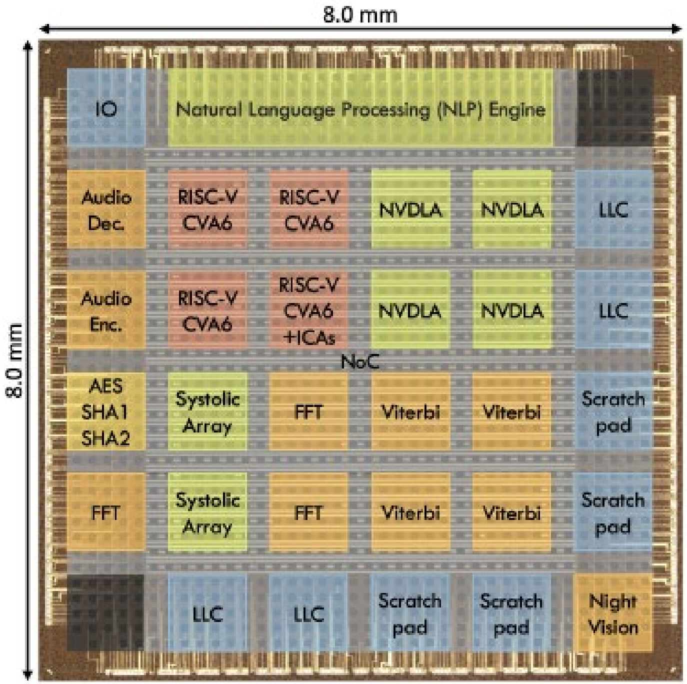
Designing Sustainable AI Systems through Agile and Collaborative Hardware Platforms

Luca P. Carloni

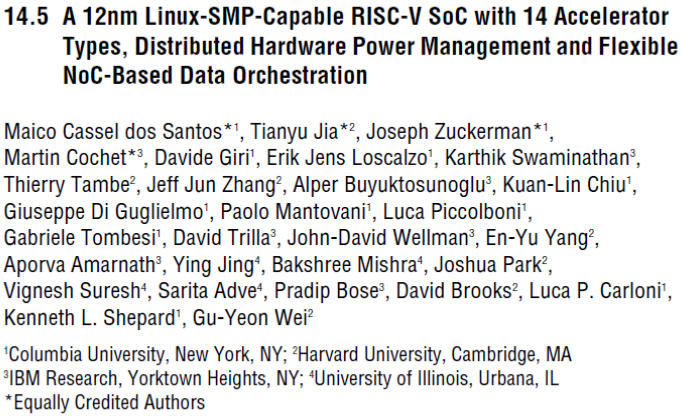
EFCL 2nd International Workshop on Sustainable AI
Christchurch, New Zealand
February 17, 2026

Columbia University
in the City of New York

ESP is Silicon Proven: The EPOCHS-1 SOC



Technology	12nm FinFET
Area	64mm^2
#IOs	340
Power Domains	23
Clock Domains	35
Power	83mW – 4.33W
Total SRAM	8.4MB
Max Frequency Range	680MHz – 1.6GHz
Example Application Domain	Collaborative Autonomous Vehicles



14.5 A 12nm Linux-SMP-Capable RISC-V SoC with 14 Accelerator Types, Distributed Hardware Power Management and Flexible NoC-Based Data Orchestration

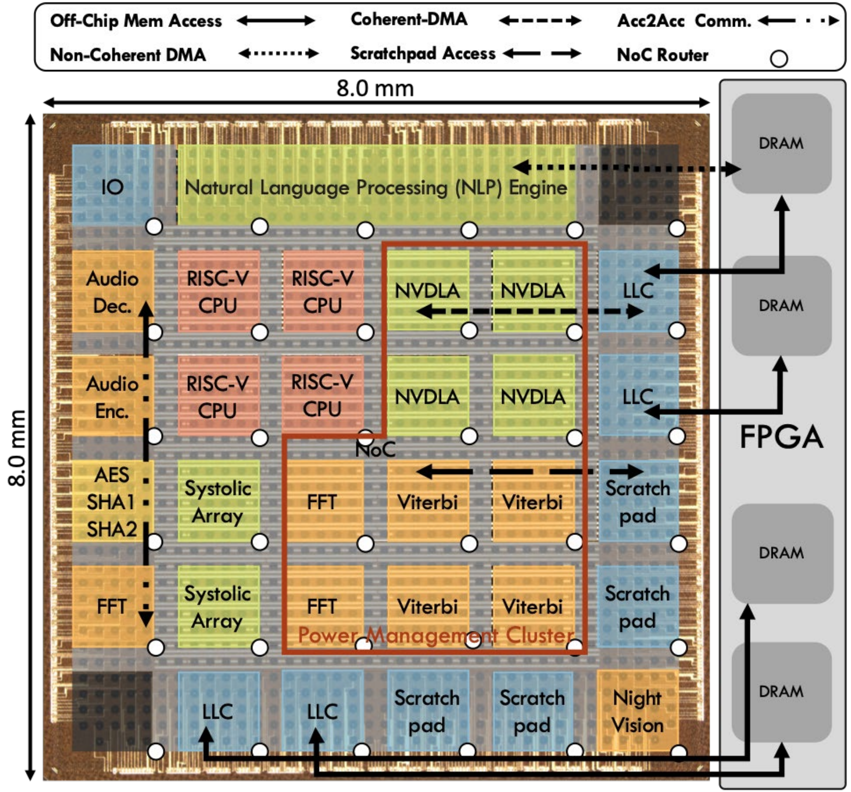
Maico Cassel dos Santos*[1], Tianyu Jia*[2], Joseph Zuckerman*[1], Martin Cochet*[3], Davide Giri[1], Erik Jens Loscalzo[1], Karthik Swaminathan[3], Thierry Tambe[2], Jeff Jun Zhang[2], Alper Buyuktosunoglu[3], Kuan-Lin Chiu[1], Giuseppe Di Guglielmo[1], Paolo Mantovani[1], Luca Piccolboni[1], Gabriele Tombesi[1], David Trilla[3], John-David Wellman[3], En-Yu Yang[2], Aporva Amarnath[3], Ying Jing[4], Bakshree Mishra[4], Joshua Park[2], Vignesh Suresh[4], Sarita Adve[4], Pradip Bose[3], David Brooks[2], Luca P. Carloni[1], Kenneth L. Shepard[1], Gu-Yeon Wei[2]

[1]Columbia University, New York, NY; [2]Harvard University, Cambridge, MA
[3]IBM Research, Yorktown Heights, NY; [4]University of Illinois, Urbana, IL
*Equally Credited Authors

ISSCC 2024 / SESSION 14 / DIGITAL TECHNIQUES FOR SYSTEM ADAPTATION, POWER MANAGEMENT AND CLOCKING / 14.5

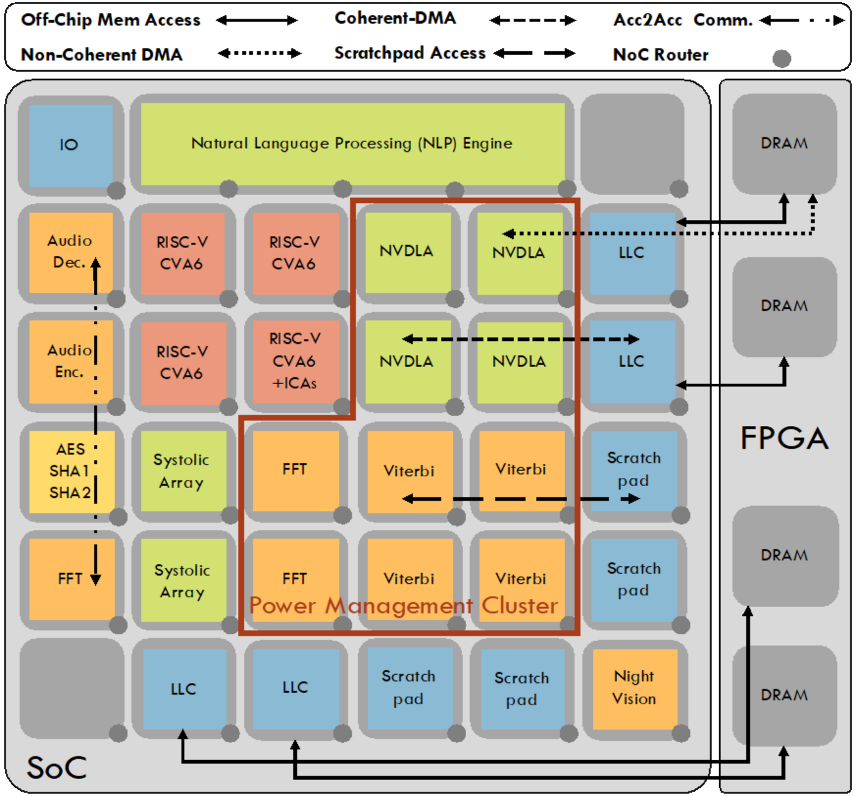
The EPOCHS-1 SoC: Chip Highlights

- 64 mm^2 SoC designed in 12 nm FinFET
- 35 clock domains; 23 power domains
- 8.4 MB on-chip SRAM memory
- Tile-based SoC architecture



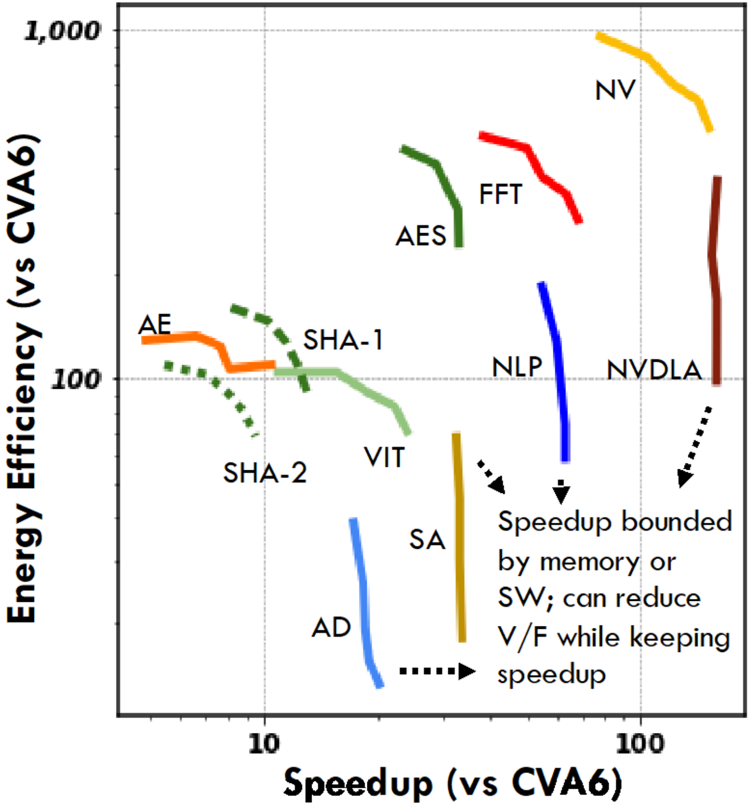
The EPOCHS-1 SoC: Chip Highlights

- 64 mm^2 SoC designed in 12 nm FinFET
- 35 clock domains; 23 power domains
- 8.4 MB on-chip SRAM memory
- Tile-based SoC architecture
- 34 tiles connected by a 6-plane, 2-D mesh NoC
- The 74 Tbps NoC provides flexible orchestration of data
- 23 accelerators of 14 different types
- 10 accelerators compose a cluster demonstrating a novel distributed hardware power management (DHPM) scheme
- Designed by a small team of PhD students, postdocs, and industry researchers in 3 months with ESP, our open-source platform for agile SoC design



