ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ICCD 2024 IEEE International Conference on Computer Design
November 18-20 - Milan, Italy

HeroSDK: Streamlining Heterogeneous RISC-V Accelerated Computing From Embedded To High-Performance Systems

International Conference on Computer Design (ICCD)
Milan, Italy, 19-11-2024

Cyril Koenig[1] cykoenig@iis.ee.ethz.ch
Björn Forsberg[2] bjorn.forsberg@ri.se
Luca Benini[1,3] lbenini@iis.ee.ethz.ch

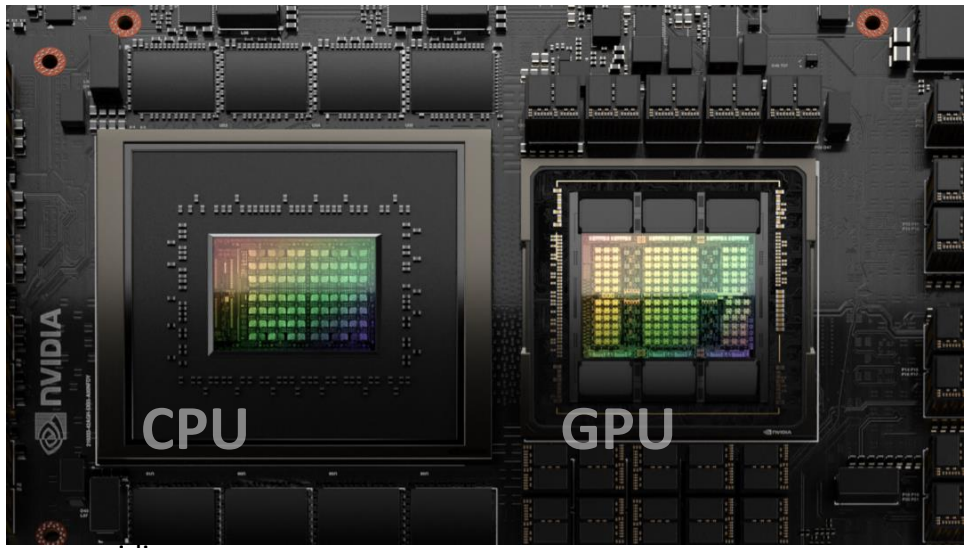
[1] IIS, ETH Zürich, Switzerland
[2] RISE, Sweden
[3] DEI, University of Bologna, Italy

PULP Platform
Open Source Hardware, the way it should be!

@pulp_platform
pulp-platform.org
youtube.com/pulp_platform

Where do we find heterogeneous systems?

www.apple.com



www.nvidia.com

- **Embedded Heterogeneous SoCs (HeSoC)**
- **Heterogeneous HPC platforms**
- **9 of the 10 most powerful supercomputers are heterogeneous**

TOP 500
The List.

This heterogeneity brings challenges

- **Benefits**
  - Energy efficiency
  - Peak performance

- **Challenges**
  - Synchronization
  - Communication
  - Programmability
  - ISA and ABI differences
  - Portability

These challenges are reduced with RISC-V

- **Benefits**
  - Energy efficiency
  - Peak performance
  - Large design space exploration of accelerators *(open source ISA)*
- **Challenges**
  - Synchronization
  - Communication
  - Programmability
  - ISA ~~and ABI~~ differences
  - Portability simplified *(same memory model, same base ISA, same non-ISA IP specs)*

RISC-V heterogeneous platforms are already there

	OpenPiton [1]	Cohort [2]	Vortex [3]	HeroSDK (Our work)
Host	CVA6	CVA6	x86	**CVA6/SG2042**
Accelerator	MIAOW GPU	Dataflow accelerators	Vortex GPGPU	**Snitch/Spatz clusters**
On-chip offload	NoC	Crossbar	*None*	**Crossbar**
Off-chip offload	Chip Bridge	*None*	PCIe	**PCIe**
Programming	Custom	C++ Boost library	OpenCL	**OpenMP**

[1] J. Balkind et al., "OpenPiton: An Open Source Hardware Platform for Your Research", IEEE MICRO 40 (2020)
[2] T. Wei et al., "Cohort: Software-Oriented Acceleration for Heterogeneous SoCs", ASPLOS'20 (2020)
[3] B. Tine et al., "Vortex: Extending the RISC-V ISA for GPGPU and 3D-Graphics", MICRO '21 (2021)

High level heterogeneous programming for both on-chip and off-chip accelerators

HeroSDK: Contributions

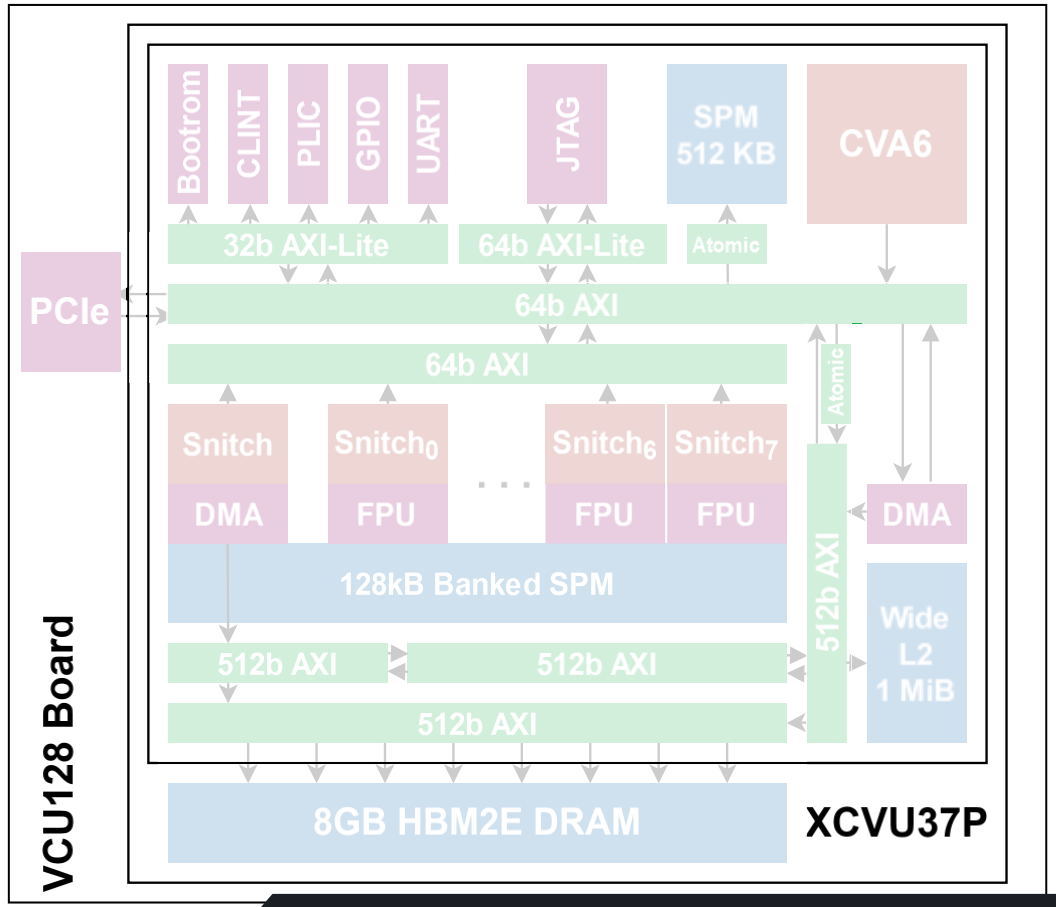
- A generic **OpenMP offloading runtime for RV based HeSoC** targeting **portability** across RV-accelerators.
- Its implementation on **two emulated HeSoC running Linux**.
- A Linux module to **export this framework from HeSoC to PCIe based accelerator cards with 100% code reuse**.
- We demonstrate matrix-vector multiplication with **5.9× speedup** with **19% offload-to-work ratio** in plain C-code with embedded accelerator.

