Existence of concurrent activity
Size of logic and operands

Eight Alternatives

Architectural semantics		
Asynchronous	**Access memory**	**Has context**
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Characteristics of “no memory access” choice?

Good for smaller operands

Simpler, e.g., no virtual memory

No ‘Little’s Law’ storage requirement

Eight Alternatives

Architectural semantics		
Asynchronous	**Accesses memory**	**Has context**
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Characteristics of “no context” choice?

Simpler, no context switch mechanism

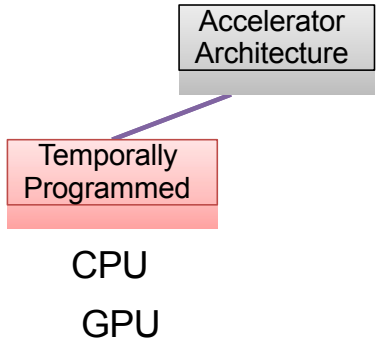
Long operations run to completion More limited reuse opportunities

Accelerator Architectural Alternatives

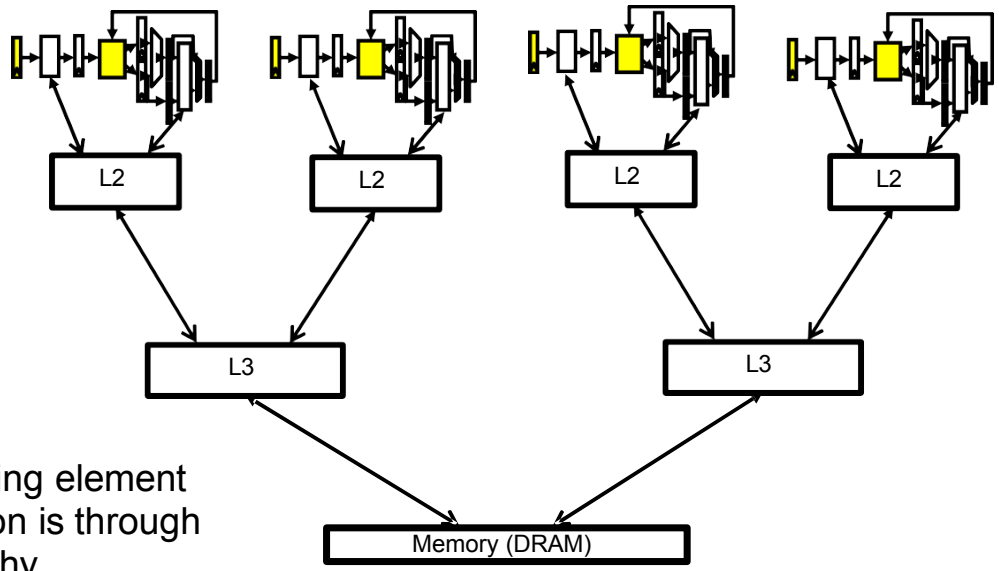
Architectural semantics			Example
Asynchronous	**Accesses memory**	**Has context**	
0	0	0	New function unit, like tensor core in GPU
0	0	1	Accumulating data reduction instruction
0	1	0	Memory-to-memory vector unit
0	1	1	Register-based vector unit including load store ops
1	0	0	Complex function calculator?
1	0	1	Security co-processor (TPM)
1	1	0	Network adapter
1	1	1	GPU with virtual memory

Operation Sequencing

Accelerator Taxonomy



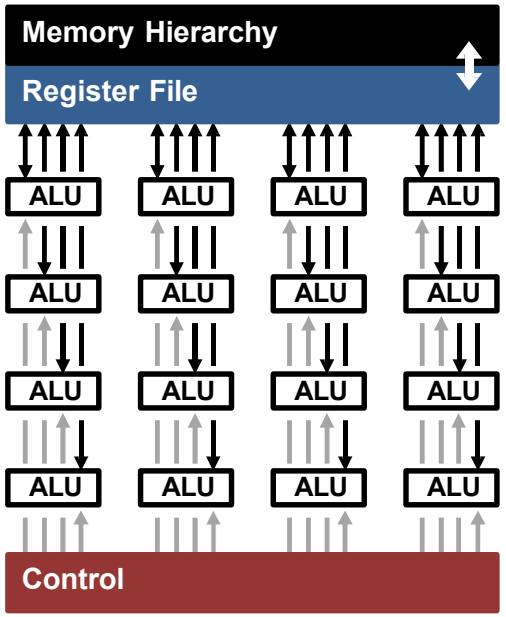
Multiprocessor



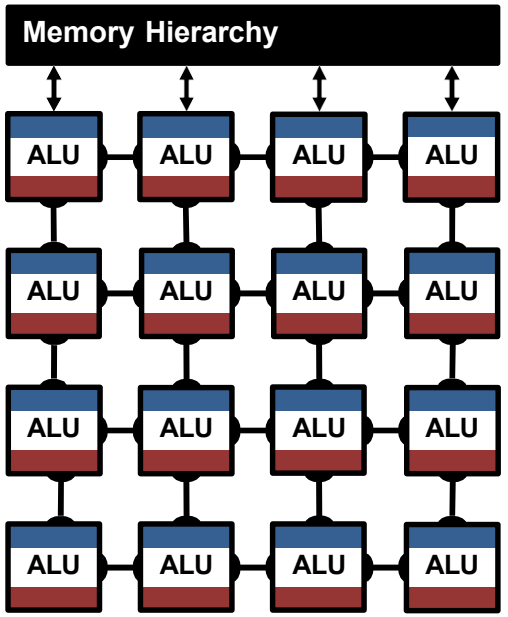
Inter-processing element communication is through cache hierarchy

