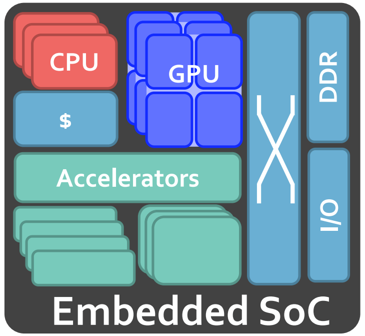
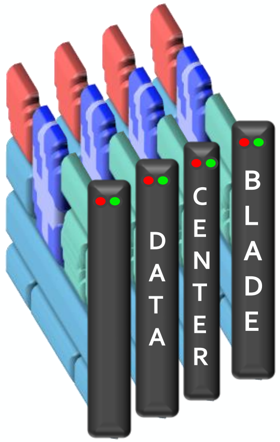
Prototyping RISC-V Based Heterogeneous Systems-on-Chip with the ESP Open-Source Platform

Paolo Mantovani, Giuseppe Di Guglielmo, Davide Giri and Luca P. Carloni

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Why ESP?





Heterogeneous systems are pervasive

Integrating accelerators into a SoC is hard

Doing so in a scalable way is very hard

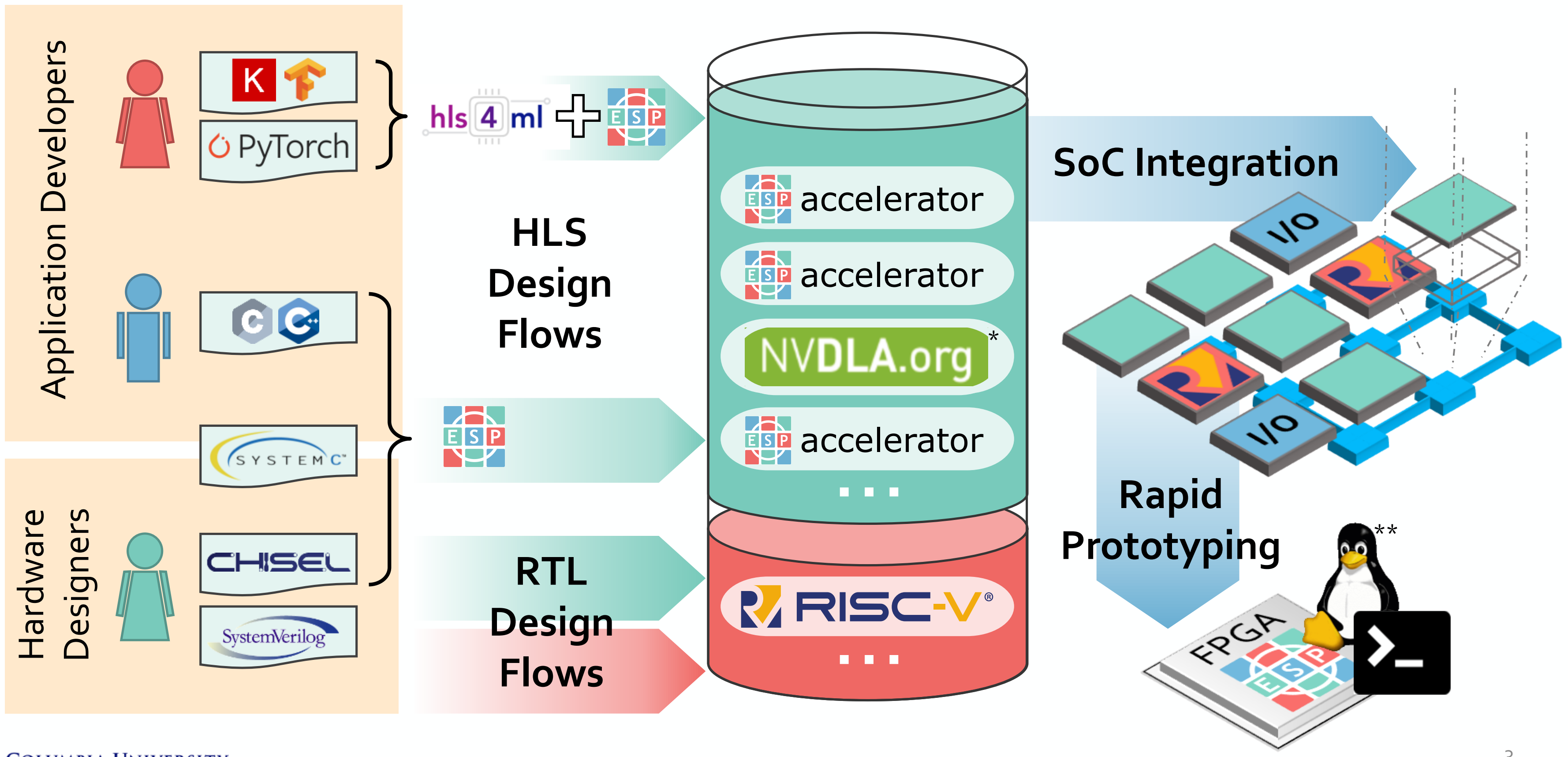
Keeping the system simple to program while doing so is even harder

