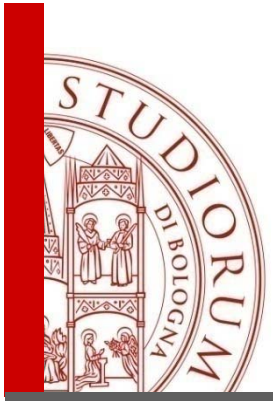


ETH zürich
Integrated Systems Laboratory

PULP
Parallel Ultra Low Power

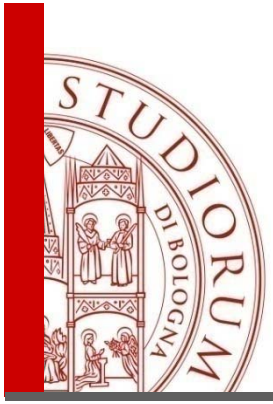
Multitherman

ERC GRANT N° 291125

IWES17
September 07-08, 2017,
Rome (Italy)

ENABLING LOW-COST AND LIGHTWEIGHT ZERO-COPY OFFLOADING ON HETEROGENEOUS MANY-CORE ACCELERATORS: THE PULP EXPERIENCE

Alessandro Capotondi (alessandro.capotondi@unibo.it)
Andrea Marongiu
Luca Benini
University of Bologna



ETH zürich
Integrated Systems Laboratory

PULP
Parallel Ultra Low Power

Multitherman

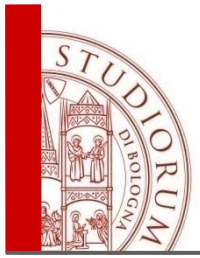
ERC GRANT N° 291125

IWES17
September 07-08, 2017,
Rome (Italy)

ENABLING LOW-COST AND LIGHTWEIGHT ZERO-COPY OFFLOADING ON HETEROGENEOUS MANY-CORE ACCELERATORS: THE PULP EXPERIENCE

TLTR: Low-cost Unified Virtual Memory Support on Embedded SoC

Alessandro Capotondi (*alessandro.capotondi@unibo.it*)
Andrea Marongiu
Luca Benini
University of Bologna



Heterogenous Manycores

Ever-increasing demand for computational power has recently led to radical evolution of computer architectures

Two design paradigms have proven effective in increasing performance and energy efficiency of compute systems

> Many-cores

> Architectural **Heterogeneity**

A common template is one where a powerful general-purpose processor (the ***host***) is coupled to ***one or more a many-core accelerators***

Heterogenous Manycores

Gyoukou

Xeon D-1571
16C 1.3Ghz
Infiniband EDR
PEZY-SC2

Titan Cray X47

Opteron 6274
16C 2.2GHz
Cray Gemini
NVIDIA K20x

Tianhe-2

Xeon E5-2692
12C 2.2GHz
TH Express-2
Intel Xeon Phi

HPC / SERVER

True in every computing domain and at every scale!

Heterogenous Manycores

HPC / SERVER

SoC

THE GREEN 500

TOP 500 The List.

Gyoukou

Xeon D-1571
16C 1.3Ghz
Infiniband EDR
PEZY-SC2

Titan Cray X47

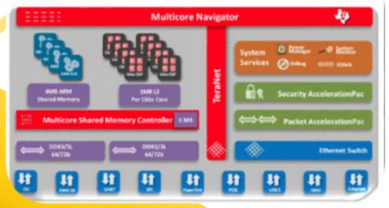
Opteron 6274
16C 2.2GHz
Cray Gemini
NVIDIA K20x

Tianhe-2

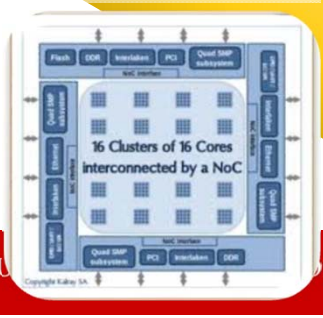
Xeon E5-2692
12C 2.2GHz
TH Express-2
Intel Xeon Phi

TI Keystonell

66AK2H12 system-on-chip

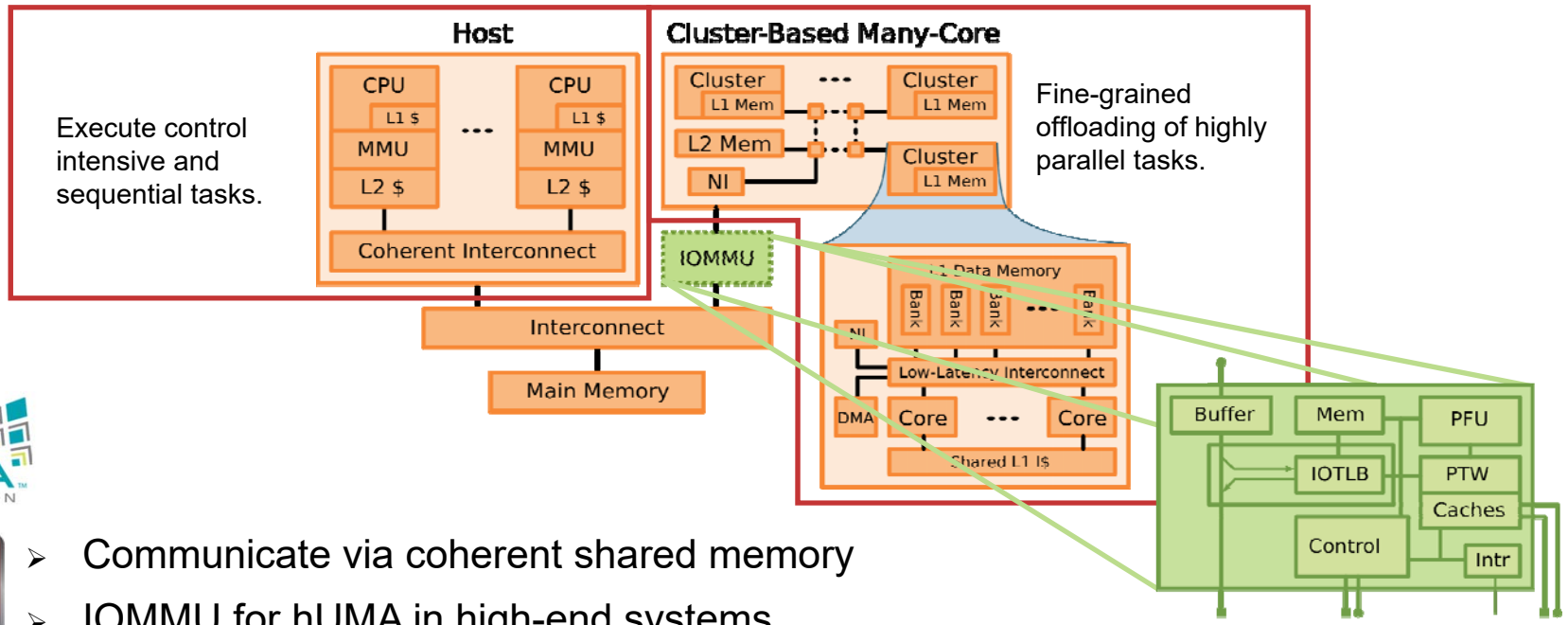


Kalray MPPA256



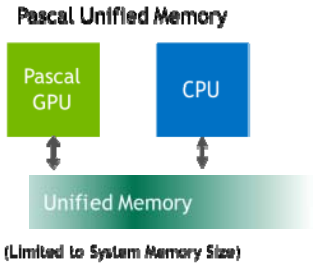
NVIDIA Tegra X1

Heterogenous Manycores

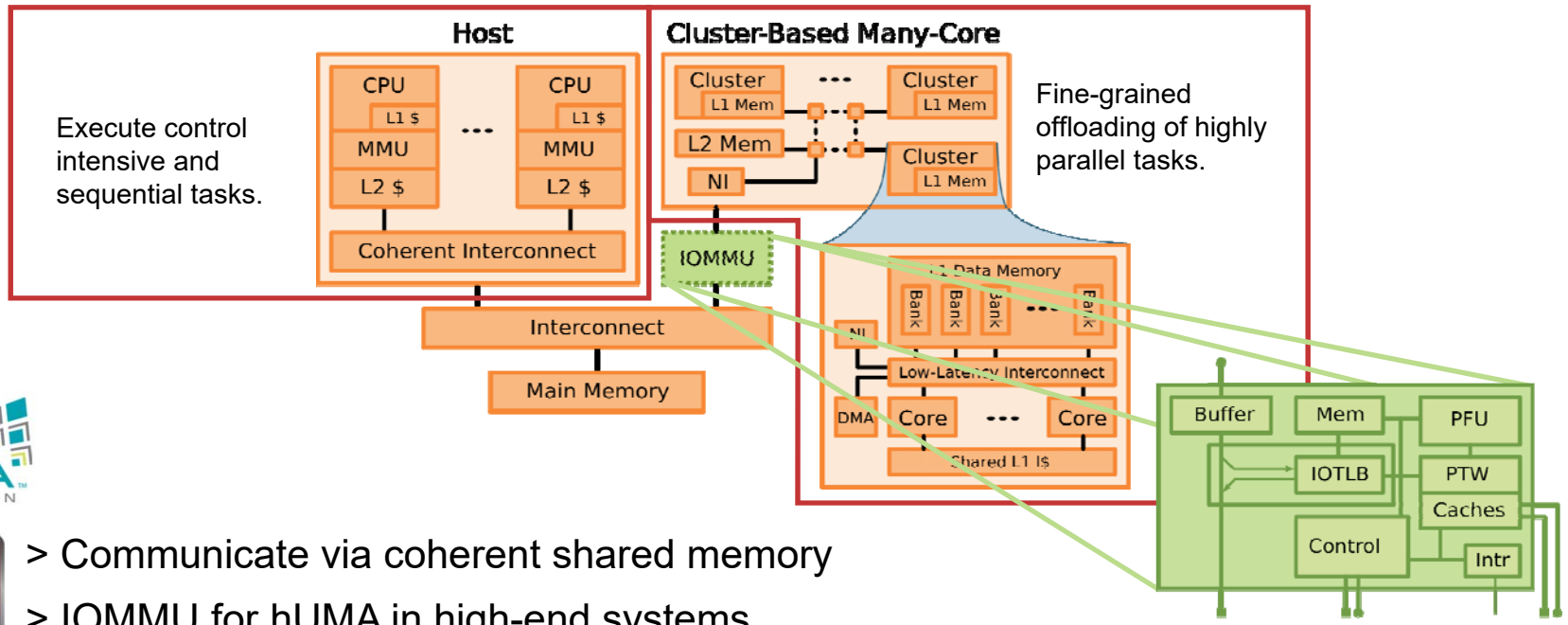


- Communicate via coherent shared memory
- IOMMU for hUMA in high-end systems

- CUDA 6 Unified Virtual Memory
- Pascal Architecture and Tegra X series



Heterogenous Manycores



HSA FOUNDATION

ACCELERATED A-SERIES PROCESSOR AMD

> Communicate via coherent shared memory

> IOMMU for hUMA in high-end systems

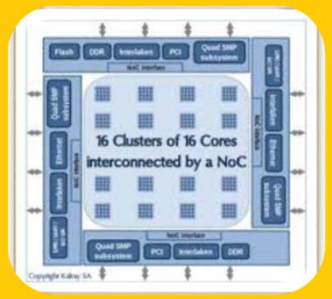
What about low-power, embedded systems?

Embedded Heterogenous SoCs

Adapteva

parallella

Kalray MPPA256



STHORM

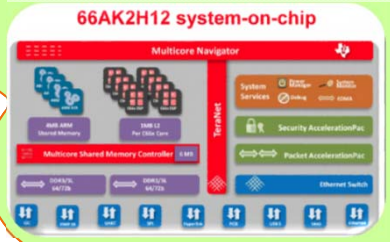
life.augmented

Many-Core Accelerators

DSP/ASIC/FPGA Accelerators

Altera Arria

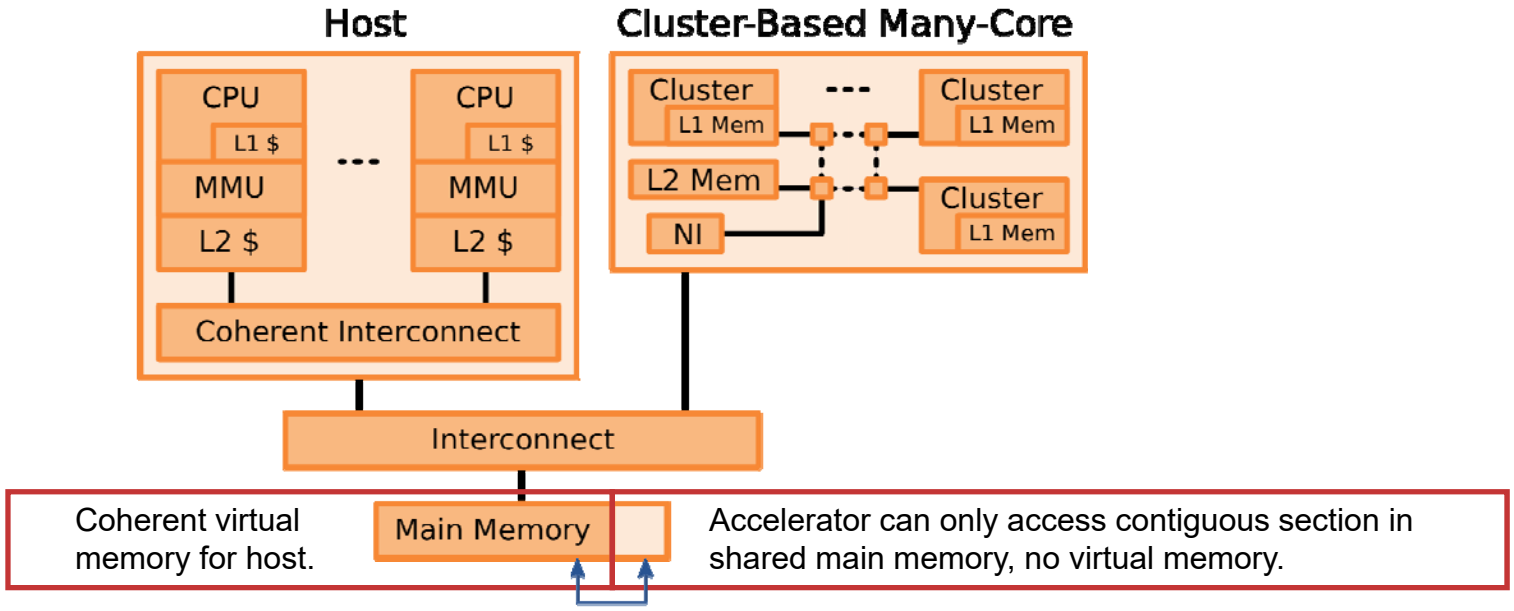
TI Keystonell



Xilinx Zynq

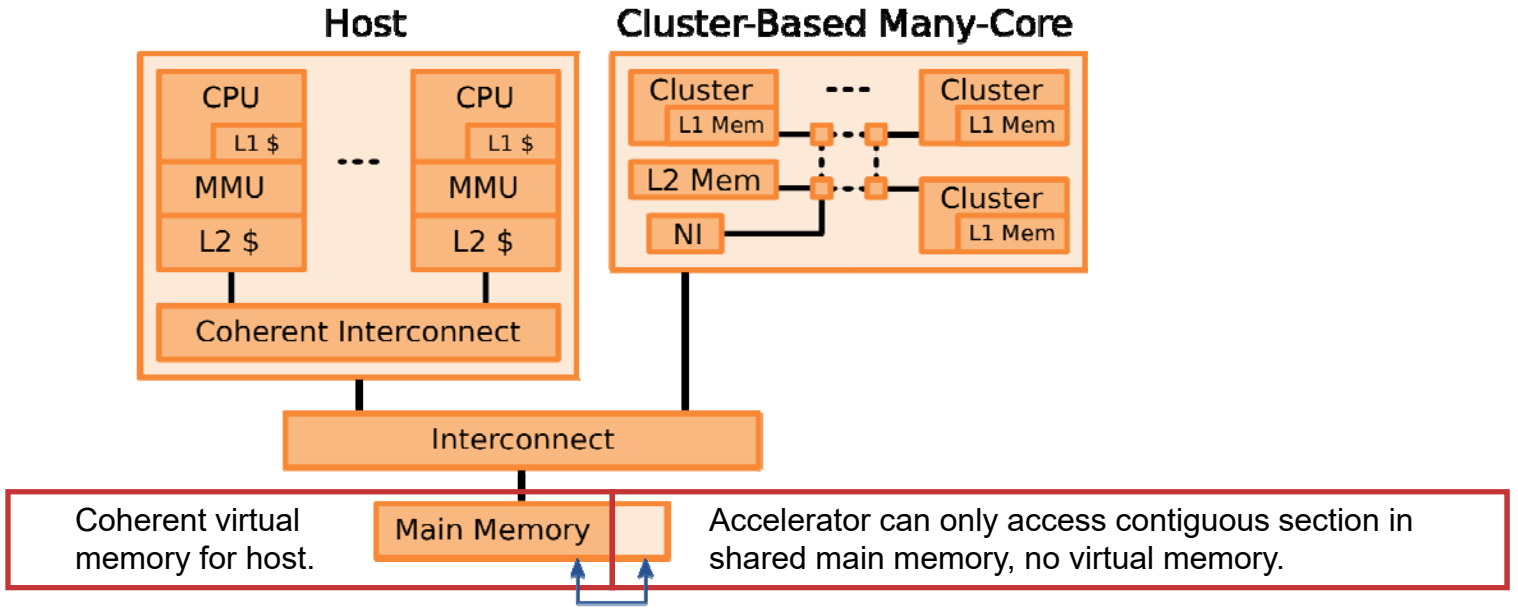
Embedded Heterogenous SoCs

copy-based approach



Embedded Heterogenous SoCs

copy-based approach



Pros

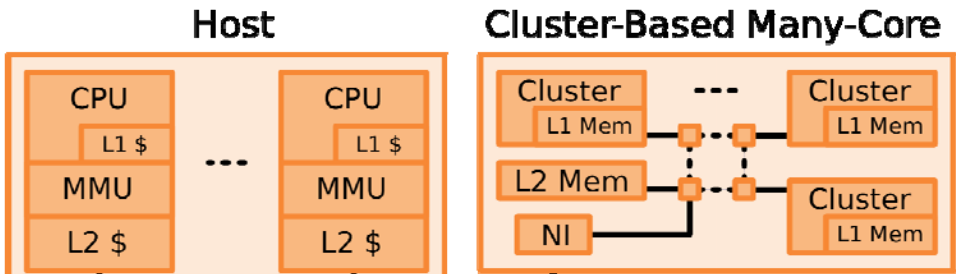
- Do not require specific HW
- Cheap and low-power

Cons

- Overheads for copying data from/to the dedicated memory
- Complex data structures require ad-hoc transfer
- Performance issue on not-paged sections

Embedded Heterogenous SoCs

copy-based approach



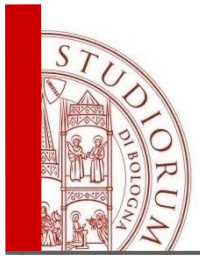
PageRank (10k vertices)	Execution Time	# Lines of Source Code
Host Implementation	18.9 ms	79
Copy & Translate	14.7 ms	71
Overhead	78 %	90 %

Pros

- Do not require specific HW
- Cheap and low-power

Cons

- Overheads for copying data from/to the dedicated memory
- Complex data structures require ad-hoc transfer
- Performance issue on not-paged sections



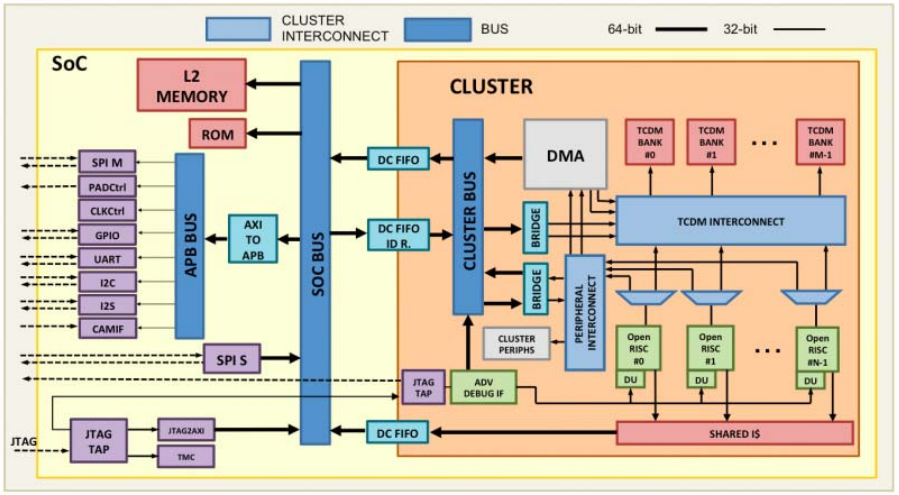
Contributions

- **Lightweight mixed HW/SW managed IOMMU for UVM support**
  - *PULP architecture*
  - *IOMMU Implementation*
- **GNU GCC Toolchain Extensions for offloading to PULP accelerator**
  - *Compiler Extensions*
  - *Runtime/Libraries Extensions*
- **UVM Experimental evaluation on OpenMP offloading**

PULP - An Open Parallel Ultra-Low-Power Processing-Platform

This is a joint project between the Integrated Systems Laboratory (IIS) of ETH Zurich and the Energy-efficient Embedded Systems (EEES) group of UNIBO to develop an **open**, **scalable Hardware and Software** research platform with the goal to break the pJ/op barrier within a power envelope of a few *mW*.

