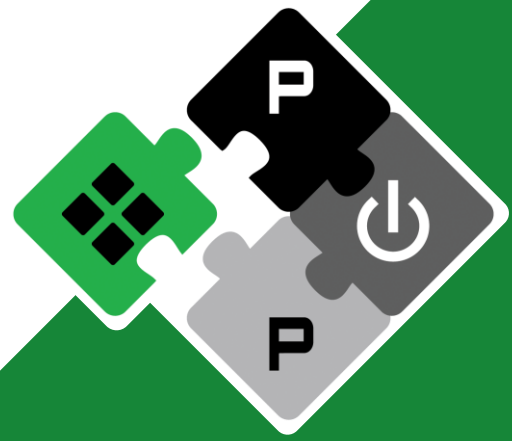
ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

CachePool: Many-core cluster of customizable, lightweight scalar-vector PEs for irregular L2 data-plane workloads

Integrated Systems Laboratory (ETH Zürich)

Zexin Fu, Diyou Shen zexifu, dishen@iis.ee.ethz.ch

Alessandro Vanelli-Coralli avanelli@iis.ee.ethz.ch
Luca Benini lbenini@iis.ee.ethz.ch



@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

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

Outline

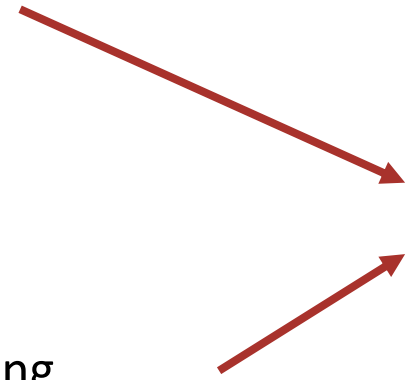
- **Recall: The target workload & Extracted kernels**
- **Initially proposed architecture candidates**
  - Private L1 + Shared L2
  - Share L1 with configurable partition
- **Open discussions**

Recall: The Target Workload

- **RLC Packet Handling**
  - Massive unstructured sparse data handling in double linked-list format **-> Data Cache**
  - Minimum calculation overhead **-> Control-oriented**
  - Need buffer for ACK retransmission handling **-> Large memory management**
- **RLC Control**
  - Mixed scalar/vector instruction (40%-60%) **-> Need vector unit**
  - Need to support different types of INT (8, 16, 32, 64) **-> 64-bit data path**
- **Performance target**
  - High throughput **-> Manycore cluster**
  - Tight TTI **-> Low latency interconnect**

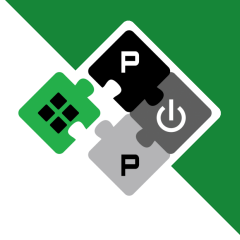
Recall: Extracted Kernels

- **RLC Packet Handling**
  - Linked List
  - Pointer Chasing
- **RLC Control**
  - Sparse Matrix-Vector Multiplication(SpMV)
  - Maximum Value Sorting
  - Logarithm Calculation Using Taylor Expansion
  - Sum Reduction



Tiles with Scalar+Vector cores,
Large size of data cache

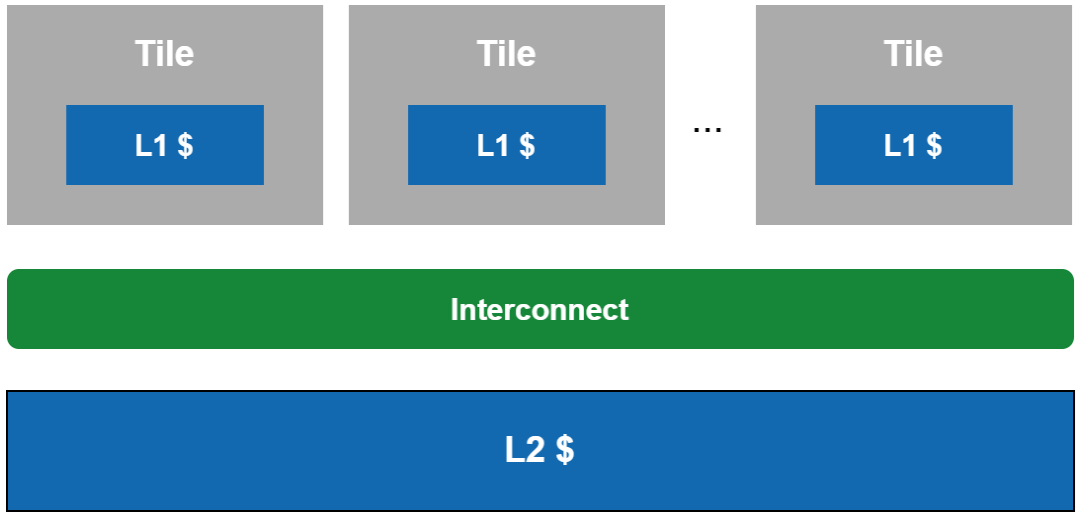
Overhead of HW Managed Coherence



- **Snoop-based protocol**
  - Coherence traffic in the system interconnect
  - Long snoop latency
- **Directory-based protocol**
  - Extra area and power for directory structure
  - The directory can be the bandwidth bottleneck
- **Ways to avoid the coherence overhead**
  - Write-through cache
  - Shared cache
  - Software data management
- **Our initial candidate designs avoid using invalidate-based coherence protocol**
  - We are still open to any low-cost scalable coherence design

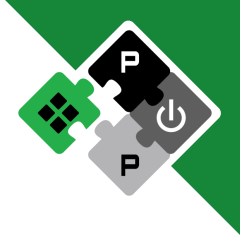
Design Candidate 1: Private Write-through L1 + Shared L2

- **Small inclusive private L1 cache**
  - To handle frequently accessed data
  - Write-though to avoid cache coherence data transfer overhead
  - Very small to reduce the data duplication
- **Large shared L2 cache**
  - PE direct write through to L2
  - Multi-sliced for large bandwidth



The Cluster-Level System Diagram

Private Write-through L1 + Shared L2



- **Pros**
  - Higher data locality within a Tile.
  - Reduce the interconnect cost between Tiles.

