AGILE DESIGN AND INTEGRATION OF ACCELERATORS IN HETEROGENEOUS SOC ARCHITECTURES

PhD Proposal Exam

Davide Giri

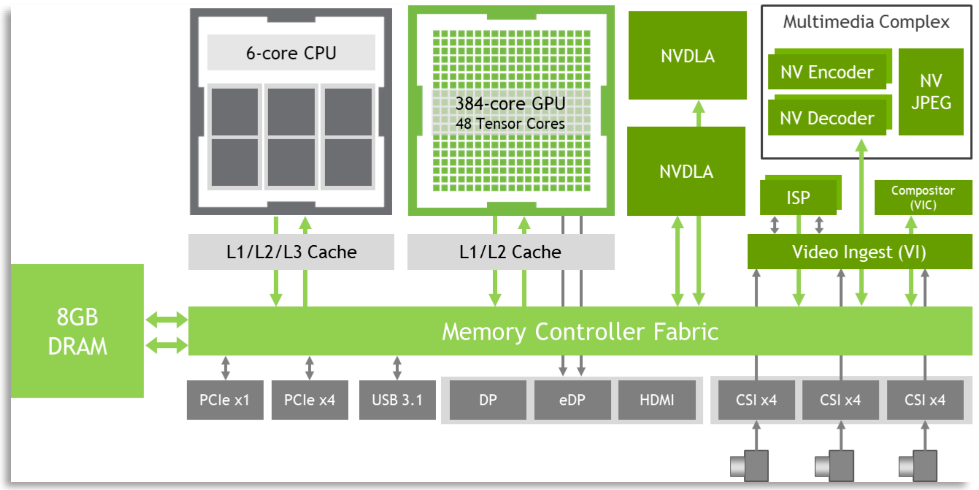
January 21, 2021

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

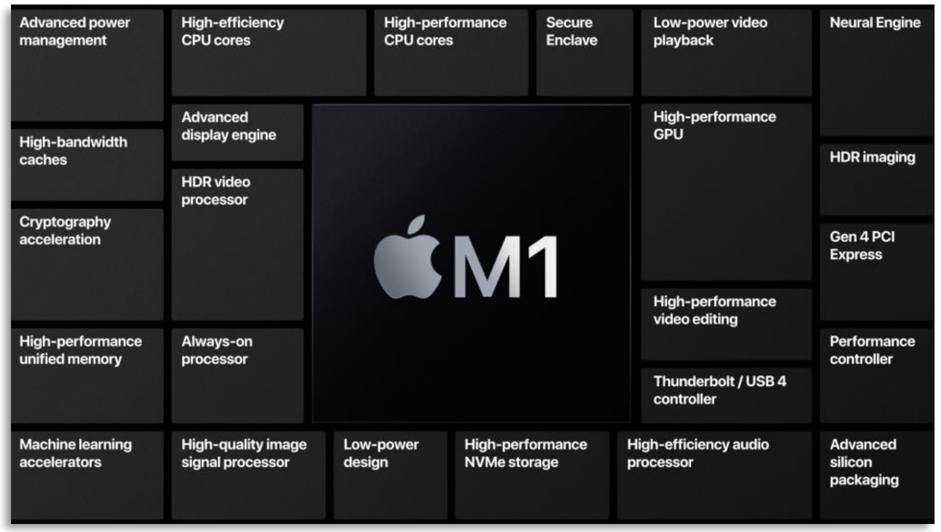
CS@CU COMPUTER SCIENCE

THE HETEROGENEOUS SOC

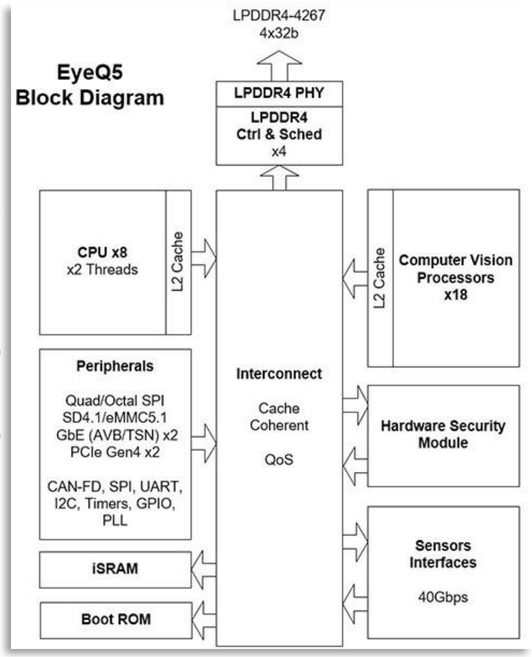
NVIDIA Xavier (2018)



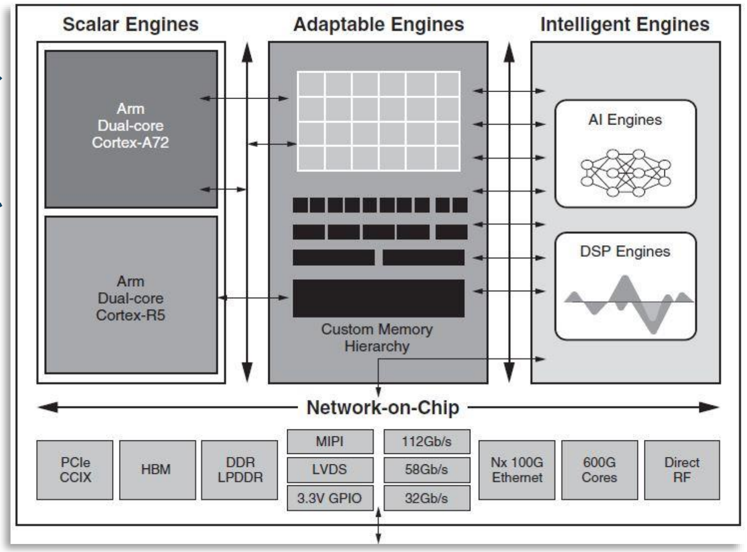
Apple M1 (2020)


Advanced power management
High-efficiency CPU cores
High-performance CPU cores
Secure Enclave
Low-power video playback
Neural Engine
High-bandwidth caches
Advanced display engine
High-performance GPU
HDR imaging
HDR video processor
M1
Gen 4 PCI Express
Cryptography acceleration
High-performance video editing
Performance controller
High-performance unified memory
Always-on processor
Thunderbolt / USB 4 controller
Machine learning accelerators
High-quality image signal processor
Low-power design
High-performance NVMe storage
High-efficiency audio processor
Advanced silicon packaging

Mobileye EyeQ5 (2020)



Xilinx Versal (2019)

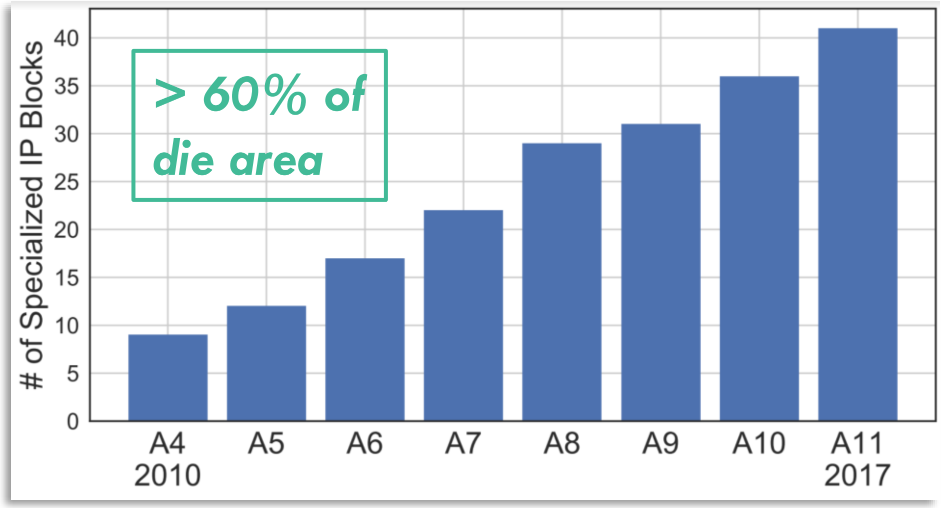


THE ERA OF ACCELERATORS

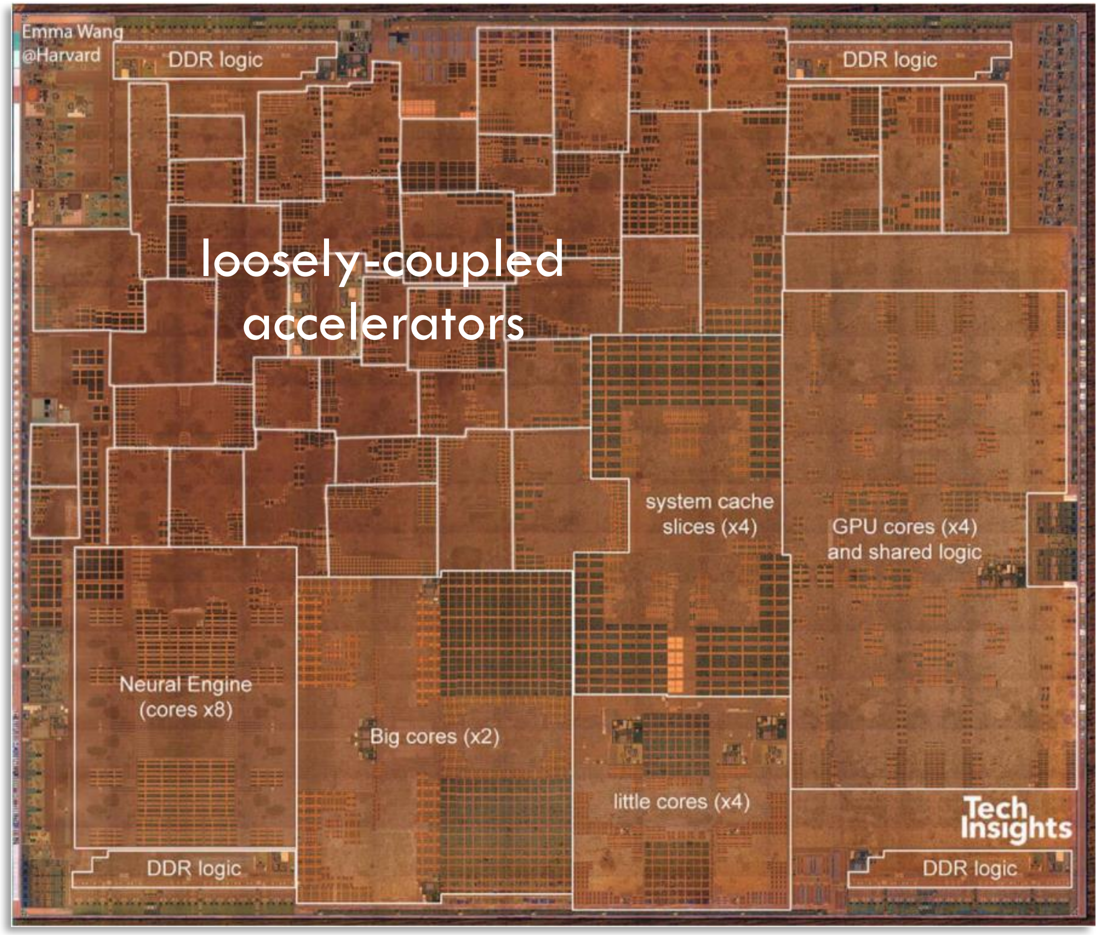
Modern SoCs are increasingly heterogeneous

- They integrate a growing number of accelerators

Accelerators in Apple SoCs

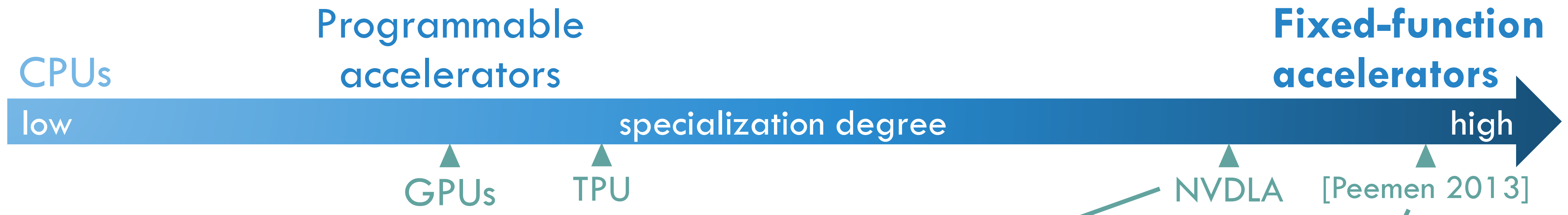


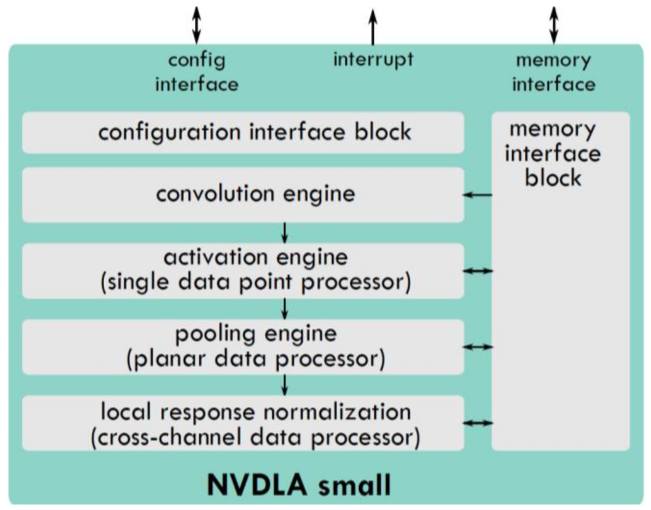
Apple A12 SoC

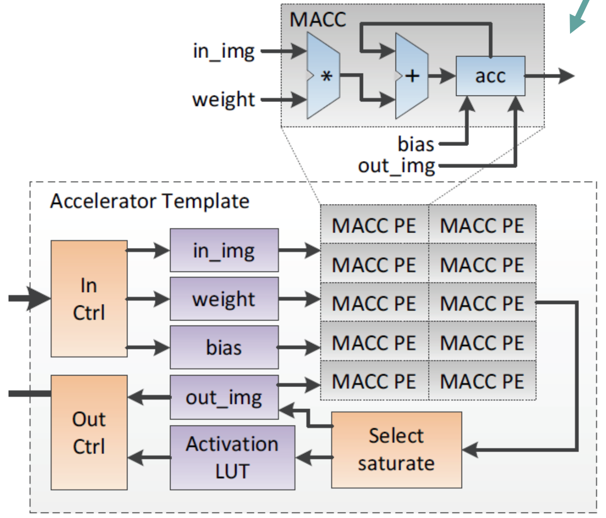


Images from vlsiarch.eecs.harvard.edu/research/accelerators/die-photo-analysis

ACCELERATORS TAXONOMY: SPECIALIZATION



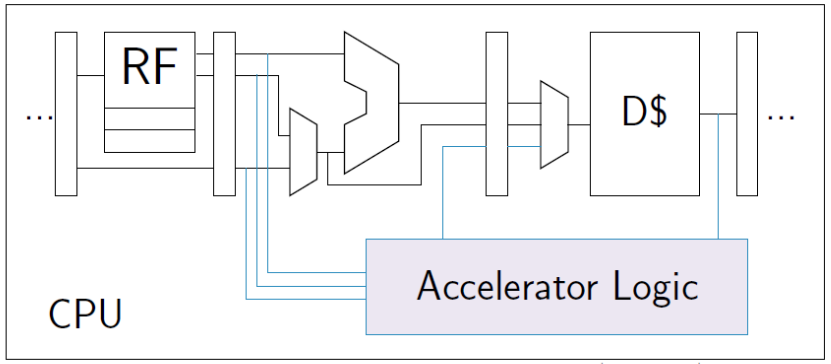

config interface
interrupt
memory interface
configuration interface block
memory interface block
convolution engine
activation engine (single data point processor)
pooling engine (planar data processor)
local response normalization (cross-channel data processor)
NVDLA small



