DESIGN, AUTOMATION AND TEST IN EUROPE
THE EUROPEAN EVENT FOR ELECTRONIC SYSTEM DESIGN & TEST
31 MARCH – 2 APRIL 2025
LYON, FRANCE
CENTRE DE CONGRÈS DE LYON

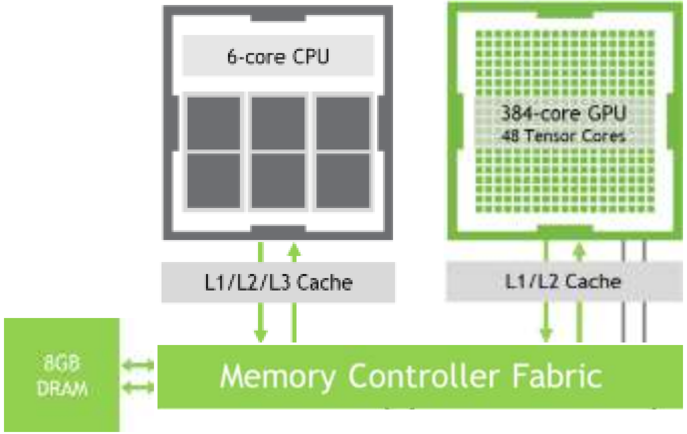
# Evaluating IOMMU-Based Shared Virtual Addressing for RISC-V Embedded Heterogeneous SoCs

**Cyril Koenig, Enrico Zelioli, Luca Benini**

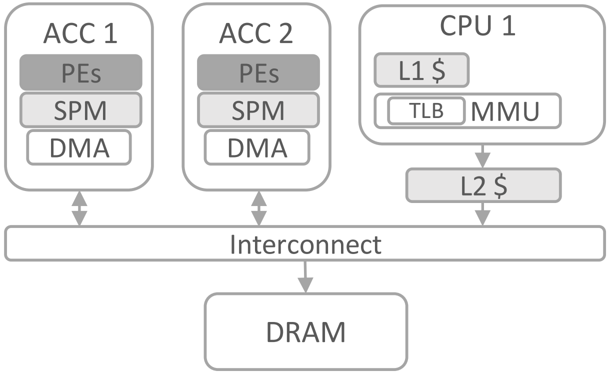
Integrated Systems Laboratory – ETH Zürich

*{cykoenig, ezelioli, lbenini}@iis.ee.ethz.ch*

# Heterogeneous SoCs: iGPU and PMCA




[1]




[2]

- CPU + iGPU
- PEs access data in caches
- Caches ⇔ DRAM via refill

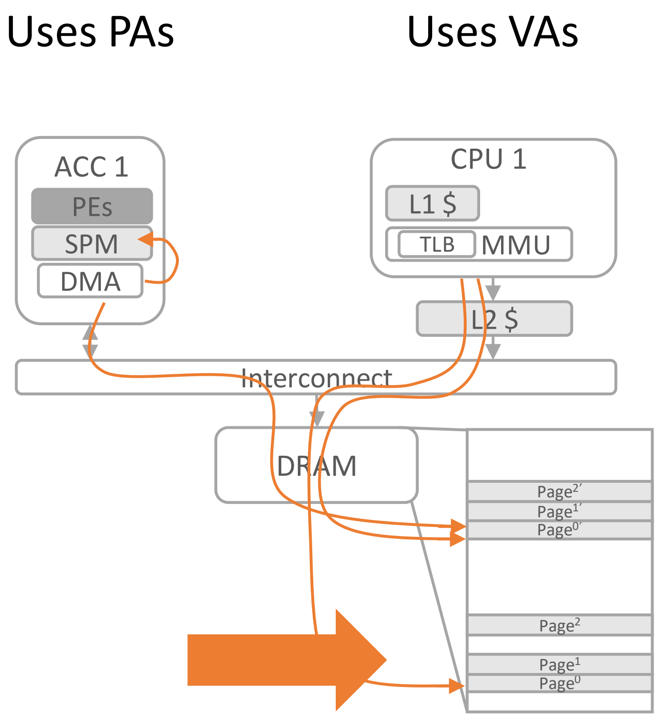
- CPU + PMCA (“NPU“)
- PEs access data in scratchpads (SPM)
- SPM ⇔ DRAM via DMA

[1] Adapted from https://www.nvidia.com/ [2] Adapted from “Fu et Al, Active Forwarding: Eliminate IOMMU Address Translation for Accelerator-rich Architectures”

# Sharing host allocated data

Virtual addressing issues:

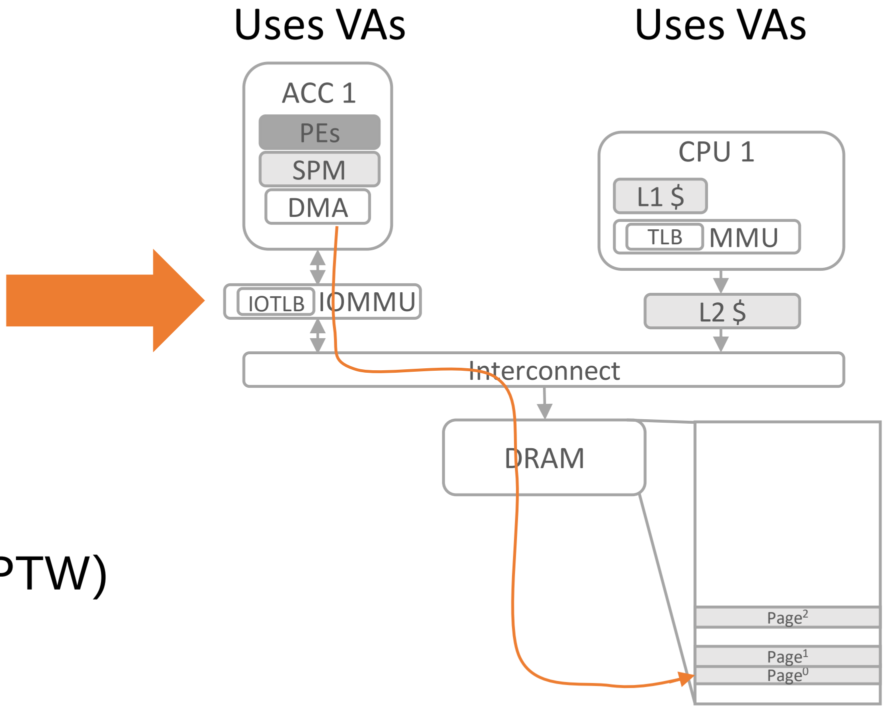
- Fragmentation
  - Physical pages are fragmented on the host
  - Device DMA needs contiguous memory
  - *Host needs to create contiguous copies*



