



SystemVerilog Assertion Verifier

Proof, not simulation, that every assertion of your RTL design holds — for every input and every start state.

Simulation checks the input sequences someone wrote. The Giolit SystemVerilog Assertion Verifier checks **all of them**. It reads your RTL and its SystemVerilog Assertions, as they are, and answers for every property:

PROVED

No run of the design violates the assertion: any inputs your assumptions allow, any number of cycles. You get the proof — an invariant in your own signal names — and a signed certificate.

FAILED

A concrete run that breaks it, clock cycle by clock cycle, as a waveform (VCD) and as a re-play testbench for your own simulator.

0

wrong answers on the 340 properties of the release benchmark

219

of 231 designs with a known answer decided correctly (EBMC 160, SymbiYosys 104)



safety margins: how far each proved bound is from failing

100%

in-house engines: no third-party solver, runs fully on your premises

Why designs still fail in silicon

The expensive bugs are the input sequences nobody simulated.

An arbiter grants two masters in the same cycle after a particular request pattern. A FIFO reports full and empty together when a push and a pop meet at the wrap-around. A handshake drops a request when valid rises during the reset release. A state machine reaches a state its designer thought impossible.

Each needs a precise sequence of inputs, often many cycles deep. Constrained-random simulation finds the common ones; the corner cases are rare by nature, and they are found late — in emulation, in silicon, or by a customer.

The Giolit SystemVerilog Assertion Verifier reasons about **every** run of the design at once. When it says PROVED, there is no input sequence, of any length, that breaks the assertion. When there is one, it gives you the run.

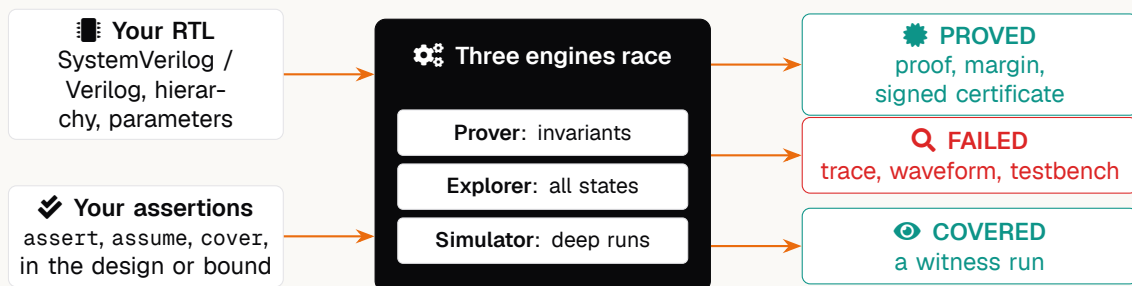
>_ One command

```
$ sva-verify --reset=rst --report=cert buffer.sv
PROVED assert margins.a_level
PROVED assert margins.a_latency
Safety margins:
a_level level <= 12 margin 3
a_latency ##[1:6] margin 3 cycles
Giolit Safety Resilience Score: ★★★★★
Report: cert/certificate.pdf
```

Exit codes and JSON for scripts and CI; a C++ SDK for your own flows.

How it works

Your RTL and its assertions in; a decision with evidence out.



The first engine to settle a property decides it. Every FAILED or COVERED result is a real run of the design; every PROVED result is backed by an inductive invariant or an exhaustive exploration of the reachable states.

Features, and what each one does for you

Every feature answers a question a designer, a verification lead or a customer asks.

SystemVerilog Assertions as written

Sequences and properties: `##[m:n]`, `[*]`, `[->]`, `|->`, `|=>`, `until`, `$past`, `$stable`, `disable iff`, immediate assertions.

✔ **Why it matters:** Your existing assertion IP is verified as it is; no rewriting into another language.

Liveness

`s_eventually` and fairness: proved for every run, or refuted with a lasso — a run that loops forever without the good thing happening.

✔ **Why it matters:** Starvation and deadlock are shown as concrete loops, not as a simulation that “ran long enough”.

Proof certificates

Every PROVED comes with its inductive invariant in your signal names. The PDF certificate, waveforms and testbenches are bound by SHA-256 in a statement signed with your license key; one script checks it.