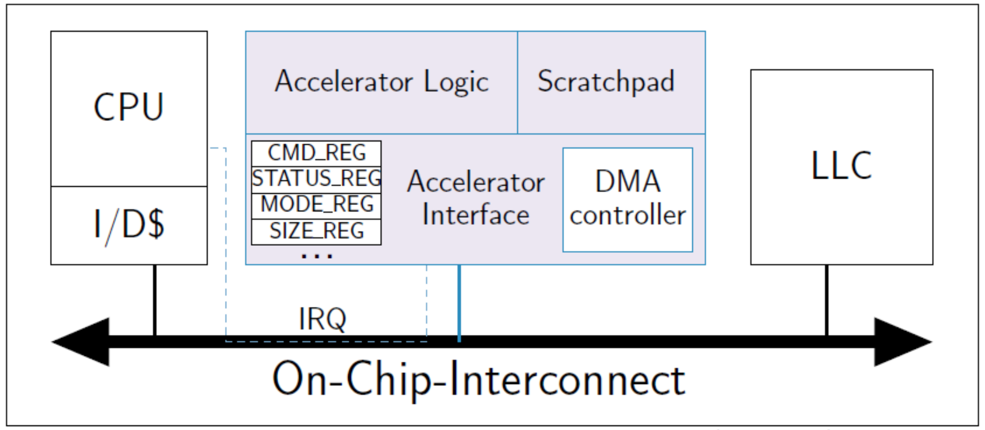
[Cascaval 2010] [Shao 2015]

ACCELERATORS TAXONOMY: COUPLING

- Tightly coupled
  - Part of the processor pipeline or
  - Attached to the private caches



- Loosely coupled
  - **Attached to the on-chip interconnect** or
  - Off-chip



SOC DESIGN CHALLENGES

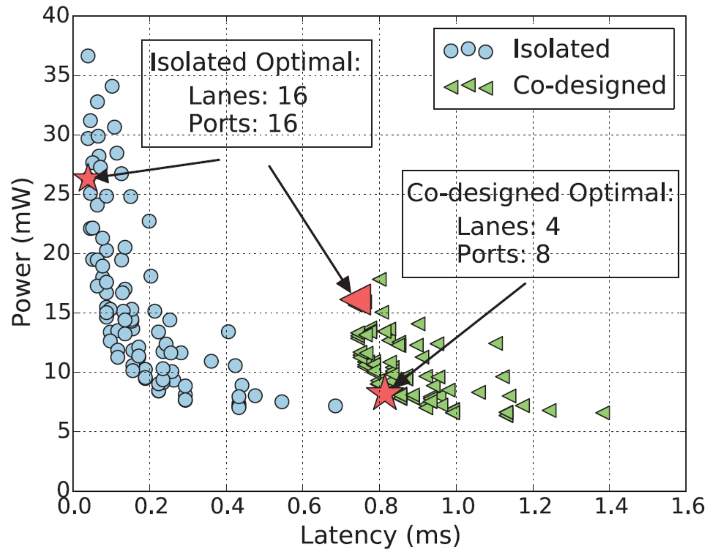
The engineering effort keeps growing

- SoCs are increasingly heterogeneous and specialized

The integration of accelerators is complex and critical to the system performance

- Accelerators are typically designed in isolation with little attention to system integration

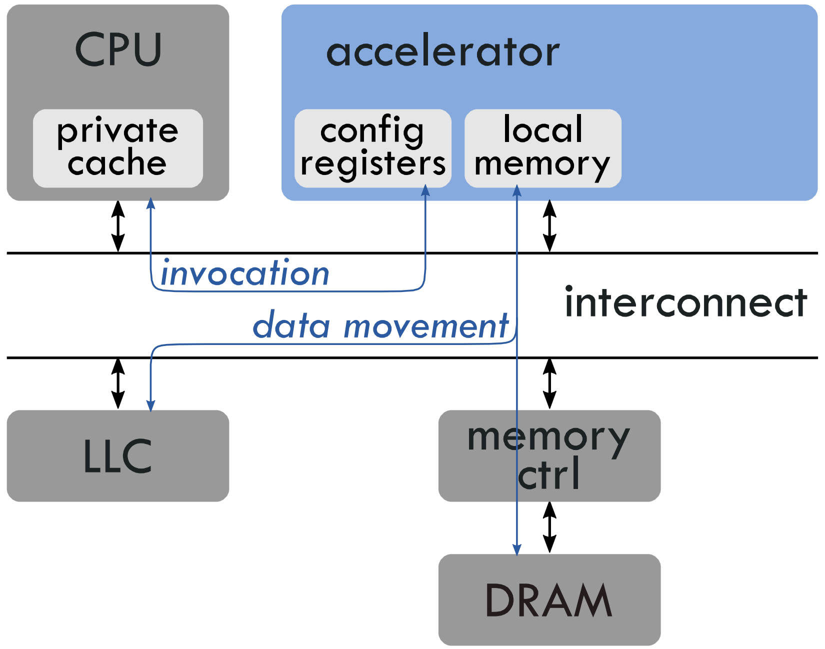
"The co-design of the accelerator microarchitecture with the system in which it belongs is critical to balanced, efficient accelerator microarchitectures." *[Shao 2016]*



ACCELERATOR INTEGRATION CHALLENGES

- Programming
  - Invocation, configuration, data allocation, …
- Data movement
- Integration flow

"Existing research on accelerators has focused on computational aspects and has disregarded design decisions with practical implications, such as the model for accelerator invocation from software and the interaction between accelerators and the components […] surrounding them." [Cota 2015]



AGILE ACCELERATOR INTEGRATION

Increase SoC design productivity by mitigating the complexity of accelerator design and integration

- Provide flexible pre-validated hw/sw SoC *services* for accelerators
  - Avoid “reinventing the wheel”.
- Provide an automated flow for accelerator design and integration
  - Need to support multiple languages and tools for accelerator design
- Contribute to open-source
  - Foster collaboration and IP reuse

THESIS AND CONTRIBUTIONS

By providing hardware and software services for accelerator programming and data movement paired with automated accelerator design and integration flows, an open-source platform can effectively mitigate the increasing complexity of SoC design

Accelerator data movement

- Cache coherence
- Point-to-point communication
- Shared scratchpad

Accelerator programming

- Accelerator API library

Accelerator design and integration flow

- C/C++ accelerator flow
- Deep learning accelerator flow
- Third-party accelerator flow

Open-source contribution

- ESP release
- Accelerator benchmark suite

ESP

Open-source SoC platform

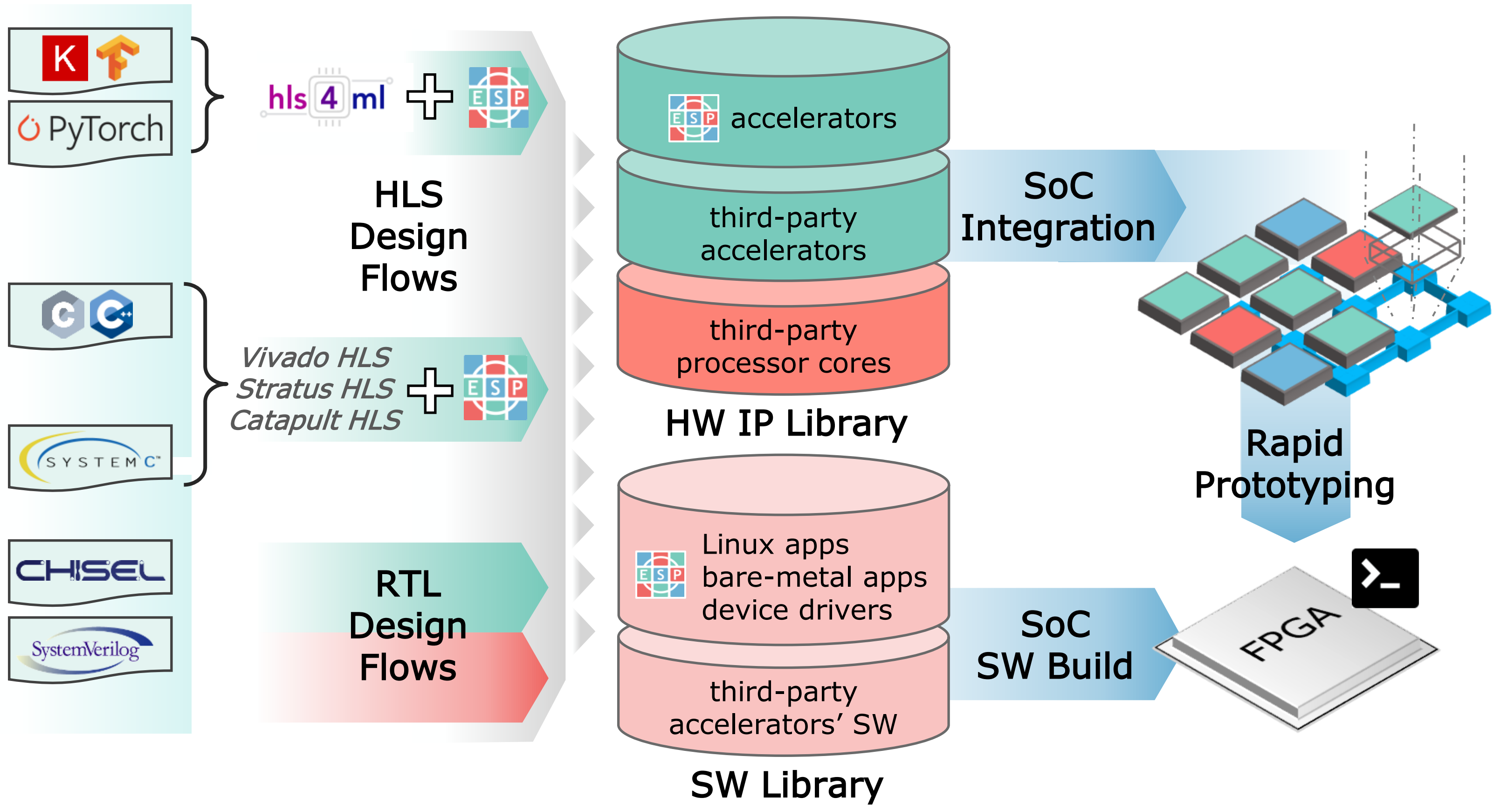
ESP METHODOLOGY

Accelerator Flow

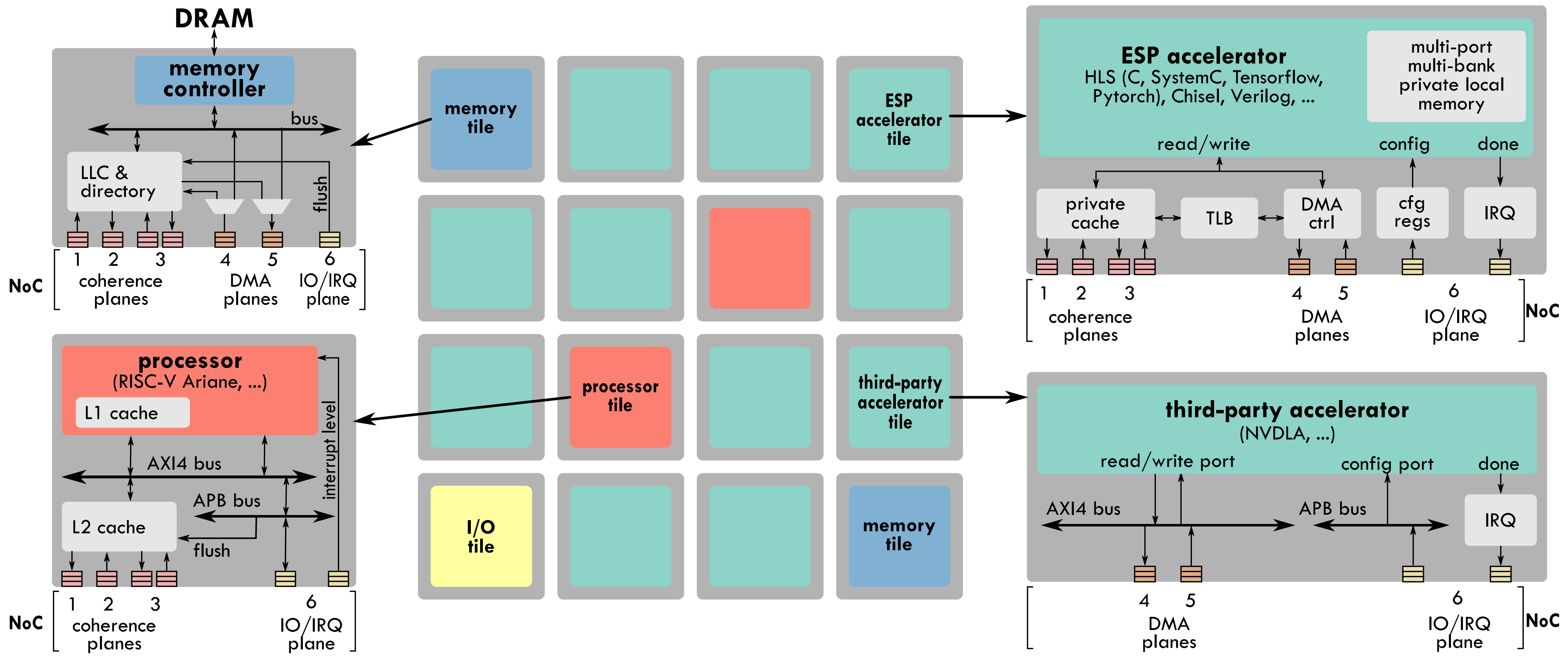
- Simplified design
- Automated integration

