System-Level Computer Architecture Research with Open ESP

Joseph Zuckerman

Davide Giri, Paolo Mantovani

Maico Cassel Dos Santos, Kuan-Lin Chiu, Giuseppe Di Guglielmo,

Guy Eichler, Jihye Kwon, Luca Piccolboni, Biruk Seoyum, Gabriele Tombesi

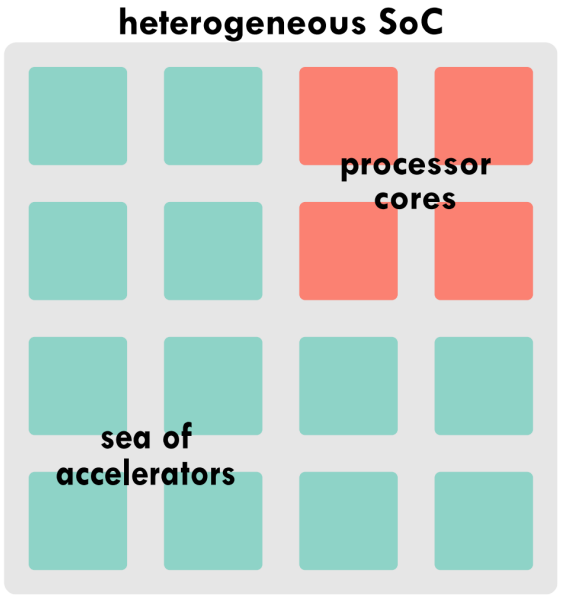
Luca P. Carloni

Columbia University
IN THE CITY OF NEW YORK

CS@CU Computer Science

The Age of Heterogeneous Computing

- SoCs are increasingly heterogeneous
  - CPUs, GPUs, accelerators, I/O peripherals, sensors…
  - Across computing domains
- Heterogeneity increases engineering effort [1]
  - Capabilities of new generations limited by team size
  - Biggest challenges are in system-integration



[1] Khailany, DAC '18

Open-Source Hardware

- Gaining momentum in academia, industry, government programs, etc.
- Most contributions focus on the development of SoC components

- Key challenge: *How to realize a complete SoC for a target domain with heterogeneous, OSH components designed by different teams using different tools?*

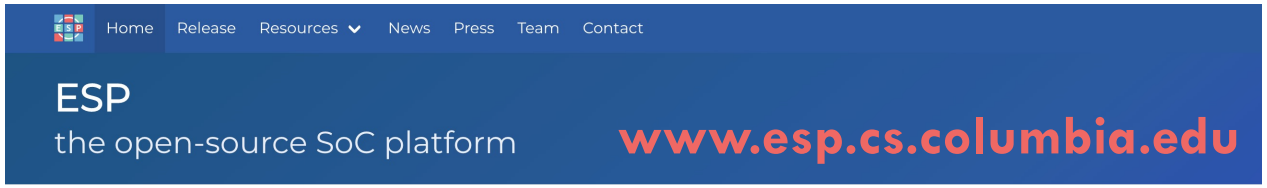
RISC-V: The Free and Open RISC Instruction Set Architecture

The Concept of Platform

- **Innovation in SoC architectures and their design methodologies is needed to promote design reuse and collaboration**
- ***Platform = architecture + methodology***
  - **An SoC architecture that simplifies the integration of many components enables *design reuse***
  - **An SoC methodology that allows designers to choose their preferred languages and tools enables *collaboration***
- **Together, maximizes the potential of open-source hardware**
  - Mitigates engineering challenges and drives innovation [2]

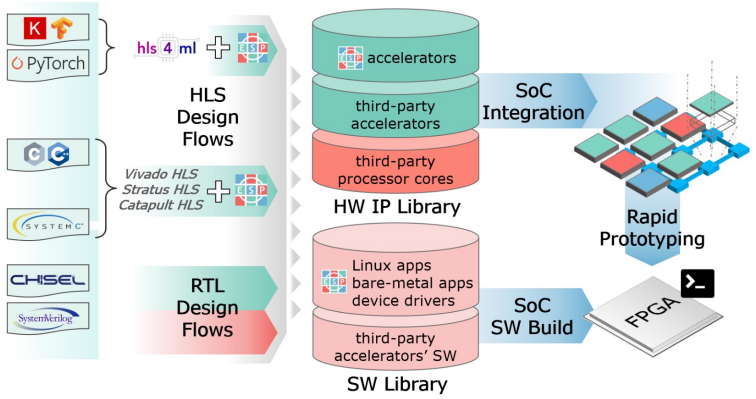
[2] Gupta, IEEE Computer '17

ESP : An Open-Source Platform for SoC Design



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.



Latest Posts

Release 2022.1.0

A new GitHub Release 2022.1.0 of ESP is now available.

Read more

Published: Apr 2, 2022

Collaborators and Users:

