R265: Advanced Topics in Computer Architecture

Seminar 7: HW accelerators and accelerators for machine learning

Robert Mullins

This lecture

- Computer architecture trends
- Hardware accelerators
  - Design choices and trade-offs
- Hardware accelerators for machine learning (briefly)
- Challenges

Trends in Computer Architecture

Time



Early computers	Gains from bit-level parallelism
Pipelining and superscalar issue	+ Instruction-level parallelism
Multicore / GPUs	+ Thread-level parallelism / data-level parallelism
Greater integration (large SoCs), heterogeneity and specialisation	+ Accelerator-level parallelism

Note: Memory hierarchy developments have also been significant. The memory hierarchy typically consumes a large fraction of the transistor budget.

Power limited design

- Today we often need to look beyond general-purpose programmable processors to meet our design goals
- We trade flexibility for efficiency
- Optimise for a narrower workload
- These “accelerators” can be 10-1000x better than a general-purpose solution in terms of power and performance

Specialisation

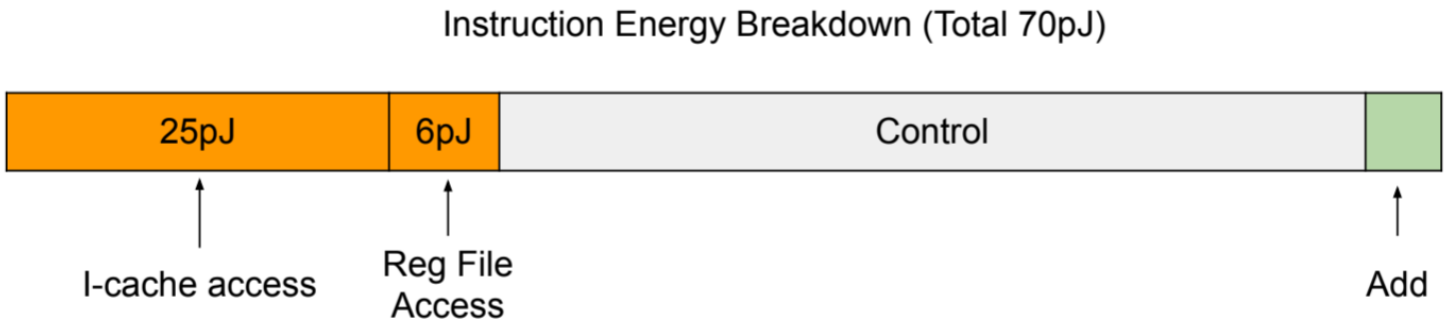
What does specialisation allow us to do?

- Remove infrequently used parts of the processor
- Tune instruction set for common operations or replace with hardwired control
- Exploit forms of parallelism abundant in the application(s) – we often see a specialised processing element and local memory reproduced many times.
  - Can we also accelerate irregular programs?
- Instantiate specialised memories and tune their widths and sizes
- Provide specialised interconnect between components
- Optimise data-use patterns
  - Memory hierarchies, tiling, exploit opportunities for multi-cast/broadcast

The “Turing tax”

Floating Point Arithmetic	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1pJ
32 bit	4pJ

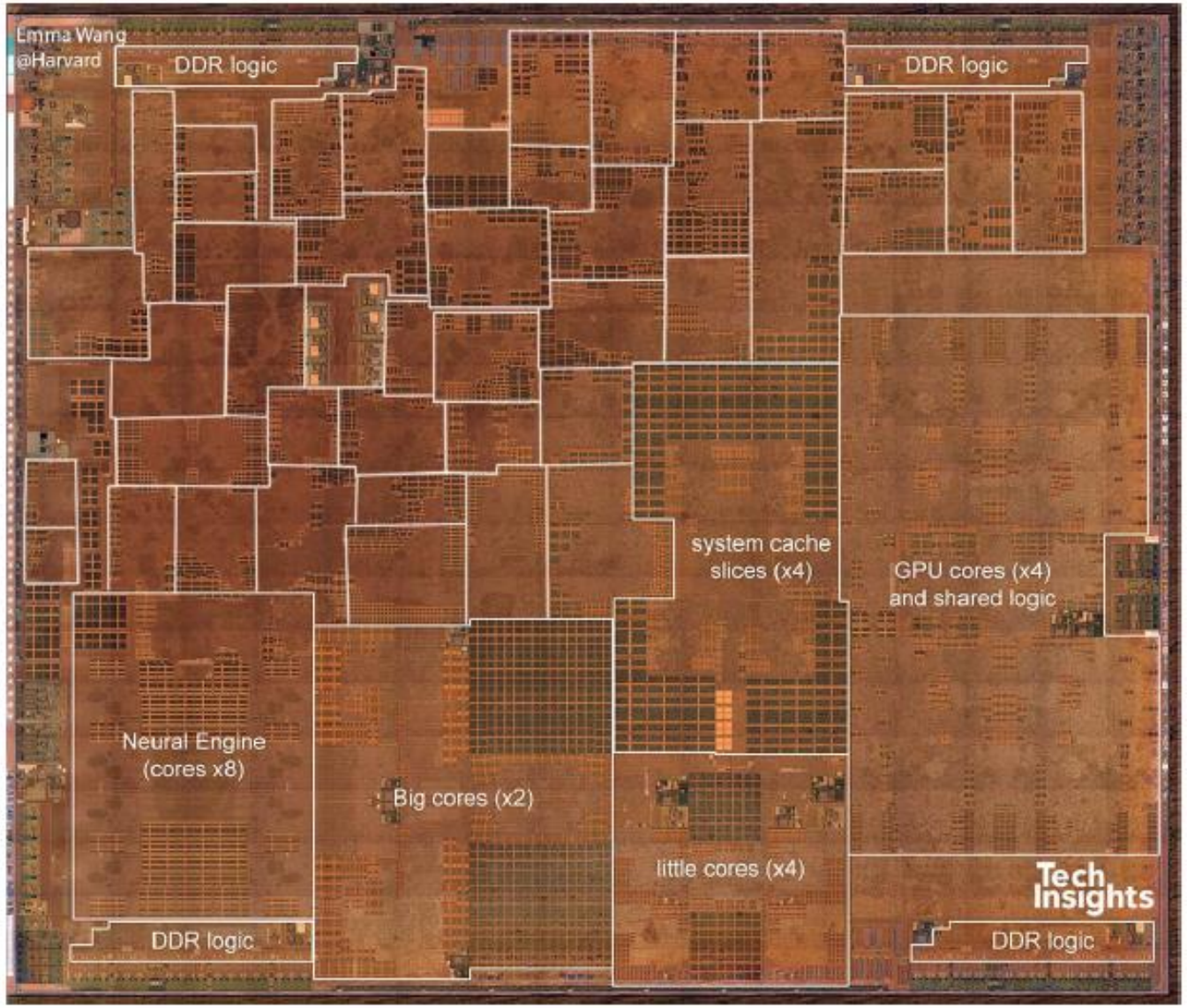
Memory	
Cache	(64 bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3 - 2.6nJ