SoC Flow

- Mix&match floorplanning GUI
- Rapid FPGA prototyping



ESP ARCHITECTURE



ACCELERATOR DATA MOVEMENT

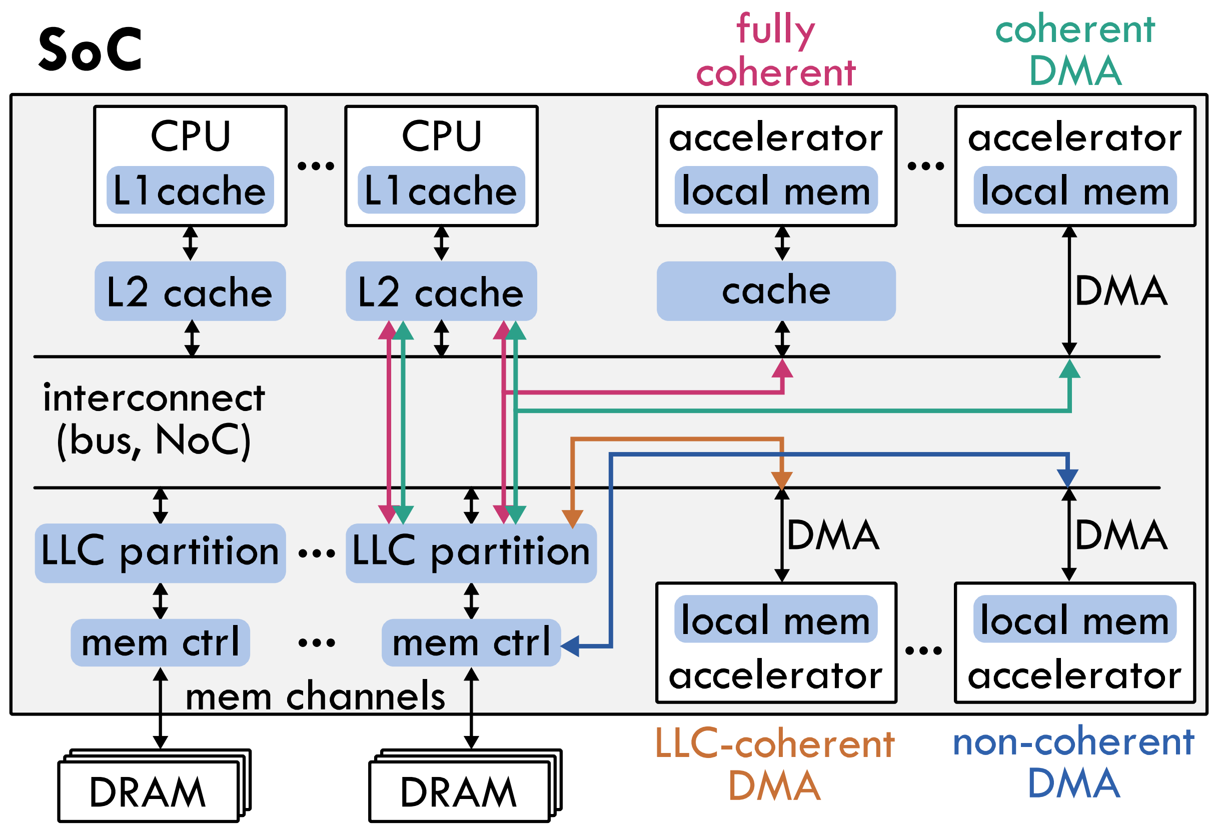
Cache coherence

Point-to-point communication

Shared scratchpad

ACCELERATOR CACHE-COHERENCE MODES

- **Coherent cache access** (**fully-coherent**)
  - Access to local private cache (same as CPUs)
  - Hardware-managed coherence
- **Coherent DMA**
  - Direct access to LLC
  - Hardware-managed coherence
- **LLC-coherent DMA**
  - Direct access to LLC
  - Hybrid hardware- and software-managed coherence
- **Non-coherent DMA**
  - Direct access to main memory
  - Software-managed coherence



ACCELERATOR CACHE-COHERENCE IN LITERATURE

	non-coh DMA	LLC-coh DMA	coh DMA	fully coh
Chen, ICCD’13	✓			
Cota, DAC’15	✓	✓		
Fusion, ISCA’15			✓	✓
gem5-aladdin, MICRO’16	✓			✓
Bhardwaj, ISLPED’20	✓	✓		✓
Spandex, ISCA’18				✓
ESP, ICCAD’20	✓	✓	✓	✓
NVDLA	✓			
Buffets, ASPLOS’19	✓			
Kurth, ArXiv’20	✓			

	non-coh DMA	LLC-coh DMA	coh DMA	fully coh
Cavalcante, CF’20			✓	
BiC, DAC’11	✓			
Cohesion, IEEEMicro’11		✓		
ARM ACE/ACE-Lite			✓	✓
Xilinx Zynq	✓		✓	
IBM Power7+			✓	
IBM Wirespeed			✓	
Arteris Ncore			✓	✓
IBM CAPI			✓	
CCIX			✓	✓

[Giri, NOCS'18]

PROPOSED CACHE HIERARCHY

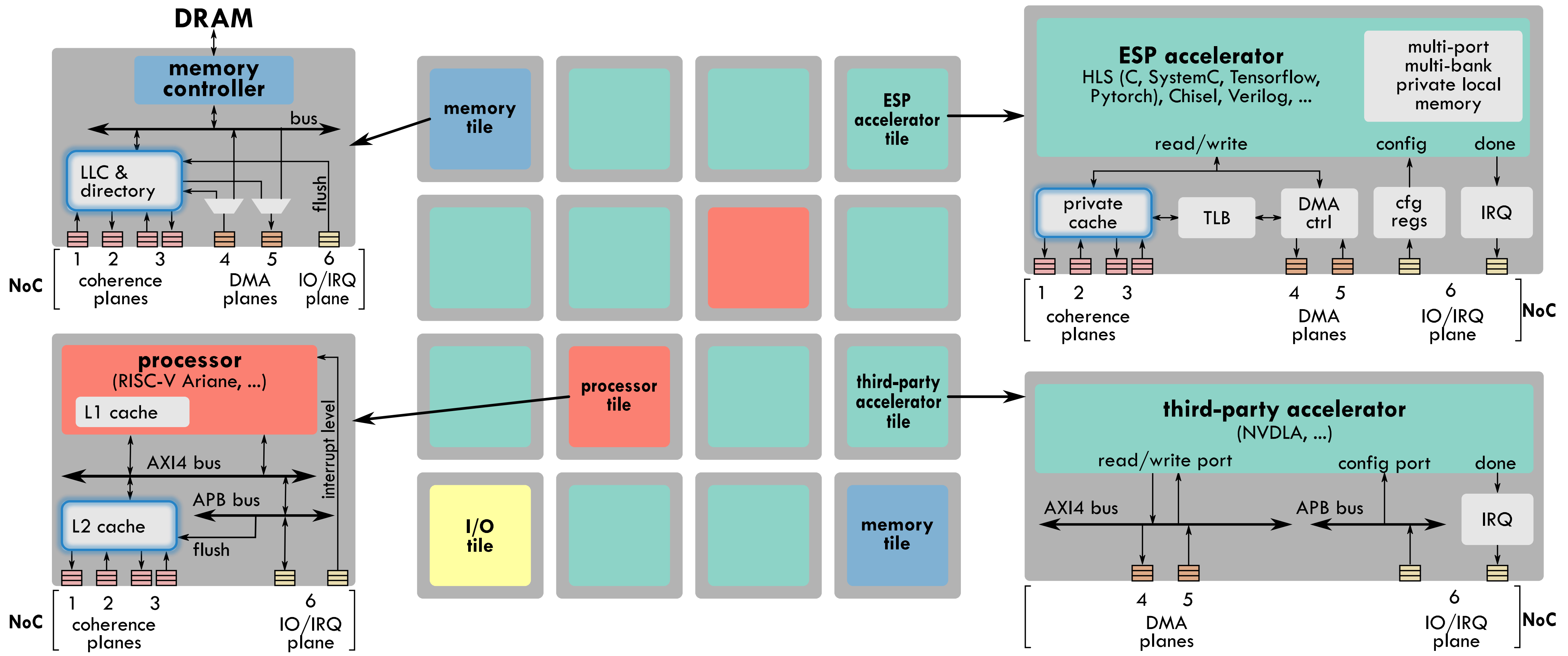
Protocol

- We modified a classic MESI directory-based cache-coherence protocol
  - Work over a NoC
  - Support all 4 coherence modes for accelerators

Implementation

- We implemented a cache hierarchy and we integrated it in ESP
  - Multi-core Linux execution
  - 4 coherence modes for accelerators
  - Run-time selection of the coherence mode for each accelerator
  - Run-time coexistence of heterogeneous coherence modes for accelerators

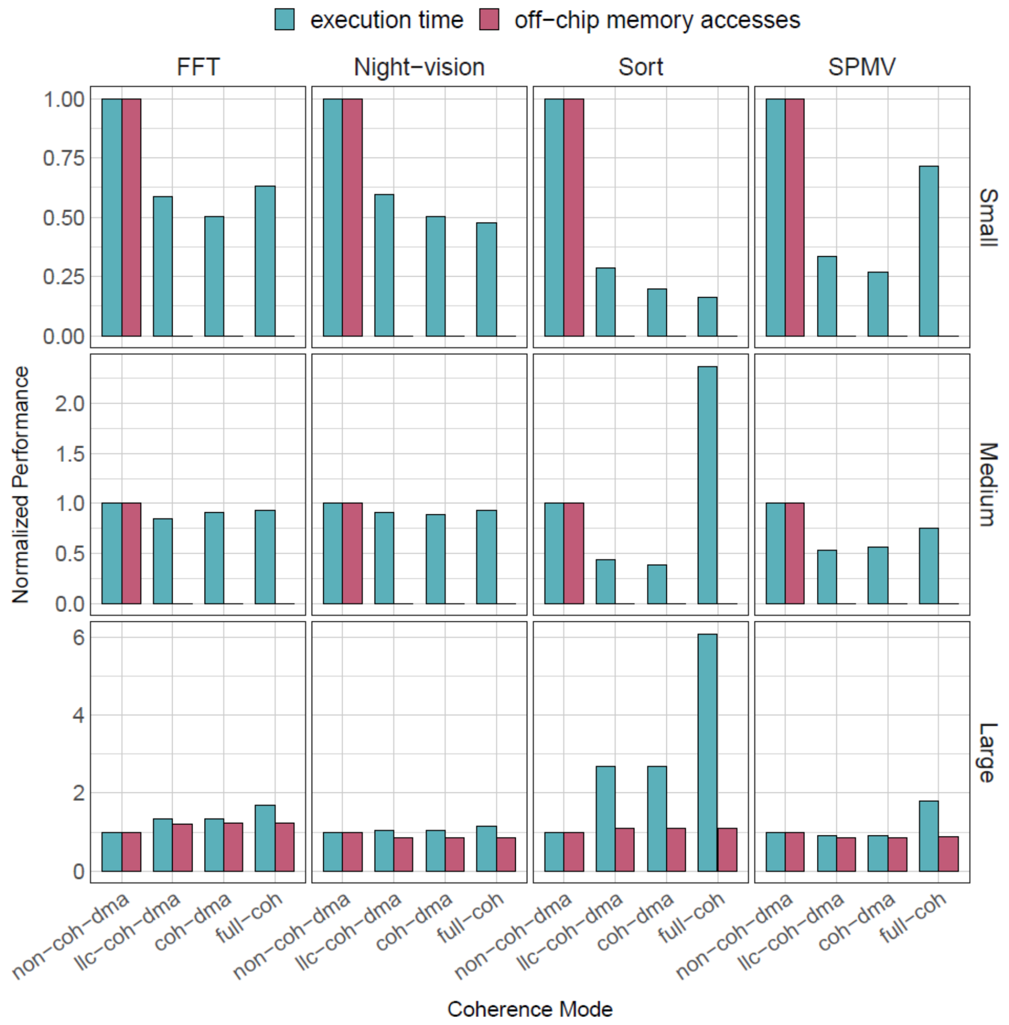
ESP CACHE HIERARCHY



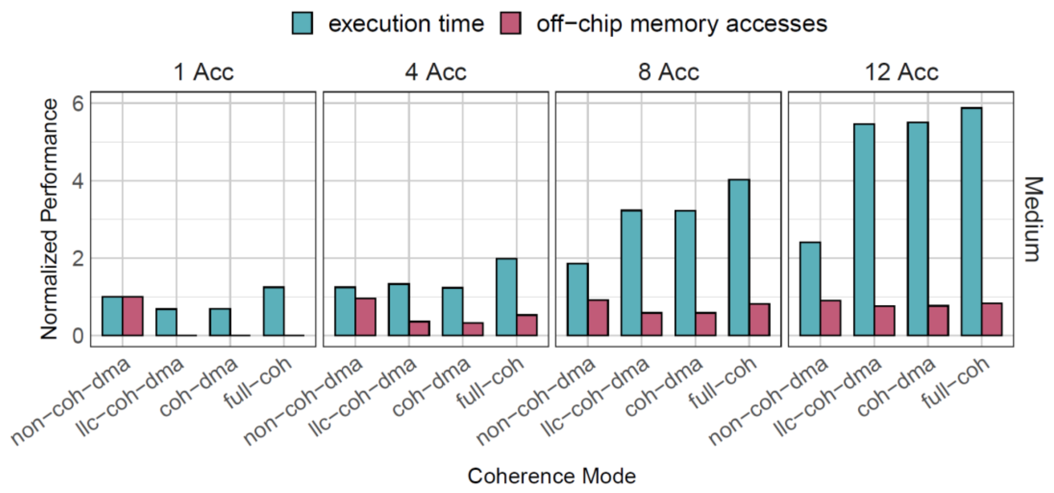
[Giri, IEEE Micro'18]

COHERENCE MODES EVALUATION

Single accelerator execution



Parallel accelerator execution



No absolute best coherence mode!

Depends on:

- Workload size and caches size
- Accelerator characteristics
- Number of active accelerators

[Giri, ASPDAC'18]

ADAPTIVE COHERENCE RECONFIGURATION

How to exploit heterogeneous and reconfigurable accelerator coherence?

We propose two adaptive approaches to select the best coherence mode dynamically at runtime

- Manually-tuned algorithm
- Learning-based approach (*Cohmeleon*)

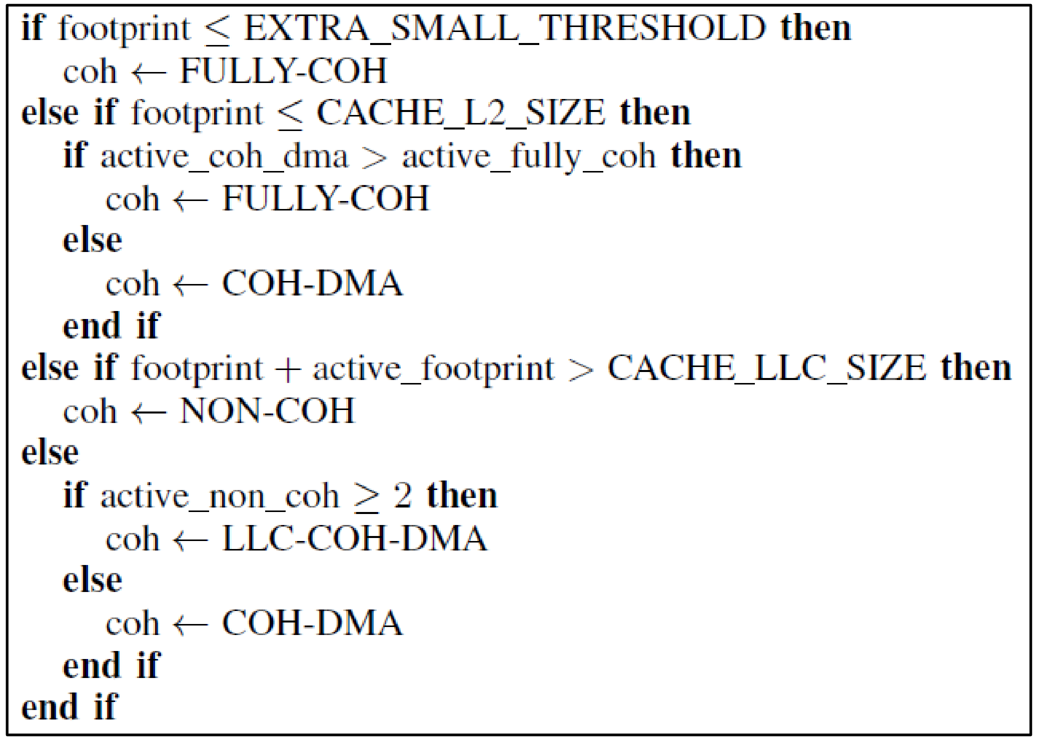
Framework

- Track system status
- At each accelerator invocation select coherence based on system status and invocation parameters
- Track performance to train learning model