The **PULP platform** is a multi-core platform achieving leading-edge energy-efficiency and featuring widely-tunable performance.



cluster-based

scalable

silicon-proven

OpenRISC/RISC-V

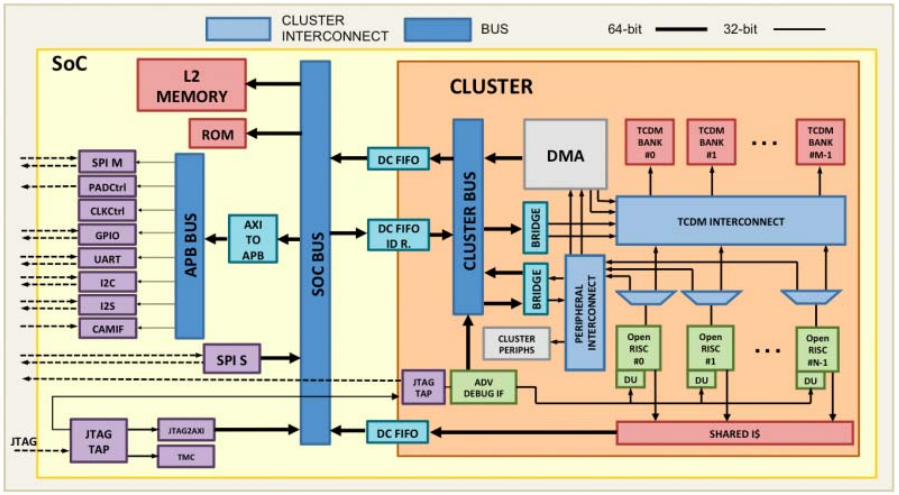
PULP
Parallel Ultra Low Power

not only ULP power envelop!

PULP - An Open Parallel Ultra-Low-Power Processing-Platform

This is a joint project between the Integrated Systems Laboratory (IIS) of ETH Zurich and the Energy-efficient Embedded Systems (EEES) group of UNIBO to develop an **open**, **scalable Hardware and Software** research platform with the goal to break the pJ/op barrier within a power envelope of a few *mW*.

The **PULP platform** is a multi-core platform achieving leading-edge energy-efficiency and featuring widely-tunable performance.



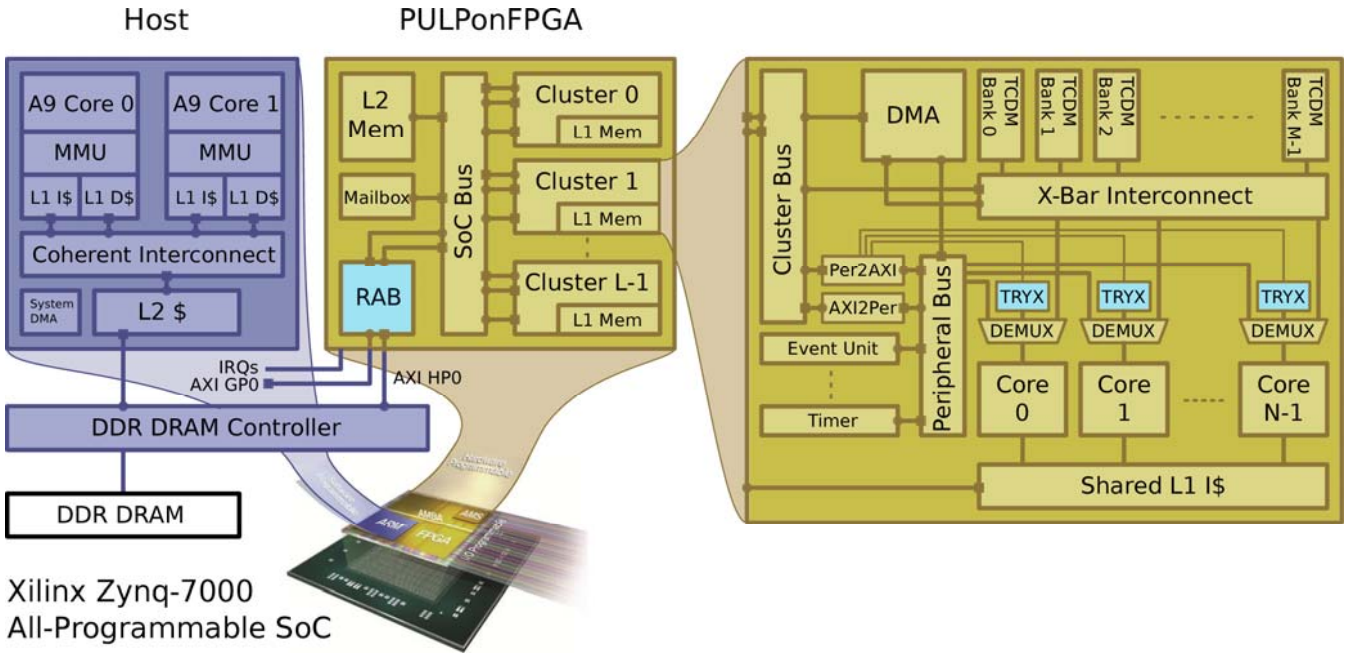
cluster-based

scalable

silicon-proven

OpenRISC/RISC-V

PULP as heterogeneous programmable accelerator emulator



Host: Dual-Core ARM Cortex-A9 running full fledged Ubuntu 16.04

Accelerator: 8 core – PULP Fulmine cluster (www.pulp-platform.org)

Lightweight UVM
Unified Virtual Memory

Goals:

- **Sharing of virtual address** pointers
- **Transparent** to application developer
- **Zero-copy offload**, performance predictability
- **Low complexity**, low area, low cost
- **Non-intrusive** to accelerator architecture

Mixed Hardware/Software Solution:

> Input/output translation lookaside buffer (IOTLB)

> Special-purpose *TRYX* Control register

Requires:

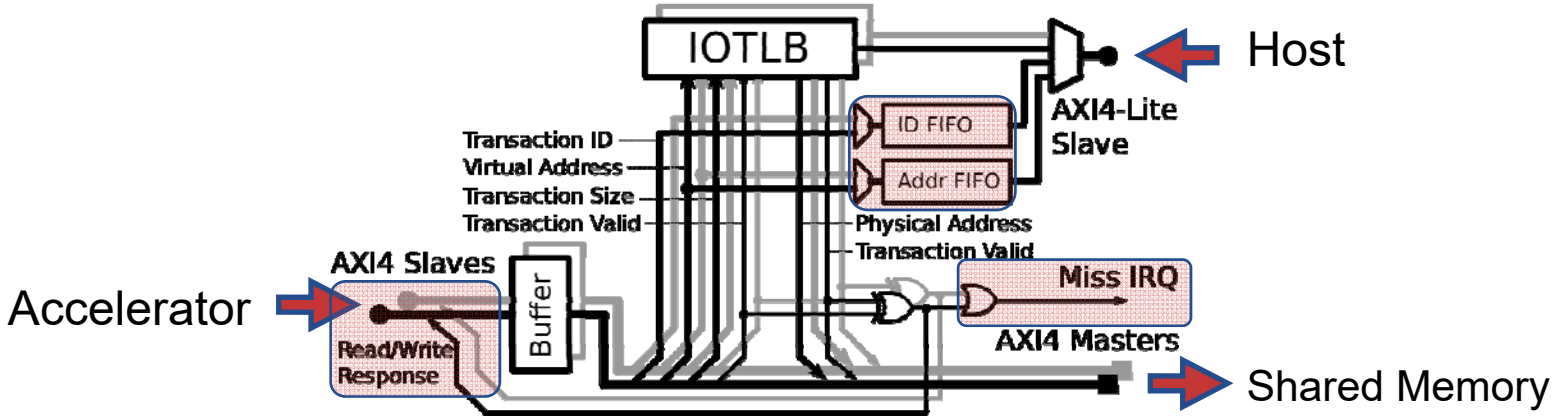
> Compiler extension to insert *tryread/trywrite* operation

