# Enabling Heterogeneous, Multicore SoC Research with RISC-V and ESP

**Joseph Zuckerman**

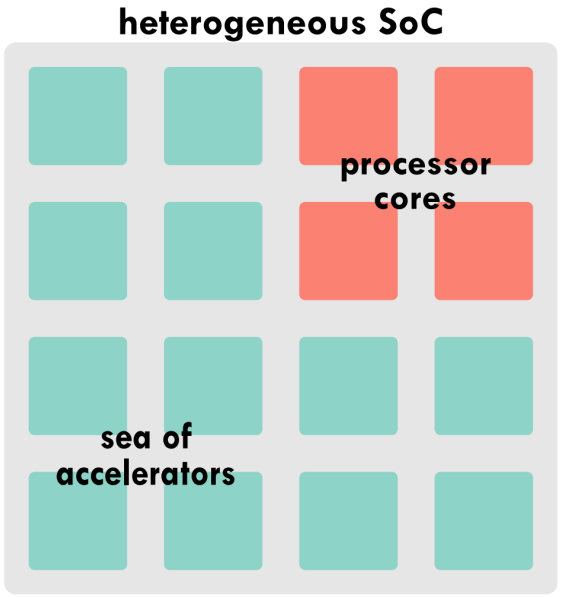
Paolo Mantovani, Davide Giri, Luca P. Carloni


COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

CS@CU COMPUTER SCIENCE

# Motivation – Heterogeneity in SoC Design

- SoCs are increasingly heterogeneous
- Heterogeneity increases engineering effort [1]
- OSH mitigates this by promoting collaboration and design reuse [2]
  - Lots of open-source RISC-V cores



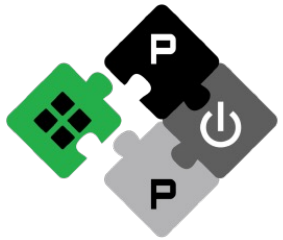


OPENHW GROUP
PROVEN PROCESSOR IP


Berkeley Architecture Research


SiFive


T-HEAD

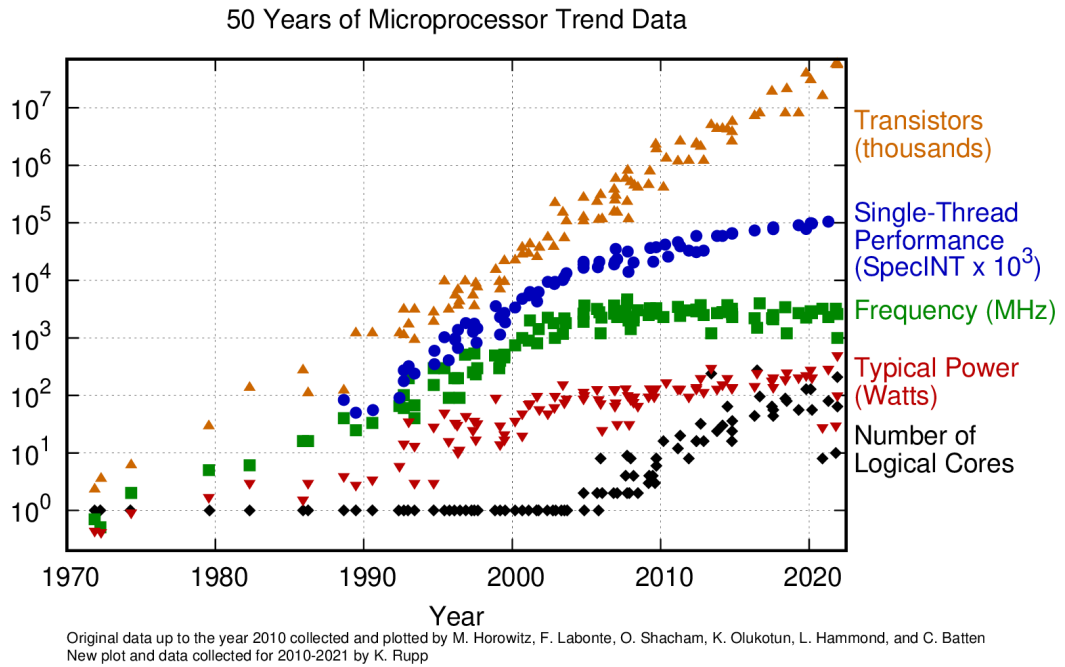


CAES
PIONEERING ADVANCED ELECTRONICS

[1] Khailany, DAC '18
[2] Gupta, IEEE Computer '17

# Motivation – Multicore Architectures

- Diminishing returns of parallel architectures are a driver towards heterogeneity
- Yet, multicore is (clearly) still an important feature of modern systems
- Supporting multiple cores imposes additional challenges for SoC design
  - Coherence
  - Synchronization




**→ Need a platform that enables the seamless design of multicore SoC architectures with heterogeneous IP blocks**

# Contributions

- We augment ESP to support SoCs with up to 4 CVA6 RISC-V cores
  - Capable of booting Linux SMP and running multithreaded applications on FPGA
- Our modifications rely on standardized interfaces
  - Simplifies the integration of new cores