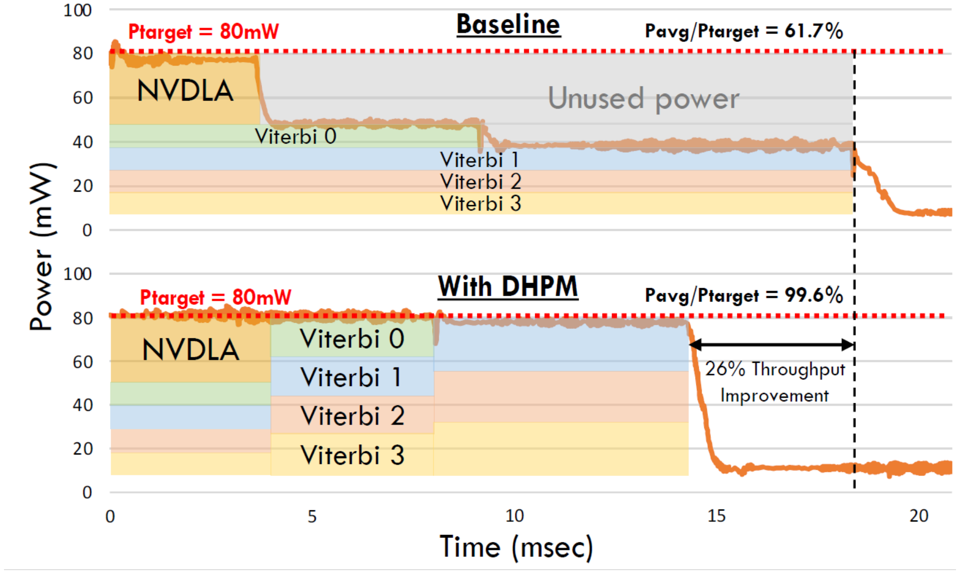
The EPOCHS-1 SoC: What Can It Do?

- Accelerate a lot of different applications!
- Speedups of 6-159x and energy efficiency gains of 6 – 959x versus software



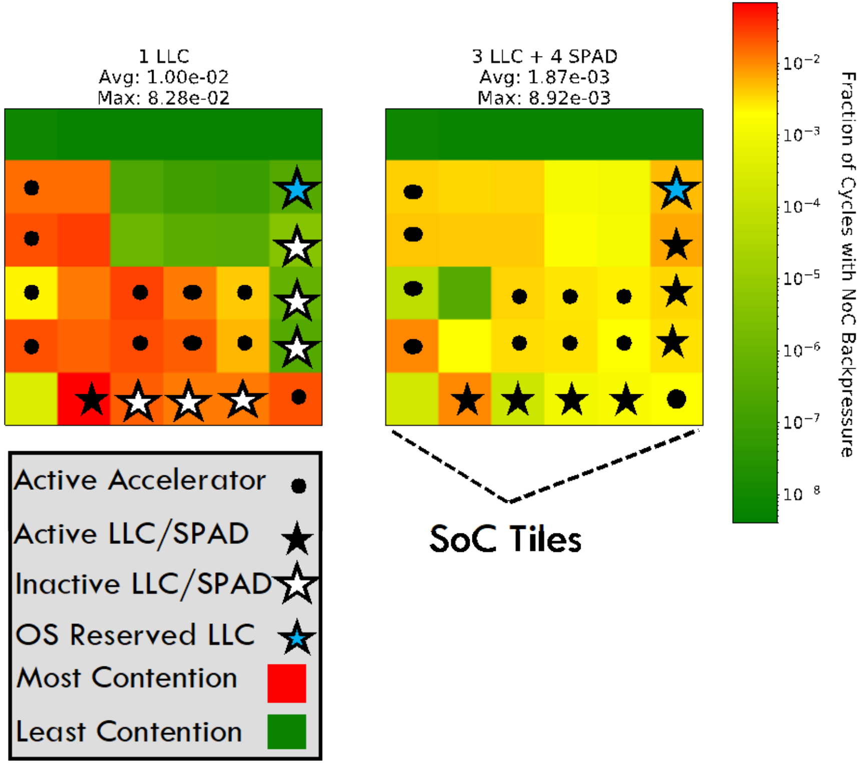
The EPOCHS-1 SoC: What Can It Do?

- Accelerate a lot of different applications!
- Speedups of 6-159x and energy efficiency gains of 6 – 959x versus software
- **Dynamically allocate power to accelerators without involving SW**
  - **Improves power utilization by 39% and throughput by 27%**



The EPOCHS-1 SoC: What Can It Do?

- Accelerate a lot of different applications!
- Speedups of 6-159x and energy efficiency gains of 6 – 959x versus software
- Dynamically allocate power to accelerators without involving SW
  - Improves power utilization by 39% and throughput by 27%
- **Tailor its memory hierarchy and on-chip communication to applications' needs**

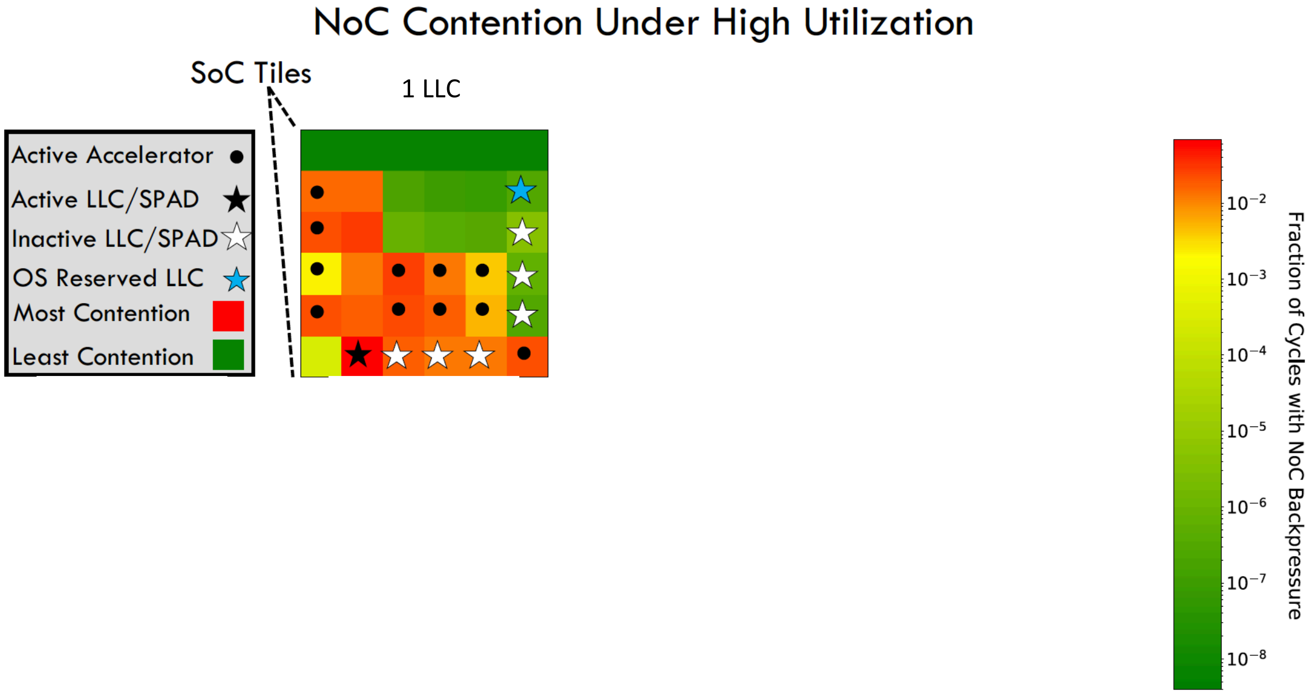


The EPOCHS-1 SoC: What Can It Do?

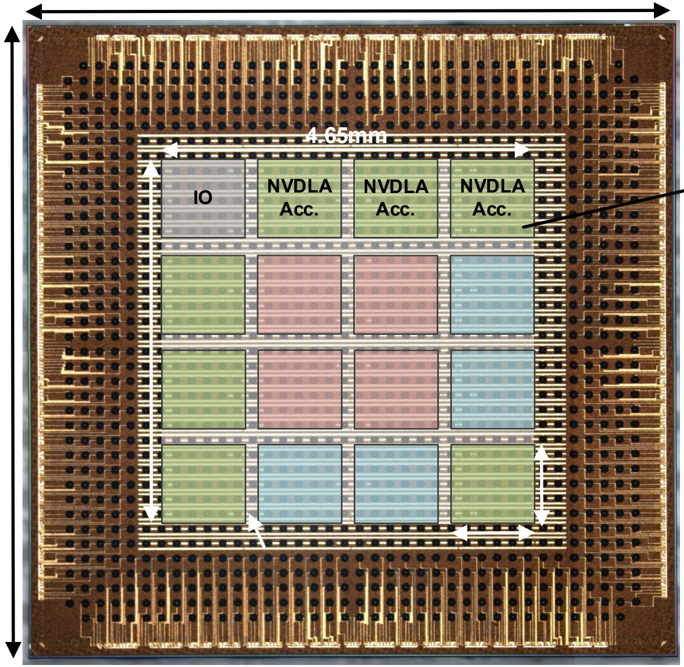
- NoC traffic with 11 accelerators executing in parallel
  - “Contention” = # of cycles when a queue is full and asserts backpressure