Data assumes a 45nm process @0.9V, source: “Computing’s energy problem (and what we can do about it)”, Mark Horowitz, ISSCC 2014

Apple A12 SoC

- 2019
- 40+ accelerators



Design-space continuum

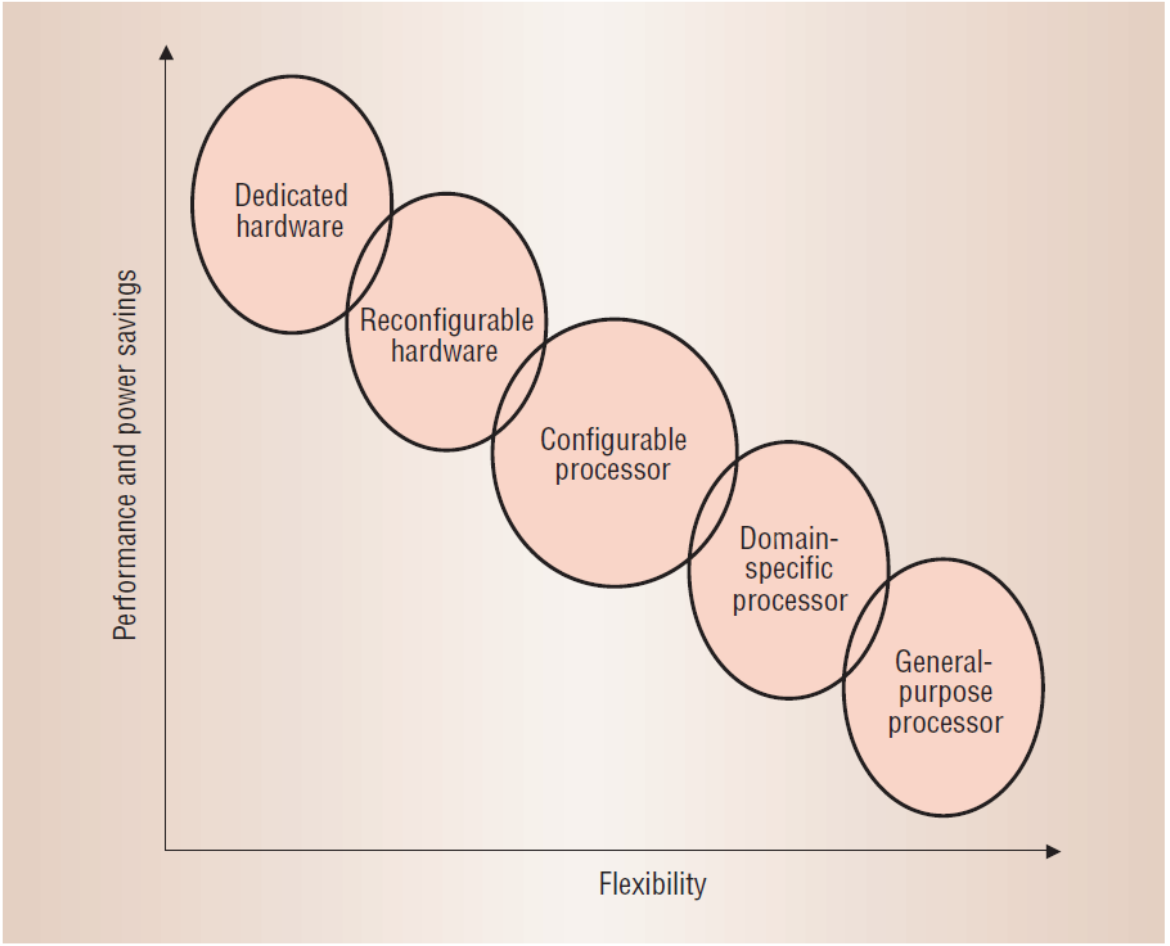
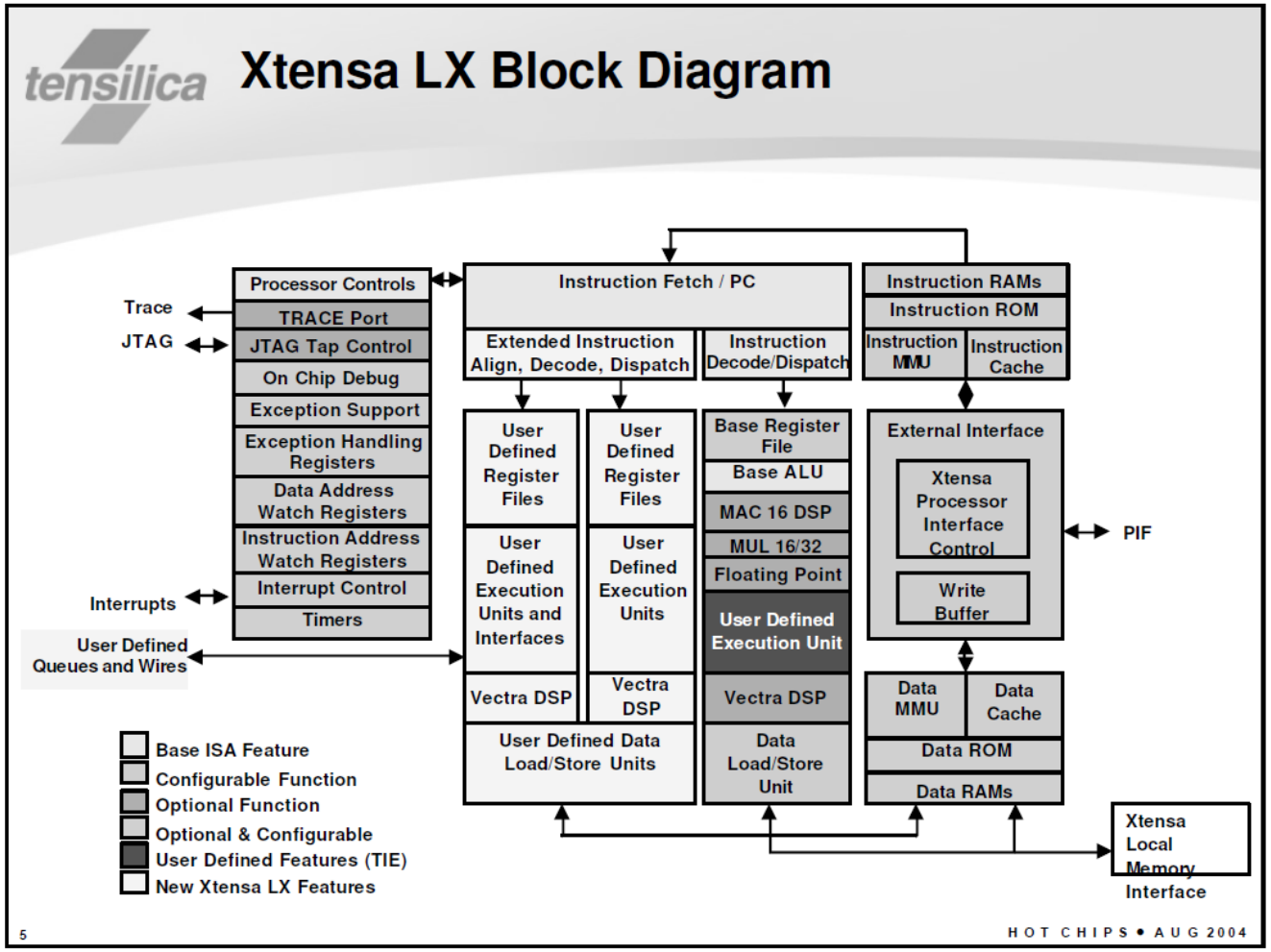