IBM

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Fermilab

Pacific Northwest NATIONAL LABORATORY

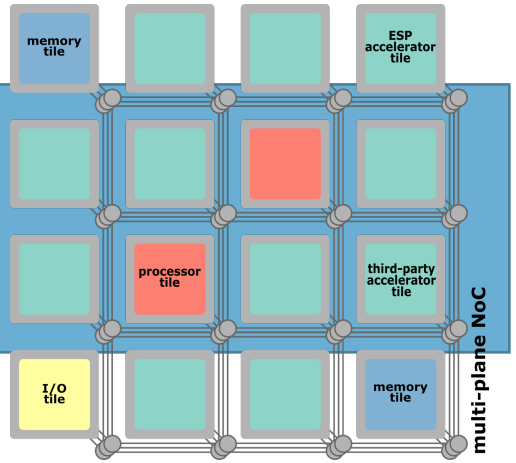
SIEMENS

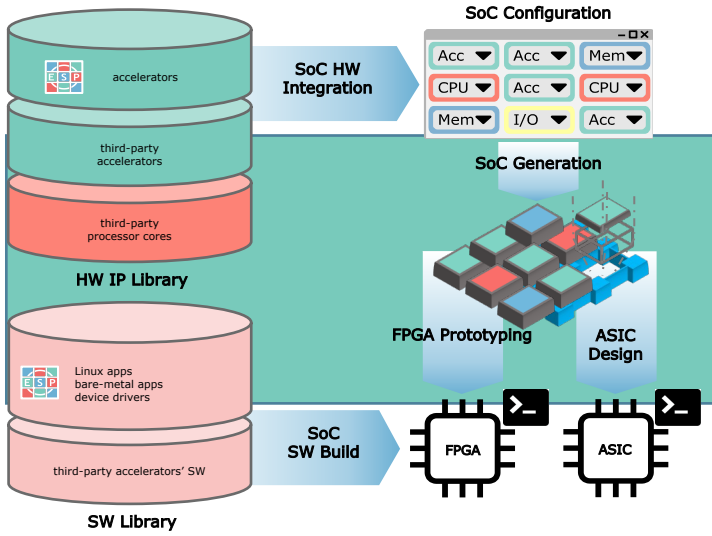
POLITECNICO MILANO 1863

GlobalFoundries™

Outline

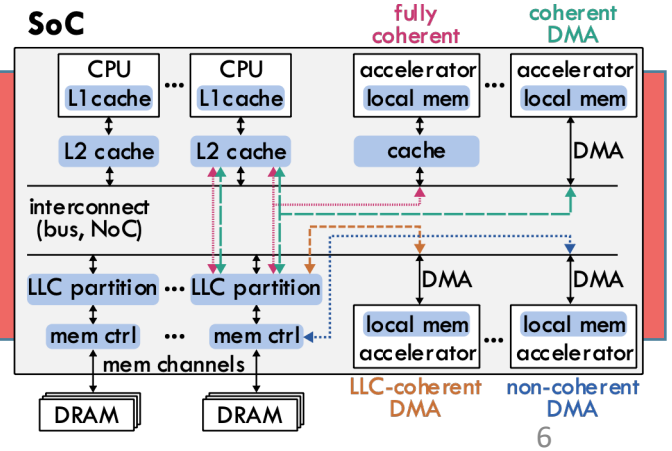
The ESP Architecture





The ESP Methodology

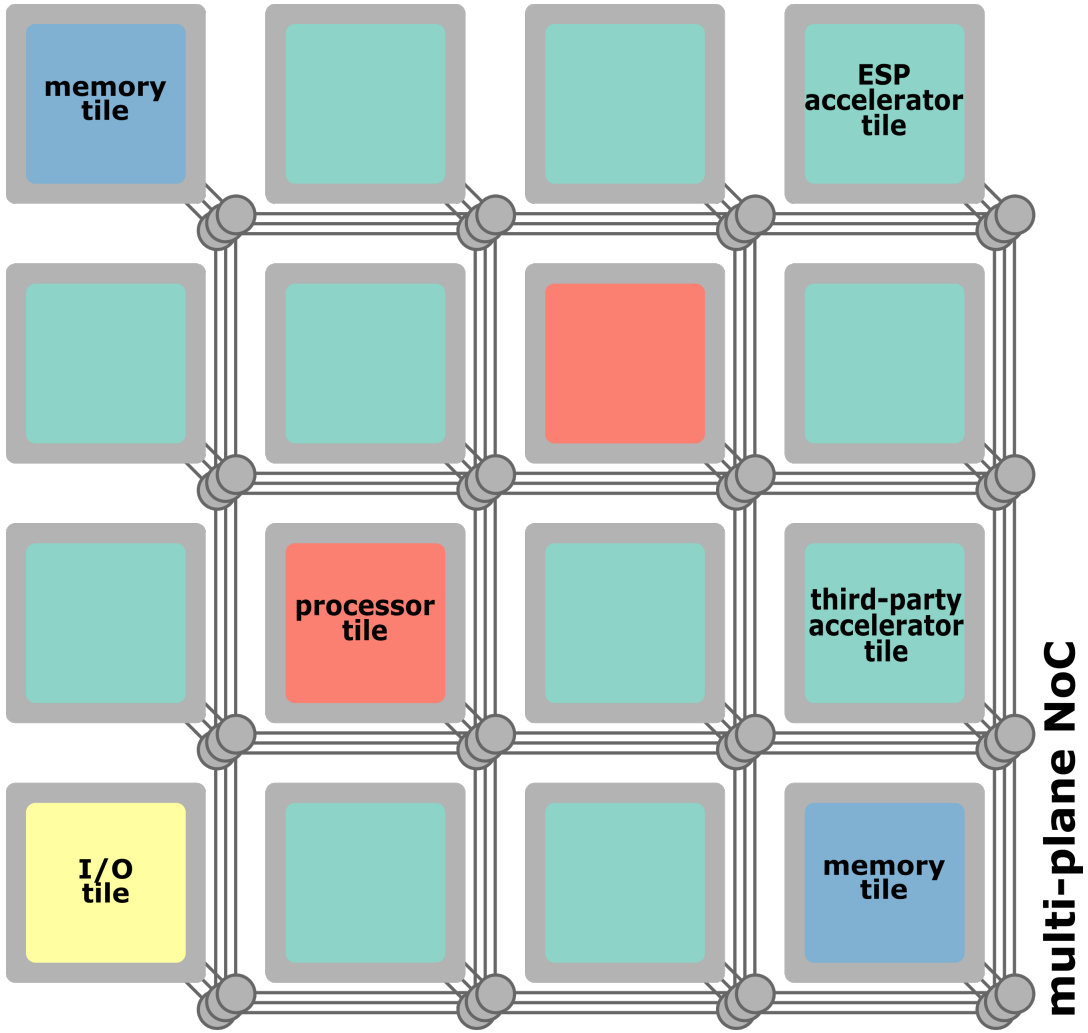
Computer Architecture Research with ESP



ESP Architecture

- Multi-Plane NoC
- Many-Accelerator
- Distributed Memory

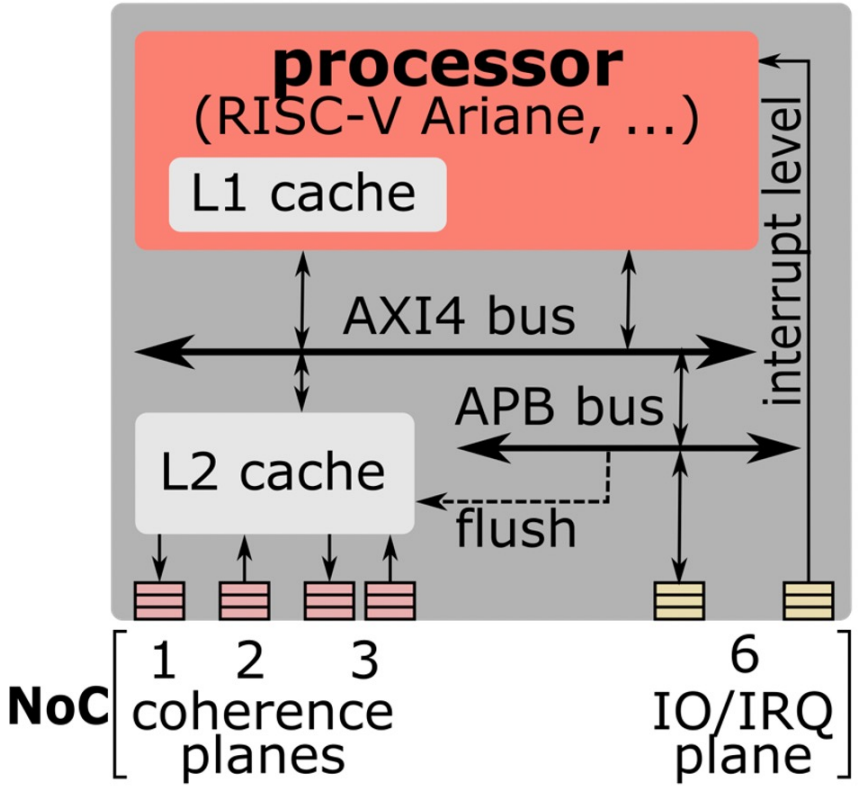
The ESP architecture implements a **distributed** system, which is **scalable**, **modular** and **heterogeneous**, giving processors and accelerators similar weight in the SoC [3]



[3] Mantovani, ICCAD '20

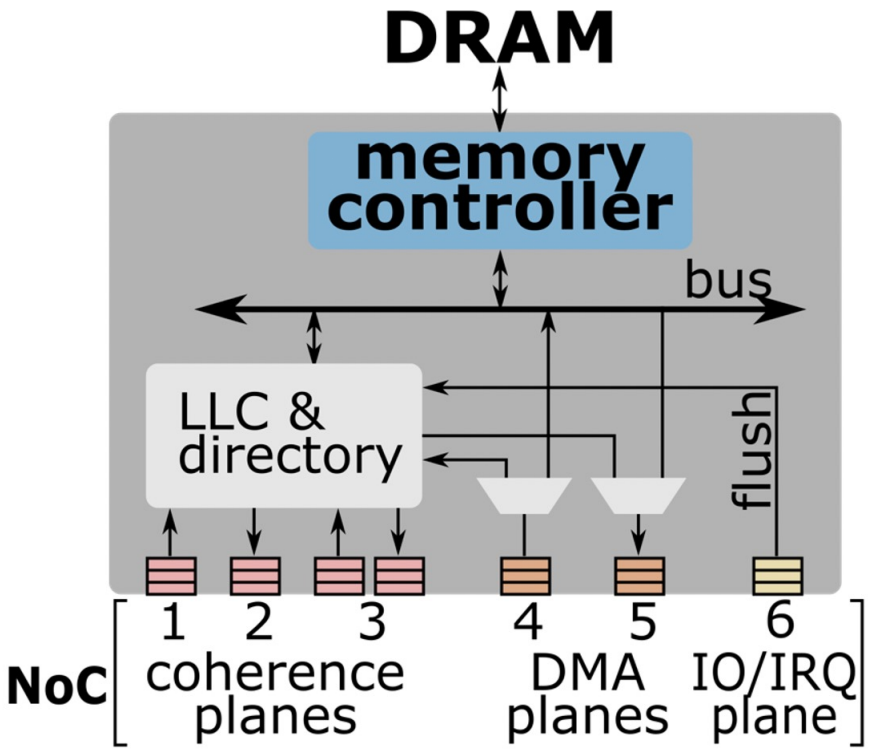
ESP Architecture: Processor Tile

- Processor off-the-shelf
  - **RISC-V CVA6-Ariane (64 bit)**
  - **SPARC V8 Leon3 (32 bit)**
  - **RISC-V IBEX (32 bit)**
  - L1 private cache
- L2 private cache
  - Configurable size
  - MESI protocol
- IO/IRQ channel
  - Un-cached
  - Accelerator config. registers, interrupts, flush, UART, …