Highly-Parallel Compute Paradigms

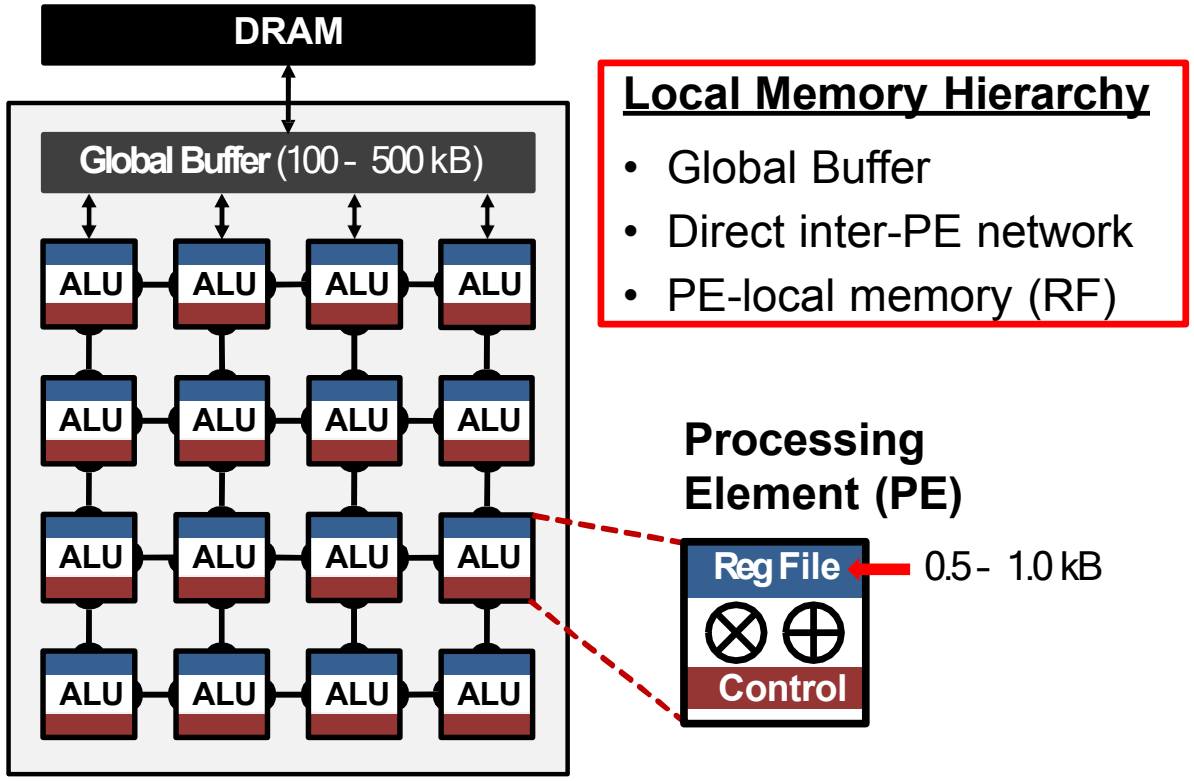
Temporal Architecture (SIMD/SIMT)



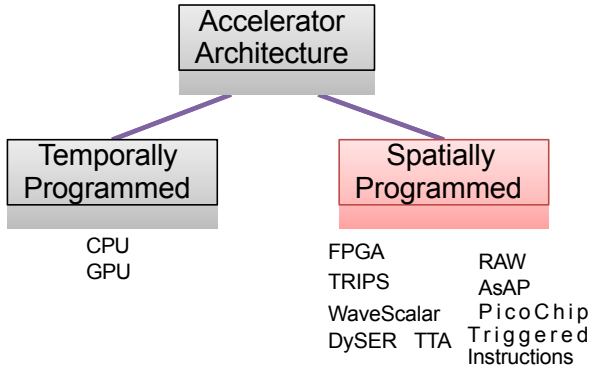
Spatial Architecture (Dataflow Processing)



