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
Open Source Hardware, the way it should be!

@pulp_platform
pulp-platform.org
youtube.com/pulp_platform

Literature Study Report

Report by:
Diyou Shen, Ph.D. Student
Zexin Fu, Ph.D. Student
dishen, zexifu @iis.ee.ethz.ch

Supervised by:
Prof. Alessandro Vanelli-Coralli
Prof. Luca Benini
avanelli, lbenini @iis.ee.ethz.ch

Integrated Systems Laboratory,
ETH Zürich,
Zürich, Switzerland

Content

Abstract

- This project focuses on hardware architecture design targeting at RLC layer in 5G communication, with the following specifications:
- **Manycore Architecture Design**: a manycore architecture handling the RLC packet transmission, receiving and retransmission required in the Acknowledge mode (AM) in RLC layer.
- **Scalable Cache Design**: a scalable cache architecture for the sparse packet processing in the RLC layer.
- **Vector PE Design**: the vector processor PE microarchitecture for RLC layer control handling, supporting scalar-vector hybrid processing.

Literature Study

- **Cache Coherence**
  - A Novel Hybrid Cache Coherence with Global Snooping for Many-core Architectures
    https://dl.acm.org/doi/10.1145/3462775
  - SelectDirectory: A Selective Directory for Cache Coherence in Many-Core Architectures
    https://ieeexplore.ieee.org/document/7092378
  - Heterogeneous System Coherence for Integrated CPU-GPU Systems
    https://dl.acm.org/doi/10.1145/2540708.2540747

A Novel Hybrid Cache Coherence with Global Snooping for Many-core Architectures

- **Paper Summary**
  - This paper introduce WiSH (Wireless-enabled Share-aware Hybrid) coherence protocol to address scaling challenges in many-core processors.
  - The protocol is built on a Snoopy-over-Directory **hybrid coherence** including **local directory based** within clusters and **global snoopy bus** across clusters by wireless interconnection network.
- **Advantages**
  - Area efficiency: reduce the memory requirements by up to 98% compared to traditional directory-based approach.
  - Better scalability and less traffic.

A Novel Hybrid Cache Coherence with Global Snooping for Many-core Architectures

- **Inspirations and Concerns**
  - The **hybrid approach** offers a promising solution for reducing overheads in scaling many-core architectures. Similar strategies, such as **Directory-over-Snoopy**, also warrant consideration as potential solutions for our **cache coherence architecture**.
  - The reliance on **wireless communication between clusters**, while innovative, introduces practical challenges. These include the complexity of **digital-analog co-design**, as well as issues related to **signal integrity**, **power consumption**, and **noise mitigation**. This makes it **inapplicable to our architecture**, which requires simpler and more predictable interconnect mechanisms.
  - Without the wireless communication, the high bandwidth requirement of Snoopy bus between clusters might become the bottleneck in latency and scaling.

SelectDirectory: A Selective Directory for Cache Coherence in Many-Core Architectures

- **Paper Summary**
  - This paper introduces SelectDirectory, a compact directory-based cache coherence protocol for many-core architecture.
  - It contains the following key design points
    - Selective Directory: separate tag array and coherence metadata (data) array. Tag Array tracks all memory block, but Data Array only allocates entries for actively shared blocks
    - Coherence Management: Fill in Data Array entry when a block transit from private to shared and deallocate it when from shared to private.
- **Advantages**
  - Save the SRAM space required for directory up to 2x.
  - Better scalability for many-core architecture

SelectDirectory: A Selective Directory for Cache Coherence in Many-Core Architectures

- **Inspirations and Concerns**
  - The approach can be treated as **a "pro" version of MESI** directory based coherence, which also reduce the number of metadata needed to be stored. It can be a good addition on top of other approaches, e.g. Snoopy-over-Directory and Directory-over-Snoopy.
  - This approach alone still face problems when scaling up to hundreds and thousands of cores. For the highly parallel workload, such a strong coherence may not be needed globally. It is possible to apply it across 8-16 cores, and uses software synchronization at a higher hierarchy.

Heterogeneous System Coherence for Integrated CPU-GPU Systems

- **Paper Summary**
  - This paper introduces Heterogeneous System Coherence (HSC), a hardware solution designed to address the performance bottlenecks caused by high memory bandwidth demands and coherence traffic in heterogeneous systems.
  - It contains the following key design points
    - Region-Based Coherence: HSC replaces the conventional block-level directory with a region directory and introduces region buffers. These structures track coherence permissions at a coarser granularity, reducing the bandwidth and overhead of coherence traffic.
    - Directory bypassing: By granting permissions for entire memory regions rather than blocks, most L2 cache misses can directly access memory without accessing the directory.
- **Advantages**
  - Make the directory more scalable in terms of both access bandwidth and capacity.
  - Allow the system to move bandwidth from the coherence network to the high-bandwidth direct-access bus without sacrificing coherence.