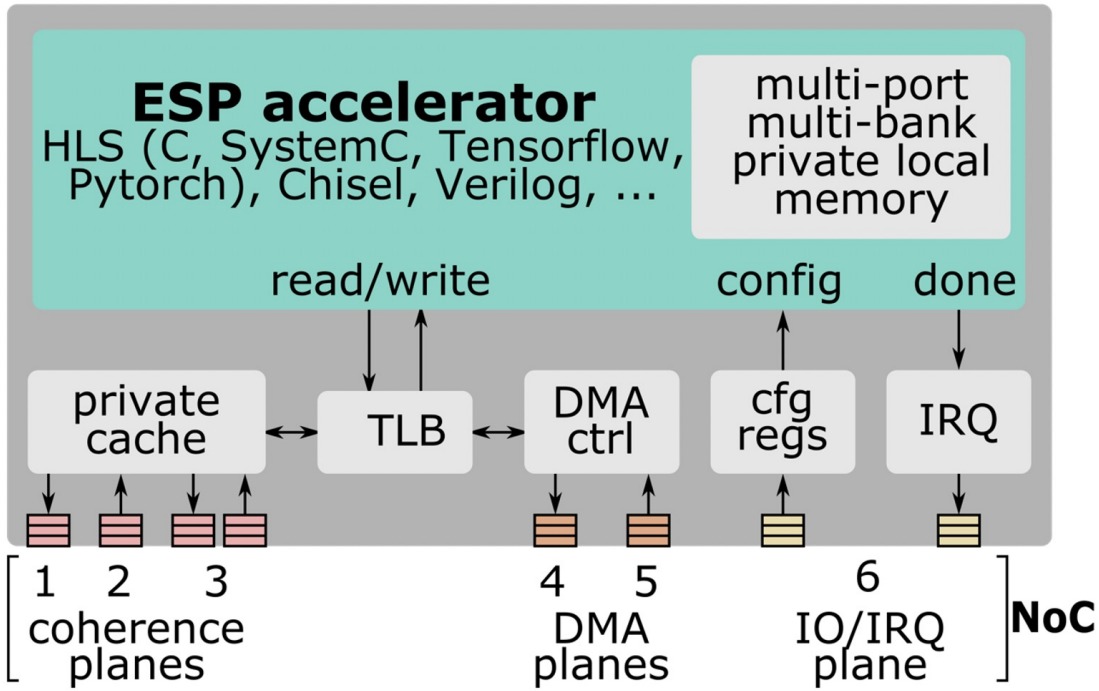
ESP Architecture: Memory Tile

- **External Memory Channel**
- LLC and directory partition
  - Configurable size
  - Extended MESI protocol
  - Supports coherent-DMA for accelerators
- DMA channels
- IO/IRQ channel

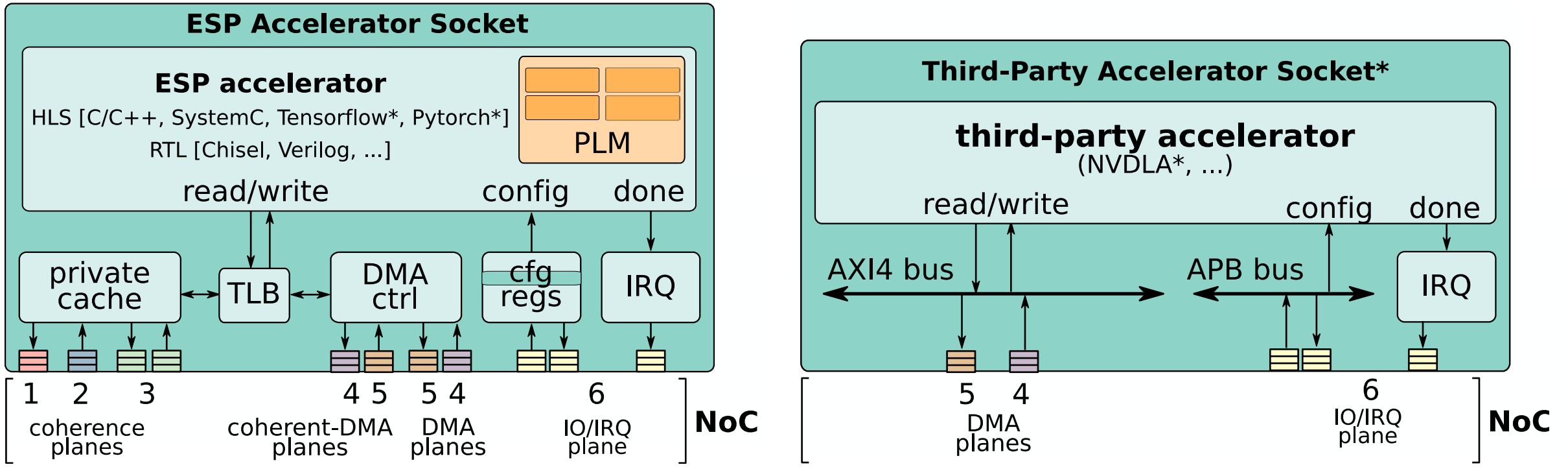


ESP Architecture: Accelerator Tile

- **Accelerator Socket w/ Platform Services**
  - Direct-memory-access
  - Coherence
  - Transparent address translation
  - User-defined registers

