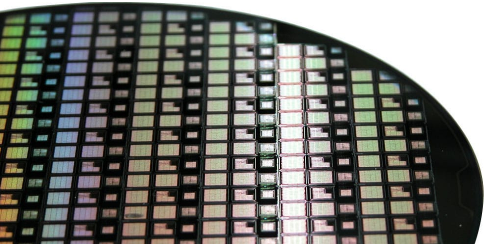
HERO: Open-Source Heterogeneous Embedded Research Platform for Exploring RISC-V Manycore Architectures on FPGA

First Workshop on Computer Architecture Research with RISC-V (CARRV) @ MICRO 50

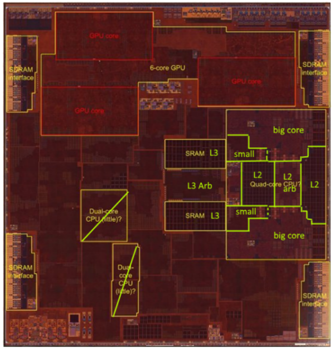
2017-10-14

Andreas Kurth
Pirmin Vogel
Alessandro Capotondi
Andrea Marongiu
Luca Benini

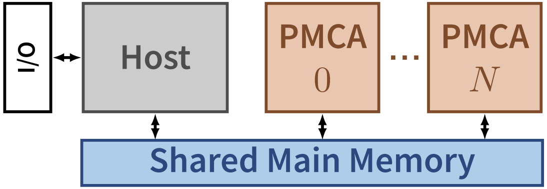
ETH zürich
Integrated Systems Laboratory
Digital Circuits and Systems Group



Heterogeneous Embedded Systems on Chip (HESoCs)



Die shot of an Apple A11 SoC (Source: chipworks).



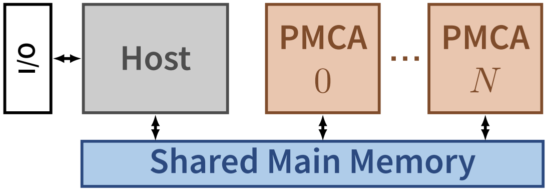
Architectural template of HESoCs.

- HESoCs co-integrate a **general-purpose host processor** and efficient, **domain-specific programmable manycore accelerators (PMCAs)**.
- They combine versatility with extreme nominal energy efficiency.

While industry rapidly advances products, …

The Research Gap on HESoCs

… **research on HESoCs lags behind**!



Architectural template of HESoCs.

There are **many open questions in various areas** of computer engineering:

- programming models, task distribution and scheduling,
- memory organization, communication, synchronization,
- accelerator architectures and granularity, …

But there is no research platform for HESoCs!

Problems with Simulating HESoCs

Developing HESoC components in isolation and estimating their system-level performance is problematic:

- Complex interactions between host, accelerators, and memory hierarchy make (reasonably accurate) **simulations orders of magnitude slower** than running prototypes.
- Even full-system simulators (e.g., GEM5) do not model all HESoC components.
- Models make assumptions about non-deterministic processes. The validity of results thus entirely depends on the validity of assumptions, and the assumptions for modeling HESoCs are very complex.

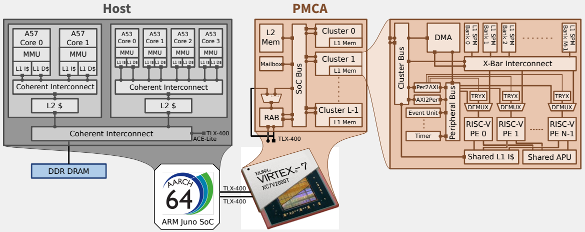
Conclusion: A research platform for HESoCs must be available.

This is not only about hardware:

- For system-level research, the platform must be **efficiently programmable**.
- Additionally, the platform should come with tools to **increase the observability** and **decrease the validation and implementation overhead** of the prototype.

