



HAL
open science

Breaking the memory wall with a flexible open-source L1 Data-cache

Davy Million, Noelia Oliete-Escuín, César Fuguet

► **To cite this version:**

Davy Million, Noelia Oliete-Escuín, César Fuguet. Breaking the memory wall with a flexible open-source L1 Data-cache. DATE 2024 - 2024 Design, Automation and Test in Europe Conference, Mar 2024, Valence, Spain. , 2024. <cea-04600891>

HAL Id: cea-04600891

<https://cea.hal.science/cea-04600891v1>

Submitted on 4 Jun 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

Memory Caches

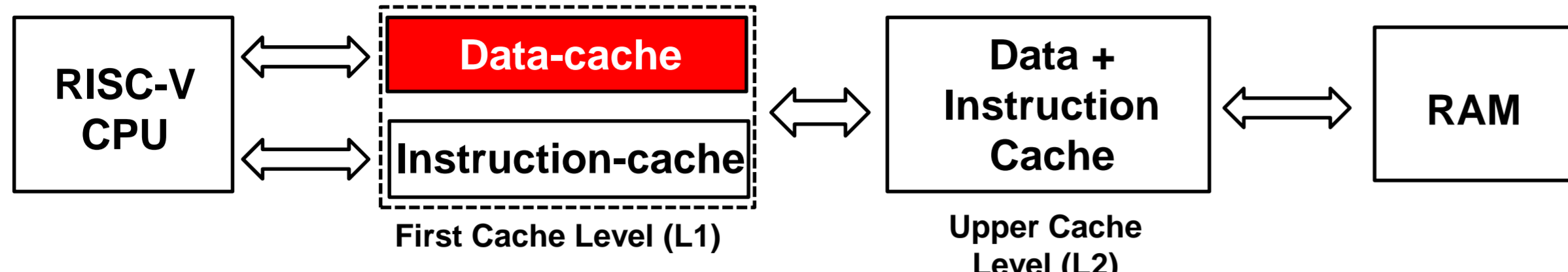


Fig 1. Memory caches in a standard computer architecture, with the data-cache near the CPU

Memory caches goal: reduce the average latency when accessing memory

Key performance metrics of the DRAM technology are evolving at different speeds:

- Recently, 3D-DRAM such as HBM allowed to continue increasing the capacity and bandwidth ;
- But latency lags behind.