# IO-virtual memory

Virtual addresses issues:

- Fragmentation
  - Physical pages are fragmented on the host
  - Device DMA needs contiguous memory
  - *Host needs to create contiguous copies*
- Virtualization
- Security




IOMMU may reduce available DMA bandwidth due to page table walking (PTW)

# How to reduce IO-PTW overhead?

- IO-TLBs coalescing [1]
- Multi-level TLBs [2]

Used both for iGPU and PMCA

- Host page walking [2]
- Active forwarding [3]

Used for embedded HeSoC with PMCA

<u>Limitations:</u>

- Some proposed method requires high architectural changes on the host
- Work often based on simulators, no open-source reference platform for research shared virtual adressing for RISC-V

[1] B. Pham, et Al: "CoLT: Coalesced Large-Reach TLB" [2] Y. Hao, et Al. "Supporting Address Translation for Accelerator-Centric Architectures"
[3] Fu et Al, "Active Forwarding: Eliminate IOMMU Address Translation for Accelerator-rich Architectures"

# Contributions

- Release an **open-source RISC-V heSoC** with **PMCA** and an open-source **IOMMU** IP [1]
- Heterogeneous benchmarks using **OpenMP offloading** with **shared virtual addresses**.
- We show IO-page walking **overhead ranging from 4.2% to 17.6%** of **GEMM** execution time under different DRAM latencies.
- By using a **shared last-level cache**, we show that the IO-PTW overhead **is below 1% in the same conditions.**

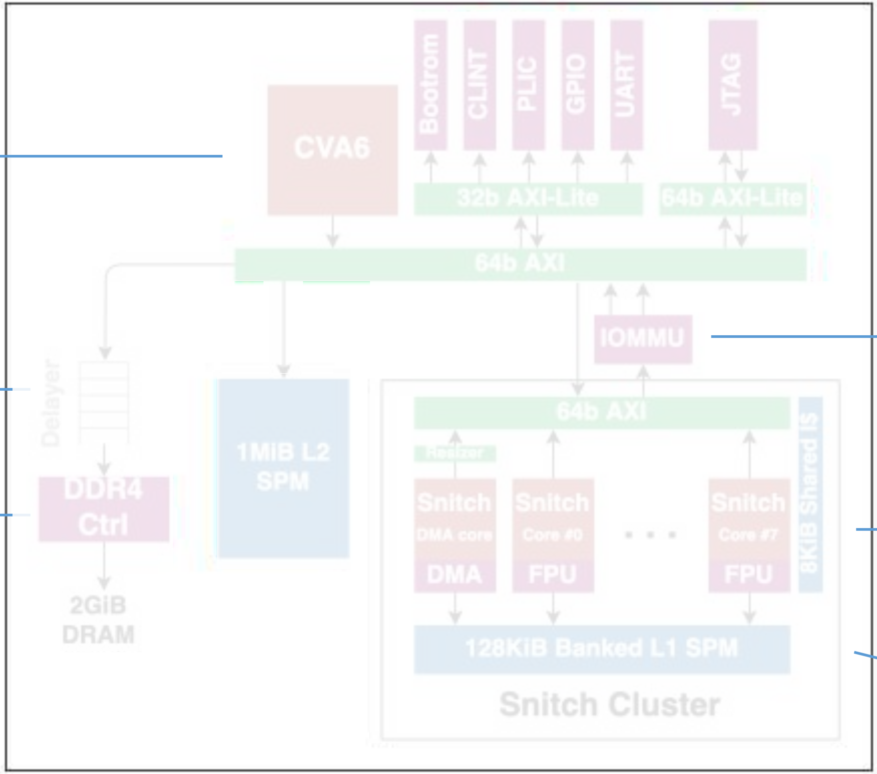
[1] M. Rodríguez, et Al: "Open-source RISC-V Input/Output Memory Management Unit (IOMMU) IP"

# The platform (based on Cheshire [1])

CVA6 Host running Linux

AXI delayer

Shared DRAM (VCU128 FPGA)




RISC-V IOMMU [2]

8-cores RV32IMAFD

DMA and scratchpad


[1] A. Ottaviano, et Al: “Cheshire: A Lightweight, Linux-Capable RISC-V Host Platform for Domain-Specific Accelerator Plug-In”
[2] M. Rodríguez, et Al: “Open-source RISC-V Input/Output Memory Management Unit (IOMMU) IP”

# Heterogeneous benchmarks with OpenMP

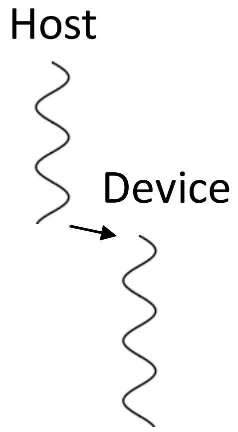
```
int main() {

  // Allocate host data
  float *a = malloc(...);

  // Create IO PT entries for it
  float *a_dev = iommu_map(a, ...);

  // Wake up accelerator
  #pragma omp target device(1) map(to: a_dev)
  {
    // Copy data for tile N+1
    dma_in();
    // Operate on tile N
    device_kernel();
    // Copy out data for tile N
    dma_out();

  }

}
```