Implementation

- Integrated in ESP's software stack
- Transparent to the accelerator programmer
- Negligible hw/sw overhead

Example of manually-tuned algorithm

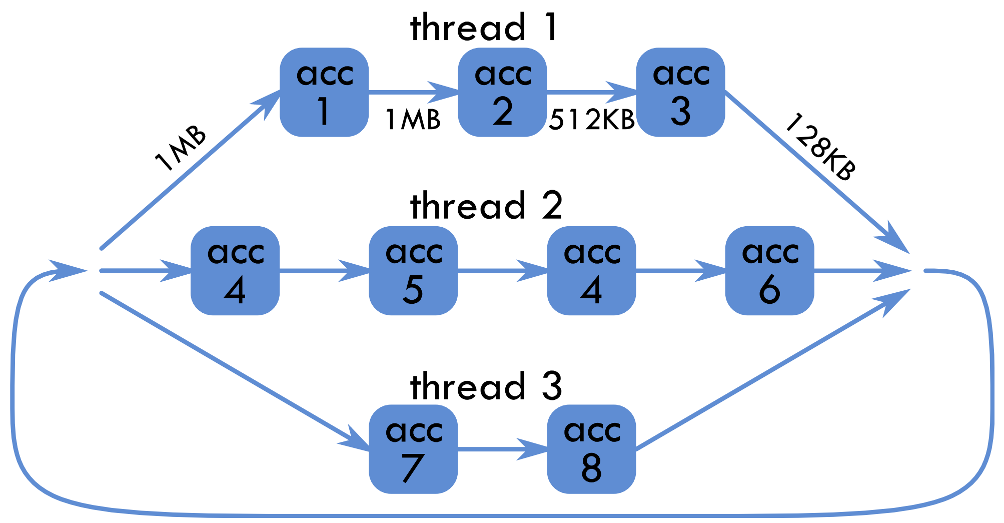


```
if footprint $\leq$ EXTRA_SMALL_THRESHOLD then
    coh $\leftarrow$ FULLY-COH
else if footprint $\leq$ CACHE_L2_SIZE then
    if active_coh_dma $>$ active_fully_coh then
        coh $\leftarrow$ FULLY-COH
    else
        coh $\leftarrow$ COH-DMA
    end if
else if footprint $+$ active_footprint $>$ CACHE_LLC_SIZE then
    coh $\leftarrow$ NON-COH
else
    if active_non_coh $\geq$ 2 then
        coh $\leftarrow$ LLC-COH-DMA
    else
        coh $\leftarrow$ COH-DMA
    end if
end if
```

ADAPTIVE COHERENCE EVALUATION

- Evaluation applications with multiple phases
- Highly-configurable traffic generator accelerator

Example of possible app phase

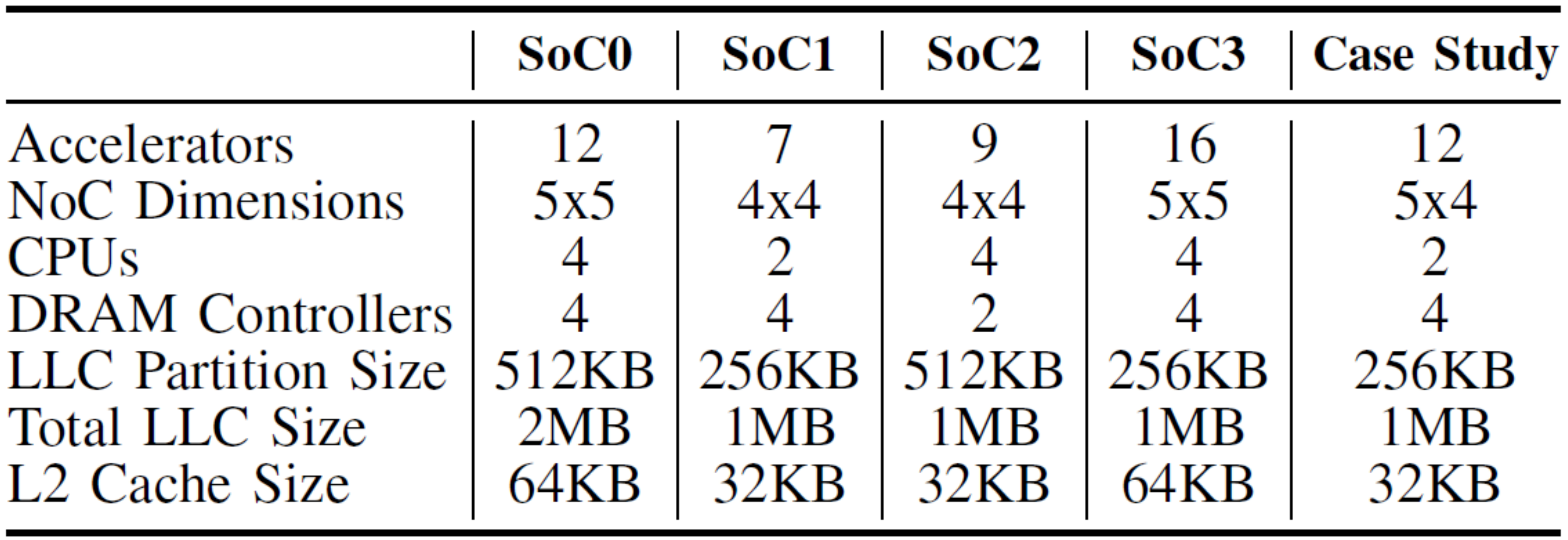


Example of possible app

App phases	Memory footprints sizes	Max active accelerators
1	variable	1
2	large	1
3	small	1
4	variable	6
5	large	6
6	small	6
7	variable	12
8	large	12
9	small	12

ADAPTIVE COHERENCE EVALUATION

FPGA-based prototyping of multiple ESP SoCs

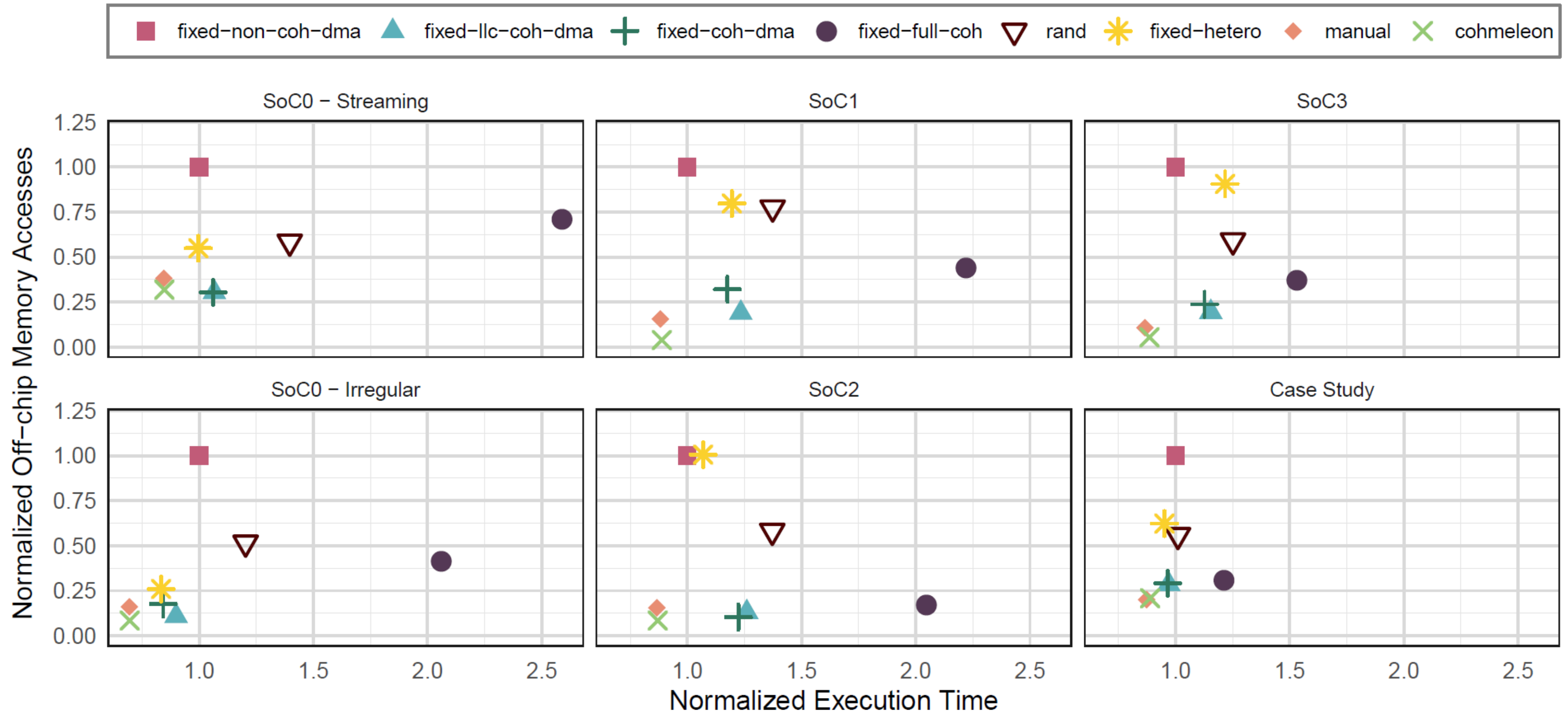


	SoC0	SoC1	SoC2	SoC3	Case Study
Accelerators	12	7	9	16	12
NoC Dimensions	5x5	4x4	4x4	5x5	5x4
CPUs	4	2	4	4	2
DRAM Controllers	4	4	2	4	4
LLC Partition Size	512KB	256KB	512KB	256KB	256KB
Total LLC Size	2MB	1MB	1MB	1MB	1MB
L2 Cache Size	64KB	32KB	32KB	64KB	32KB

ADAPTIVE COHERENCE EVALUATION

On average, both our approaches reduce

- **execution time** by 40%
- **off-chip accesses** by 74%



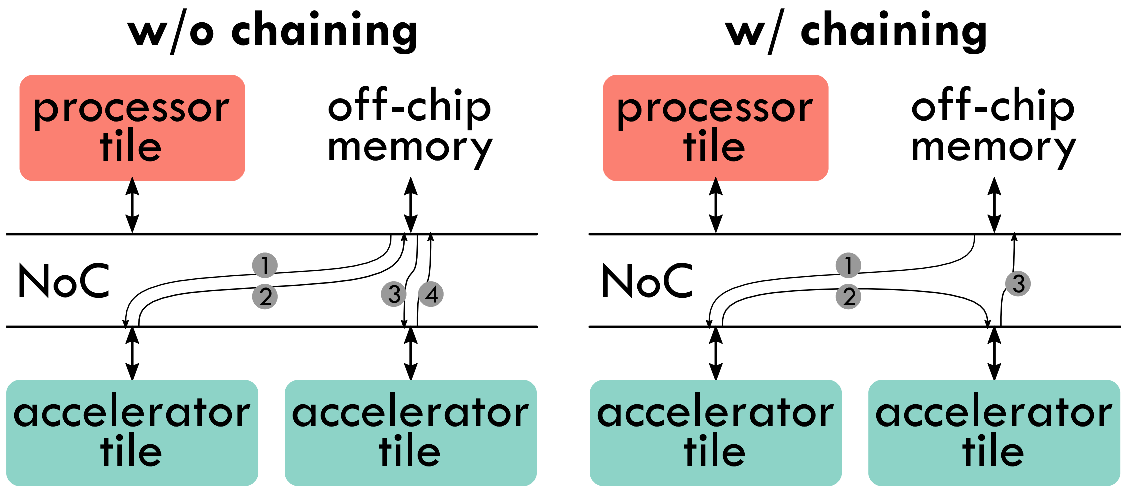
P2P COMMUNICATION

Accelerators can exchange data with:

- Shared memory
- Other accelerators (**new!**)

Benefits

- Avoid roundtrips to shared memory
- Fine-grained accelerators synchronization
  - Higher throughput
  - Lower invocation overhead



[Giri, DATE'20]

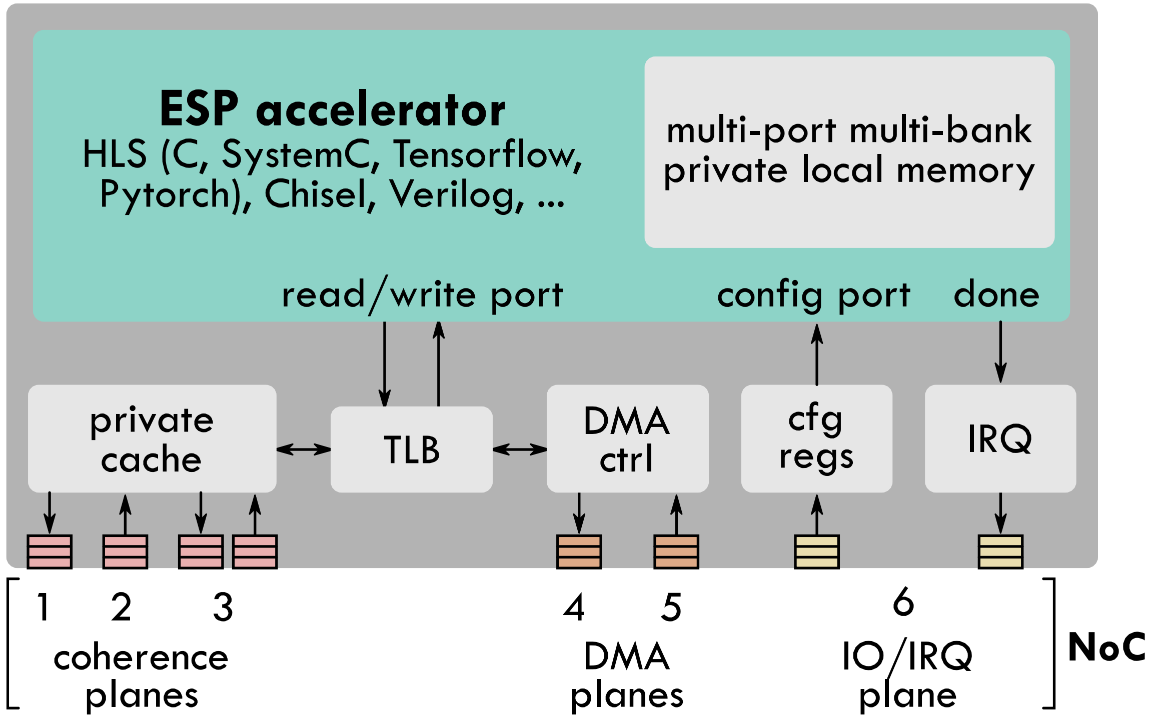
P2P COMMUNICATION

Implementation

- No need for additional queues or NoC channels
- Configured at invocation time
- On demand: initiated by receiver
- Support for one-to-many and many-to-one