→ Need for latency hiding mechanism in the computer memory hierarchy

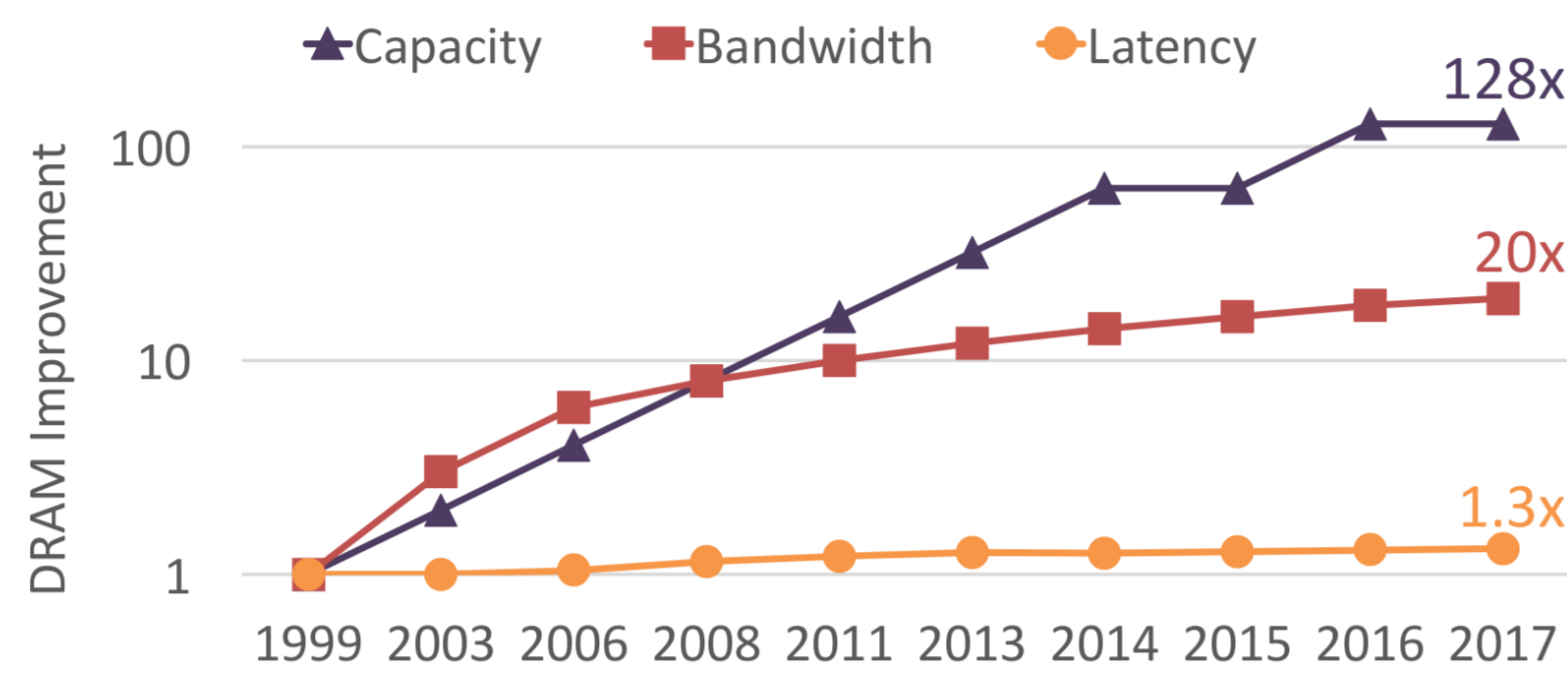


Fig 2. Memory technology evolution, reproduced from [1]

Available Open-Source Data-caches

Open-Source L1 Data-cache	Non-blocking for Miss-under-Miss ([2])	MSHR Size Scalability	Hardware Description Language	Configurability
STDcache (CVA6)	X	N/A	SystemVerilog	Low
WTDcache (CVA6)	X	N/A	SystemVerilog	Low
Rocket/BOOM	✓	Low (Flip-flops)	Chisel	Low

Table 1. Comparison of Open-Source L1 data-cache

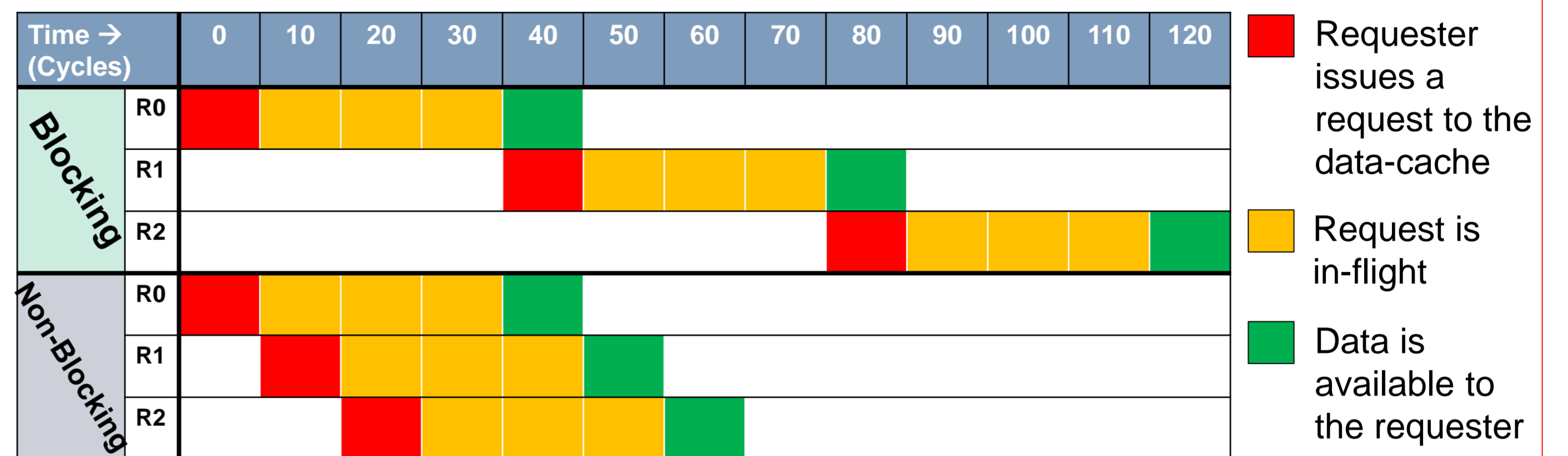


Fig 3. Total latency comparison of successive, independent requests between Miss-under-miss blocking and non-blocking data-cache