- ESP distinguishes itself from other open-source multicore-capable RISC-V SoC platforms by:
  - adopting a system-centric rather than a processor-centric mindset
  - relying on standardized interfaces and bus protocols
  - utilizing an architecture that scales well to large SoC designs



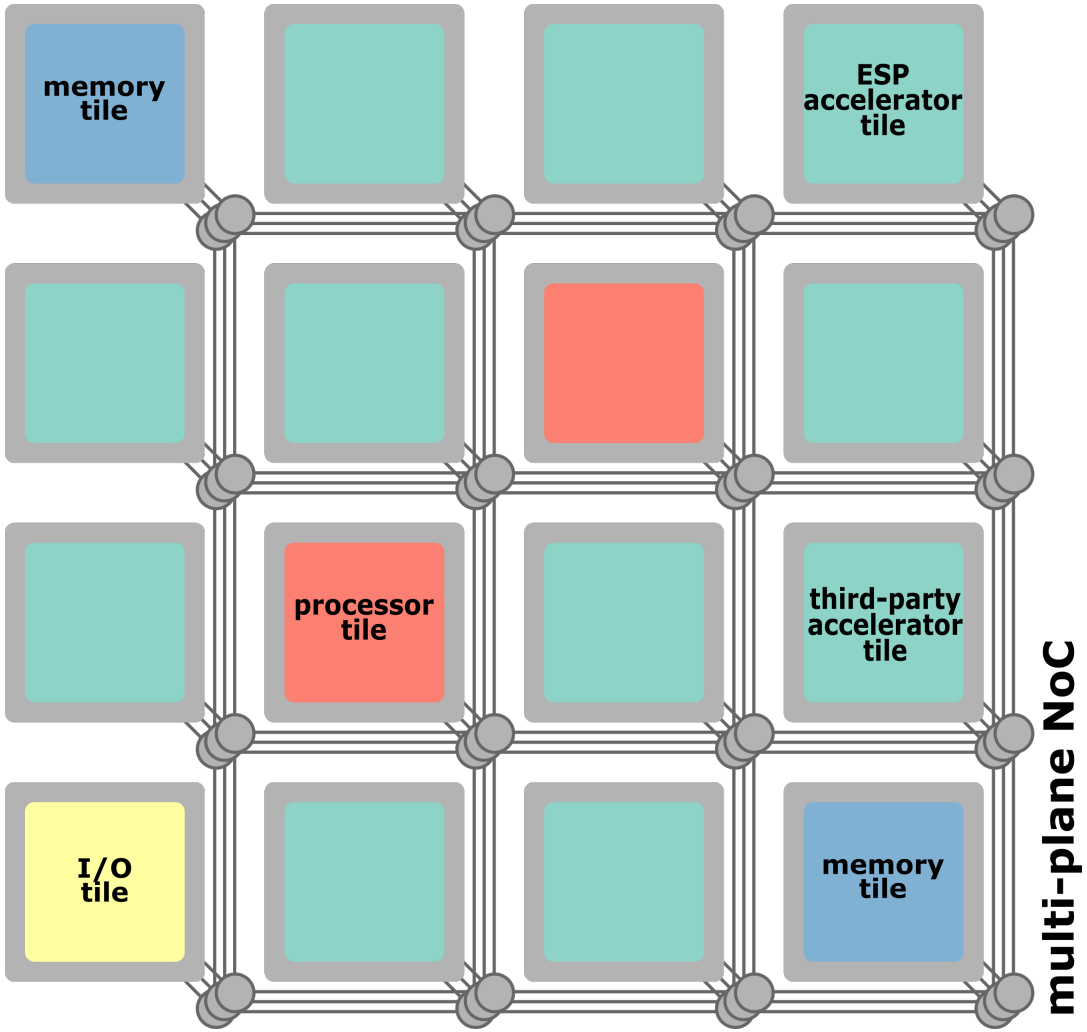
BlackParrot

# ESP Overview

# ESP Architecture

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

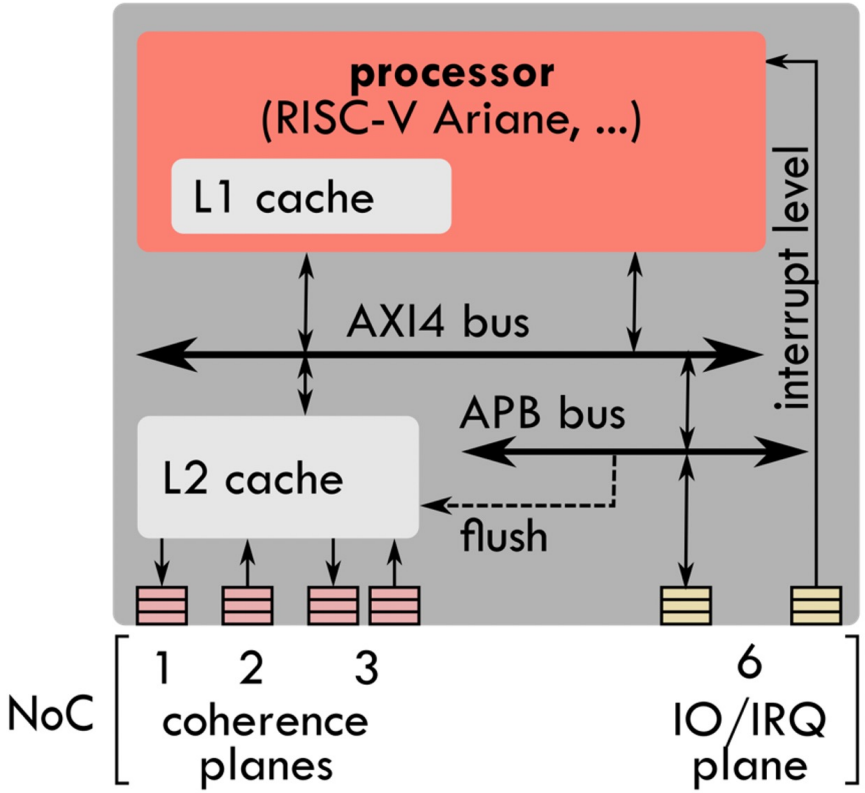
The ESP architecture implements a **distributed** system, which is **scalable**, **modular** and **heterogeneous**, giving processors and accelerators similar weight in the SoC [3]