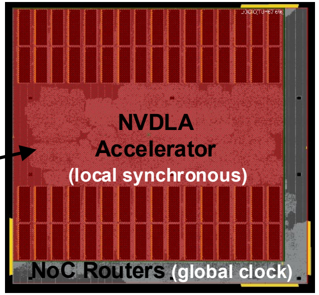
- 7 different configurations of the memory hierarchy

- Scaling up the memory hierarchy alleviates contention and distributes traffic



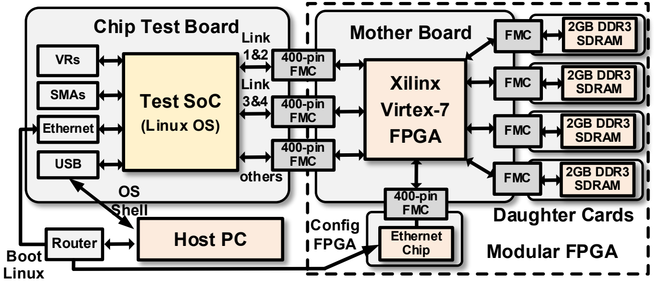
The EPOCHS-o Chip

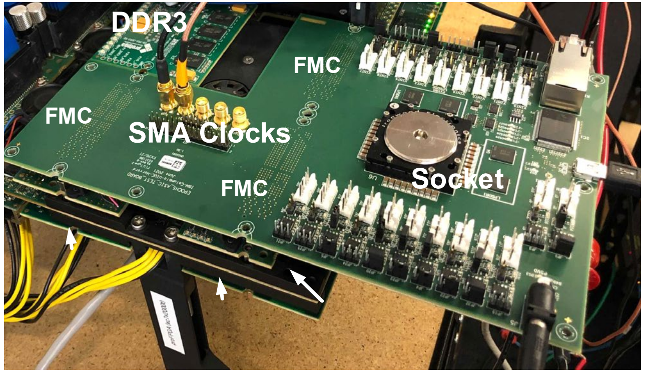



NVDLA Accelerator (local synchronous)
NoC Routers (global clock)

C4 Bump #	1439

12nm FinFET test chip



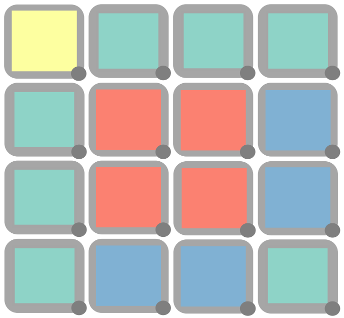


Test Setup

[T. Jia, et al. "A 12nm Agile-Designed SoC for Swarm-Based Perception with Heterogeneous IP Blocks, a Reconfigurable Memory Hierarchy, and an 800MHz Multi-Plane NoC, ESSCIRC 2022]

A Scalable Approach to Chip Design

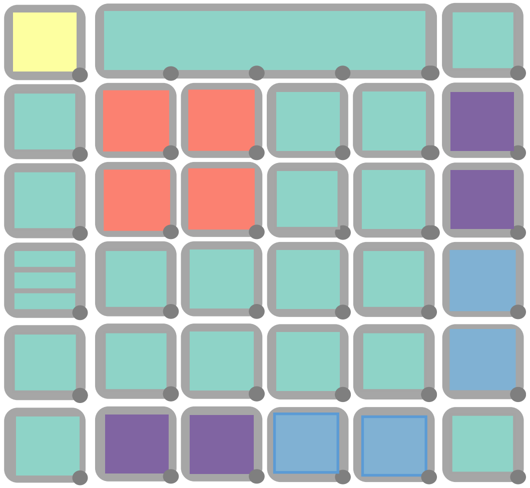
EPOCHS-0



7 new accelerators tiles
2.25x more tiles
2.18x more clock domains
2.25x more power domains
2.96x more area
Same tile imp. running time
+29% top imp. running time

- 4x4 tiles
- 21.62 mm^2
- 17 clock domains
- 16 power domains
- Tile: 12 hours in 16-core 64GB RAM machine
- Top: 51 hours in 64-core 376 GB RAM machine

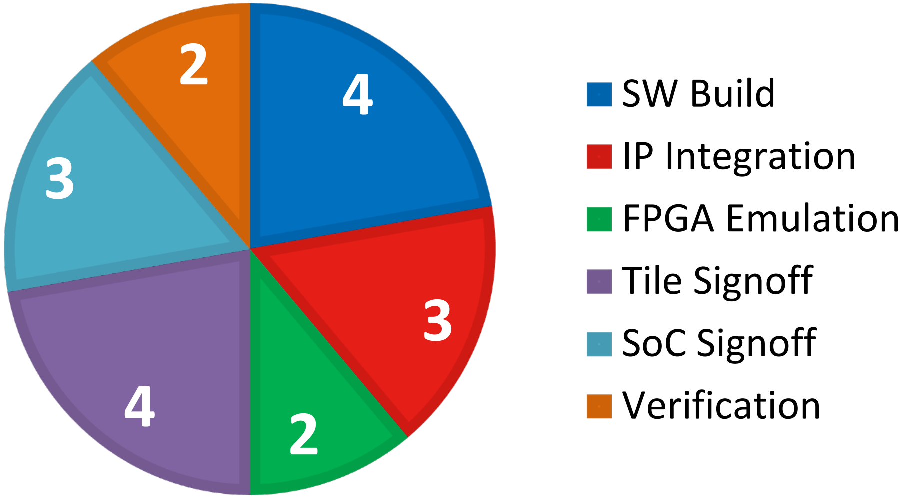
EPOCHS-1



- 6x6 tiles
- 64 mm^2
- 37 clock domains
- 23 power domains
- Tile: 12 hours in 16-core 64GB RAM machine
- Top: 66 hours in 64-core 376 GB RAM machine

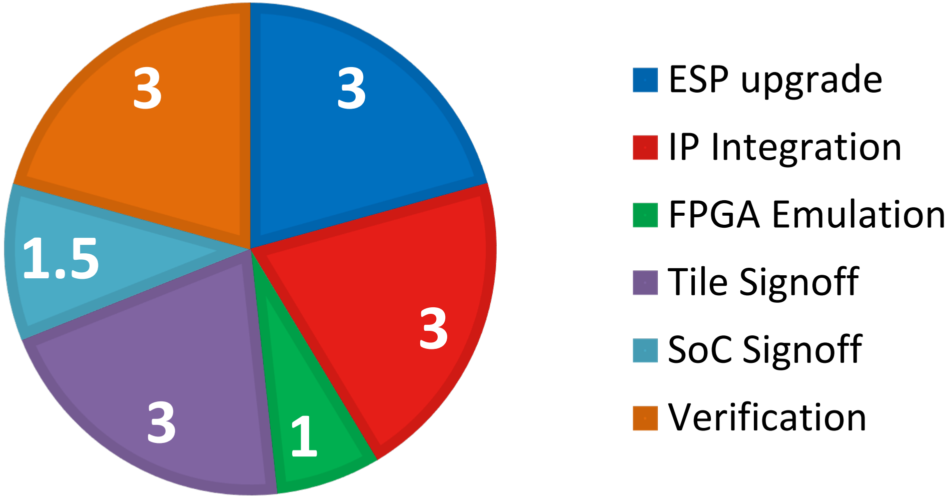
A Scalable Approach to Chip Design

EPOCHS-0 DESIGN CYCLE (WEEKS)



~ 4 months

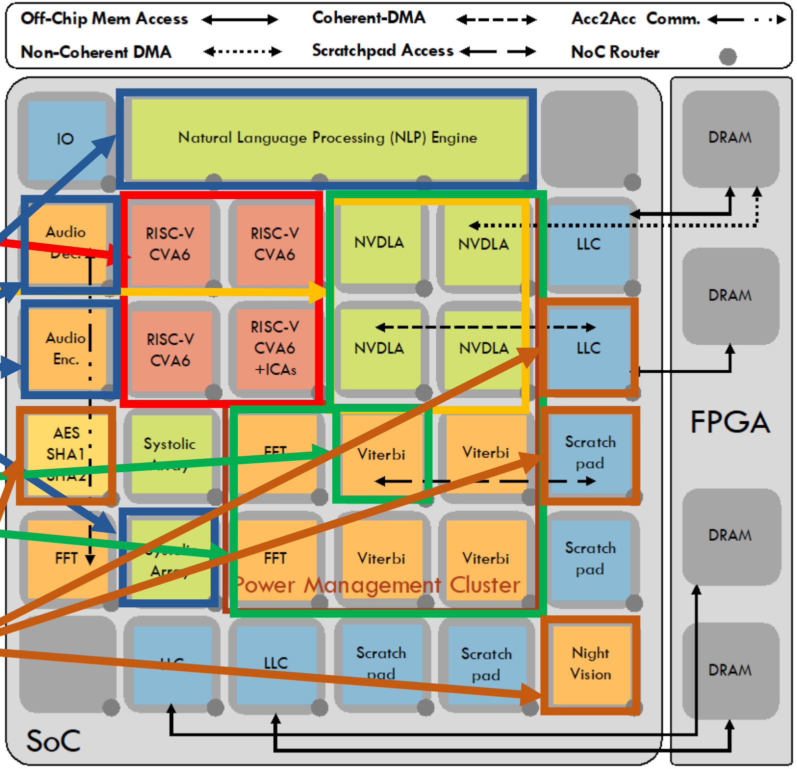
EPOCHS-1 DESIGN CYCLE (WEEKS)