ESP makes it easy

ESP combines a scalable architecture with a flexible methodology

ESP enables several accelerator design flows
and takes care of the hardware and software integration

ESP Vision: Domain Experts Can Design SoCs

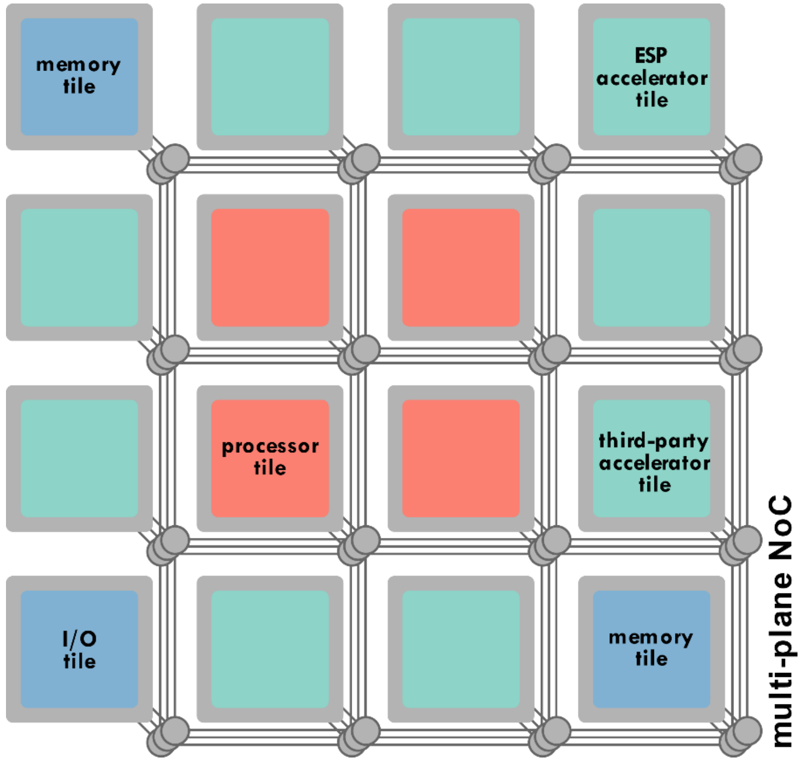


* By Nvidia Corporation
** By lewing@isc.tamu.edu Larry Ewing and The GIMP

ESP Architecture

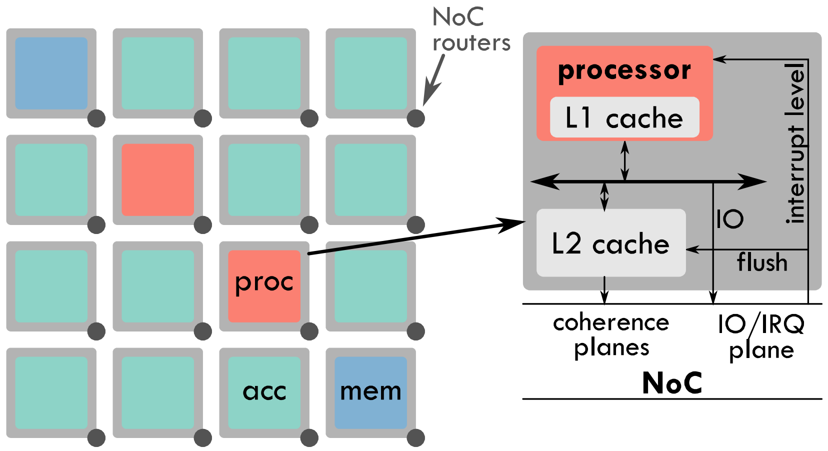
- RISC-V Processors
- Many-Accelerator
- Distributed Memory
- Multi-Plane NoC