[3] Mantovani, ICCAD '20

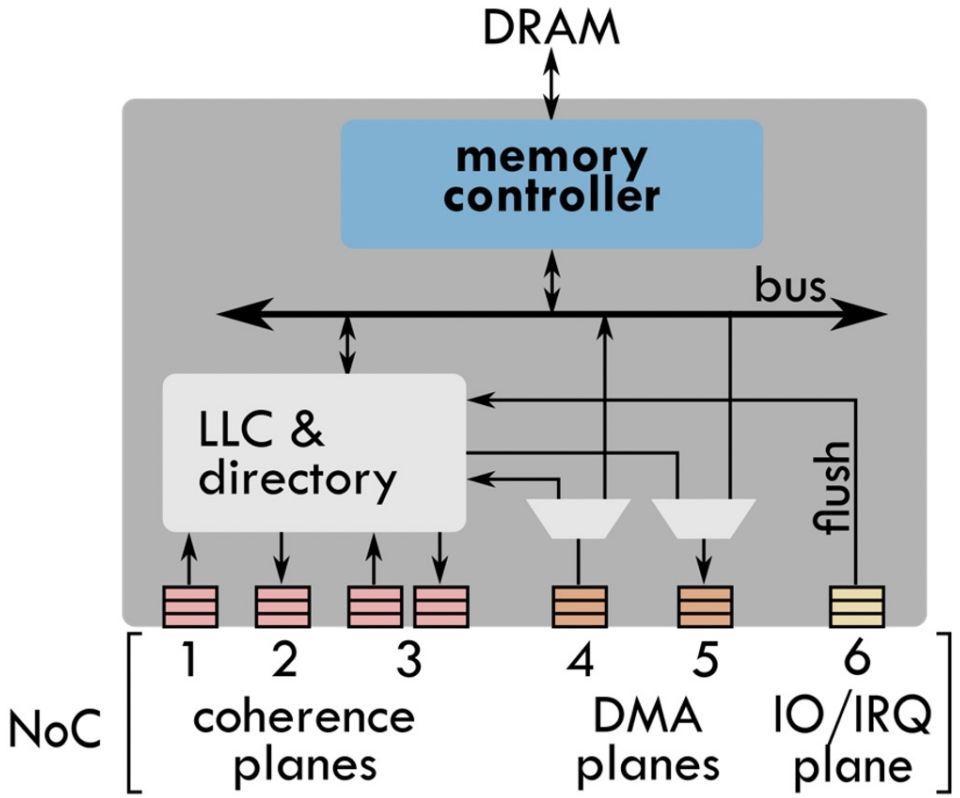
# ESP Architecture: Processor Tile

- 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, …



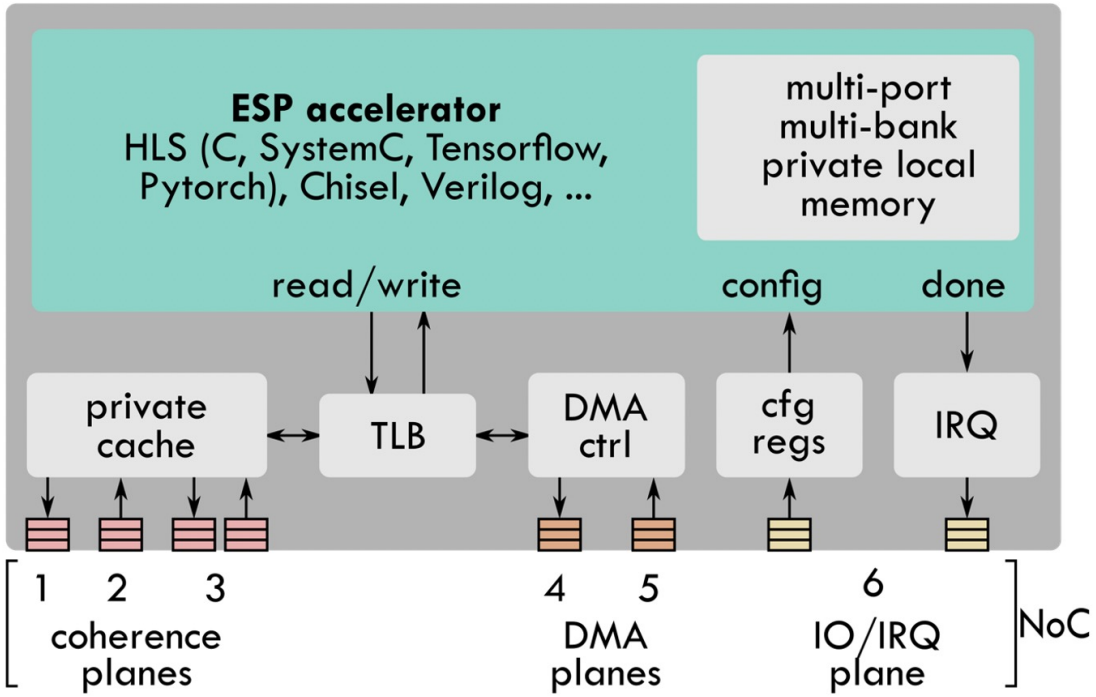
# ESP Architecture: Memory Tile

- **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