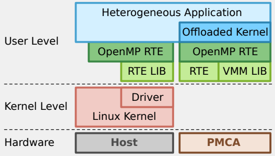
HERO: Open-Source Heterogeneous Embedded Research Platform

Heterogeneous Hardware Architecture



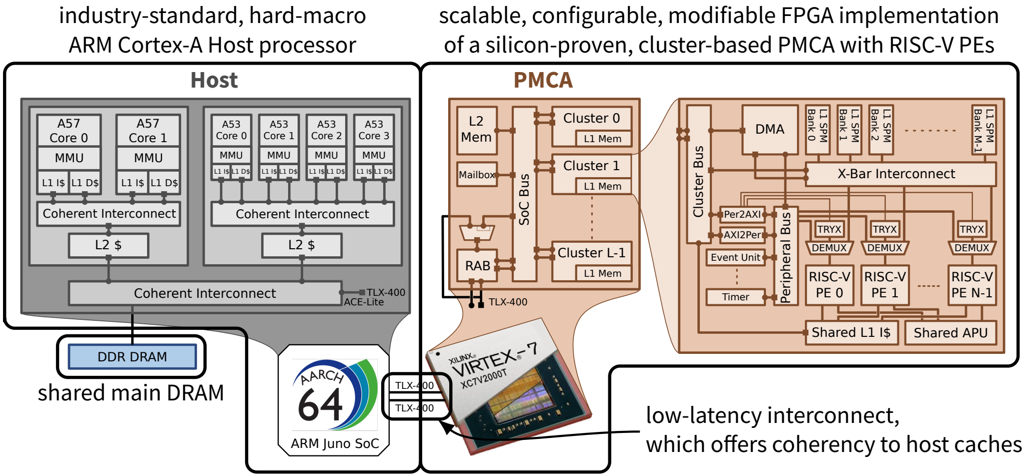
Heterogeneous Software Stack

- single-source, single-binary cross compilation toolchain
- OpenMP 4.5
- shared virtual memory for Host and PMCA



Profiling and automated verification solutions

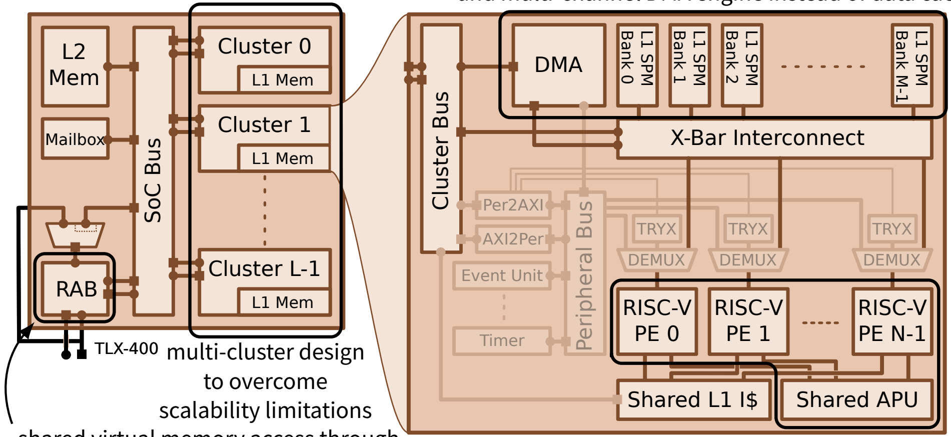
HERO’s Hardware Architecture



HERO’s hardware, as implemented on the Juno ADP.