HPDcache Overview

L1 Open-Source High Performance Data-cache, RISC-V [3] [4]:

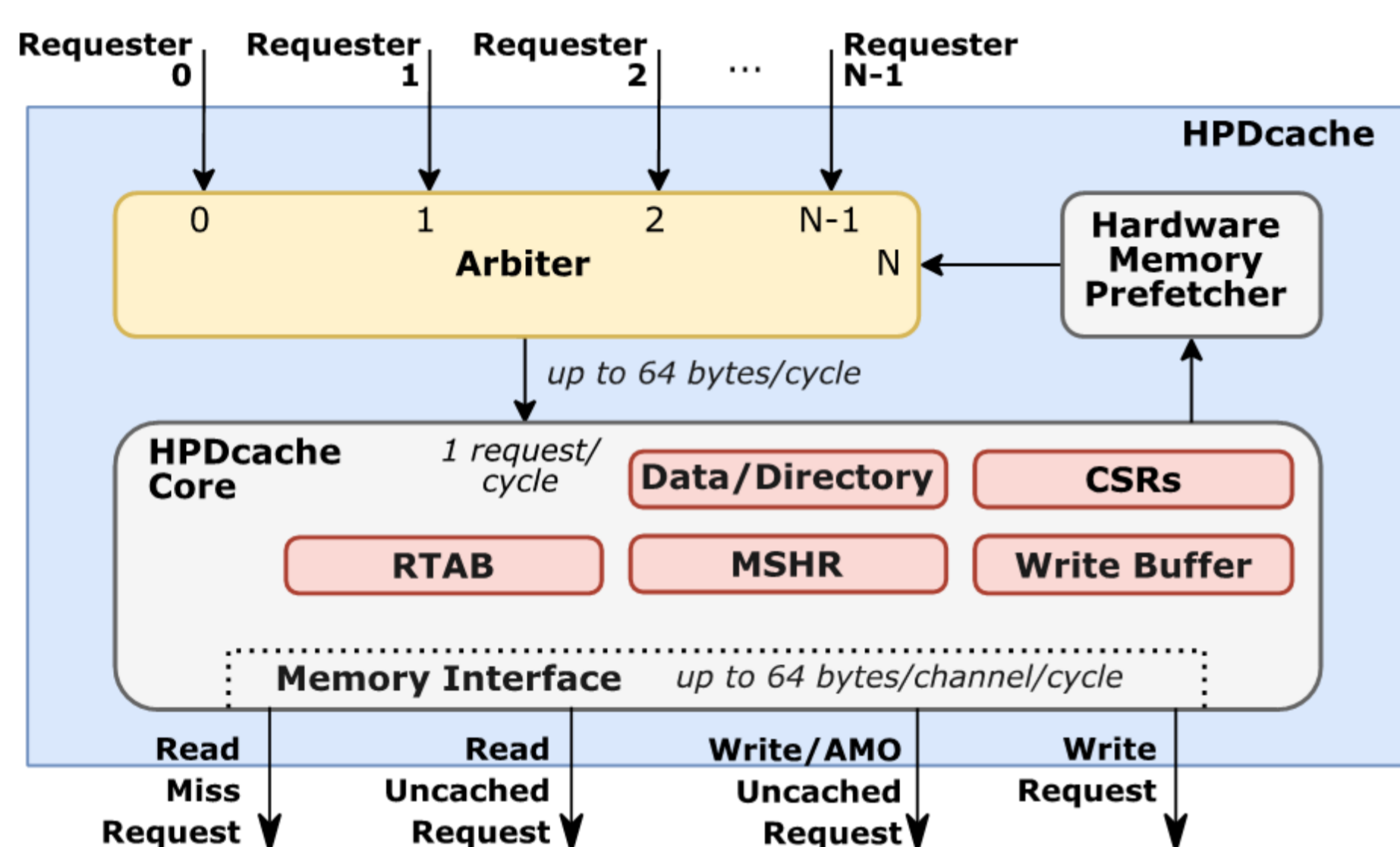


Fig 4. HPDcache overview

- Non-blocking (MSHR multi-entry, up to 128 outstanding misses) ;
- Memory interface: up to 64 bytes ;
- Out-of-order execution (RTAB) ;
- Support for RISC-V Cache Management Operation (CMOs) ;
- Highly configurable ;
- Designed to be adaptable to any requester (RISC-V core, accelerators...).



