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?

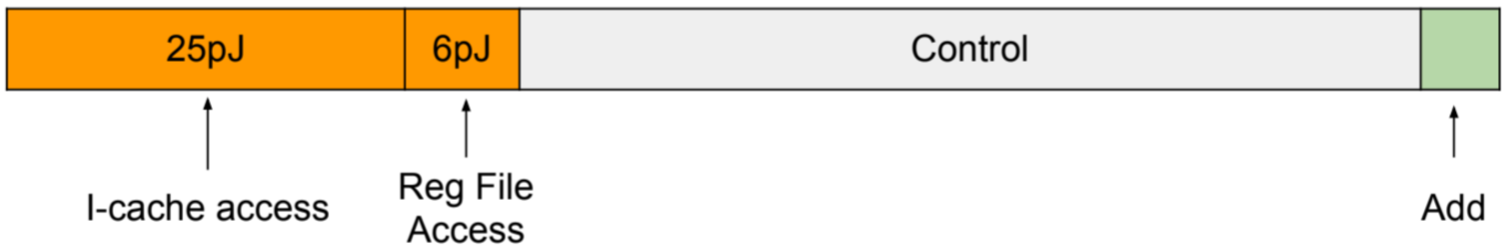
- Remove infrequently used parts of the processor
- Tune instruction set for common operations or replace with hardwired control
- Exploit forms of parallelism abundant in the application(s) – we often see a specialised processing element and local memory reproduced many times.
    - Can we also accelerate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

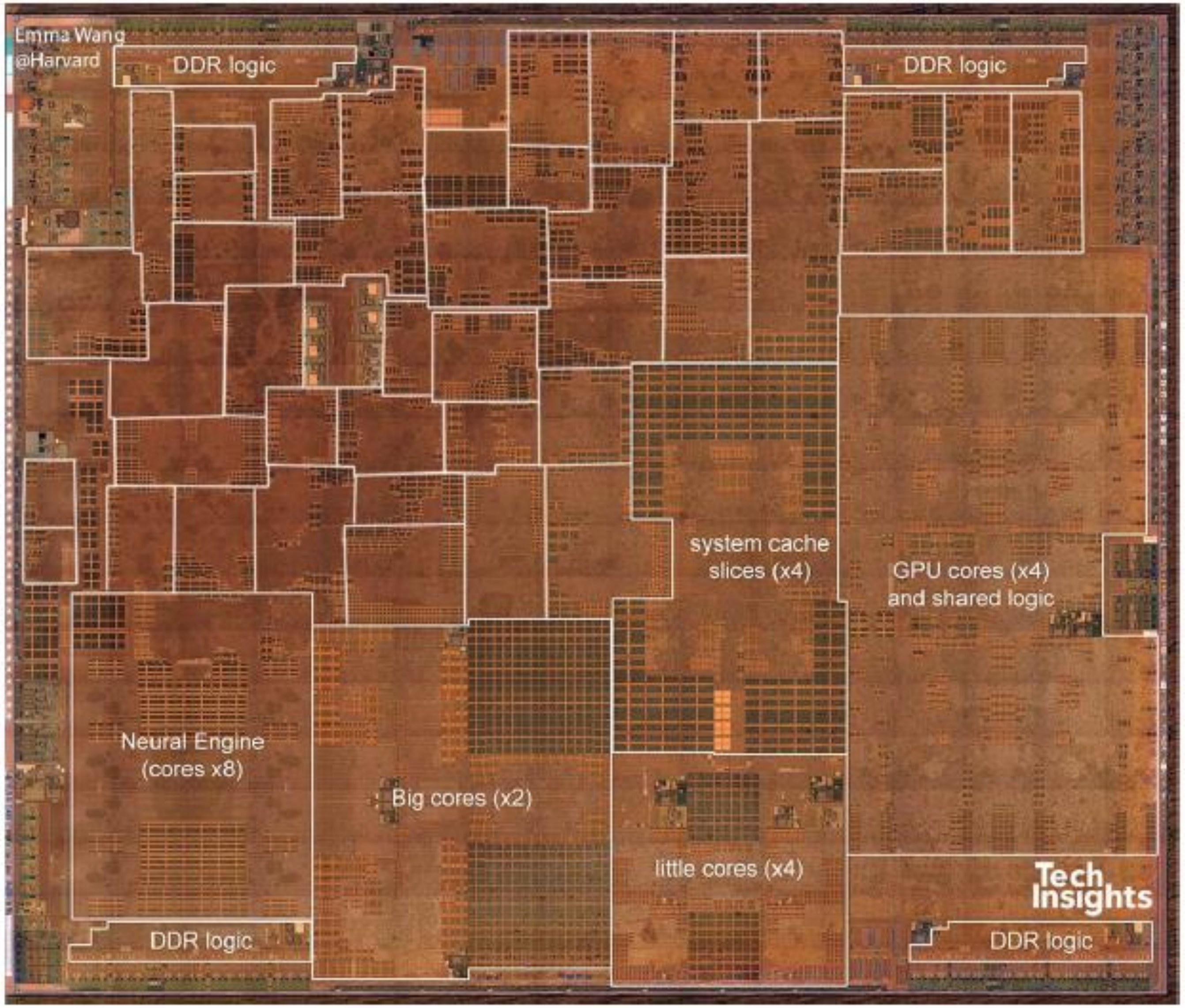
Instruction Energy Breakdown (Total 70pJ)