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



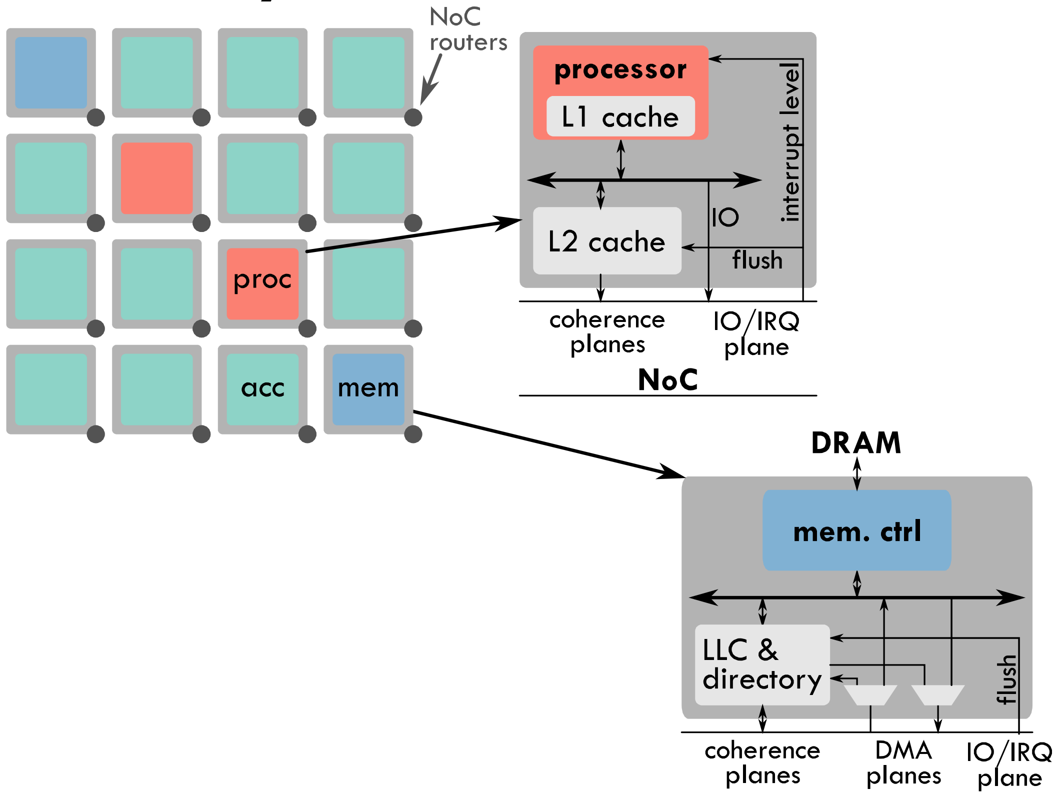
ESP Architecture: Processor Tile

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



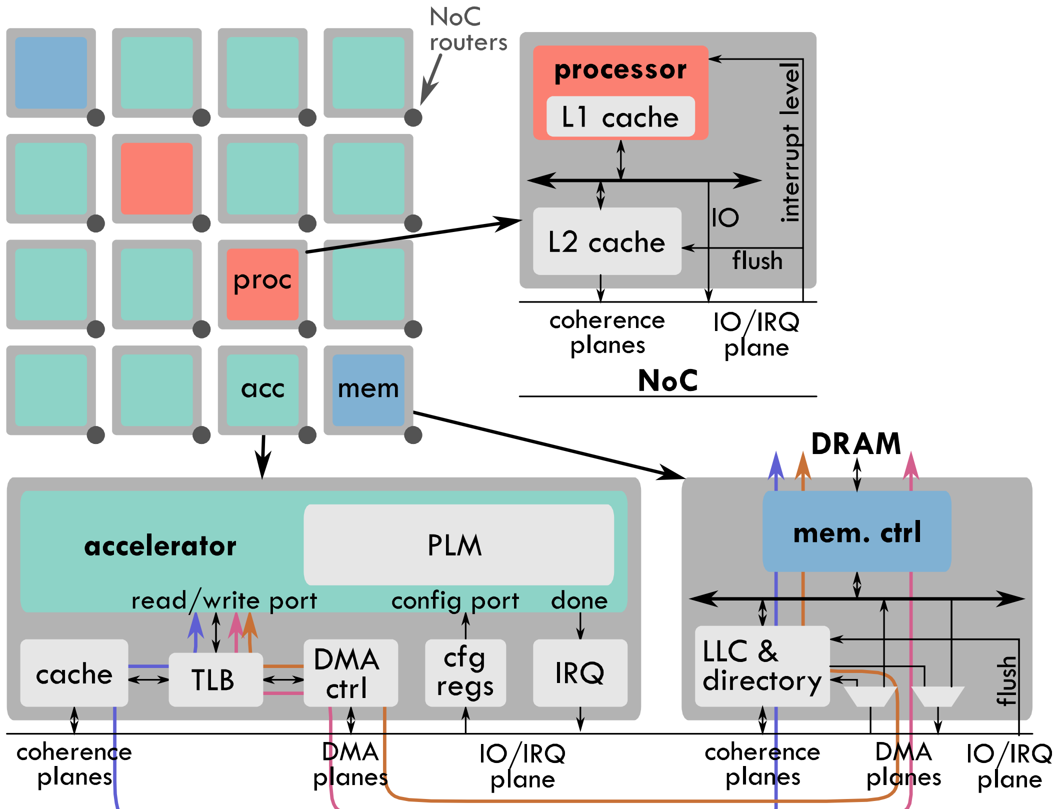
ESP Architecture: Memory Tile

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

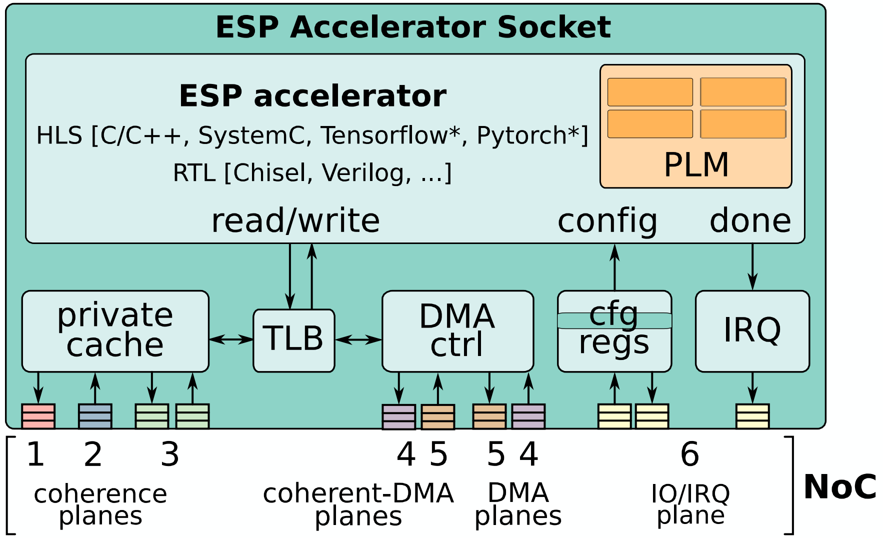