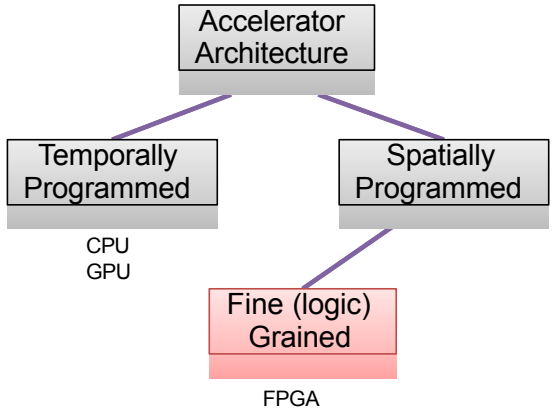
Spatial Architecture for DNN



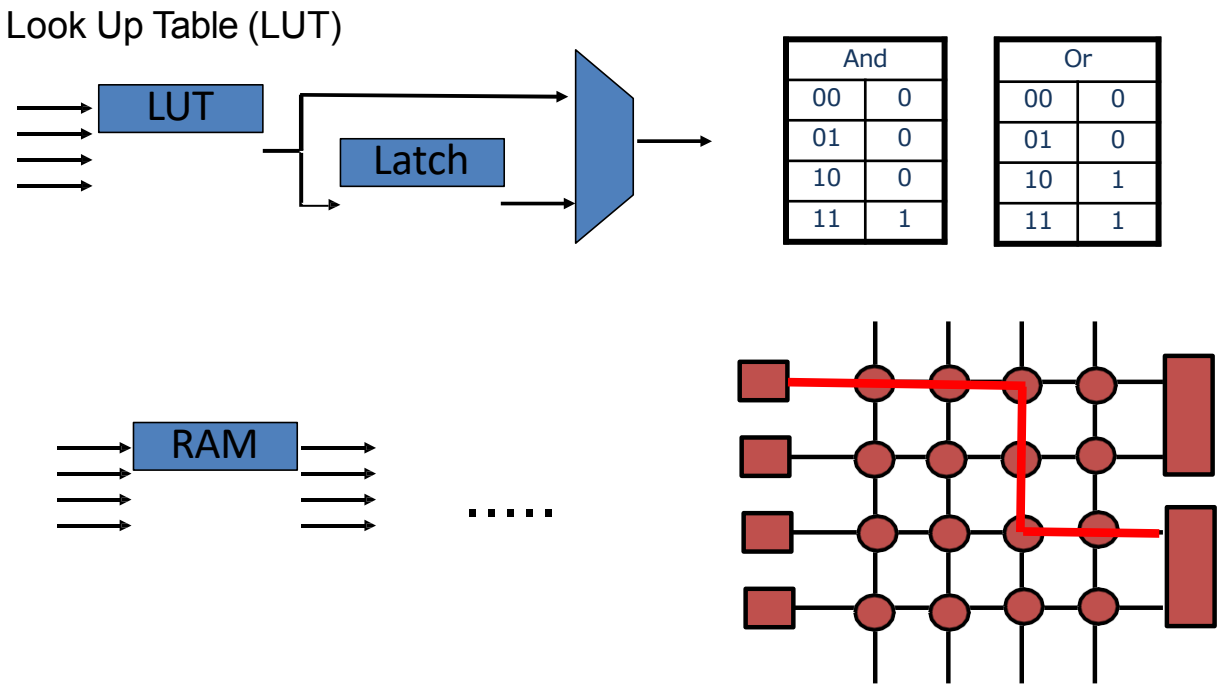
Accelerator Taxonomy



Accelerator Taxonomy



Field Programmable Gate Arrays

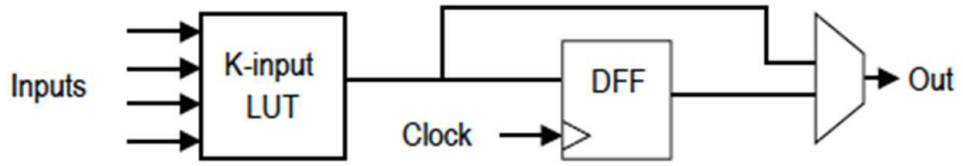


And	
00	0
01	0
10	0
11	1

Or	
00	0
01	0
10	1
11	1

Configurable Logic Blocks (CLB)

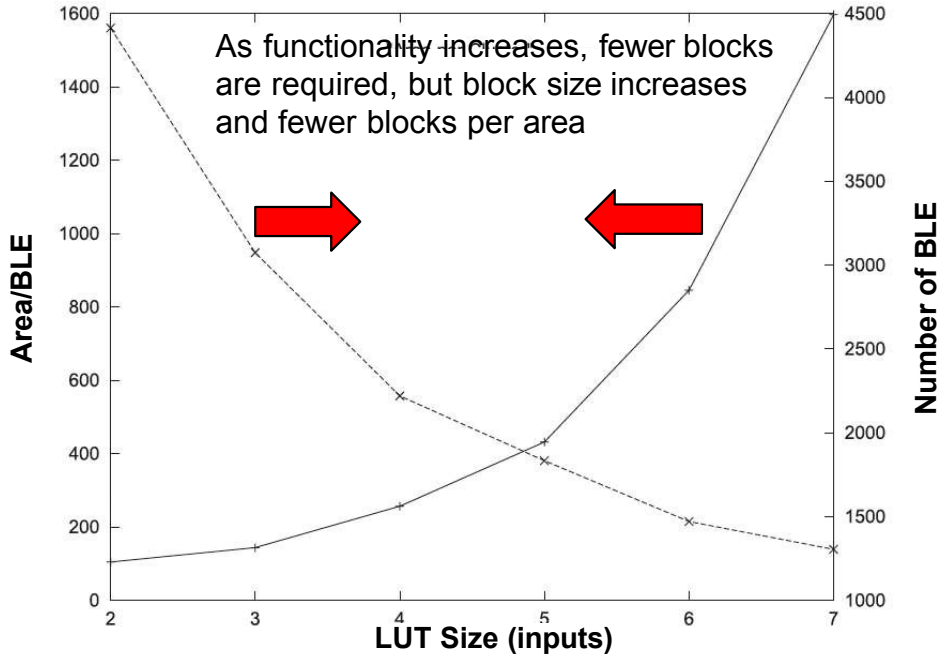
- CLB used to implement sequential and combinational logic
- CLB are comprised of several Basic Logic Elements (BLE)
- Each BLE contains:
  - Look up tables (LUT) are used to implement logic function
  - Registers to store data
  - Multiplexer to select desired output