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

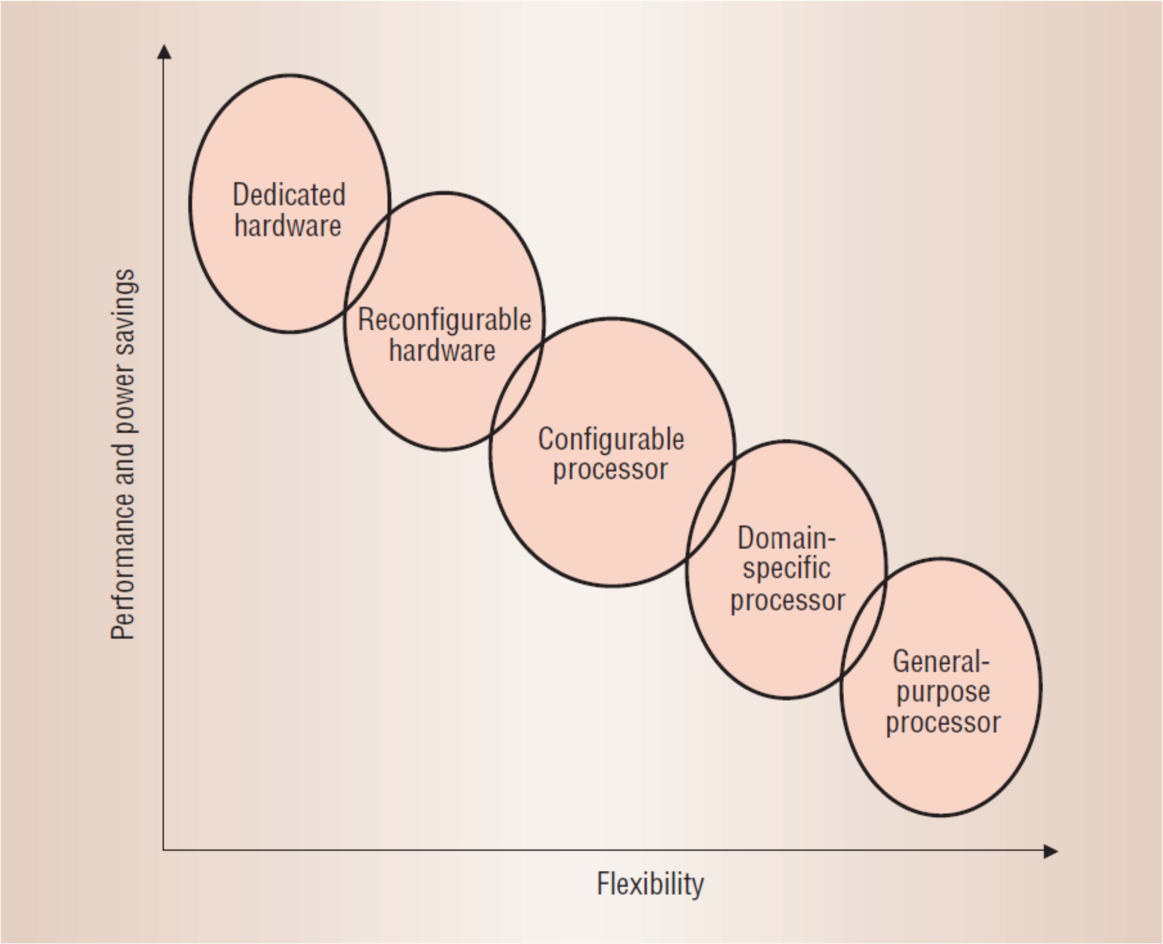
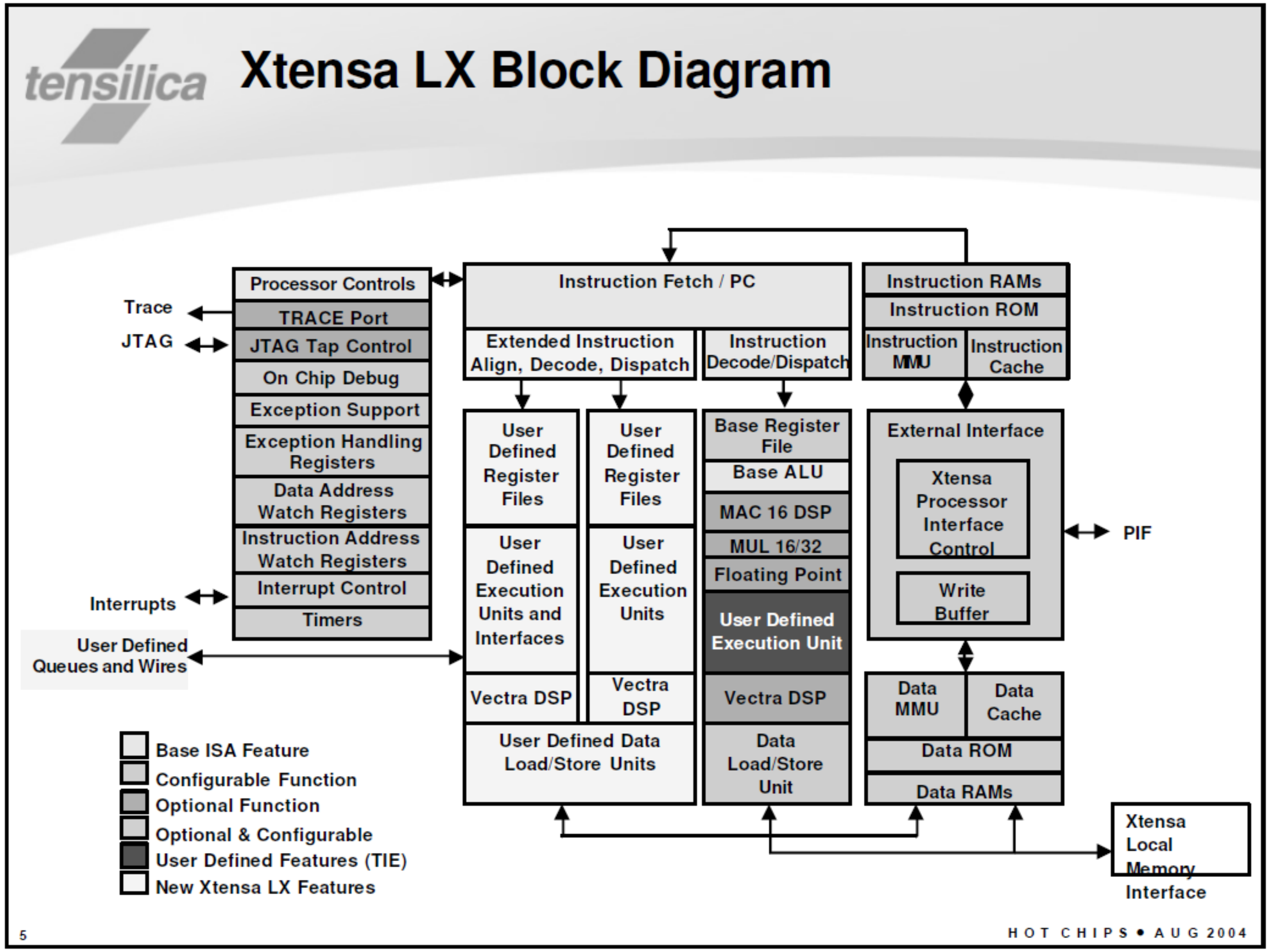