PMCA Implementation on FPGA: Overview

multi-banked, software-managed scratchpad memories (SPMs)
and multi-channel DMA engine instead of data caches



multi-cluster design to overcome scalability limitations

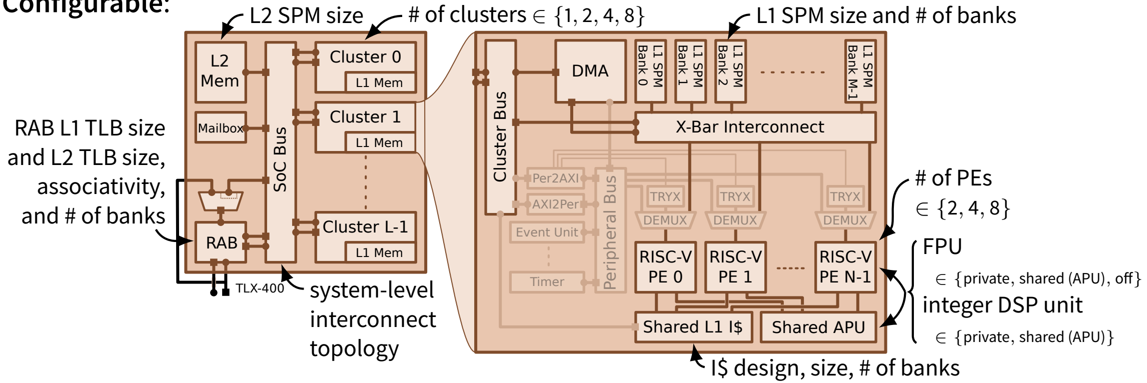
RISC-V processing elements (PEs) and shared auxiliary processing units (APUs) operating on local data

shared virtual memory access through
the software-managed, lightweight Remapping Address Block (RAB)

PMCA based on the PULP architectural template.

PMCA on FPGA: Configurable, Modifiable, and Expandable

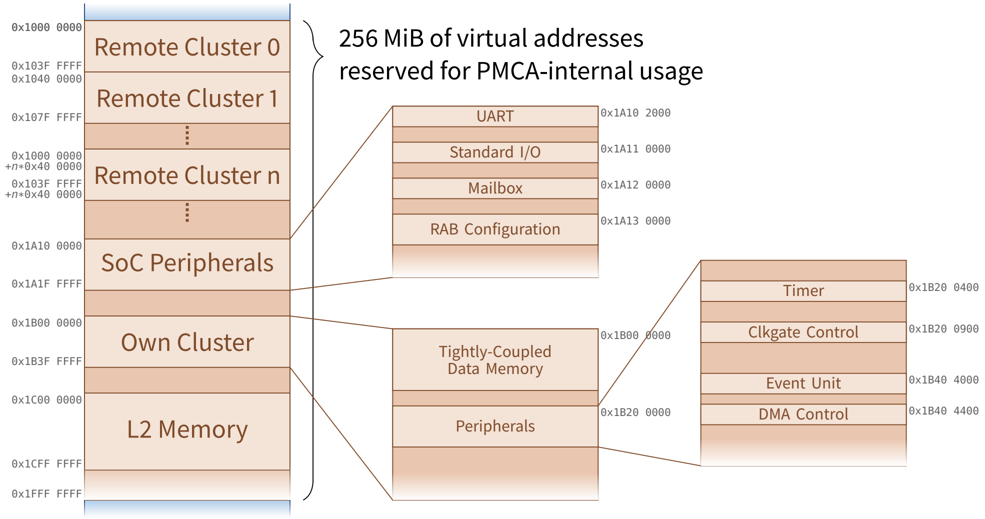
Configurable:



Modifiable and expandable:

- All components are open-source and written in industry-standard SystemVerilog.
- Interfaces are either standard (mostly AXI) or simple (e.g., stream-payload).
- New components can be easily added to the memory map.

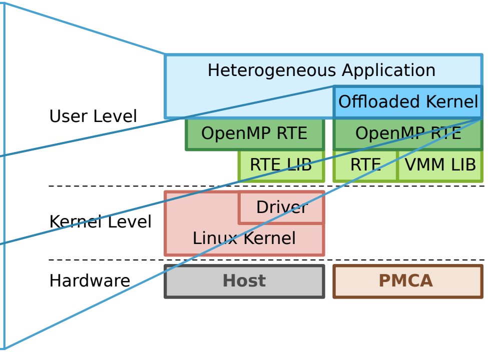
HERO’s Memory Map (from the Perspective of a PE in the PMCA)