As number of inputs grow (k), increase size of LUT by 2^k and routing

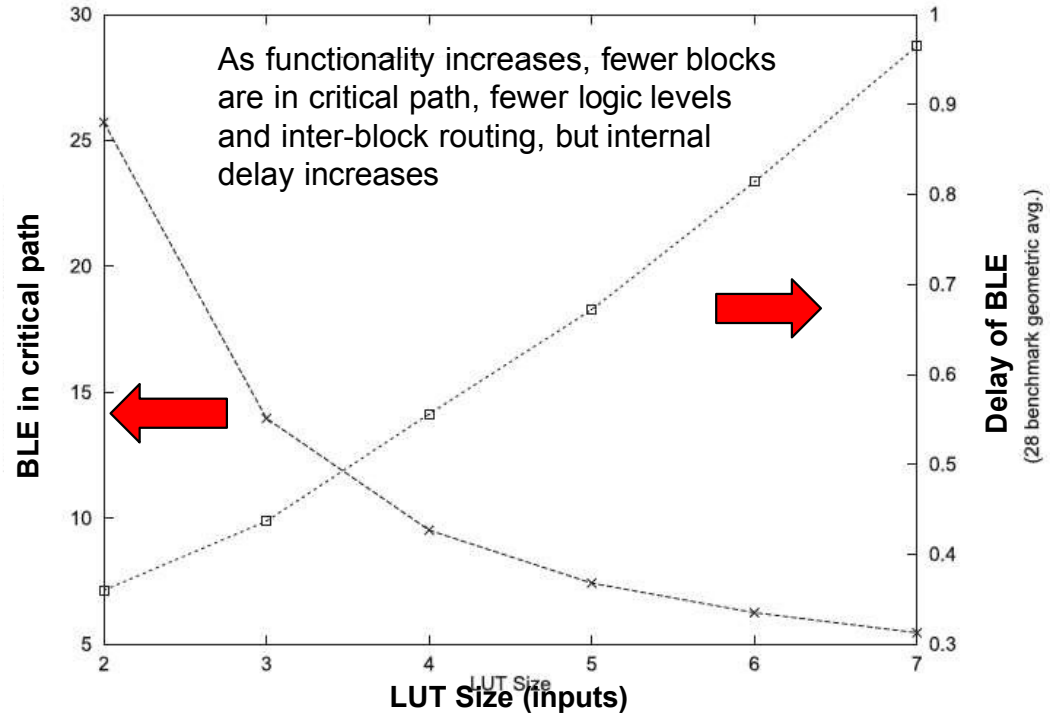
Kuon, Ian, Russell Tessier, and Jonathan Rose. ”FPGA architecture: Survey and challenges." Foundations and Trends in Electronic Design Automation 2.2 (2008): 135-253.

Area Trade-off (Size of LUT)



Kuon, Ian, Russell Tessier, and Jonathan Rose. "FPGA architecture: Survey and challenges." Foundations and Trends in Electronic Design Automation 2.2 (2008): 135-253.

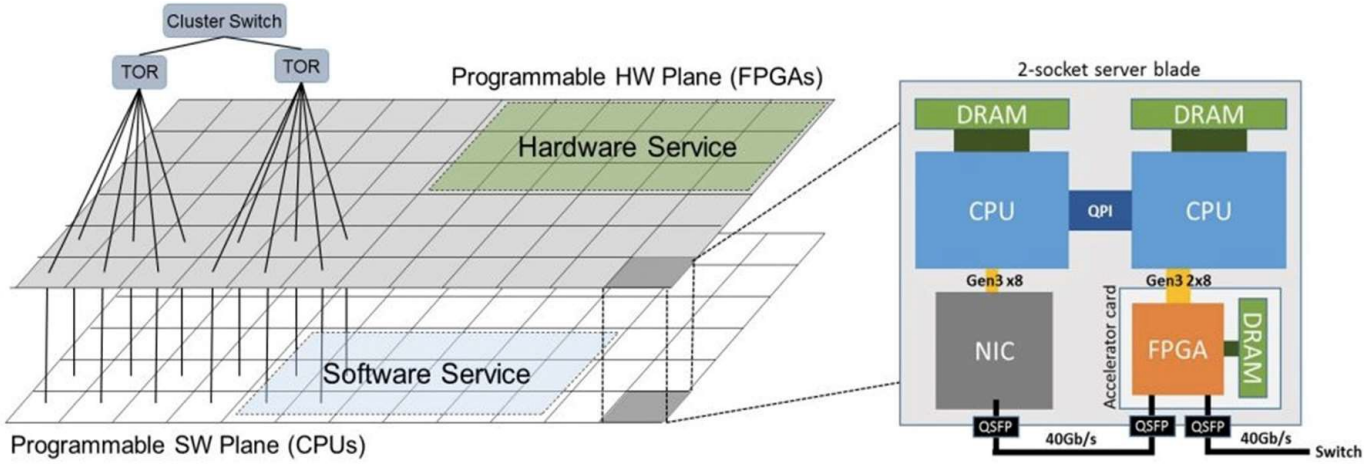
Size of LUT (Speed Trade-off)



Kuon, Ian, Russell Tessier, and Jonathan Rose. "FPGA architecture: Survey and challenges." Foundations and Trends in Electronic Design Automation 2.2 (2008): 135-253.