> Kernel-level driver module

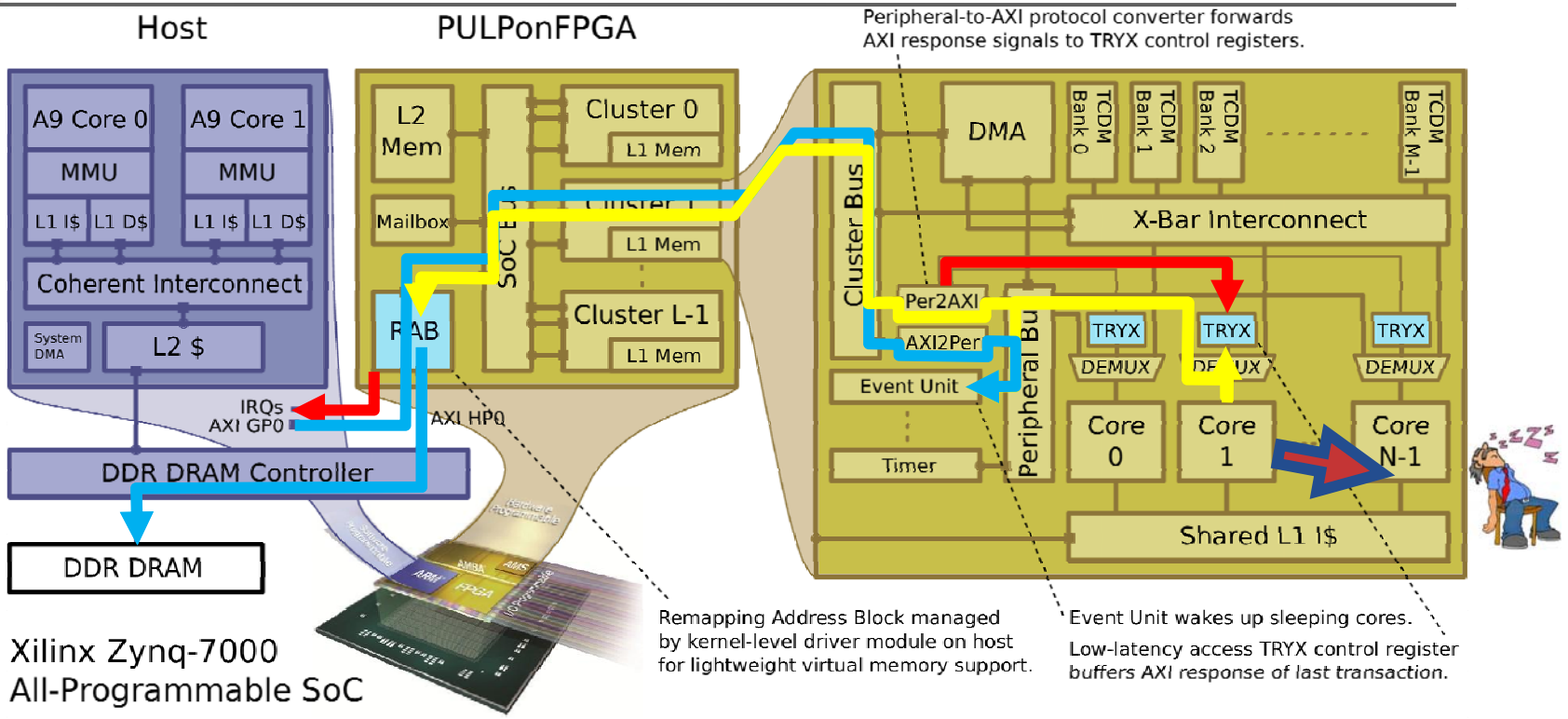
Remapping Address Block (RAB):

> Virtual-to-physical address translation

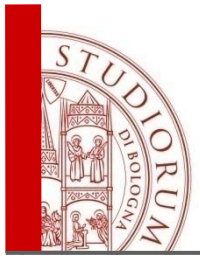
> Per-port private IOTLBs, shared configuration interface



Lightweight UVM
Unified Virtual Memory



- **No hardware modifications** to the processing elements.
- **Portable RAB miss handling** routine on the host.
- Optimized for common case: **overhead of 8 cycles.**



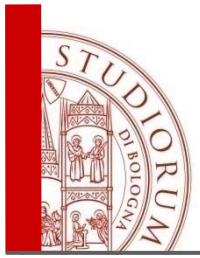
OpenMP

- De-facto standard for shared memory programming
- Support for nested (multi-level) parallelism → *good for clusters*
- Annotations to incrementally convey parallelism to the compiler → *increased ease of use*
- Based on well-understood programming practices (shared memory, C language) → *increases productivity*

	MM		LU		CONV		PI		Histogram	
	T/E	LOC	T/E	LOC	T/E	LOC	T/E	LOC	T/E	LOC
OpenMP	+++	45	++	60	+++	70	+++	32	+	28
TBB	++	45	- - -	59	+	53	+	42	-	58
OpenCL	++	120/108	- -	120/225	-	120/186	-	120/128	- - -	120/150

Table 1: Time/effort (T/E) and number of lines of code for given bench- marks for the three frameworks. MM(Matrix Multiplication), LU(LU Factorization), CONV(Image convolution). [+++ : Least time/effort - - - : Most time/effort]

"OpenCL for programming shared memory multicore CPUs" by Akhtar Ali , Usman Dastgeer , Christoph Kessler

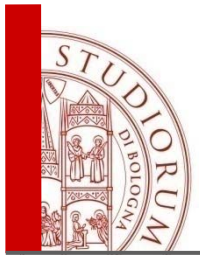


OpenMP

- De-facto standard for shared memory programming
- Support for nested (multi-level) parallelism → *good for clusters*
- Annotations to incrementally convey parallelism to the compiler → *increased ease of use*
- Based on well-understood programming practices (shared memory, C language) → *increases productivity*
- *Since Specification 4.0 OpenMP support Heterogenous Execution Model based on offloads!*

At the moment **GCC** supports OpenMP offloading **ONLY** to:

- **Intel Xeon Phi**
- **Nvidia PTX** (only through **OpenACC**)



OpenMP **target** example

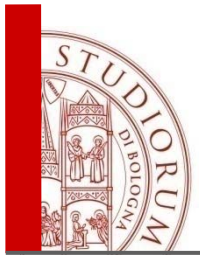
```
void vec_mult()
{
  double p[N], v1[N], v2[N];

# pragma omp target map(to: v1, v2)\
                    map(from: p)
  {
#   pragma omp parallel for
    for (int i = 0; i < N; i++)
      p[i] = v1[i] * v2[i];
  }
}
```

1. Initialize target device
2. Offload target image
3. Map **TO** the device mem
4. Trigger execution target region
5. Wait termination
6. Map **FROM** the device mem

The ***compiler*** outlines the code within the target region and generates a binary version for each accelerator (multi-ISA)

The ***runtime*** libraries are in charge to:

- ***manage*** the accelerator devices
- ***map*** the variables
- ***run/wait*** execution of target regions

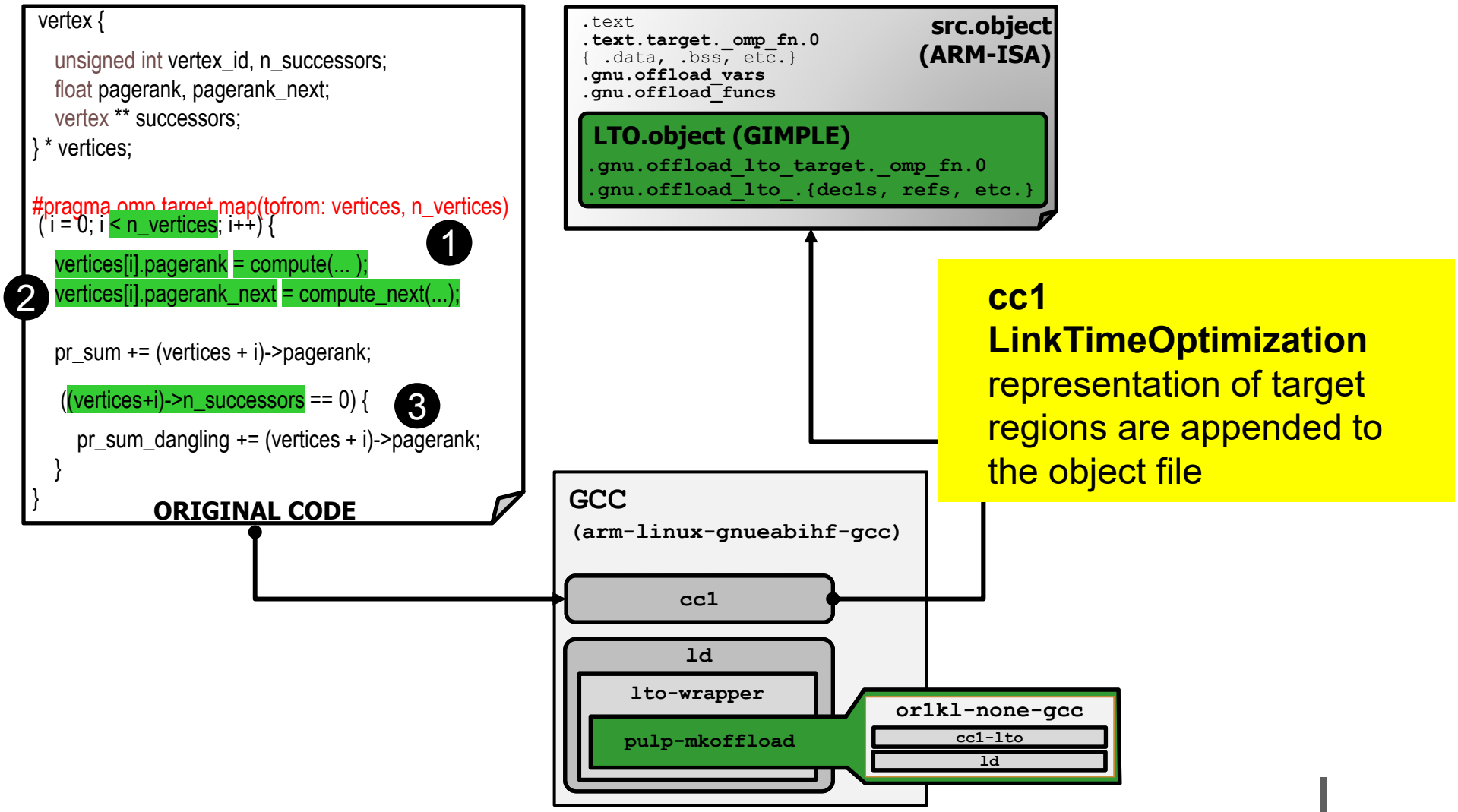


GNU GCC - Extensions

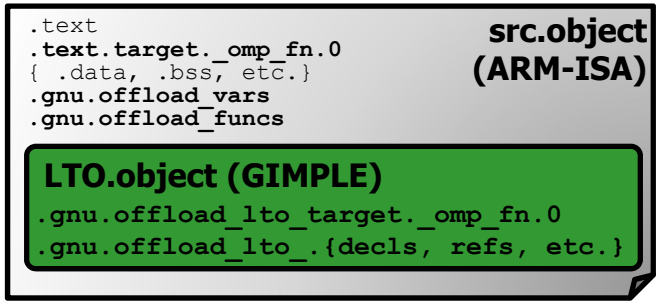
- **Added PULP as target accelerator**
  - Enabled OpenRISC back-end as OpenMP4 accelerator supported ISA
  - Created *ad-hoc lto-wrapped* linker tool for PULP offloaded region (***pulp-mkoffload***)

