# LEN5: an Out-Of-Order, Modular, Edge-Oriented RISC-V CPU

Vincenzo Petrolo, Flavia Guella, Michele Caon, Guido Masera, and Maurizio Martina

{vincenzo.petrolo,flavia.guella,michele.caon,guido.masera,maurizio.martina}@polito.it

VLSI Lab, Politecnico di Torino

**Abstract**

*Data-driven workloads expose the limits of traditional embedded systems, driving the shift toward heterogeneous Systems on Chip (SoCs) that combine Central Processing Unit (CPU) versatility with accelerator efficiency. However, optimizing scheduling and resource usage at compile time remains challenging. This work presents LEN5, a 64-bit RISC-V processor with a modular Out-of-Order (OoO) execution pipeline designed to leverage Instruction-Level Parallelism. By efficiently handling dependencies and masking latency, LEN5 improves performance achieving over 20% higher Instructions Per Cycle (IPC) than in-order designs and up to 20% frequency boost compared to a edge-oriented CPU. Additionally, its 64-bit Instruction Set Architecture (ISA) reduces the instruction count for precision-sensitive workloads by up to 2.4×.*

## Introduction

In recent years, the shift towards data-driven workloads has emphasized the limitations of traditional Von Neumann architecture. Data-driven algorithms and Moore's Law's slowdown require moving computation closer to data sources. Heterogeneous embedded SoCs tackle edge computing challenges by offloading tasks to memory-mapped accelerators, boosting speed and efficiency over CPUs. but requiring specialized software and ad-hoc scheduling. However, variable latency offloaded instructions complicate CPU stall mitigation. This work presents LEN5, a highly configurable, modular 64-bit RISC-V CPU featuring in-order issue, OoO execution and OoO commit. The main features of LEN5 architecture are:

- Modular core infrastructure for easy configuration and extension.
- Efficient latency masking and dependency handling to ensure high utilization.

Preliminary Embench results show LEN5 's advantages but highlight limitations with complex workloads. LEN5 improves clock frequency at a moderate area cost, making it suitable for edge-oriented SoCs.

## CPU microarchitecture

LEN5 microarchitecture (Figure 1) employs branch prediction, speculative OoO execution, and OoO commit. LEN5 's minimal setup supports the `RV64I` base ISA and `Zicsr` extension, with the option to enable `M`, `F` and `D` extensions.

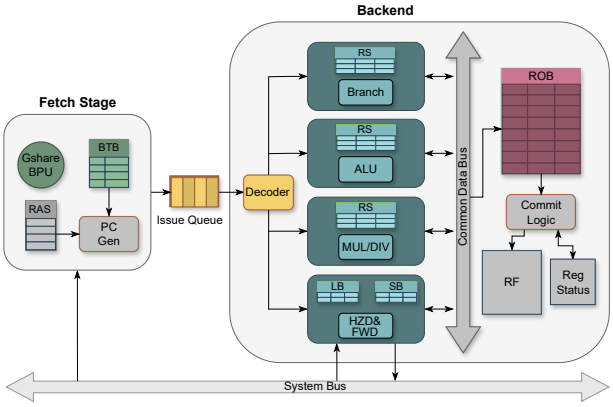


**Figure 1:** *Block diagram of the LEN5 microprocessor.*

**Instruction Fetch** A *gshare* predictor and Branch Target Buffer update the Program Counter (PC) for fetched branches and jumps, reducing misprediction penalties. Instructions enter an Issue Queue (IQ) before decoding and dispatch. On mispredictions, the fetch unit updates the PC and repopulates the IQ while the backend completes prior instructions.

**Instruction Execution** LEN5 is based on Tomasulo's scheduling for OoO execution. Decoded instructions enter a Reservation Station (RS) inside the target Execution Unit (EU) and wait in the ReOrder Buffer (ROB) for commit. Operands come from the Common Data Bus (CDB), ROB, or register file. If unavailable at dispatch, instructions wait in the RS until execution completes, allowing OoO scheduling. Results broadcast via the CDB enable dependency resolution. Thanks to LEN5's distributed control flow and execution isolation, custom extensions are integrated by extending the main instruction decoder, adding