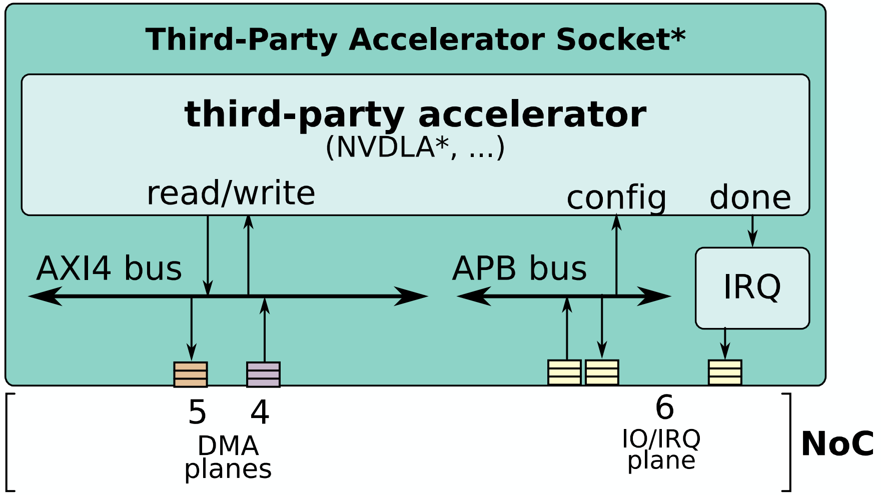
ESP Architecture: Accelerator Tile

- 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



ESP Accelerator Socket





ESP Platform Services

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

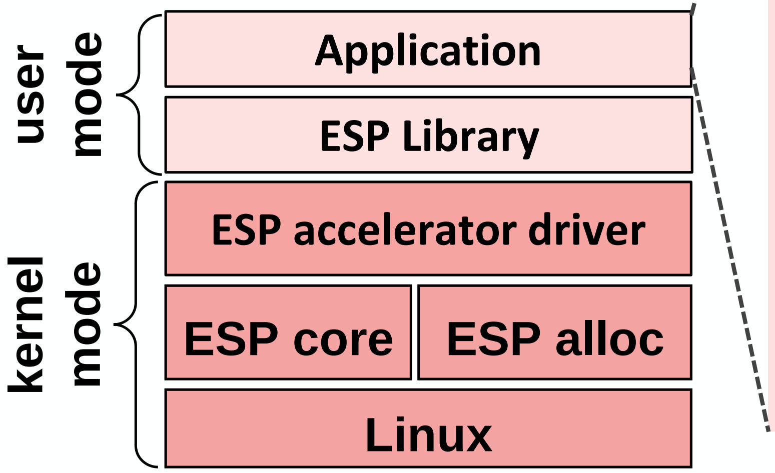
Processor Tile
- Coherence
- I/O and un-cached memory
- Distributed interrupts
- DVFS controller