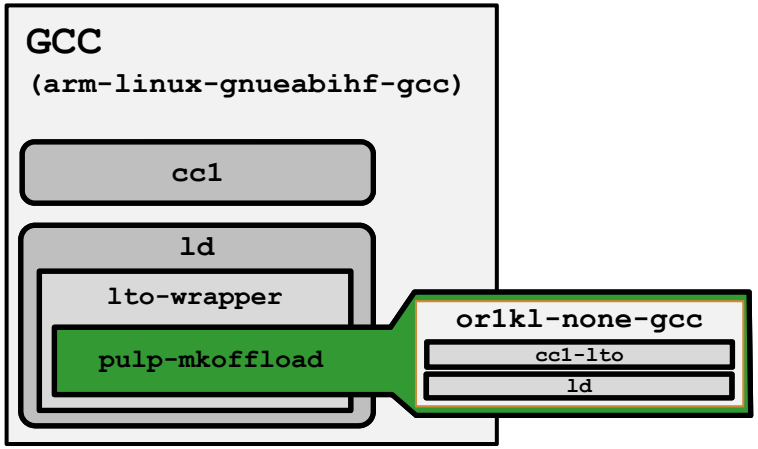
- **Enabled UVM (zero-copy) support for PULP**
  - Added new SSA pass to protect usage of shared mapped variables between the accelerator and the host

Added PULP as target accelerator (1)



Added PULP as target accelerator (2)

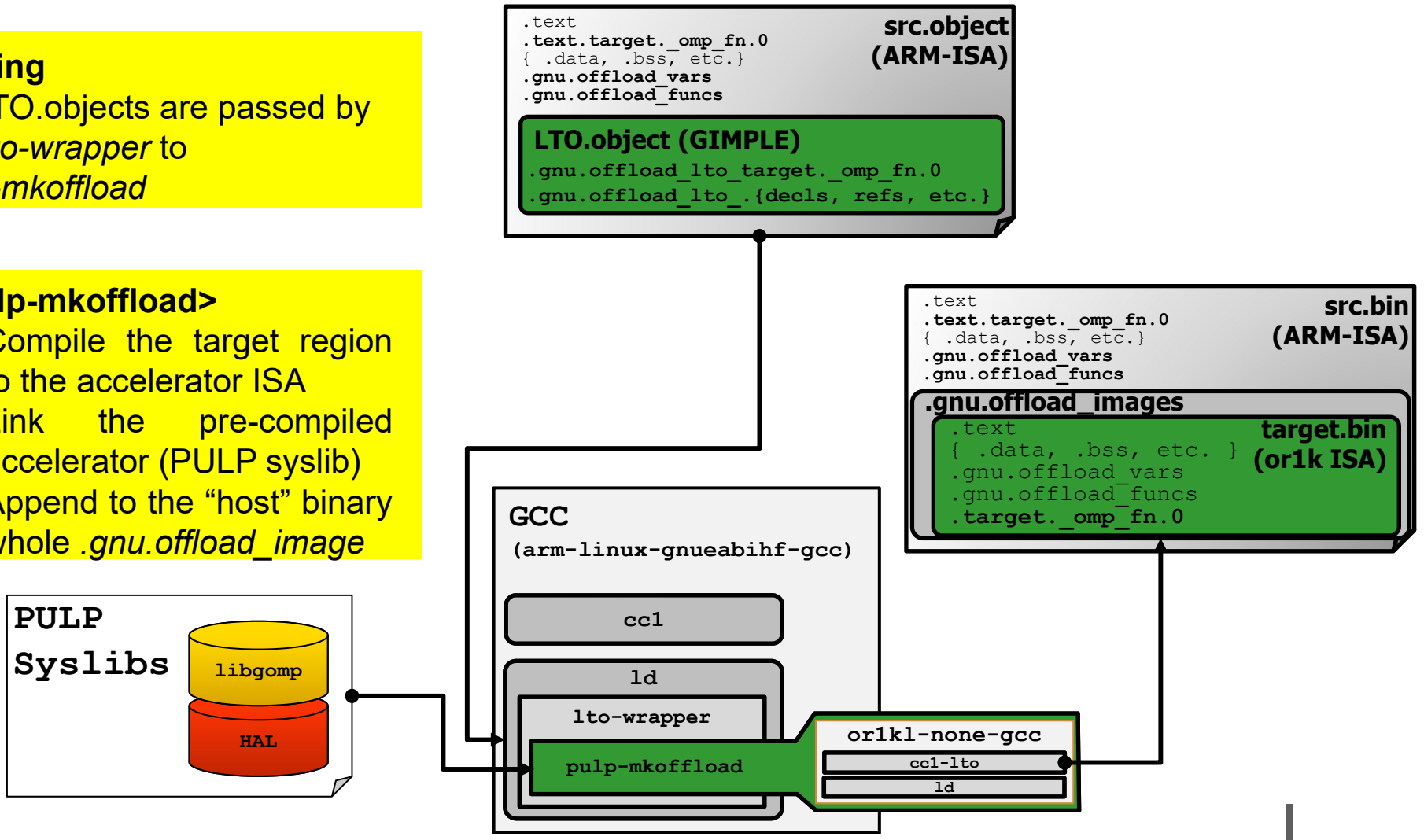




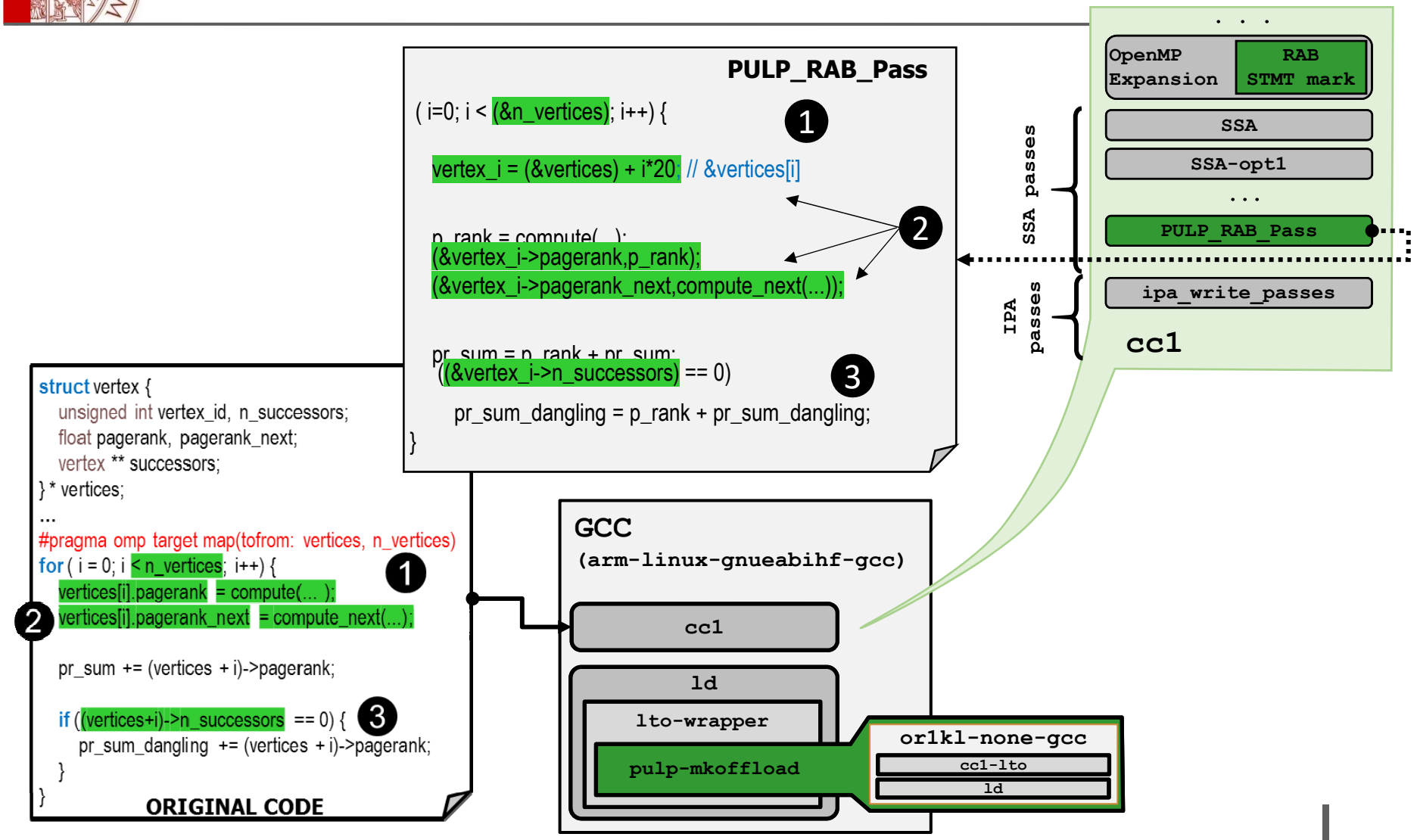
Added PULP as target accelerator (2)