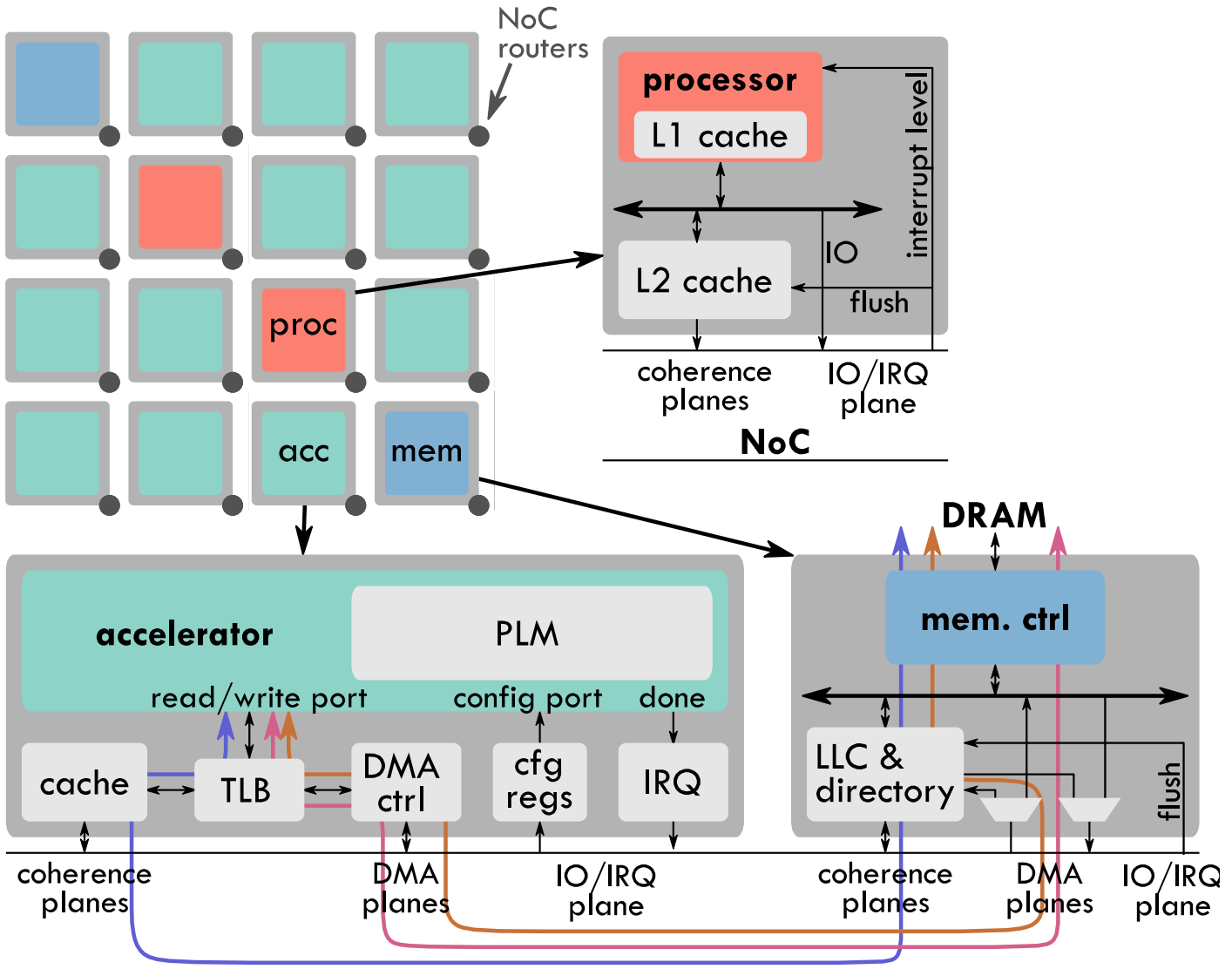
- **Accelerator Socket w/ Platform Services**
  - Direct-memory-access
  - Run-time selection of coherence model [5, 6] :
    - Fully coherent
    - LLC coherent
    - Non coherent
  - User-defined registers
  - Distributed interrupt