Microsoft Project Catapult

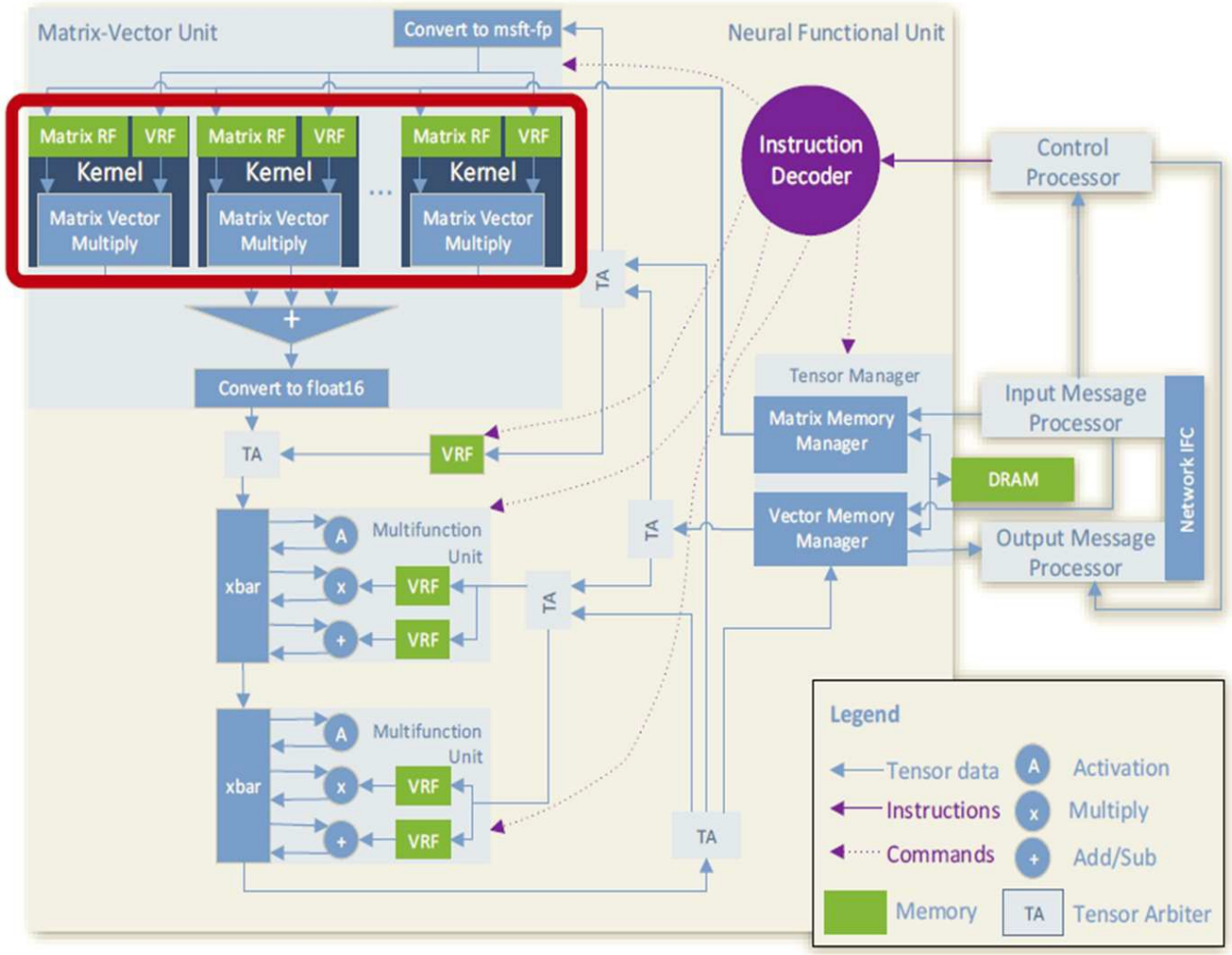
Configurable Cloud (MICRO 2016) for Azure