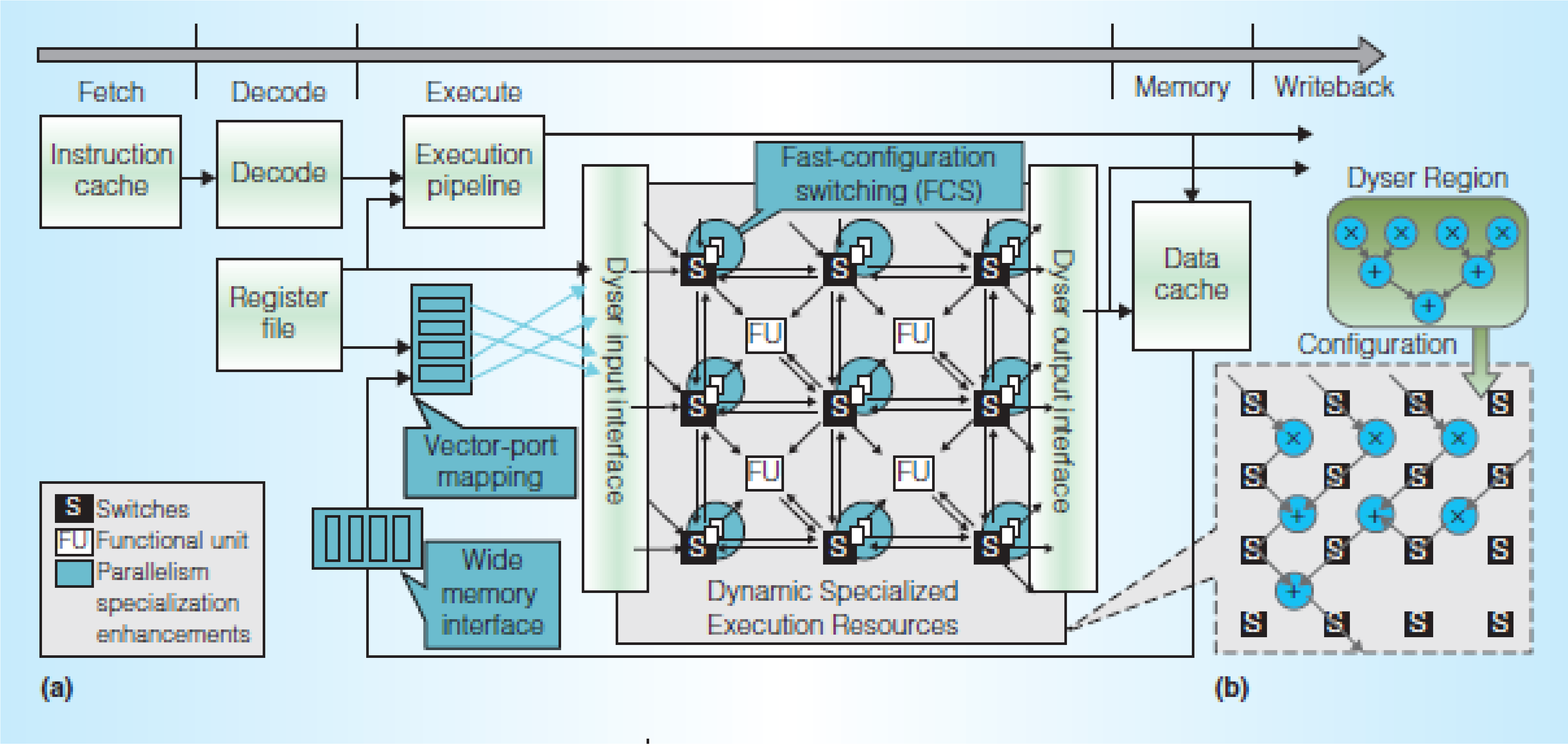
Figure 1. Embedded SoC design-space continuum trades the performance and power savings of dedicated hardware for the flexibility of software-based solutions.