Recent Improvements

Virtual indexing (VIPT) support into the cache

→ Requesters can use VIPT to shorten the latency (1 cycle) for non-missed load

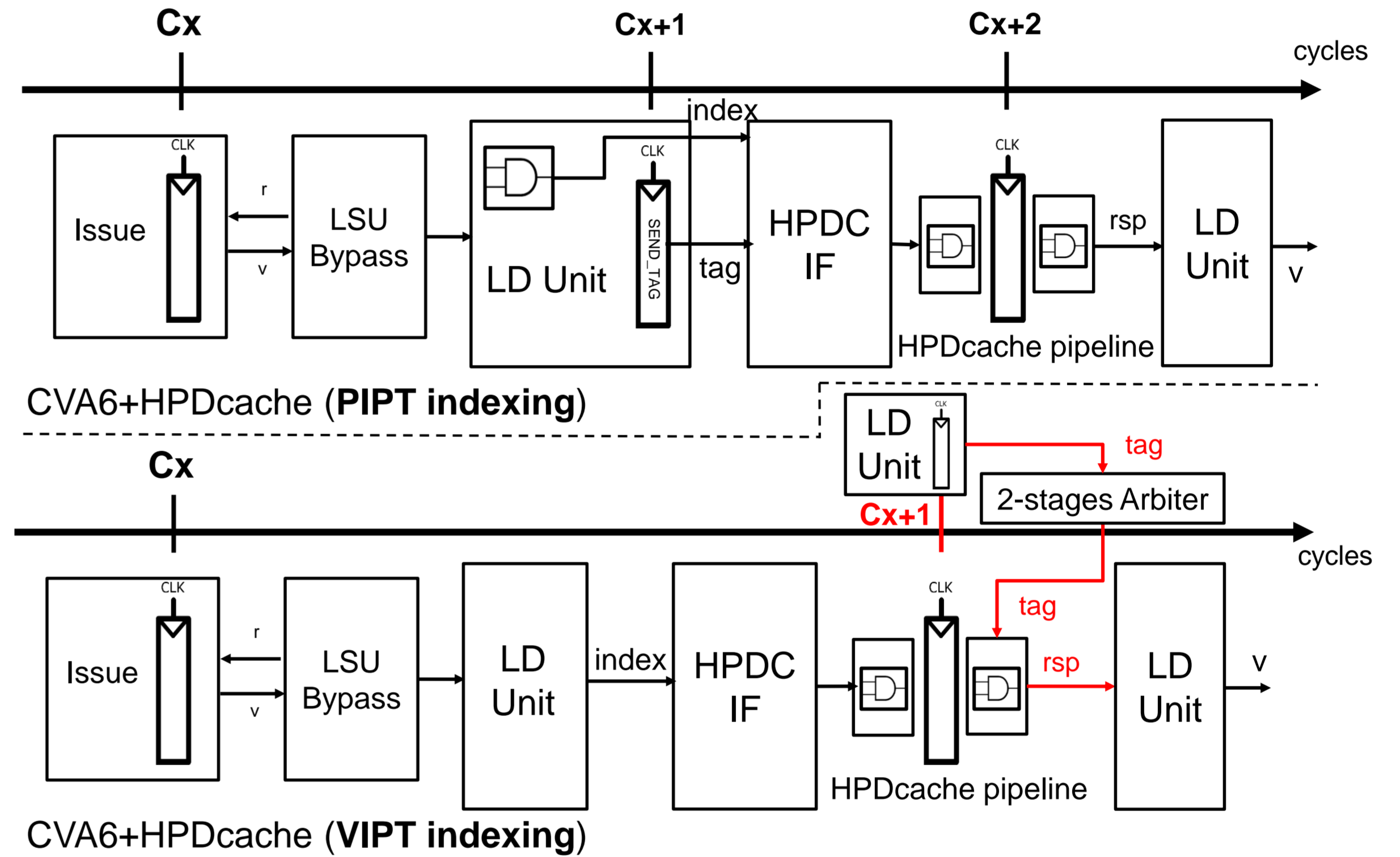


Fig 5. Latency comparison between PIPT and VIPT indexing in CVA6+HPDcache

Improvements in the selection policy of the re-ordering mechanism, to prioritize dependent on-hold requests ;

Improvements in the coalescing policy of the Write Buffer to virtually increase its size.