Accelerate and reduce latency for

- Bing search
- Software defined network
- Encryption and Decryption

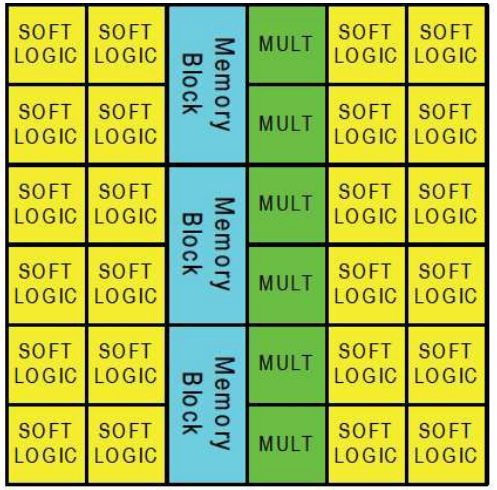
Microsoft Brainwave Neural Processor



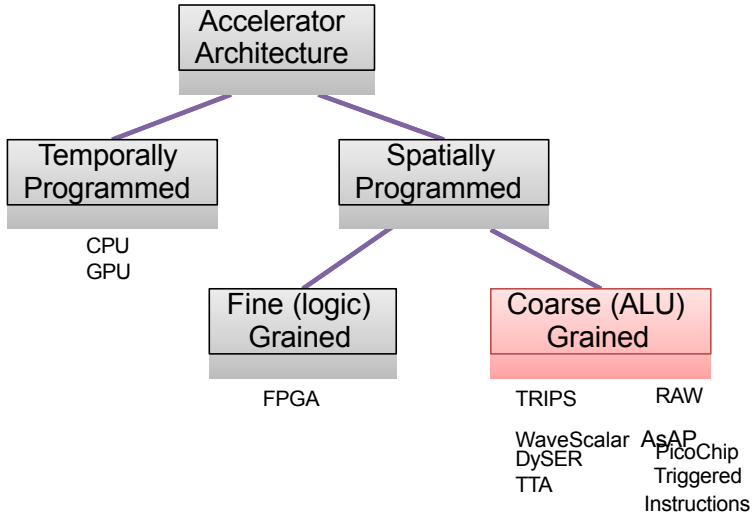
Source: Microsoft

Heterogeneous Blocks

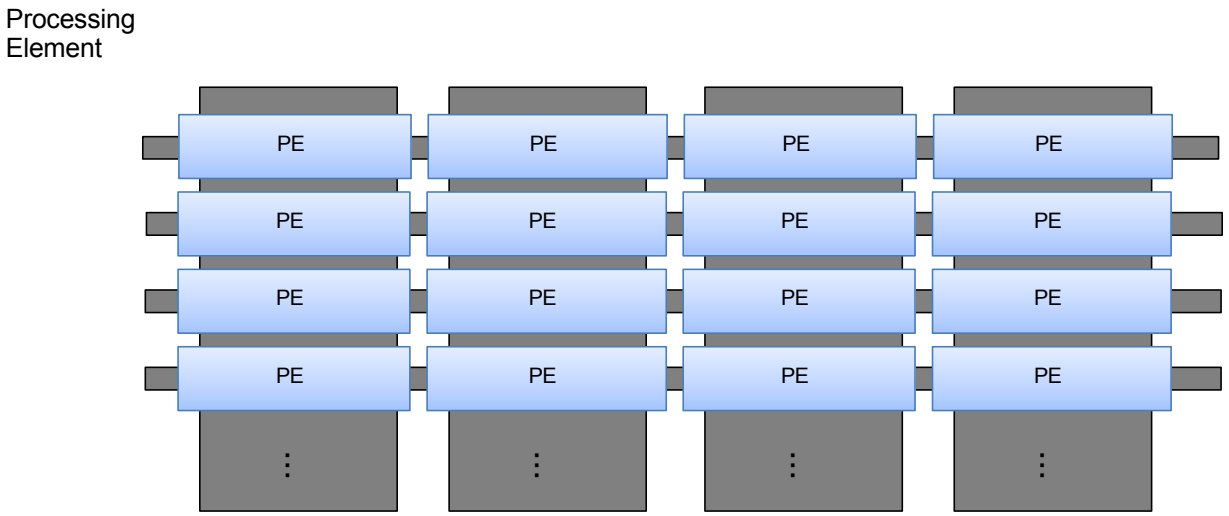
- Add *specific purpose logic* on FPGA
  - Efficient if used (better area, speed, power), wasted if not
- Soft fabric
  - LUT, flops, addition, subtraction, carry logic
  - Convert LUT to memories or shift registers
- Memory block (BRAM)
  - Configure word and address size (aspect ratio)
  - Combine memory blocks to large blocks
  - Significant part for FPGA area
  - Dual port memories (FIFO)
- Multipliers /MACs → DSP
- CPUs and processing elements



Accelerator Taxonomy

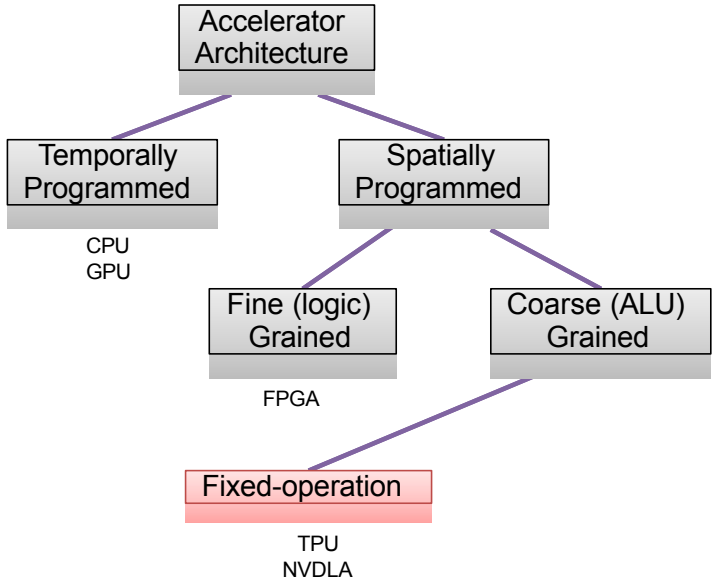


Programmable Accelerators



Many Programmable Accelerators look like an array of PEs, but have dramatically different architectures, programming models and capabilities

Accelerator Taxonomy

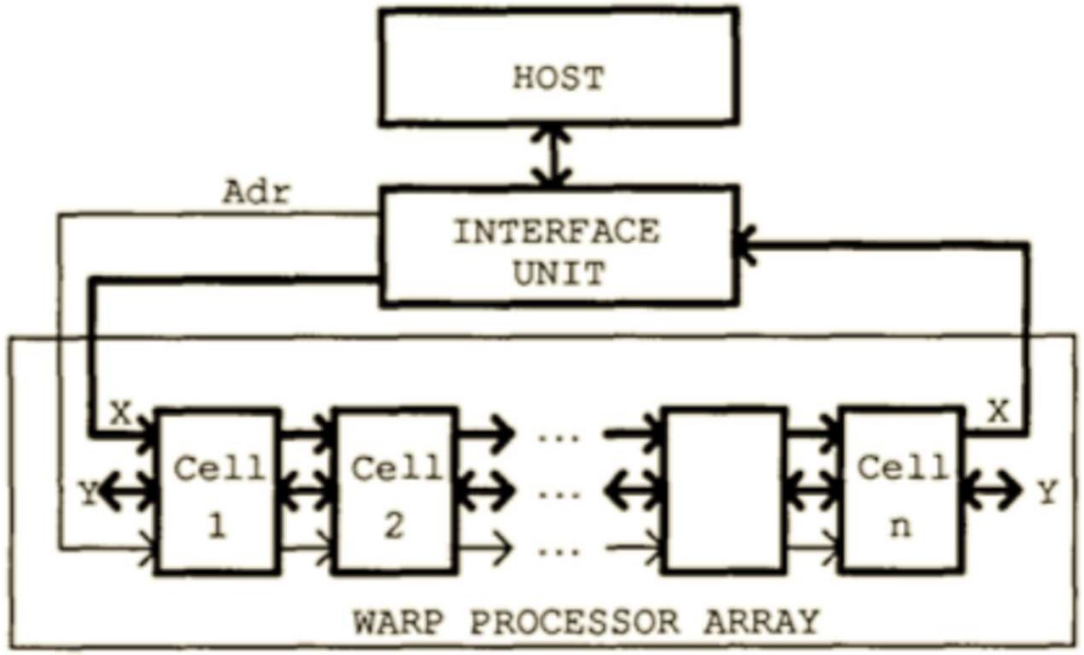


Fixed Operation - Systolic Array

- Each PE hard-wired to one operation
- Purely pipelined operation
  - no backpressure in pipeline

- Attributes
  - High-concurrency
  - Regular design, but
  - Regular parallelism only!