# Heterogeneous benchmarks with OpenMP

Host

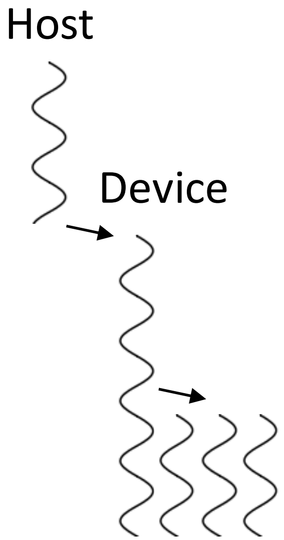
```
int main() {

  // Allocate host data
  float *a = malloc(...);

  // Create IO PT entries for it
  float *a_dev = iommu_map(a, ...);

  // Wake up accelerator
  #pragma omp target device(1) map(to: a_dev)
  {
    // Copy data for tile N+1
    dma_in();
    // Operate on tile N
    device_kernel();
    // Copy out data for tile N
    dma_out();

  }

}
```

# Heterogeneous benchmarks with OpenMP



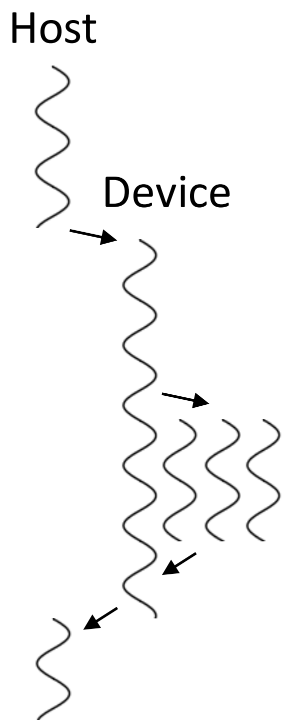

```c
int main() {

  // Allocate host data
  float *a = malloc(...);

  // Create IO PT entries for it
  float *a_dev = iommu_map(a, ...);

  // Wake up accelerator
  #pragma omp target device(1) map(to: a_dev)
  {
    // Copy data for tile N+1
    dma_in();
    // Operate on tile N
    device_kernel();
    // Copy out data for tile N
    dma_out();

  }

}
```

# Heterogeneous benchmarks with OpenMP



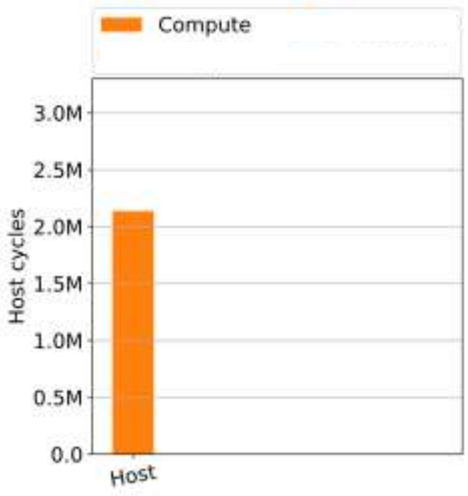

```
int main() {

  // Allocate host data
  float *a = malloc(...);

  // Create IO PT entries for it
  float *a_dev = iommu_map(a, ...);

  // Wake up accelerator
  #pragma omp target device(1) map(to: a_dev)
  {
    // Copy data for tile N+1
    dma_in();
    // Operate on tile N
    device_kernel();
    // Copy out data for tile N
    dma_out();

  }

}
```

# Heterogeneous benchmarks with OpenMP




```
int main() {

  // Allocate host data
  float *a = malloc(...);

  // Create IO PT entries for it
  float *a_dev = iommu_map(a, ...);

  // Wake up accelerator
  #pragma omp target device(1) map(to: a_dev)
  {
    // Copy data for tile N+1
    dma_in();
    // Operate on tile N
    device_kernel();
    // Copy out data for tile N
    dma_out();

  }

}
```

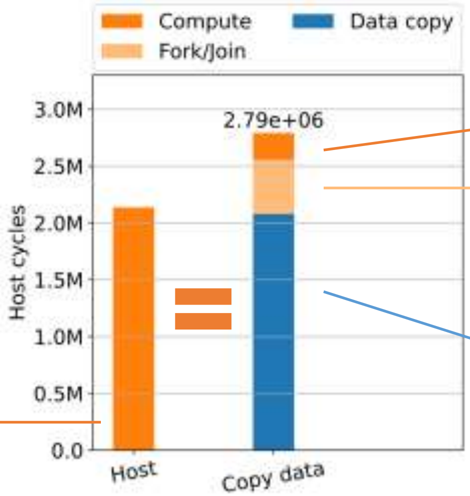
# Shared virtual memory benefits (AXPY)