~ 3 months

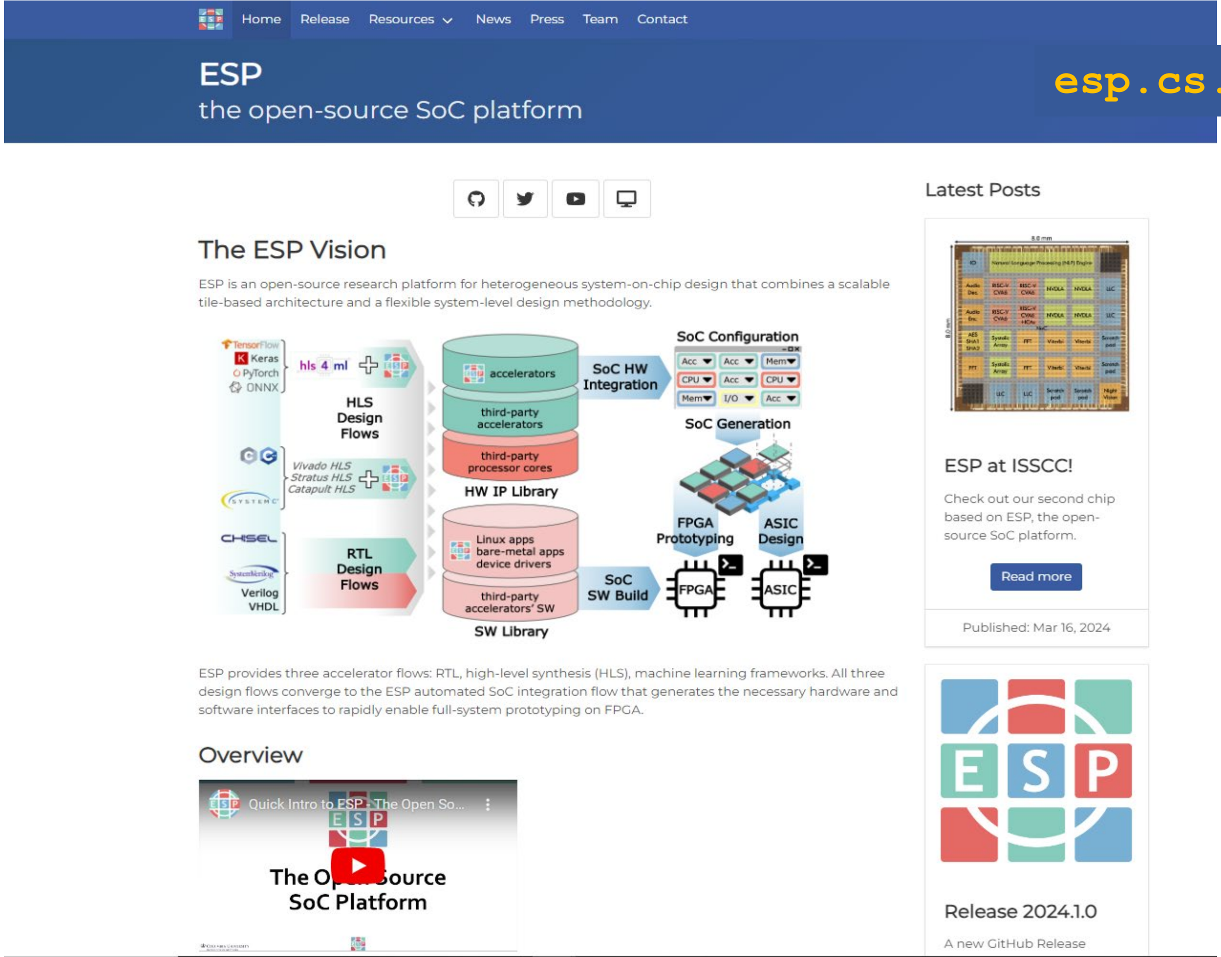
The EPOCHS-1 SoC: Sources of OSH IPs

- Sources of Open-Source Hardware IPs:
  - 4 RISC-V CVA6 cores from ETH Zurich/OpenHW Group
  - 4 NVIDIA Deep Learning Accelerators
  - 4 Accelerators designed at Harvard
  - 1 Accelerator and Power Management designed at IBM Research
  - 3 Accelerators, Memory Hierarchy, and Network-on-Chip designed at Columbia



ESP : An Open-Source Platform for SoC Design

esp.cs.columbia.edu



Home Release Resources News Press Team Contact

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

Quick Intro to ESP - The Open So...
ESP
The Open-Source SoC Platform

Latest Posts

ESP at ISSCC!

Check out our second chip based on ESP, the open-source SoC platform.

Read more

Published: Mar 16, 2024

Release 2024.1.0

A new GitHub Release

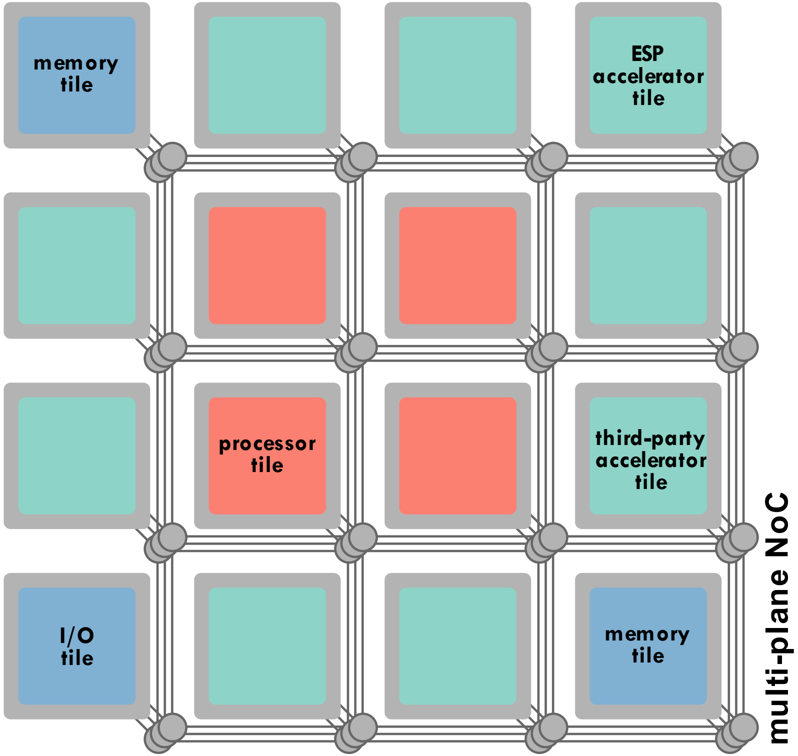
The Concept of Platform

- **Innovation in SoC architectures and their design methodologies is needed to promote design reuse and collaboration**
  - Architectures and methodologies must be developed together
- ***Platform = architecture + methodology***
  - An SoC architecture enables design reuse when it simplifies the integration of many components that are independently developed
  - An SoC methodology enables design collaboration when it allows designers to choose the preferred specification languages and design flows for the various components
- **An effective combination of architecture and methodology is a platform that maximizes the potential of open-source hardware**
  - by scaling up the number and type of components that can be integrated in an SoC and by enhancing the productivity of the designers who develop and use them

ESP Architecture

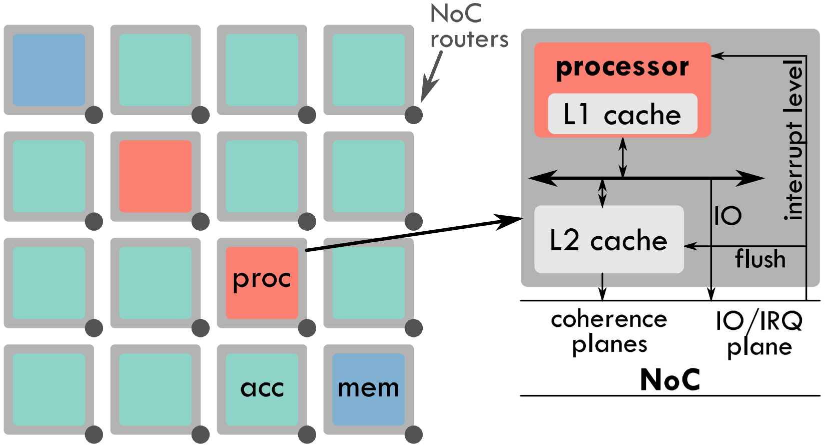
- RISC-V Processors
- Many-Accelerator
- Distributed Memory
- Multi-Plane NoC

