Towards Generalized On-Chip Communication for Programmable Accelerators in Heterogeneous Architectures

Joseph Zuckerman[1]

John-David Wellman[2], Ajay Vanamali[1], Manish Shankar[1], Gabriele Tombesi[1],

Karthik Swaminathan[2], Kevin Lee[1], Mohit Kapur[2], Robert Philhower[2],

Pradip Bose[2], Luca P. Carloni[1]

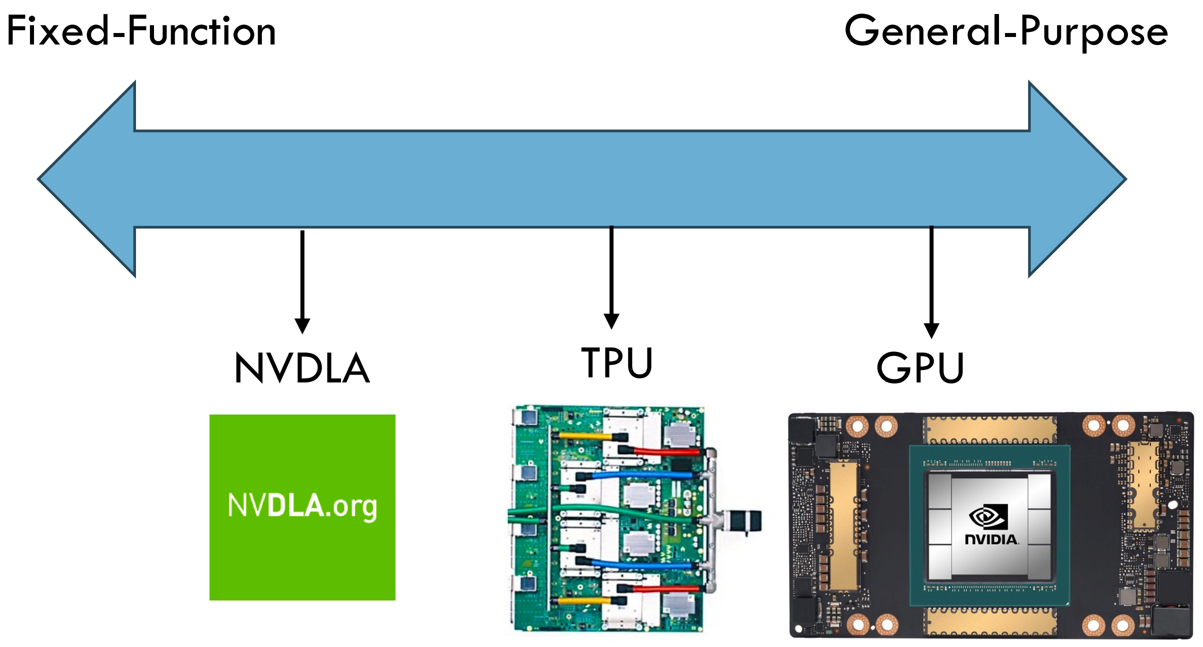
Columbia University[1], **IBM Research**[2]

DOSSA Workshop '24
6/30/24

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

CS@CU COMPUTER SCIENCE

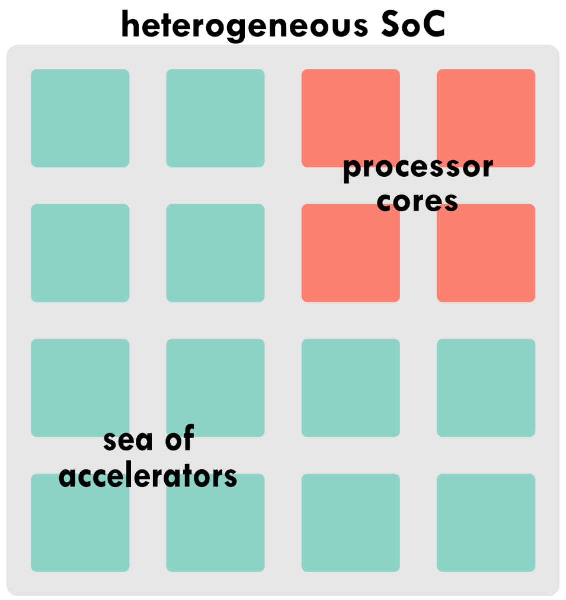
Accelerator Specialization



- *Programmable accelerators* are optimized for an application domain, but execute instructions
  - Generated by a compiler, another tool, or hand written
- Balance specialization with programmability
  - Have made them widely popular in industry and academia
- Mostly studied as monolithic entities

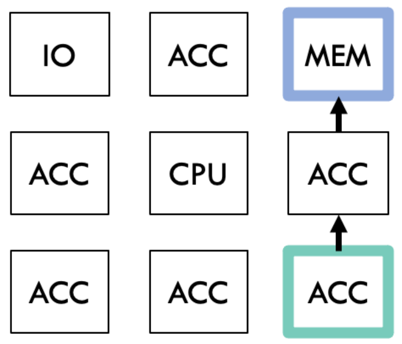
The Age of Heterogeneous Computing

- SoCs are growing increasingly heterogeneous across various computing domains
  - CPUs, GPUs, accelerators, I/O peripherals, sensors…
  - Possibly multiple instances of complex, programmable accelerators
- Heterogeneity increases engineering effort
  - Capabilities of new generations limited by team size
  - Biggest challenges are in system-integration

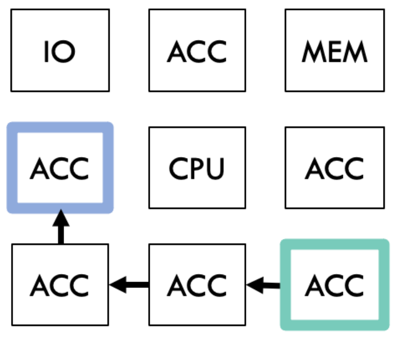


Challenges and Opportunities for SoC Design

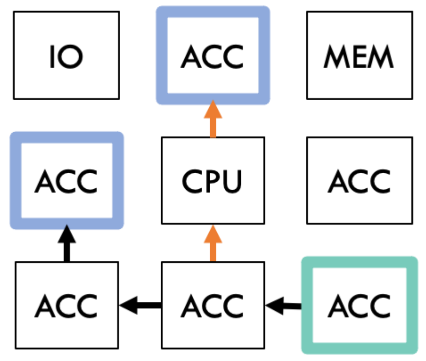
- Integration of programmable accelerators in heterogeneous architectures is not well-studied
- Programmability at the accelerator level could benefit from flexibility at the system level
- Complex workloads might feature multiple data-dependency patterns and require synchronization
- Can we develop *platform-level services* that provide these capabilities transparently to the design of the accelerator?



