ETH zürich

# Designing and Scaling Versatile Manycore Systems

Samuel Riedel advised by Luca Benini at Integrated Systems Laboratory (IIS), ETH Zürich, Switzerland

## Massively parallel and versatile?

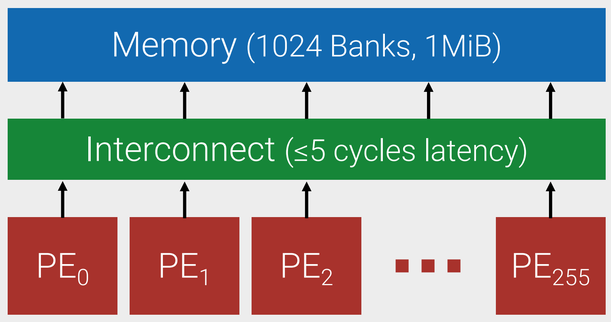
### Modern Workloads

- Machine Learning
- Computational Photography
- Communication
- Graph processing

*Can we run all modern applications efficiently on one architecture?*

### Requirements

- Massively parallel → manycore
- Fast evolving → programmable
- Power budget → energy-efficient




### Shared-memory cluster

+ Individually programmable cores
+ Flexible memory access patterns
+ Widely used (GPU, accelerator)
− Limited scalability?

### Challenges in scaling:

- **Connect** hundreds of cores to thousands of memory banks with a **low latency**
- Simple but **latency-tolerant** cores
- Physical implementation
- **Synchronizing** hundreds of cores
- High performance and efficiency for **irregular and regular workloads**
- **Emulating** massively parallel systems

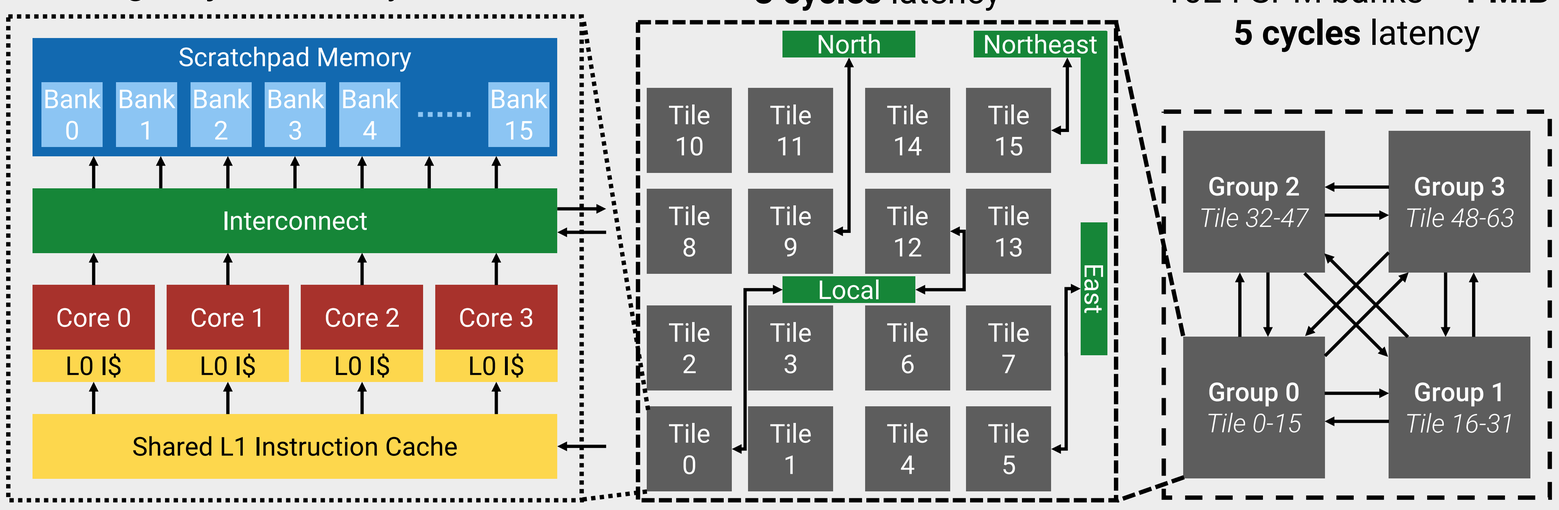
## MemPool: Scaling the shared-memory cluster

A **hierarchical** design allows scaling to up to **256 cores**, sharing access to **1024 memory banks** with less than **five cycles of latency**, which can be **hidden** by latency-tolerant RISC-V cores. A **hierarchical DMA** and **L2 interconnect** move data in and out.

**MemPool Tile**
4 32-bit cores, 16 SPM banks
**Single-cycle** memory access

**MemPool Group**
64 cores, 256 SPM banks
**3 cycles** latency

**MemPool Cluster**
256 cores
1024 SPM banks = **1 MiB**
**5 cycles** latency

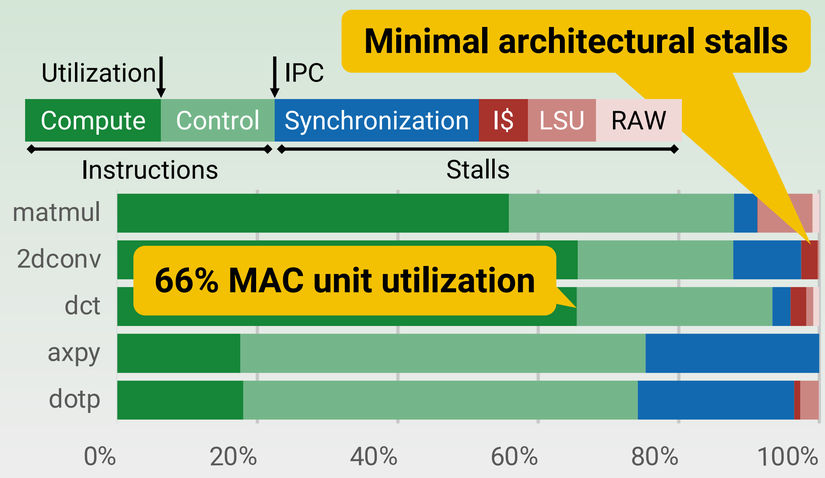



### Programming and performance?

Easy to program
- C, Rust, OpenMP, Halide
- GCC, LLVM with instruction scheduling