[4] Giri, ASPDAC '19 [5] Zuckerman, MICRO '21

# ESP Architecture: Coherence Protocol

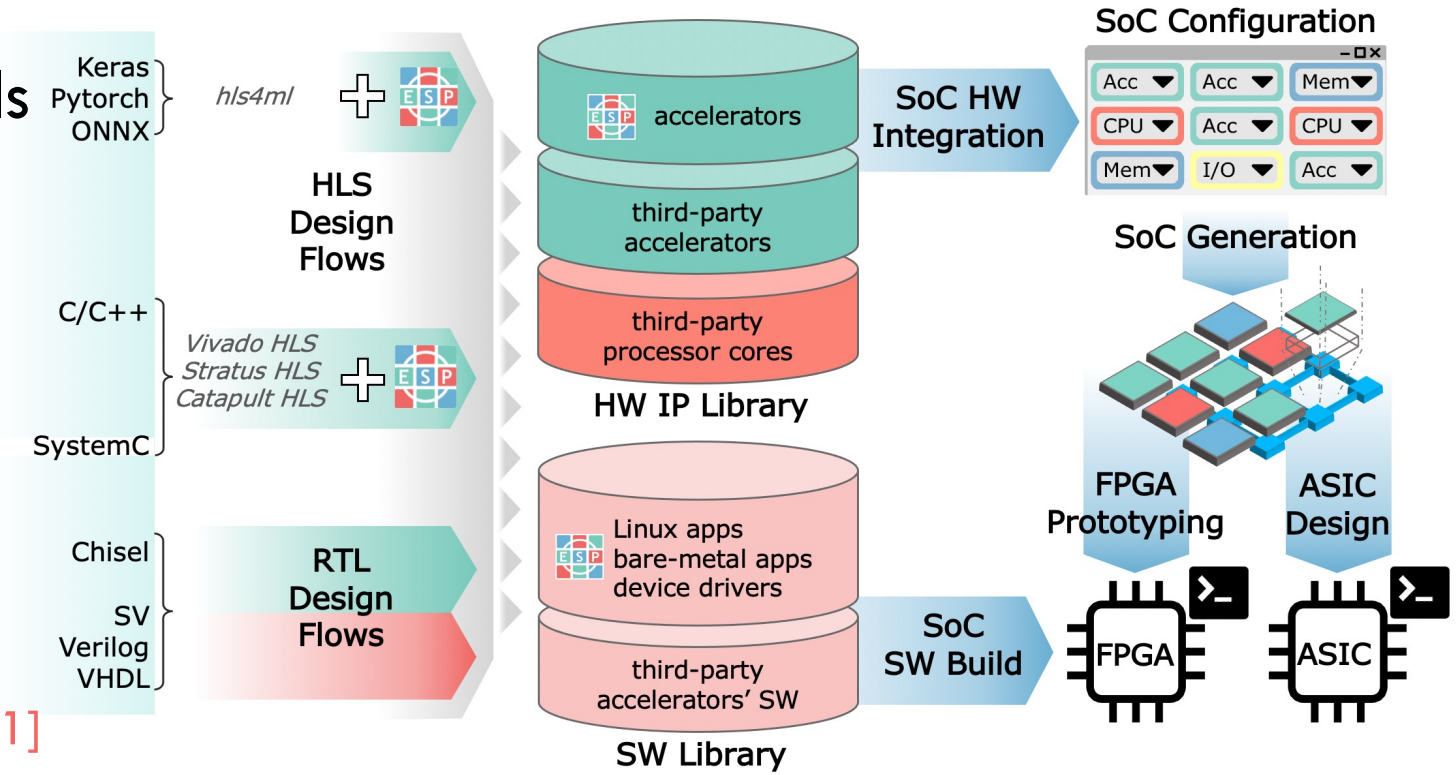
- Directory-based MESI protocol adapted to work over NoC [6]
  - Adds a Valid state
- Handles LLC-coherent DMA from accelerators [6, 7]
- Private L2 cache can be instantiated in acc or proc tiles
  - Handles atomic operations
- Designed to support SPARC LEON3 core




[6] Giri, NOCS '18 [7] Giri, IEEE MICRO '18

# ESP Methodology

- Embraces the design of new accelerators from multiple levels of abstraction [8]
- Enables the integration of existing accelerators with the third party flow [9]
- Can be used to produce complex FPGA prototypes [10] or real ASIC implementations [11]