Performance Evaluation

Comparison between HPDcache, CVA6's WTDcache and CVA6's STDcache

Evaluation Platform

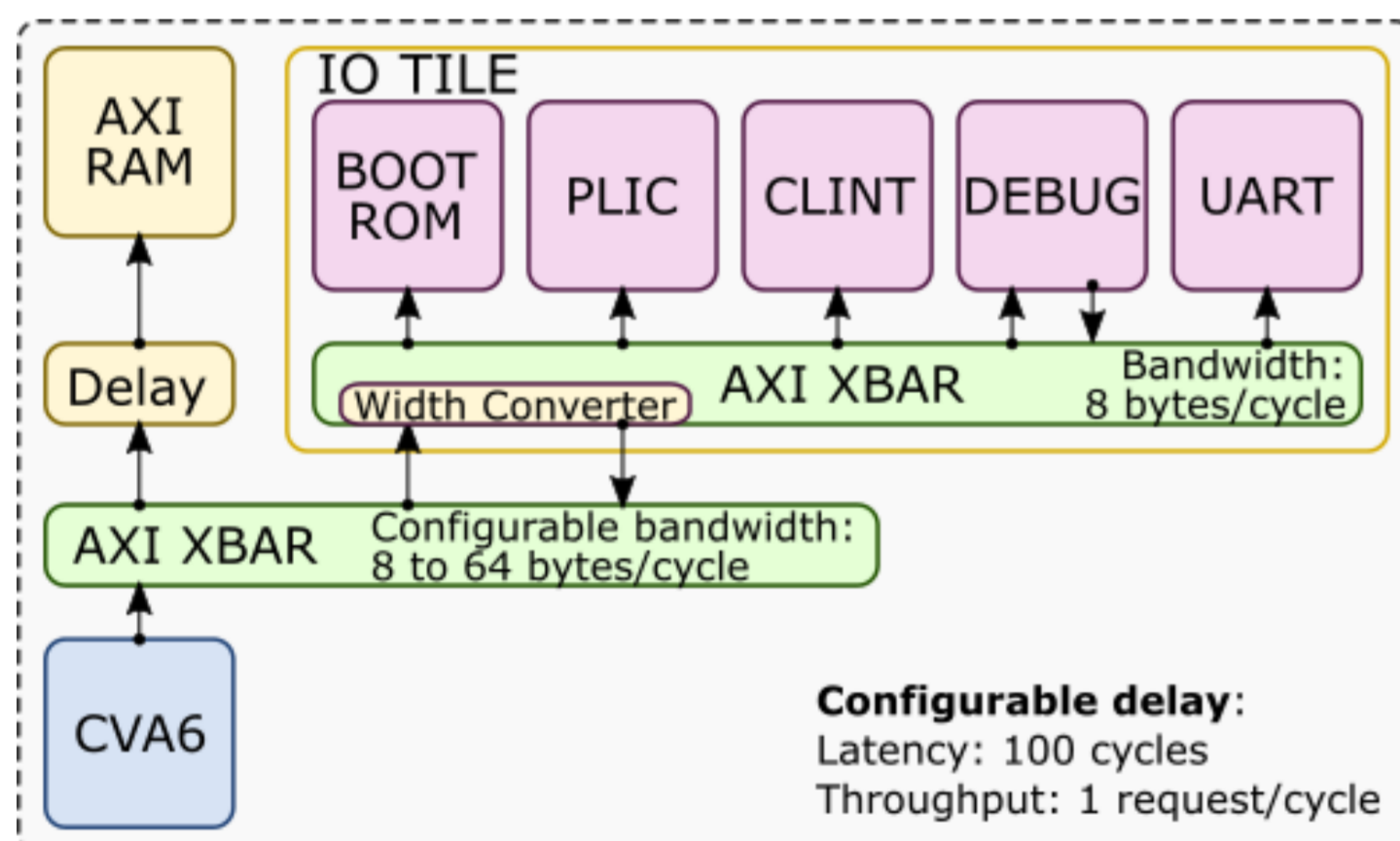


Fig 6. RTL Platform used

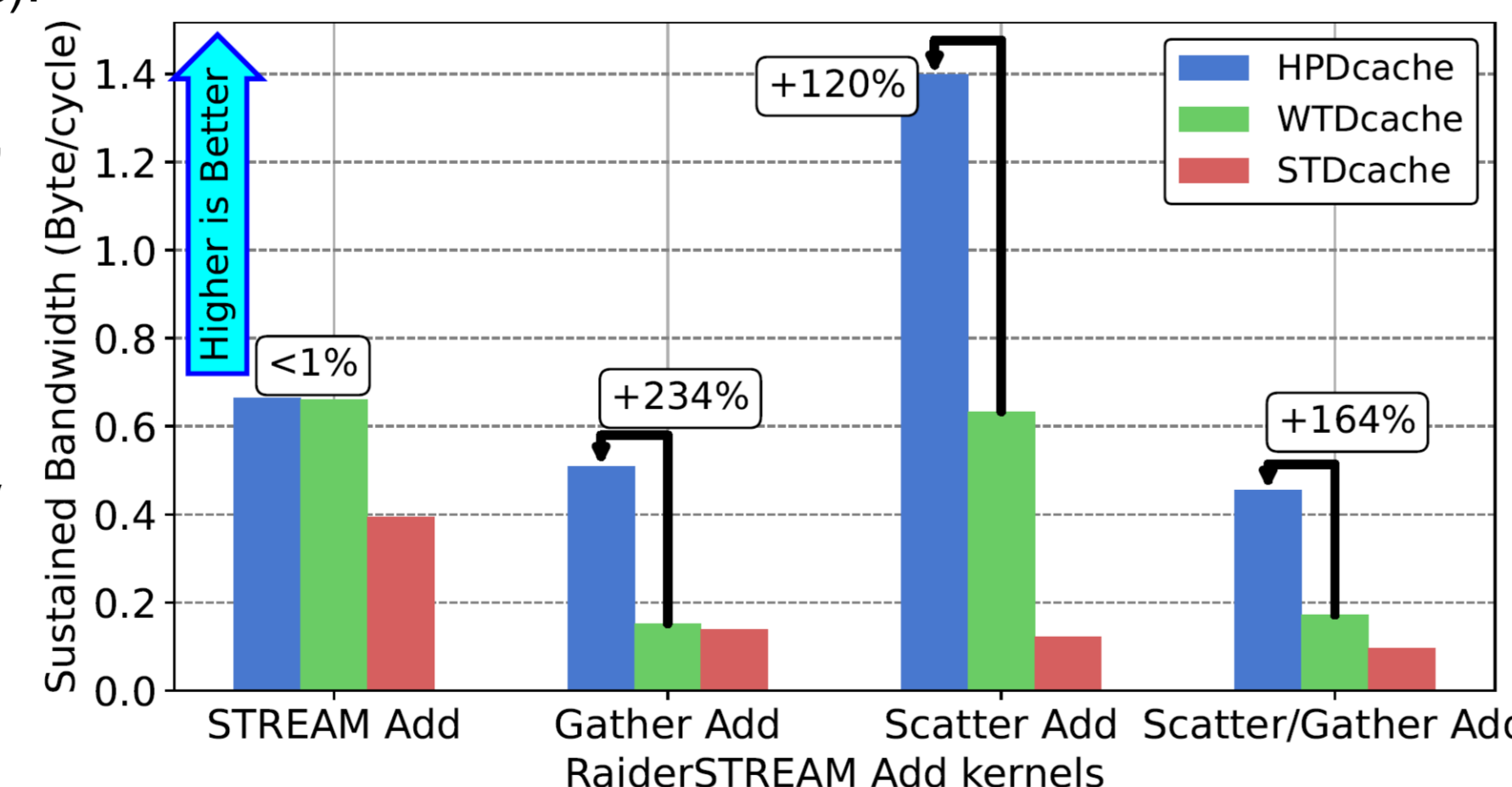
- CORE-V CVA6 with a scoreboard of 8 entries.
- L1 Data-cache (HPDcache or WTDcache or STDcache):
 - Common: 32KB size, 64B cache line, 8 ways, 8-bytes memory interface ;
 - Write-Through caches only: same write buffer size (16 * 64-bit entry) and same maximum number of in-flight store requests (16) ;
 - HPDcache only: MSHR of 8 entries.
- Fixed latency of 100 cycles between the core and memory.

Benchmarks chosen

→ SpMV and RaiderSTREAM are memory intensive programs with irregular memory patterns (independent successive loads and stores).

(1) RaiderSTREAM [5]

- STREAM benchmark with new "HPC" kernels ;
- 400KB arrays ;
- Speedup up to 234% on the kernel variants with indirections ;
- Same performance on the legacy STREAM Add variant ;
- Similar results for Copy, Scale, Triadd.



(2) SpMV (Sparse Matrix-Vector multiply)

- Three sparse matrices of size 1000x1000, 2000x2000 and 4000x4000 with 1% density and a uniform distribution of the NNZ ;
- Two "real world" matrices (DWT 2680 and GEMAT12) ;
- HPDcache speedups ranging from 9% to 43% compared to WTDcache.