Linking
All LTO.objects are passed by the *lto-wrapper* to *pulp-mkoffload*

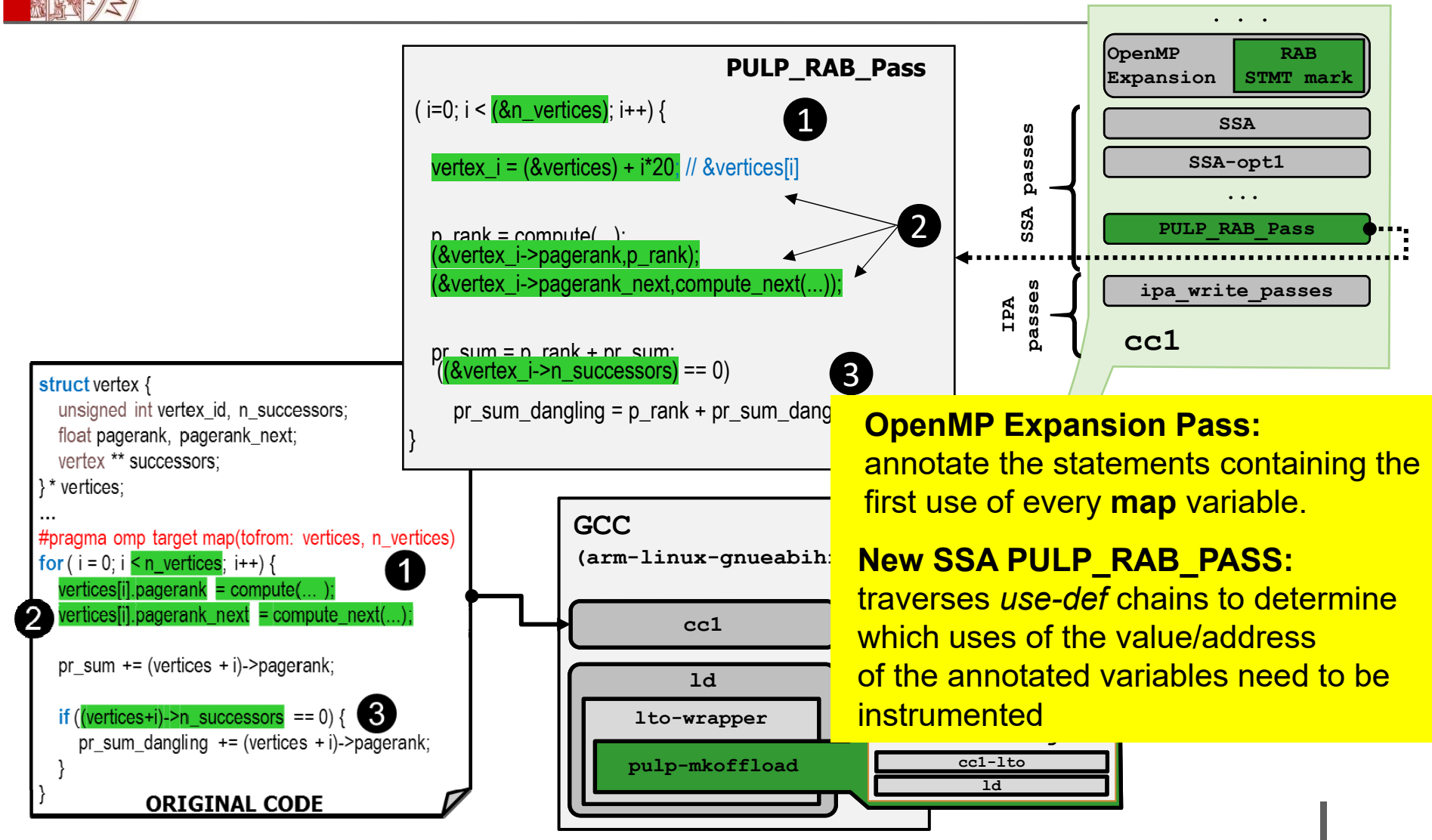
<pulp-mkoffload>
- Compile the target region to the accelerator ISA
- Link the pre-compiled accelerator (PULP syslib)
- Append to the "host" binary whole *.gnu.offload_image*



Compiler UVM support for PULP



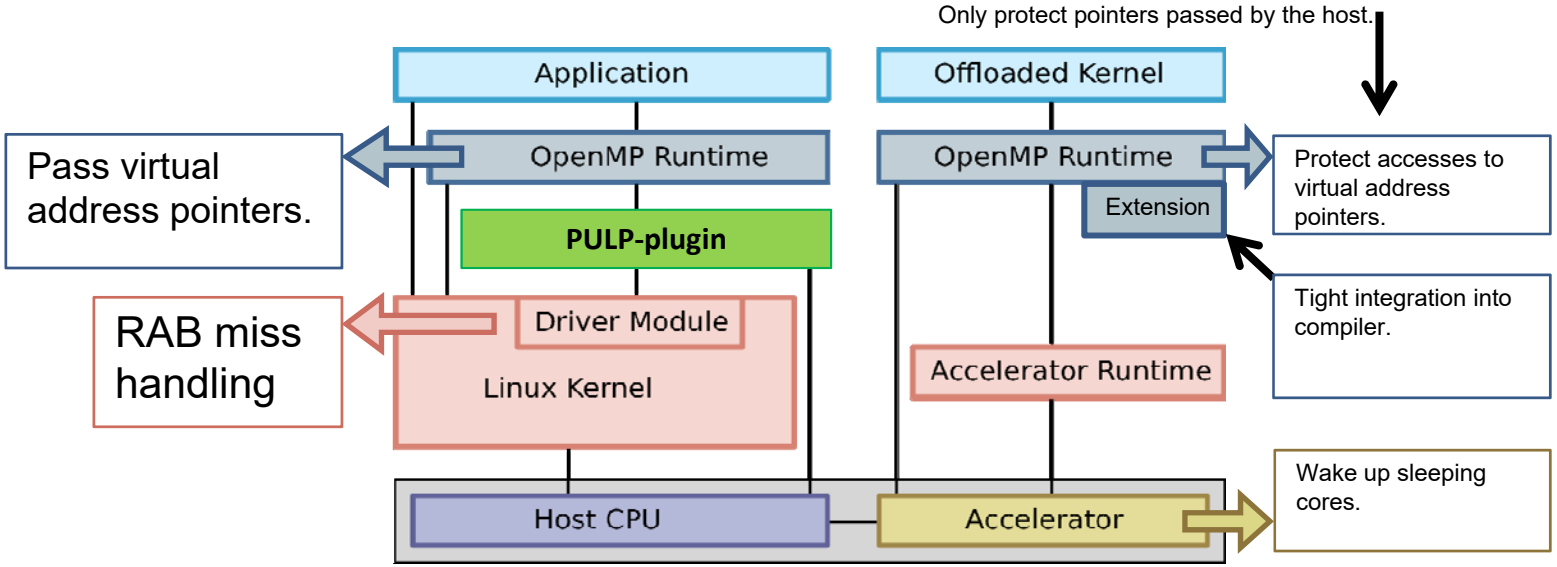
Compiler UVM support for PULP

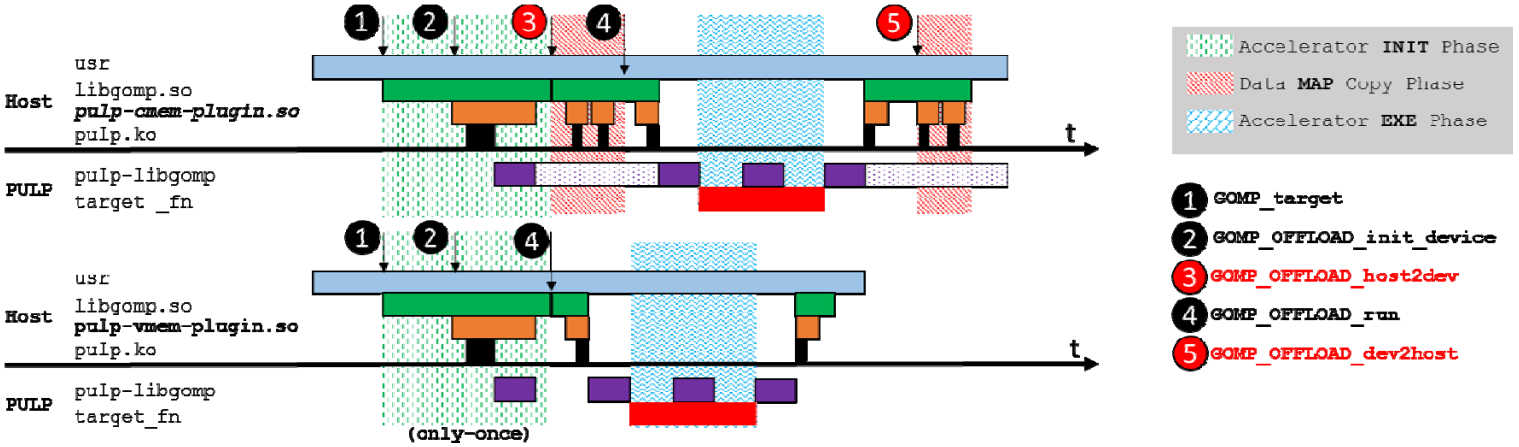


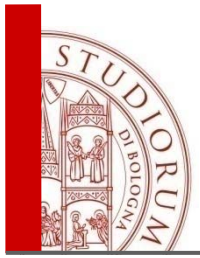
OpenMP Expansion Pass:
annotate the statements containing the first use of every **map** variable.

New SSA PULP_RAB_PASS:
traverses *use-def* chains to determine which uses of the value/address of the annotated variables need to be instrumented

Full SW stack overview







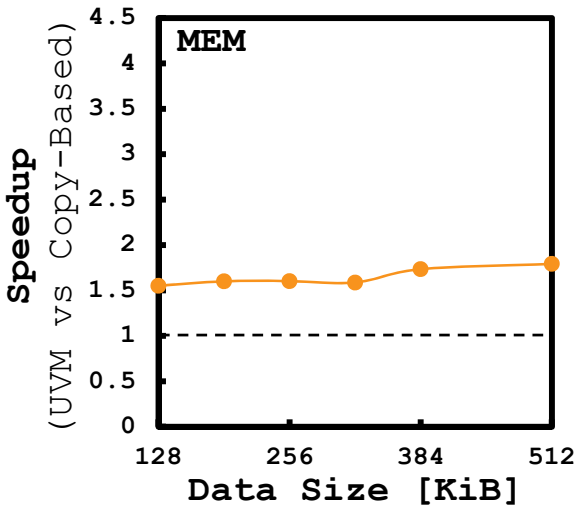
Experimental setup

Objective: while UVM's greatest advantage is simplified programmability we want evaluate the advantage of UVM on performance.

Benchmarks:

- ***memcpy* (MEM*):*** representative example for heavily memory-bound, streaming applications with regular access pattern to shared memory.
- ***pointer chasing (PC):*** is representative of graph-processing applications with highly irregular access patterns, like Page-Rank, Breadth-First Search, clustering, Nearest Neighbor Search.
- ***random forest traversal (RFT):*** is representative of irregular applications for regression, classification problem solving, and pattern recognition

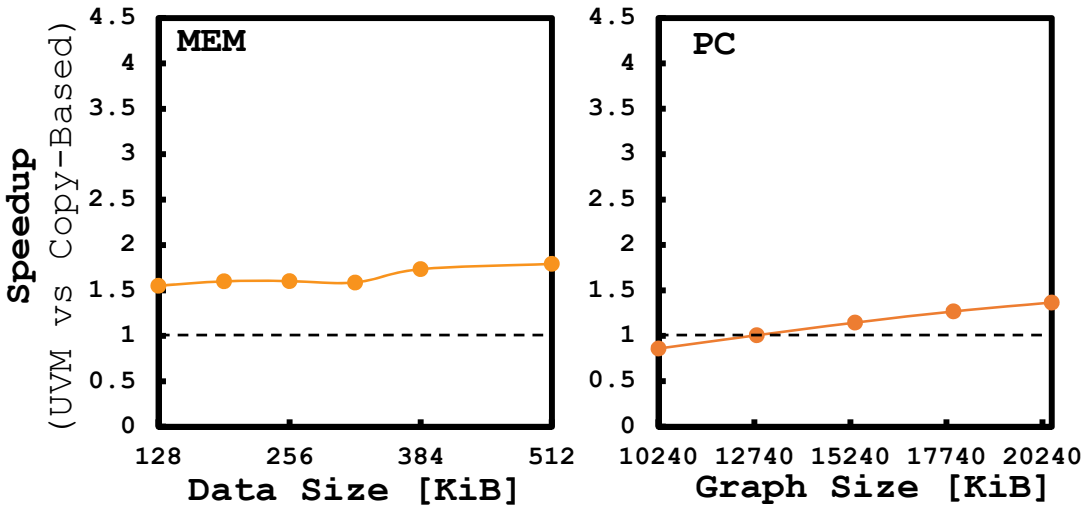
Results (1)



On regular applications UVM executes avg. 1.6× faster.