HERO’s Software Stack

Allows to write programs that start on the host but seamlessly integrate the PMCAs.

```
int main()
{
  vertex vertices[N];
  load(&vertices, N);
  #pragma omp target map(tofrom:vertices)
  {
    #pragma omp parallel for
    for (i = 0; i < N; ++i)
      vertices[i] = process();
  }
}
```



- Offloads with OpenMP 4.5 `target` semantics, zero-copy (pointer passing) or copy-based
- Integrated cross-compilation and single-binary linkage
- PMCA-specific runtime environment and hardware abstraction libraries (HAL)

Software Stack: OpenMP

The `libgomp` plugin determines how **input and output variables** are **passed between host and PMCA**:

- With copy-based shared memory, **data is copied** to and from a physically contiguous, uncached section in main memory, and **physical pointers** are passed to the PMCA.
- Shared virtual memory enables zero-copy offloads, directly passing **virtual pointers** to the PMCA.

Furthermore, the plugin implements essential OpenMP functionality such as

- `parallel` (starting parallel execution)
- `team` (definition of parallel thread teams)
- `sections` (distributed, one-time execution worksharing)
- `barrier` (synchronization barrier)
- `critical` (single-threaded execution within a parallel region)

efficiently on the specific PMCA hardware.

Software Stack: Runtime Environment and VMM Library

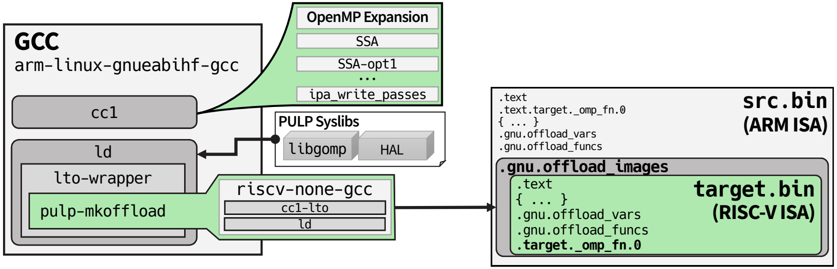
The PMCA can access the page table of the heterogeneous user-space application, and can **operate its virtual memory hardware**, the RAB, **autonomously**:

- Assume a core accesses a virtual address that is currently not in the RAB.
- That core goes to sleep and its miss is enqueued in the RAB.
- Another core handles the miss using the **VMM library** (details in the paper).

The VMM library is **compatible with any host architecture supported by the Linux kernel**.

Software Stack: Cross Compilation Toolchain

- OpenMP offloading with the GCC toolchain requires a **host compiler** plus one **target compiler** for each PMCA ISA in the system.



Details in: Capotondi *et. al.* 2017. Enabling zero-copy OpenMP offloading on the PULP many-core accelerator.

- A target compiler requires both **compiler extensions** (e.g., additional compilation and optimization passes) and **runtime extensions** (e.g., `libgomp` plugins).
- HERO includes the first non-commercial heterogeneous cross compilation toolchain.

Tools: Cycle-Accurate, Non-Interfering Tracer

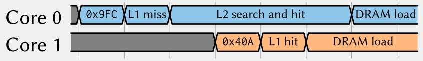
- Problem: **Poor observability of implementation internals** compared to simulation.
- Solution: **Run-time event tracing** and **post-mortem analysis**.
- **Requirements** on the tracer:
  1. must not interfere with program execution
     (e.g., inserting instructions to write memory is not an option),
  2. must be cycle-accurate yet be able to trace millions of consecutive events
     (to cover complex applications), and
  3. should use FPGA resources economically (to not hamper the evaluation of complex hardware).
- **Hybrid tracer design**: software-controlled, lightweight, customizable hardware blocks that ...
  - can use *any signals in the fabric* for data and trigger,
  - log timestamped events to dedicated, local buffers,
  - get flushed to main memory by the host while the PMCA is "frozen".
- Details are in the paper.