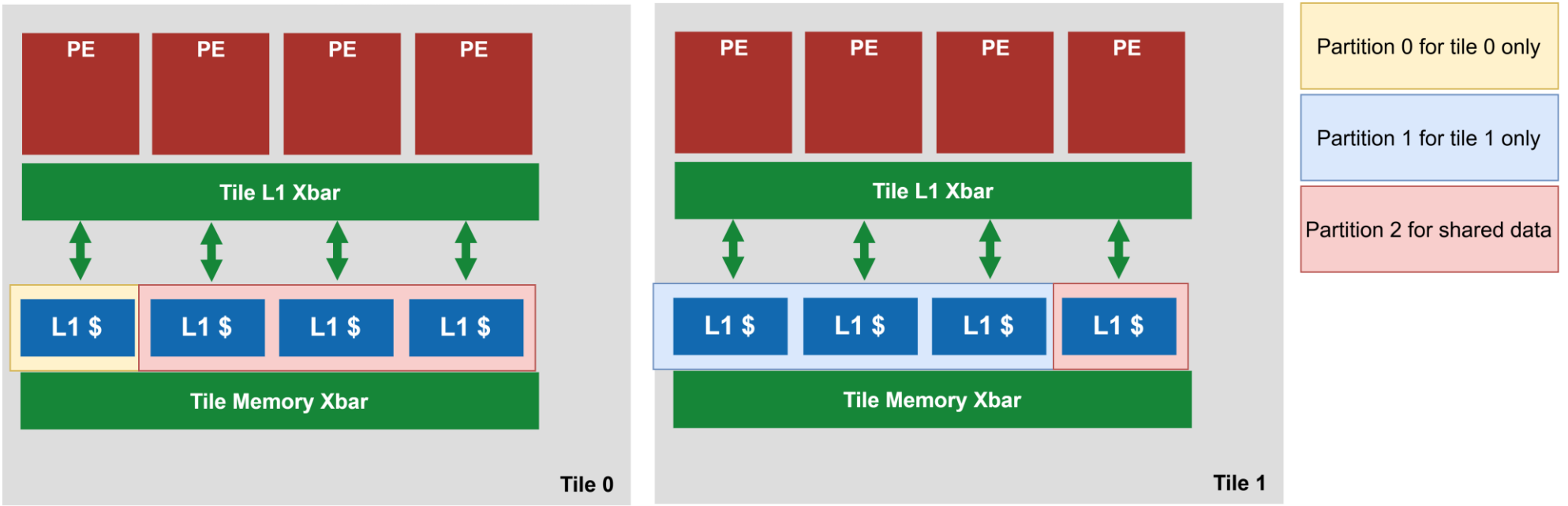
- **Cons**
  - Data duplication
    - Multiple copies in different Private L1, and between L1 and L2, influence the actual usable memory size
  - Write-through
    - The latency and bandwidth for writes. Performance and power degrade for write-intensive kernels.

Shared L1 Cache

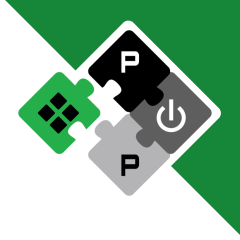
- **Traditional: Shared L1 Cache Design General View**
  - Similar to TeraPool/MemPool interconnection hierarchy.
    - Replace SPM with Cache
  - A large multi-banked L1 cache shared by all Tiles.
  - Each Tile will have its own Cache Controllers to manage the cache slices (multi-banked).
- **Pros**
  - No coherence policy is needed
  - No overhead from L2: Duplicated data, Tag, Control logic
- **Cons**
  - The physical-feasible interco. will limit the remote accessing BW
  - High ratio of long-latency remote access

Design Candidate 2: Shared L1 Cache with Partitioning

- Add partition on top of the shared multi-banked L1.
  - Configurable partition schemes for Tile-private or global-shared use.
  - The granularity of partition would be the cache bank.
  - Keep a globally shared partition to handle shared data and coherence.
- This can make keep private data closer to the consumer PEs to increase locality.



Shared L1 Cache with Partitioning



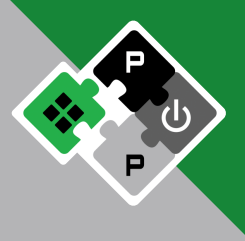
- **Pros**
  - Same as previous, no coherence and no L2 overhead.
  - Higher data locality, less remote access.
  - Flexible cache resource allocation, maximizes the effective cache capacity utilization.
- **Cons**
  - Configurable partitioning logic overhead in timing, area and design effort

Open Discussions

- **Design choices**
  - Private L1 + Shared L2
  - Share L1 with configurable partition
  - The design is not settled, open to any new design choices.

- **Kernel information**
  - Kernel parameters sheet and document mentioned in the last meeting

- **Next meeting schedule**



Thank you!