Heterogeneous System Coherence for Integrated CPU-GPU Systems

- **Inspirations and Concerns**
  - The **Region-Based Coherence** approach introduced in the paper capitalizes on the spatial and temporal locality of memory accesses by granting coherence permissions for **larger memory regions** rather than **individual cache blocks**. This technique is particularly beneficial in systems where specific memory regions are actively used by a small set of cores (e.g., a tile) for a period and then handed off to another set of cores.
  - To further improve efficiency, we can enhance the design by integrating hints or configurations from software side to **dynamically adjust the size** of region directory entries. This adaptability would enable finer-grained coherence management, tailored to specific workload characteristics, further optimizing coherence traffic, memory access bandwidth and directory utilization.

Literature Study

- **Cache Design**
  - Stop Crying Over Your Cache Miss Rate: Handling Efficiently Thousands of Outstanding Misses in FPGAs

    https://dl.acm.org/doi/10.1145/3289602.3293901

  - Analyzing and Leveraging Shared L1 Caches in GPUs

    https://dl.acm.org/doi/pdf/10.1145/3410463.3414623

Stop Crying Over Your Cache Miss Rate: Handling Efficiently Thousands of Outstanding Misses in FPGAs

- **Paper Summary**
  - The paper proposes a miss-optimized memory architecture for FPGAs to address low cache hit rates in irregular, memory-bound applications (e.g., sparse linear algebra, graph analytics).
  - The design focuses on a scalable, MSHR-rich non-blocking cache architecture tailored for FPGAs to efficiently handle thousands of outstanding misses in memory-bound applications. Central to the design is the use of cuckoo hashing for MSHR management, which enables scalability by storing MSHRs in multiple hash tables indexed by independent hash functions, reducing collision probabilities. Displaced entries are temporarily stored in a small content-associative stash, allowing collisions to be resolved in the background without stalling the pipeline. Each MSHR tracks primary misses, while dynamically allocated subentry buffers manage secondary misses, with subentries organized as linked lists for flexibility and efficient use of memory resources.
  - This architecture balances memory-level parallelism and resource utilization, providing Pareto-optimal solutions across benchmarks like sparse matrix-vector multiplication.

Stop Crying Over Your Cache Miss Rate: Handling Efficiently Thousands of Outstanding Misses in FPGAs

- **Inspirations**
  - The innovative use of cuckoo hashing and dynamic subentry allocation offers a practical solution for scaling MSHRs, addressing a critical bottleneck in memory-bound FPGA applications. This approach inspires similar scalable designs for our cache-based systems with potential many miss handling requirement.
  - The reallocation of MSHR storage into FPGA block RAM shows how resource utilization can be designed to significantly enhance scalability. By leveraging block RAM (or SRAM in ASIC), the design overcomes the limitations of flip-flop-based associative arrays, which are less scalable and consume more area. This strategic shift enables the handling of thousands of outstanding misses, addressing the scalability bottleneck and unlocking the potential for FPGAs to handle irregular, memory-bound applications effectively.

Analyzing and Leveraging Shared L1 Caches in GPUs

- **Paper Summary**
  - This paper addresses the inefficiencies in the conventional private L1 cache organization in GPUs, where data replication across cores reduces effective cache capacity and bandwidth utilization.
  - It proposes a shared L1 cache design, where cores collectively cache a single copy of data by dividing the address space into non-overlapping slices. This eliminates data replication but introduces latency from inter-core communication when a core accesses data not mapped to its address range.
  - To mitigate inter-core communication latency, the authors develop lightweight communication optimizations and a dynamic scheme to switch between shared and private L1 cache organizations based on application behavior.
  - Evaluations across 28 GPGPU applications show that the dynamic scheme improves performance by 22% (up to 52%) and energy efficiency by 49% for shared-friendly applications, with a modest hardware overhead of 0.09 mm^2 per core.

Analyzing and Leveraging Shared L1 Caches in GPUs

- **Inspirations**
  - The idea of a shared L1 cache that eliminates data replication across cores demonstrates the potential for significantly improving effective cache utilization and bandwidth in multi-core or GPU systems. In our case, this can also help to eliminate the overhead of hardware coherence support.
  - The dynamic scheme to switch between shared and private L1 cache organizations based on runtime application behavior also helps to meet diverse workload needs.
  - Selective allocation of cache resources by mapping address ranges to specific cores is helpful for efficient resource management, especially when the memory access pattern is relative regular.
  - Optimizations like selective data fetching (chunking) and traffic balancing offer practical strategies for minimizing the latency and energy costs of inter-core communication in shared memory systems.