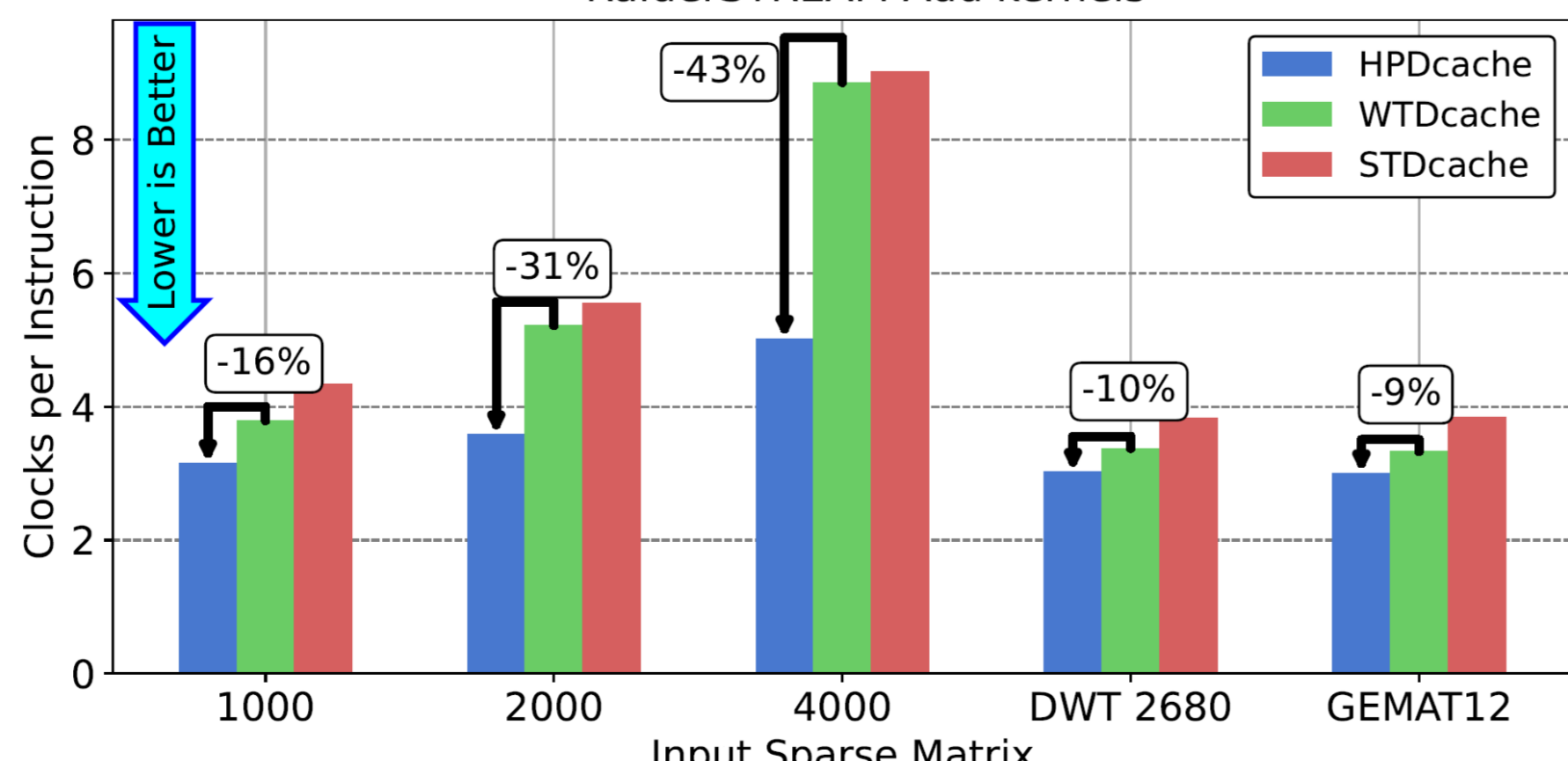


Fig 7. Performance comparison between HPDcache, WTDcache and STDcache on RaiderSTREAM and SpMV

Area and Timing Evaluation

Synthesis in the GF22FDX technology (SSG 0.72V/125C corner)

- Same platform/configuration used as for benchmarking

System considered	Area (mm ²)	Frequency (MHz)
CVA6-HPDcache	0.386	950
CVA6-WTDcache	0.364	931

Table 2. Area and timing comparison between CVA6-HPDcache and CVA6-WTDcache

→ CVA6+HPDcache is 5.92% larger, 2.06% faster than CVA6+WTDcache

- Increase HPDcache memory interface to 64-bytes implies no area impact (<1%)
- But improves its performance up to 10% (RaiderSTREAM) and 6% (SpMV)