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



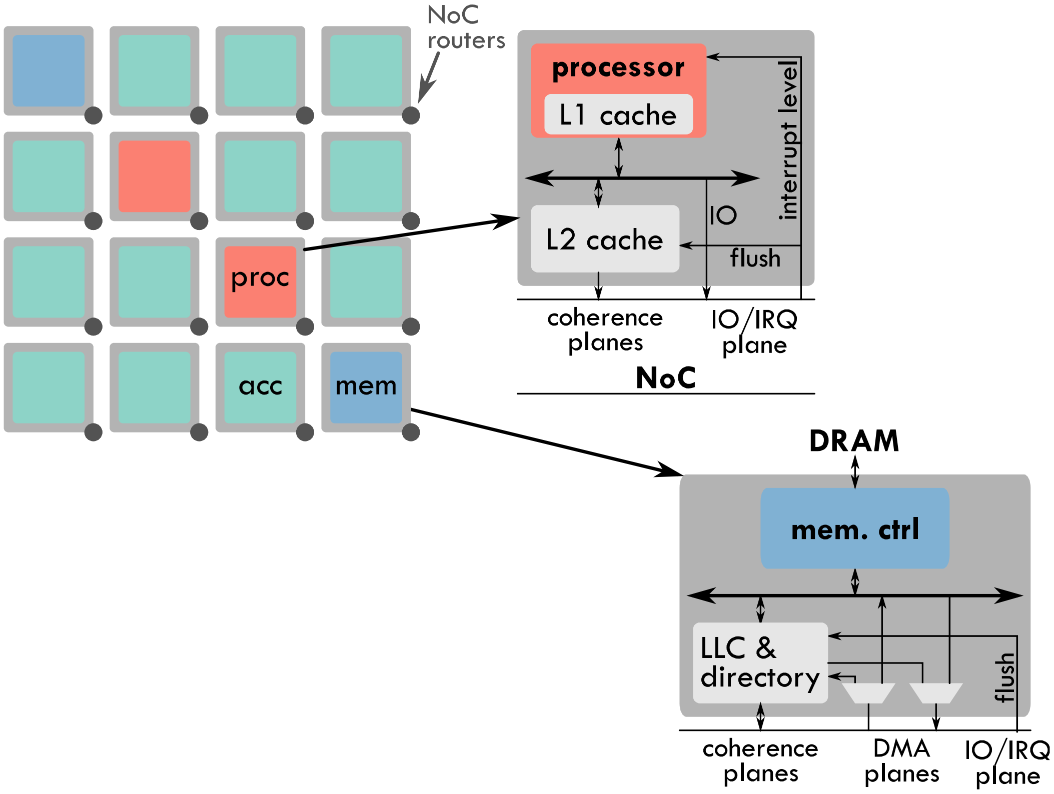
ESP Architecture: Processor Tile

- 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, …



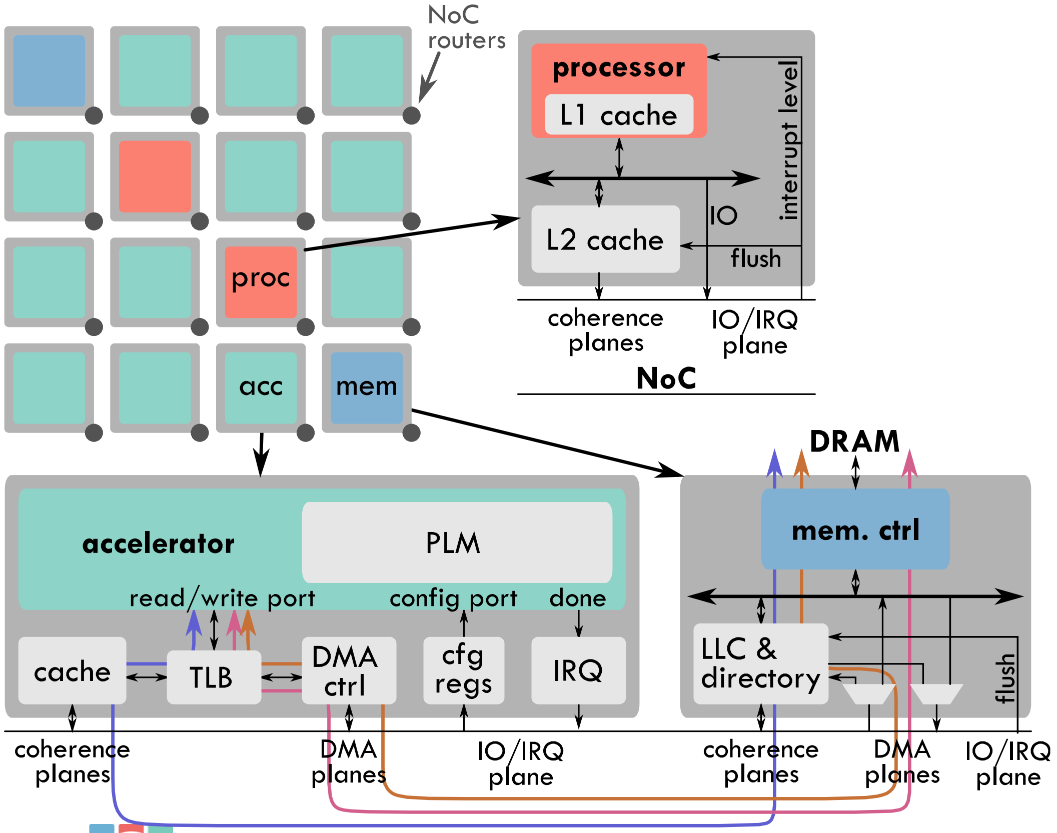
ESP Architecture: Memory Tile

- **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

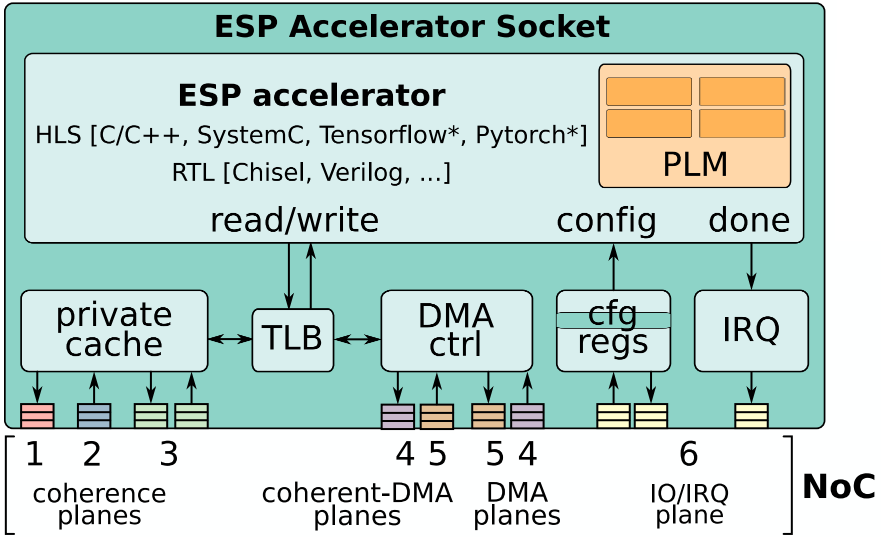
- Accelerator Socket w/ Platform Services
  - Direct-memory-access
  - Run-time selection of coherence model:
    - Fully coherent
    - LLC coherent
    - Non coherent
  - User-defined registers
  - Distributed interrupt

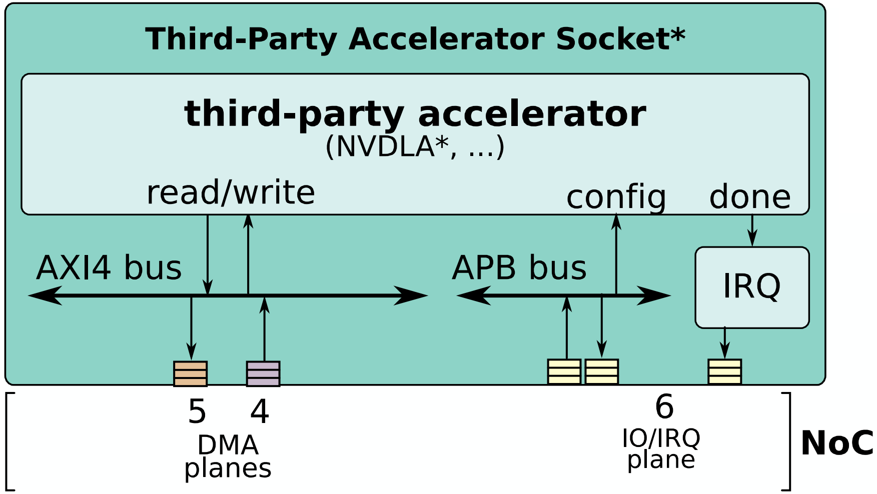


©Luca Carloni

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

ESP Accelerator Socket





ESP Platform Services

Accelerator tile

- DMA
- Reconfigurable coherence
- Point-to-point
- ESP or AXI interface
- DVFS controller

Processor Tile

- Coherence
- I/O and un-cached memory
- Distributed interrupts
- DVFS controller

Miscellaneous Tile

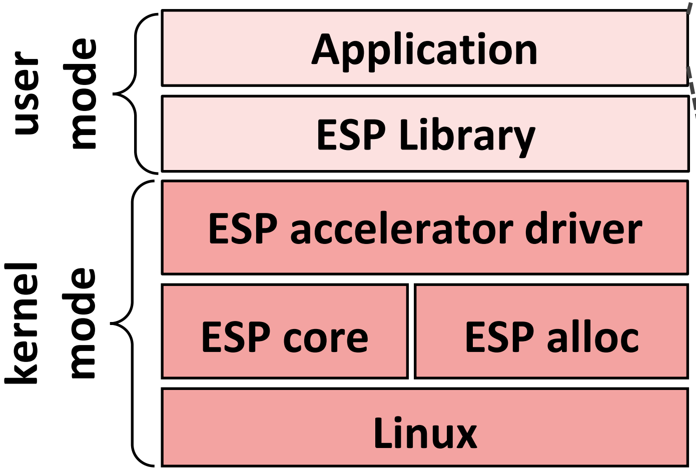
- Debug interface
- Performance counters access
- Coherent DMA
- Shared peripherals (UART, ETH, …)

Memory Tile

- Independent DDR Channel
- LLC Slice
- DMA Handler

ESP Software Socket

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

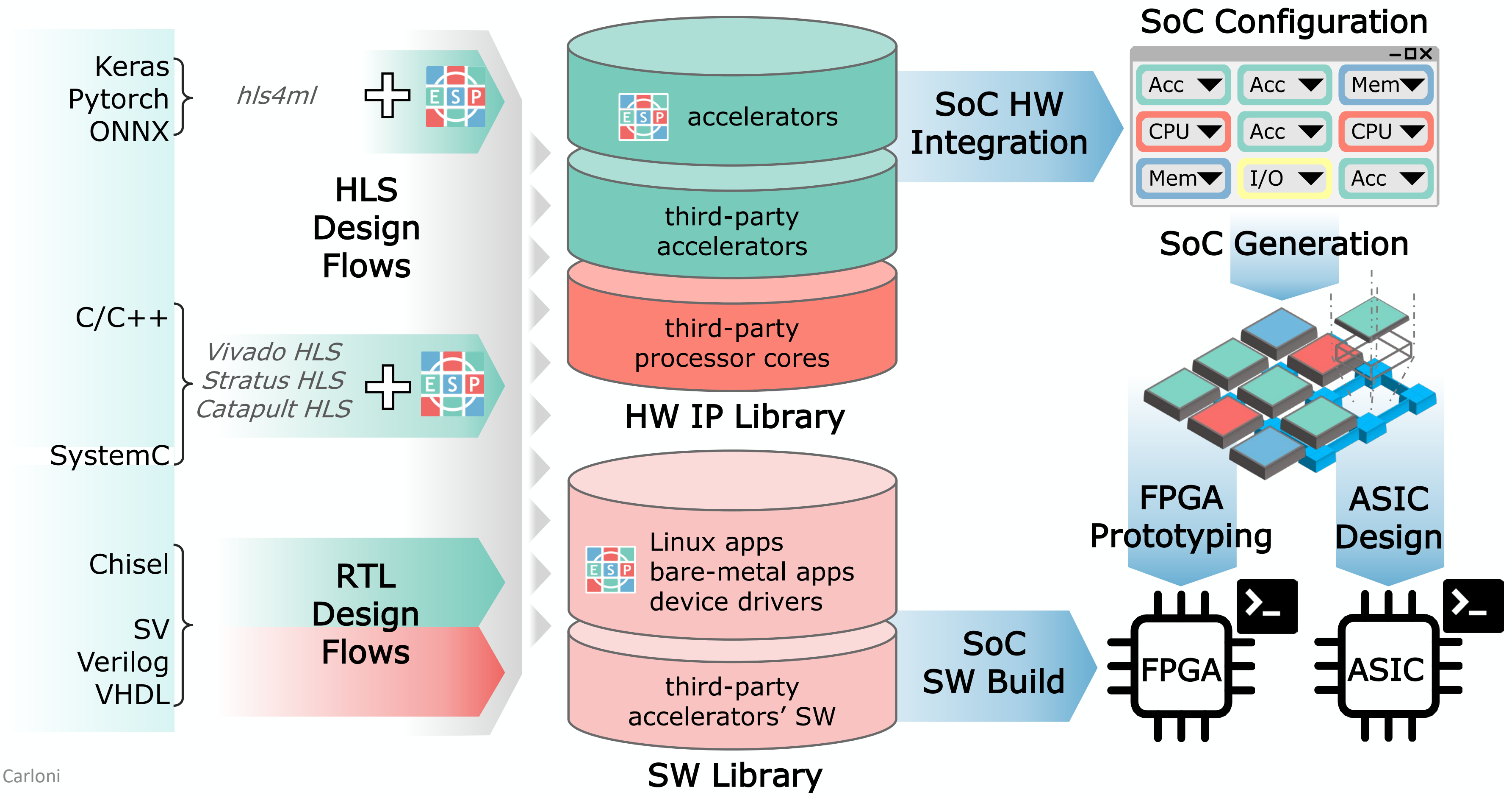


```
/*
 * Example of existing C application with ESP
 * accelerators that replace software kernels 2, 3,
 * and 5. The cfg_k# contains buffer and the
 * accelerator configuration.
 */
{
  int *buffer = esp_alloc(size);

  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        */
}
```