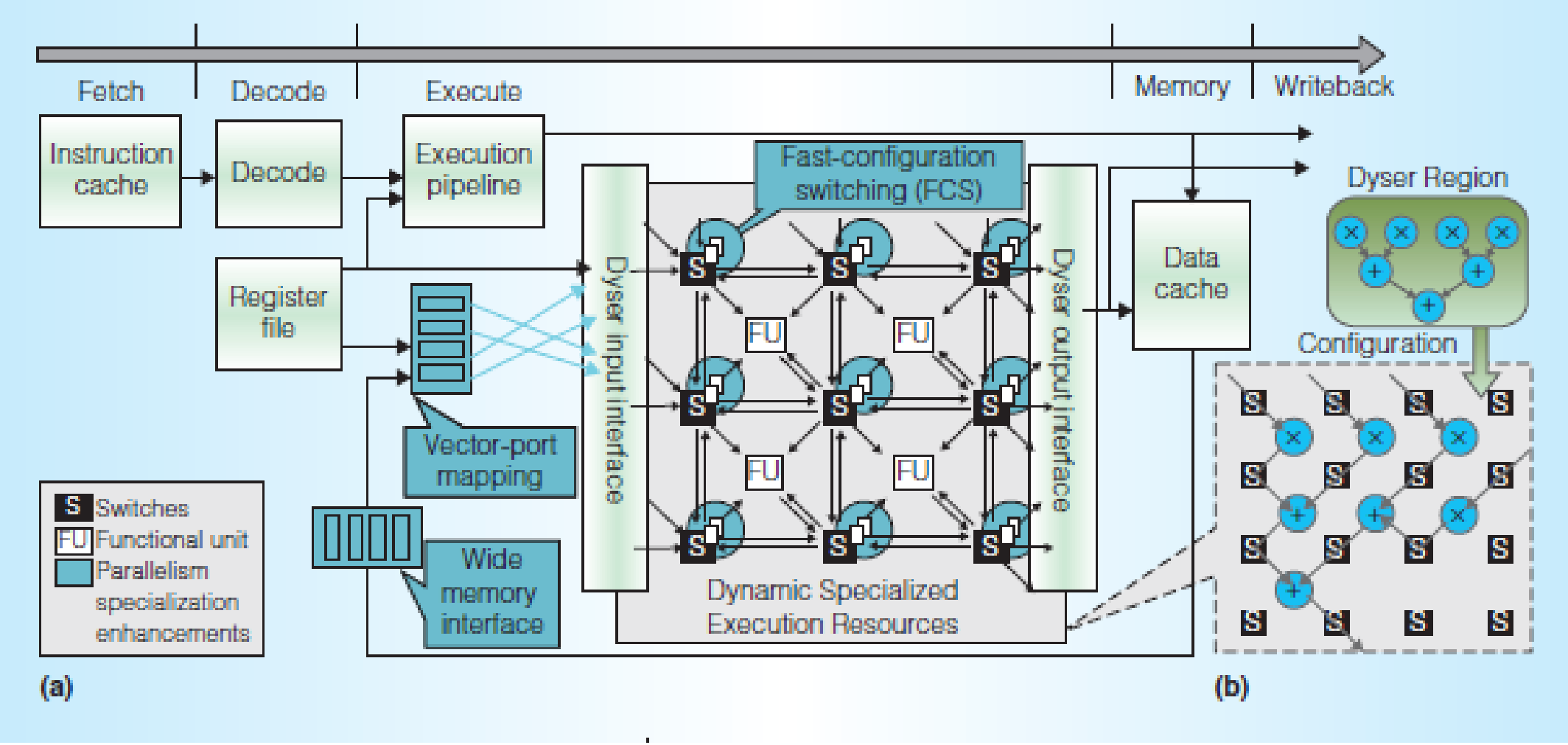
Figure 1. Embedded SoC design-space continuum trades the performance and power savings of dedicated hardware for the flexibility of software-based solutions.