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

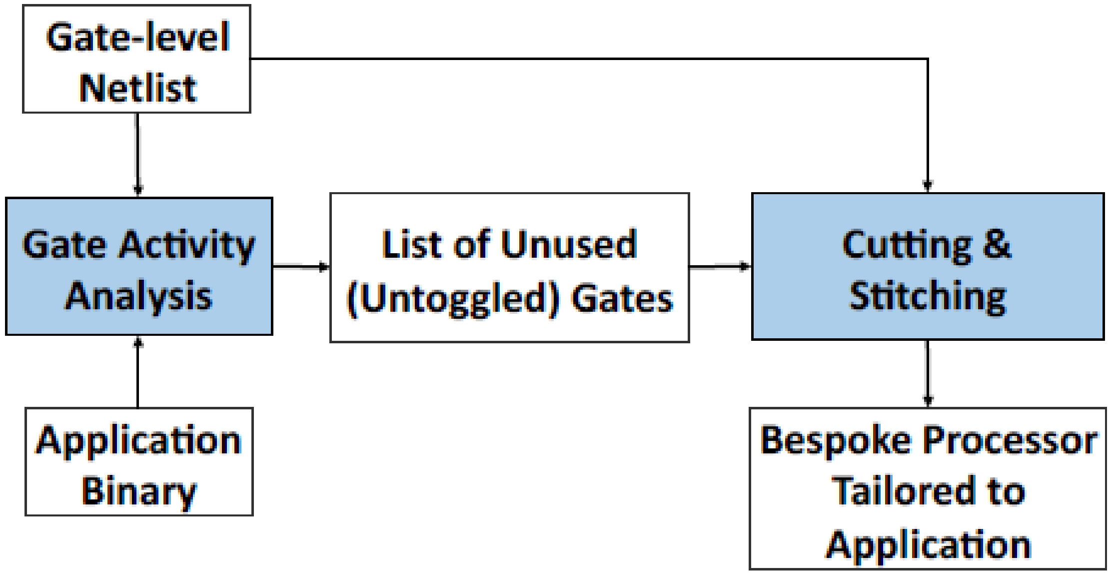
Configurable processors (Tensilica/Cadence)