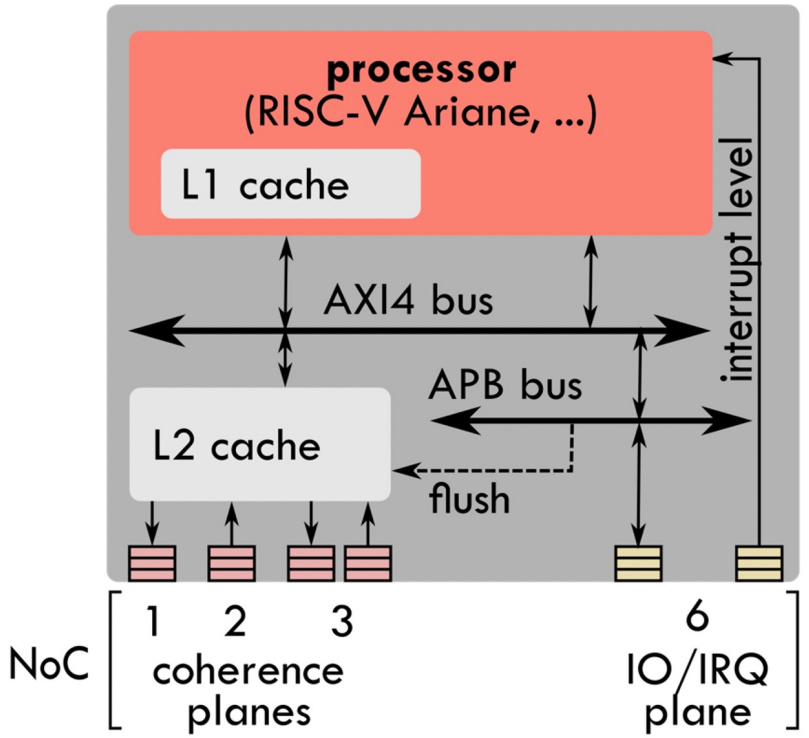


[8] Giri, DATE '20
[9] Giri, IEEE MICRO '21
[10] Mantovani, DAC '16
[11] Jia, ESSCIRC '22

# Enabling Multicore RISC-V

# CVA6 Integration

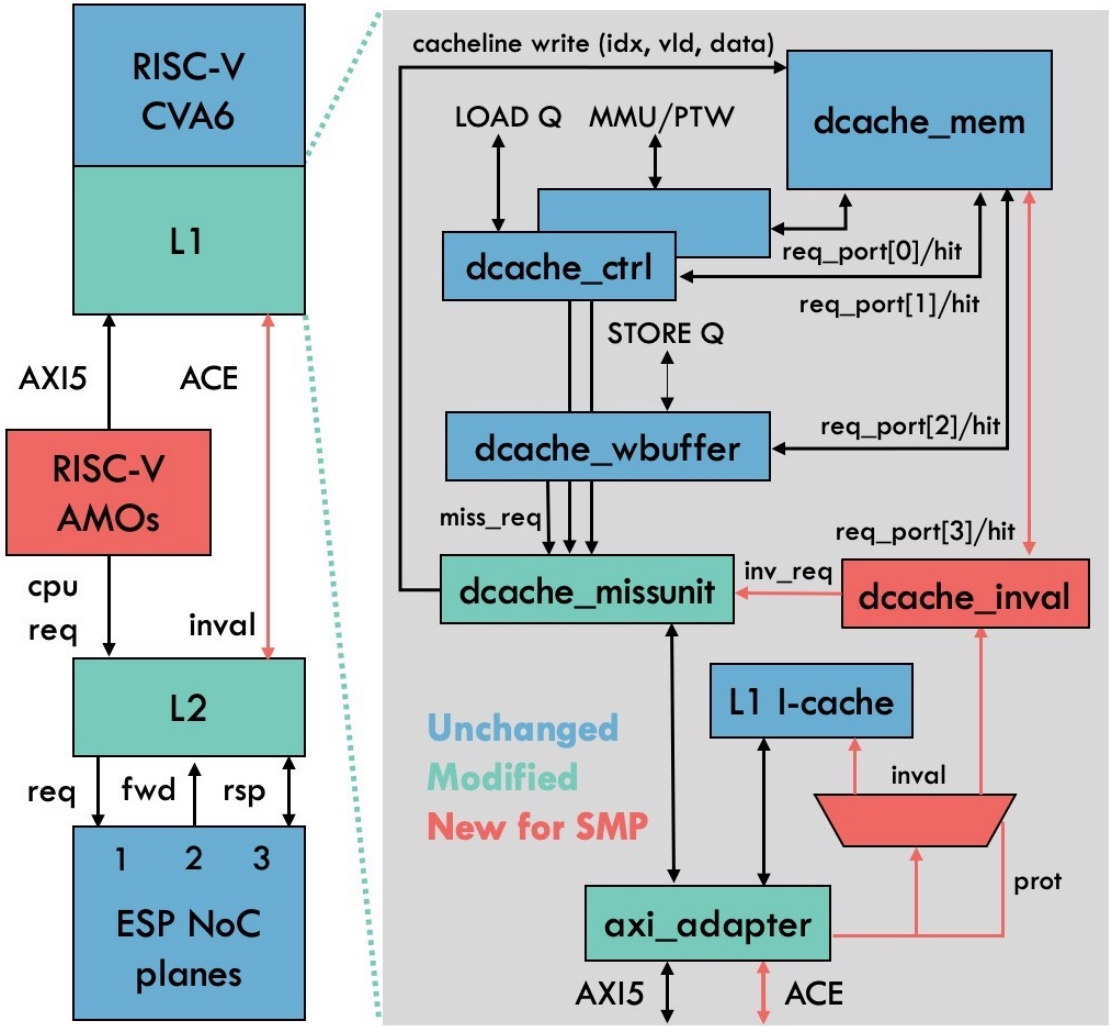
- Prior work integrated the CVA6 (Ariane) [12,13] core with ESP [14]
- AHB Bus → AXI crossbar in processor tile
- New AXI wrapper for L2 cache
- Handle little-endian writes in the L2
  - LLC unchanged
- Key challenges to enable multicore:
  - L1 Invalidation
  - RISC-V atomic instructions