a) Memory Access



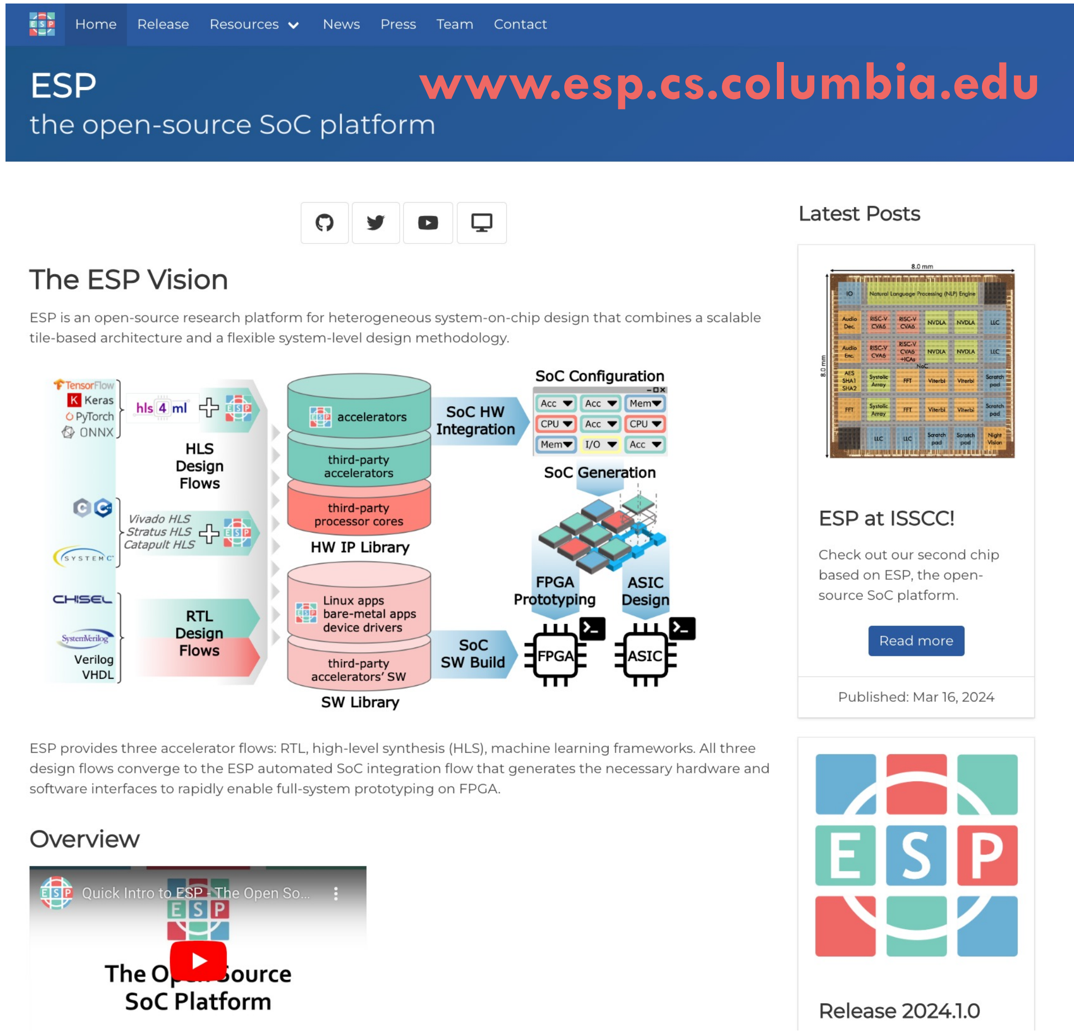
b) Accelerator P2P



c) Accelerator Multicast

ESP : An Open-Source Platform for SoC Research

- We propose a set of enhancements to an SoC architecture to support the integration of programmable accelerators
  - Flexible point-to-point accelerator
  - Multicast communication supported by the NoC
  - Intra-accelerator communication
  - A simple accelerator interface to manage communication modes
  - Accelerator ISA extensions for on-chip communication
- Implement these enhancements in ESP, but the principles can apply to other architectures



ESP
www.esp.cs.columbia.edu
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

Latest Posts

ESP at ISSCC!