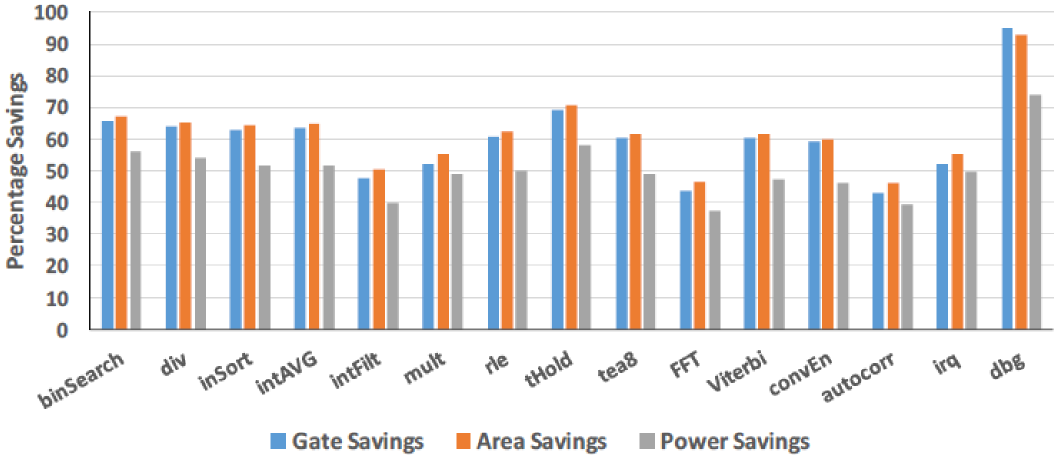


Dynamically specialised execution resources (DYSER, IEEE Micro 2012)