Host runtime

# Shared virtual memory benefits (AXPY)

Host runtime

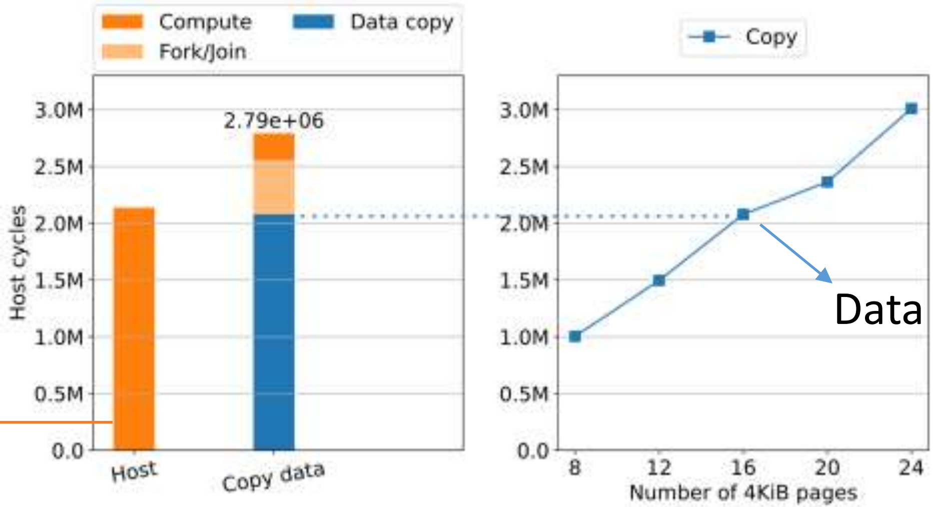



Accelerator runtime

Offload

Data copy DRAM ⟺ DRAM (Defragmentation)

# Shared virtual memory benefits (AXPY)

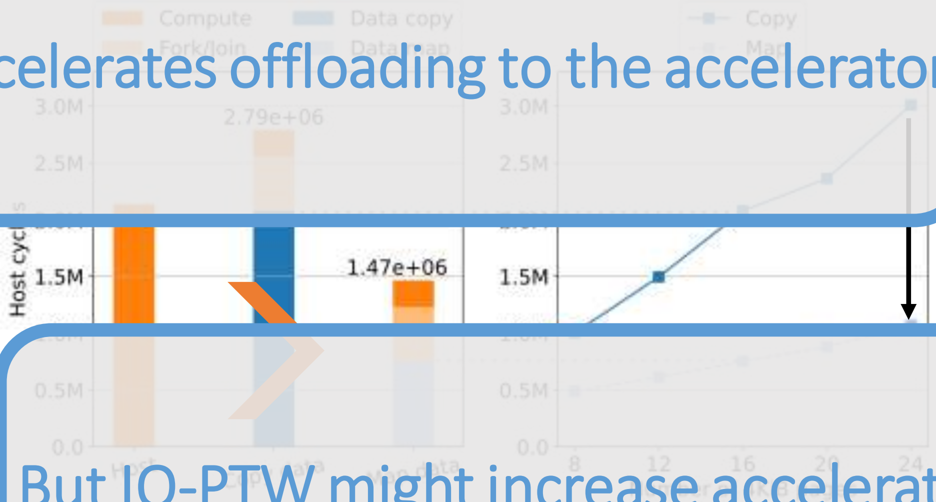



Host runtime

Data copy DRAM ⟺ DRAM
(Defragmentation)

# Shared virtual memory benefits (AXPY)

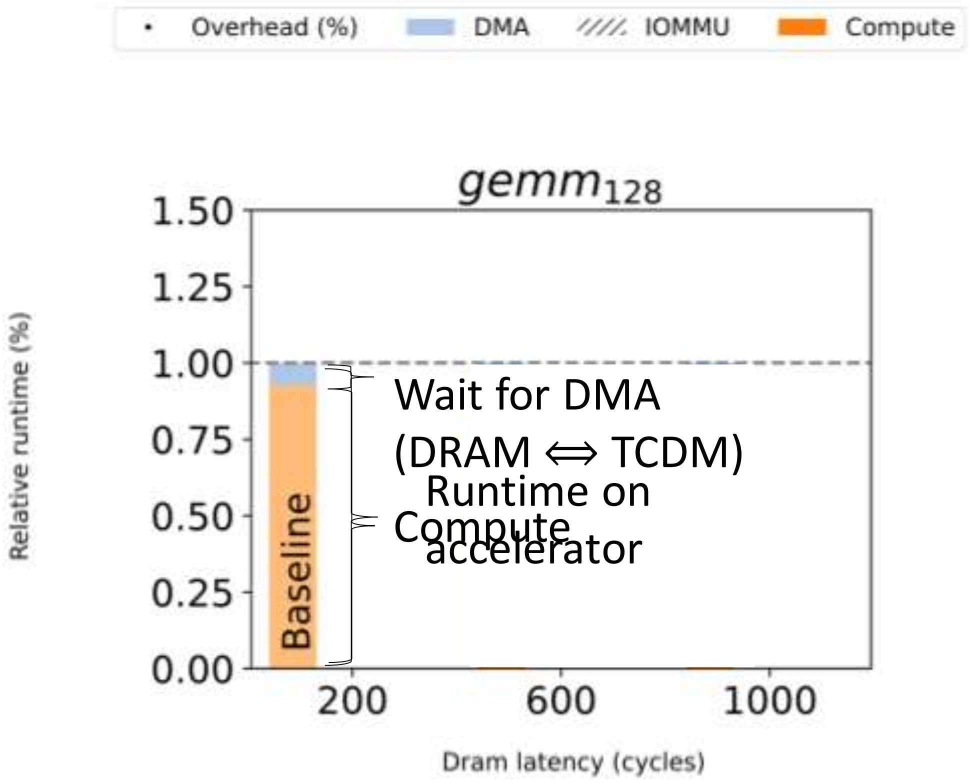
IOMMU accelerates offloading to the accelerator