Original Snitch-based demonstrator

- **Snitch embedded platform**
  - RV64G single core host
  - 8-core RV32IMAFD cluster
  - Cluster scratchpad and DMA
- **Snitch accelerator card**
  - 8-cores RV32IMAFD cluster
  - Cluster scratchpad and DMA
  - PCIe endpoint

+ SG2042 [1] RISC-V CPU

- RV64G 64-cores host



www.github.com/pulp-platform/occamy

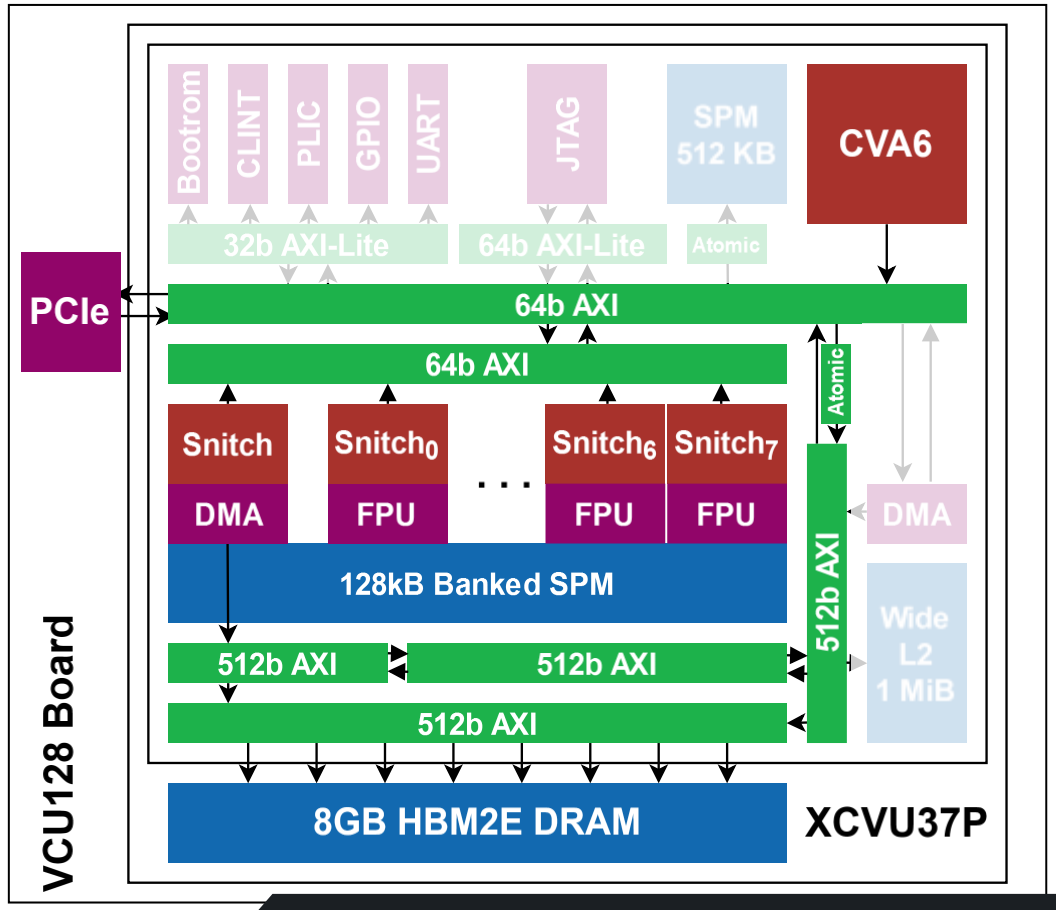
[1] https://en.sophgo.com

Original Snitch-based demonstrator

- **Snitch embedded platform**
  - RV64G single core host
  - 8-core RV32IMAFD cluster
  - Cluster scratchpad and DMA
- **Snitch accelerator card**
  - 8-cores RV32IMAFD cluster
  - Cluster scratchpad and DMA
  - PCIe endpoint

+ SG2042 [1] RISC-V CPU

- RV64G 64-cores host



www.github.com/pulp-platform/occamy

[1] https://en.sophgo.com

Four demonstrator platforms

Embedded

RV32IMAFD

- **Snitch embedded platform**
  - RV64G single core host
  - 8-core RV32IMAFD cluster
  - Cluster scratchpad and DMA
- **Snitch accelerator card**
  - 8-core RV32IMAFD cluster
  - Cluster scratchpad and DMA
  - PCIe endpoint

RV32IMAFDV*

- **Spatz embedded platform**
  - RV64G single core host
  - 2-core RV32IMAFD**V*** cluster
  - Cluster scratchpad and DMA
- **Spatz accelerator card**
  - 2-core RV32IMAFD**V** cluster
  - Cluster scratchpad and DMA
  - PCIe endpoint