Reproduced from "*configurable processors for embedded computing*", Dutt and Choi, IEEE Computer, vol 36, issue 1, 2003, pp. 120-123

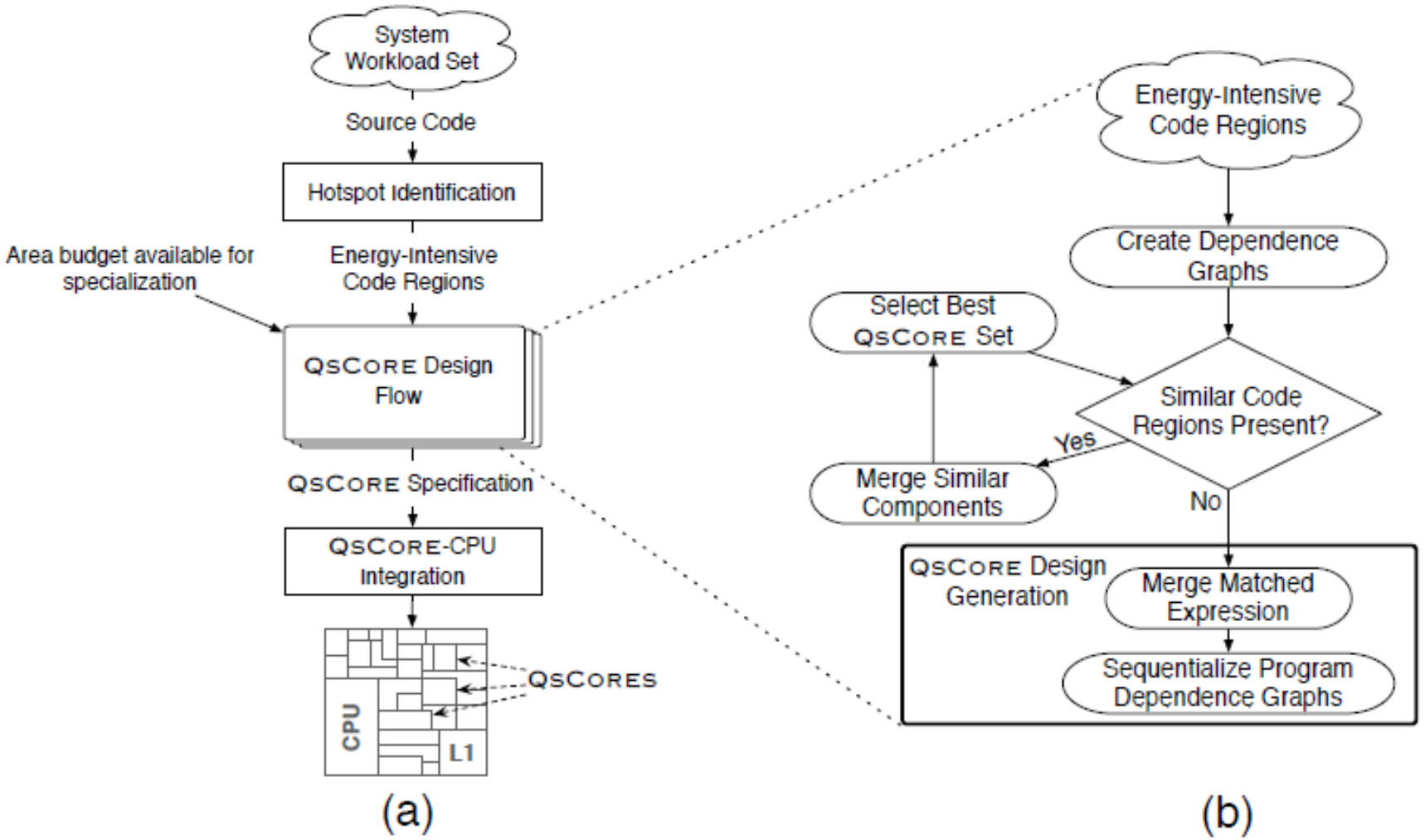
Configurable processors (Tensilica/Cadence)



Dynamically specialised execution resources (DYSER, IEEE Micro 2012)



Quasi-Specific cores (QSCOREs) [Micro 2011]



QSCOREs generated using C-to-HW compiler

Compiler builds HW datapath and control state machine based on data and control flow graphs