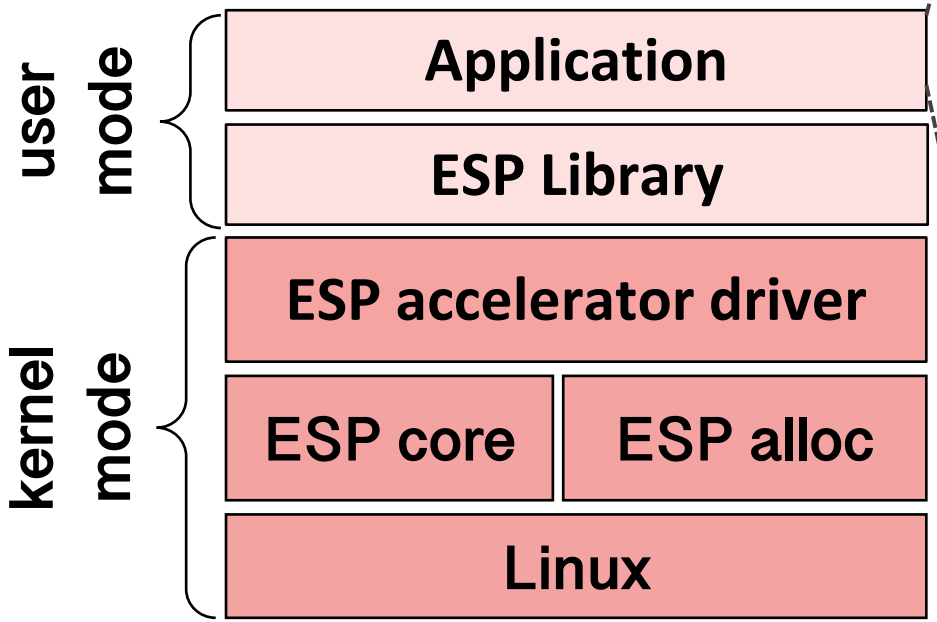


ESP Accelerator Socket



ESP Software Socket

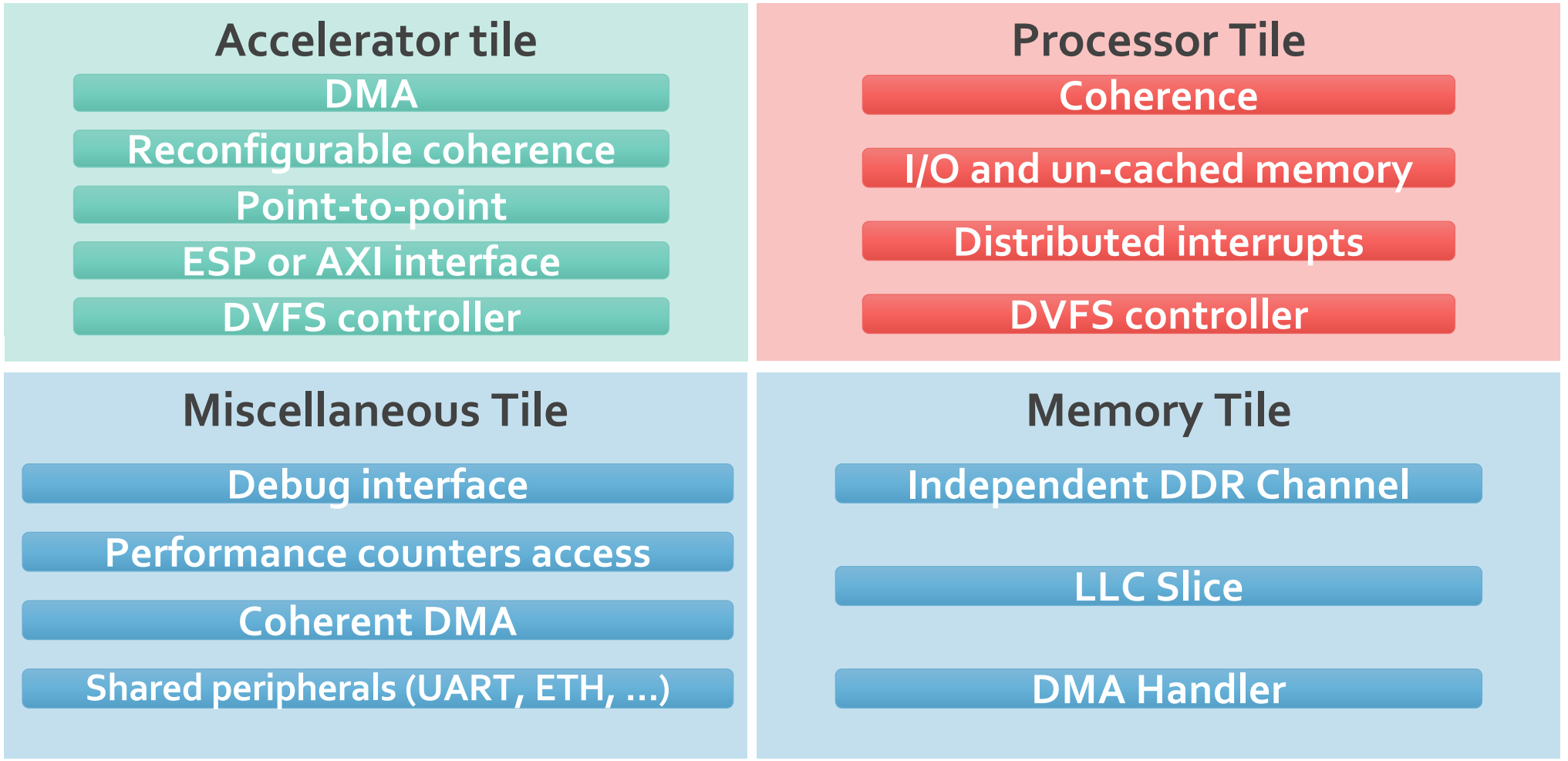
- **ESP accelerator API [4]**
  - Generation of device driver and unit-test application
  - Seamless shared memory




```
/*
 * Example of existing C application with ESP
 * accelerators that replace software kernels 2, 3,
 * and 5. The cfg_k# contains buffer and the
 * accelerator configuration.
 */
{
  int *buffer = esp_alloc(size);

  for (...) {

    kernel_1(buffer,...); /* existing software  */

    esp_run(cfg_k2);      /* run accelerator(s) */
    esp_run(cfg_k3);

    kernel_4(buffer,...); /* existing software  */

    esp_run(cfg_k5);
  }

  validate(buffer);       /* existing checks    */

  esp_free();           /* memory free        */
}
```