High performance

The HeroSDK software stack

Your application using OpenMP

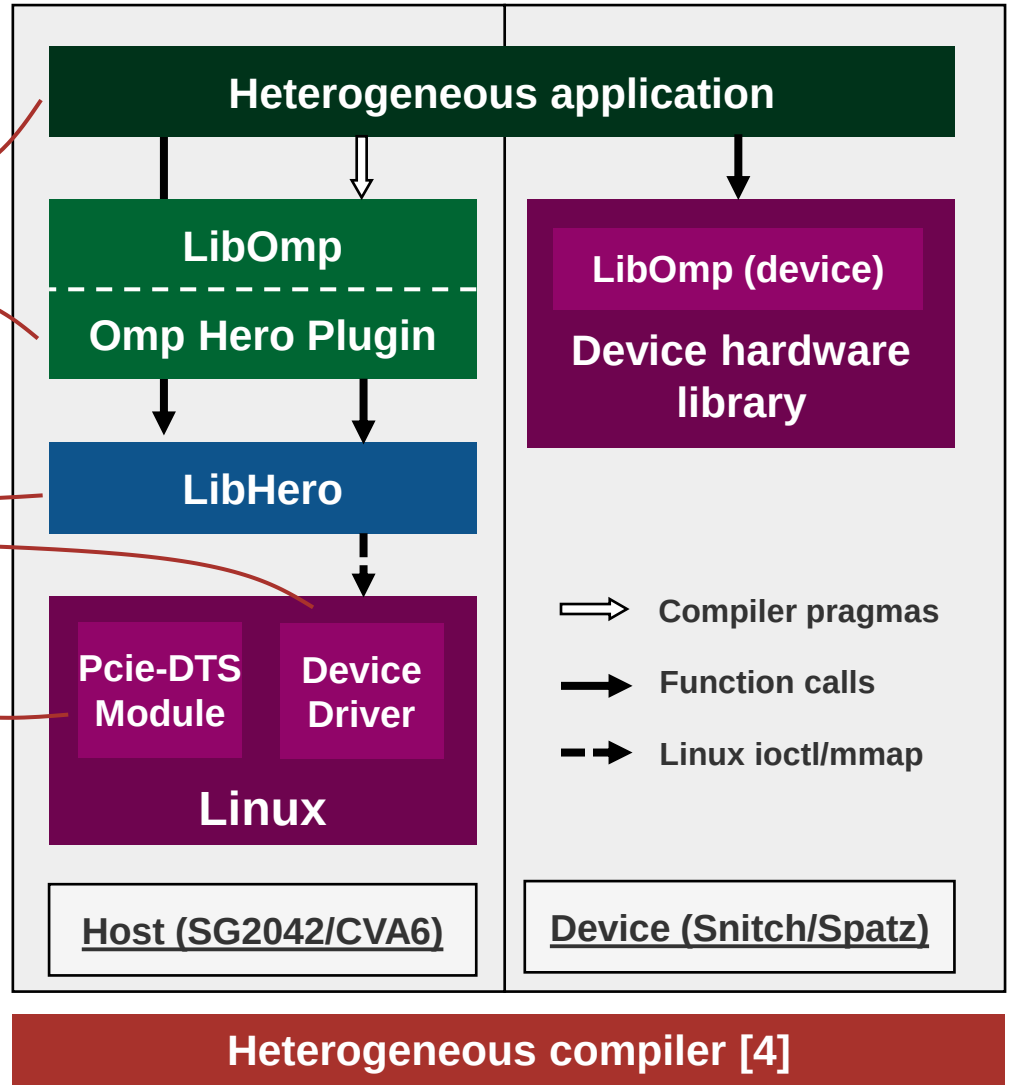
Reusable OpenMP offload plugin

Middle layer user-space library

Device driver (kernel module)

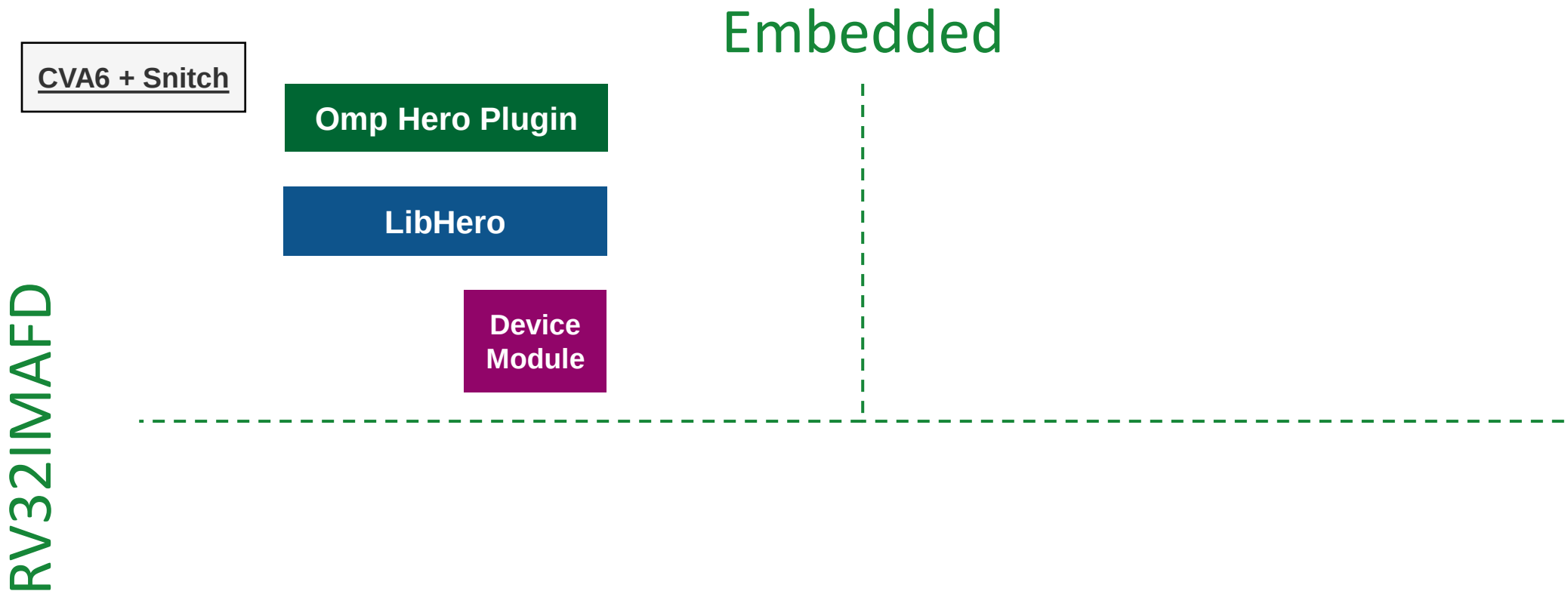
Extension to PCIe cards via device tree overlays [5]

```
pcie@7062000000
├── ranges
├── pci@0,0
│   ├── ranges
│   ├── dev@0,0
│   │   ├── ranges
│   │   ├── accelerator-soc
│   │   │   ├── ranges
│   │   │   ├── snitch-cluster@10000000
...
```