Future work

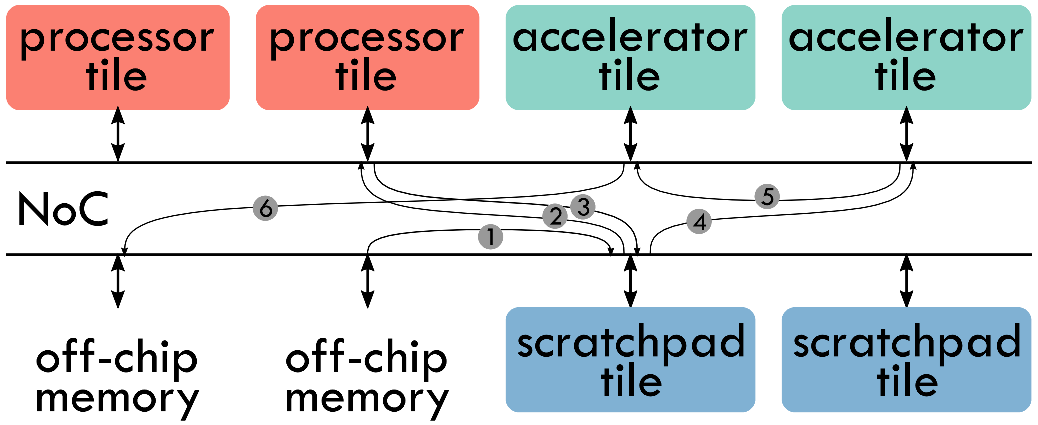
- Improve generality of one-to-many and many-to-one options
- Mixed p2p and regular communication



SHARED SCRATCHPAD

Two size thresholds for the accelerator scratchpad

- Minimum size to support the parallelism of the datapath
  - Must be tightly coupled with the datapath
- Minimum size to maximize reuse and minimize memory accesses
  - Must be on-chip



Goal: increase scratchpad utilization and reduce memory accesses

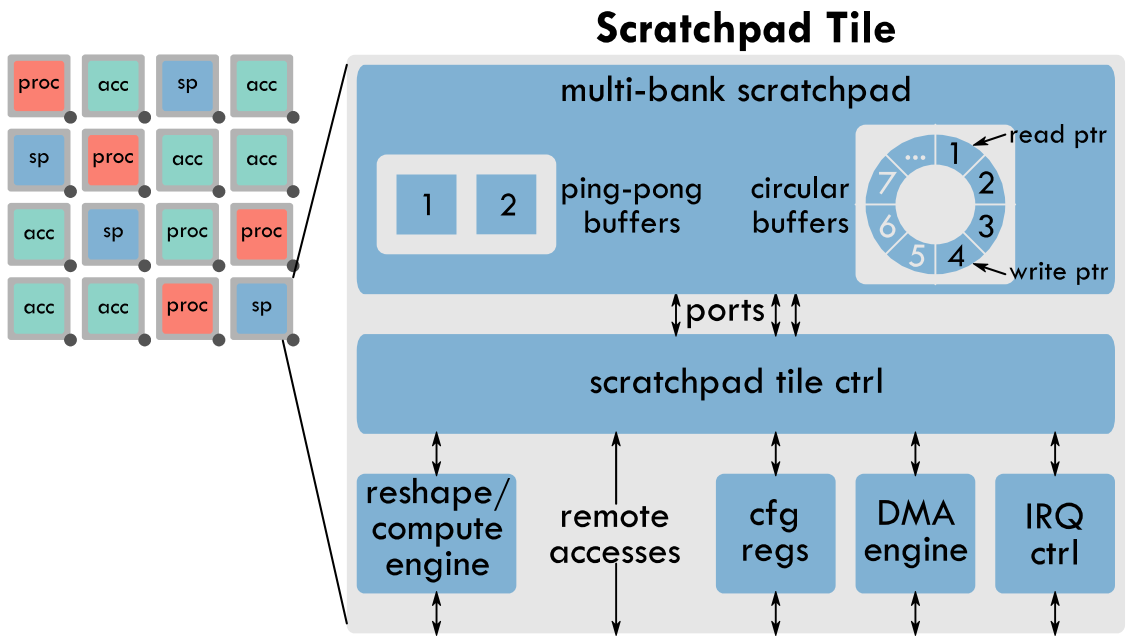
Solution: shared scratchpad tiles

[Lyons 2012] [Pellauer 2019]

SHARED SCRATCHPAD

Planned implementation

- Multi-bank scratchpad
- Highly configurable DMA engine
- Basic computation and data reshape engine
- Same programming model of accelerators
- Memory-mapped: accessible by processors and accelerators



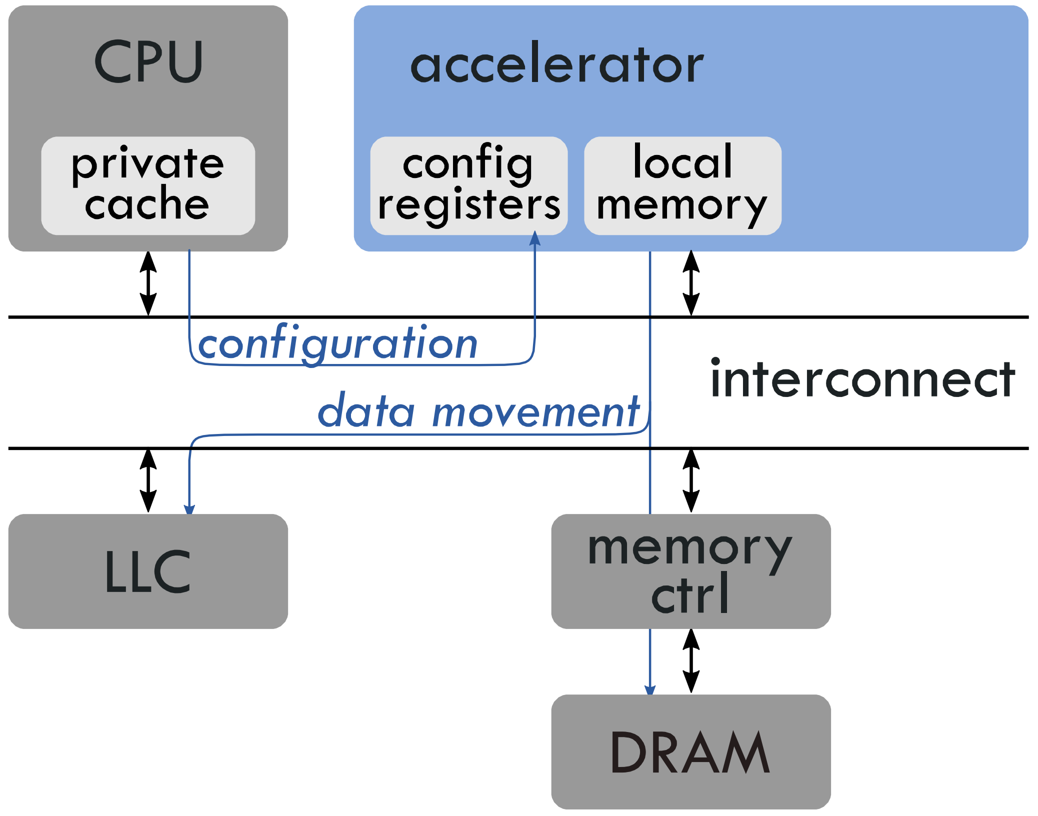
ACCELERATOR PROGRAMMING

Accelerator API library

ACCELERATOR PROGRAMMING MODEL

Device driver approach

- A user app calls the device driver
- The device driver
  - (optional) Flushes the caches
  - Configures the accelerator
  - Waits for the accelerator completion
  - Returns control to the user app

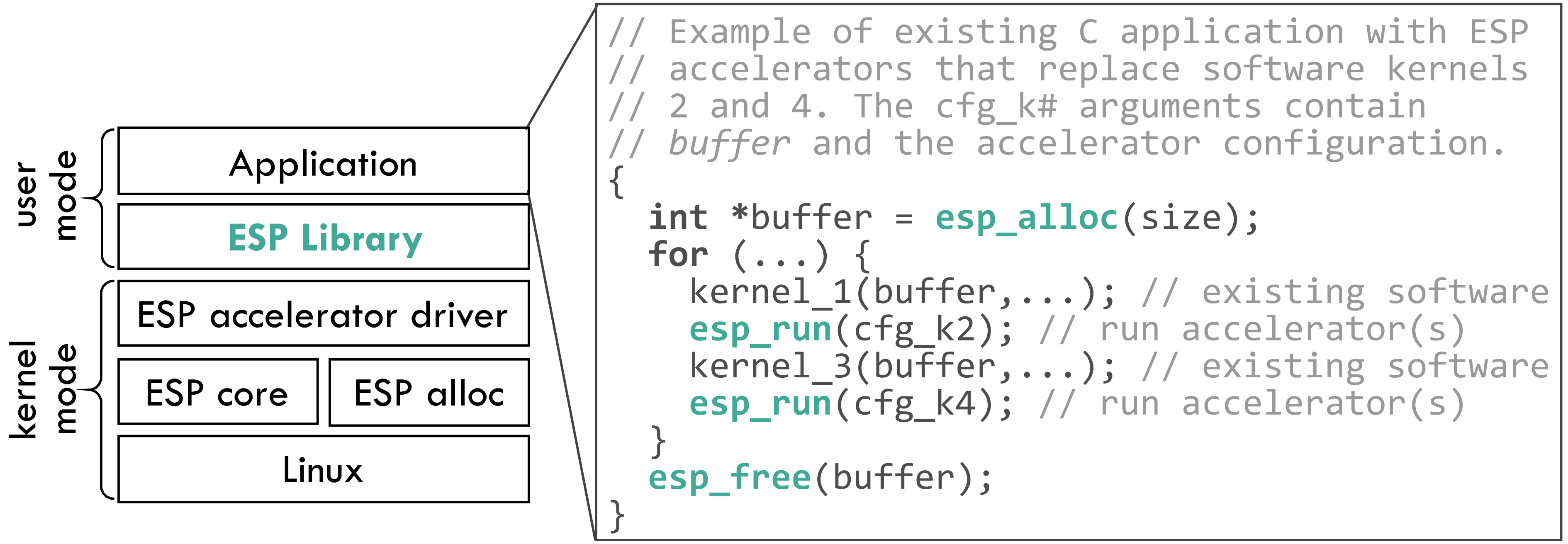


[Giri, DATE'20]

ACCELERATOR API

API for the invocation of accelerators from a user application

- Exposes only 3 functions to the programmer



```
// Example of existing C application with ESP
// accelerators that replace software kernels
// 2 and 4. The cfg_k# arguments contain
// buffer and the accelerator configuration.
{
  int *buffer = esp_alloc(size);
  for (...) {
    kernel_1(buffer,...); // existing software
    esp_run(cfg_k2); // run accelerator(s)
    kernel_3(buffer,...); // existing software
    esp_run(cfg_k4); // run accelerator(s)
  }
  esp_free(buffer);
}
```

ACCELERATOR API

Usage

- Can be targeted by existing applications with minimal modifications
- Can be targeted to automatically map tasks to accelerators

Accelerator Invocation

- Invokes accelerators through automatically-generated Linux device drivers
- Enables shared memory between processors and accelerators
  - No data copies
- Invoke multiple pipelines of accelerators in parallel

SoC services

- Simplifies the management of the hardware SoC services
  - Cache coherence, p2p, shared scratchpad

ACCELERATOR DESIGN AND INTEGRATION FLOW

C/C++ accelerator flow
Deep learning accelerator flow
Third-party accelerator flow

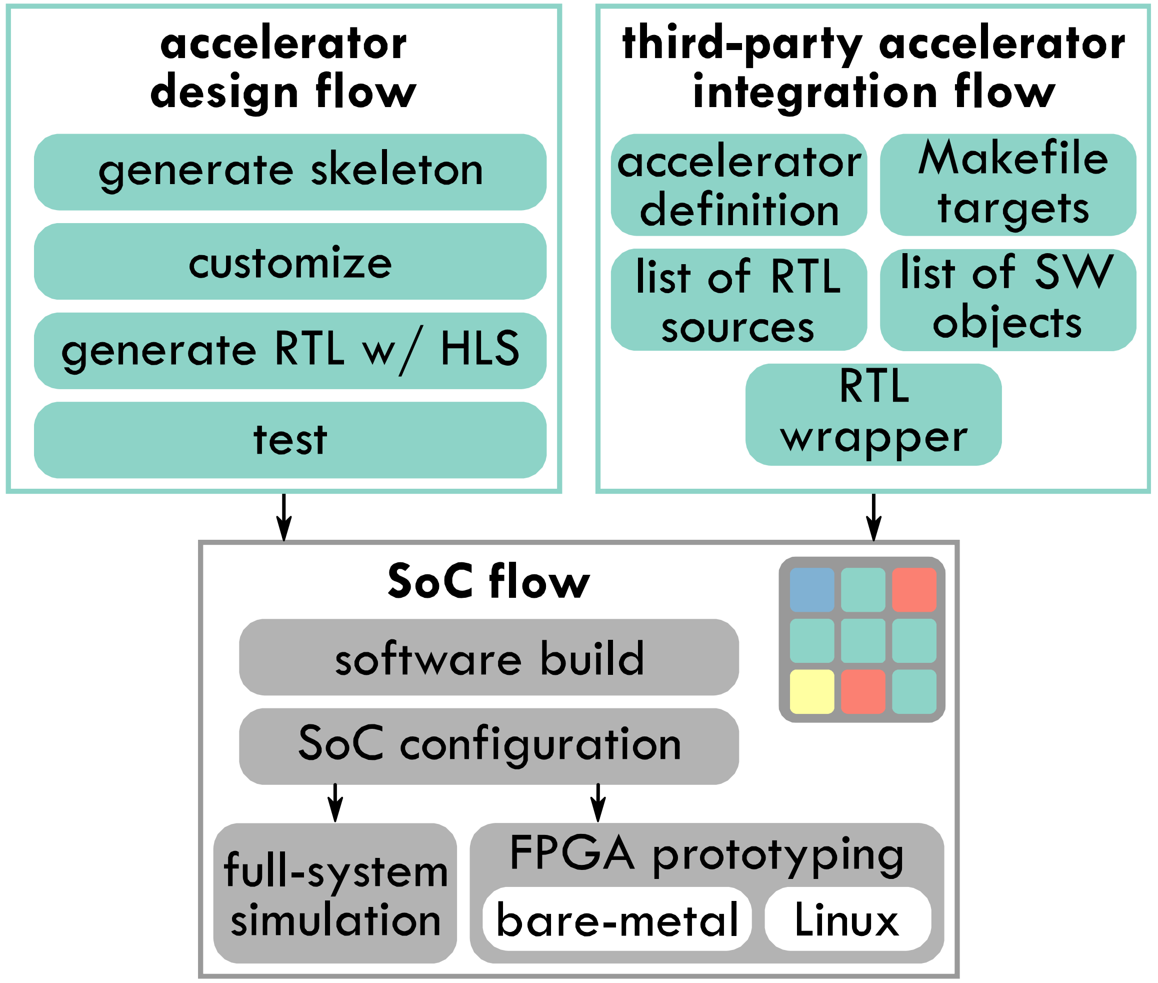
ACCELERATOR DESIGN AND INTEGRATION FLOWS

New accelerator design flows

- C/C++ with Vivado HLS
  - Tutorial: esp.cs.columbia.edu/docs/cpp_acc
- Keras/Pytorch/ONNX with hls4ml
  - Tutorial: esp.cs.columbia.edu/docs/hls4ml