Capacity RAB misses when the data size exceeds the TLB capacity limits the speedup at 1.79×

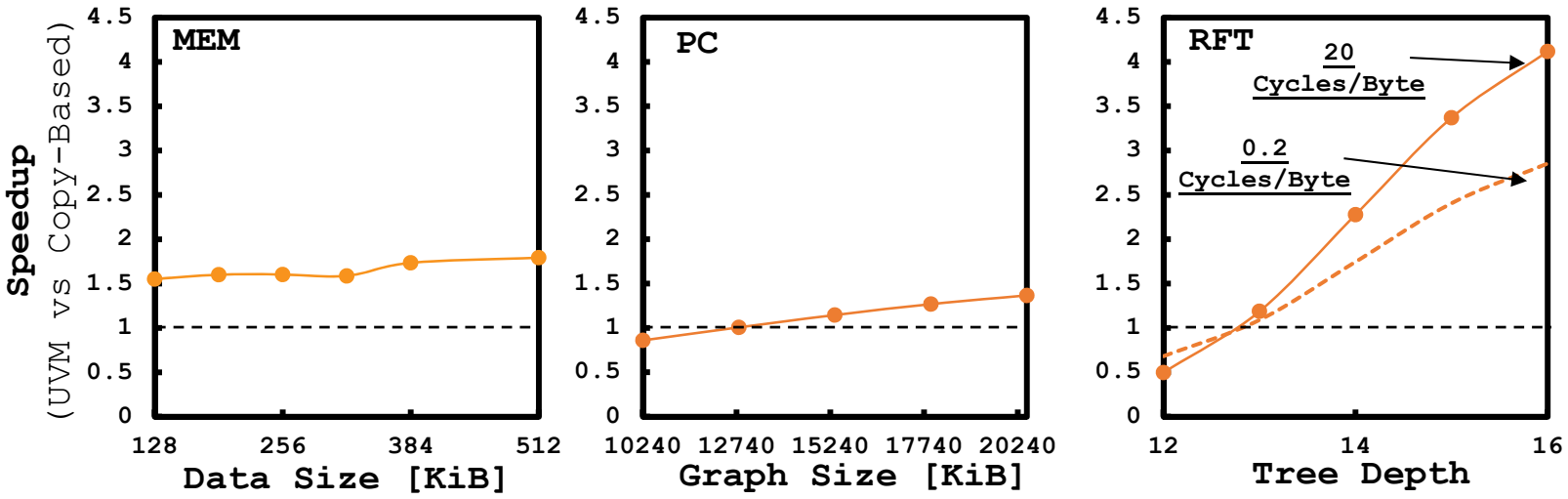
Results (2)



PC shows a slowly but steadily increasing speedup (up to 1.4× for the considered data sets)

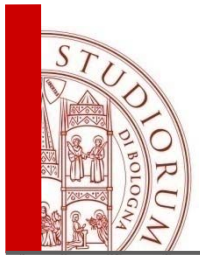
Small graphs are penalized by the higher RAB handling costs compared to regular applications like MEM

Results (3)



RFT reaches 4.11× and 2.85× speedup, for CCRs equal to 20 and 0.2, respectively

The higher speedups are due to the fact that in copy-based a lot of data is copied that is (potentially) never accessed.



Conclusion

We presented a **RTL-proven mixed HW/SW lightweight IOMMU** for low-power embedded many-core accelerator.

We presented **a full implementation of OpenMP 4 on GCC** for PULP architecture.

We **extended the toolchain at compiler and runtime level** to enable Unified Virtual Memory support achieved by a low-cost, low-area, IOTLB infrastructure.

UVM enables, with **smaller programming effort**, a **performance gain** compare **standard copy-based** offloading mechanisms.

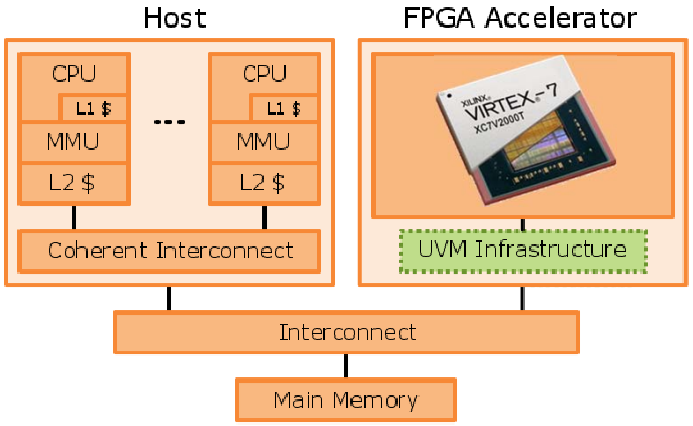
Conclusion (2)

Current status:

- Make the first OpenMP-ready, RISC-V accelerator!
- bring UVM support to FPGA accelerators (custom or HLS flow, SDSoC, ecc…)

Looking ahead

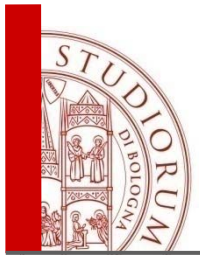
- **release Open-Source (near future)**
- Looking at ultra large scale of accelerator (tens, hundreds clusters)



PULP
Parallel Ultra Low Power

http://www.pulp-platform.org/

Contact us! If you are interested to use it as research platform or to join as collaborator!



ETH zürich
Integrated Systems Laboratory

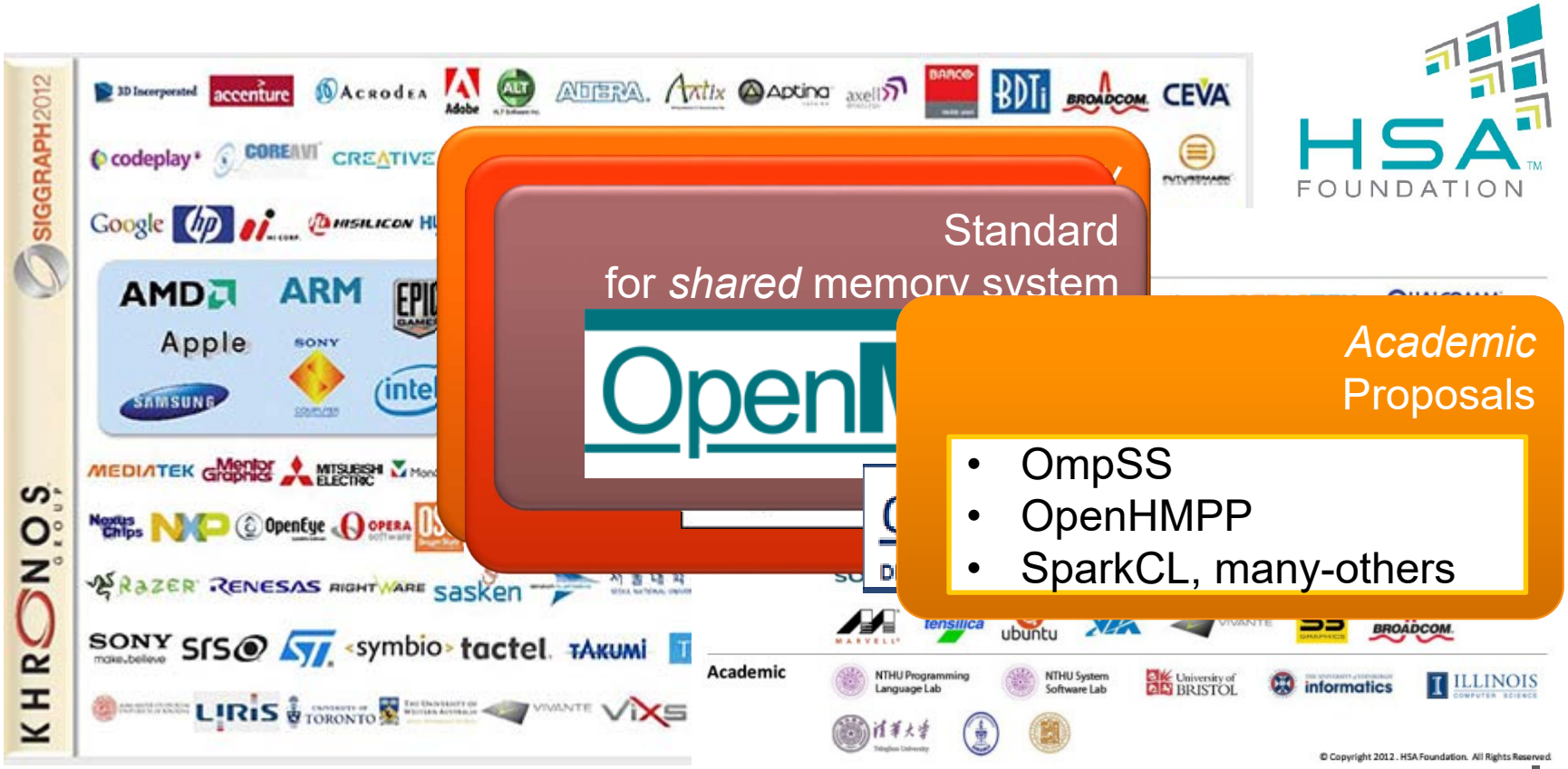
Work supported by
Multitherman
ERC GRANT N° 291125

Thank You!

Questions? Answers?

Alessandro Capotondi (alessandro.capotondi@unibo.it)

How many parallel programming models?