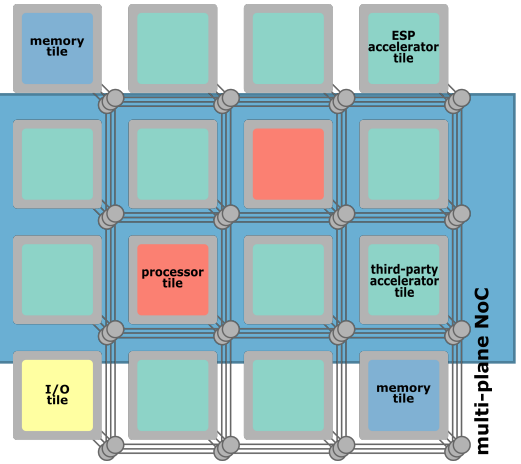
[4] Giri, DATE '20

ESP Platform Services

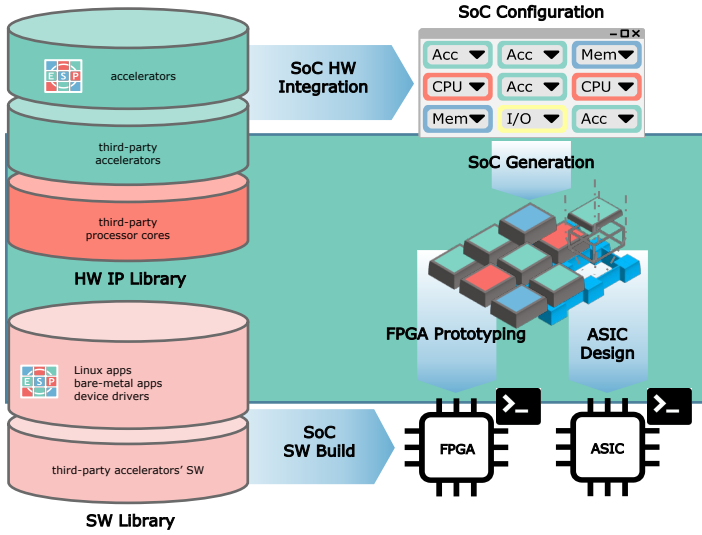


Outline

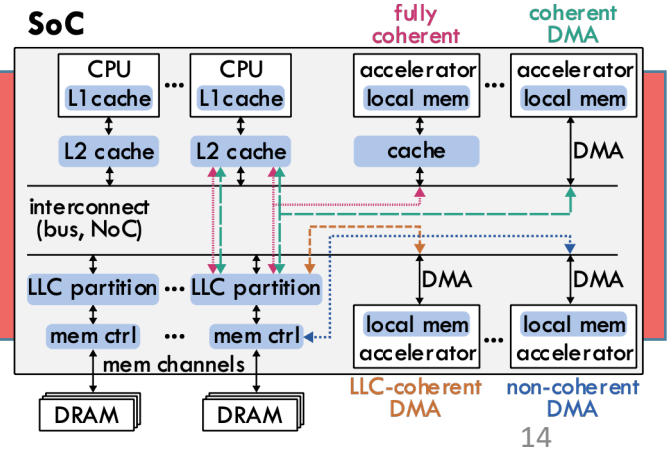
The ESP Architecture



The ESP Methodology

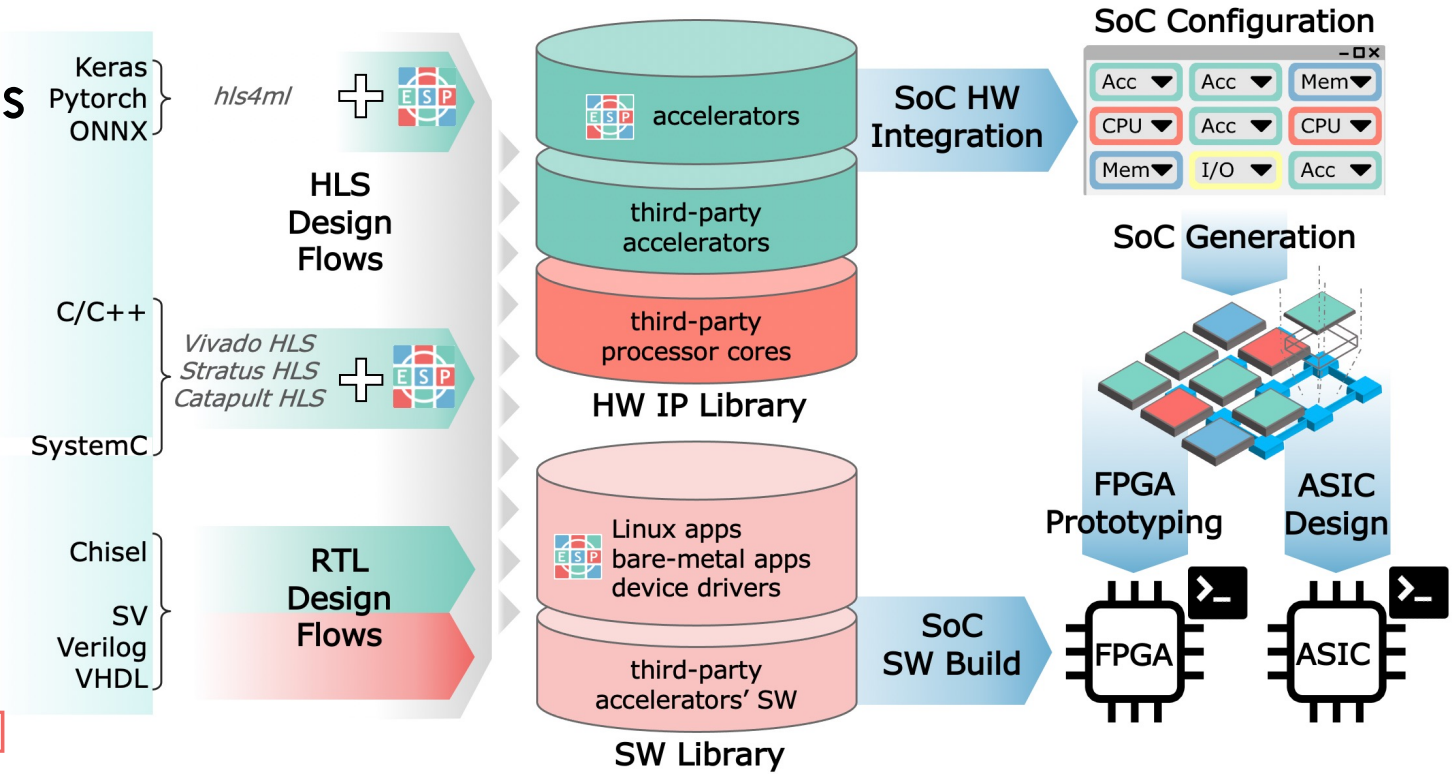


Computer Architecture Research with ESP



The ESP Vision: Domain Experts Can Design SoCs

- Embraces the design of new accelerators from multiple levels of abstraction [4]
- Enables the integration of existing accelerators with the third party flow [5]
- Can be used to produce complex FPGA prototypes [6] or real ASIC implementations [7]



[4] Giri, DATE '20
[5] Giri, IEEE Micro '18
[6] Mantovani, DAC '16
[7] Jia, ESSCIRC '22

ESP Methodology In Practice

automated
interactive
manual (opt.)
manual

Accelerator Flow

Generate accelerator

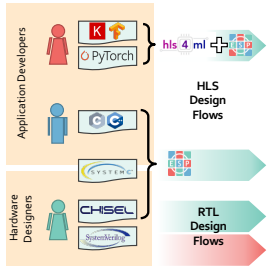
Specialize accelerator
* this step is automated
* for ML applications

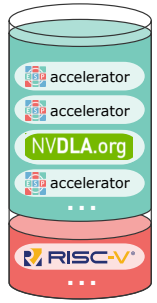
Test behavior

Generate RTL

Test RTL

Optimize RTL





SoC Flow

Generate sockets

Configure RISC-V SoC

Compile bare-metal

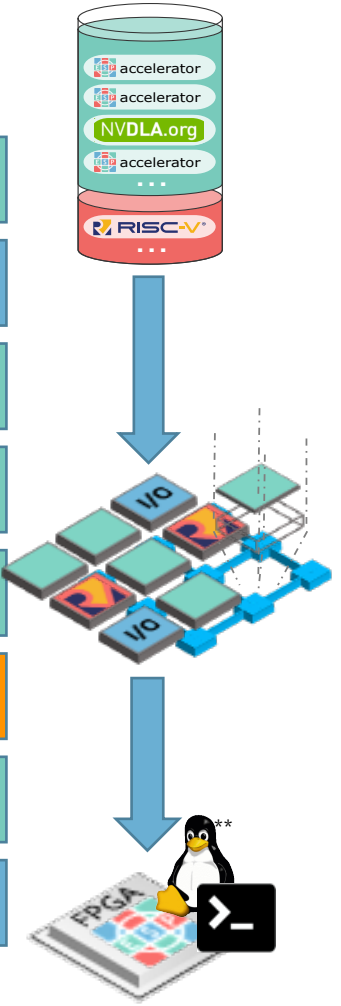
Simulate system

Implement for FGPA

Configure runtime

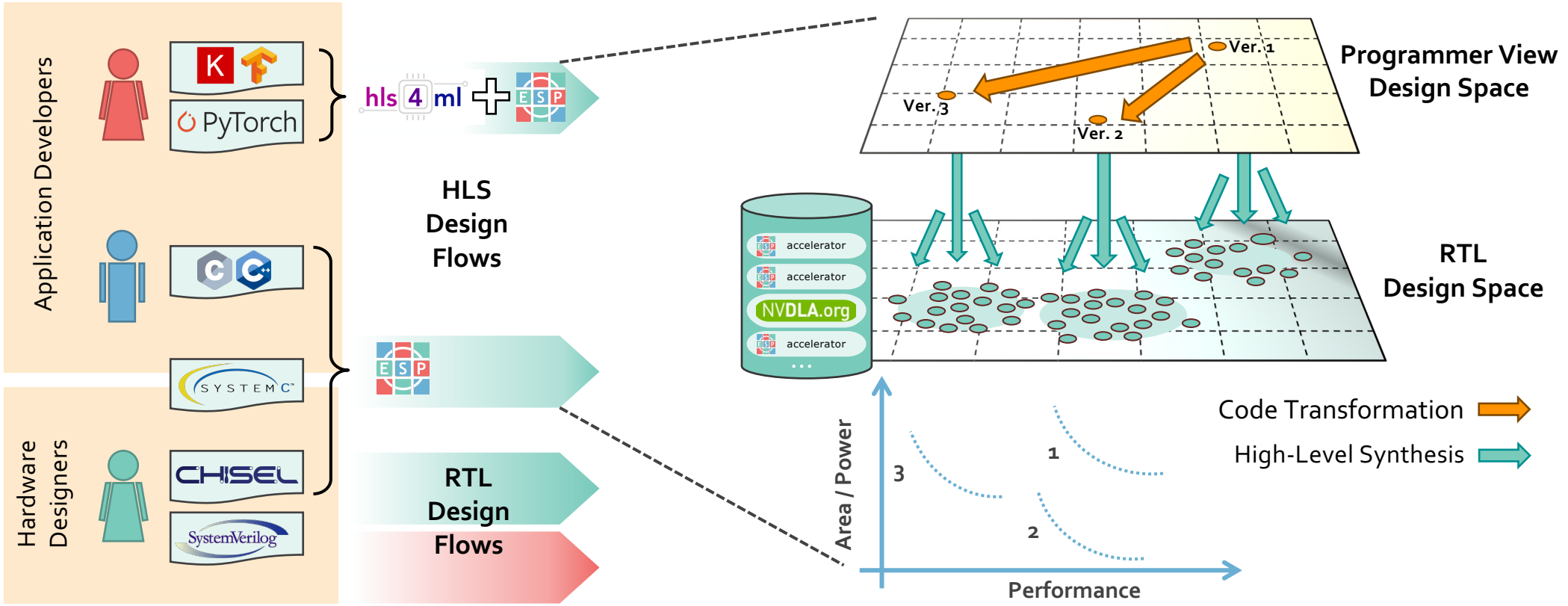
Compile Linux

Deploy prototype



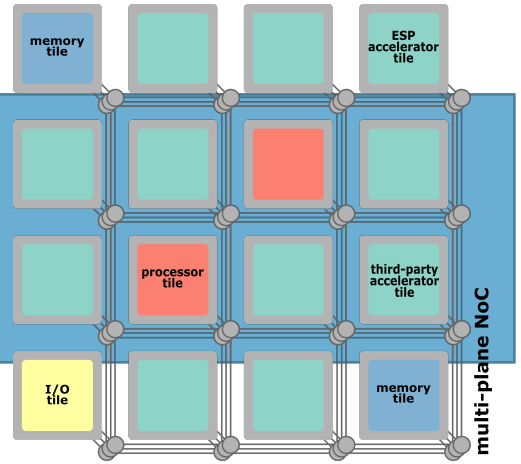
ESP Accelerator Flow

Developers focus on the **high-level specification**, **decoupled** from memory access, system communication, hardware/software interface