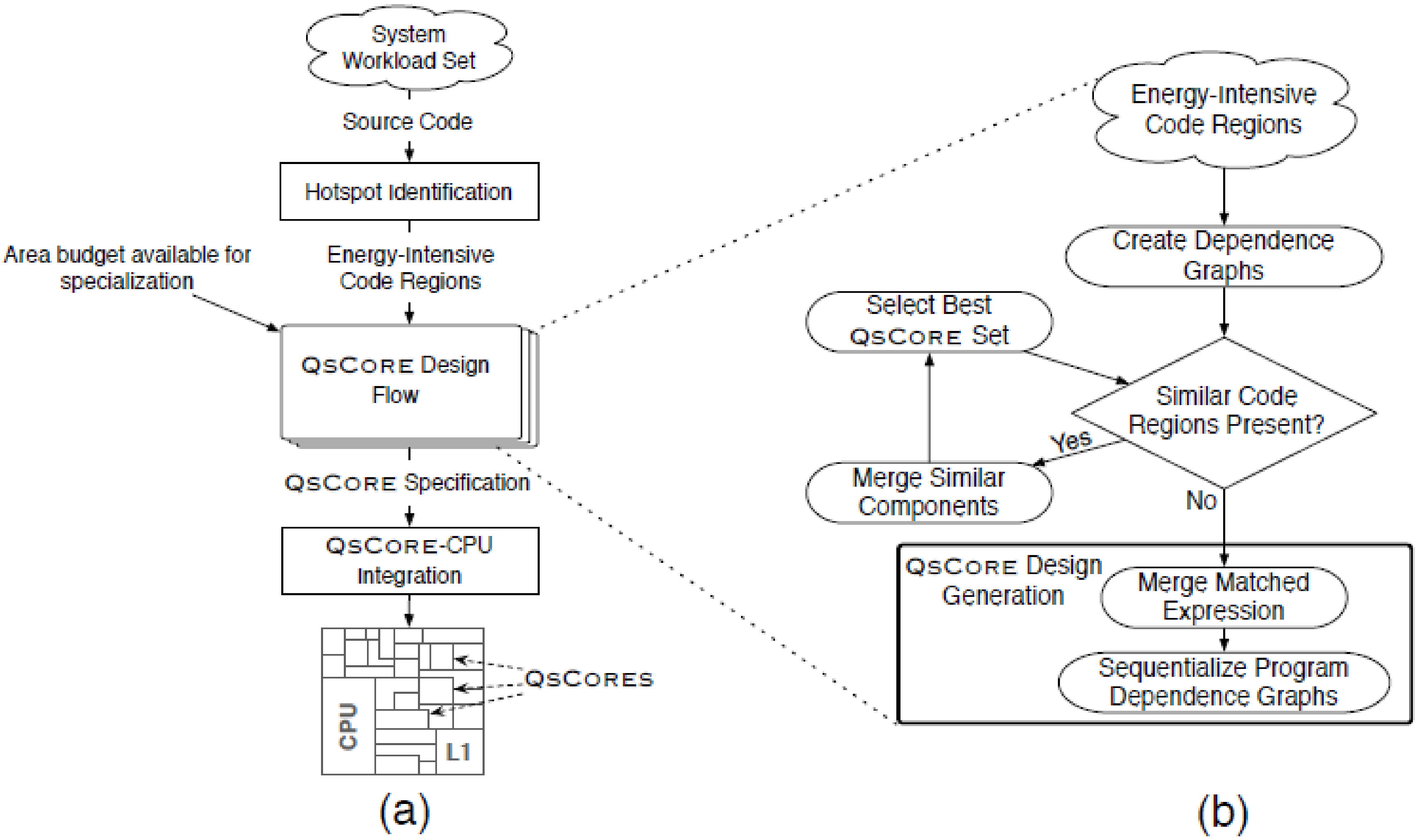


Bespoke processors [ISCA 2017]





Quasi-Specific cores (QSCOREs) [Micro 2011]



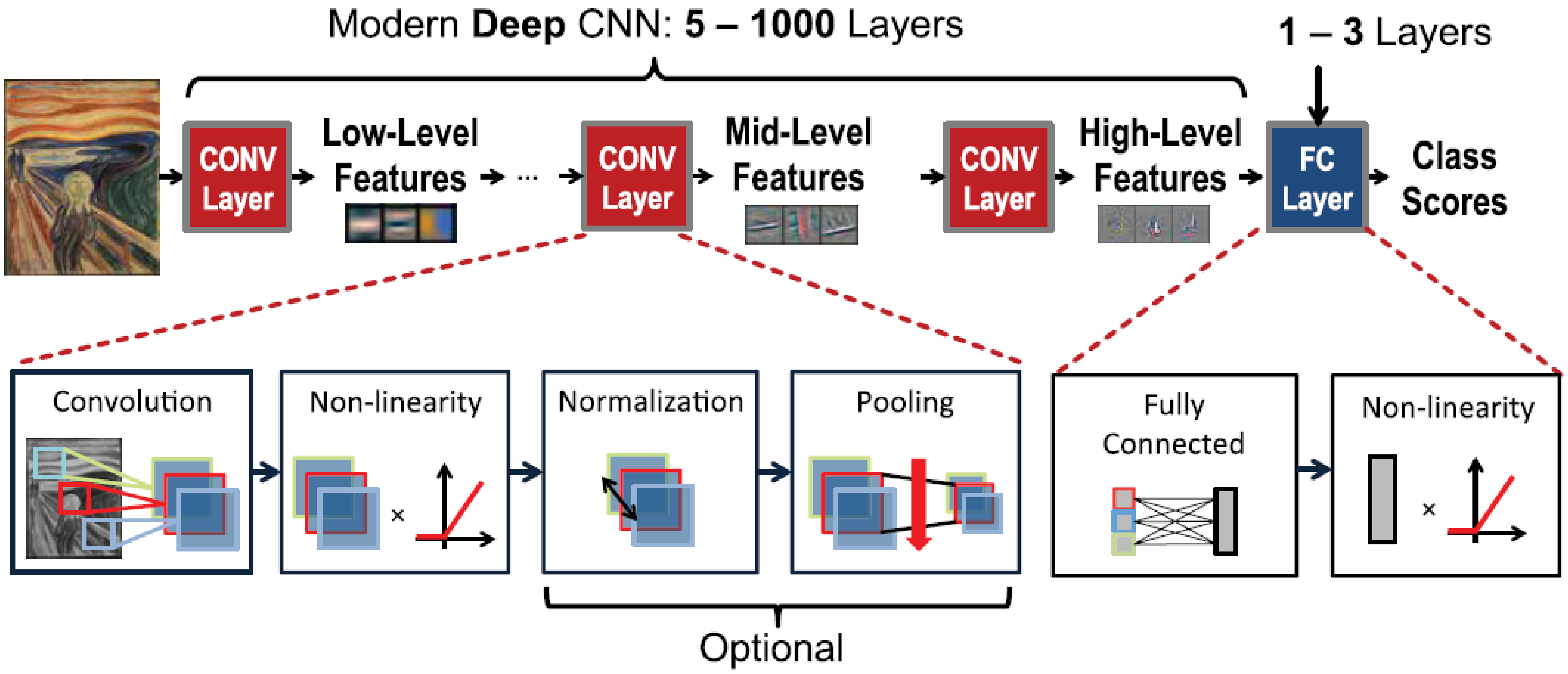
QSCOREs generated using C-to-HW compiler