Configurable Systolic Array - WARP



Source: WARP Architecture and Implementation, ISCA 1986

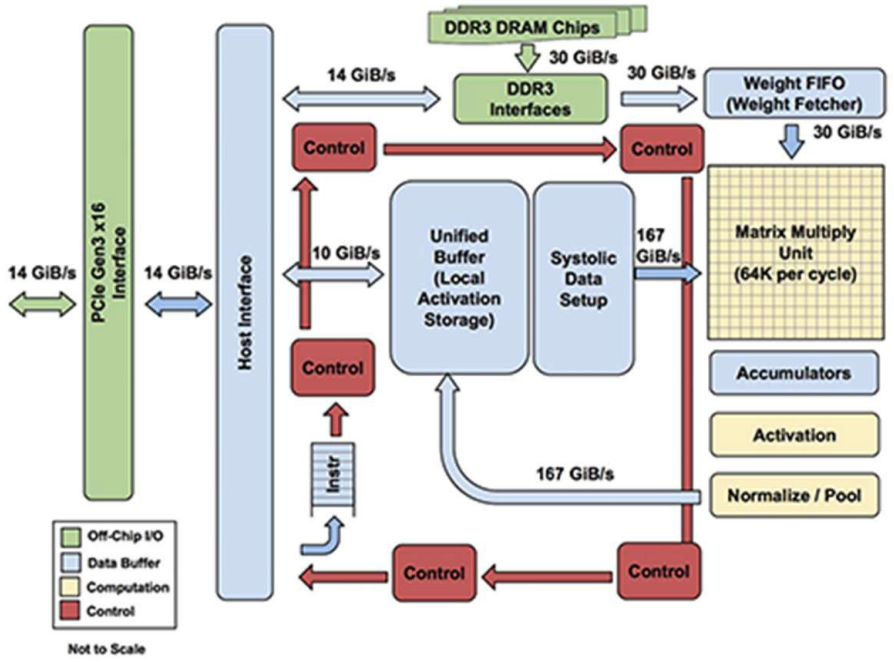
Fixed Operation - Google TPU

Where is TPU Used?

Google AI workloads
(e.g., Google Search, Translate, Photos)

TensorFlow-based models (TPUs are optimized for TensorFlow)

Large-scale deep learning training and inference
(e.g., BERT, Vision Transformers)