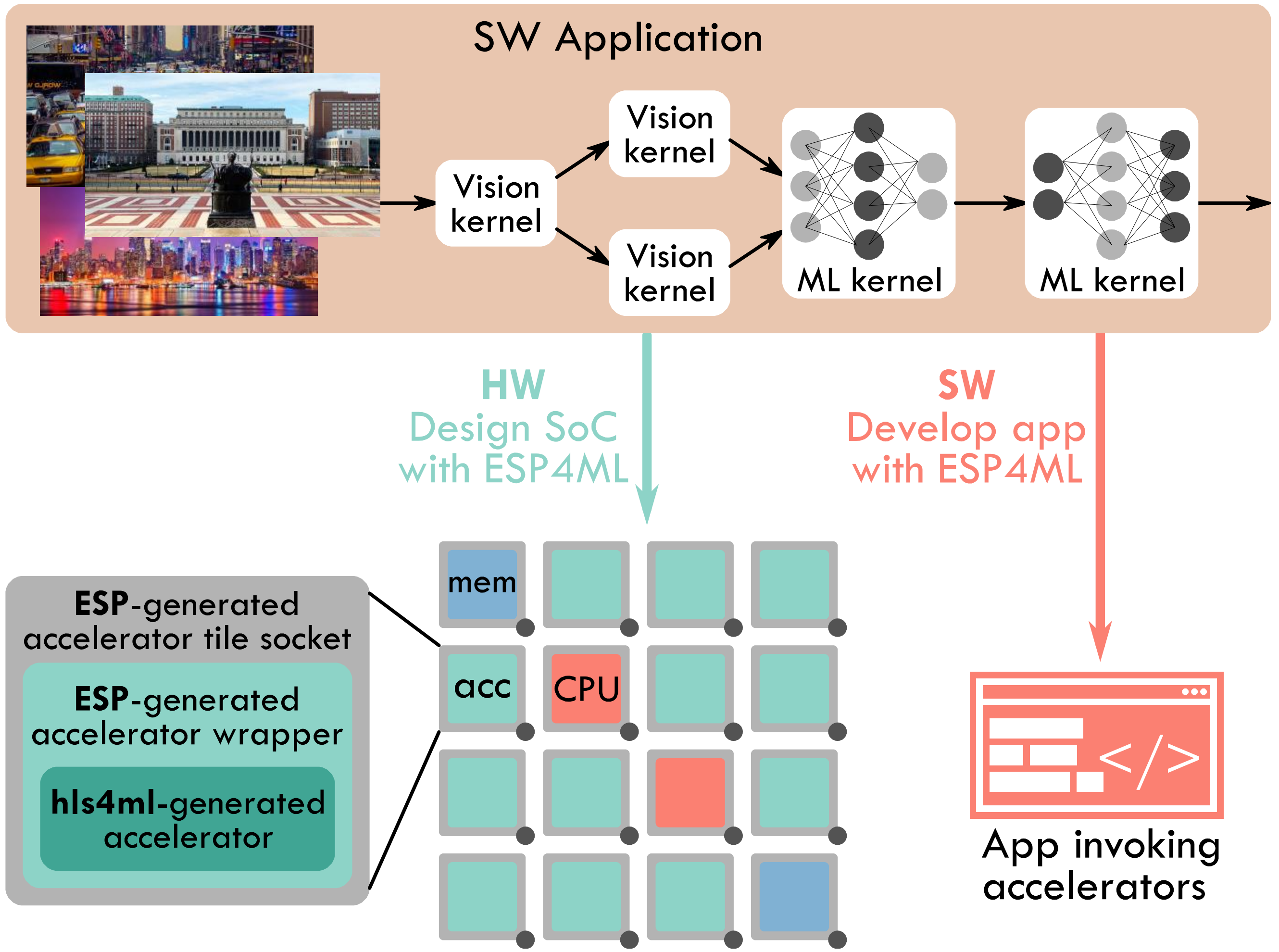
New third-party accelerator integration flow

- Tutorial: esp.cs.columbia.edu/docs/thirdparty_acc



ESP4ML

[Giri, DATE'20]

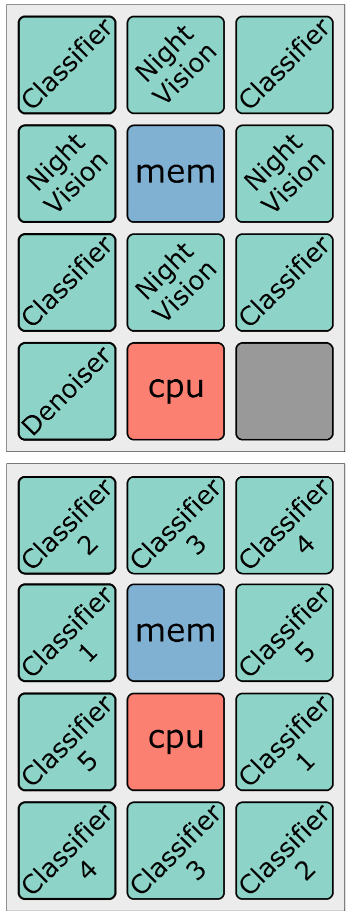


ESP4ML: CASE STUDY

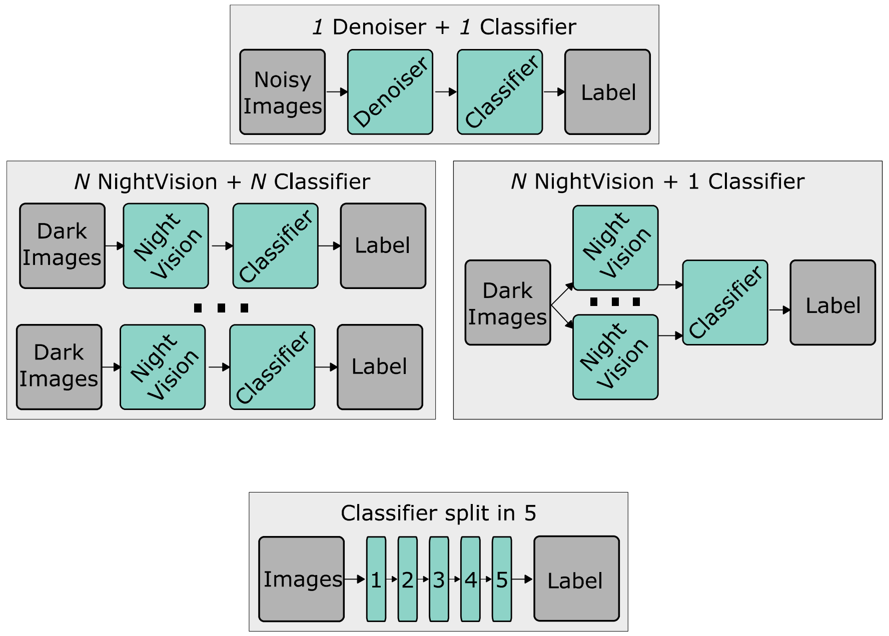
Featured accelerators

- Image classifier (hls4ml)
  - Street View House Numbers (SVHN) dataset from Google
- Denoiser (hls4ml)
  - Implemented as an autoencoder
- Night-vision (Stratus HLS)
  - Noise filtering, histogram, histogram equalization

SoCs



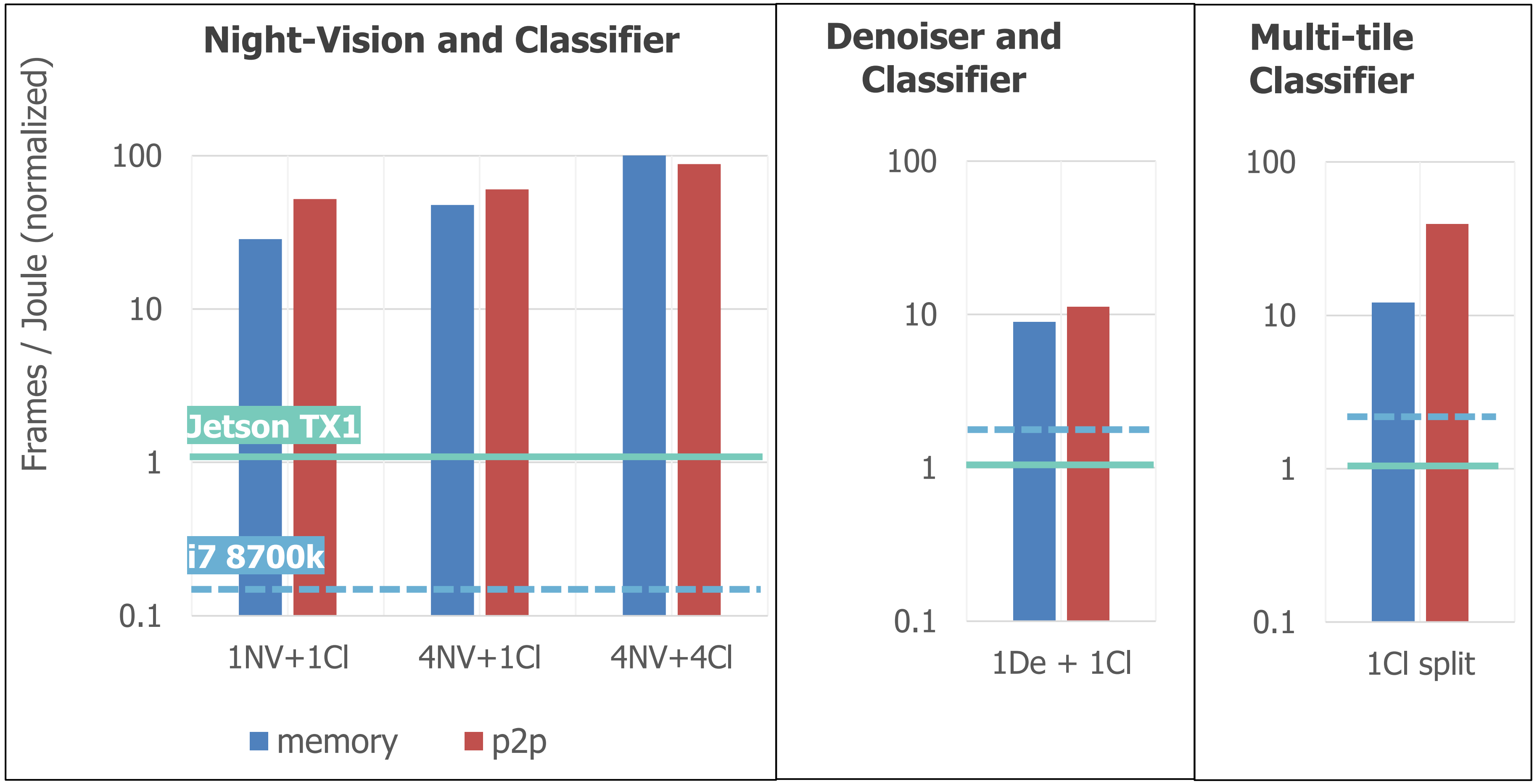
Applications



ESP4ML: ENERGY EFFICIENCY

Our SoCs achieve better energy efficiency than Jetson and i7.

Chaining accelerators brings additional energy savings.

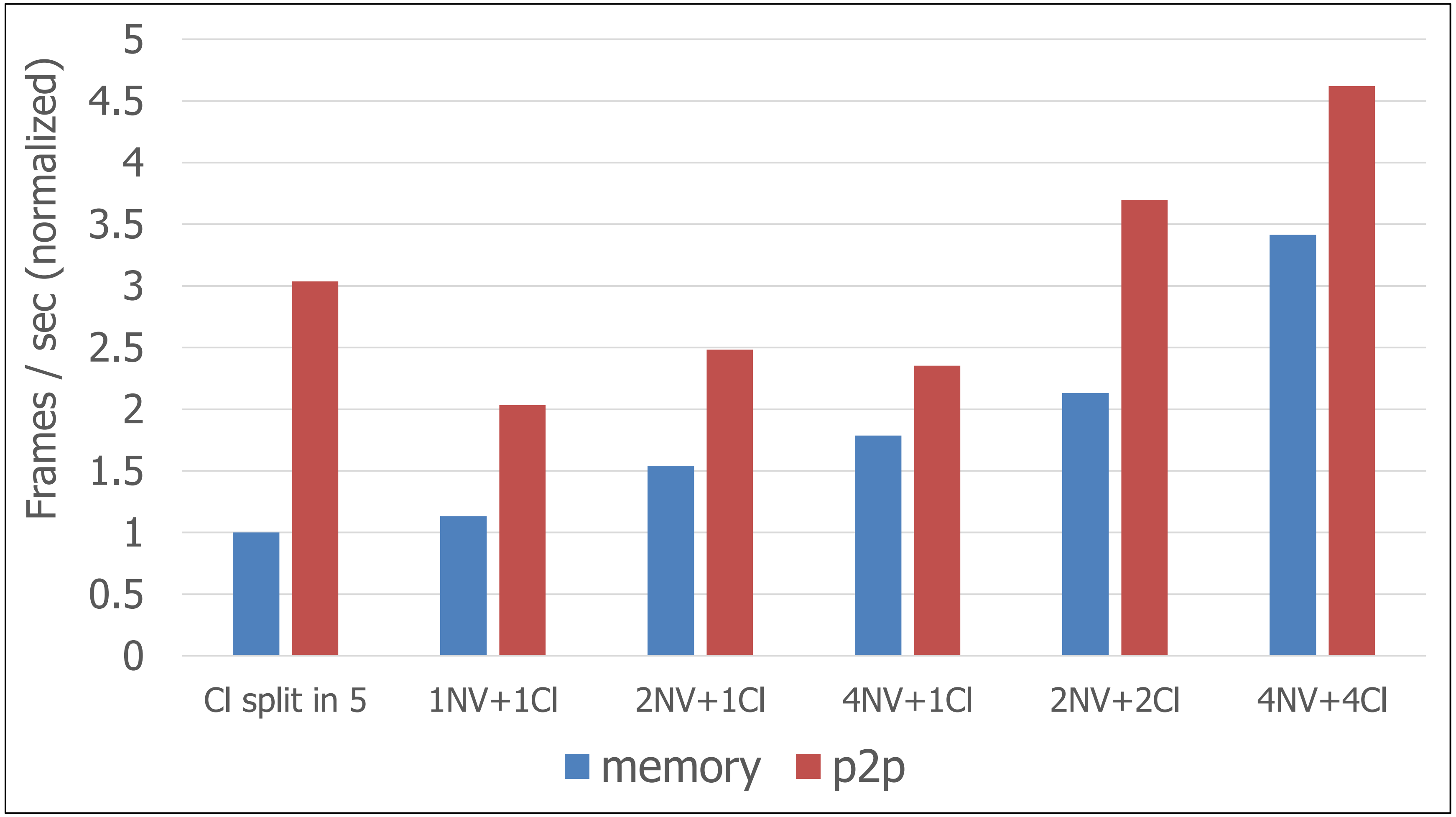


ESP4ML: PERFORMANCE

Performance increases to up to 4.5 times thanks to

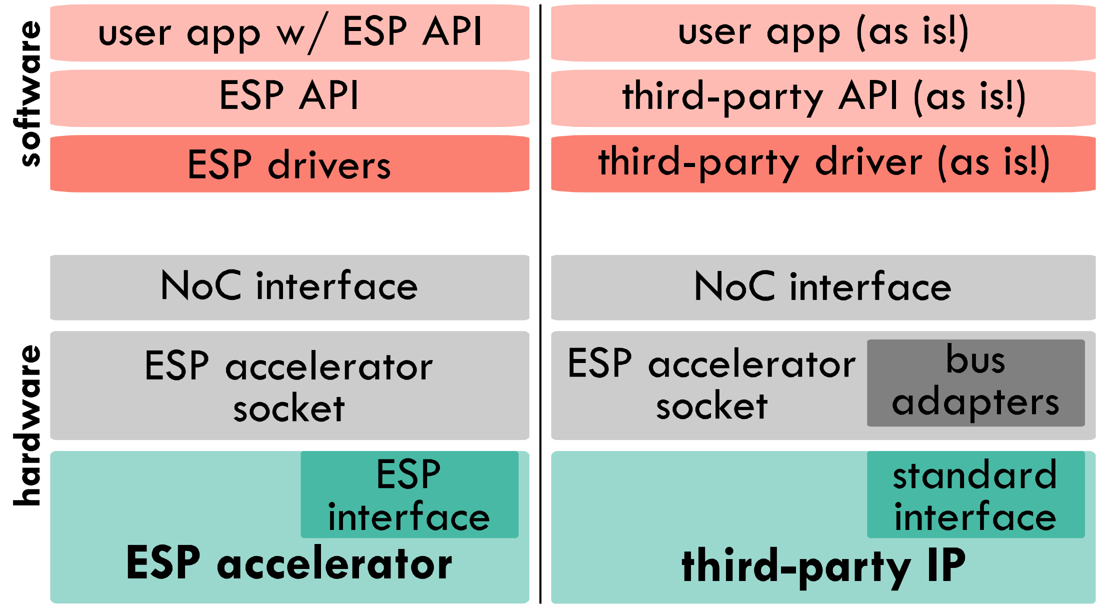
- Parallelization
- Chaining (p2p)