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

Published: Mar 16, 2024

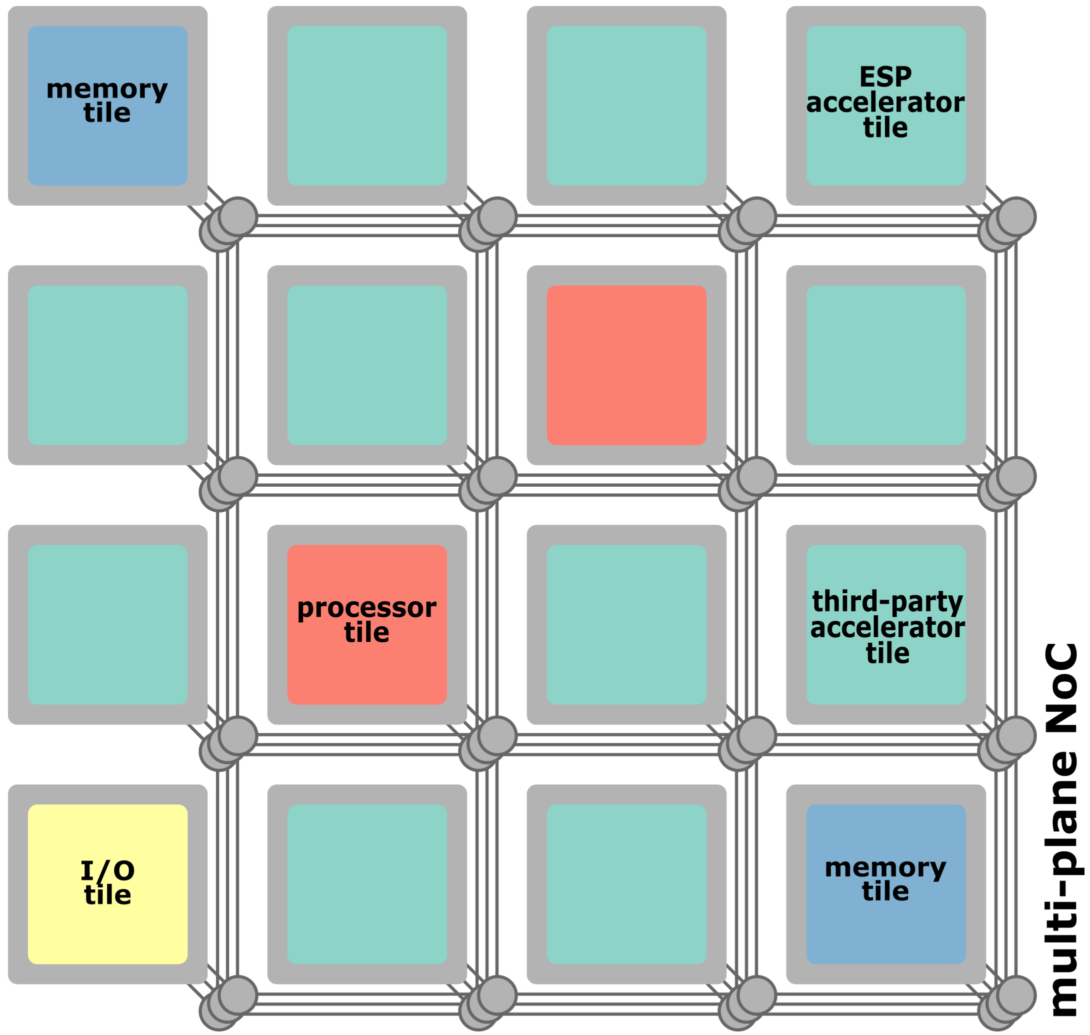
Release 2024.1.0

ESP Background

ESP Architecture

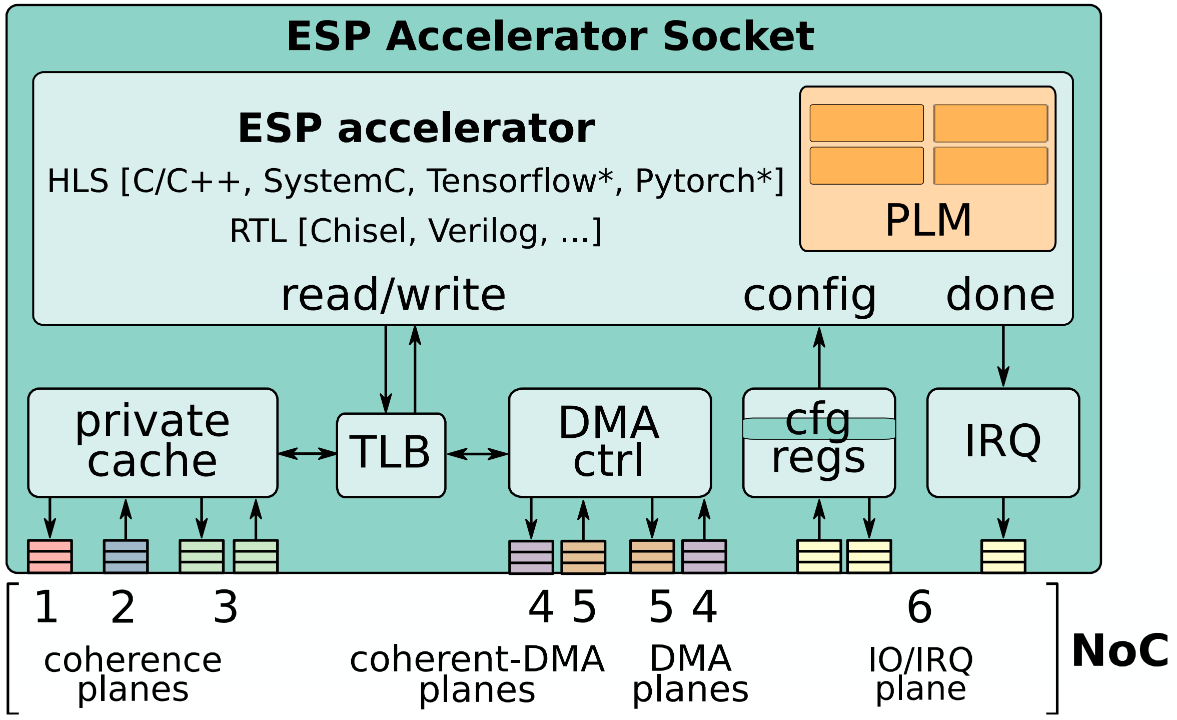
- Multi-Plane NoC
- Many-Accelerator
- RISC-V Cores
- Distributed Memory

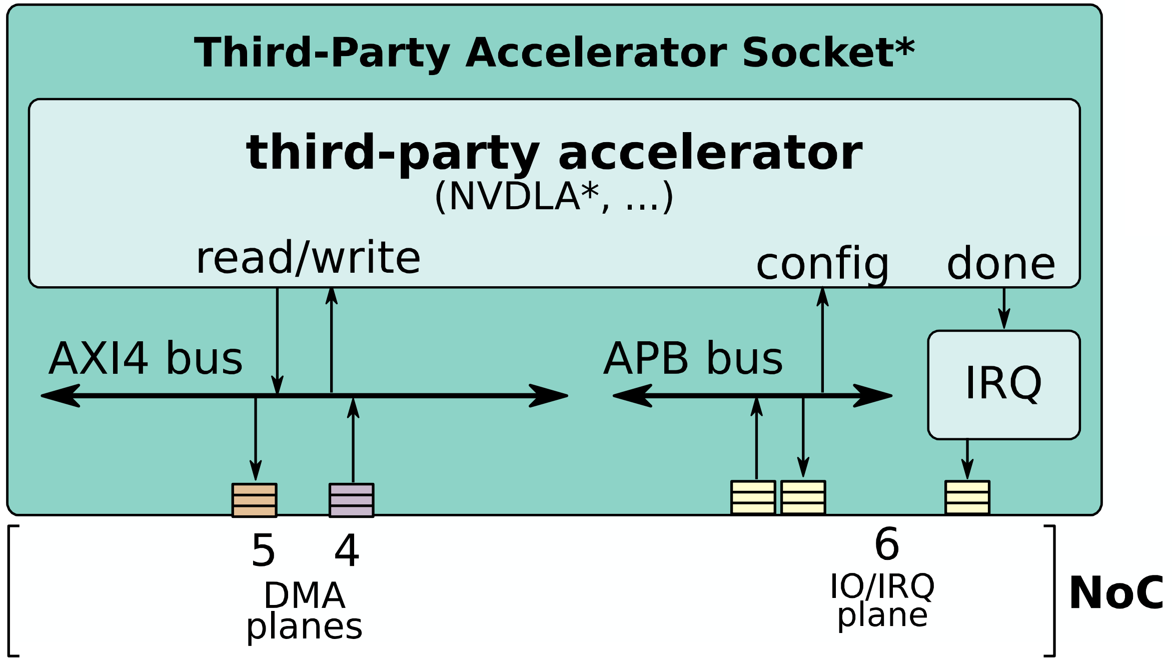
The ESP architecture implements a **distributed** system, which is **scalable**, **modular** and **heterogeneous**, giving processors and accelerators similar weight in the SoC [3]



[3] Mantovani, ICCAD ‘20

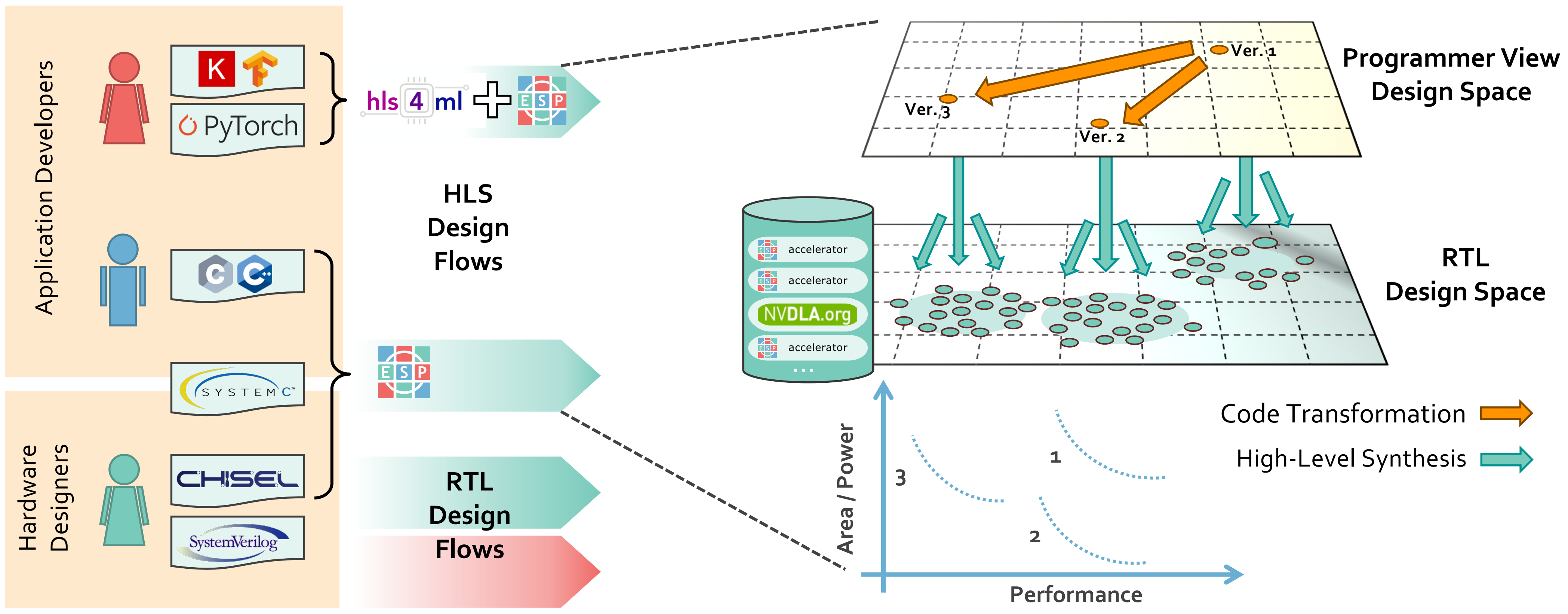
ESP Accelerator Socket



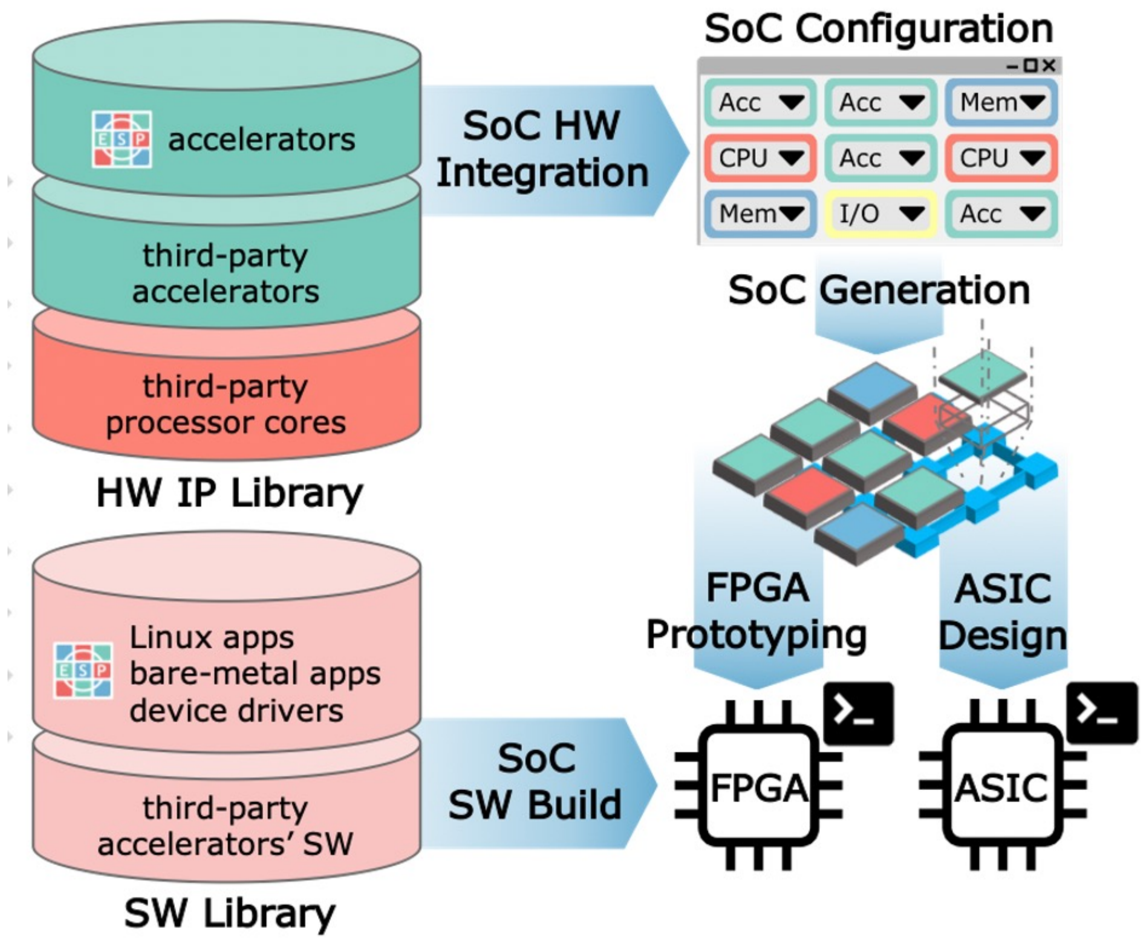


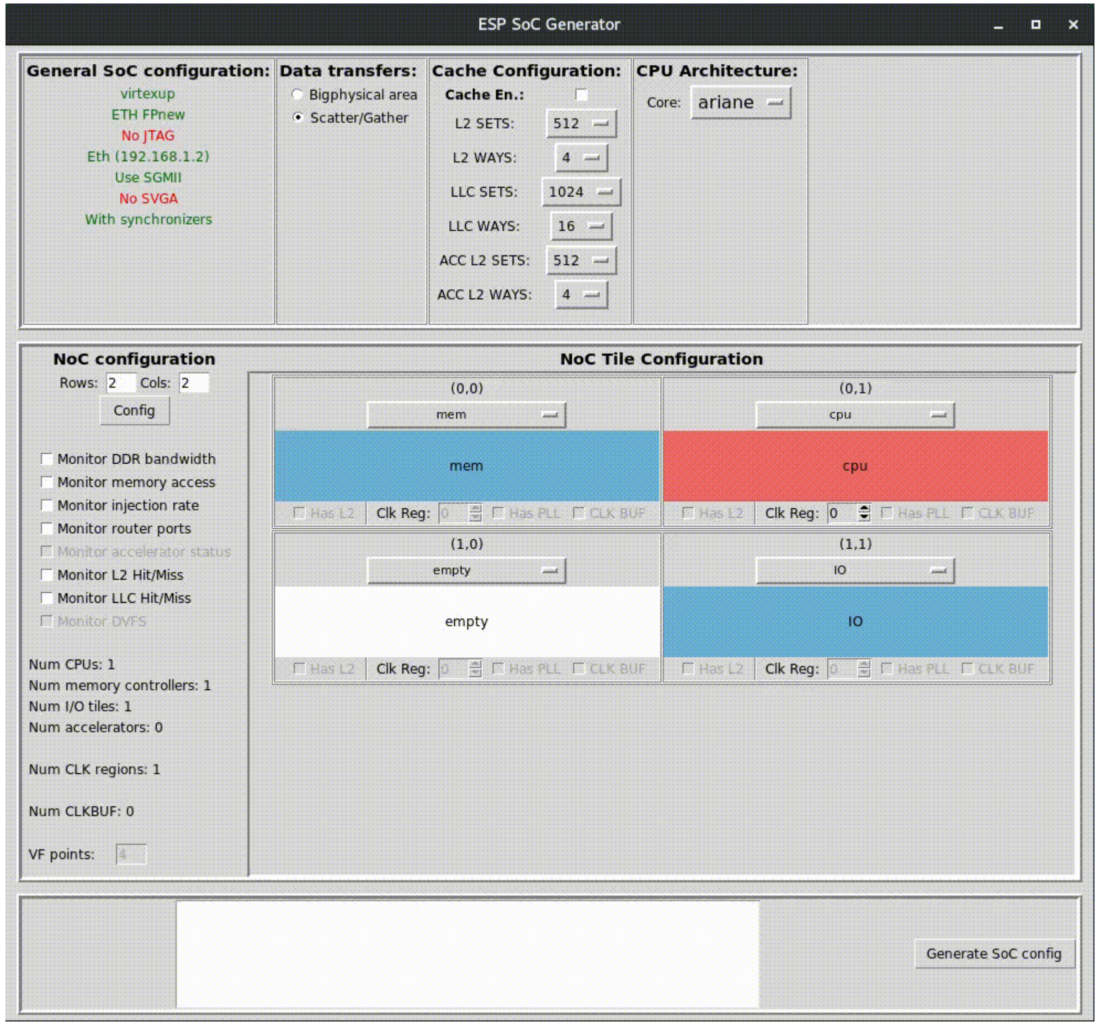
ESP Accelerator Flow

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



ESP SoC Flow





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