✔ **Why it matters:** Evidence your customer or assessor can check without our tool — and without trusting it.

Debug-ready failures

Every FAILED and COVERED result as a cycle table, a VCD waveform and a self-checking replay testbench for Verilator or your simulator.

✔ **Why it matters:** The bug is reproduced in the tools your team already uses, in minutes.

C++ SDK

`libgiolit_sva`: verify, read proofs and counterexamples, compute margins and issue certificates from your own regression and sign-off flows. Examples included.

✔ **Why it matters:** Formal results become part of your flow, not a separate step.

Real designs

Hierarchy, parameters (`-G`), generate, interfaces, packages, memories, and checkers attached with bind — the files and options you pass to your simulator.

✔ **Why it matters:** The design stays untouched; properties can live in a separate file.

Assumptions and covers

`assume property` constrains the environment; `cover property` shows a witness or proves it **unreachable**.

✔ **Why it matters:** Unreachable covers expose dead logic and over-constrained assumptions before they hide a bug.

Safety margins and resilience score

For every proved bound (`count <= DEPTH`) and latency window (`##[1:6]`): how far it can be tightened and still hold. The weakest gives the Giolit Safety Resilience Score, 1 to 5 stars.

✔ **Why it matters:** Know whether a proof holds with room to spare or by a single count.

Several clocks

One global clock by default, as in SymbiYosys and EBMC; with `--multiclock`, every interleaving of the clocks' edges (dividers, gated and derived clocks, crossings).

✔ **Why it matters:** Clock-domain behaviour is verified, not assumed away.

On premises, offline

Every engine is Giolit's own: no external solver, no network, no upload. Linux x86-64.

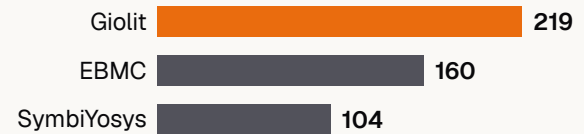
✔ **Why it matters:** Your RTL never leaves your machines.

Measured against SymbiYosys and EBMC

The same designs, the same 20-second budget per design, every result in the package.

	Giolit	SymbiYosys	EBMC 6.0
EBMC regression suite: correct, of 168	163	55	114
SymbiYosys examples: correct, of 63	56	49	46
TACAS 2016 designs decided, of 12	4	7	6
Wrong answers, on 340 properties	0	—	—
Proof certificate (signed invariant)	✓	—	—
Safety margin and resilience score	✓	—	—

All 231 designs with a known answer



Designs decided correctly. SymbiYosys with abc pdt, EBMC with a portfolio of its engines (k-induction, IC3, BDD, bounded). EBMC and SymbiYosys decide more of the bit-level TACAS 2016 designs. Every result: the Tool Comparison and the package's results/ folder.

Fits the way you work

A command, a library, a sign-off gate.

- › **Command line:** sva-verify design.sv — one line per property; exit codes 0 to 4 for scripts.
- › **CI:** JSON results; a failed assertion fails the build with the run attached.
- › **Quick checks:** -p 'EXPR' asserts always EXPR without editing the design.
- › **One property:** --property NAME verifies it on its cone of influence.
- › **Evidence bundles:** --report=DIR — certificate or report (PDF), proofs, waveforms, testbenches, signed statement.
- › **Licensing:** per-licensee signed licenses; a trial of 1000 runs over 50 days.
- › **Benchmark included:** the designs, their known answers and this release's results ship with the trial — rerun any of them.

Verification services

When you need the properties as well as the proofs.

Giolit engineers write the assertions for your design from its specification, prove them, and deliver the certificates: bus protocols and handshakes, data integrity through FIFOs and pipelines, arbitration and fairness, liveness, and formal sign-off evidence for safety standards (ISO 26262, DO-254, IEC 61508).

See your own design verified.

Send us a block and its specification: we write or review the assertions with you and show you the first proof — or the first run that breaks it.

 www.giolit.com
 business.giolitlabs@gmail.com
 Giolit Labs, Thrissur,
Kerala, India