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

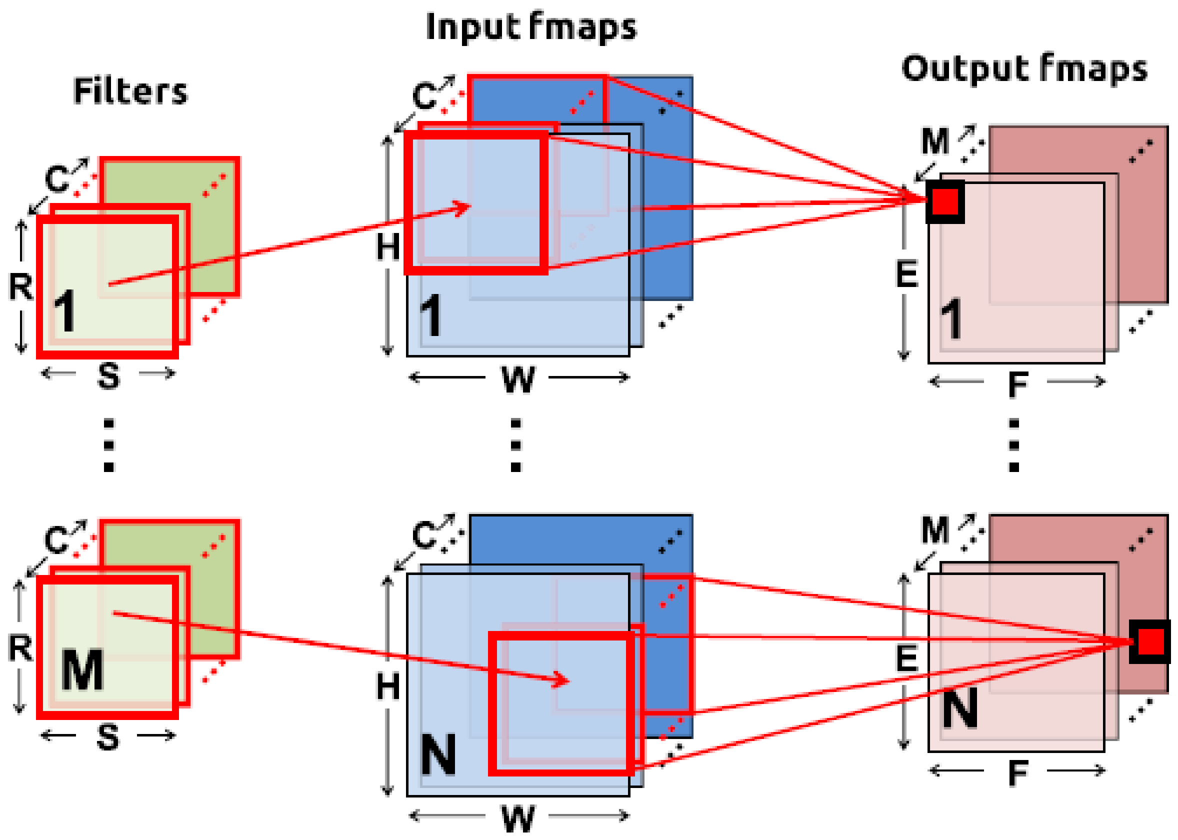
Use of configurable ALUs too

Memory operations access same data cache as GPP

Hardware accelerators for machine learning



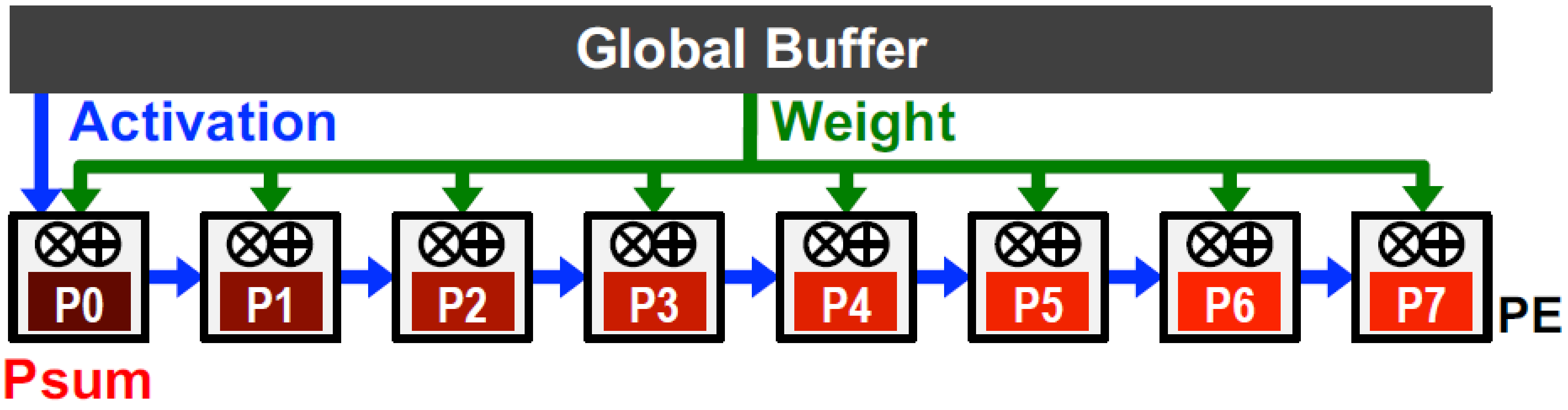
Hardware accelerators for machine learning



Data reuse patterns

- Memory access is likely bottleneck – very large volumes of data
  - Weights, activations, (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
- **Let’s explore dataflows in reading group**

Hardware accelerators for machine learning

- IoT
  - Interesting work to target very resource constrained devices
- Mobile
  - Arm, Huawei, Samsung, Apple, …. all have NPU designs
- Edge
  - Wave Computing (CGRA), NVIDIA
- Server (training)
  - Google TPU (3 generations)
  - Groq (ex-TPU team members), SambaNova - CGRAs?
  - GraphCore (very large amount of on-chip SRAM)
  - Cerebras - waferscale proposal (42,255mm^2, 400,000 cores!)
  - NVIDIA
- PIM proposals, SRAM based, analog neural networks, neuromorphic designs....

Challenges

- Designing NPUs is difficult
  - e.g. sparse vs. dense
  - e.g. convolutional layers for fully-connected layers
- Workload is still evolving
  - Often need to compromise support for some types of network to reduce overheads:
  - Also not just all about images, will need to accelerate other applications e.g. ASR (Automatic Speech Recognition), speech translation, text to speech etc.
  - But compromise will lead to lower utilisation of resources
  - Computer architecture is always trade-off!

Challenges

- Hard to fix precision (i.e. bit width of weights, activations and gradients, if training)
  - Some work on composing larger integer units from small ones
- Data type (arithmetic) is flexible too, e.g. binary, shift weights, fixed point, floating point (and variations)
- Often very high target TOP/s, but highly power constrained, constrained by memory BW too!
- Business or “social” issues