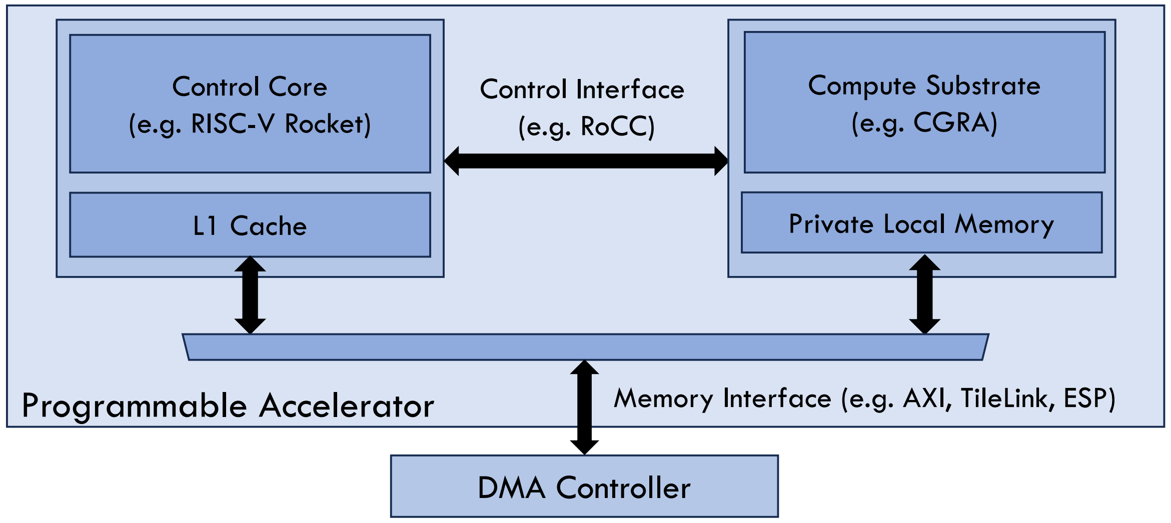
(0,0)	(0,1)
mem	cpu
mem	cpu
☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF	☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF
(1,0)	**(1,1)**
empty	IO
empty	IO
☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF	☐ Has L2 Clk Reg: 0 ☐ Has PLL ☐ CLK BUF

