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
- 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?

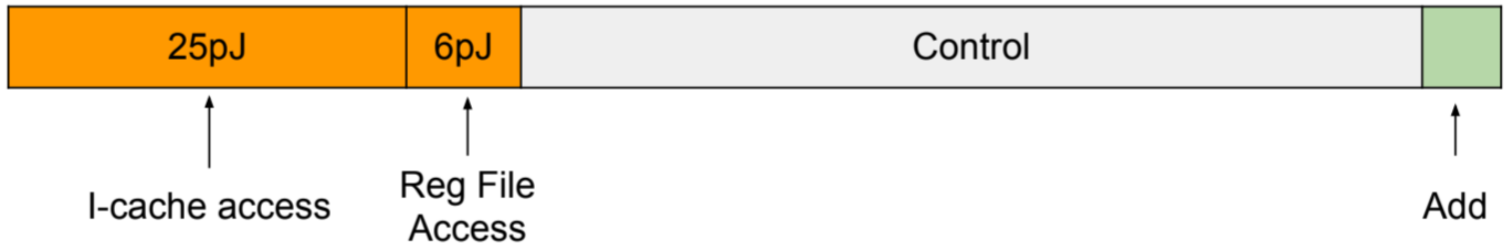
- Remove infrequently used parts of the processor
- Tune instruction set for common operations or replace with hardwired control
- Exploit forms of parallelism abundant in the application(s) – we often see a specialised processing element and local memory reproduced many times.
    - Can we also accelerate 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

Specialisation

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

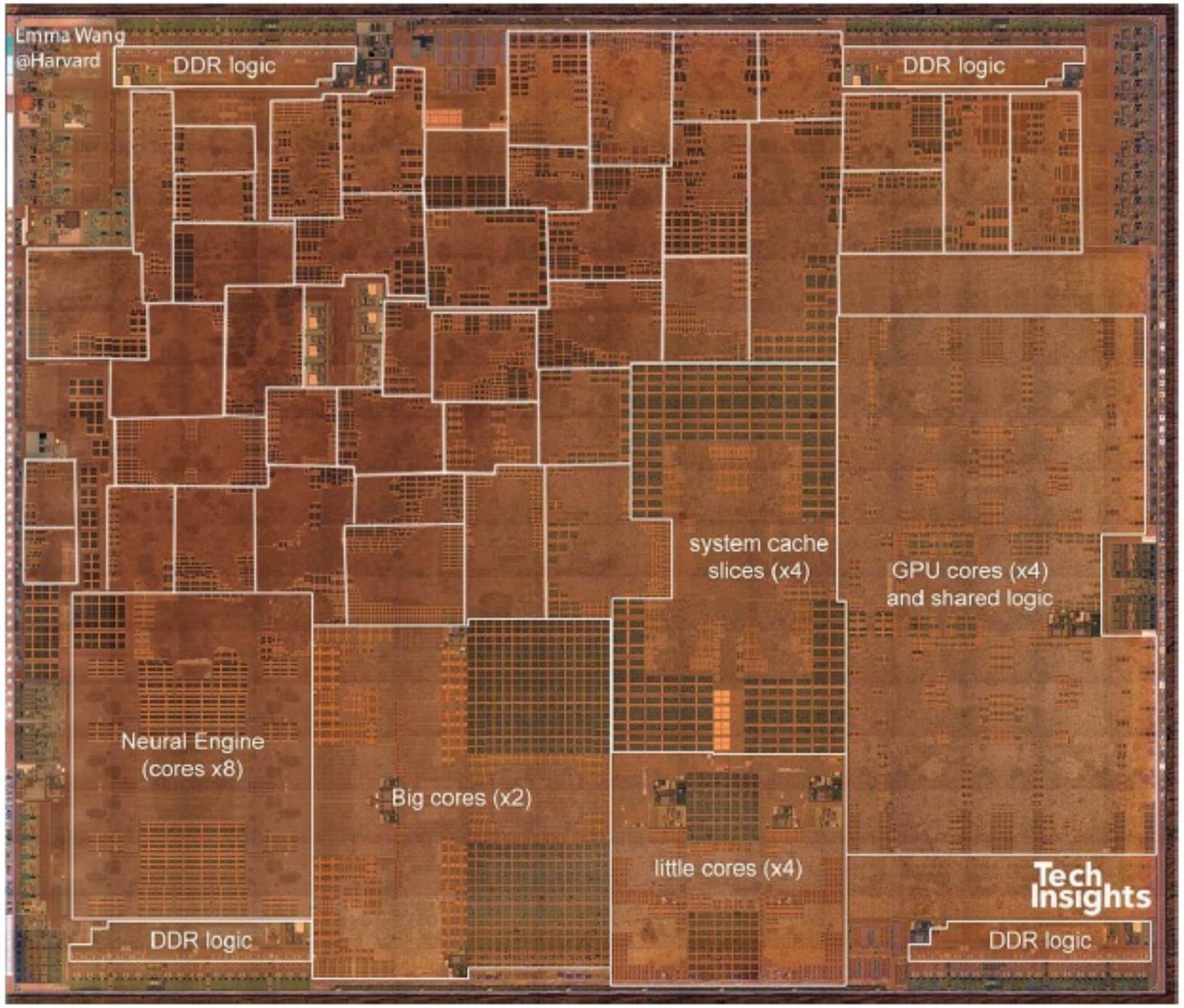
Instruction Energy Breakdown (Total 70pJ)