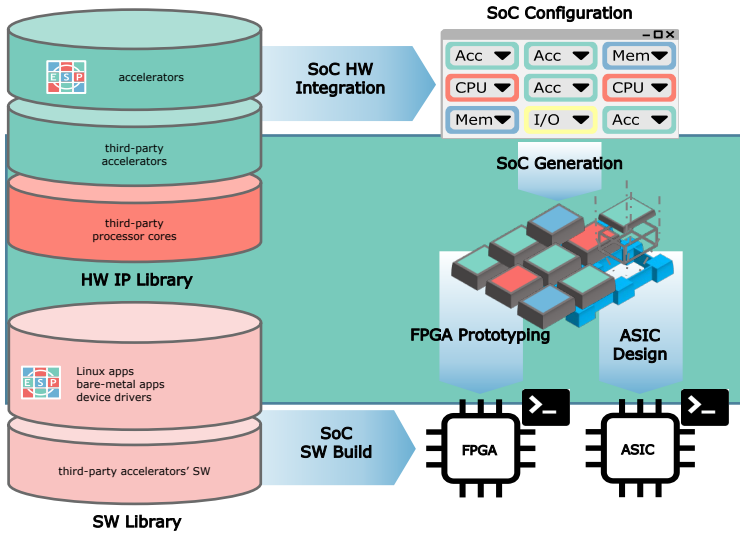


Outline

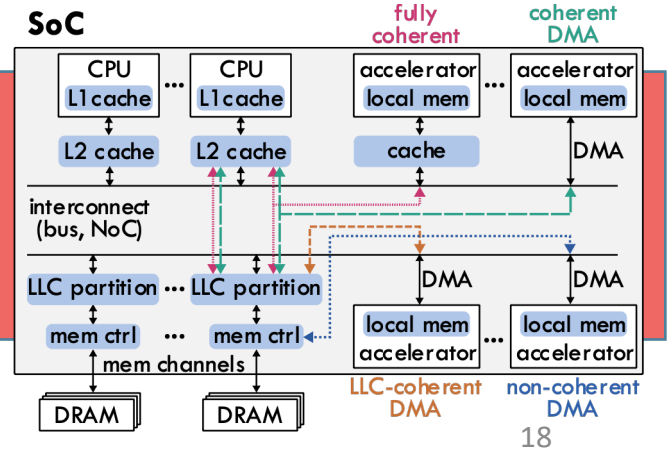
The ESP Architecture



The ESP Methodology

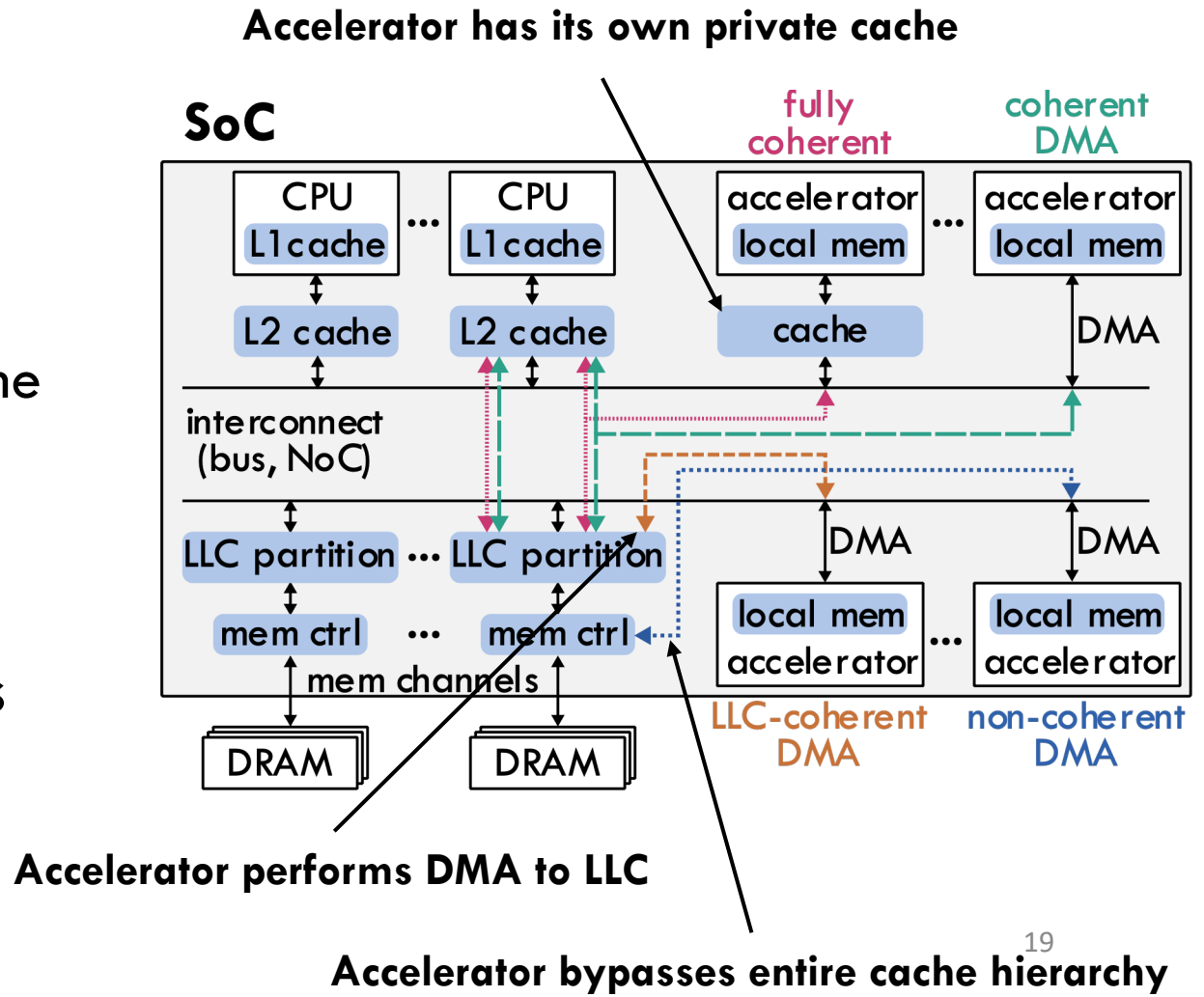


Computer Architecture Research with ESP



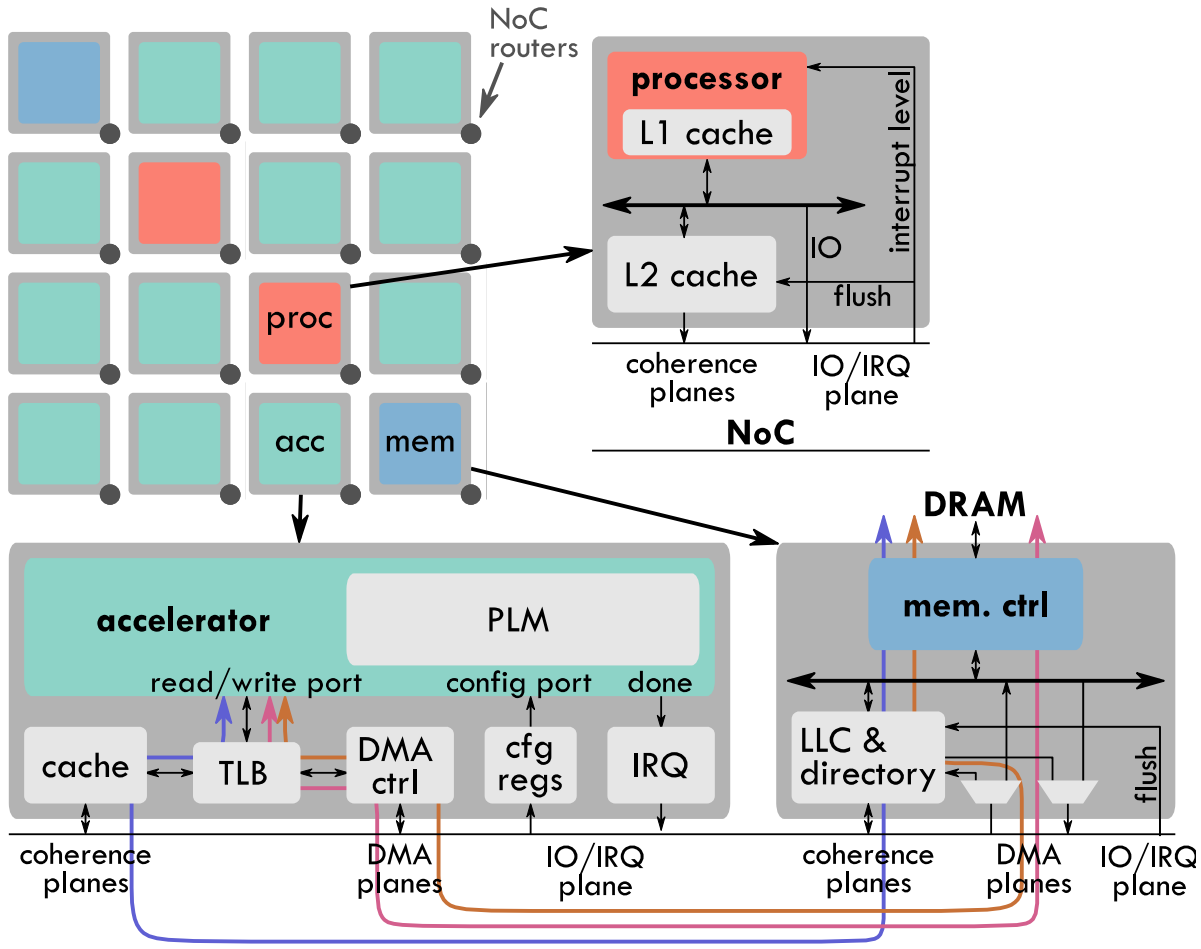
Retrospective: Accelerators and Coherence

- Interactions with the memory hierarchy is a critical aspect of integration of accelerators in SoCs
- ESP: shared-memory model
  - Accelerators and processors address the same address space
  - At what level of the cache hierarchy should the accelerators be integrated?
- Several different coherence modes for accelerators