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



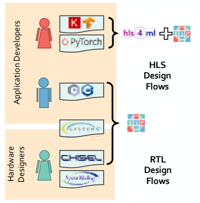
```
/*
 * Example of existing C application
 * with ESP accelerators that replace
 * software kernels 2, 3 and 5
 */
{
  int *buffer = esp_alloc(size);

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

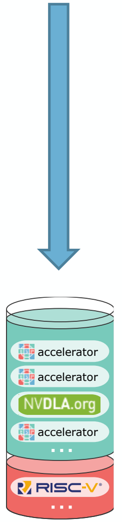
ESP Methodology In Practice

automated
interactive
manual (opt.)
manual

Accelerator Flow

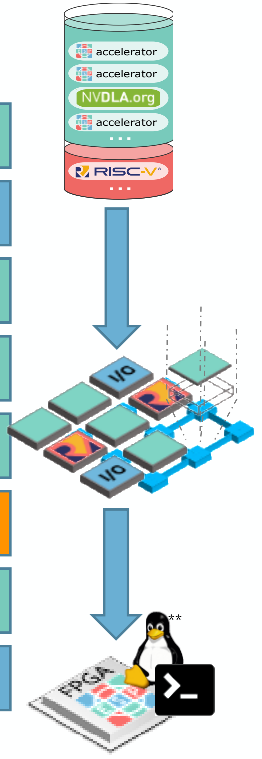
- Generate accelerator
- Specialize accelerator
  * this step is automated
  * for ML applications
- Test behavior