  - May have to work with whatever the customer provides, i.e. HW vendor may not be able to retrain network (no access to original training dataset)

Challenges

- NPU architectures?
  - How are PEs connected (i.e. local interconnect)
  - How much local buffering or SRAM?
  - Monolithic vs. tiled?
    - Can we partition resources? How local is control?
    - Do we place general-purpose compute close by or within the NPU?
  - Heterogeneous HW?
    - i.e. separate HW for different bitwidths or datatypes or network types? Within a tile or completely separate NPUs?
    - Or incorporate options within a single NPU? E.g. select from different bitwidths or datatypes?
      - Do we overprovision some types of resource by doing this?
  - Support multi-network workloads?
  - Dynamic behaviours?

Final points

- How do accelerators and GPPs communicate and share memory? Are they coherent?
- When we add accelerators to our system, how do we 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

Papers

Week 8: HW Accelerators and accelerators for machine learning

Pushing the limits of accelerator efficiency while retaining programmability, Nowatzki, Gangadhar, Sankaralingam and Wright, HPCA 2016

(LSSD, nice paper identifying common features of many highly parallel accelerators)

Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks, Chen, Emer and Sze, ISCA 2016

(CNN accelerator, good discussion of data reuse patterns and trade-offs, see also Eyeriss v2)

EIE: Efficient Inference Engine on Compressed Deep Neural Network, Han, Liu, Mao, Pu, Pedram, Horowitz and Dally, ISCA 2016
(sparse data after pruning, skip zero activations)

Other optional/background material for week 8

Efficient Processing of Deep Neural Networks: A Tutorial and Survey Sze, Chen, Yang, Emer, Proceedings of the IEEE, Vol. 105, No. 12, Dec. 2017

Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures Williams, Waterman and Patterson, Communications of the ACM, vol. 52, Issue 4, April 2009, pp 65-76.