Conclusion & Future Work

HPDcache accelerates memory intensive applications with irregular memory accesses

→ Up to 234% (RaiderSTREAM) and 43% (SpMV), with a negligible area impact (5.92%)

Successful integration of the HPDcache in OpenPiton [6]

- Were able to boot Linux on a 64 core (CVA6) configuration

→ Evaluate HPDcache performance in a multi-core environment

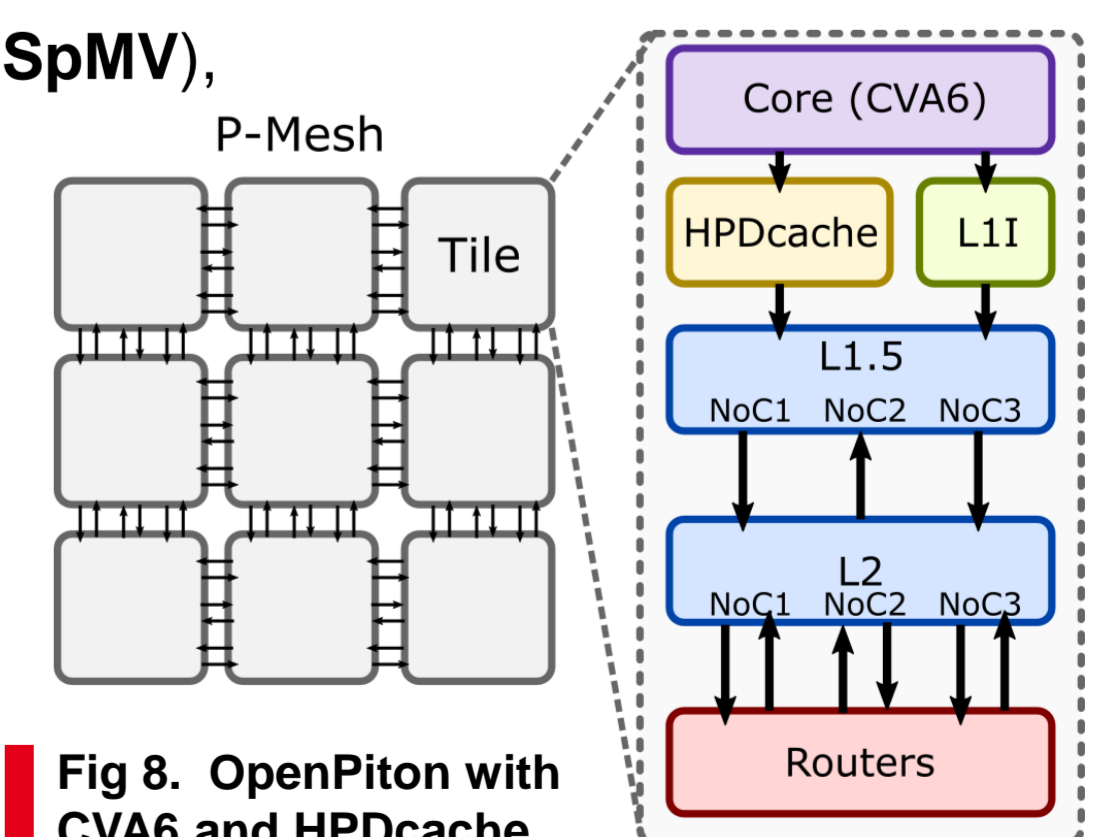


Fig 8. OpenPiton with CVA6 and HPDcache overview

Phase 1: Single-core performance evaluation

Phase 2: Multi-core performance evaluation

[1] K. Chang, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization," in SIGMETRICS '16, 2016.

[2] D. Kraft, "LookUp-free Instruction Fetch/Prefetch Cache Organization," in Proc. 8th Ann. Int'l Symp. Comput. Architect., 1991.

[3] C. Fuguet, "HPDcache: Open-Source High-Performance L1 Data Cache for RISC-V Cores," in Proceedings of the 20th ACM International Conference on Computing Frontiers, Association for Computing Machinery, 2023.

[4] <https://github.com/openhwgroup/cv-hpdcache>

[5] M. Beebe et al., "RaiderSTREAM: Adapting the STREAM Benchmark to Modern HPC Systems," in IEEE High Performance Extreme Computing Conference, 2022.

[6] <https://github.com/PrincetonUniversity/openpiton/pull/136>