Generate SoC config

Proposed Architecture

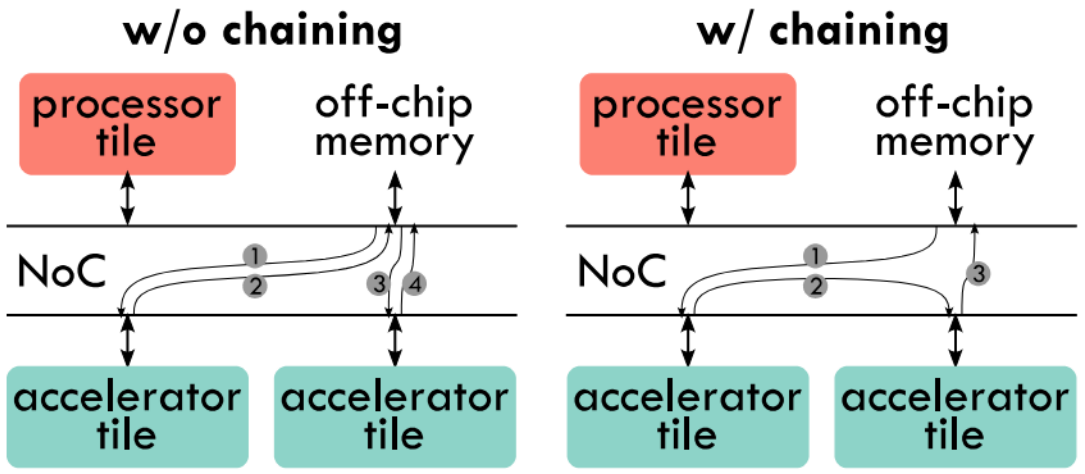
Programmable Accelerator Model

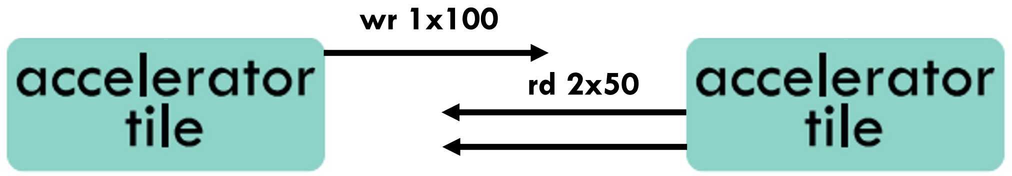
- Loosely-coupled from the host core
  - Won't boot an OS, etc.
- Executes coarse-grained tasks on large workloads
  - Needs a DMA interface to bring in large amounts of data from the SoC
- Highly-customized datapath and private local memory
- Can rely on a lightweight core for instruction fetch, decode, commit, etc.
  - E.g. RISC-V Rocket Core + RoCC interface
  - ISA extensions for leveraging the datapath

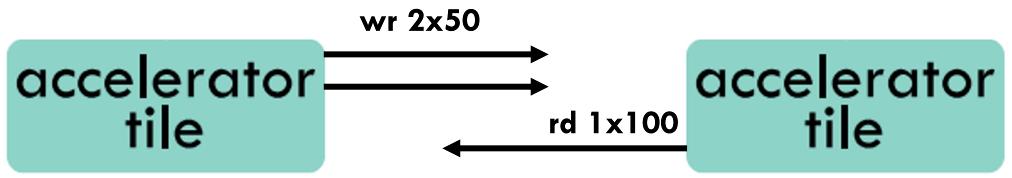


Flexible Point-to-Point Accelerator Communication

- Point-to-point (P2P) accelerator communication improves performance of applications with data dependencies
  - Avoids round-trip to memory
  - Pipelining with fine-grained synchronization
- Implemented with a *pull-based model*
  - Consumer initiates each exchange to satisfy the *consumption assumption*
- Previously, data access patterns must be exactly the same
  - Relaxed this by adding a *length*
  - Total amount of data must match

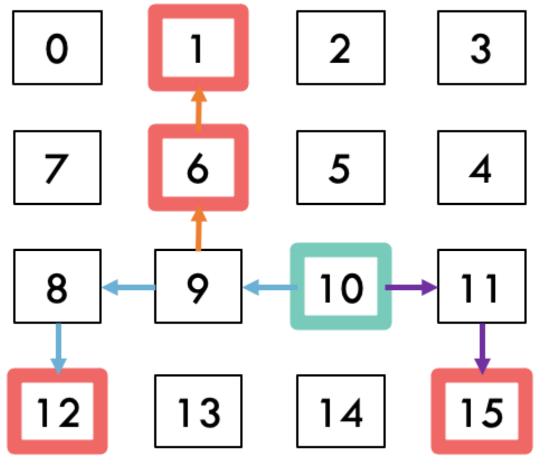




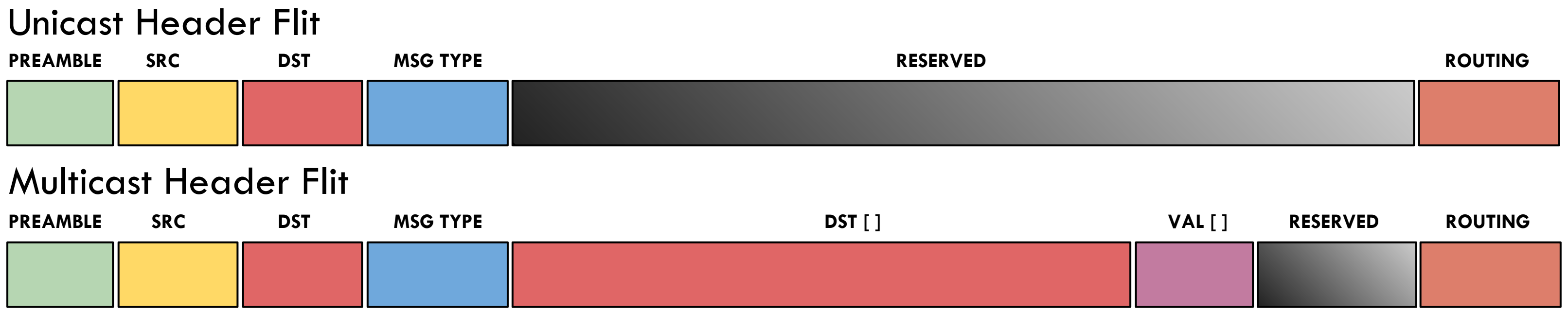


Multicast NoC

- Support multiple-consumer dependencies with a single NoC message
- With minimal modifications to the ESP NoC
- Header *flit* modified to carry an array of destinations
  - Max # of consumers limited by NoC bitwidth
- Lookahead routing logic replicated for each destination