[12] https://github.com/openhwgroup/cva6
[13] Zaruba, IEEE TVLSI '19
[14] Giri, CARRV '20

# L1 Invalidation of CVA6

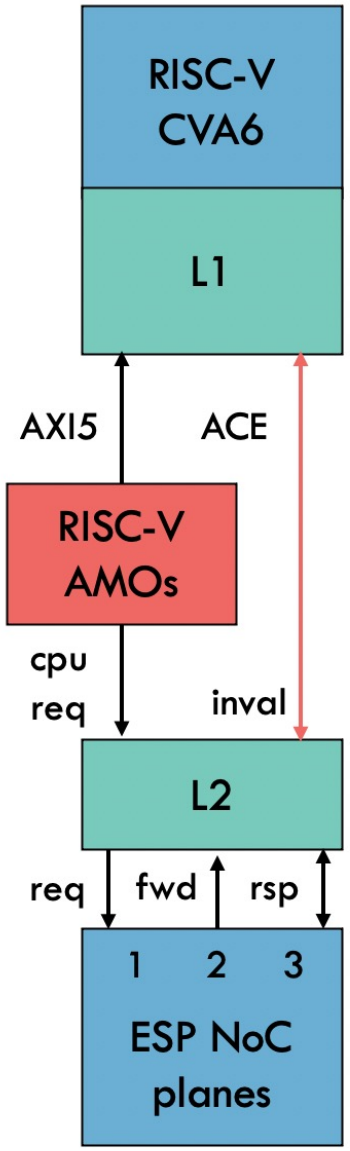
- CVA6 does not natively accept invalidation on its AXI interface
  - Prior work integrated CVA6 in the OpenPiton architecture, but relied on a custom interface [15]
- We leverage the AXI Coherency Extensions to send invalidation
- L2 drives the Snoop Address (AC) channel with a `MakeInvalid` command
  - Also sends the protection bits to route between the I-cache and D-cache
- New `dcache_inval` unit performs lookup in cache memory and invalidates on a hit
- L1-Flush signal exposed for accelerator invocations




[15] Balkind, CARRV '19

# RISC-V AMOs

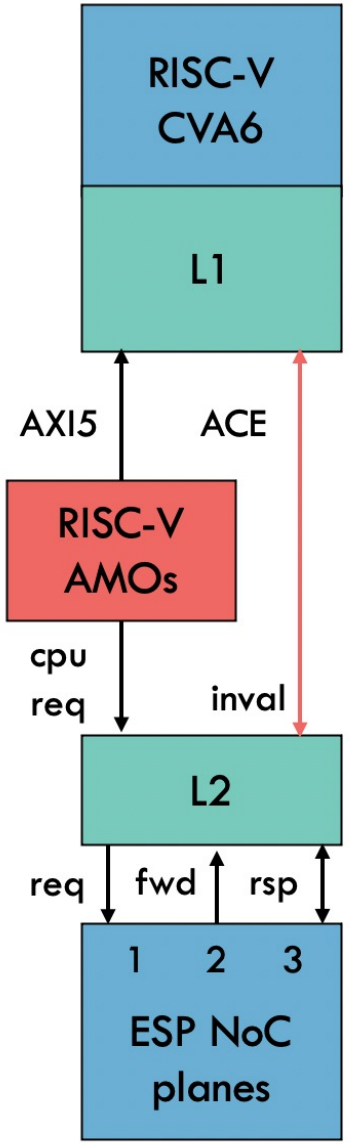
- Handle RISC-V atomic memory operations by instantiating AXI Adapter for RISC-V atomics from PULP [16]
  - Converts AMO request from one write transaction to a downstream read and write
  - Small ALU performs computation
  - `lock` field on AXI bus signals to L2 that the read and write are part of an AMO
- Atomicity enforced by L2, leveraging prior implementation for handling SPARC atomics
  - Forward requests for a cache line with an ongoing atomic are stalled
  - Minor changes to determine when an atomic is "over"