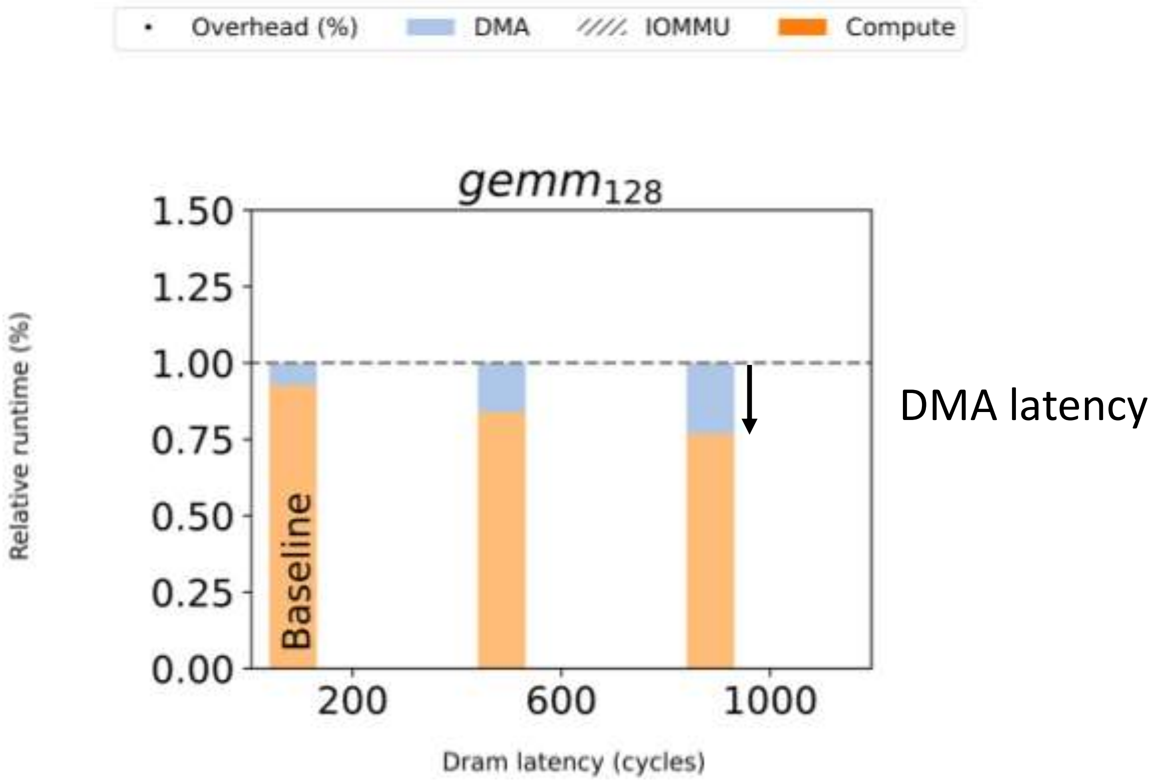

Creating an IO-PT is faster than copying data pages

But IO-PTW might increase accelerators' runtime

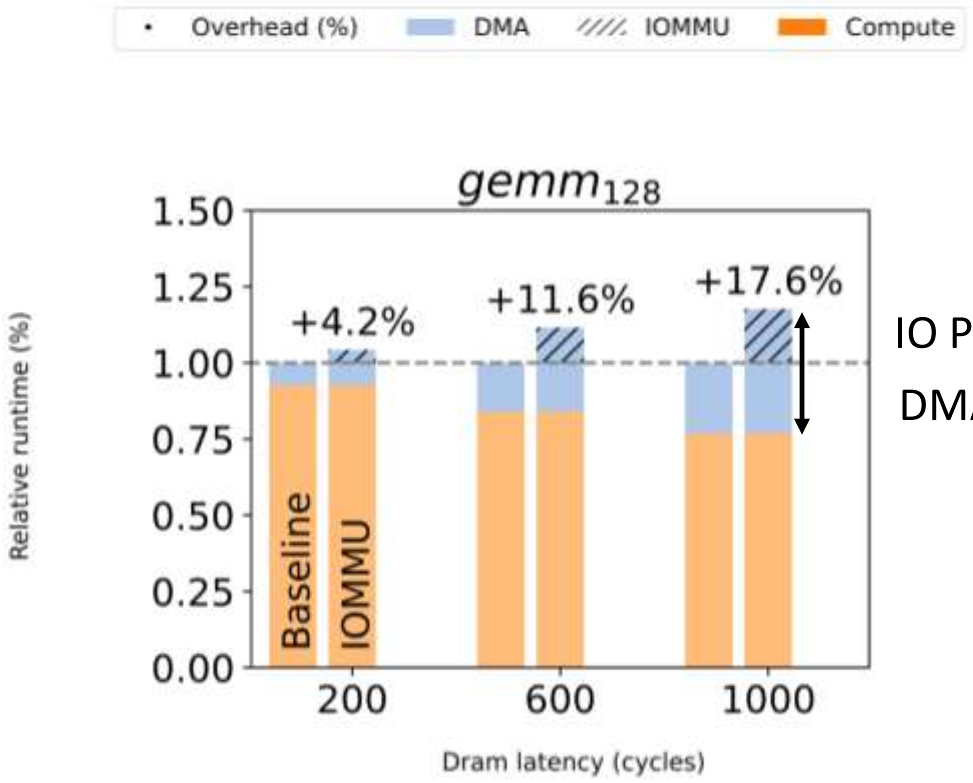
# Baseline accelerator runtime



# Impact of DRAM latency



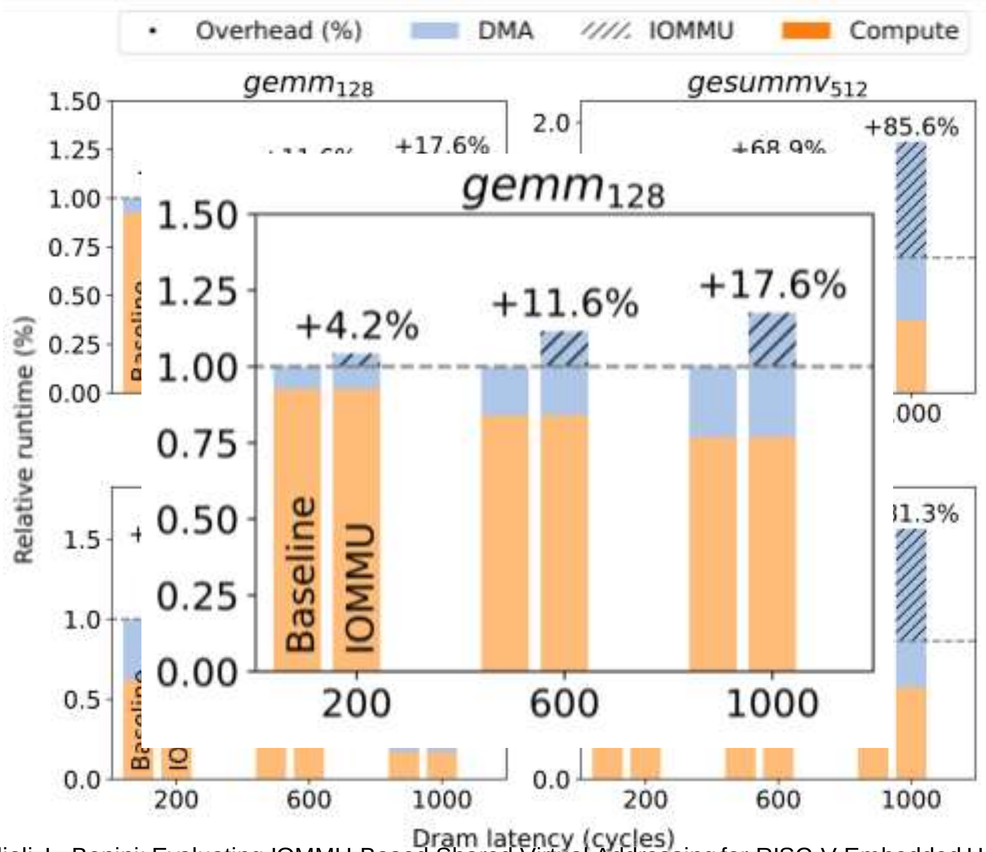