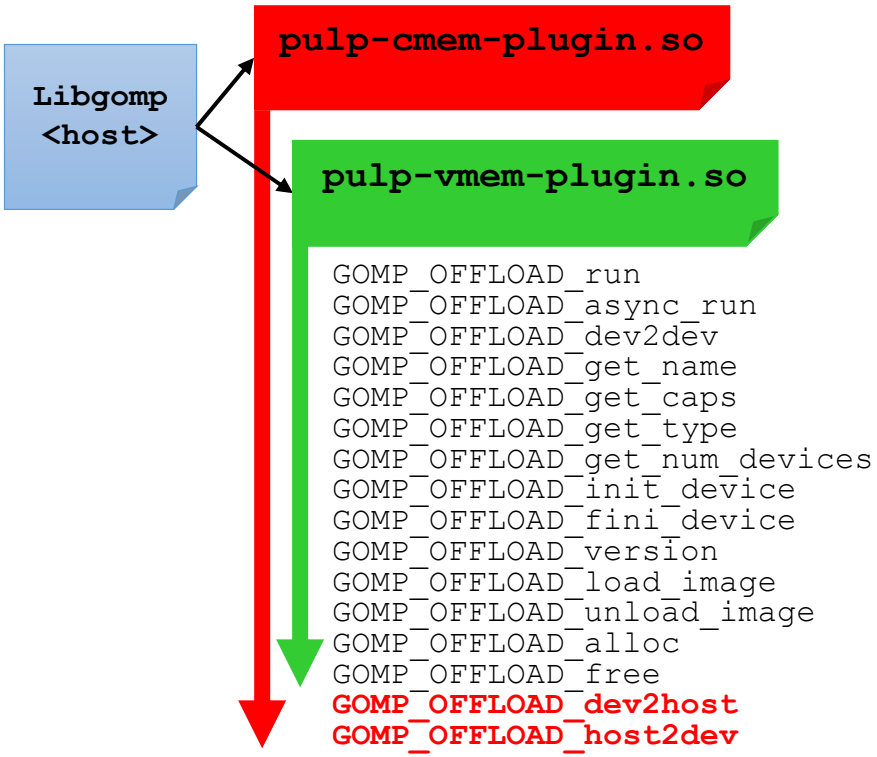
GCC Runtime library extensions

On host side:

- Modified the standard libgomp to remove forced device to/from host data transfer
- Created two libgomp plugin for the PULP accelerator

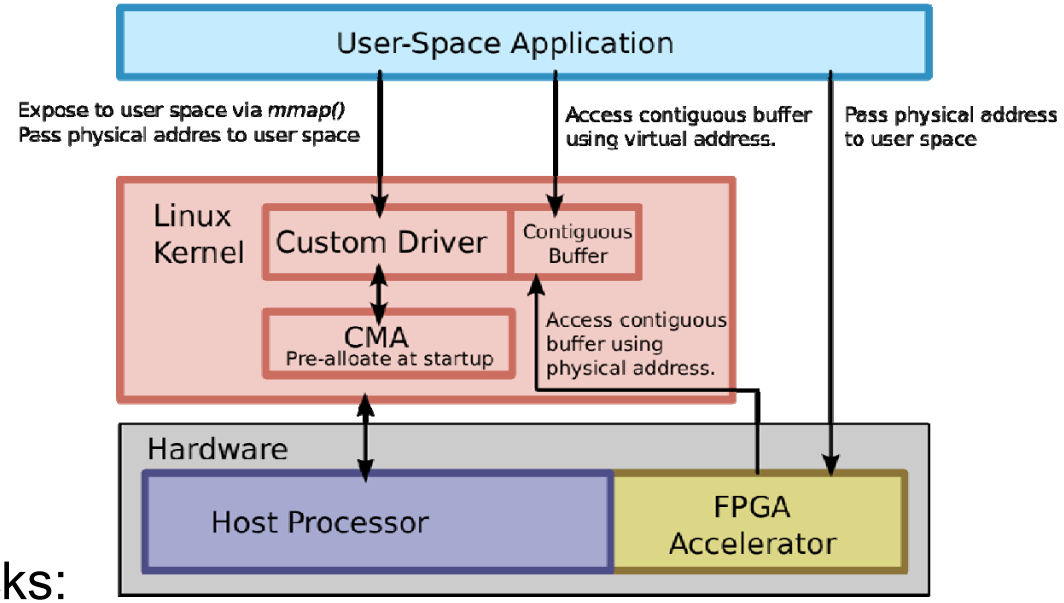
On PULP side:

- Customized – already existing – libgomp to manage offload requests



Memory Sharing in Embedded Systems (1)

- Contiguous Memory Allocator (CMA):
  - Pre-allocate a contiguous kernel-space buffer at boot time.
  - Apply a constant offset for virtual-to-physical address translation.
  - Zero-copy



- Drawbacks:
  - Requires custom kernel module to expose the contiguous memory to user-space and to get the physical address.
  - High latency, no guarantees on the availability
  - Contiguous Buffer is un-cached on ARM

ETH zürich

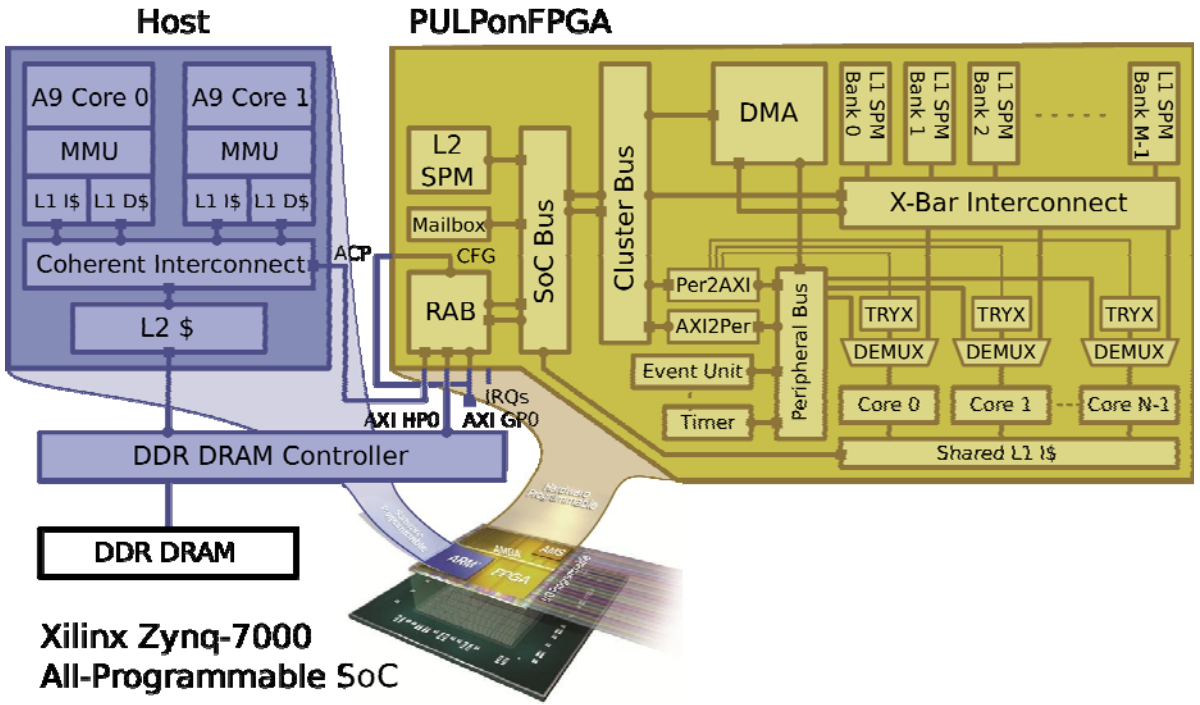
Results: FPGA Resource Utilization

- Fulmine cluster with 8 Cores, 256 KiB L1, 8 KiB I$
- RAB:
  - L1 TLB: 4 + 32 slices
  - L2 TLB: 1024 entries
- IOMMU [Kornaros et al., SoC'14]:
  - 64-entry IOTLB, 6 cycles look-up latency

Block	Slice LUTs [K]	Slice Regs [K]	Block RAM [Kb]
PULP Cluster	120	56	2163
L1 TLBs	6.6	4.7	0
L2 TLB	0.3	0.1	45
Buffer & Control	1.8	2.7	0
RAB Total	8.7	7.5	45
IOMMU [1]	11.15		407.65

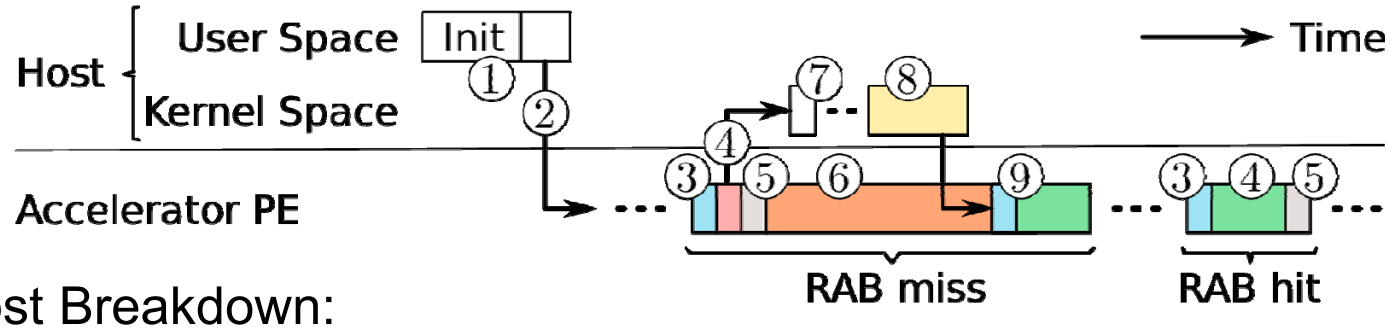
Platform Details

- ARM @ 333 MHz, PULP @ 50 MHz, DDR @ 300 MHz
- RAB could be clocked at @ 100 MHz, peak bandwidth to shared memory of 6.4 Gbps
- FIFO replacement strategy for RAB management



RAB Miss Handling: Cost Breakdown Analysis

- Average RAB miss handling time ~5500 cycles
- RAB miss handler
  - Not optimized to host architecture, fully portable
  - Page table walker not executable in interrupt context
  - Use of Concurrency Managed Workqueue API of Linux



- Cost Breakdown:
  - 20% until host starts to handle the interrupt = schedule work (7)
  - 50% until the worker thread starts to handle the miss (8)
  - 30% actual miss handling
    - 23% *get_user_pages()*