CVA6S+: A Superscalar RISC-V Core with High-Throughput Memory Architecture

Riccardo Tedeschi[1]*, Gianmarco Ottavi[1], Côme Allart[2,3], Nils Wistoff[4], Zexin Fu[4], Filippo Grillotti[5], Fabio De Ambroggi[5], Elio Guidetti[5], Jean-Baptiste Rigaud[3], Olivier Potin[3], Jean Roch Coulon[2], César Fuguet[6], Luca Benini[1,4] and Davide Rossi[1]

[1]DEI, University of Bologna, Bologna, Italy
[2]Thales DIS, Meyreuil, France
[3]Mines Saint-Etienne, CEA, Leti, Centre CMP, F-13541 Gardanne, France
[4]IIS, ETH Zurich, Zurich, Switzerland
[5]STMicroelectronics, Agrate Brianza, Italy
[6]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, TIMA, 38000 Grenoble, France

Abstract

Open-source RISC-V cores are increasingly adopted in high-end embedded domains such as automotive, where maximizing instructions per cycle (IPC) is becoming critical. Building on the industry-supported open-source CVA6 core and its superscalar variant, CVA6S, we introduce CVA6S+, an enhanced version incorporating improved branch prediction, register renaming and enhanced operand forwarding. These optimizations enable CVA6S+ to achieve a 43.5% performance improvement over the scalar configuration and 10.9% over CVA6S, with an area overhead of just 9.30% over the scalar core (CVA6). Furthermore, we integrate CVA6S+ with the OpenHW Core-V High-Performance L1 Dcache (HPDCache) and report a 74.1% bandwidth improvement over the legacy CVA6 cache subsystem.

Introduction

CVA6 [1] is an application-class RISC-V core from the PULP platform, currently maintained by OpenHW Group. It features a six-stage, in-order pipeline and supports both ASIC and FPGA implementations, with a configurable 32- or 64-bit data path width. However, its IPC (Instructions Per Clock) is constrained by its simple, scalar in-order front-end microarchitecture.

Allart et al. [2] introduced a superscalar variant, CVA6S, supporting dual-issue execution. CVA6S retains the CVA6 pipeline structure with a 64-bit instruction fetch bus capable of fetching two 32-bit or four 16-bit compressed instructions per cycle. The decode and issue logic are duplicated, and a second ALU is added. We focus on enhancing and extending CVA6S. We introduce the CVA6S+ dual issue RISC-V core, which achieves a 43% IPC improvement over the scalar CVA6 and an 11% improvement over CVA6S in the Embench IoT benchmarks.

Additionally, we integrate CVA6S+ with the HPDCache [3], a highly configurable, pipelined data cache designed to deliver an Out-of-Order (OoO), non-blocking cache subsystem for RISC-V cores. Compared to the previous cache subsystem, we demonstrate an average 74% bandwidth improvement in memory-intensive accesses under both regular and irregular access patterns, using the RaiderSTREAM benchmarks. The total area overhead of CVA6S+ over the scalar CVA6 core is less than 10%.

*Corresponding author: riccardo.tedeschi6@unibo.it
This work has received funding from the Swiss State Secretariat for Education, Research, and Innovation (SERI) under the SwissChips initiative.

Implementation

Building on CVA6S, we introduce key enhancements to improve its microarchitecture to boost performance:

1. **Register Renaming:** To mitigate Write-After-Write (WAW), register renaming tracks the latest instruction writing to integer or floating-point registers, ensuring correct operand forwarding.
2. **Branch Prediction:** A two-level branch predictor with private history per entry (128 entries, 3-bit history) is integrated, reducing misprediction penalties by 30% across the Embench-IoT suite compared to the simpler bimodal predictor used in CVA6 and CVA6S.
3. **ALU-to-ALU Operand Forwarding:** To reduce execution latency, lightweight operand forwarding is introduced for cases where two ALU instructions are issued in the same cycle. This allows the second instruction to directly use the result of the first one without delay.
4. **FPU Integration:** The floating-point unit (FPU) is integrated into the dual-issue CVA6S+, as CVA6S lacks this feature. To minimize area overhead, it shares the write-back (WB) port with the secondary ALU, requiring additional hazard logic to prevent contention.

Furthermore, we replace the blocking architecture of the legacy WB cache with the HPDcache. It features a three stage pipeline with multiple request ports, a deep MSHR, out-of-order execution, a hardware prefetcher, and support for write-through and write-back policies, cache management, and atomic operations.

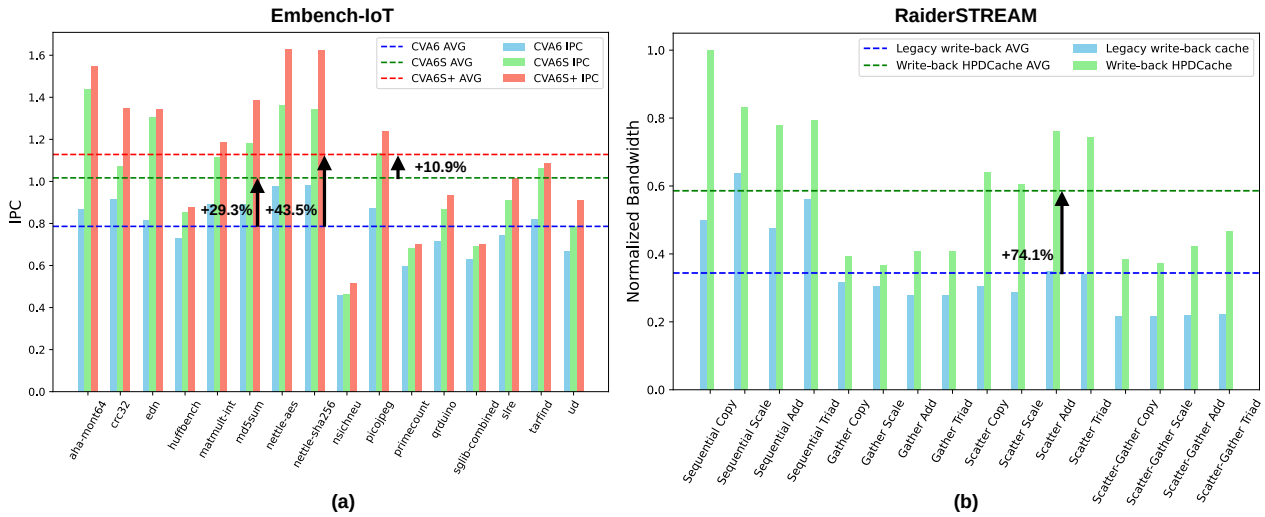

Figure 1: *Performance assessment of the different CVA6 versions and cache subsystems. (a) compares IPC improvements on the integer kernels of the Embench IoT suite, (b) the bandwidth gain from adopting the HPDCache.*

Evaluation

We use Cheshire [4] as our testbed, a modular, Linux-capable SoC platform designed to support application-class cores. The SoC is implemented on the Genesys 2 Kintex-7 FPGA. Our evaluation considers the 32-bit versions of CVA6, CVA6S, and CVA6S+, all supporting the `RV32IMAC` ISA along with the `a`, `b`, `c`, and `s` bitmanip extensions. The FPU is available only in CVA6 and CVA6S+. Thus, the `F` extension is not present in CVA6S. The write-back data cache (legacy or HPDCache) is 32 kiB with 8-way associativity, while the instruction cache is 16 kiB with 4-way associativity. An optional Load Unit output register is enabled to improve timing closure, adding one cycle of latency for load operations. To assess pipeline improvements, we use the integer kernels from the Embench-IoT suite, as the working sets fit into the core caches. As shown in Figure 1a, we observe an average IPC gain of 43.5% over the scalar version and a significant 10.9% improvement over CVA6S. CVA6, CVA6S and CVA6S+ achieve 2.83, 3.41 (+20.2%) and 3.69 (+30.2%) CoreMark/MHz respectively.

We use the RaiderSTREAM [5] benchmarks to evaluate the cache subsystem of CVA6S+ with the legacy data cache and the HPDCache. The working set is set to ×2 the data cache size. Figure 1b reports the bandwidth for sequential and non-sequential memory access patterns, with an average improvement of 74.1%.

Table 1 presents the topographical synthesis results using the GF22 FDX process under the worst-case conditions of 0.72V at -40°C. Since CVA6S lacks the FPU, resulting in a smaller area, the analysis focuses on CVA6, CVA6S+, and their cache subsystems. The overall area of CVA6S+ increases by only 9.30%. Excluding the caches, the pipeline area overhead is 28.6%. Switching to the HPDCache in CVA6S+ leads to an area reduction of 19% of the DCache due to more efficient SRAM organization. Less than 0.5% frequency degradation is reported from CVA6 to CVA6S+. HPDcache's area savings enable more synthesis optimizations, boosting the maximum frequency by 13.5%.

Table 1: *Area at 900 MHz and frequency estimations for the different cores and cache subsystems (16 kiB 4 way ICache, 32 kiB 8 way DCache).*

Core	DCache	Pipeline [mm2]	ICache [mm2]	DCache [mm2]	Max. Freq. [MHz]
CVA6	HPDCache	0.057	0.041	0.077	1090
CVA6S+	Legacy	0.070	0.041	0.095	960
CVA6S+	HPDCache	0.073	0.041	0.077	1095

Conclusion

We introduce CVA6S+, an enhanced superscalar extension of the CVA6 RISC-V application-class core, adding key features upon its existing dual-issue architecture CVA6S. CVA6S+ integrates the OpenHW HPDCache, combining pipeline improvements with a non-blocking cache subsystem. Our results show a 43.5% IPC gain and a 74.1% bandwidth boost with the HPDCache, while incurring under 10% area overhead.

References

[1] Florian Zaruba and Luca Benini. "Cost of Application-Class Processing: Energy and Performance Analysis of a 1.7-GHz 64-bit RISC-V Core". In: *IEEE Trans. VLSI Syst.* 27 (2019), pp. 2629–40.

[2] Côme Allart et al. "Using a Performance Model for Superscalar CVA6 Implementation". In: *Proc. ACM CF: Workshops.* 2024, pp. 43–6.

[3] César Fuguet. "HPDcache: Open-Source High-Performance L1 Data Cache for RISC-V". In: *Proc. 20th ACM CF.* 2023, pp. 377–8.

[4] Alessandro Ottaviano et al. "Cheshire: A Lightweight, Linux-Capable RISC-V Host Platform". In: *IEEE Trans. Circuits Syst. II: Express Briefs* 70 (2023), pp. 3777–81.

[5] Michael Beebe et al. "RaiderSTREAM: Adapting STREAM for Modern HPC Systems". In: *Proc. IEEE HPEC.* 2022, pp. 1–7.