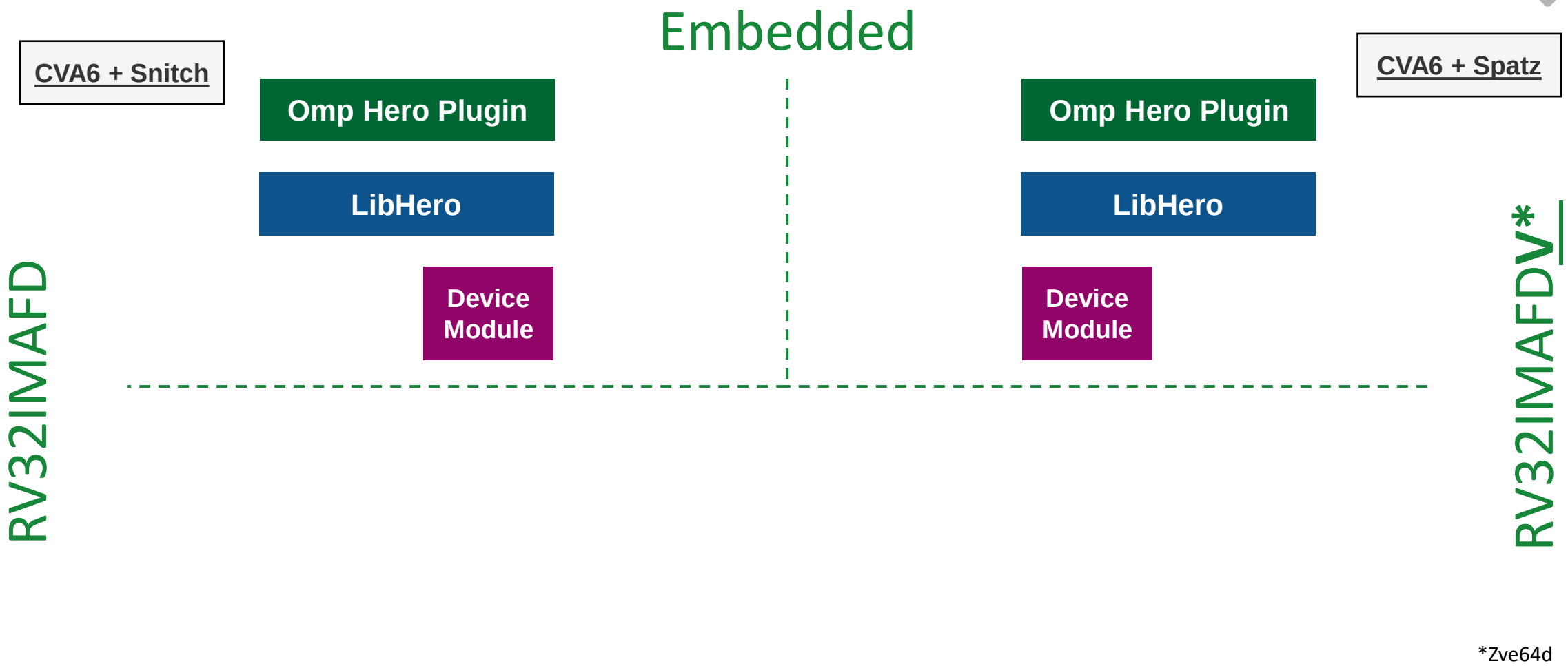


[4] Kurth et al. HEROv2: Full-Stack Open-Source Research Platform for Heterogeneous Computing
[5] PCIe node introduced in Linux 6.6-rc5 (Lizhi Hou)

First support for RV32IMAFD

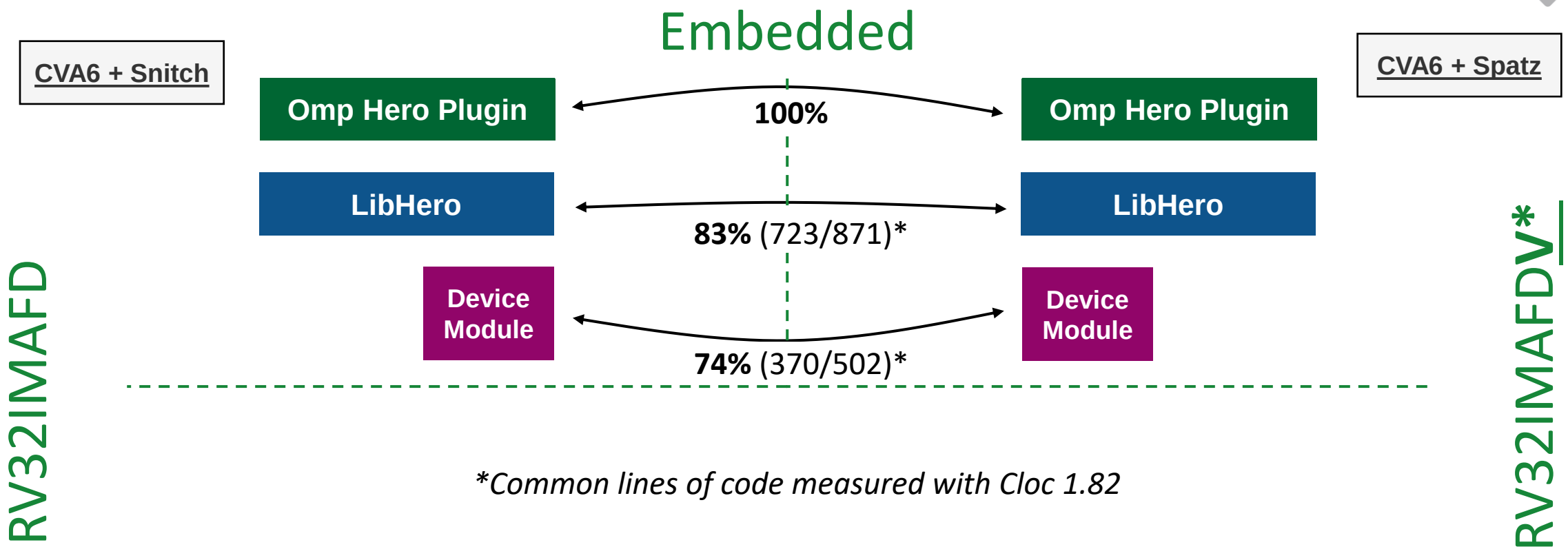


Extension to RV32IMAFDV



*Zve64d

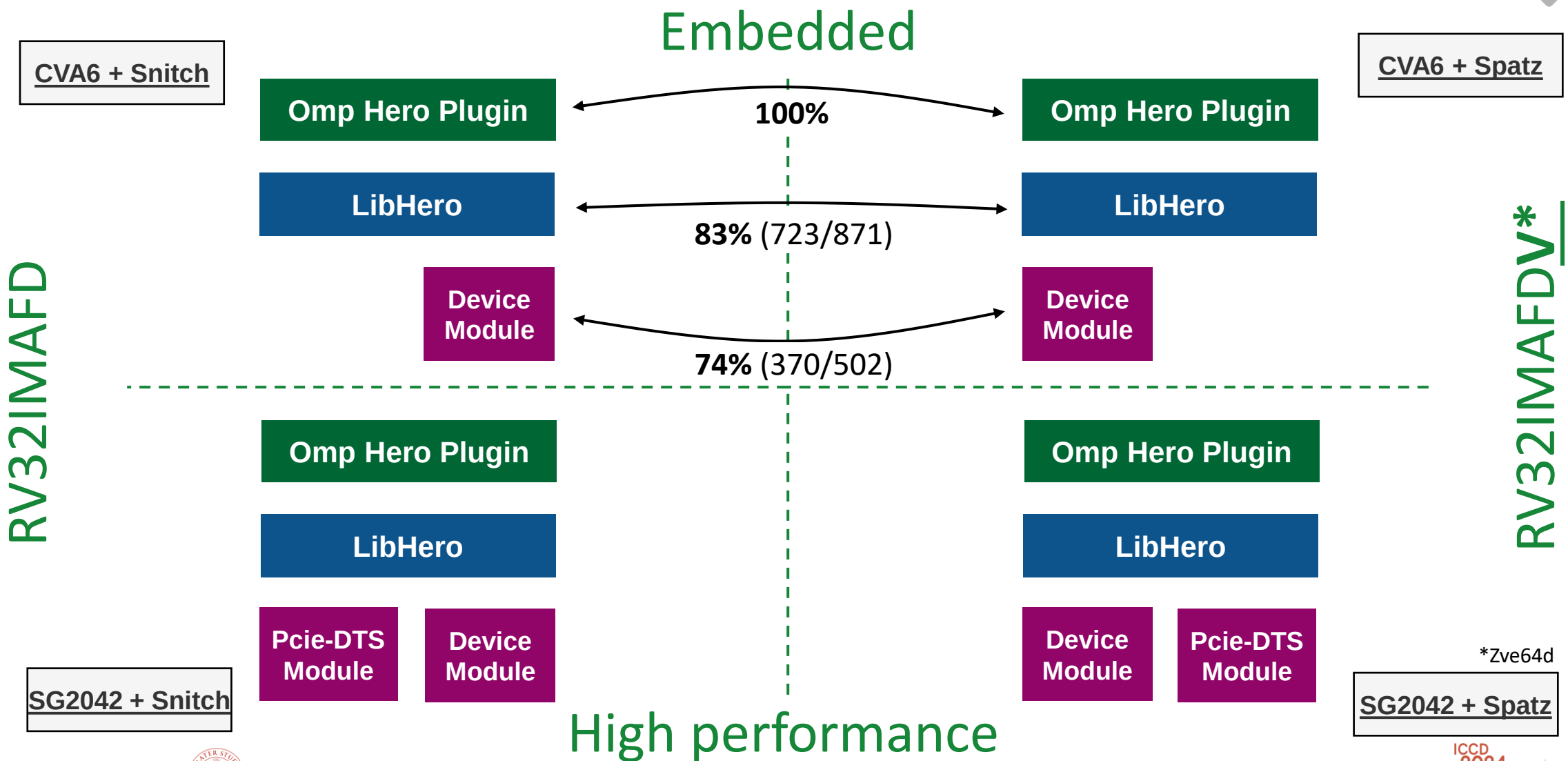
Extension to RV32IMAFDV with high reusability