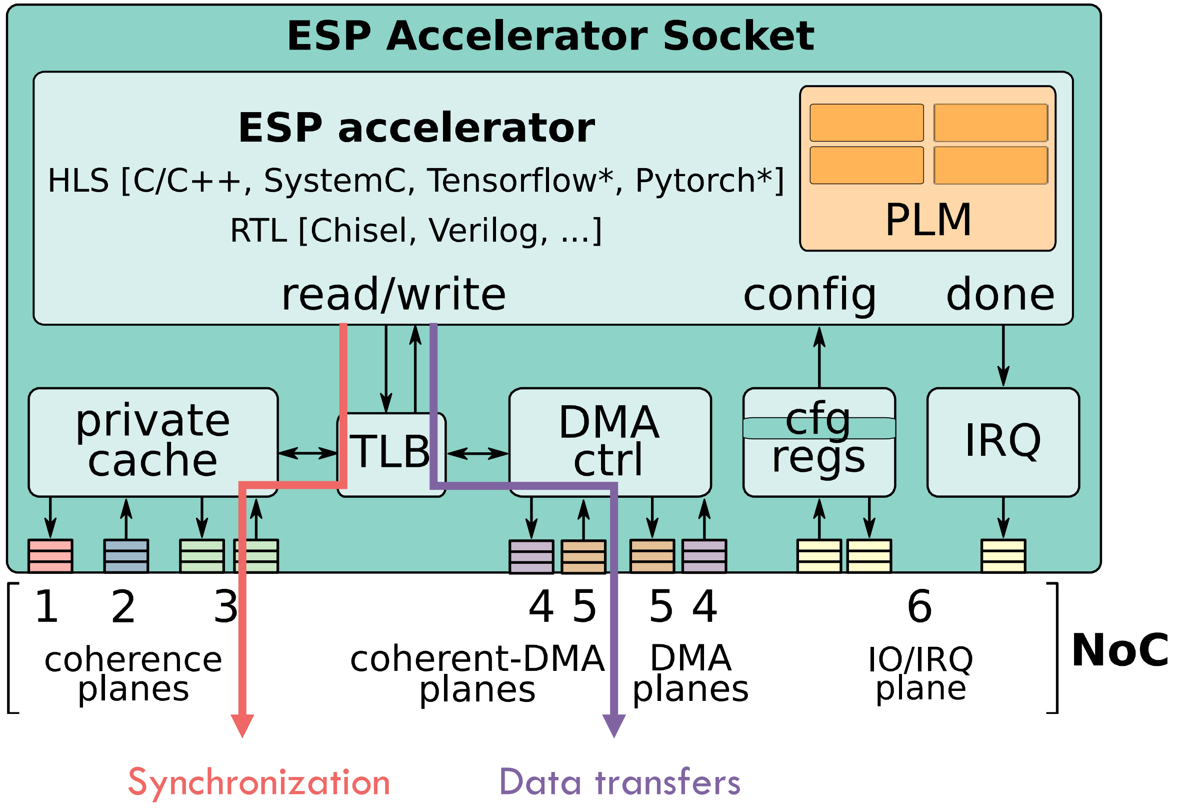


- Integrated with ESP's P2P capabilities



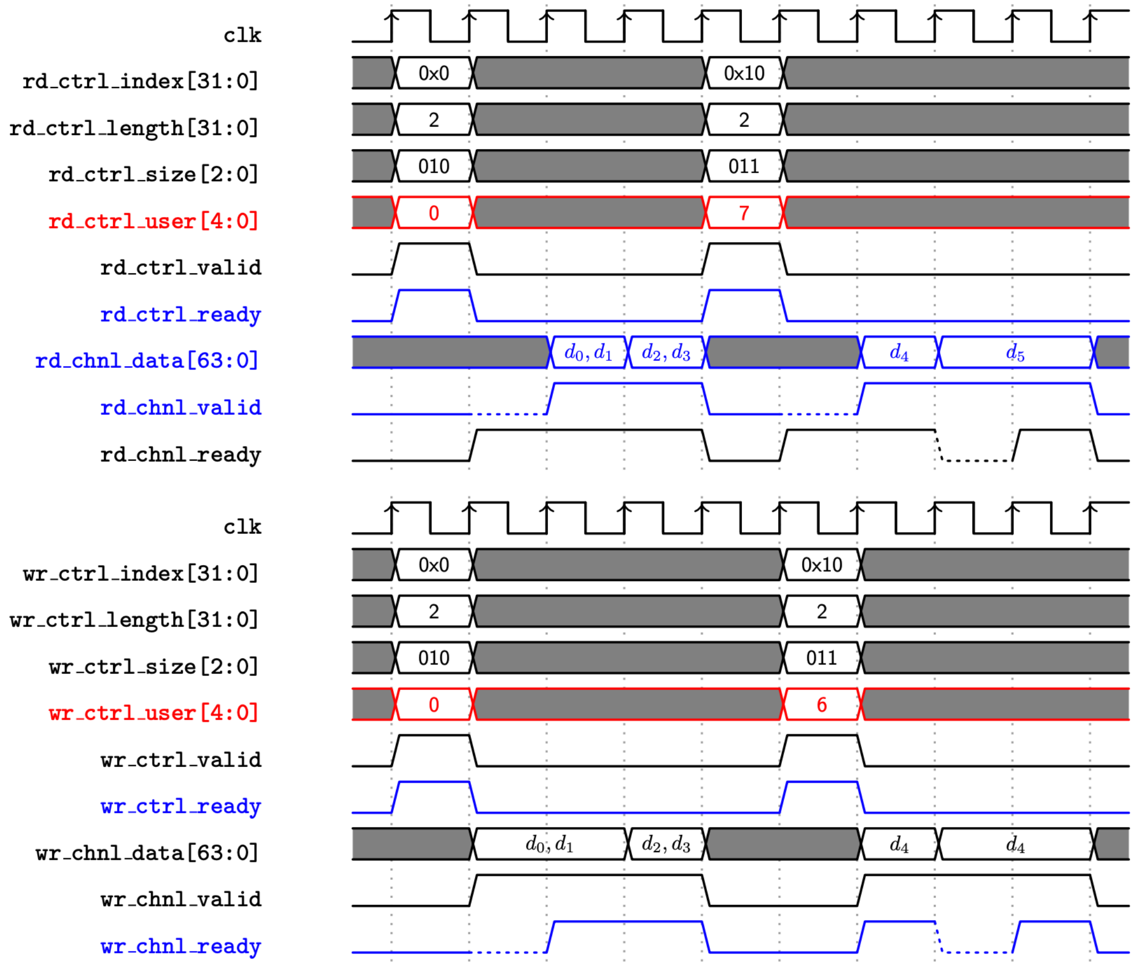
Accelerator Synchronization

- Programmable accelerators might need to *synchronize* with each other throughout their execution
  - E.g. ensuring they have both arrived to a pre-defined point in the workload before exchanging data
- Requires *coherence* between accelerators
  - ESP supports coherent accelerators, but coherence must be used for all data transfers → inefficient
- Enable *dynamic switching* between DMA for data transfers and coherence for synchronization



