ExaVerif

Exhaustive Verification for RISC-V Custom Instructions

SSCCS Foundation

2026-08-06

Code	References	Reports	Reports
Github	neXus	CVA6 CV-X-IF	Ibex RV32IMCB
References	Other Formats		
OpenHW	HTML		

The Project

ExaVerif is an open-core (Apache 2.0) CLI engine for exhaustive verification of RISC-V custom instruction extensions. A single command consumes a YAML specification of a custom instruction extension, enumerates every valid combination of its fields, and evaluates each one deterministically:

```
ev verify --target spec.xif.yaml
```

The specification declares every instruction field, its range, alignment, and constraints (oneof, range, bitmask, cross, eq, neq, and others). The CLI provides three subcommands: `verify` for constraint evaluation, `simulate` for instruction execution, and `synth` for RTL generation and synthesis. Each result is recorded as a structured Fact, consumable by the neXus pipeline and by autonomous agents. No result is discarded; every run raises the baseline, so verification output accumulates across projects, teams, and design phases instead of resetting with each tapeout decision.

The engine is validated against two production RISC-V cores, CVA6 (OpenHW Group) and Ibex (lowRISC), with exact agreement against their hardware decoders. Expansion to a broad range of additional industry silicon specifications is planned.

The Spec Space

The unit of verification is the Spec Space, a tripartite model:

- Axes: field domains, each an instruction field with its admissible values.

- Constraints: admissible subspaces over the axes, including oneof, range, bitmask, cross, eq, and neq.
- Projector: a mapping from each point of the space to a result value.

The engine compiles constraints into the enumeration itself. The iterator visits the valid space directly instead of the full cross product, so a specification completes in milliseconds regardless of constraint complexity. This structural enumeration is the source of the speedups measured in the benchmarks below, and it is independent of the target: any deterministic system that maps onto axes, constraints, and a projector benefits from the same pipeline.

Why the Gain Is Algorithmic

The speedup is an O(N) vs O(V) structural gain, not a language effect.

1. Standard `evaluate_all` checks every combination of the full space N; `struct_enum` visits the valid space V directly, constraints encoded into the enumeration.
2. Both pipelines are Rust, the same release build, measured by the same harness: the ratio is the enumeration strategy alone, and it survives a port to any language.
3. riscv-dv (UVM/simulator) and formal tools (SAT/BMC) do not enumerate at all; comparing against them is comparing apples to oranges.
4. Interpreted verification software (e.g., Python-based) dominates the industry landscape. The baseline here is a release-build Rust standard implementation, which in the combined assessment of memory safety-efficiency and performance already holds an overwhelming lead over C/C++; the structural gain therefore stands against an already high-performance baseline.

Benchmarks

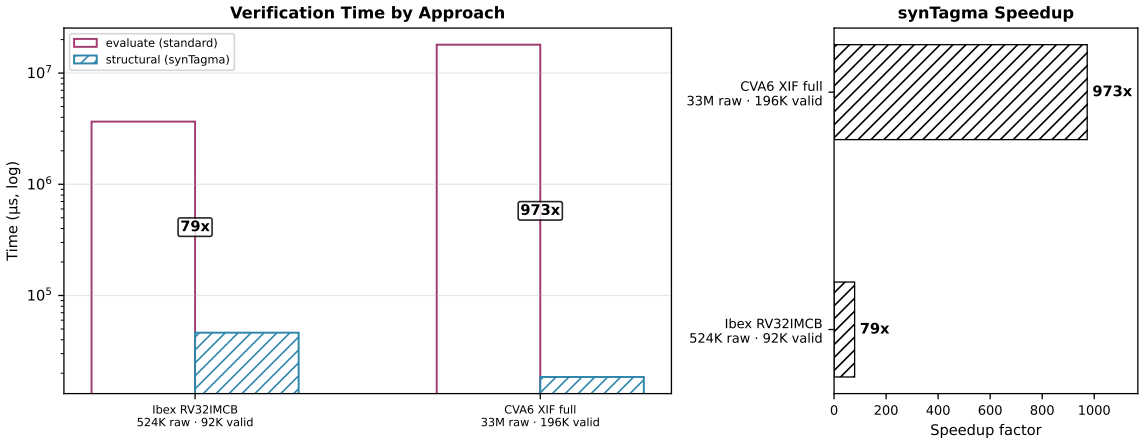

Figure 1: Verification time: standard evaluate vs structural pipeline. At CVA6 33M scale, structural_verify achieves ~973x speedup.