Tools: Event Analysis

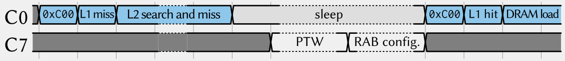
Input: Recorded events from tracers, e.g., memory transactions at the RAB (with meta information on source, read/write, and hit/miss):

timestamp	address	meta
4789575	0x00580384	(1, 7), R, M
4790083	0x00581d2c	(1, 2), W, H
4790493	0x00580384	(1, 5), R, H

Example: Time sequence analysis of events



VM hardware:
L1 TLB hit-under-miss behavior and L2 TLB latency

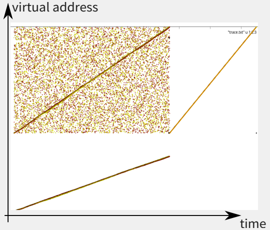


VM software:
Page table walk and TLB entry replacement

Example: Memory access pattern analysis

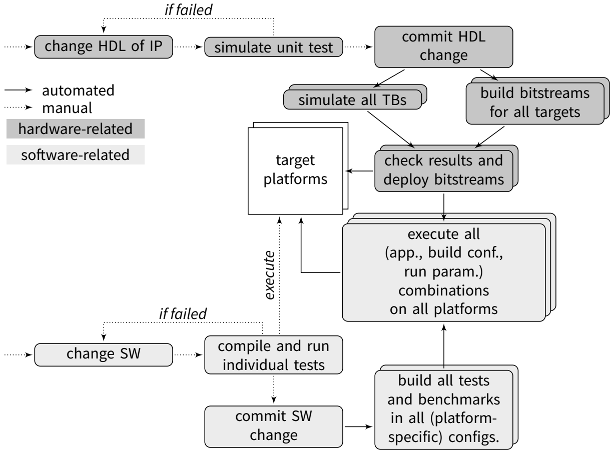
Two phases of a parallel graph processing algorithm are shown; different colors are different PEs.

1. First phase: Two linear traversals and sparse memory accesses in parallel.
2. Second phase: Single linear traversal.



Tools: Automated Builds and Tests

Automated full-system builds and tests are a prerequisite for many effective development paradigms. They are fairly standard for the **hardware** or the **software alone**, but the **combination** is **highly complex on HESoCs**. Our solution is described in the paper.



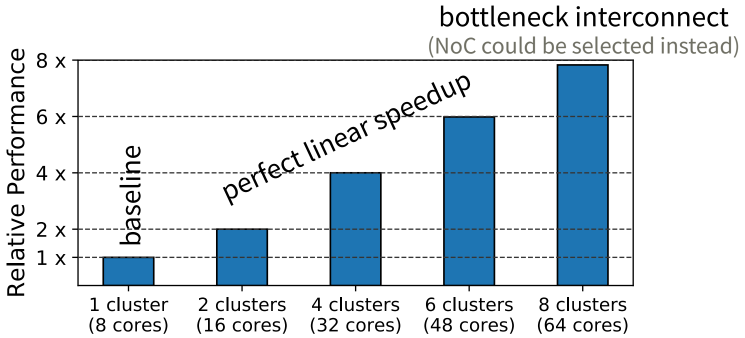
if failed
change HDL of IP
simulate unit test
commit HDL change
automated
manual
hardware-related
software-related
simulate all TBs
build bitstreams for all targets
target platforms
check results and deploy bitstreams
execute all (app., build conf., run param.) combinations on all platforms
execute
if failed
change SW
compile and run individual tests
commit SW change
build all tests and benchmarks in all (platform-specific) configs.