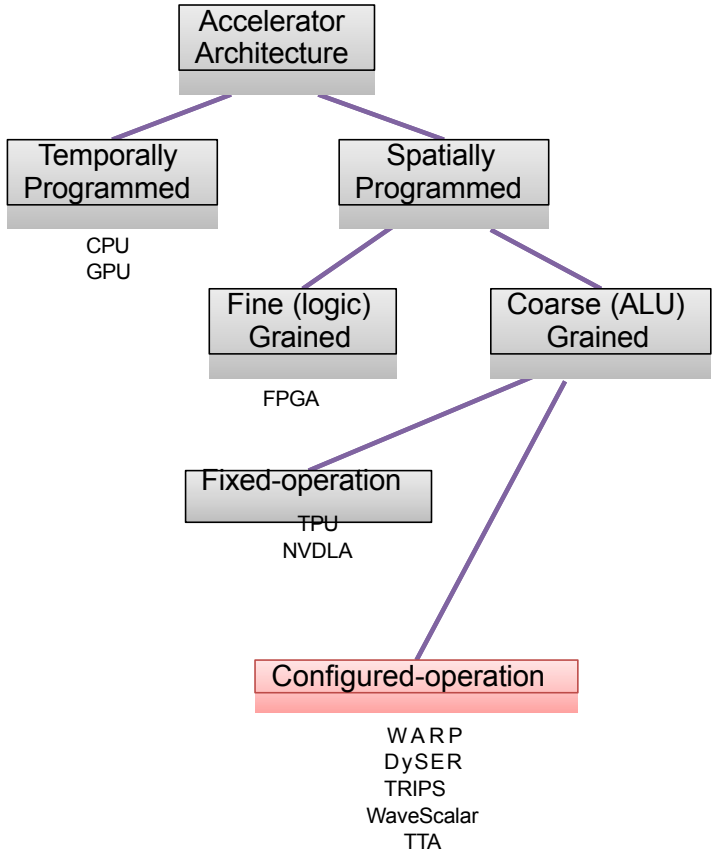


Systolic array does 8-bit 256x256 matrix-multiply accumulate

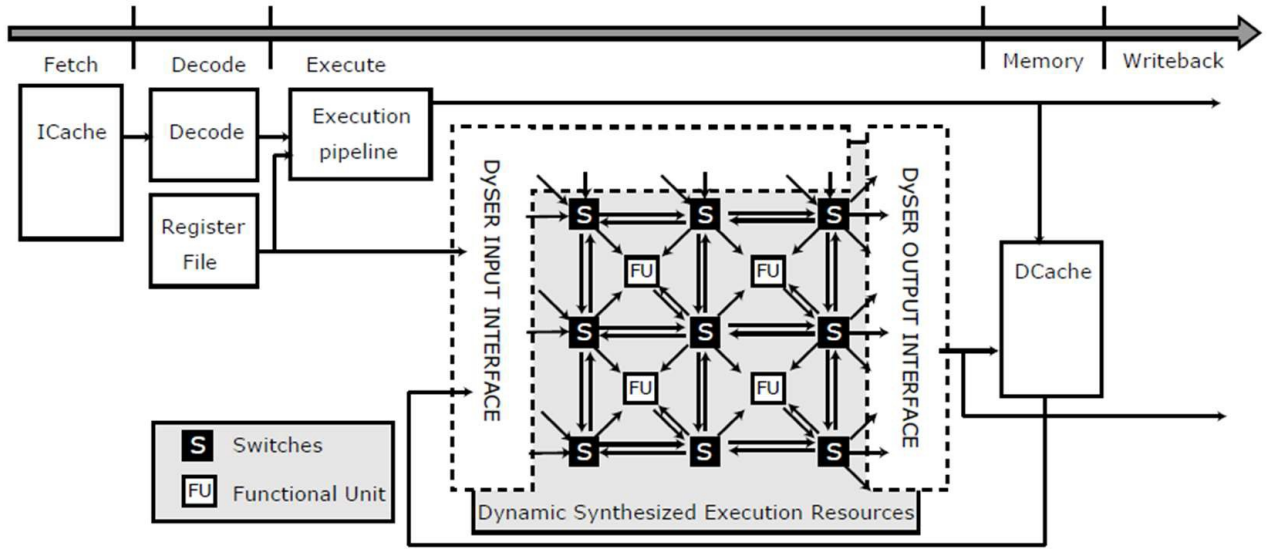
Source: Google

accelerates matrix multiplications, reducing data movement and maximizing throughput.

Accelerator Taxonomy



Single Configured Operation - Dyser



Source: Dynamically Specialized Datapaths for Energy Efficient Computing. HPCA11

Dyser Architecture Overview

Coarse-Grained Reconfigurability

Dyser can dynamically reconfigure its execution units to match different computational tasks, focusing on specific workloads like matrix multiplications, convolutions, or encryption.

Acceleration of Specific Workloads

Dyser accelerates tasks such as matrix multiplications, convolutions, or encryption by tailoring its execution units.

Integration with CPU Pipeline

Dyser is tightly coupled with a conventional CPU pipeline and acts as a co-processor, offloading computation-heavy tasks from the main processor.

Network of Execution Units

Contains a network of small execution units (e.g., ALUs or Multiply-Accumulate units) that can connect dynamically to form larger computational structures.

Functional Unit Flexibility

The execution units in Dyser provide flexibility, dynamically connecting to perform complex operations efficiently, optimizing resource use.

Offloading Computation Tasks

By acting as a co-processor, Dyser alleviates the main processor's workload, enhancing system performance and efficiency.

Dyser

Efficient Dataflow Processing:

The architecture is **optimized for dataflow computing**, meaning it minimizes unnecessary data movement.

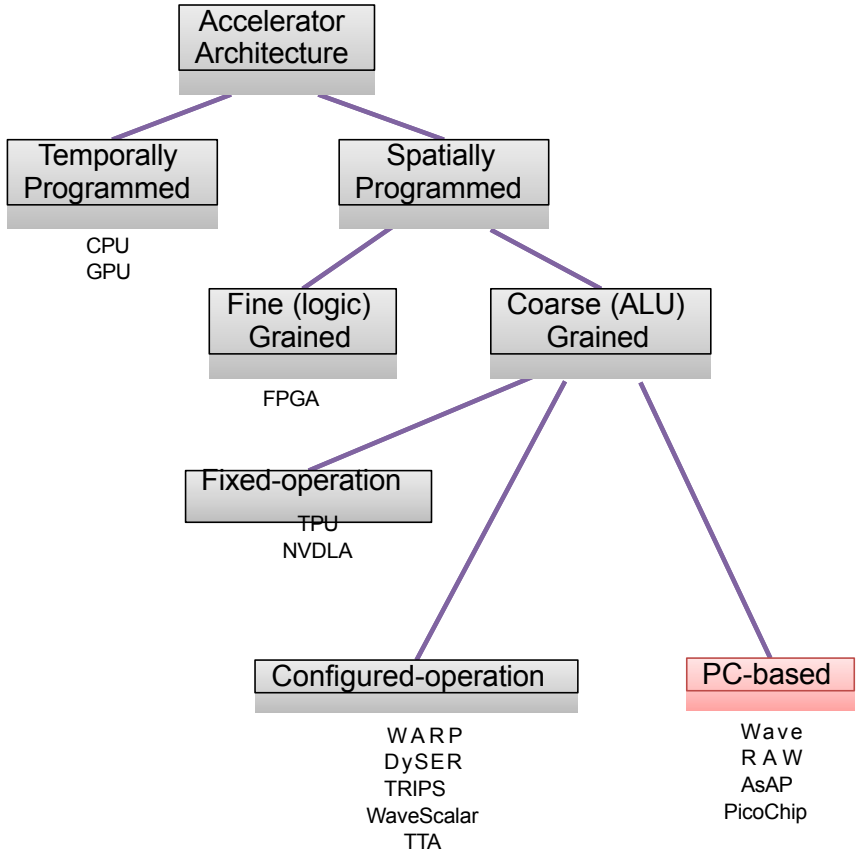
Reduces **memory access overhead**, improving energy efficiency.

Application-Specific Speedup:

Dyser provides significant speedup for workloads like:

- Cryptography (AES, SHA)
- Signal Processing (FFT, DCT)
- Machine Learning (Matrix operations)
- Scientific Computing (Graph algorithms, Linear Algebra)

Accelerator Taxonomy



PC-based Control – Wave Computing