The Pillars of the ESP Approach

- Develop platforms, not just architectures
  - A platform combines an architecture and a companion design methodology
- Move from a processor-centric to an SoC-centric perspective
  - The processor core is just one component among many others
- Raise the level of abstraction
  - Move from RTL design to domain-specific *system-level design* with high-level synthesis…
  - …but keep supporting different abstraction levels and design flows
- Promote Open-Source Hardware for Agile, Collaborative Design
  - Build libraries of reusable components
  - Support the integration of third-party IP components

The ESP Vision: Domain Experts Can Design SoCs



Keras
Pytorch
ONNX

hls4ml + ESP

C/C++
SystemC

Vivado HLS
Stratus HLS
Catapult HLS + ESP

HLS Design Flows

Chisel
SV
Verilog
VHDL

RTL Design Flows

ESP accelerators
third-party accelerators
third-party processor cores

HW IP Library

ESP Linux apps
bare-metal apps
device drivers

third-party accelerators' SW

SW Library

SoC HW Integration

SoC SW Build

SoC Configuration

Acc ▼	Acc ▼	Mem ▼
CPU ▼	Acc ▼	CPU ▼
Mem ▼	I/O ▼	Acc ▼

SoC Generation

FPGA Prototyping

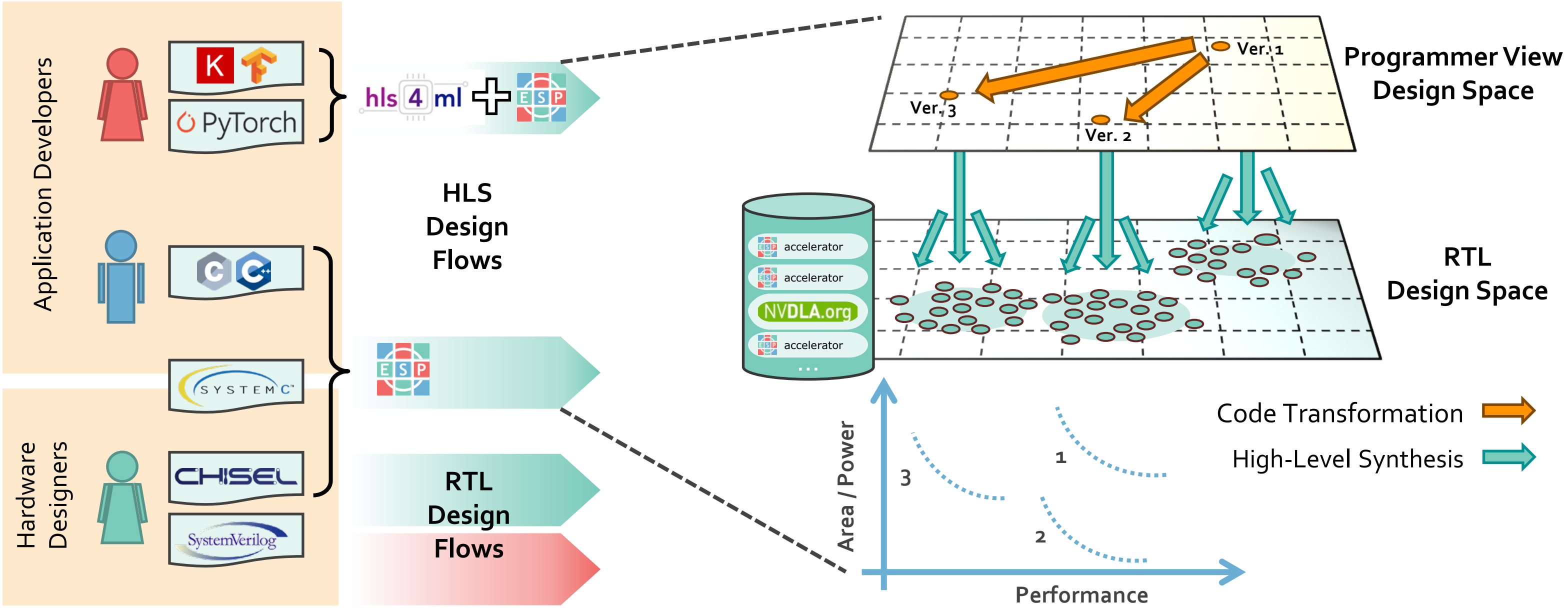
ASIC Design

FPGA

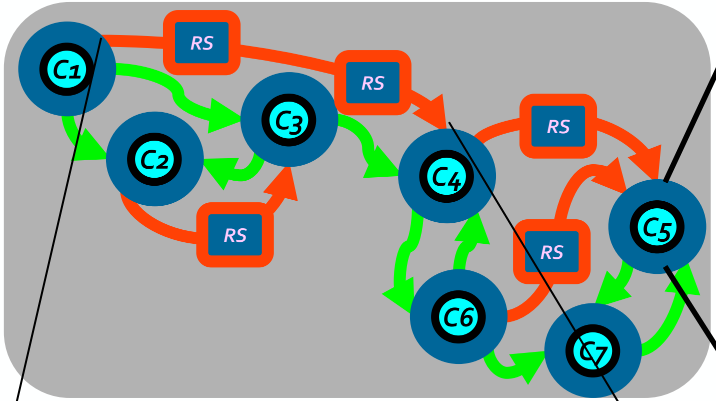
ASIC

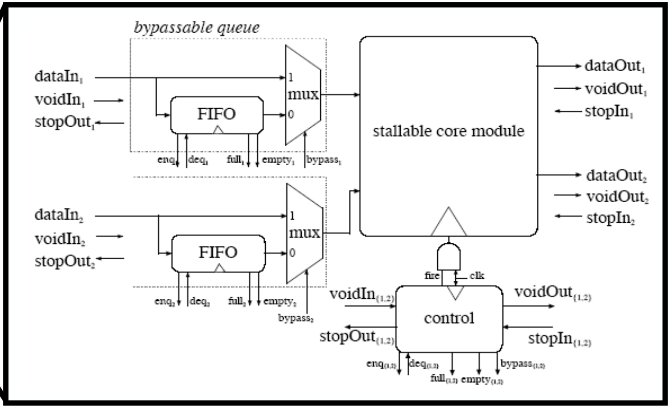
ESP Accelerator Flow

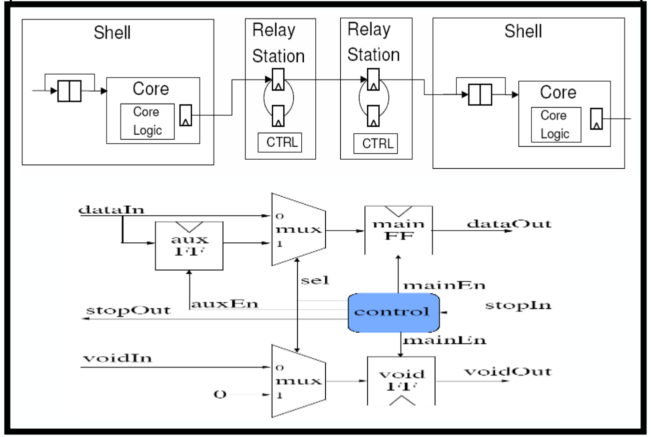
Developers focus on the **high-level specification**, **decoupled** from memory access, system communication, hardware/software interface



Retrospective: Latency-Insensitive Design [Carloni et al. 1999]



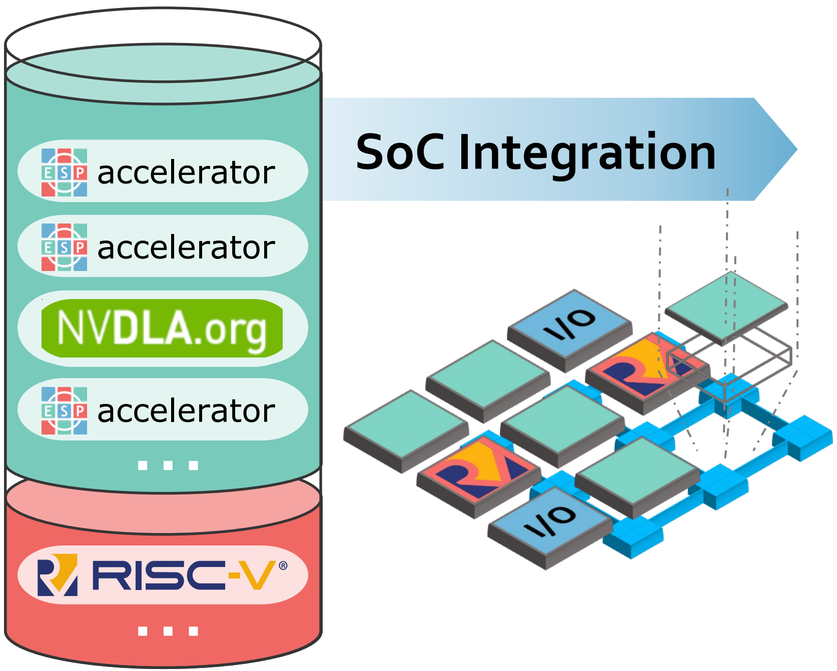


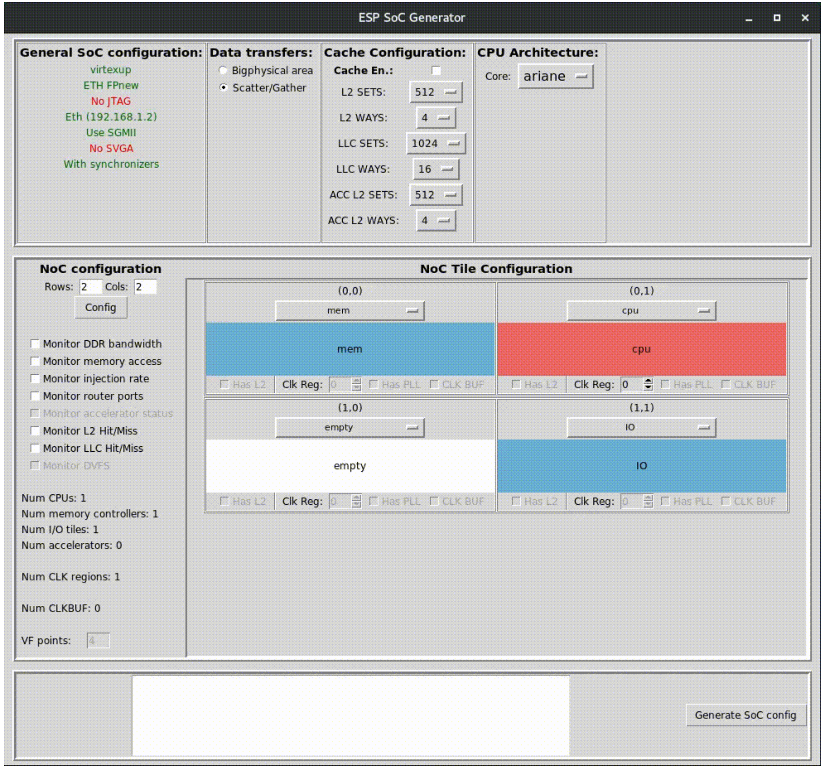


Latency-Insensitive Design

- is the foundation for the *flexible synthesizable RTL representation*
- anticipates the separation of computation from communication that is proper of Transaction-Level Modeling with SystemC
  - through the introduction of the Protocols & Shell paradigm