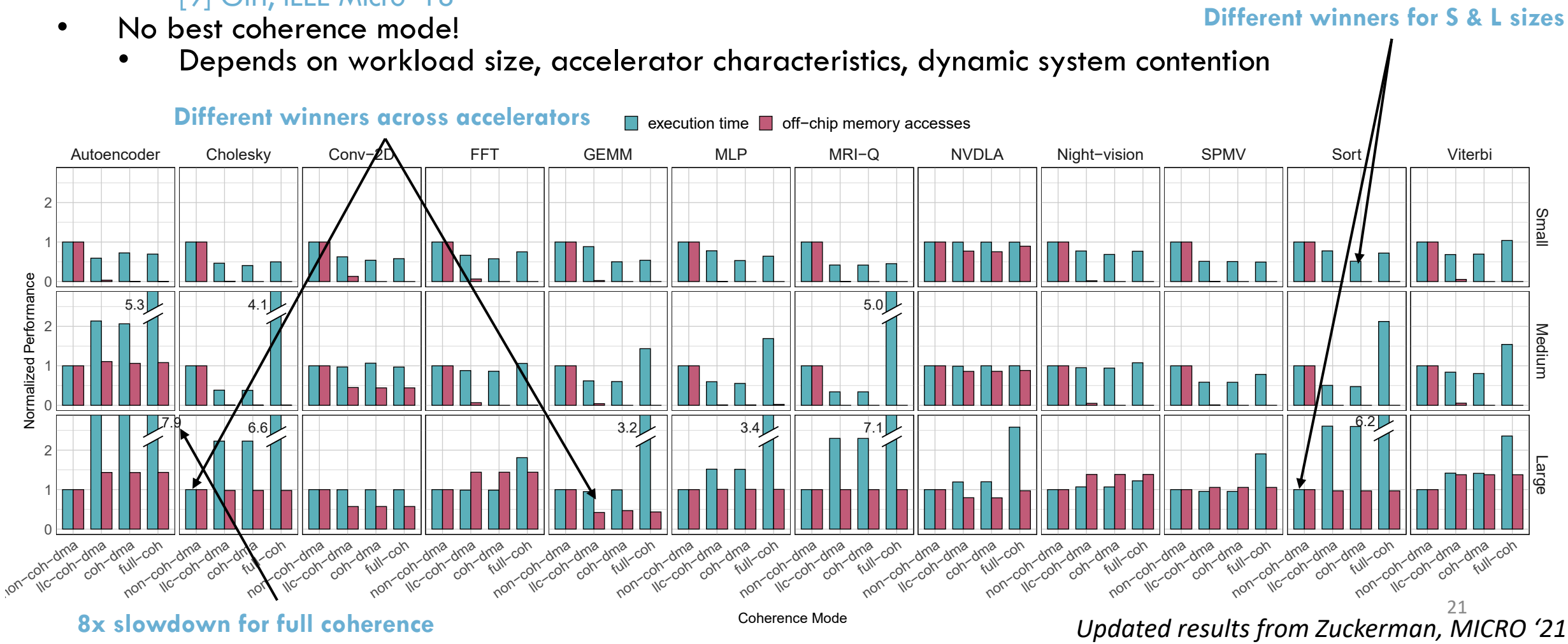
ESP's Coherence Protocol

- ***NoC-Based Support of Heterogeneous Cache-Coherence Models for Accelerators.***
  - [8] Giri, NOCS '18
- Adapted a standard MESI directory protocol to work over a NoC and support 3 accelerator coherence modes
- LLC can handle DMA requests
  - Addition of a Valid state
- Private L2 can be instantiated in processor and accelerator tiles



Performance of Accelerator Coherence Modes

- ***Accelerators and Coherence: an SoC Perspective***
  - [9] Giri, IEEE Micro '18
- No best coherence mode!
  - Depends on workload size, accelerator characteristics, dynamic system contention

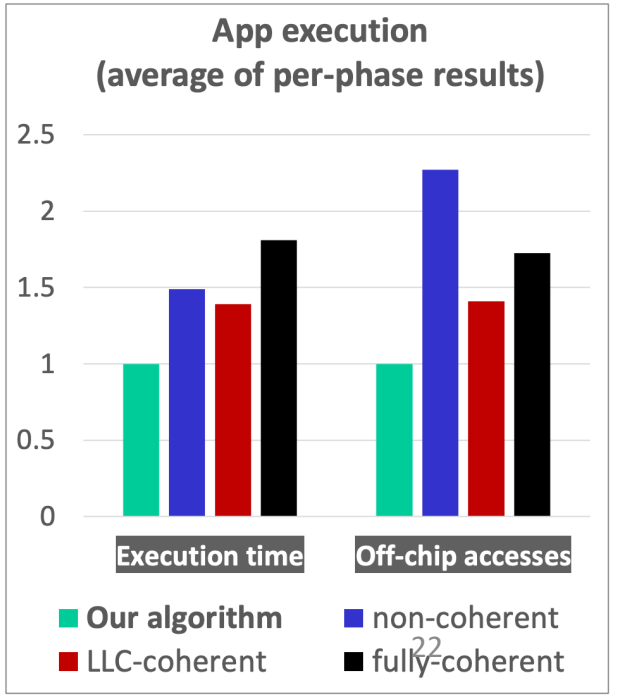


Updated results from Zuckerman, MICRO '21

Reconfigurable Coherence for Accelerators

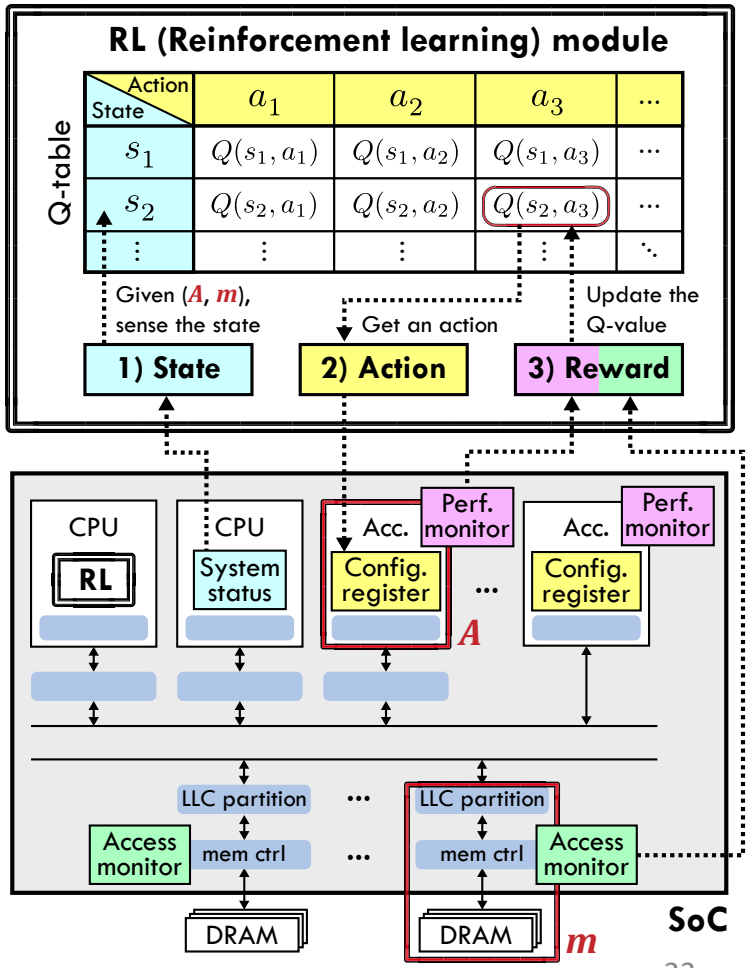
- ***Runtime Reconfigurable Memory Hierarchy in Embedded Scalable Platforms***
  - [10] Giri, ASPDAC '18
- Enabled runtime selection of accelerator's coherence mode
- Hand-tuned algorithm for selection at invocation time
- Evaluation on synthetic application reduces:
  - execution time by 40%
  - off-chip memory accesses by 30%

```
1  if (footprint < PRIVATE_CACHE_SIZE)
2    if (n_fully_coherent < MAX_FULLY_COHERENT)
3      coherence = FULLY_COHERENT;
4    else
5      coherence = LLC_COHERENT;
6
7  else if ((current_llc_footprint + footprint)
8           > LLC_SIZE)
9    coherence = NON_COHERENT;
10
11 else if (n_acc_on_llc_or_fully_coherent
            >= N_MEM_TILES * MAX_ACC_PER_LLC)
12   coherence = NON_COHERENT;
13
14 else
15   coherence = LLC_COHERENT;
```



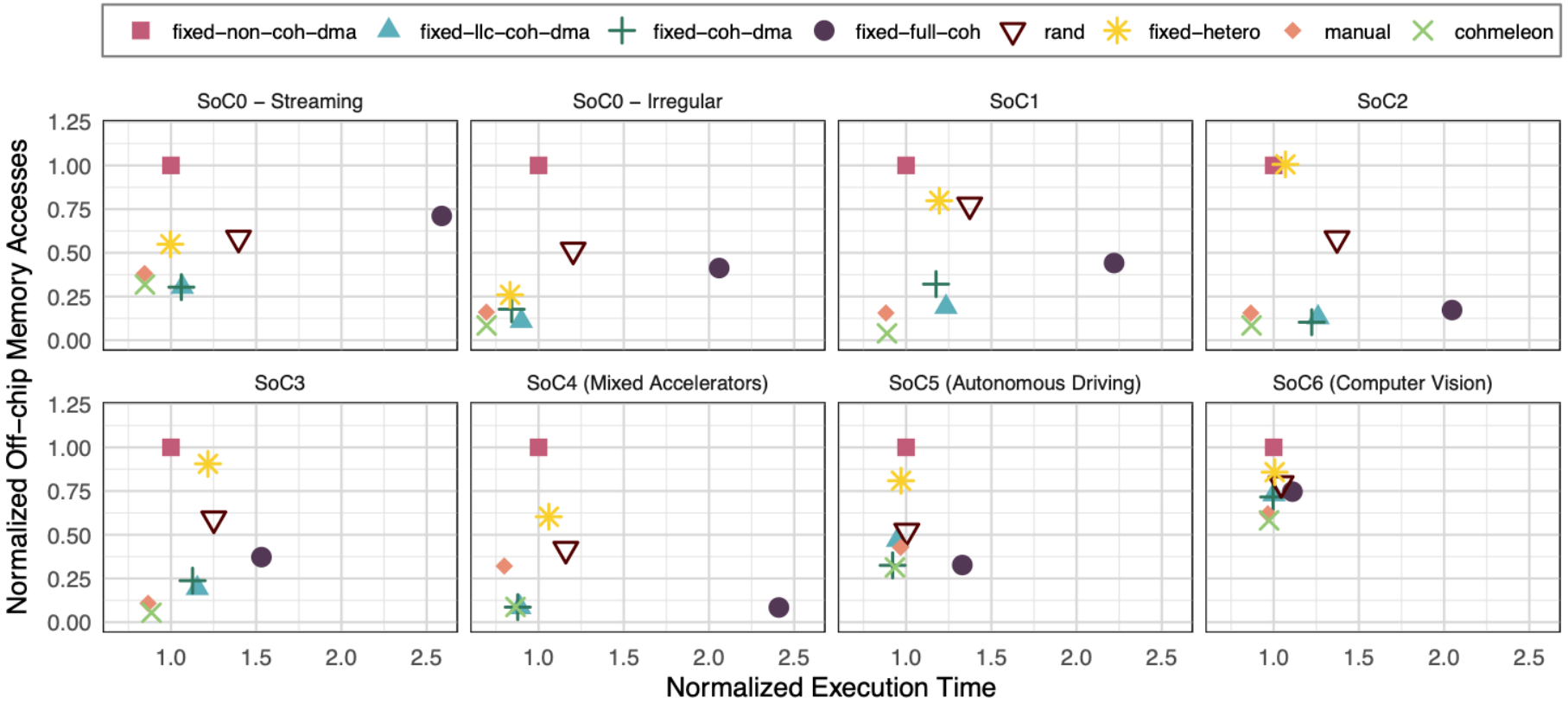
Reinforcement Learning for Coherence Selection

- ***Cohmeleon: Learning-Based Orchestration of Accelerator Coherence in Heterogeneous SoCs***
  - [11] Zuckerman, MICRO '21
- Hand-designed algorithm depends on many factors and requires tuning for target architecture
- Reinforcement-learning solution trains online during normal SoC operation
- Continuously updates itself by observing system status and measuring performance
  - New ESP performance monitoring system and API



Evaluating Cohmeleon

- 7 different many-accelerator SoCs on FPGA
  - Use real accelerators to target particular domains
- Avg speedup of 38% with a 66% reduction of off-chip memory accesses when compared to design-time solutions.



In Summary: ESP for Computer Architecture Research

- ESP enables studying contributions in the context of complete systems
  - Modular architecture eases integration of existing IP
  - Flexible methodology for developing new components
  - Growing library of accelerators from various domains
    - neural network inference, collaborative autonomous driving, computer vision, signal processing, brain-computer interfaces, cryptography
  - Rapid prototyping on FPGA
  - Hardware monitoring system and accompanying API for performance evaluation
- We invite you to use ESP for your projects and to contribute to ESP!

Thank you from the ESP team!

sld.cs.columbia.edu


ColumbiaSld

esp.cs.columbia.edu


sld-columbia/esp


c/ESP-platform

System-Level Computer Architecture Research with Open ESP

J. Zuckerman, D. Giri, P. Mantovani, M. Cassel Dos Santos, K. Chiu, G. Di Guglielmo,
G. Eichler, J. Kwon, L. Piccolboni, B. Seoyum, G. Tombesi, L.P. Carloni

Image Sources:
https://chipsalliance.org/
https://github.com/nvdla
https://www.openhwgroup.org/
https://parallel.princeton.edu/openpiton/
https://pulp-platform.org/
https://riscv.org/
https://www.ibm.com/
https://www.seas.harvard.edu/
https://www.seas.harvard.edu/
https://illinois.edu/
https://www.princeton.edu/
https://www.fnal.gov/
https://www.pnnl.gov/
https://www.polimi.it/
https://www.siemens.com/
https://gf.com/
https://www.gatech.edu/

OSCAR 2022