Scaling: O(N) vs O(V)

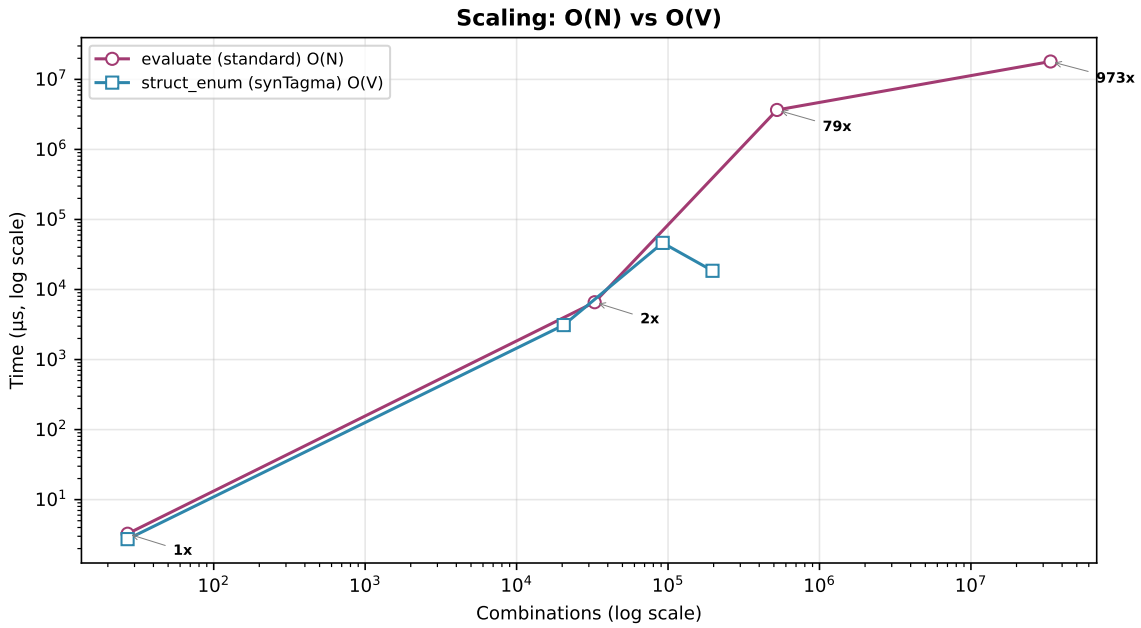

Figure 2: Scaling across fixtures: evaluate runs in O(N), struct_enum in O(V). The gap widens as valid density drops; CVA6 full (0.6% density) shows 973x.

Speedup follows approximately `k / D` (D = valid density): sparse spaces gain the most (CVA6 full 0.6% density, 973x; Ibex 17.6%, 79x; dense spaces 1-2x).

Fixture Speedups

Fixture	Raw	Valid	evaluate	structural	Speedup
Small	27	27	3.26 µs	2.73 µs	1.2x
Medium	32,768	20,480	6.58 ms	3.11 ms	2.1x
Ibex R-type	524,288	92,160	3.66 s	46.3 ms	79x
CVA6 R4	16,384	2,560	4.93 ms	247 µs	20x
CVA6 full	33,554,432	196,608	18.0 s	18.5 ms	973x

Status

- **CVA6 CV-X-IF** (OpenHW): 33.5M combinations, 196,608 valid, 18.0 s vs 18.5 ms (~970x). Space derived from the hardware decoder mask table; the DV class generates encodings the decoder rejects, a documented divergence. Master report: CVA6 CV-X-IF Report (Standard Reference).

- **Ibex RV32IMCB** (lowRISC): 524,288 combinations, 92,160 valid, 3.66 s vs 46.3 ms (79x). Report: Ibex RV32IMCB Report.

Limits

- Exhaustive coverage grows exponentially with parameters; the speedup depends on valid-space density, not raw size.
- Counts are field-value combinations and equal distinct instruction words (no enable_mask collapsing in the CVA6 fixtures).

Ev is a project of the SSCCS Foundation. Inquiries: ev@ssccs.org.