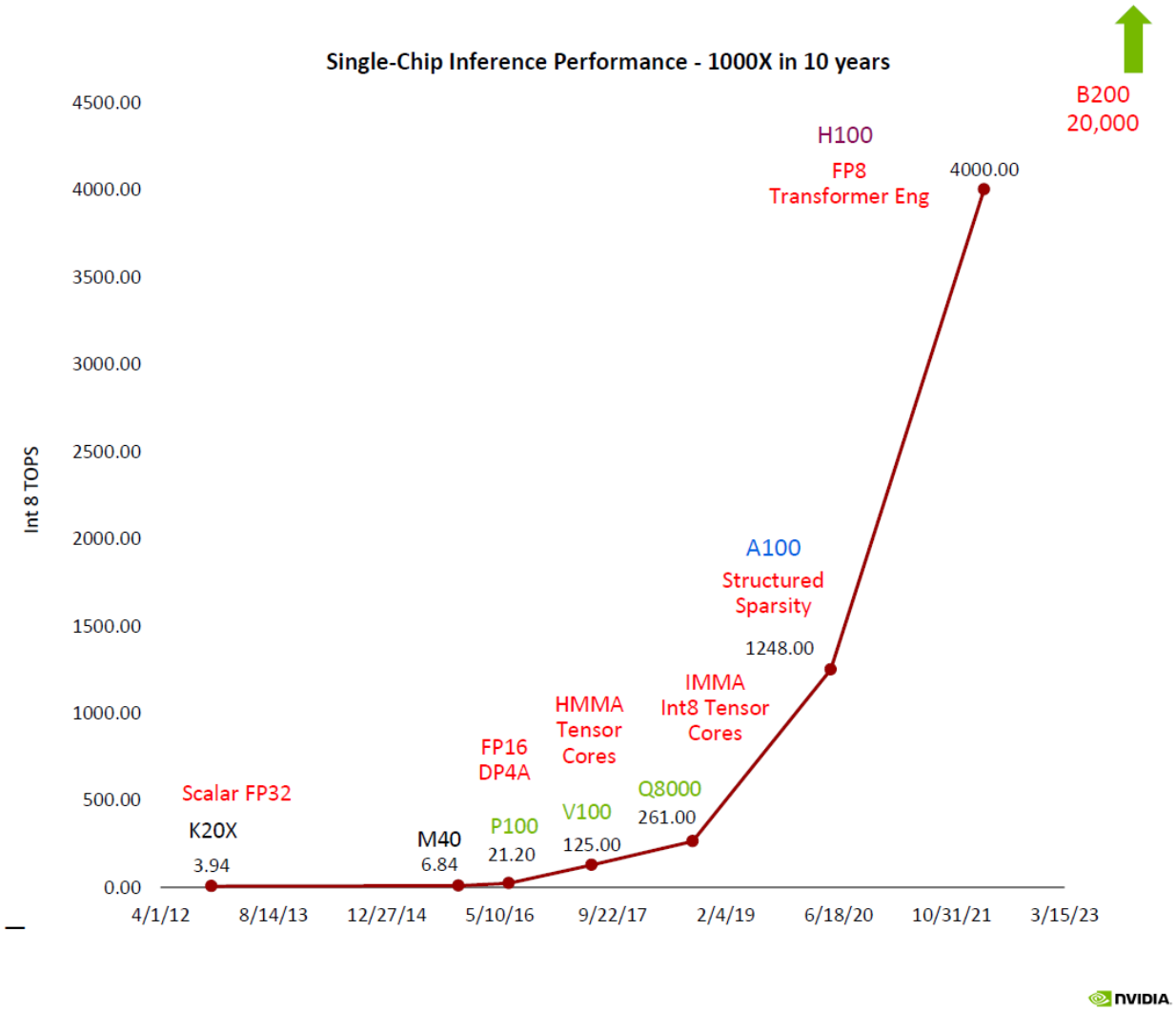
Some degree of configurability allows a single QSCORE to execute similar code segments

Memory operations access same data cache as GPP

Machine Learning (GPUs)

Gains from

- Number Representation
  - FP32, FP16, Int8, FP4
  - (TF32, BF16)
  - ~16x, 32x
- Complex Instructions
  - DP4, HMMA, IMMA
  - ~12.5x
- Process
  - 28nm, 16nm, 7nm, 5nm, 4nm
  - ~2.5x, 3x
- Sparsity ~2x
- Die Size 2x
- Model efficiency has also improved – overall gain > 1000x



Source: Bill Dally, NVIDIA

Specialised instructions

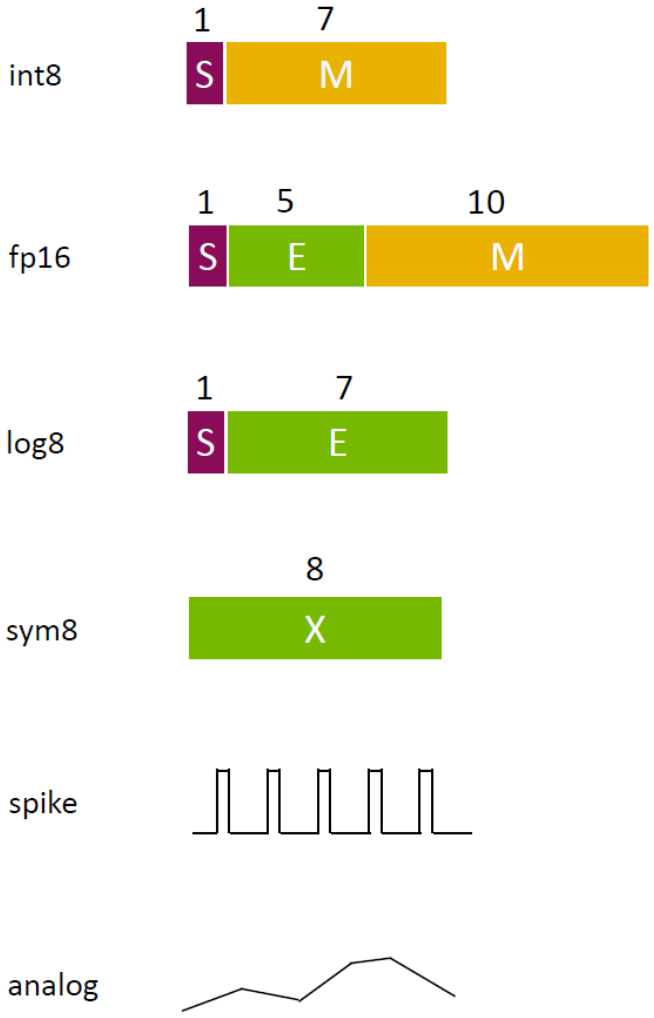
- Half-precision MAC
- Half-precision Dot Product 4-Add
- Half-precision Matrix Multiply-Accumulate
- Integer Matrix Multiply-Accumulate

(overhead is in fetch, decode, operand fetch etc. Energy numbers normalised to 45nm process)