# Impact of IO-PTW




IO PTW latency

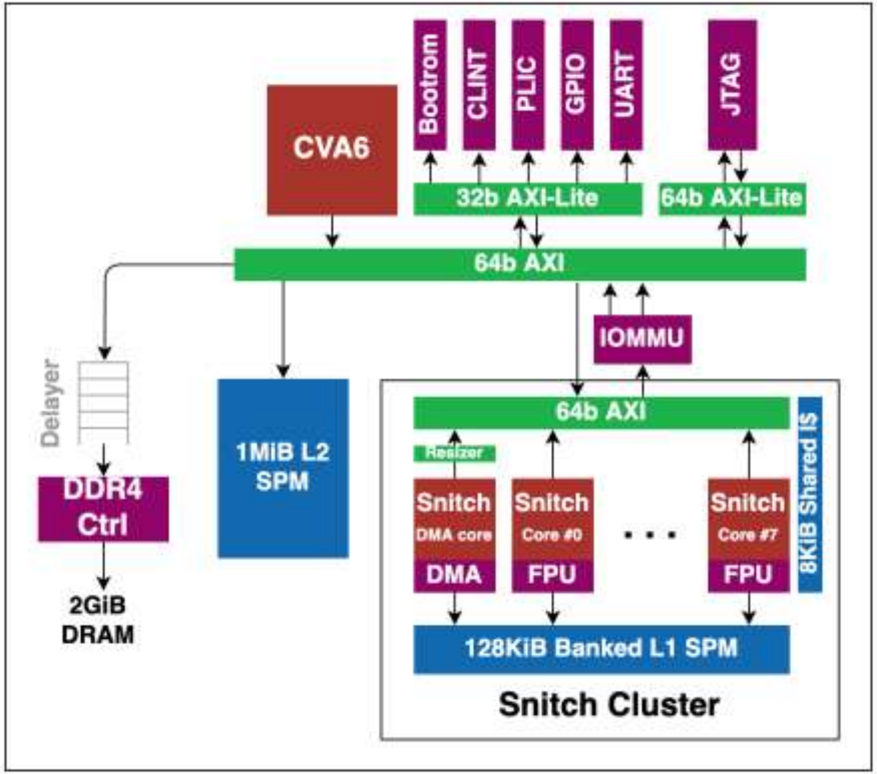
DMA latency

# Impact of IO-PTW




C. Koenig, E. Zelioli, L. Benini: Evaluating IOMMU-Based Shared Virtual Addressing for RISC-V Embedded Heterogeneous SoCs