Versatile and flexible
- DSP kernels, ray tracing, 5G communication, transformer models

High performance
- Hide latencies and achieve close to ideal scaling



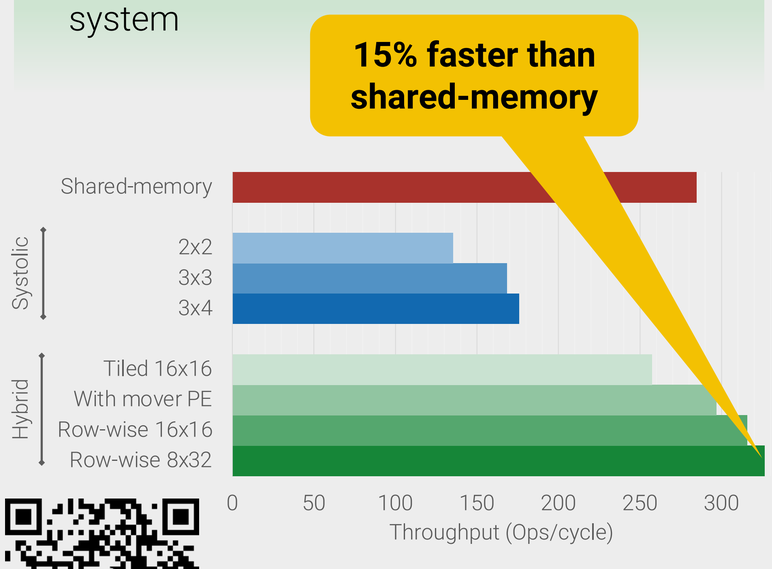

## Hybrid-Systolic-Shared-Memory

- Enable efficient systolic execution on MemPool through lightweight extensions.
- Allow fast core-to-core communication between any cores.

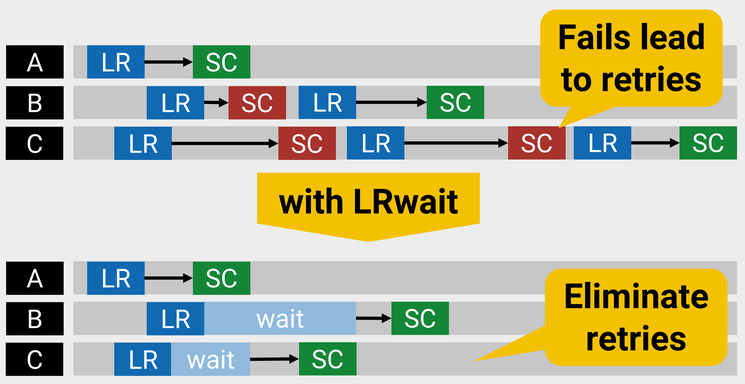
→ Allow arbitrary systolic topologies.

→ Combine shared-memory and systolic execution to achieve optimal performance.

→ Efficiently execute systolic workloads while keeping the flexibility of the shared-memory system



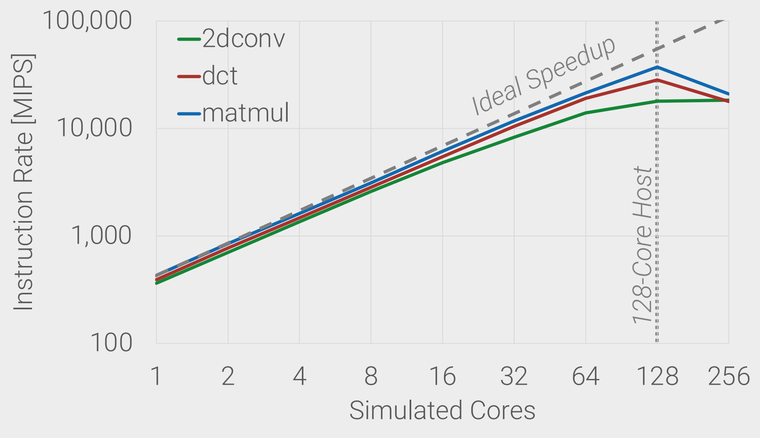

## Polling-free Synchronization




- Order atomic operations at the LRwait instruction instead of SCwait
  - Build a queue of reservations
  - Release the next reservation after each SC
  - **Colibri**: Build a scalable, hardware-efficient distributed queue
- Eliminate retries and polling.
- Better fairness and performance.
- Outperforms LR/SC by a factor of 6.5x
- 8.8x more energy efficient.

## Emulating manycore systems

- Banshee: A functional simulator designed to simulate hundreds of cores
- Designed for cluster-based architectures
- Easily extensible
- Static binary translation
- Instruction accurate
- Emulates up to 72 GIPS
- 1.5 times faster in single-core scenario
- Up to 44x faster for manycore simulations




◁ open-source on GitHub!

sriedel@iis.ee.ethz.ch

pulp-platform.org

@pulp_platform