Supported Platforms and Configurations

Property	**ARM Juno** (with a Xilinx Virtex-7 2000T)	**Xilinx Zynq ZC706**
Host CPU	64-bit ARMv8 big.LITTLE	32-bit ARMv7 dual-core A9
Shared main memory	8 GiB DDR3L	1 GiB DDR3
PMCA clock frequency	**31 MHz**	**57 MHz**
# of RISC-V PEs	**64 in 8 clusters**	**8 in 1 cluster**
Integer DSP unit	private per PE	
L1 SPM	256 KiB in 16 banks	
Instruction cache	8 KiB in 8 single-ported banks	4 KiB in 4 multi-ported banks
Slices used by clusters	80%	65%
Slices used by infrastructure	7%	12%
BRAMs used by clusters	89%	70%
BRAMs used by infrastructure	6%	13%
Price	25 000 $	2500 $

Case Study: Parallel Speedup Analysis

- Benchmarking parallel execution and data transfers of the PMCA on the Juno ADP
- Matrix-matrix multiplication $C = AB$
- *A* and *C* are tiled row-wise over the clusters, and each row is parallelized block-wise over the PEs. Data is transferred with DMA bursts, and all PEs operate on data in local SPMs.



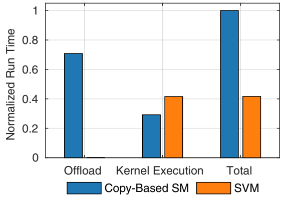
▸ HERO allows to make architectural choices based on **measured results** of benchmarks.

Case Study: Shared Virtual Memory Performance Analysis

The main motivation for shared virtual memory (SVM) is programmability. However, SVM can also significantly improve performance!

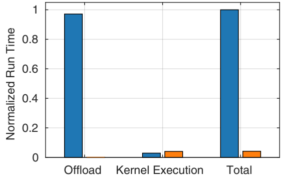
PageRank is a well-known algorithm for analyzing the connectivity of graphs.

- The overhead of manipulating pointers at offload-time in **copy-based offloading** *exceeds* the run-time overhead of translating pointers with **shared virtual memory**.
- In this case, SVM reduces the run time by nearly 60 %.



MemCopy simply copies a large array from DRAM to the PMCA and back, which is representative for streaming applications with little actual work.

- Letting the host **copy data to physically contiguous, uncached memory** is much slower than **letting the PMCA access data directly** with high-bandwidth DMA transfers.
- In this case, SVM reduces the run time by more than 95 %.



▶ HERO allows to back research claims with **reproducible, falsifiable implementation results.**

Conclusion

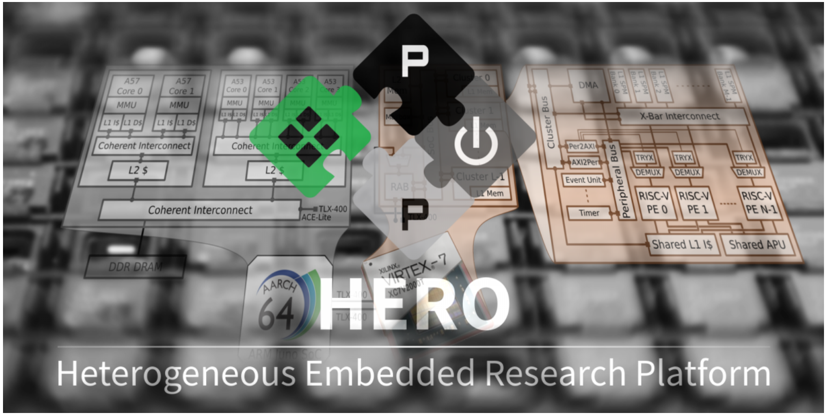
HERO is the **first open-source heterogeneous embedded research platform**. It unites an **ARM Cortex-A host processor** with a **fully modifiable RISC-V manycore implemented on an FPGA**.

HERO enables efficient hardware and software research on HESoCs through

- a **heterogeneous software stack**, which supports **shared virtual memory** and **OpenMP 4.5**—tremendously simplifying porting of standard benchmarks and real-world applications, and
- **profiling and automated verification solutions**.

We have been successfully using HERO in our research over the last years and will continue its development as **open-source hardware and software**!

HERO Will Be Released Open-Source!



Coming Q4 2017

pulp-platform.org/hero