Accelerator Interface + ISA Extensions

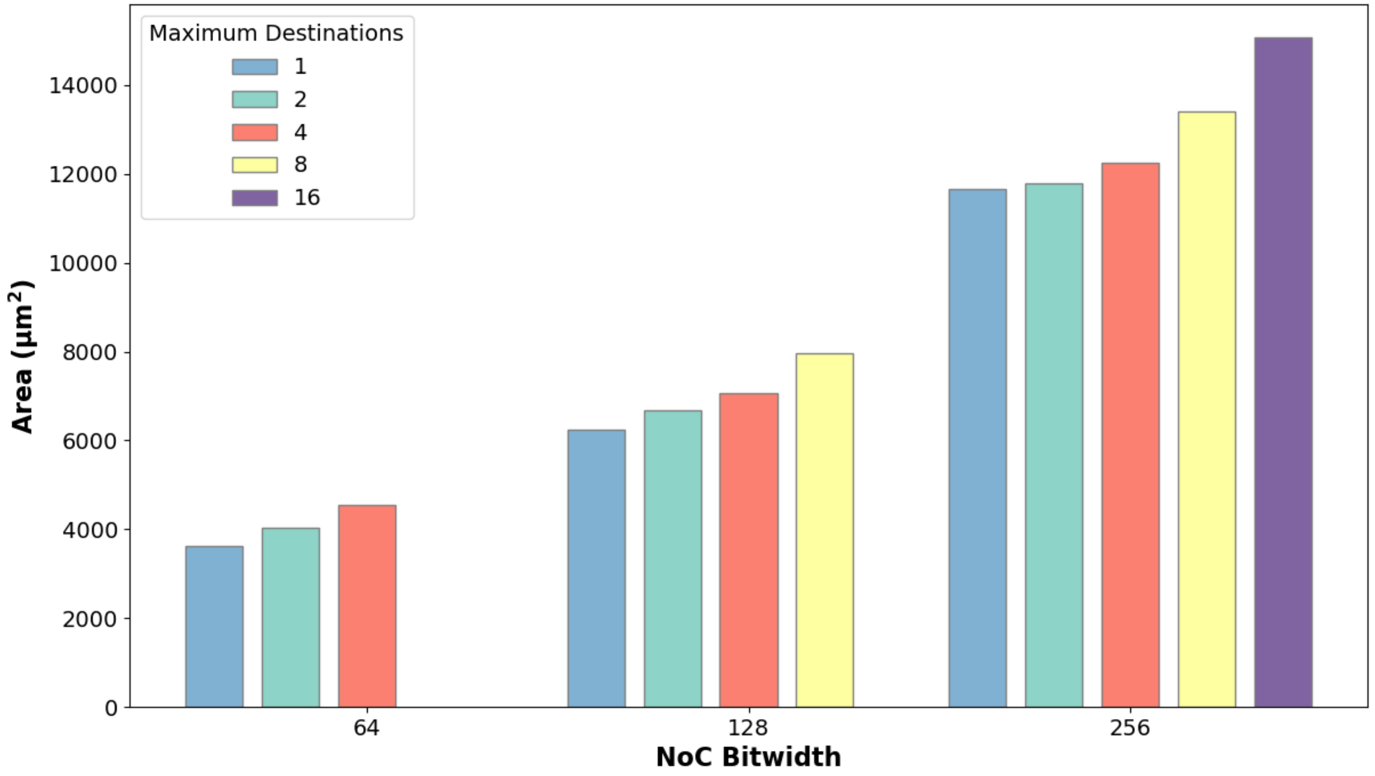
- Give control of the mode of each transaction to the accelerator itself
  - Just one additional signal on the read and write control interfaces
- For reads, the *user* field specifies the source of the data
  - 0 for memory
  - 1 to N-1 for other accelerators
- For writes, the *user* field specifes the # of consumers of the data
  - 0 for memory
  - 1 for regular P2P
  - 2 to N-1 for a multicast P2P
- ISA Extensions
  - Initiate DMA (IDMA) – encodes the index, length, size, and user as expected by ESP
  - Check DMA (CDMA) – query if DMA has finished



Results

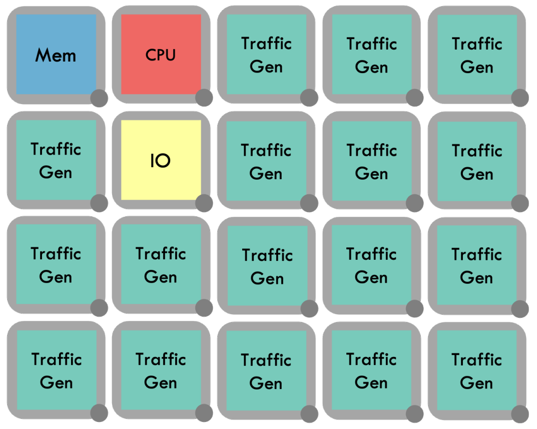
Results: Multicast Router Area

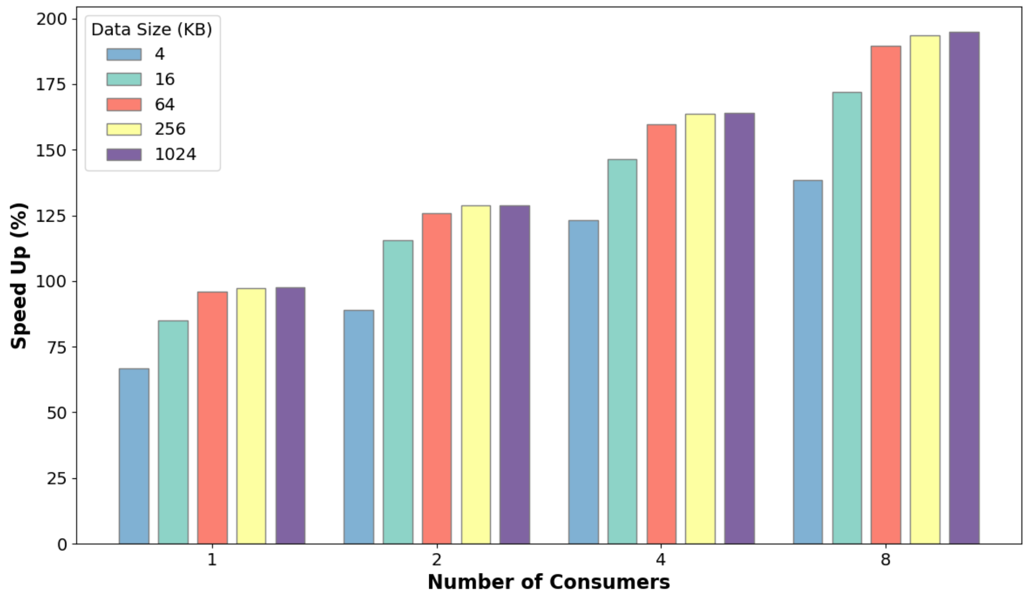
- Synthesized in Cadence Genus targeting a 12nm technology
- Varying the bitwidth of the NoC and the maximum number of supported multicast destinations



Results: Multicast Performance

- Developed a 5x4 SoC prototype with 17 traffic generator accelerators
- 256-bit NoC allows us to test multicast to 16 consumers
- Deployed on a Xilinx VCU128 FPGA at 78MHz
- Baremetal ”workload” that creates a 1 to N data dependency
- Compared multicast performance to communication through shared memory





Thank you from the ESP team!

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

Towards Generalized On-Chip Communication for Programmable Accelerators in Heterogeneous Architectures

Joseph Zuckerman[1]

John-David Wellman[2], Ajay Vanamali[1], Manish Shankar[1], Gabriele Tombesi[1],

Karthik Swaminathan[2], Kevin Lee[1], Mohit Kapur[2], Robert Philhower[2],

Pradip Bose[2], Luca P. Carloni[1]

Columbia University[1], **IBM Research**[2]

DOSSA 2024

Columbia University in the City of New York

Computer Science