- Generate RTL
- Test RTL
- Optimize RTL





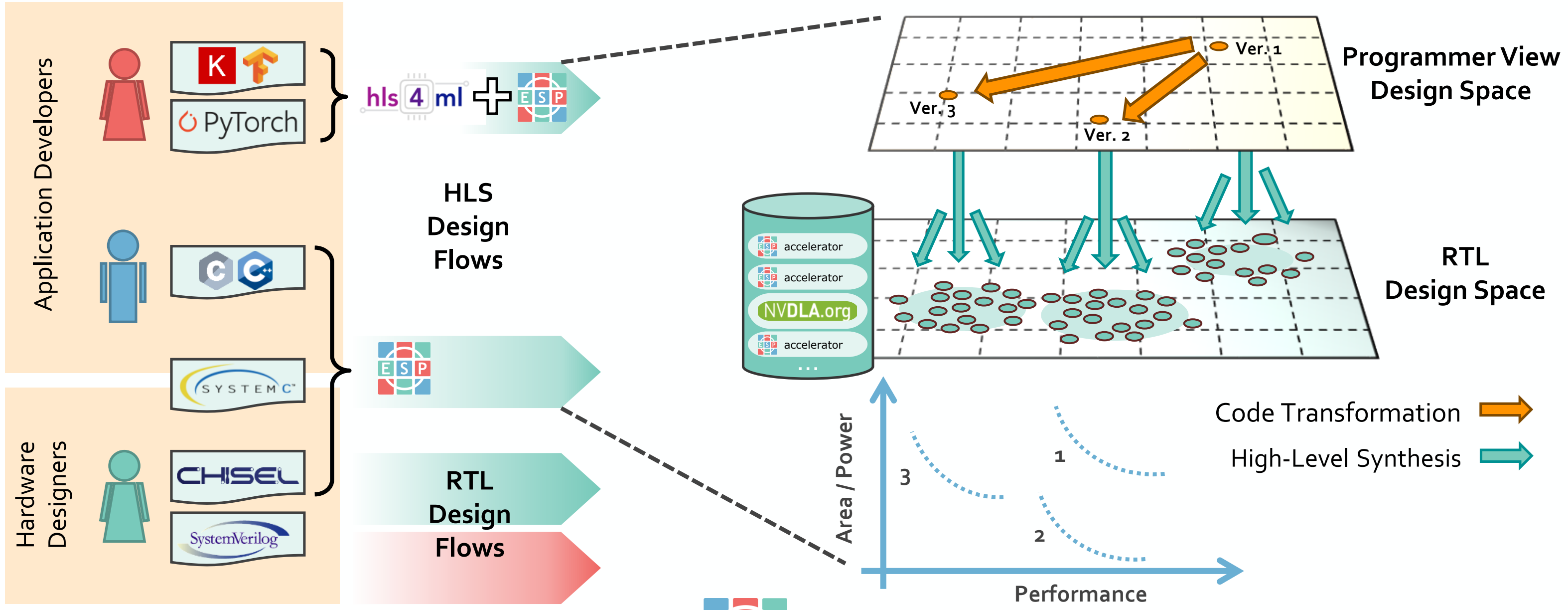
SoC Flow

- Generate sockets
- Configure RISC-V SoC
- Compile bare-metal
- Simulate system
- Implement for FGPA
- Configure runtime
- Compile Linux
- Deploy prototype

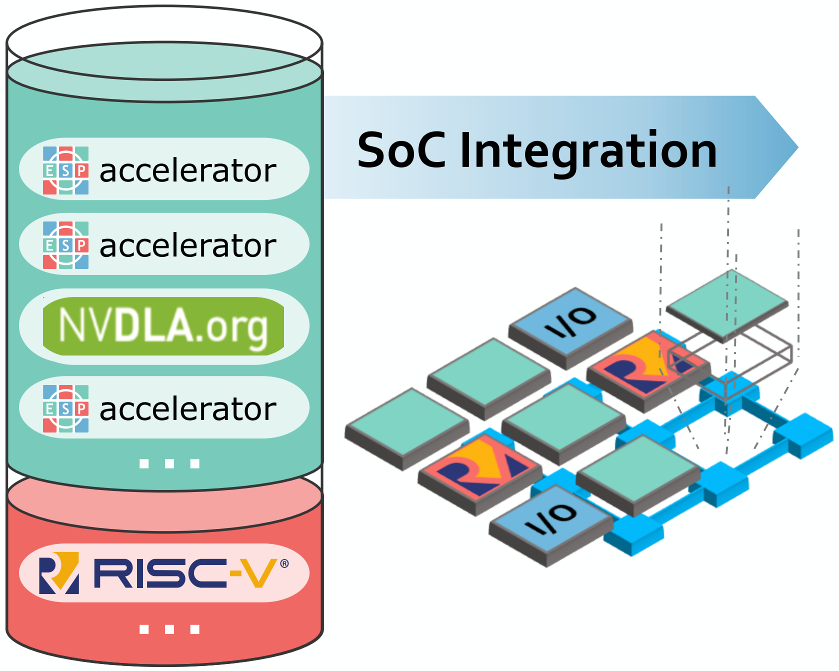


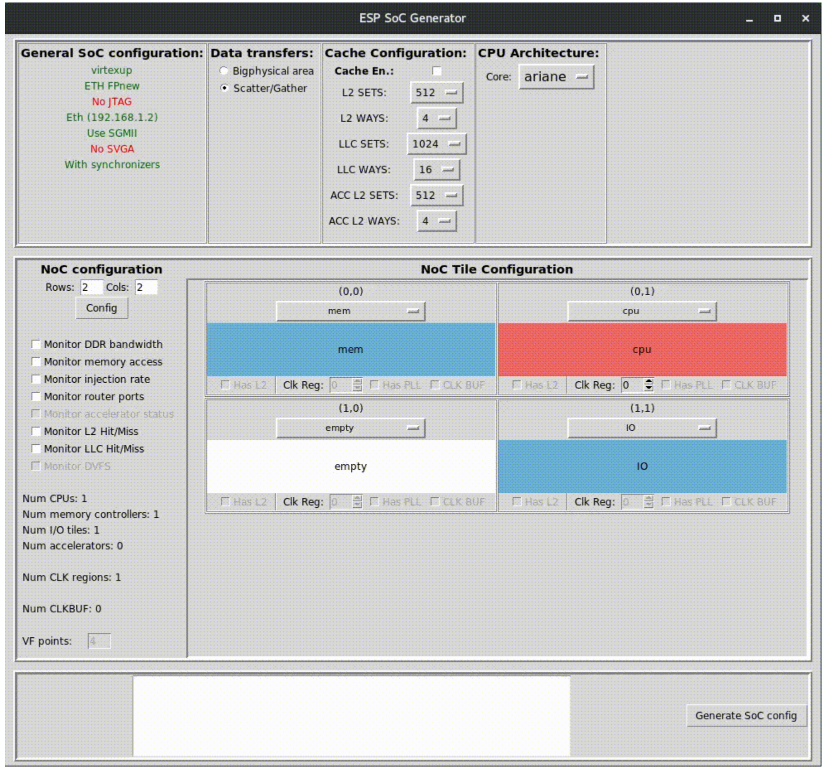
ESP Accelerator Flow

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



ESP Interactive SoC Flow





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
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

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