[16] https://github.com/pulp-platform/axi_riscv_atomics

# RISC-V LR/SC

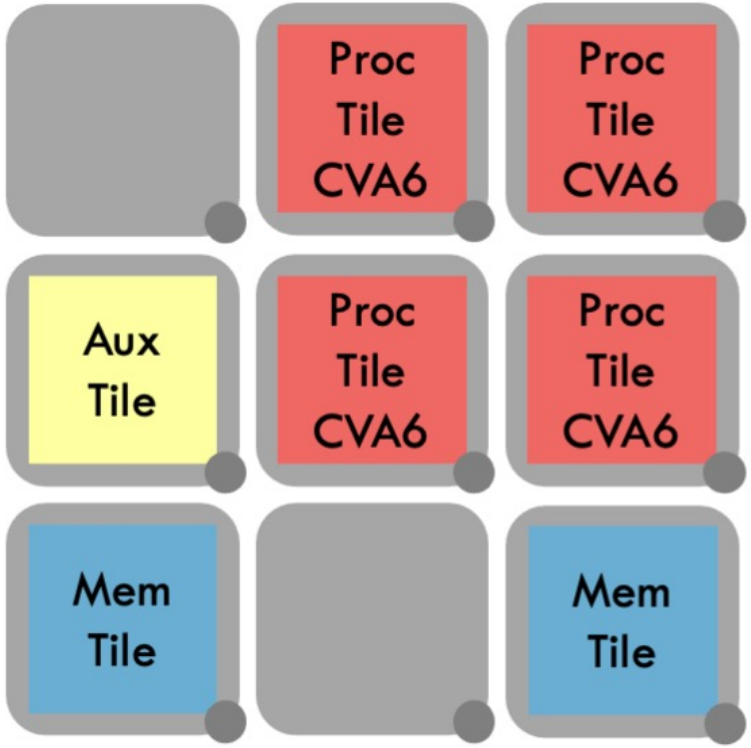
- Uses same infrastructure as AMOs, but some changes because SC is not guaranteed to succeed
- L2 must reply with success or failure on the write response channel
- Forwards are served to a cache line with an ongoing LR/SC
  - "atomic" is marked as ended, and the SC will fail upon arrival
- Instruction reads must be served between LR/SC
  - ISA prohibits data loads/stores between LR/SC
  - Tricky because SPARC implementation was not designed to handle any memory accesses during an ongoing atomic
- LR/SC are distinguished from AMO using the `atop` field
  - Forwarded over the `user` field of the AR channel



# Evaluation

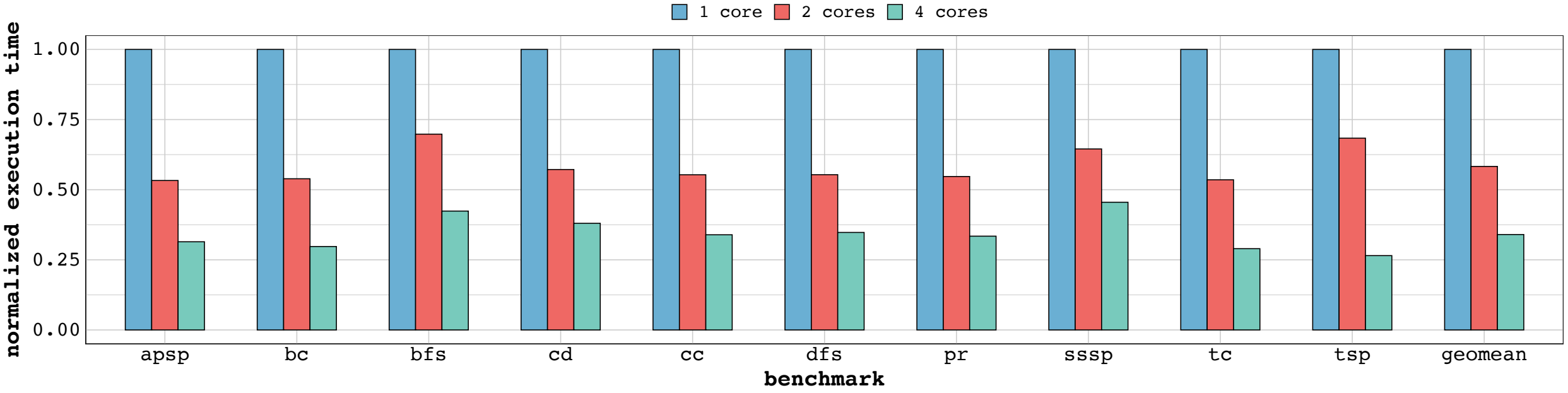
# CRONO Multithreaded Benchmark Suite

- Implementation of L1 invalidation and RISC-V AMOs took a few weeks each
- Shortly thereafter, ran first multicore baremetal programs and booted Linux SMP for first time
  - ~2 additional months for minor bug fixes
- Used the CRONO suite [17] for debugging and evaluation purposes
- Multithreaded graph algorithms
  - Path planning
  - Search
  - Graph Processing
- Ran on top of Linux SMP on FPGA with 1, 2, and 4 cores on evaluation SoC




[17] Ahmad, ISWC '15

# Experimental Results




- Near-linear performance scaling with core count
  - 2-cores 58% execution time
  - 4-cores 34% execution time
  - Roughly matches evaluation by CRONO authors [17]

[17] Ahmad, ISWC '15

# Thank you from the ESP team!

sld.cs.columbia.edu ColumbiaSld esp.cs.columbia.edu sld-columbia/esp c/ESP-platform

## Enabling Heterogeneous, Multicore SoC Research with RISC-V and ESP

**Joseph Zuckerman**, Paolo Mantovani, Davide Giri, Luca P. Carloni

**Image Credits:**
https://www.openhwgroup.org/
https://bar.eecs.berkeley.edu/
https://www.sifive.com/
https://www.t-head.cn/
https://pulp-platform.org/
https://www.gaisler.com/
https://github.com/karlrupp/microprocessor-trend-data
https://github.com/ucb-bar/chipyard
https://github.com/PrincetonUniversity/openpiton
https://github.com/black-parrot/black-parrot

CARRV 2022