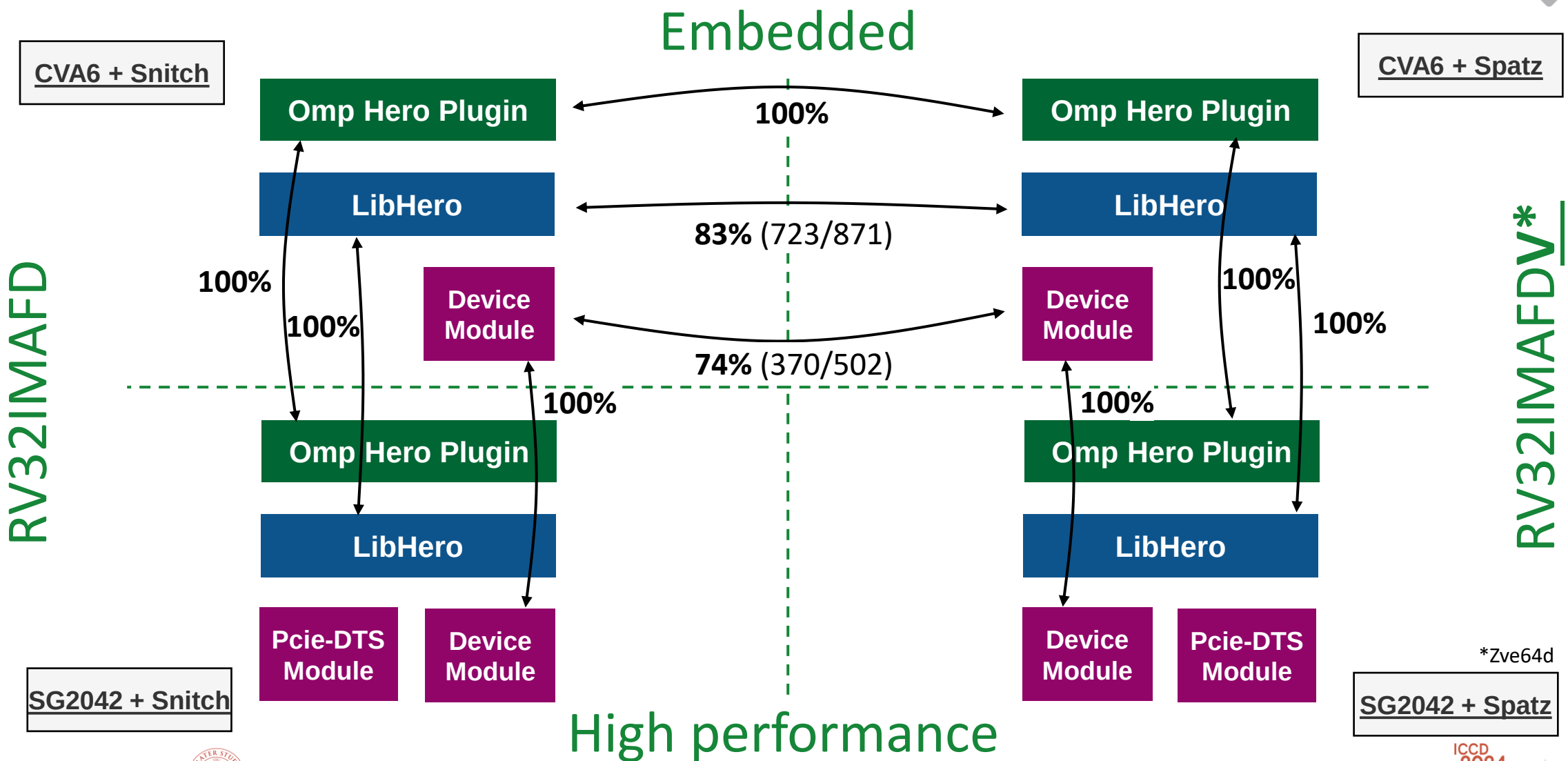
*Common lines of code measured with Cloc 1.82

*Zve64d

Extension to PCIe platform



Extension to PCIe platform with 100% reusability



Measuring offload latency of HeroSDK

Three micro kernels to measure offload latency

a) No-op

b) Input read

c) Input read and Output write

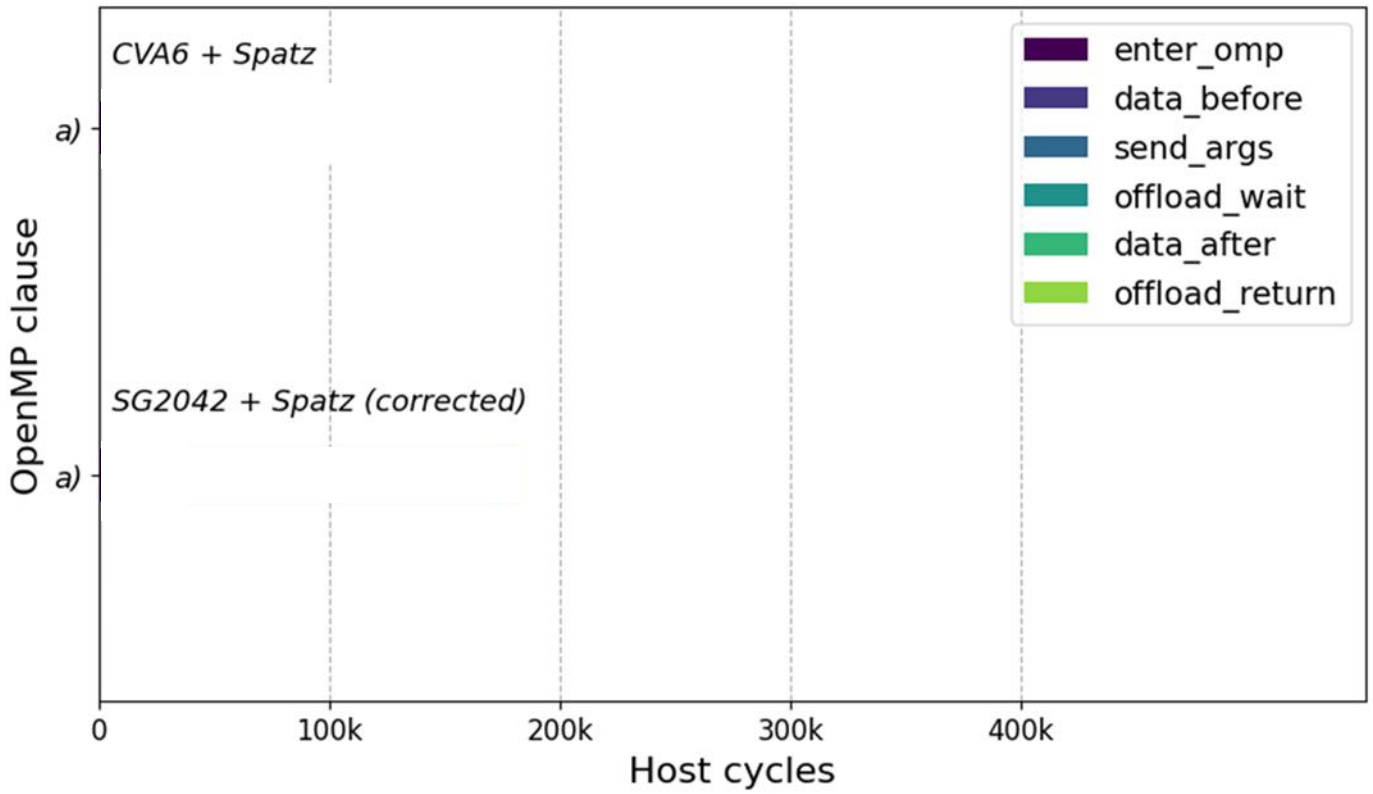
Process kernel info on the host

> Faster on SG2042 (triple issue C910 core)

Less than 100k cycle OpenMP offload overhead

Synchronous fork/join on device

> Requires PCIe access on SG2042



Offload benchmarks

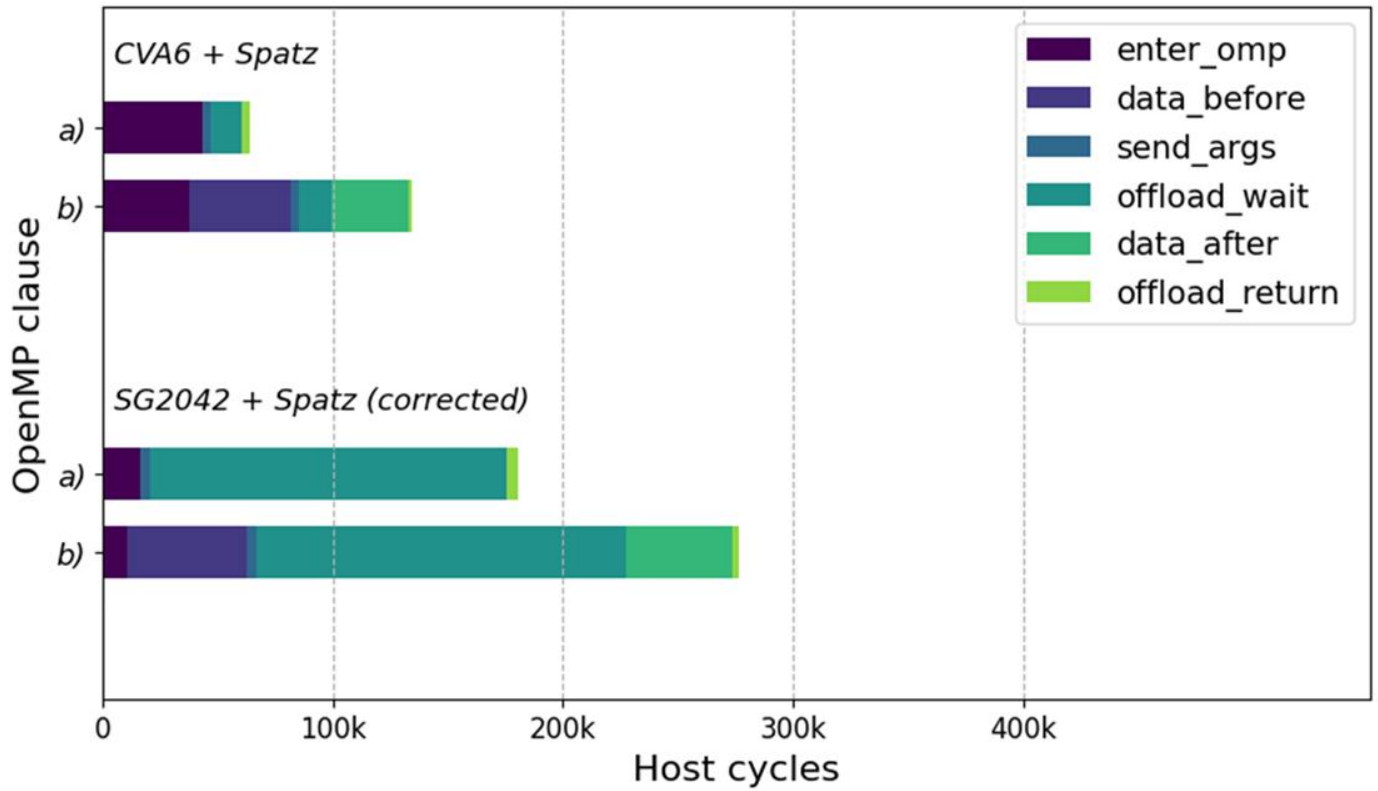
Three micro kernels to measure offload latency
a) No-op
b) Input read
c) Input read and Output write

Process kernel infos on the host
→ Faster on SG2042 (triple issue C910 core)

→ Allocate and send inputs to device

Synchronous fork/join on device
→ Requires PCIe access on SG2042

→ De-allocate



Offload benchmarks

Three micro kernels to measure offload latency
a) No-op
b) Input read
c) Input read and Output write

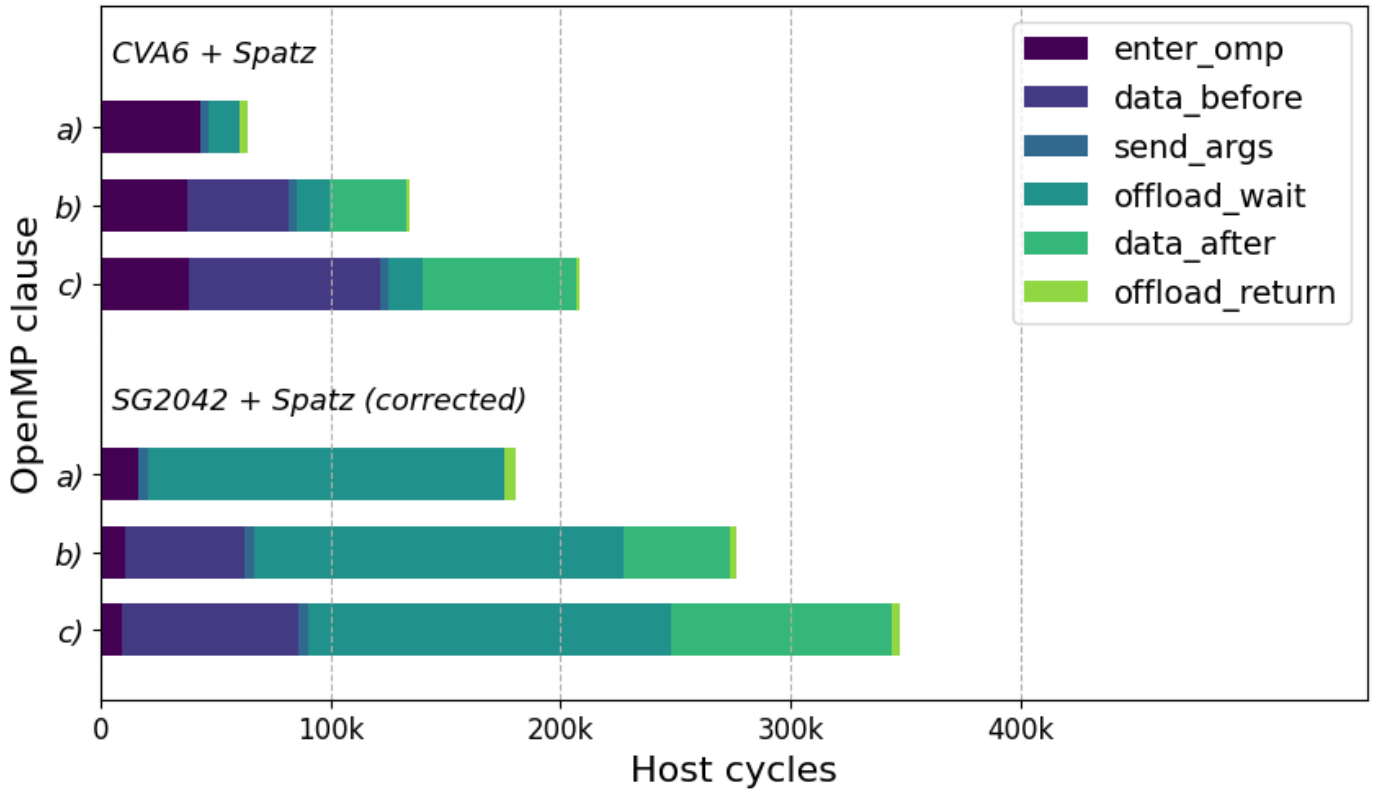
Process kernel infos on the host
→ Faster on SG2042 (triple issue C910 core)

→ Allocate and send inputs to device

Synchronous fork/join on device
→ Requires PCIe access on SG2042

→ **De-allocate and read outputs from device**

O(100k) offload overhead
This impacts the choice of kernels to offload



Matrix vector toy example

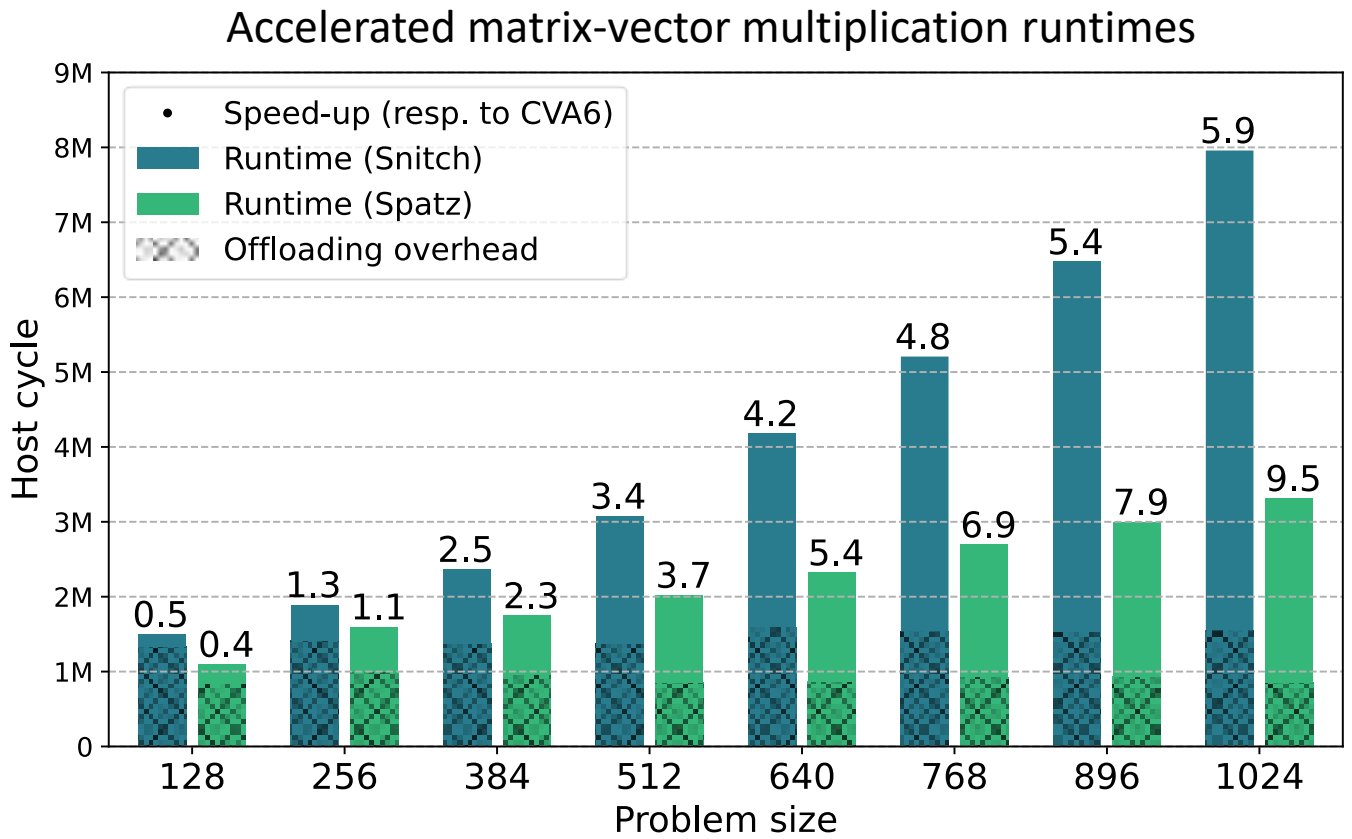
- **Matrix vector multiplication**

- **5.9× speedup**
  19 % offload-to-work ratio

  (Snitch)

- **9.5× speedup**
  25 % offload-to-work ratio

  (Spatz)



RISC-V unlocks heterogeneous programming across applications and devices!

- **HeroSDK proposes a framework for all-RV heterogeneous programming**
- **It can be easily extended to new devices**
- **It suits embedded or performance software stack**

www.github.com/pulp-platform/hero-tools

PULP Platform

Open Source Hardware, the way it should be!

Cyril Koenig cykoenig@iis.ee.ethz.ch
Björn Forsberg bjorn.forsberg@ri.se
Luca Benini lbenini@iis.ee.ethz.ch

www.github.com/pulp-platform/hero-tools

Institut für Integrierte Systeme – ETH Zürich
Gloriastrasse 35
Zürich, Switzerland

DEI – Università di Bologna
Viale del Risorgimento 2
Bologna, Italy

ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

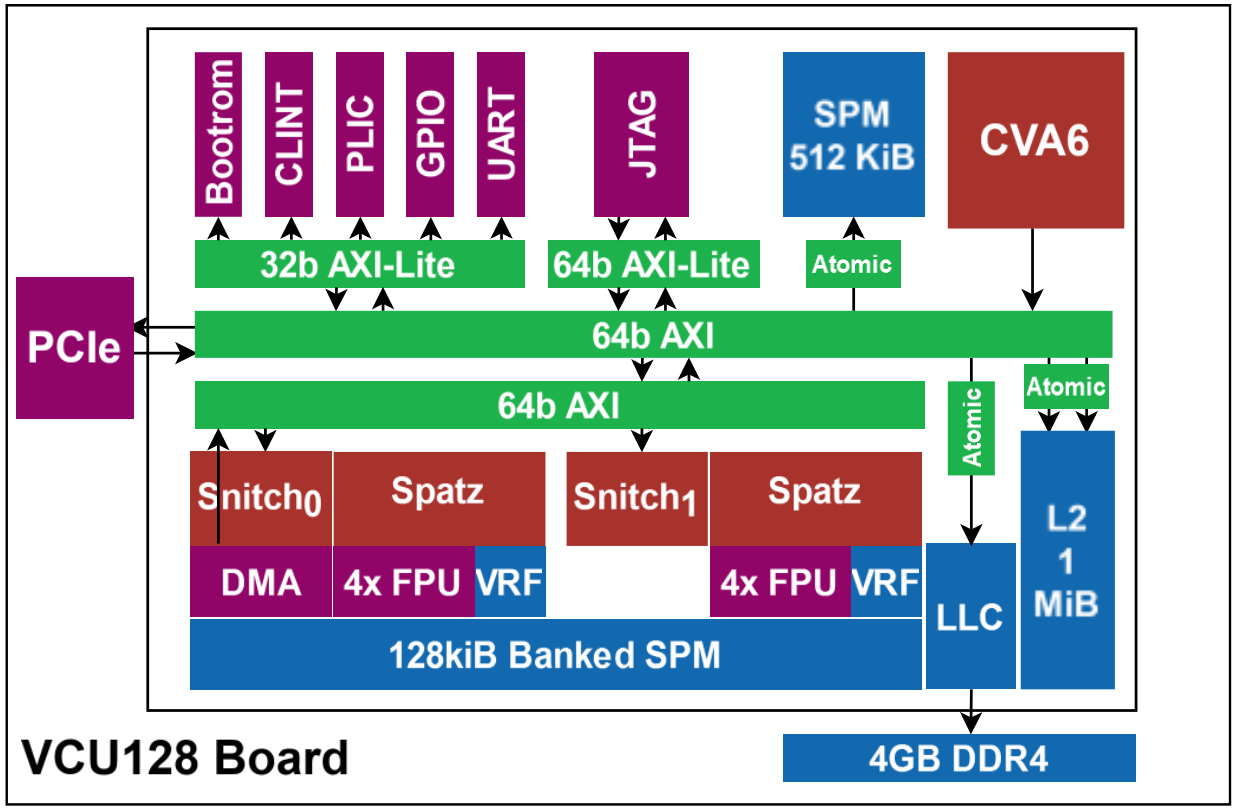
Demonstrated Spatz platforms

- **Spatz embedded platform**
  - RV64G single core host
  - 2-core RV32IMAFD**V** cluster
  - Cluster scratchpad and DMA

- **Spatz accelerator card**
  - 2-core RV32IMAFD**V** cluster
  - Cluster scratchpad and DMA
  - PCIe endpoint

+

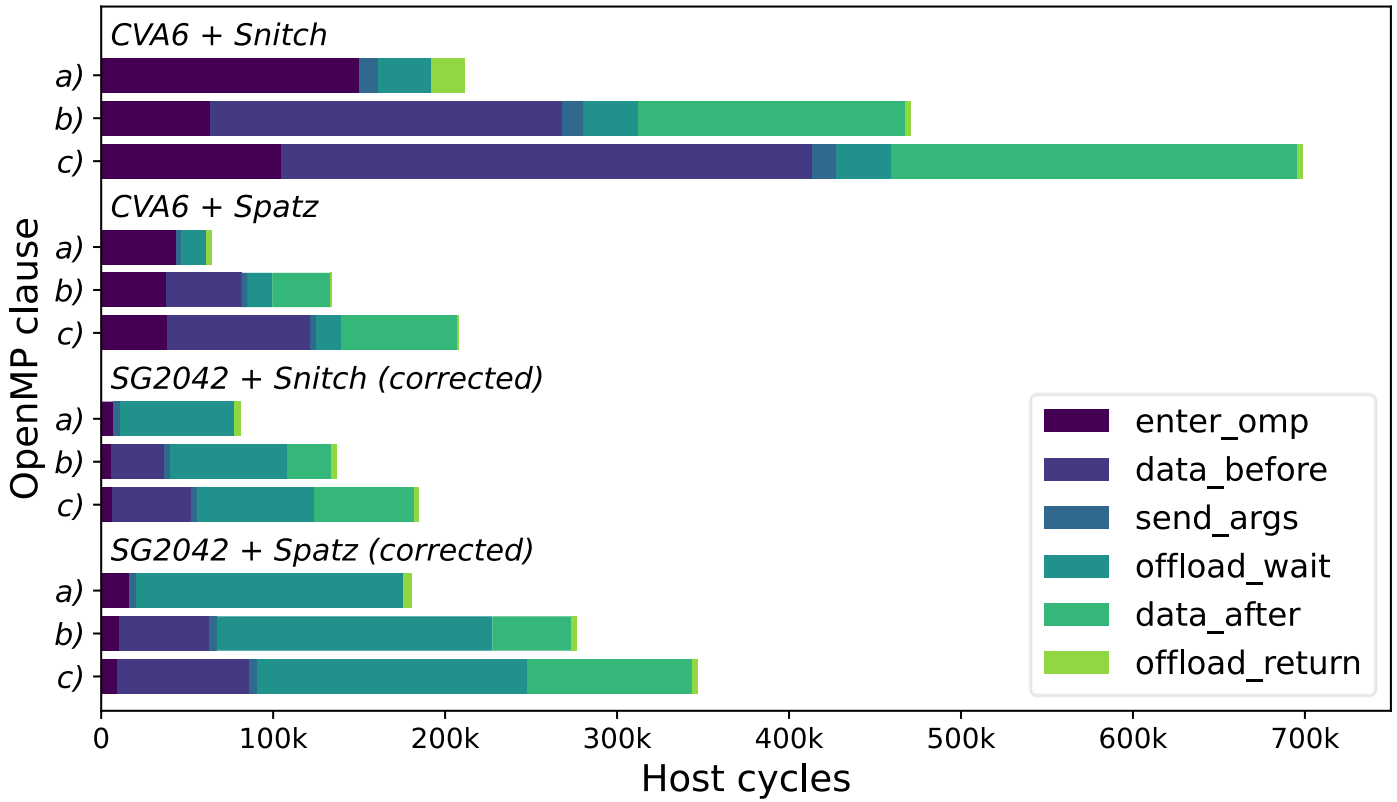
- **SG2042 RISC-V CPU**
  - RV64G 64-cores host



www.github.com/pulp-platform/carfield

Offload benchmark

```
1  #pragma omp target device(1)
2    asm volatile ("nop");
3
4  #pragma omp target device(1) map(to: a)
5    volatile uint32_t b = a;
6
7  #pragma omp target device(1) map(to: a) map(tofrom: b)
8    b = a;
```



Example application

```
1
2  void matvec( float *W_p, float *X_p, float *Y_p,
       unsigned int N ) {
3    // Offload to manager accelerator core
4    #pragma omp target device(1) map(to: W_p, X_p, Y_p, N)
5    {
6      // Allocate and move input data to local memory
7      X_l1 = snitch_l1_alloc(N);
8      W_rows_l1 = snitch_l1_alloc (CORES * N);
9      snitch_dma_memcpy (X_l1, X_p, N)
10     // Tile matrix rows
11     for ( offset = 0; offset += CORES; offset < N ) {
12       snitch_dma_memcpy(W_rows_l1, X_p[offset], N*CORES)
13       snitch_dma_wait();
14       // Execute in parallel
15       #pragma omp parallel for
16       for ( int p = 0; p < CORES; p++ )
17         Y_p[offset+p] = dotp(X_l1, &W_rows_l1[p*n], N)
18     }
19   }
20   return;
21 }
```

Heterogeneous compiler