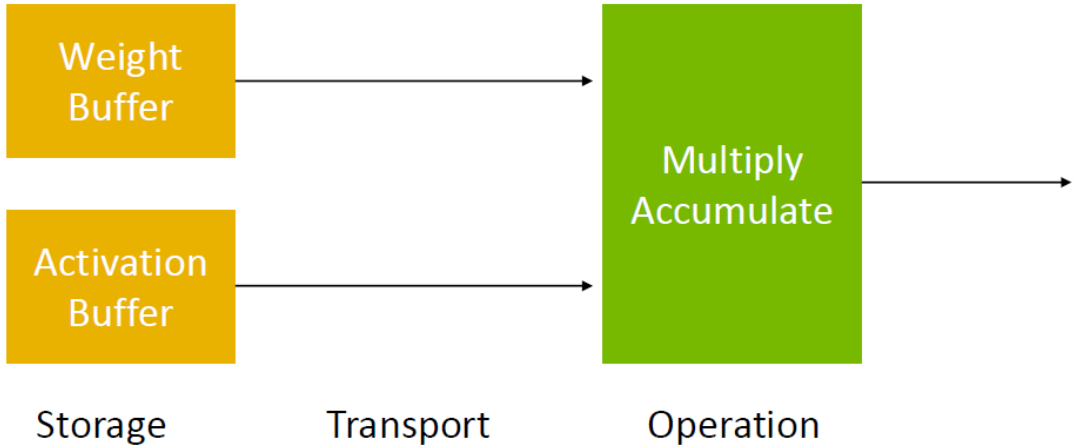
Operation	Energy**	Overhead*
HFMA	1.5pJ	2000%
HDP4A	6.0pJ	500%
HMMA	110pJ	22%
IMMA	160pJ	16%

Source: Bill Dally, NVIDIA

Number representations

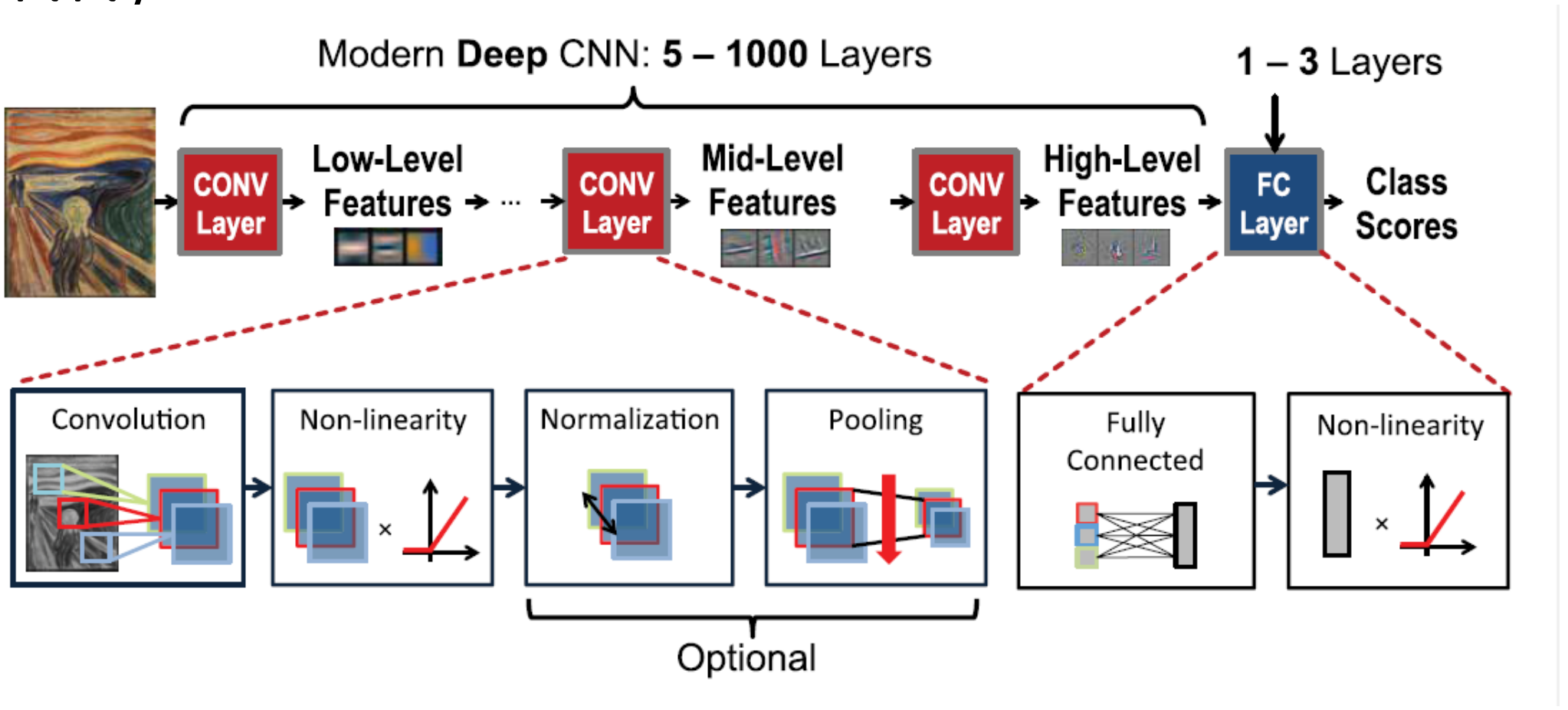


- Attributes:
  - Cost
    - Operation energy
    - Movement energy
  - Accuracy
    - Dynamic range
    - Precision (error)



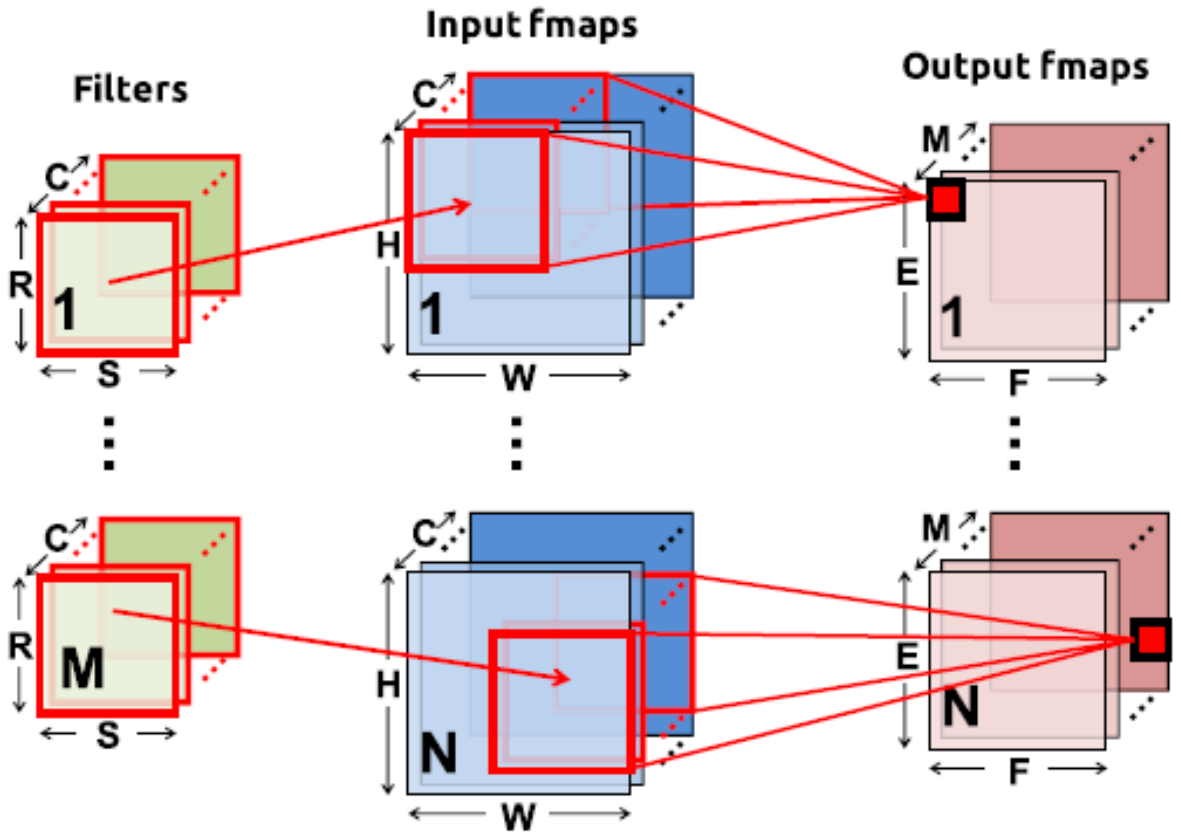
NVIDIA

Hardware accelerators for machine learning (CNN)



Reproduced from: Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, vol. 105, no. 12

Hardware accelerators for machine learning (CNNs)

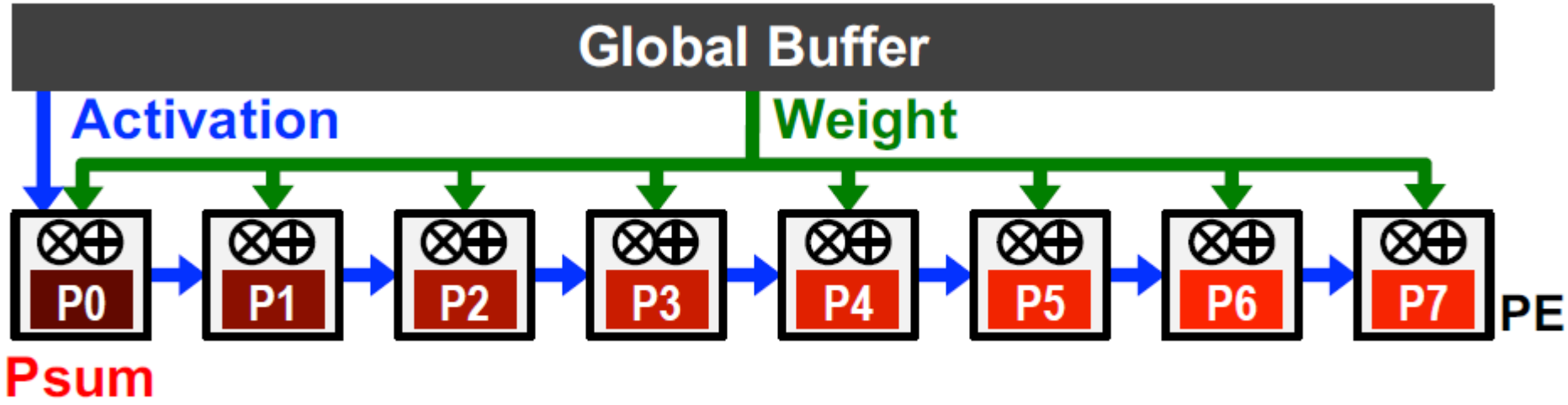


Reproduced from: Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, vol. 105, no. 12

Data reuse patterns

- Memory access is likely bottleneck – very large volumes of data
  - Weights, activations, and gradients if training
- How can we avoid this?
  - Make best use of local memory (reuse data values)
  - Broadcast data values
  - Careful data tiling to maximise benefits of multi-level memories
- Need to select a particular “dataflow”

Example dataflow: output stationary



- Broadcast filter weights
- Reuse activations

Hardware accelerators for machine learning

- IoT
- Mobile
- Edge
- Server

Training at the edge is now more common – edge compute represents enormous distributed compute resource

Interesting work allowing some training even on microcontrollers

Challenges

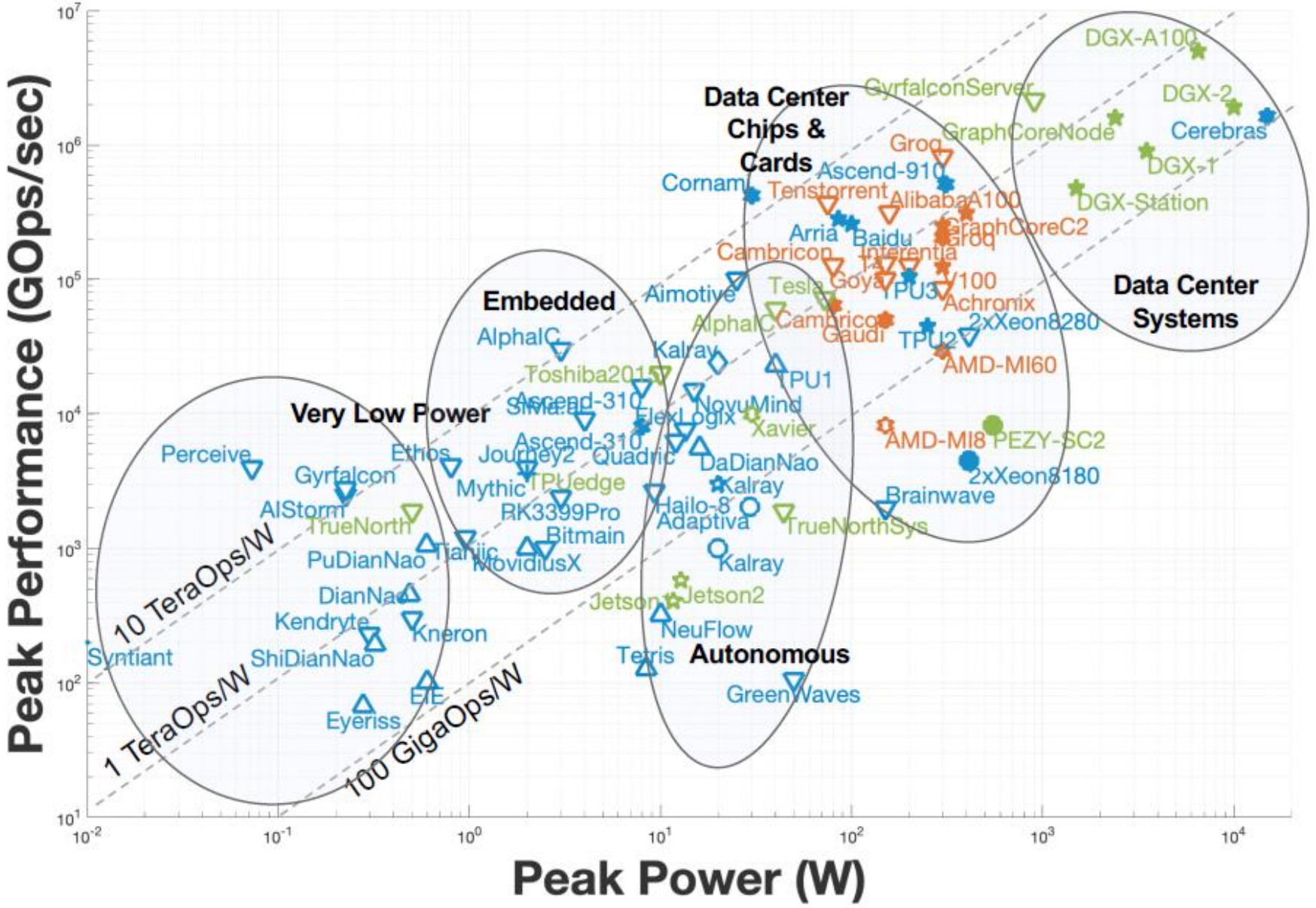
- Workload can vary
  - Already many different types of network and associated optimisations
    - e.g. sparse vs. dense, convolutional layers vs. fully-connected layers etc.
    - Transformers
  - Problem size and required performance may differ
    - At edge, we see the use of “tiny”, “small” and “big” NPUs (as we do for processors)
- Workload is still evolving
  - New network types being developed
  - more dynamic and conditional behaviours etc., greater sparsity?
- Add flexibility to our accelerator?
  - Risk is that flexibility reduces performance and efficiency
  - We can instantiate multiple accelerators optimised for different cases (but will make design larger or reduce peak performance)
  - Computer architecture is always a trade-off!

Challenges

- NPU architectures?
  - How are PEs connected (i.e. local interconnect)
  - How much local buffering or SRAM?
  - Monolithic vs. tiled? Heterogeneous HW?
  - Support for different network types?
  - Support for dynamic network behaviours?
  - ….
- An interesting co-design problem
  - Search for the best neural network architecture and hardware together
- Is the hardware the easy part?
  - Don’t want to make compiler / mapping software’s task too difficult

Machine learning accelerators: peak perf. vs peak power

(Reuther et. al 2020)



ARIA Scaling Compute Programme

- Goal: >1000 lower costs
- Simulators
- Advanced Networking and Interconnect
  - E.g. novel optical interconnects for the datacentre
- New Computational Primitives
  - Low-precision and back-propagation-free algorithms (sparse updates?)
  - Thermodynamic hardware accelerator
    - physics-inspired compute – encode problem as a physic system, evolve towards the low-energy solution
  - Analogue (equilibrium propagation) / mixed signal / in-memory / neuromorphic

aria.org.uk/scaling-compute

Final thoughts

- Better tools –
  - Specialisation puts pressure on us to create new designs
  - Can we undertake computer architecture much more rapidly?
  - Complex co-design problems – some projects are using machine learning help search and explore the combined design space.
- Redefine the HW/SW interface? Do we over specify at the moment?
  - Computer Architecture Research Centre PhD studentships!
  - www.cl.cam.ac.uk/research/cascade
- When we add accelerators to our system does this change the workload of our general-purpose cores?
- Specialisation isn't immune to the concept of diminishing returns[1]

[1] "The Accelerator Wall: Limits of Chip Specialization", HPCA 2019