ESP Interactive Flow for SoC Integration





ESP SoC Generator

General SoC configuration:

virtexup
ETH FPnew
No JTAG
Eth (192.168.1.2)
Use SGMII
No SVGA
With synchronizers

Data transfers:

○ Bigphysical area
● Scatter/Gather

Cache Configuration:

Cache En.: □
L2 SETS: 512
L2 WAYS: 4
LLC SETS: 1024
LLC WAYS: 16
ACC L2 SETS: 512
ACC L2 WAYS: 4

CPU Architecture:

Core: ariane

NoC configuration

Rows: 2 Cols: 2

□ Monitor DDR bandwidth
□ Monitor memory access
□ Monitor injection rate
□ Monitor router ports
□ Monitor accelerator status
□ Monitor L2 Hit/Miss
□ Monitor LLC Hit/Miss
□ Monitor DVFS

Num CPUs: 1
Num memory controllers: 1
Num I/O tiles: 1
Num accelerators: 0

Num CLK regions: 1

Num CLKBUF: 0

VF points: 4

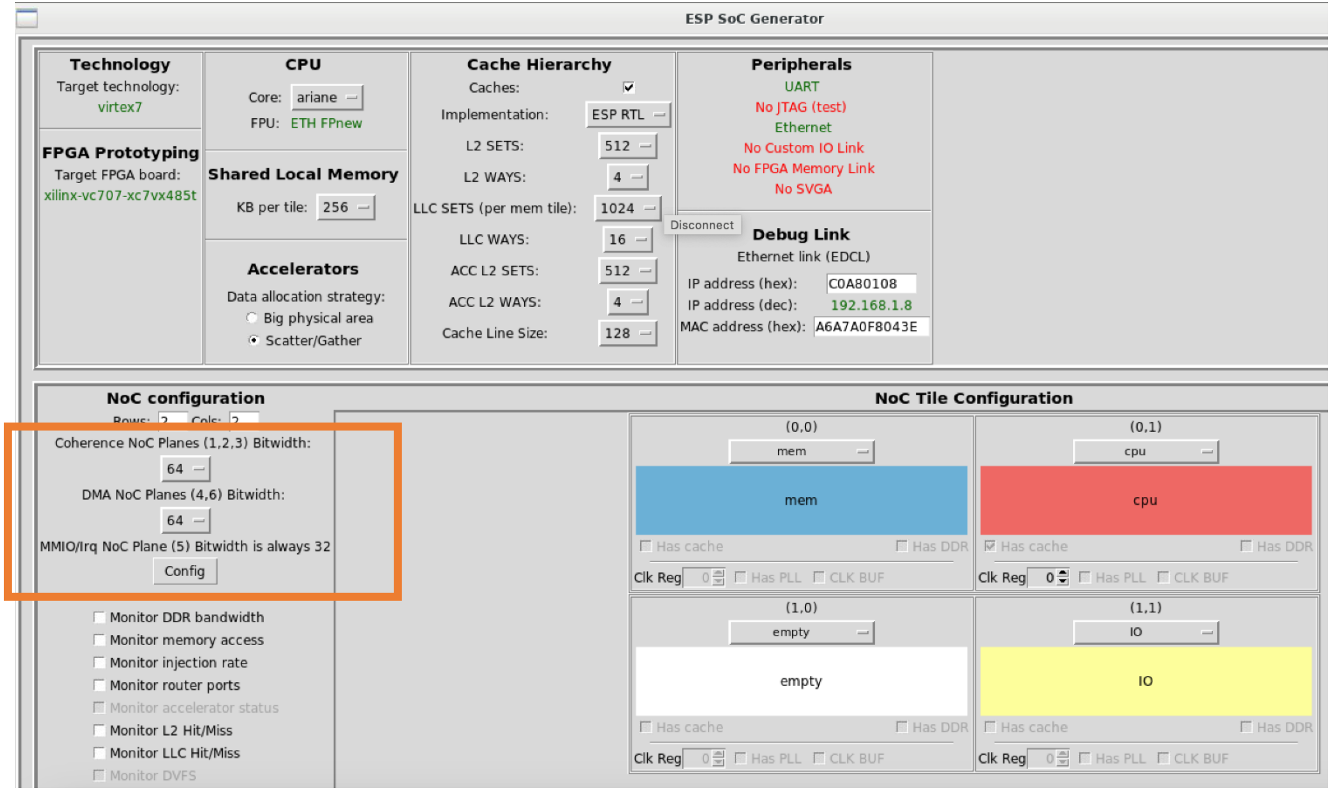
NoC Tile Configuration

(0,0) mem — mem — □ Has L2 Clk Reg: 0 □ Has PLL □ CLK BUF
(0,1) cpu — cpu — □ Has L2 Clk Reg: 0 □ Has PLL □ CLK BUF
(1,0) empty — empty — □ Has L2 Clk Reg: 0 □ Has PLL □ CLK BUF
(1,1) IO — IO — □ Has L2 Clk Reg: 0 □ Has PLL □ CLK BUF

Towards a Chipletized ESP Architecture

NoC Bandwidth Configurability

- Coherence and DMA planes are now independently configurable up to 1024 bits



ESP SoC Generator

Technology
Target technology: virtex7

FPGA Prototyping
Target FPGA board: xilinx-vc707-xc7vx485t

CPU
Core: ariane
FPU: ETH FPnew

Shared Local Memory
KB per tile: 256

Accelerators
Data allocation strategy:
○ Big physical area
● Scatter/Gather

Cache Hierarchy
Caches: ☑
Implementation: ESP RTL
L2 SETS: 512
L2 WAYS: 4
LLC SETS (per mem tile): 1024
LLC WAYS: 16
ACC L2 SETS: 512
ACC L2 WAYS: 4
Cache Line Size: 128

Peripherals
UART
No JTAG (test)
Ethernet
No Custom IO Link
No FPGA Memory Link
No SVGA

Disconnect

Debug Link
Ethernet link (EDCL)
IP address (hex): C0A80108
IP address (dec): 192.168.1.8
MAC address (hex): A6A7A0F8043E

NoC configuration
Rows: 2 Cols: 2
Coherence NoC Planes (1,2,3) Bitwidth: 64
DMA NoC Planes (4,6) Bitwidth: 64
MMIO/Irq NoC Plane (5) Bitwidth is always 32
Config

□ Monitor DDR bandwidth
□ Monitor memory access
□ Monitor injection rate
□ Monitor router ports
□ Monitor accelerator status
□ Monitor L2 Hit/Miss
□ Monitor LLC Hit/Miss
□ Monitor DVFS

NoC Tile Configuration

(0,0)
mem
mem
□ Has cache □ Has DDR
Clk Reg 0 □ Has PLL □ CLK BUF

(0,1)
cpu
cpu
☑ Has cache □ Has DDR
Clk Reg 0 □ Has PLL □ CLK BUF

(1,0)
empty
empty
□ Has cache □ Has DDR
Clk Reg 0 □ Has PLL □ CLK BUF

(1,1)
IO
IO
□ Has cache □ Has DDR
Clk Reg 0 □ Has PLL □ CLK BUF

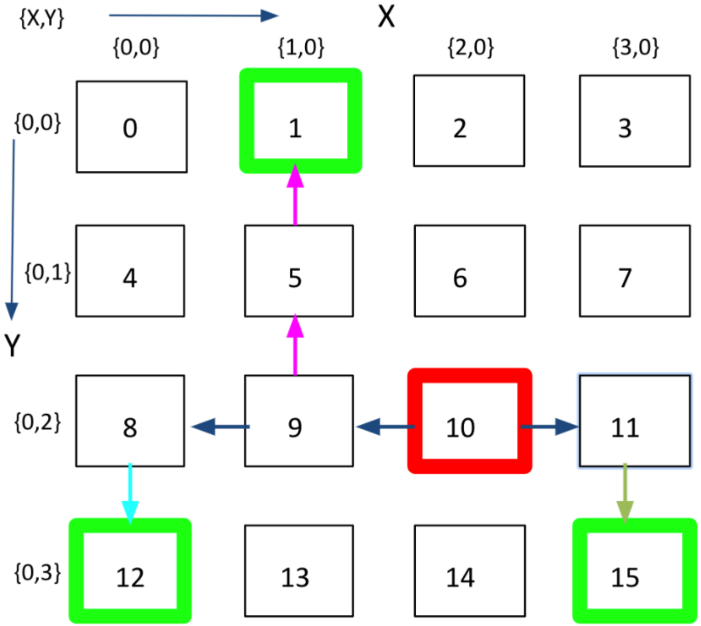
Updated ESP GUI

Multicast-Enabled NoC

- Support multicast with minimal changes to the NoC
  - The NoC is a critical component of ESP and current version is silicon proven
  - Don’t want to change routing algorithms, etc. that guarantee deadlock freedom in ESP