Accelerator chaining (p2p) reduces the memory accesses by 2-3 times

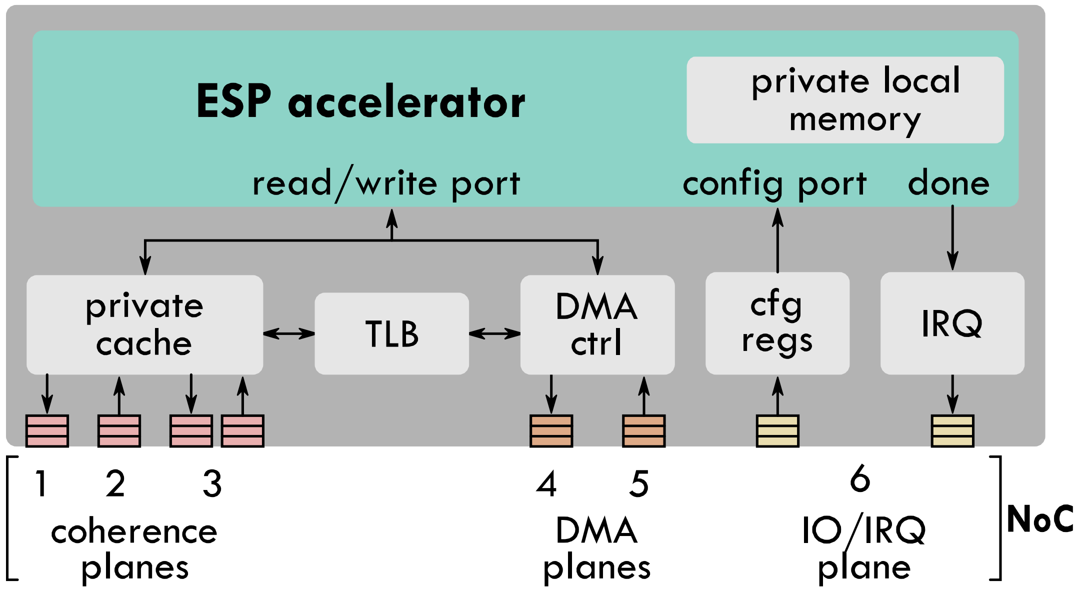


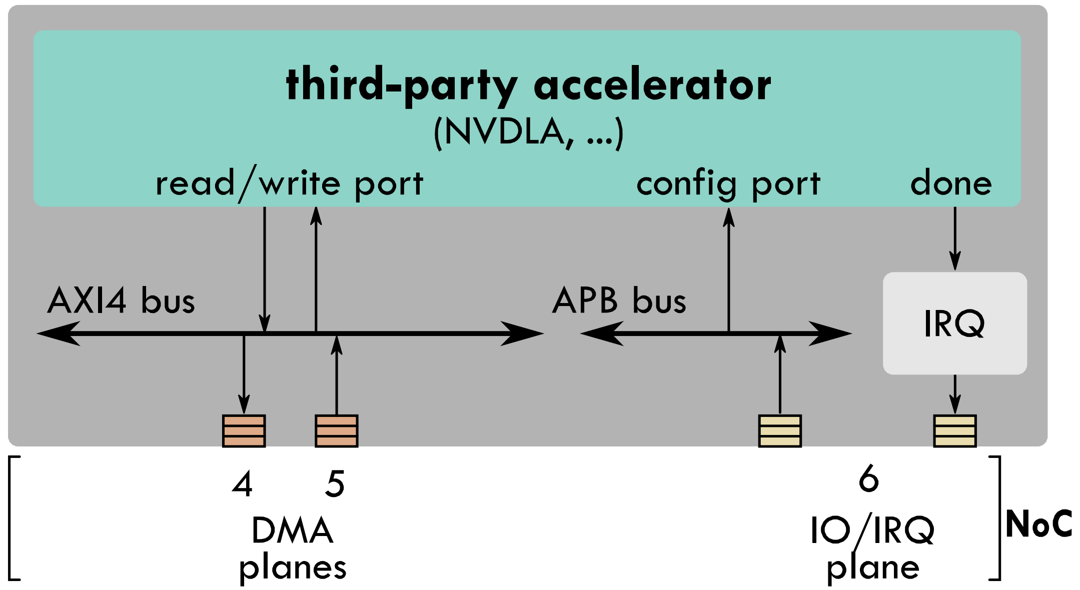
[Giri, CARRV'20]

THIRD-PARTY ACCELERATOR INTEGRATION FLOW



THIRD-PARTY ACCELERATOR TILE





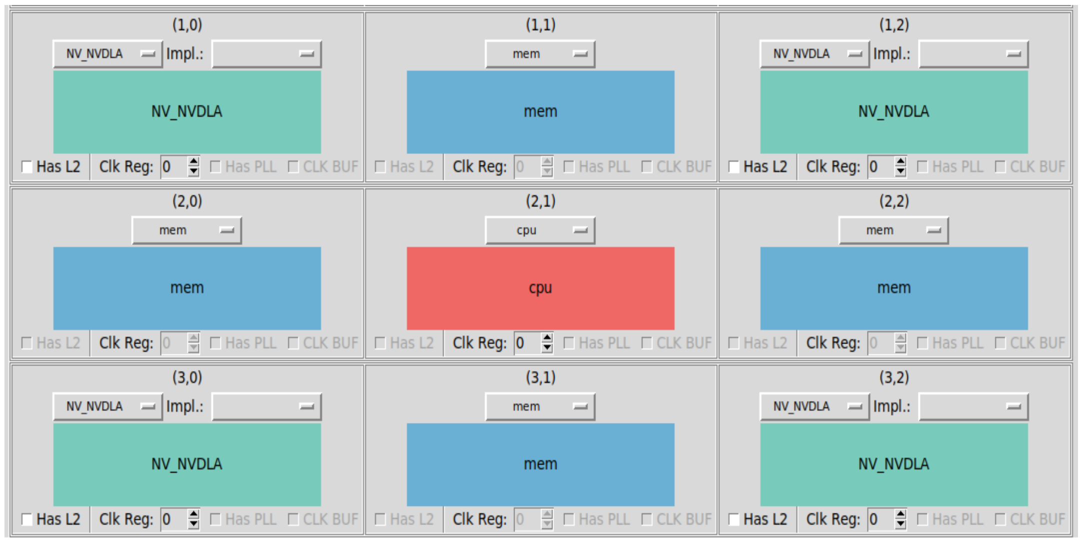
TPF: NVDLA CASE STUDY

NVIDIA Deep Learning Accelerator
- Open source
- Fixed function
- Highly configurable

NVDLA small
- 8-bit integer precision
- 64 MAC units
- 128 KB local memory

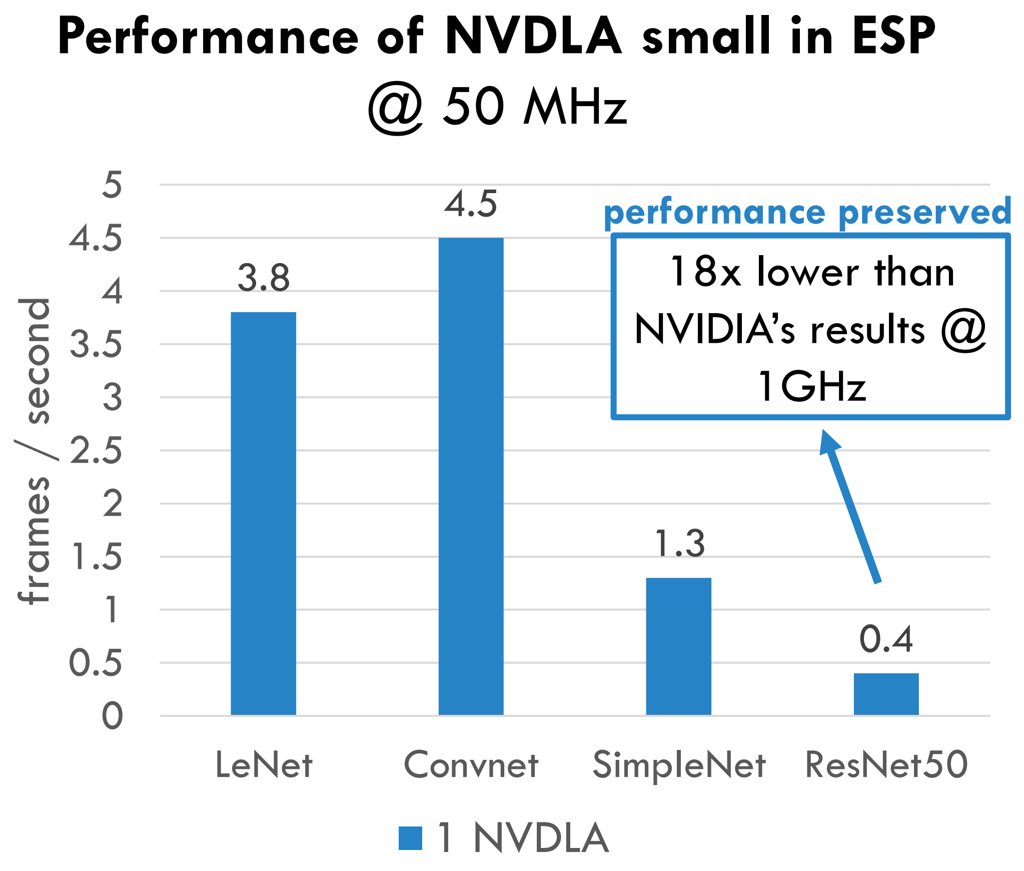
Integrated with the new third-party accelerator flow!

SoCs evaluated on FPGA (Xilinx XCVU440)
- Ariane core
- 1-4 NVDLA tiles
- 1-4 memory channels



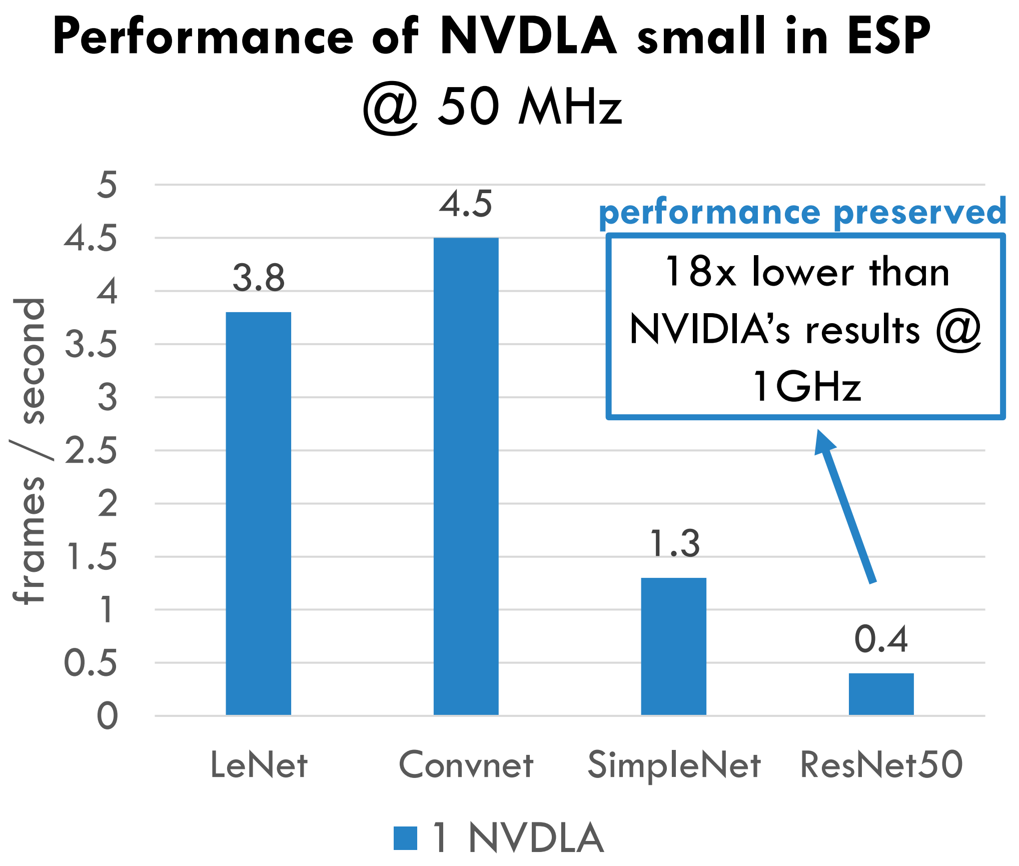
(1,0) NV_NVDLA Impl.: NV_NVDLA ☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF	(1,1) mem mem ☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF	(1,2) NV_NVDLA Impl.: NV_NVDLA ☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF
(2,0) mem mem ☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF	(2,1) cpu cpu ☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF	(2,2) mem mem ☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF
(3,0) NV_NVDLA Impl.: NV_NVDLA ☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF	(3,1) mem mem ☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF	(3,2) NV_NVDLA Impl.: NV_NVDLA ☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF

TPF: RESULTS

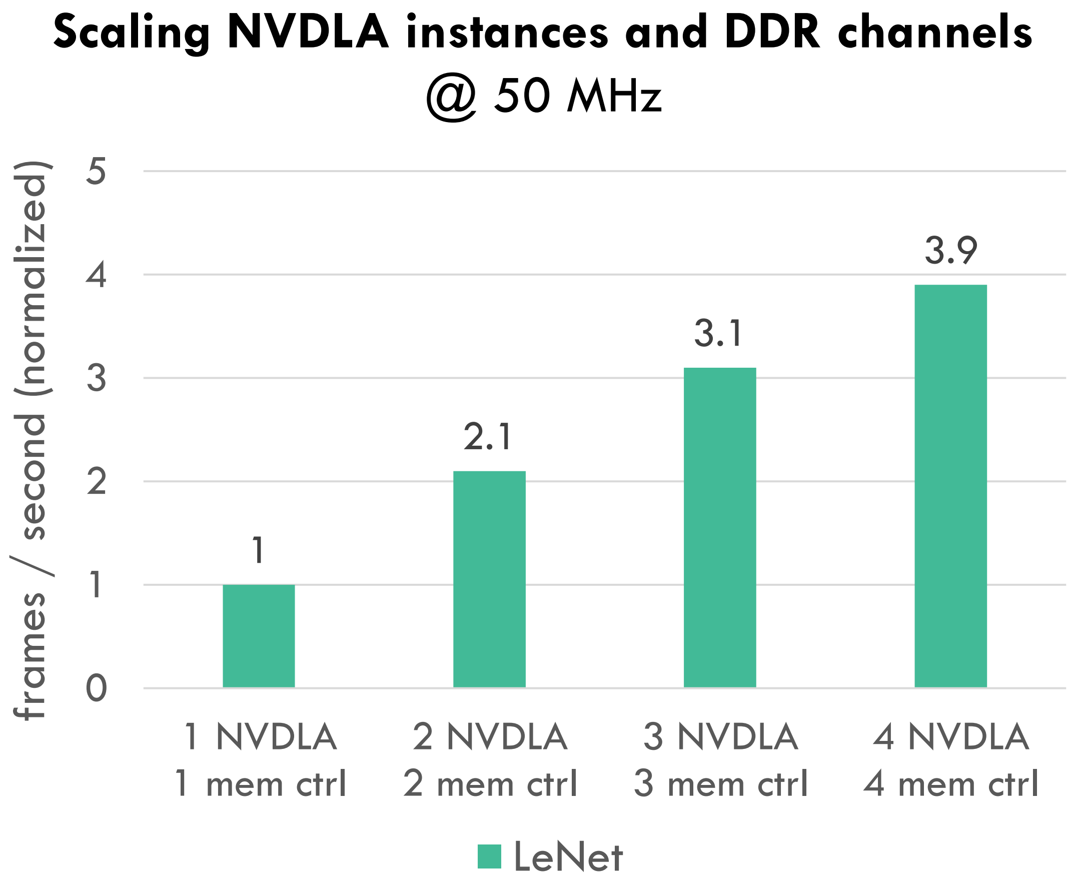


Model	Dataset	Layers	Input	Model Size
LeNet	MNIST	9	1x28x28	1.7MB
Convnet	CIFAR10	13	3x32x32	572KB
SimpleNet	MNIST	44	1x28x28	21MB
ResNet50	ILSVRC2012	229	3x224x224	98MB

TPF: RESULTS



Performance of NVDLA small in ESP
@ 50 MHz
frames / second
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
3.8
4.5
1.3
0.4
LeNet
Convnet
SimpleNet
ResNet50
performance preserved
18x lower than NVIDIA's results @ 1GHz
1 NVDLA



Scaling NVDLA instances and DDR channels
@ 50 MHz
frames / second (normalized)
5
4
3
2
1
0
1
2.1
3.1
3.9
1 NVDLA 1 mem ctrl
2 NVDLA 2 mem ctrl
3 NVDLA 3 mem ctrl
4 NVDLA 4 mem ctrl
LeNet

OPEN-SOURCE CONTRIBUTION

ESP release

Accelerator benchmark suite

OPEN-SOURCE SOC DESIGN PLATFORMS

Chipyard, Pulp, Openpiton, BlackParrot

- Processor-centric
- Limited support for loosely-coupled accelerators

We released ESP in open-source,
including the contributions of this work.

We integrated the ESP accelerators in the DECADES simulator and SoC platform
(decades.cs.princeton.edu, github.com/PrincetonUniversity/MosaicSim)

CHIPYARD
github.com/ucb-bar/chipyard

PULP
Parallel Ultra Low Power
pulp-platform.org

OpenPiton
parallel.princeton.edu/openpiton

BlackParrot
github.com/black-parrot/black-parrot

[Mantovani, ICCAD'20]

ESP RELEASE

Release: github.com/sld-columbia/esp

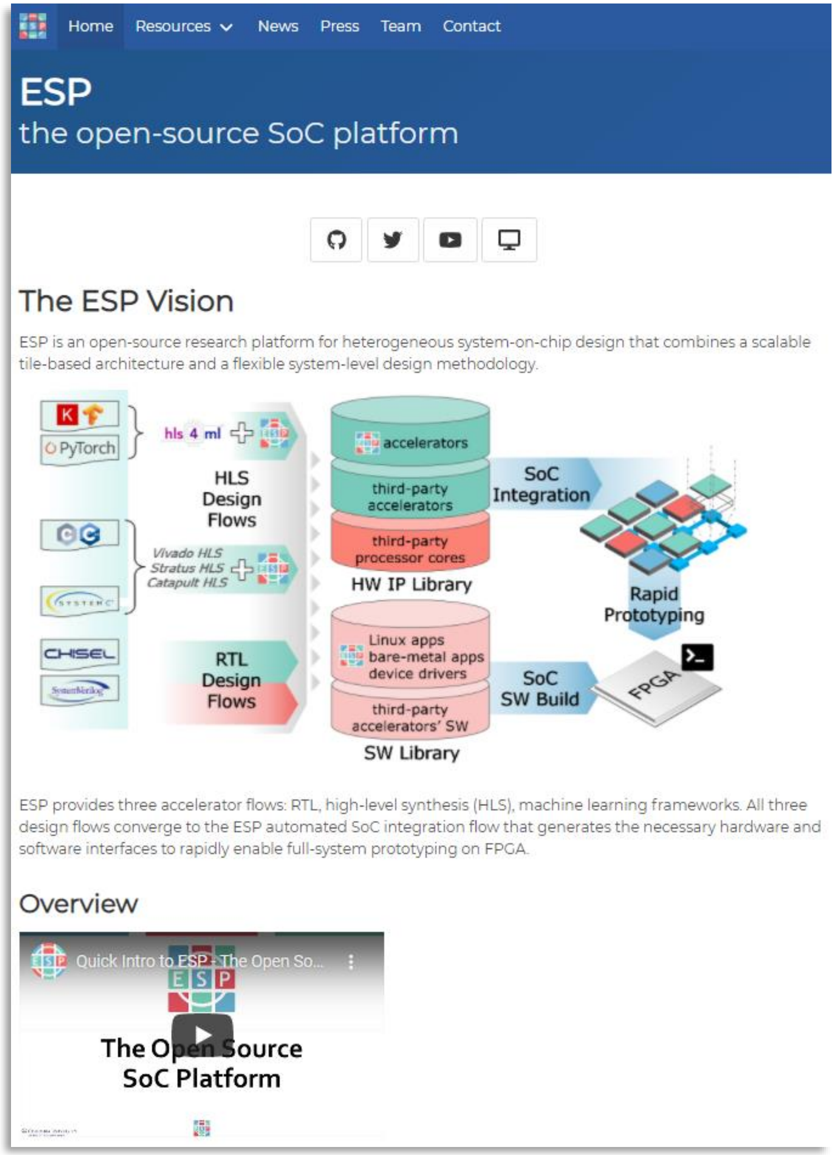
Website: esp.cs.columbia.edu

Channel: 13 videos

Documentation: 9 hands-on tutorials

Conference Tutorials: ESWeek'19, ASPLOS'20, MICRO'20, ISPASS'21, ASPLOS'21

Teaching: Class projects of CSEE E6868 at Columbia University



ESP

the open-source SoC platform

The ESP Vision

ESP is an open-source research platform for heterogeneous system-on-chip design that combines a scalable tile-based architecture and a flexible system-level design methodology.

ESP provides three accelerator flows: RTL, high-level synthesis (HLS), machine learning frameworks. All three design flows converge to the ESP automated SoC integration flow that generates the necessary hardware and software interfaces to rapidly enable full-system prototyping on FPGA.

Overview

ACCELERATOR BENCHMARK SUITE

Motivation

- Many accelerators designed and integrated in ESP over the years
- Lack of open-source accelerator benchmark suites

Release a **new accelerator benchmark suite**

- Multiple specification languages and tools
- Seamless SoC integration (ESP)
- Rapid FPGA prototyping
  - ESP SoCs or single-accelerator on Xilinx Zynq MPSoC
- Included software stack for accelerator programming

ACCELERATOR BENCHMARK SUITE

Benchmark Suite	# Apps	Language	HLS-Ready	HLS Scripts	Optimized	Invocation SW	Test on FPGA	SoC Integration
Cortex	20	C						
PERFECT	15	C						
CHStone	12	C	✓					
S2CBench	21	SystemC	✓					
MachSuite	12	C	✓	✓ (Vivado HLS)				
Rosetta	6	C/C++	✓	✓ (Vivado HLS)	✓	✓	✓	
Proposed Suite	>20	C/C++ SystemC Chisel* Keras Pytorch ONNX	✓	✓ (Vivado HLS Catapult HLS Stratus HLS hls4ml Chisel*)	✓	✓	✓	✓

** Generating RTL code from Chisel does not require HLS*

PUBLICATIONS

Davide Giri, Paolo Mantovani, Luca P. Carloni
Accelerators and Coherence: An SoC Perspective
IEEE Micro (Special Issue: Hardware Acceleration), 2018

Davide Giri, Paolo Mantovani, Luca P. Carloni
NoC-Based Support of Heterogeneous Cache-Coherence Models for Accelerators
NOCS (IEEE/ACM International Symposium on Networks-on-Chip), 2018

Davide Giri, Paolo Mantovani, Luca P. Carloni
Runtime Reconfigurable Memory Hierarchy in Embedded Scalable Platforms
ASPDAC (Asia and South Pacific Design Automation Conference), invited, 2019

Luca P. Carloni, Emilio Cota, Giuseppe Di Guglielmo, Davide Giri, Jihye Kwon, Paolo Mantovani, Luca Piccolboni, Michele Petracca
Teaching Heterogeneous Computing with System-Level Design Methods
WCAE (Workshop on Computer Architecture Education), 2019

Davide Giri, Kuan-lin Chiu, Giuseppe Di Guglielmo, Paolo Mantovani, Luca P. Carloni
ESP4ML: Platform-Based Design of Systems-on-Chip for Embedded Machine Learning
DATE (Design, Automation and Test in Europe Conference), best paper nominee, 2020

Davide Giri, Kuan-Lin Chiu, Guy Eichler, Paolo Mantovani, Nandhini Chandramoorthy, Luca P. Carloni
Ariane + NVDLA: Seamless Third-Party IP Integration with ESP
CARRV (Workshop on Computer Architecture Research with RISC-V), 2020

P. Mantovani, Davide Giri, G. Di Guglielmo, L. Piccolboni, J. Zuckerman, E. G. Cota, M. Petracca, C. Pilato, L. P. Carloni
Agile SoC Development with Open ESP
ICCAD (IEEE/ACM International Conference On Computer Aided Design), invited, 2020

O. Matthews, A. Manocha, Davide Giri, M. Orenes-Vera, E. Tureci, T. Sorensen, T. J. Ham, J. L. Aragon, L. P. Carloni, M. Martonosi
MosaicSim: A Lightweight, Modular Simulator for Heterogeneous Systems
ISPASS (IEEE International Symposium on Performance Analysis of Systems and Software), best paper nominee, 2020

Paolo Mantovani, Robert Margelli, Davide Giri, Luca P. Carloni
HL5: A 32-bit RISC-V Processor Designed with High-Level Synthesis
CICC (IEEE Custom Integrated Circuits Conference), invited, 2020

THANK YOU

AGILE DESIGN AND INTEGRATION OF ACCELERATORS IN HETEROGENEOUS SOC ARCHITECTURES

PhD Proposal Exam

Davide Giri

January 21, 2021

REFERENCES

[Alsop 2018] J. Alsop, M. Sinclair, S. Adve, **"Spandex: A Flexible Interface for Efficient Heterogeneous Coherence,"** *International Symposium on Computer Architecture (ISCA)*, 2018.

[Cascaval 2010] C. Cascaval, S. Chatterjee, H. Franke, K. J. Gildea, P. Pattnaik, **"A Taxonomy of Accelerator Architectures and their Programming Models,"** *IBM Journal of Research and Development*, 2010.

[Cong 2012] J. Cong, M. A. Ghodrat, M. Gill, B. Grigorian, G. Reinman, **"Architecture Support for Accelerator-Rich CMPs,"** *Design Automation Conference (DAC)*, 2012.

[Chen 2013] Y. Chen, J. Cong, M. A. Ghodrat, M. Huang, C. Liu, B. Xiao, Y. Zou, **"Accelerator-rich CMPs: From Concept to Real Hardware,"** *International Conference on Computer Design (ICCD)*, 2013.

[Cota 2015] E. G. Cota, P. Mantovani, G. Di Guglielmo, L. P. Carloni, **"An Analysis of Accelerator Coupling in Heterogeneous Architectures,"** *Design Automation Conference (DAC)*, 2015.

[Fu 2015] Y. Fu, T. M. Nguyen, D. Wentzlaff, **"Coherence Domain Restriction on Large Scale Systems,"** *International Symposium on Microarchitecture (MICRO)*, 2015.

[Kelm 2011] J. H. Kelm, D. R. Johnson, W. Tuohy, S. S. Lumetta, S. J. Patel, **"Cohesion: An Adaptive Hybrid Memory Model for Accelerators,"** *IEEE Micro*, 2011.

[Komuravelli 2015] R. Komuravelli, M. D. Sinclair, J. Alsop, M. Huzaifa, M. Kotsifakou, P. Srivastava, S. V. Adve, V. S. Adve, **"Stash: Have Your Scratchpad and Cache It Too,"** *International Symposium on Computer Architecture (ISCA)*, 2015.

[Kumar 2015] S. Kumar, A. Shriraman, N. Vedula, **"Fusion: Design Tradeoffs in Coherent Cache Hierarchies for Accelerators,"** *International Symposium on Computer Architecture (ISCA)*, 2015.

[Ham 2015] T. J. Ham, J. L. Aragón, M. Martonosi, **"DeSC: Decoupled Supply-compute Communication Management for Heterogeneous Architectures,"** *International Symposium on Microarchitecture (MICRO)*, 2015.

[Lustig 2013] D. Lustig and M. Martonosi, "**Reducing GPU Offload Latency Via Fine-grained CPU-GPU Synchronization,**" *International Symposium on High-Performance Computer Architecture (HPCA)*, 2013.

[Lyons 2012] M. Lyons, M. Hempstead, G. Wei, D. Brooks, "**The Accelerator Store: A Shared Memory Framework for Accelerator-based Systems,**" *ACM Transactions on Architecture and Code Optimization (TACO)*, 2012.

[Peemen 2013] M. Peemen, A. A. A. Setio, B. Mesman, H. Corporaal, "**Memory-centric Accelerator Design for Convolutional Neural Networks,**" *International Conference on Computer Design (ICCD)*, 2013.

[Pellauer 2019] M. Pellauer, Y. S. Shao, J. Clemons, N. Crago, K. Hegde, R. Venkatesan, S. W. Keckler, C. W. Fletcher, J. Emer, "**Buffets: An Efficient and Composable Storage Idiom for Explicit Decoupled Data Orchestration,**" *Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019.

[Mantovani 2016] P. Mantovani, E. G. Cota, C. Pilato, G. Di Guglielmo, L. P. Carloni, "**Handling Large Data Sets for High-performance Embedded Applications in Heterogeneous Systems-on-Chip,**" *International Conference on Compilers, Architectures, and Synthesis of Embedded Systems (CASES)*, 2016.

[Shao 2015] Y. S. Shao, D. Brooks, "**Research Infrastructures for Hardware Accelerators,**" *Synthesis Lectures on Computer Architecture, Morgan & Claypool*, 2015, chapters 1-2.

[Shao 2016] Y. S. Shao, S. L. Xi, V. Srinivasan, G. Wei, D. Brooks, "**Co-designing Accelerators and SoC Interfaces using Gem5-Aladdin,**" *International Symposium on Microarchitecture (MICRO)*, 2016.