# The platform (based on Cheshire [4])



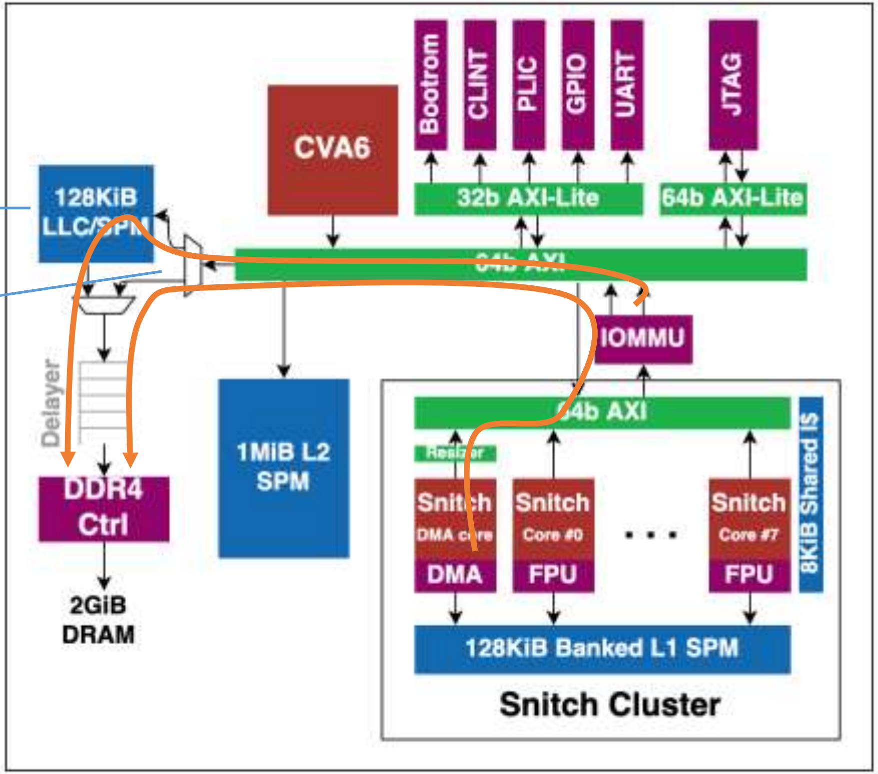

Usually PMCA based architectures don‘t have a last-level cache

→ DMA issue large bursts to DRAM
→ A last-level cache might split burst into cachelines

# The platform (based on Cheshire [4])

LLC

Bypass logic

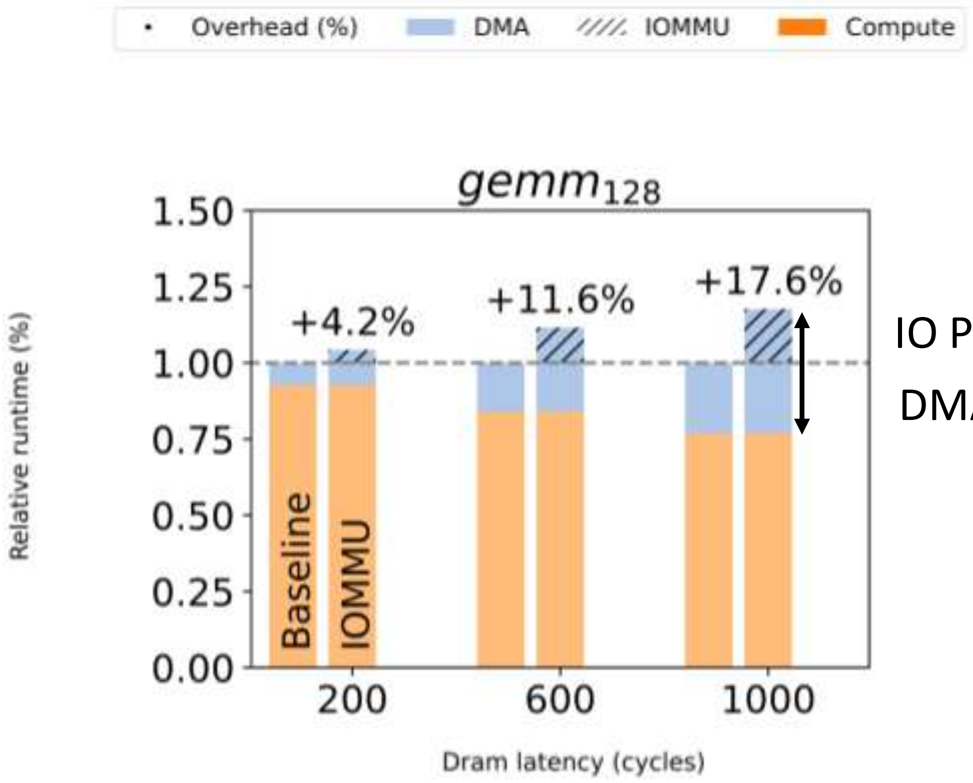



Usually PMCA based architectures don‘t have a last-level-cache

→ DMA issue large bursts to DRAM

We include a bypass logic for the DMA

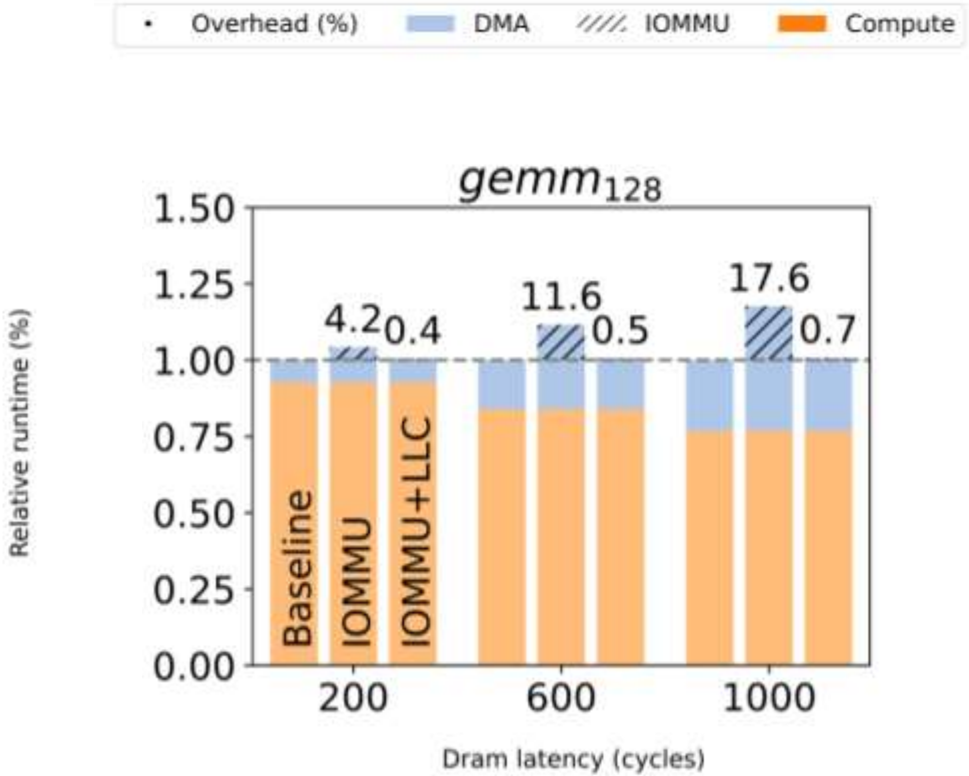
# Impact of IO-PTW



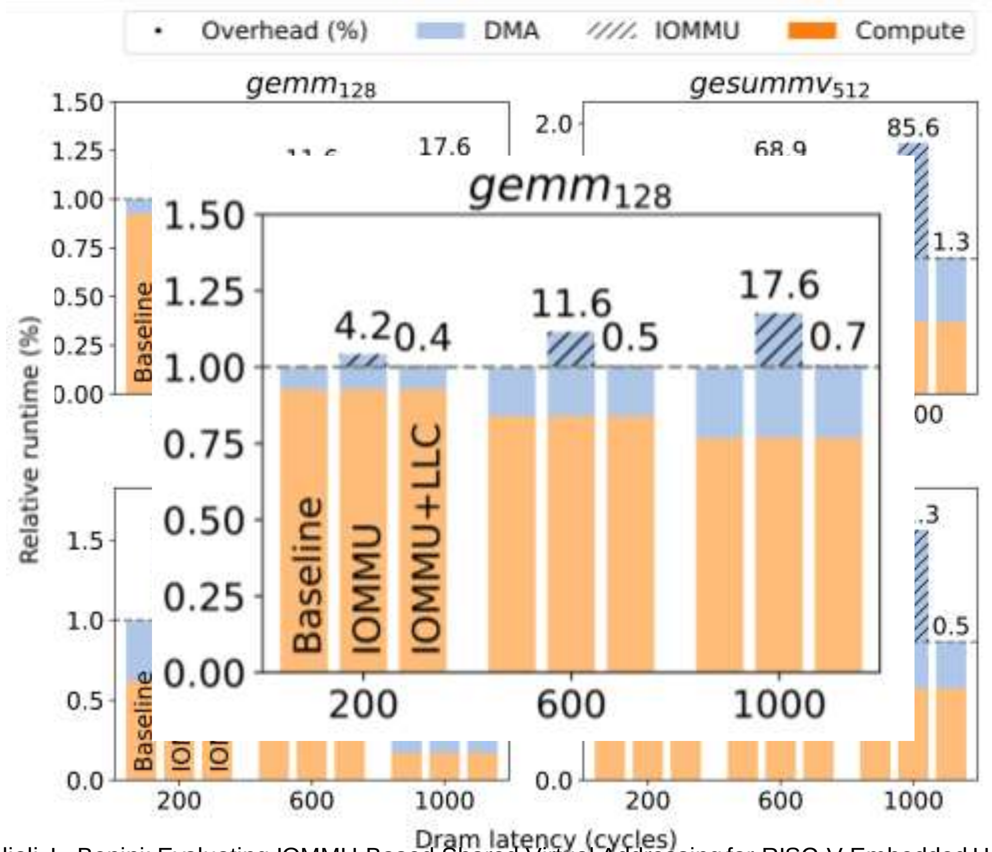

IO PTW latency

DMA latency

# Impact of IO-PTW with LLC

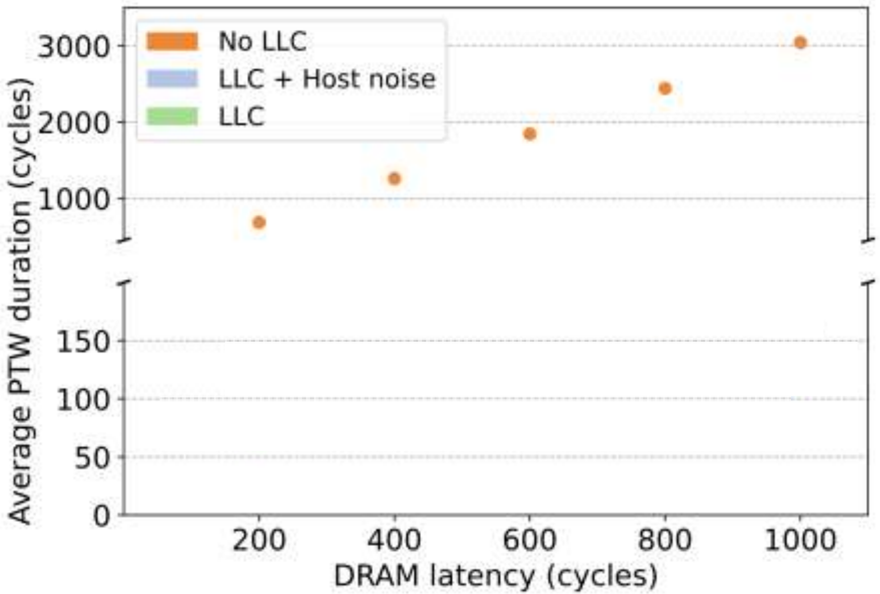


# Impact of IO-PTW with LLC



# Discussion: Impact of host noise on LLC

How do the IO-PTW duration reacts to high host noise?



# Conclusion

- The **open-source RISC-V platform for shared virtual memory research** is available at [1]
- We show that including a **shared last-level cache**, we reduce the IO-PTW overhead to **below 5% for all tested kernels.**
- We integrate a LLC bypass mechanism to maintain DMA bandwidth
- This platform and benchmarks can be used for further research on IOMMU architecture.

[1] https://github.com/pulp-platform/carfield/tree/date_iommu_evaluation