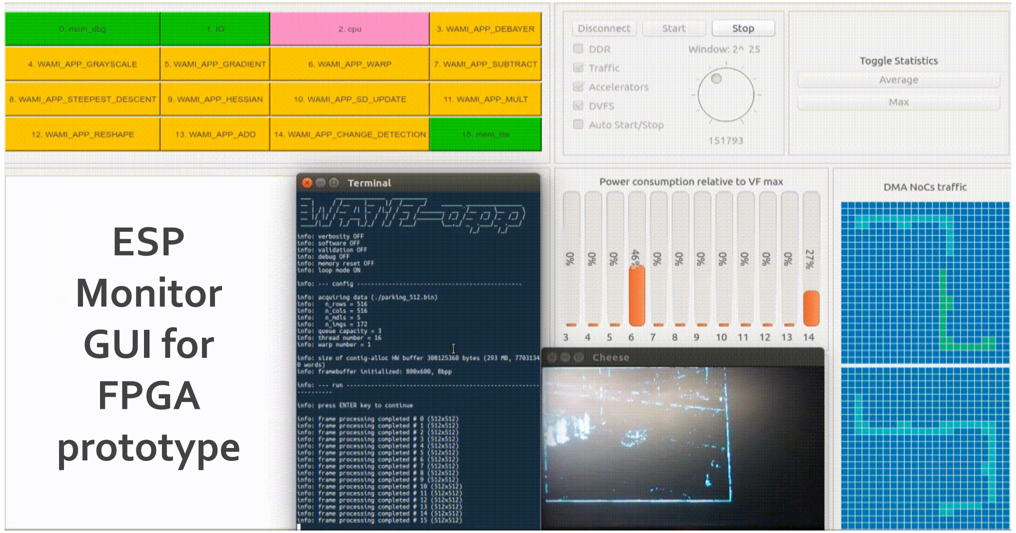
NoC Tile Configuration

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

Generate SoC config

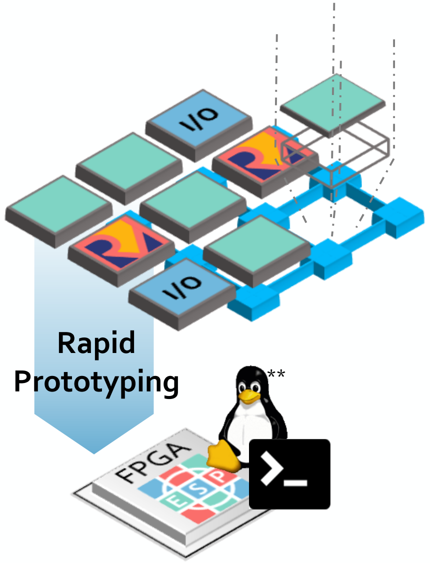
Columbia University in the City of New York

ESP: A Flexible Platform for Open-Source Hardware



0. mem_dbg	1. IO	2. cpu	3. WAMI_APP_DEBAYER
4. WAMI_APP_GRAYSCALE	5. WAMI_APP_GRADIENT	6. WAMI_APP_WARP	7. WAMI_APP_SUBTRACT
8. WAMI_APP_STEEPEST_DESCENT	9. WAMI_APP_HESSIAN	10. WAMI_APP_SD_UPDATE	11. WAMI_APP_MULT
12. WAMI_APP_RESHAPE	13. WAMI_APP_ADD	14. WAMI_APP_CHANGE_DETECTION	15. mem_lite