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

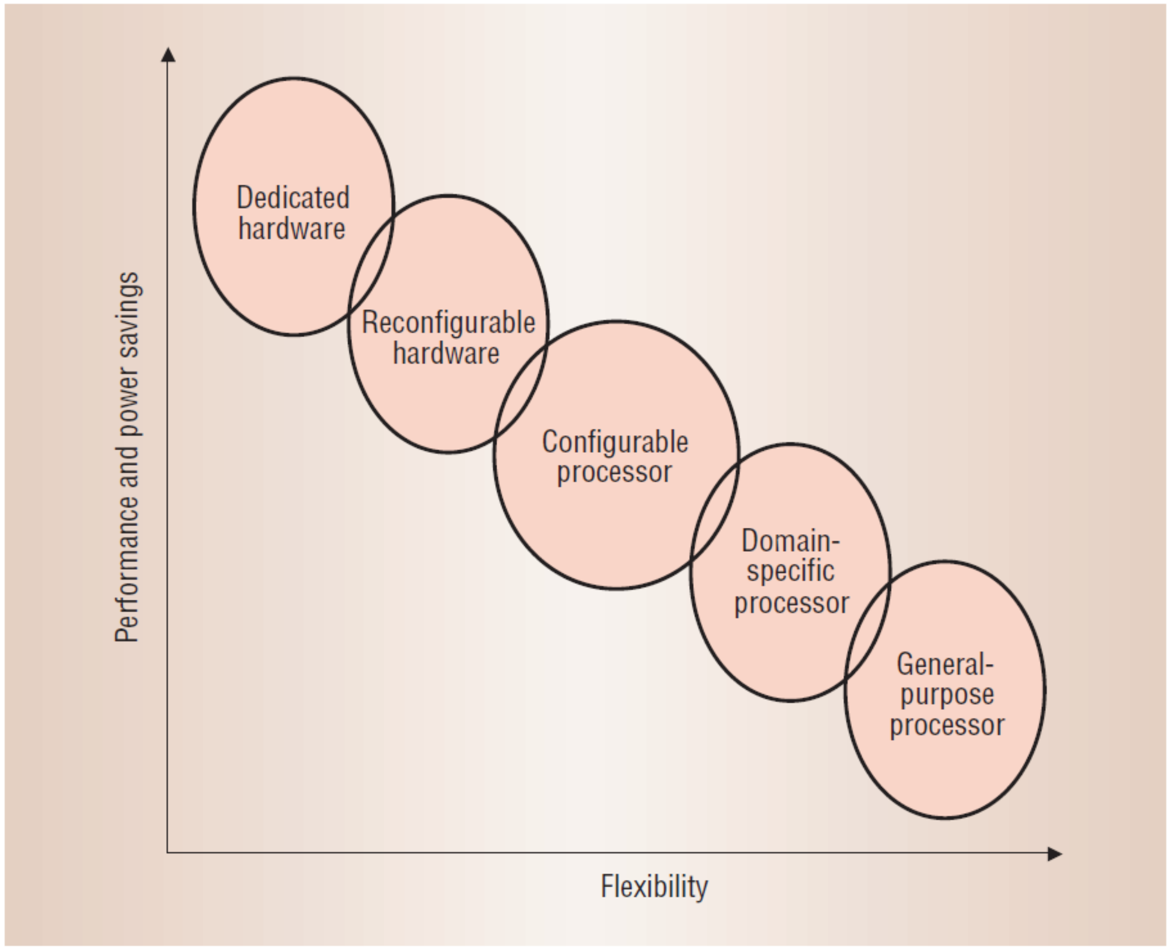
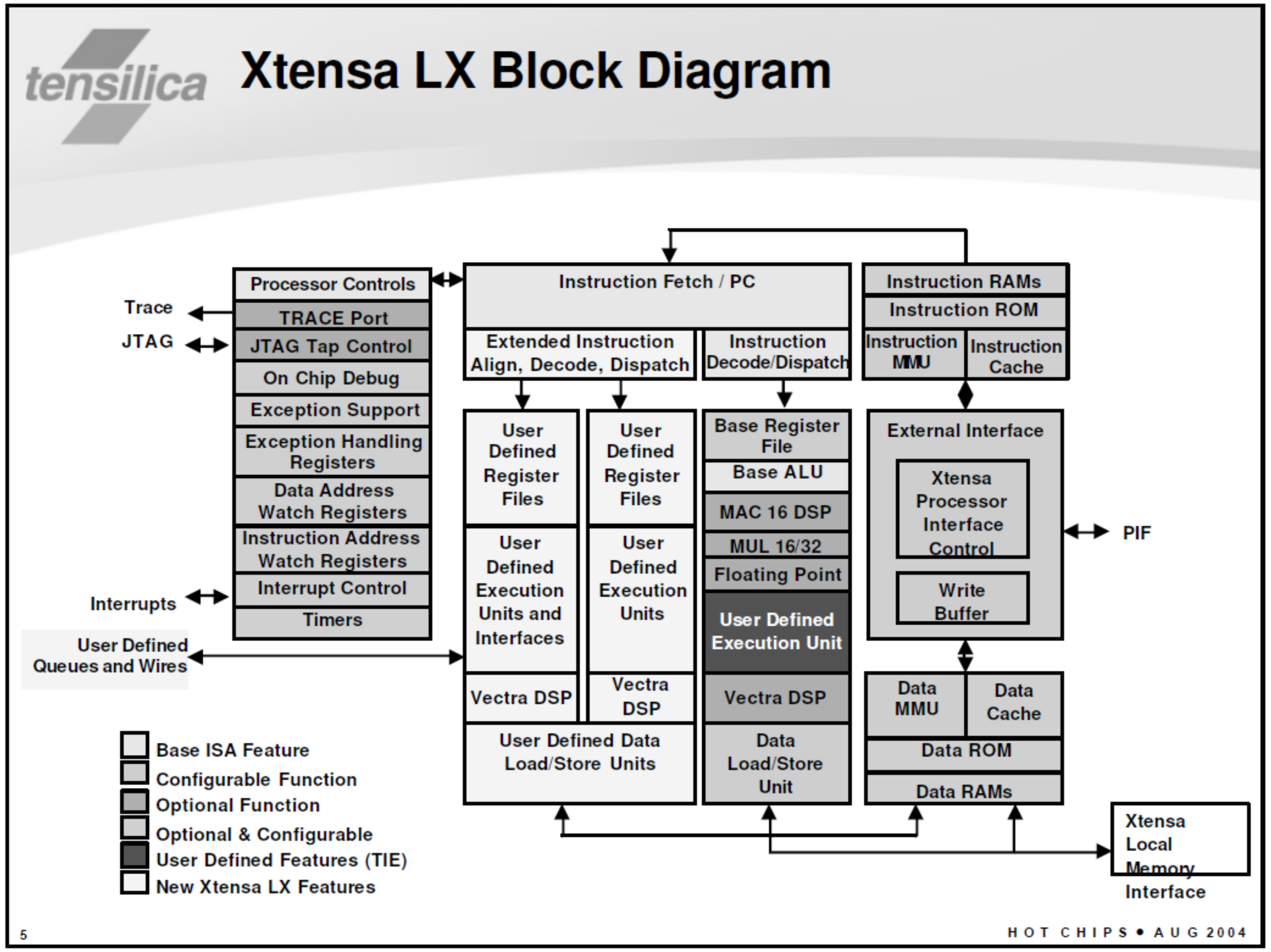

Dedicated hardware
Reconfigurable hardware
Configurable processor
Domain-specific processor
General-purpose processor
Performance and power savings
Flexibility

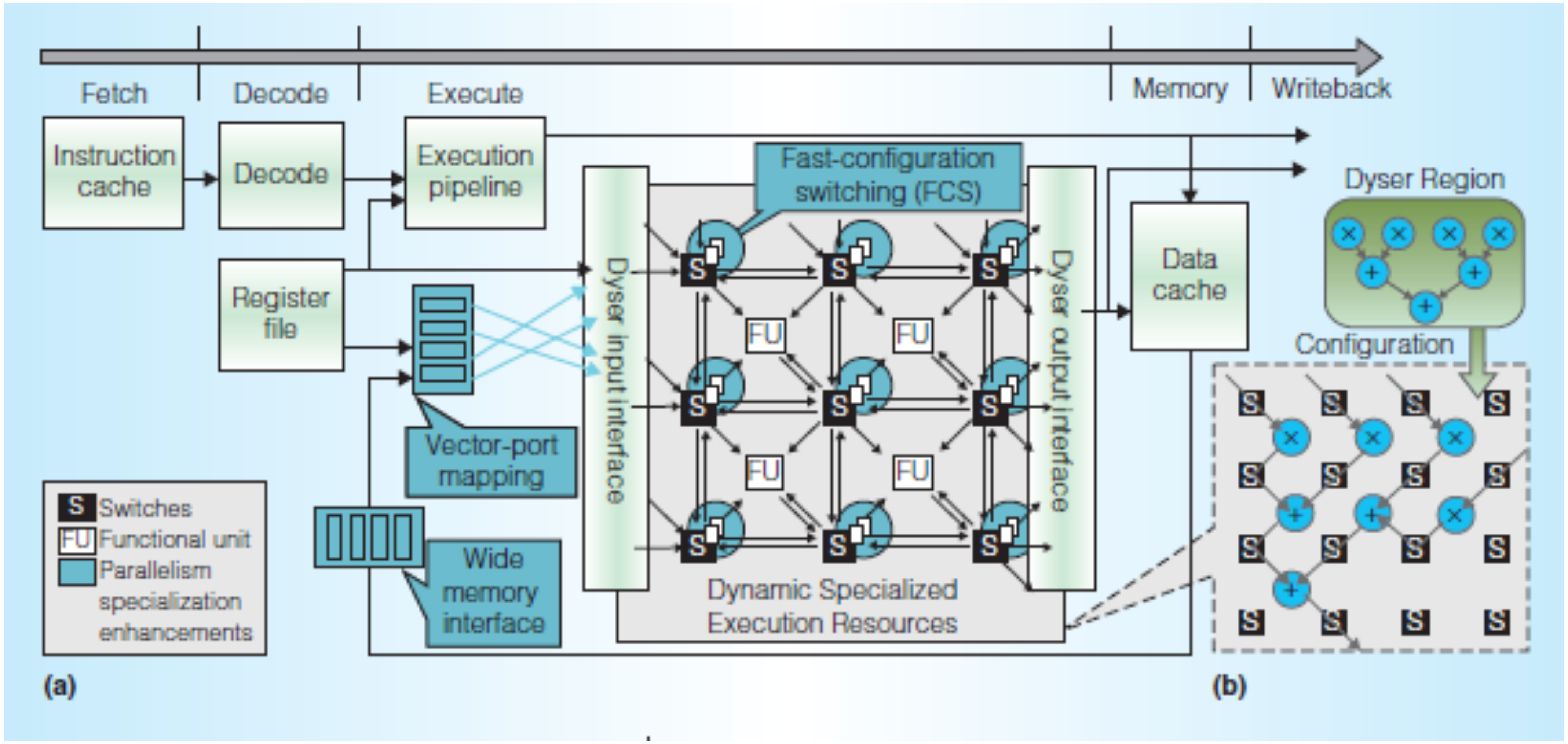
Figure 1. Embedded SoC design-space continuum trades the performance and power savings of dedicated hardware for the flexibility of software-based solutions.

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

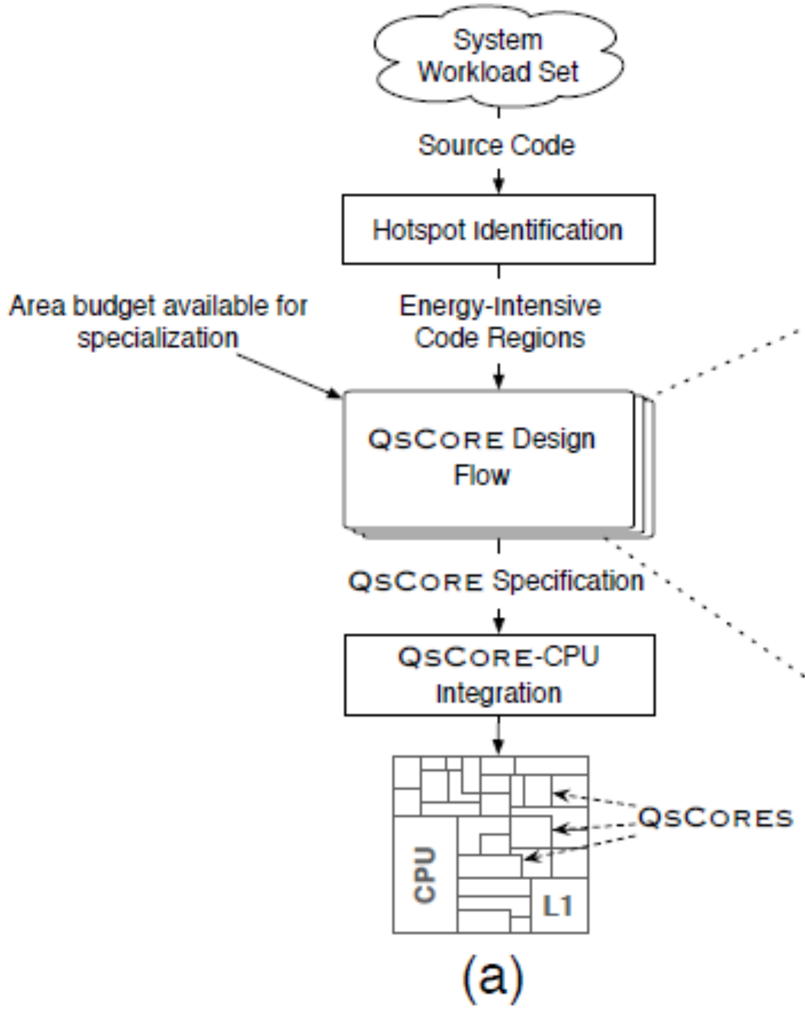
Configurable processors (Tensilica/Cadence)

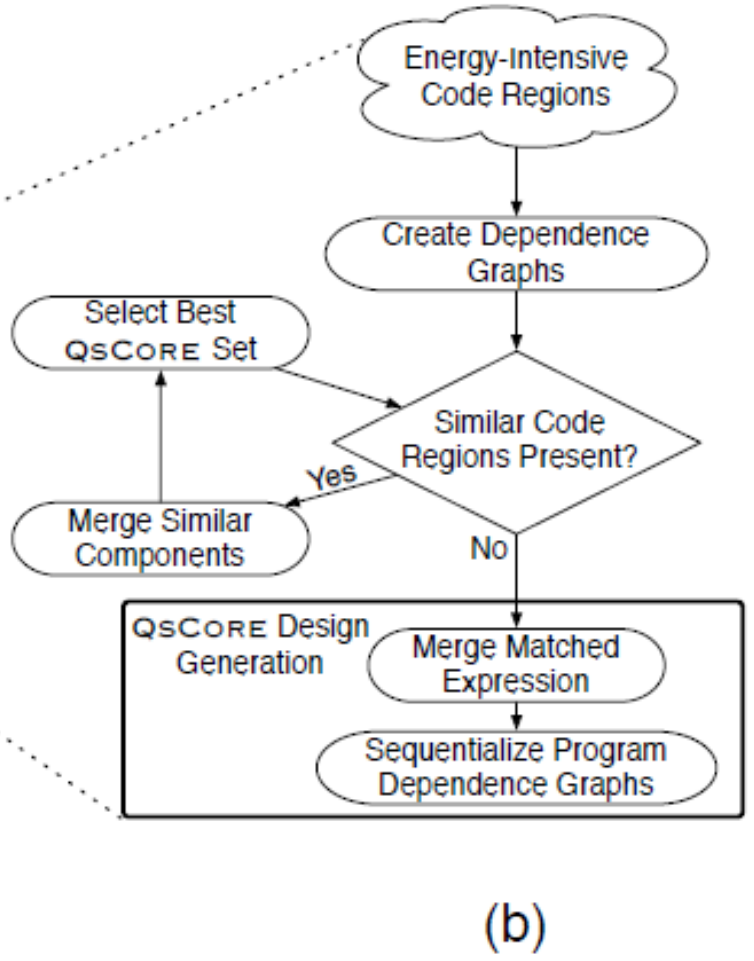


Dynamically specialised execution resources (DYSER, IEEE Micro 2012)



Quasi-Specific cores (QSCOREs) [Micro 2011]





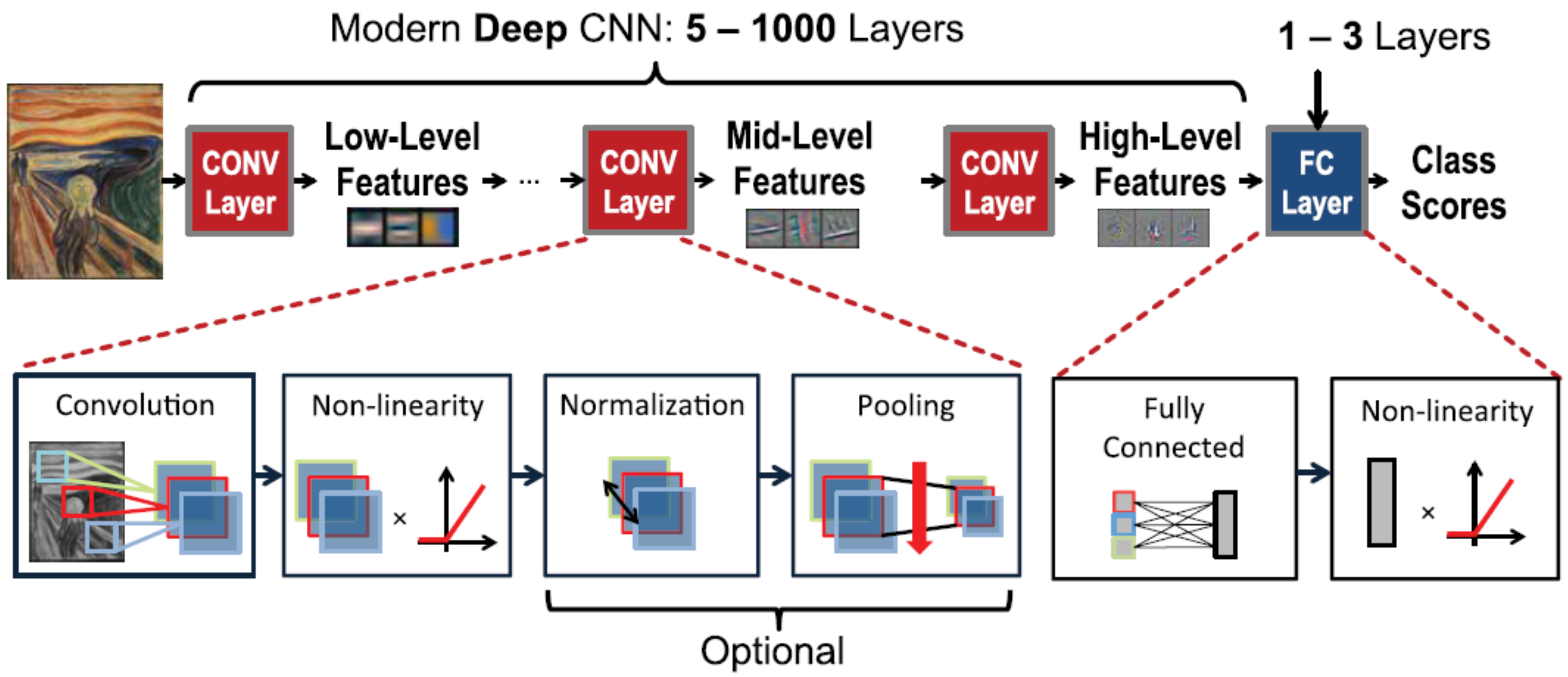
QSCOREs generated using C-to-HW compiler

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

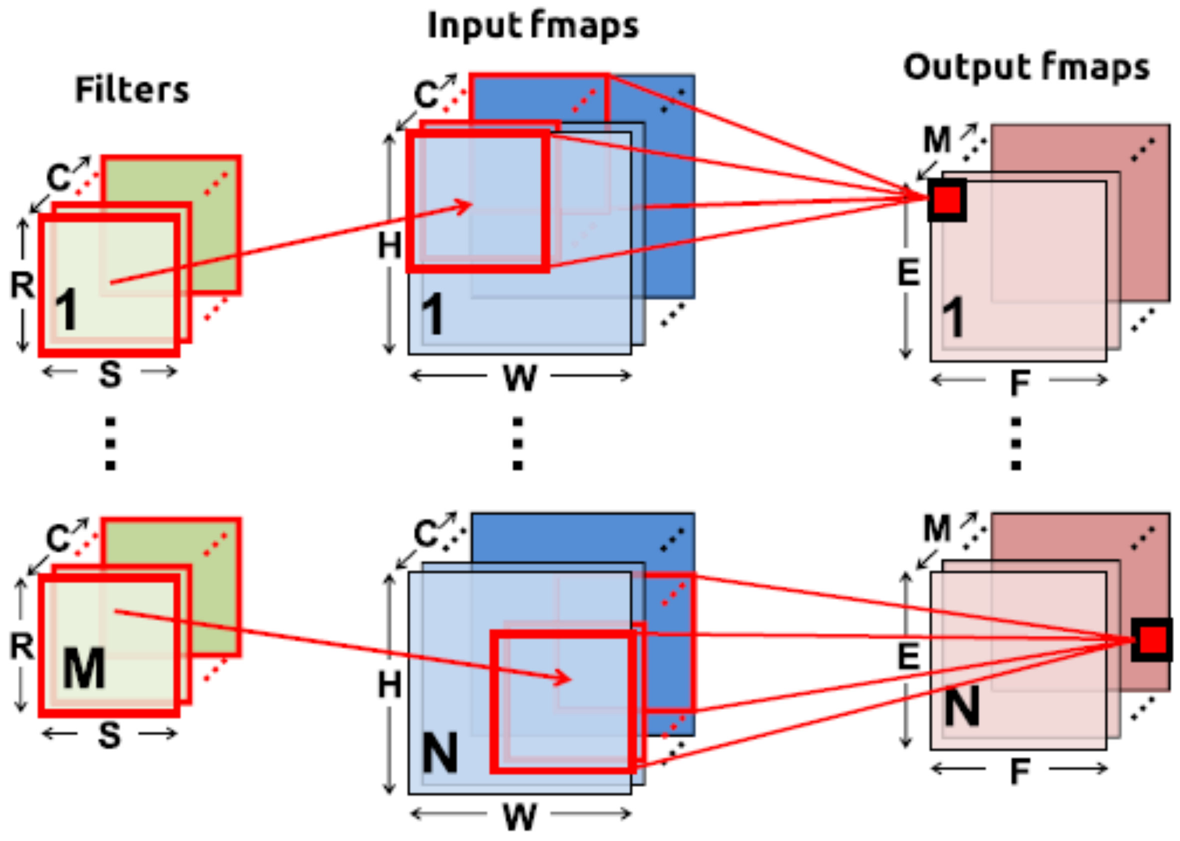
Memory operations access same data cache as GPP

Hardware accelerators for machine learning



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

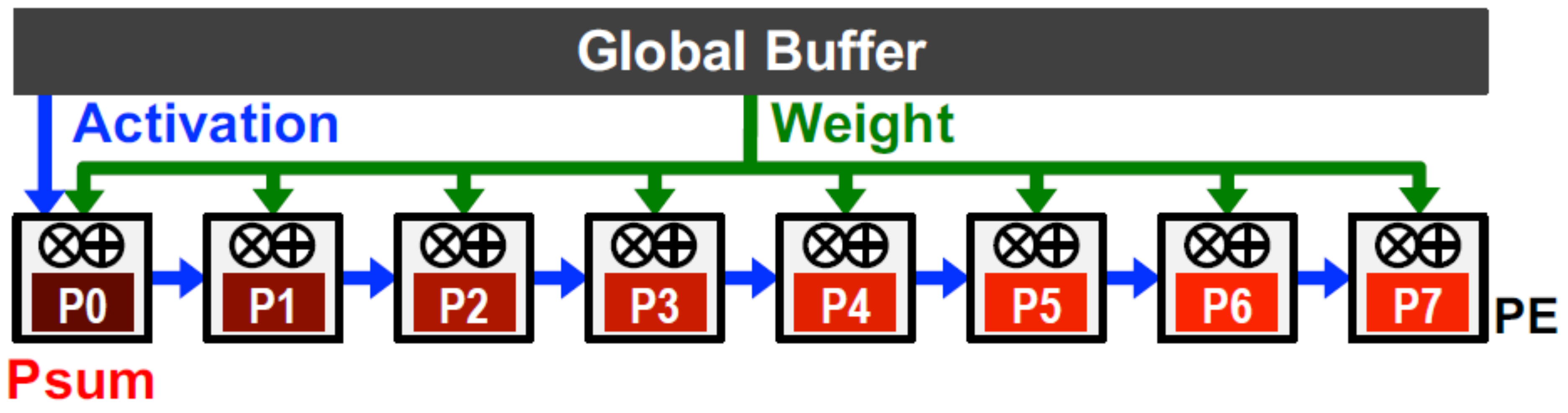


Reproduced from: Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, vol. 105, no. 12

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)
- PIM or CIM proposals
    - DRAM, SRAM, memrisistor-based memories
- Analog neural networks, neuromorphic (spiking) designs....

Challenges

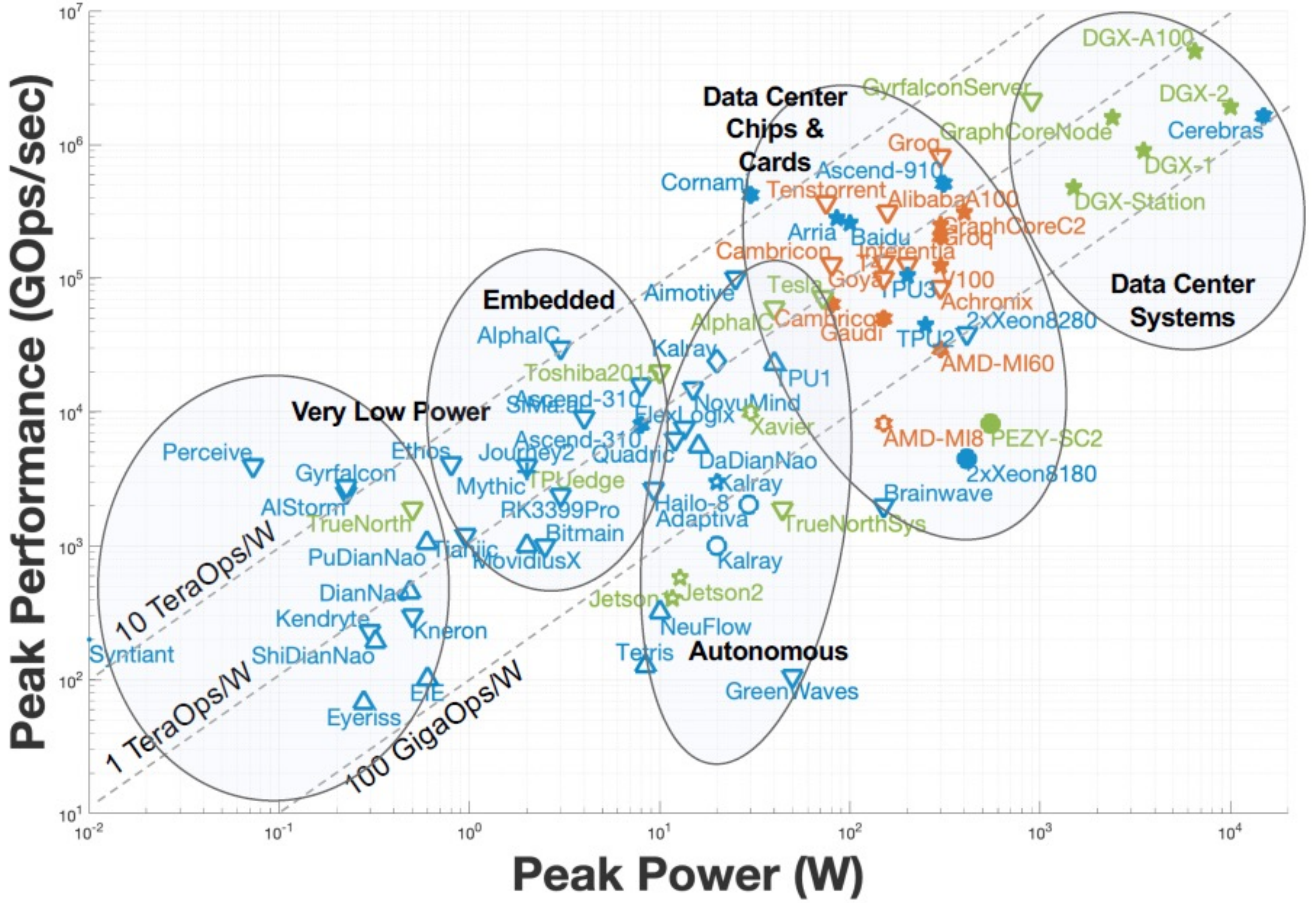
- Workload can vary
  - Already many different types of network and associated optimisations
    - e.g. sparse vs. dense, convolutional layers vs. fully-connected layers etc.
  - Problem size and required performance may differ
    - We see the use of “tiny”, “small” and “big” NPUs (as we do for processors)
- Workload is still evolving
  - New network types being developed
  - more dynamic/conditional behaviours etc., networks of networks?
- 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

Machine learning accelerators: peak perf. vs peak power

(Reuther et. al 2020)



Final thoughts

- How do accelerators and GPPs communicate and share memory? Are they coherent? Can processors share a large pool of accelerators?
- 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