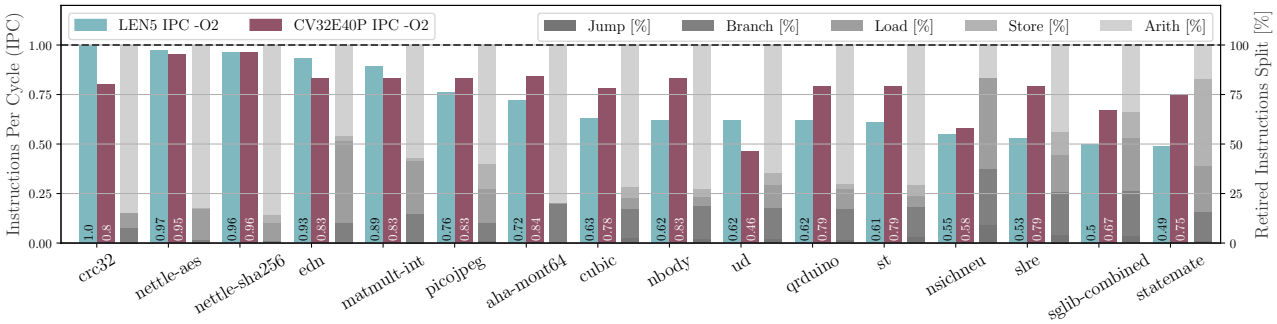


**Figure 2:** *Instructions Per Cycle comparison with* `cv32e40p` *over the Embench suite* (colour) and executed instruction composition (greyscale).*

*`huffbench` and `minver` did not finish on LEN5 and `cv32e40p` respectively.

dedicated RS, and compute engine to the backend.

**Instruction Commit** LEN5 's ROB prioritizes oldest instruction commits but allows OoO commits if no Write-After-Write (WAW) hazards exist. This allows LEN5 to commit instructions out of program order whenever the oldest instruction in the ROB has not yet completed its execution. Memory operations are handled by an OoO load buffer and in-order store buffer, which also acts as a level-zero cache, reducing memory accesses and improving efficiency.

## Logic Synthesis

LEN5 synthesis is performed on a 65 nm low power technology node in worst-case conditions. The modularity of LEN5 is shown by synthesizing three configurations. The first variant, shown in Table 1 as *Max Perf*, features a multiplier, a divider, and extends data structures to reduce stalls and achieve high performance while relaxing area constraints. *Min Area* due to the absence of multiplier and divider is suitable for area-constrained systems and workloads that tolerate longer execution times. Unlike *Max Perf*, the *Avg Perf* variant features no divider yet achieving comparable IPC limiting the area. The maximum frequency the core can achieve at 65 nm is 578 MHz. The obtained results are compared with `cv32e40p`[1] synthesized with the same technology and timing constraints. Every configuration of our core has a higher clock frequency, showing a 5 % to 20 % frequency improvement. The area overhead is estimated integrating our core into the open-source X-HEEP Microcontroller Unit (MCU) system [2] with 8 × 32 KiB SRAM banks, synthesised at a 4 ns clock period. Replacing `cv32e40p` with LEN5 *Max Perf* as the system CPU would cause an overall area increase of 12.7 %.

**Table 1:** *Area and Clock Frequency Comparison*

| | Memory Input Delay [ns] | Max Perf | Avg Perf | Min Area | cv32e40p |
|---|---|---|---|---|---|
| Clk Freq [MHz] | 0 | 490 | 394 | 578 | 465 |
| Area [µm²] | | 395352 | 223211 | 200117 | 55908 |
| Area [kGE] [1] | | 274 | 155 | 139 | 39 |

[1] GE is the 2-input drive strength-one NAND gate equivalent area.

## Benchmarking Results

LEN5 IPC performance was evaluated against the 32-bit in-order `cv32e40p` using the Embench suite, with results obtained from the *Max Perf* variant. As shown in Figure 2, LEN5 achieves higher IPC, especially in benchmarks with minimal control flow complexity. Notably, `crc32` reaches an IPC of 1.0, improving by over 20 % due to LEN5's dynamic instruction reordering. Similar gains appear in `edn` and `matmult-int`, benefiting from efficient vector and matrix operations. However, LEN5 struggles with benchmarks featuring unpredictable jumps, where higher misprediction penalties outweigh its advantages. The 64-bit `aha-mont64` benchmark demonstrates the ISA's efficiency, retiring 2.4 × fewer instructions than its 32-bit counterpart, with similar reductions in `statemate` (2 ×), `st` (1.4 ×), and `matmult-int` (1.6 ×).

## Conclusion

This work introduces LEN5, a versatile RISC-V CPU featuring OoO execution and commit, demonstrating notable gains in IPC compared to simpler architectures, along with increased clock frequencies. Future work will focus on enhancing branch prediction, supporting multiple issue capabilities, and conducting comparative analyses against leading-edge OoO CPUs.

## References

[1] Michael Gautschi et al. "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2017).

[2] Simone Machetti, Pasquale Davide Schiavone, Thomas Christoph Müller, et al. "X-HEEP: An Open-Source, Configurable and Extendible RISC-V Microcontroller for the Exploration of Ultra-Low-Power Edge Accelerators". In: *arXiv preprint* (2024).