```
info: verbosity OFF
info: software OFF
info: validation OFF
info: debug OFF
info: memory reset OFF
info: loop mode ON

info: --- config ---------------------------------------------

info: acquiring data (./parking_512.bin)
info:   n_rows = 516
info:   n_cols = 516
info:   n_mdls = 5
info:   n_imgs = 172
info: queue capacity = 3
info: thread number = 16
info: warp number = 1

info: size of contig-alloc HW buffer 308125360 bytes (293 MB, 7703134
0 words)
info: framebuffer initialized: 800x600, 8bpp

info: --- run ---------------------------------------------------
----------

info: press ENTER key to continue

info: frame processing completed # 0 (512x512)
info: frame processing completed # 1 (512x512)
info: frame processing completed # 2 (512x512)
info: frame processing completed # 3 (512x512)
info: frame processing completed # 4 (512x512)
info: frame processing completed # 5 (512x512)
info: frame processing completed # 6 (512x512)
info: frame processing completed # 7 (512x512)
info: frame processing completed # 8 (512x512)
info: frame processing completed # 9 (512x512)
info: frame processing completed # 10 (512x512)
info: frame processing completed # 11 (512x512)
info: frame processing completed # 12 (512x512)
info: frame processing completed # 13 (512x512)
info: frame processing completed # 14 (512x512)
info: frame processing completed # 15 (512x512)
```



We hope that **ESP** will serve the **OSH** community as a **Platform** to develop software for RISC-V and accelerators for any application domain

Thank you from the ESP team!

https://esp.cs.columbia.edu

https://github.com/sld-columbia/esp

System Level Design Group

COMPUTER SCIENCE

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK