HERO: A Heterogeneous Research Platform to Explore HW/SW Codesign and RISC-V Manycore Accelerators

Luca Bertaccini
<lbertaccini@iis.ee.ethz.ch>
Heterogeneous Systems-on-Chip (HeSoCs)

HOST
- General-purpose
- Linux-capable
- Versatility
- Programmability

PMCA
- Parallel Manycore Accelerator (PMCA)
- Domain-specialized
- Energy-efficient
Energy efficiency challenges (post-Moore era) → Domain Specialization

Low versatility and programmability for highly specialized design → Heterogeneous Systems
Industrial HeSoCs

- Qualcomm Snapdragon 888 (mobile processor – 5nm)
- Apple M1 (MacBook Air, MacBook Pro – 5nm)
- NVIDIA Grace (CPU for AI and HPC – 5nm)
- And many more…
HERO: Overview and Goals

HERO = Heterogeneous Research Platform

Enables research and development on heterogeneous computers:

- Algorithms and Applications
- Programming Models, Task Distribution, Scheduling
- Manycore Architectures, Hardware Accelerators, Core Microarchitecture
- Memory Organization, Communication, Synchronization

Focus of this talk
Integration Host-PMCA
Support for efficient communication
HERO: Hardware Architecture

DMA | TCDM | TCDM | TCDM | TCDM

Local Memory Interconnect

HW Accel | RV32 | RV32 | RV32 | RV32
HERO: Hardware Architecture
HERO: Hardware Architecture

On-Chip Network
- System-Level Cache
- Off-Chip Mem Ctrl

Host CPU
- xy64
- L1 $ MMU
- L1 $ MMU
- L1 $ MMU
- L2 $

Programmable Many-Core Accelerator
- DMA TCDM TCDM TCDM TCDM
- HW Accel RV32 RV32 RV32 RV32
- DMA TCDM TCDM TCDM TCDM
- HW Accel RV32 RV32 RV32 RV32
- DMA TCDM TCDM TCDM TCDM
- HW Accel RV32 RV32 RV32 RV32

International Symposium on Networks-on-Chip (NOCS 2021)
**HERO: Hardware Architecture**

<table>
<thead>
<tr>
<th>On-Chip Network</th>
<th>Host CPU</th>
<th>Programmable Many-Core Accelerator</th>
</tr>
</thead>
<tbody>
<tr>
<td>System-Level Cache</td>
<td>L2 $</td>
<td>DMA TCDM TCDM TCDM TCDM</td>
</tr>
<tr>
<td>Off-Chip Mem Ctrl</td>
<td></td>
<td>Local Memory Interconnect</td>
</tr>
<tr>
<td>I/O</td>
<td></td>
<td>HW Accel RV32 RV32 RV32 RV32</td>
</tr>
</tbody>
</table>

- **DMA**: TCDM TCDM TCDM TCDM
- **Local Memory Interconnect**: 
- **HW Accel**: RV32 RV32 RV32 RV32
- **System-Level Cache**: xy64 xy64 xy64 xy64
- **Off-Chip Mem Ctrl**: L1 $ L1 $ L1 $ L1 $
- **MMU**: MMU MMU MMU MMU
- **L2 $**: 

*International Symposium on Networks-on-Chip (NOCS 2021)*
HERO: Hardware Architecture

Virtual Memory
Coherent Cache

Physically-Addressed
Scratchpad Memory
HERO: Hardware Architecture

Host CPU

Programmable Many-Core Accelerator

DMA TCDM TCDM TCDM TCDM

Local Memory Interconnect

HW Accel RV32 RV32 RV32 RV32

On-Chip Network

DMA TCDM TCDM TCDM TCDM

Local Memory Interconnect

HW Accel RV32 RV32 RV32 RV32

Off-Chip Mem Ctrl

I/O

System-Level Cache

On-Chip Network

ETH Zürich
HERO: Hardware Architecture

Programmable Many-Core Accelerator

DMA TCDM TCDM TCDM TCDM
Local Memory Interconnect

HW Accel RV32 RV32 RV32 RV32

DMA TCDM TCDM TCDM TCDM
Local Memory Interconnect

HW Accel RV32 RV32 RV32 RV32

DMA TCDM TCDM TCDM TCDM
Local Memory Interconnect

HW Accel RV32 RV32 RV32 RV32

xy64 xy64 xy64 xy64
L1 $ L1 $ L1 $ L1 $
MMU MMU MMU MMU
L2 $

On-Chip Network

System-Level Cache

I/O

IOMMU

Off-Chip Mem Ctrl

International Symposium on Networks-on-Chip (NOCS 2021)
Hybrid IOMMU

- To bridge the gap between the different memory systems
- SW-controlled TLB
- TLB prefetching
- Shared Virtual Memory (SVM) accessible by DMA transfers without additional buffers in the IOMMU
PMCA’s DMA issues a transaction that generates a TLB miss

IOMMU responds with an error and drops the transaction

One of the PMCA cores walks the page table, adds the new TLB entry, and notifies the DMA

PMCA’s DMA re-issues the transaction whose TLB miss has been resolved
Worker Threads (WTs) and Prefetching Helper Threads (PHTs). The cores are statically allocated to WTs or PHTs.
Worker Threads (WTs) and Prefetching Helper Threads (PHTs). The cores are statically allocated to WTs or PHTs.
HERO: TLB prefetching

The PHTs are automatically generated by the compiler which checks for SVM accesses in the code.
The PHTs are automatically generated by the compiler which checks for SVM accesses in the code.

**WTs** contain additional store instructions to the L1 SPM to share the execution state while **PHTs** contain additional load instructions.
The PHTs are automatically generated by the compiler which checks for SVM accesses in the code.

**WTs** contain additional store instructions to the L1 SPM to share the execution state while **PHTs** contain additional load instructions.
HERO: TLB prefetching

The prefetch method informs the TLB miss handlers that a TLB must be set up ahead of the moment when a worker requires the data on a page.
SW stack
Efficient offloading
HERO: Software Stack

User-Space Software
- Heterogeneous Application
- Accelerated Kernel
- Application Programming Interface
- OpenMP Runtime Lib
- Accel Lib

Kernel-Space Software
- Accelerator Driver
- Linux Kernel
- Virt Mem Mgmt Lib
- Hardware Abstraction Library

HW
- Host
- Accelerator
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

- It enables zero-copy offloads, directly passing **virtual pointers** to the PMCA
HERO: Programming Model and API

Principle: single-source heterogeneous programming. Example:

```c
void host_function(unsigned n_elems, const float a, const float* x, float* y) {
    #pragma omp target map(to: n_elems, a, x, y) device(HERO_DEVICE_PULP)
    {
        float buf_x[BUF_ELEMS], buf_y[BUF_ELEMS];
        for (unsigned offset = 0; offset < n_elems; offset += BUF_ELEMS) {
            const unsigned cur_n_elems = min(n_elems - offset, BUF_ELEMS);
            const size_t cur_memcpy_size = cur_n_elems * sizeof(float);
            hero_memcpy_host2dev(buf_x, x+offset, cur_memcpy_size);
            hero_memcpy_host2dev(buf_y, y+offset, cur_memcpy_size);
            hero_dblas_saxpy(cur_n_elems, a, buf_x, buf_y);
            hero_memcpy_dev2host(y+offset, buf_y, cur_memcpy_size);
        }
    }
}```
HERO: Heterogeneous Compilation

Single-source, single-binary heterogeneous compilation ...

Single Source, Common API, Specialized Code

Separate data models
Separate ISAs
Separate Libraries

(width of pointers,
size_t, long, ...

Single Binary, Nested Executables

... provides first-class support for heterogeneous programming!

International Symposium on Networks-on-Chip (NOCS 2021)
FPGA implementations
Results
HERO on FPGA

HERO implementations have been deployed on different FPGA platforms:

• Zynq UltraScale+ MPSoC ZCU102 (Xilinx)
• Zynq ZC706 Evaluation Kit (Xilinx)
• Juno development board (Arm)
• Virtex UltraScale+ HBM VCU128 (Xilinx)

Larger programmable logic available on the FPGA enables research on larger PMCAs or multi-cluster PMCAs.
Results: Zero-Copy vs. Copy-Based

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

**PageRank** (algorithm to analyze graph connectivity):
- The overhead of manipulating pointers at offload-time in the **copy-based** approach is higher than the overhead introduced by translating pointer with **SVM**

**MemCopy** (Copy a large array from DRAM to PMCA and back)
- The host copy phase takes much more time than letting the PMCA access data directly with high-bandwidth DMA transfers
HERO: Ongoing efforts

HOSTs

- ARM
- ARM
- Ariane
- Ariane

PMCAs

- PULP
- Snitch
- PULP
- Snitch
Conclusion

• **HERO** is a research platform to explore HW/SW codesign of heterogeneous systems

• HERO enables full-system exploration of RISC-V manycore accelerators

• HERO achieves **efficient collaboration** between host and PMCA through a shared virtual memory enabled by its hybrid IOMMU

• OpenMP plugin allows for **transparent accelerator programming**
Thank you for your attention!