- Preserve dimension-ordered routing,
  X first then Y

- “Fork” whenever there are 2 destinations that require different paths

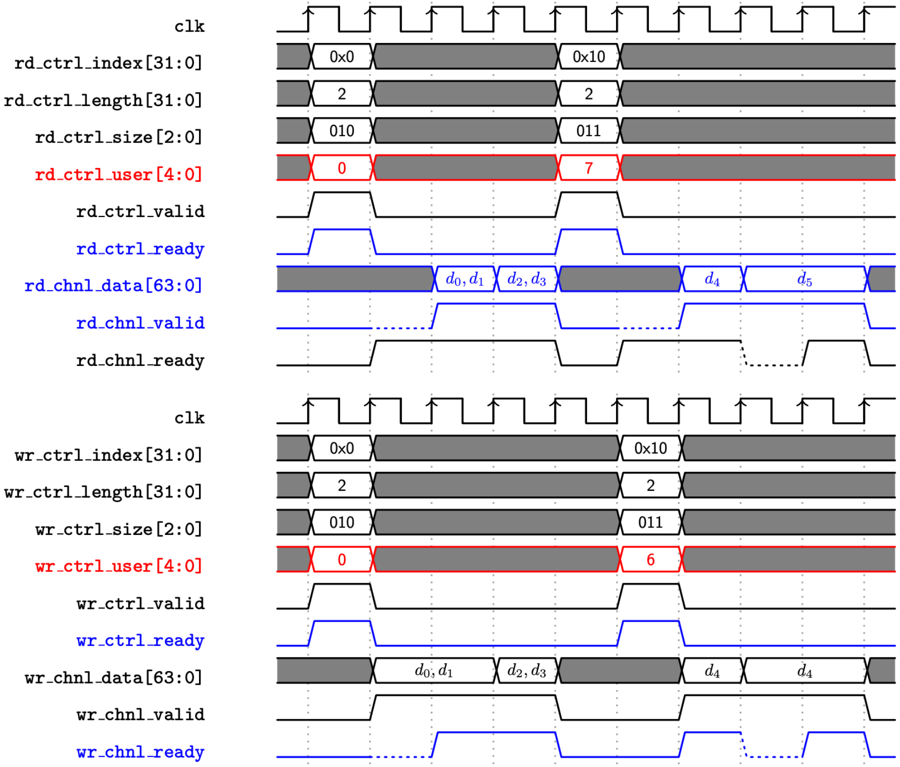


- Source: Tile 10
- Destinations: 1, 12, 15

©Luca Carloni

Fine-Grained, Reconfigurable Accelerator Communication

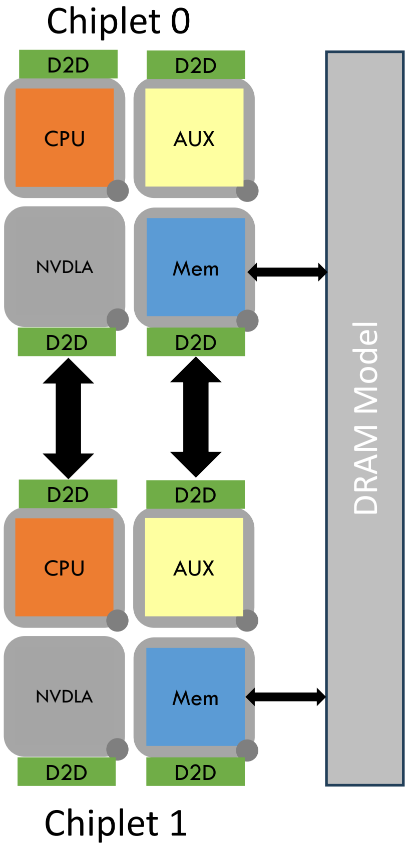
- Modify the ESP accelerator interface to give the accelerator fine grained communication control
- Multicast P2P
  - Source specifies number of targets *N*
  - Waits for *N* requests (satisfies consumption assumption)
  - Consumer sends a request to specified source (pull-based)
  - Once all *N* requests are received, data is sent with multicast



©Luca Carloni

Towards Supporting Chiplet-Based Design in ESP

- With a larger mesh of tiles, need to relax existing ESP constraints
    - Up to 16x16 mesh (previously 8x8), 16 CPU cores (previously 4), 16 memory tiles (previously 4), 512 total devices (previously 128)
- Expose the NoC at the top level of the chip to enable connections between multiple chiplets
- Generate address maps for full system of chiplets
- Successful simulation of a system of 2 ESP chiplets



Looking Ahead to the Age of AI-Aided SoC Design: How Do We…

- **Develop effective AI agents for SoC Design? Build on open platforms**
  - Open-source hardware and reusable design flows can serve as a substrate for agile, collaborative SoC design that supports the progress of agentic AI, without requiring access to proprietary industrial datasets
- **Support workforce development? Recognize that academia is key**
- **Recruit and develop talented students? Offer exciting projects**
- **Enable impactful research? Think at the system level. Prototype**
  - in integrated circuit design, computer-aided design, computer architecture
- **Enable effective collaborations? Build non-competitive hubs**
  - across University labs, industry labs, and government labs
- **Work together on this? Support Open-Source Hardware**

The OSCAR Workshop Series

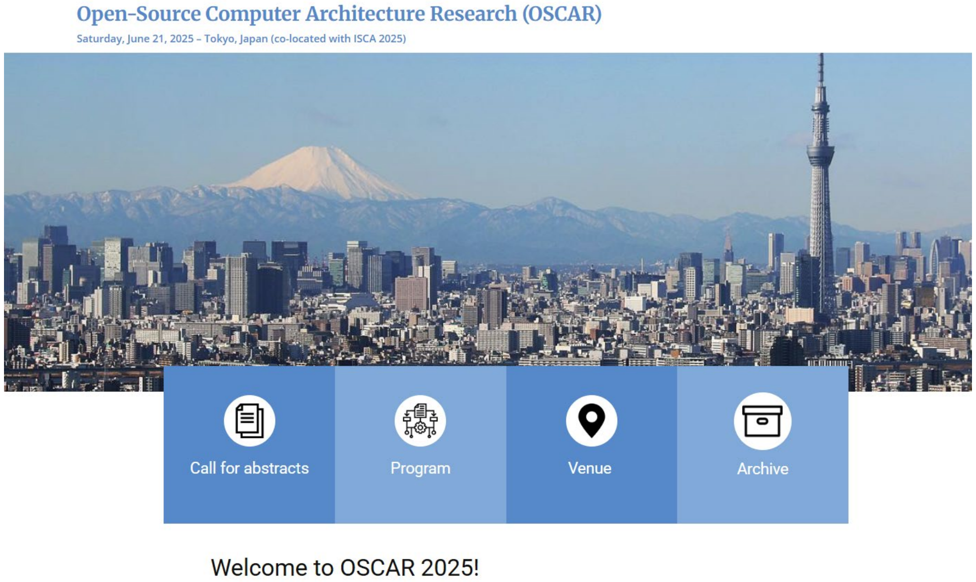
https://oscar-workshop.github.io/

- *OSCAR 2026 will be held on June 28, in Raleigh, NC USA co-located with ISCA'26 !*

- 2024 - Buenos Aires, Argentina

- 2023 - Orlando, FL

- 2022 - New York, NY



Open-Source Computer Architecture Research (OSCAR)

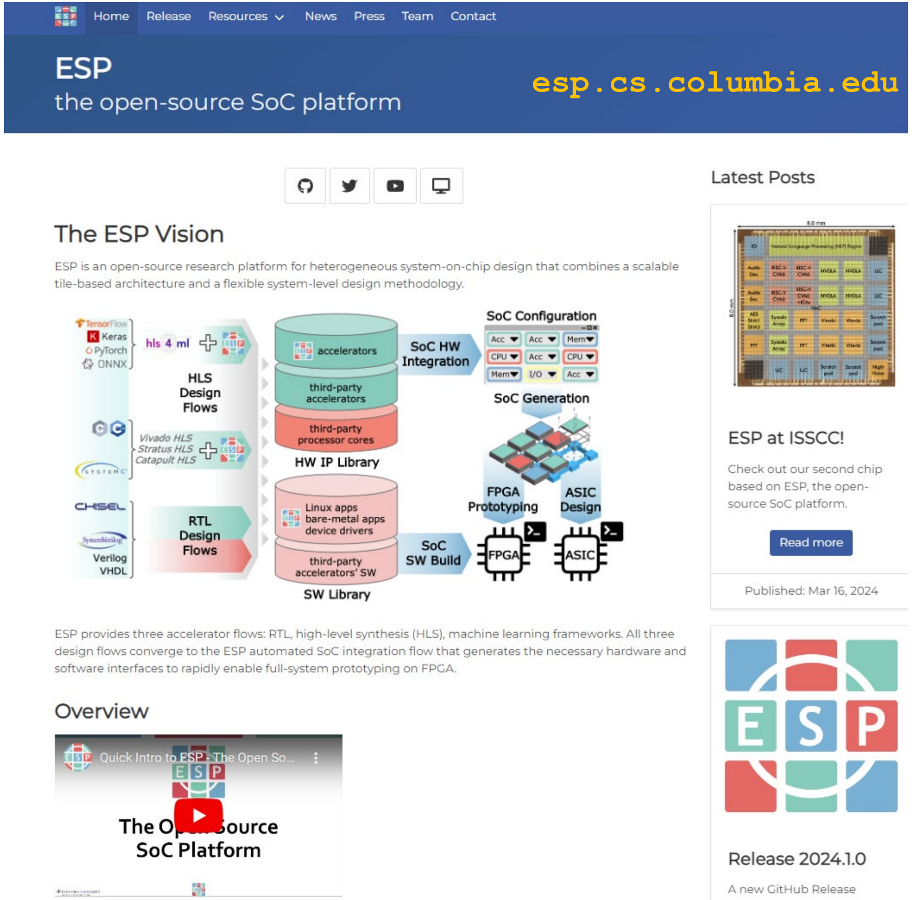
Saturday, June 21, 2025 – Tokyo, Japan (co-located with ISCA 2025)

Welcome to OSCAR 2025!

- *The goal of OSCAR is to bring together a community of researchers from academia, industry and government labs who are interested in in developing and sharing open-source hardware and software for the design of next-generation computer systems*

In Summary: ESP for Open-Source Hardware

- We contribute ESP to the OSH community in order to support the realization of
  - more scalable architectures for SoCs that integrate
  - more heterogeneous components, thanks to a
  - more flexible design methodology, which accommodates different specification languages and design flows
- ESP was conceived as a heterogeneous integration platform from the start and tested through years of teaching at Columbia University
- We invite you to use ESP for your projects and to contribute to ESP!



ESP

the open-source SoC platform

esp.cs.columbia.edu

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

Quick Intro to ESP - The Open So...
ESP
The Open-Source SoC Platform

Latest Posts

ESP at ISSCC!

Check out our second chip based on ESP, the open-source SoC platform.

Read more

Published: Mar 16, 2024

Release 2024.1.0

A new GitHub Release

E S P

Thank you from the ESP team!

esp.cs.columbia.edu

github.com/sld-columbia/esp

SLD System Level Design Group

CS @CU Computer Science