Literature Study

- **Vector Unit**
  - A 45nm 1.3GHz 16.7 Double-Precision GFLOPS/W RISC-V Processor with Vector Accelerators

    https://ieeexplore.ieee.org/document/6942056

  - Performance Modeling of Streaming Kernels and Sparse Matrix-Vector Multiplication on A64FX

    https://ieeexplore.ieee.org/abstract/document/9307836

A 45nm 1.3GHz 16.7 Double-Precision GFLOPS/W RISC-V Processor with Vector Accelerators

- **Paper Summary**
  - This paper describes a dual-core RISC-V Vector processor Hwacha with coherence cache design.
  - L1 Memory
    - On-chip private 32KiB non-blocking L1 Data Cache for each core
    - Broadcast-based MESI cache coherence protocol with L2
    - Virtualization support
  - L2 Memory
    - On-chip global 1MiB L2 Cache
  - Vector Core Design
    - 8 1R1W SRAM-based VRF

A 45nm 1.3GHz 16.7 Double-Precision GFLOPS/W RISC-V Processor with Vector Accelerators

- **Inspirations**
  - This paper proposes a dual-core RISC-V PE with Vector accelerators with the Linux capability, which serves as a baseline for the RISC-V Vector (RVV) PEs developments later. The design has a cache-based memory with virtualization support.
  - Our group has a similar dual-core RVV design Spatz (https://ieeexplore.ieee.org/document/10069431) using a SPM-based memory system. Besides the differences in memory system, the scalar core in Spatz cluster is designed to be compact for scalability, and does not have the capability of Linux booting. Spatz vector PE supports the different configurations on the number of FPU/IPUs inside for different use cases.
  - For the CachePool project, the vector PE is needed in RLC control algorithm, which does not require the system and virtualization support. However, we may need to adapt the memory system design (cache or hybrid cache and SPM) inside Spatz to support the sparse data pattern if needed.

Performance Modeling of Streaming Kernels and Sparse Matrix-Vector Multiplication on A64FX

- **Paper Summary**
  - A64FX CPU contains 48 cores with vector capability (SVE) with core-private 64 KiB L1 cache, and a sharing 8 MiB L2 among each core memory group (CMG) arranged in a cache-coherent non-uniformed memory access (ccNUMA) format.
  - This paper uses the Execution-Cache-Memory (ECM) framework to analyze A64FX's performance bottlenecks and analyzed results on real Fujitsu FX700 system.
  - The paper leverages the SELL-C-σ format to achieve the bandwidth saturation on sparse matrix-vector multiplication (SpMV) kernel
- **Advantages**
  - The paper identified the challenges with CRS format SpMV on the high memory traffic, and proposes a more efficient SELL-C-σ format for SIMD processors

Performance Modeling of Streaming Kernels and Sparse Matrix-Vector Multiplication on A64FX

- **Inspirations and Discussions**
  - This paper provides a valuable solution to efficiently calculate SpMV on a many-core vector system with cache-based memory. This may have potential interests for us on the linear algorithm for RLC control logic. However, several questions are still unclear at this stage:

    What is the input pattern for RLC control algorithm? Are they sparse data headers from different users' packets (unstructed sparsity pattern)?

    - This would determine whether we need cache, SPM or hybrid memory system for linear algorithm accelerator (scalar + vector PE).

    What is the expected computational throughput requirement? Do we need a manycore system for this calculation?

    - This would determine the number of IPUs and scale of the accelerator cluster.

Conclusion

- This literature study provided a thorough analysis of state-of-the-art designs in **cache coherence**, **cache design**, and **vector PE architectures** relevant to scalable many-core systems.
- **Key Insights:**
  - **Cache Coherence:** Hybrid coherence protocols, including **Snoopy-over-Directory**, **SelectDirectory**, and **HSC**, present scalable and efficient methods for managing data consistency across cores, though each comes with unique limitations in large-scale setups.
  - **Cache Design:** Techniques like **cuckoo hashing for MSHRs** and **shared L1 cache designs** effectively balance resource usage and minimize latency. These could be some useful ways for **scaling** our system.
  - **Vector PE Design:** Architectures such as **Hwacha** and **SVE for A64FX** offer adaptable scalar-vector processing configurations, which can help address the specific demands of **RLC control algorithms**.

Conclusion (cont'd)

- **Design Directions:**
  - **Memory Management for RLC Data:** Explore memory management patterns and design a cache system to efficiently handle sparse and irregular data access patterns while meeting memory bandwidth and latency targets. A critical focus will be balancing system performance and scalability, such as determining whether to adopt a **shared L1 cache** or **private L1 cache with hardware coherence management**.
  - **Compute of Linear Algebra for RLC Control:** Design efficient heterogeneous compute elements to handle linear algebra workloads, focusing on both scalar and vector processors, ensuring optimal performance for RLC control and QoS support.

Conclusion (cont'd)

- **Plan for Next Year:**
  - **Kernel Extraction:** Identify key kernels from both data management and linear algebra tasks that capture critical workload patterns. These kernels will guide architectural design and performance evaluations.
  - **Subsystem Initial Design:** Separately design and evaluate architectures for both **data management** and **linear algebra** subsystem, focusing on efficiency, scalability, and alignment with design targets.
  - **Evaluation and Refinement:** Conduct performance and scalability evaluations for both subsystems using extracted kernels. Identify key bottlenecks and refine each subsystem accordingly.

Thank you!

Q&A

ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA