



GIOLIT | FORMAL VERIFICATION OF SYSTEMVERILOG DESIGNS

# Certificate of Formal Verification

Document GIO-SVA-94B352394746 | issued 2026-10-14 16:24:49 UTC (UTC+0000)

**PROFESSIONAL LICENSE No. GIOLIT-PRO-2026-000001** licensed to **Giolit Labs**

## Every assertion of lockstep holds.

For every input sequence, from every state after the reset `rst`, in every clock cycle:

✓ 2 assertions proved: `a_same`, `a_limit`

This is a **mathematical proof** covering all runs of the design, not a simulation of some of them. The proofs are part of this certificate (page 3).

|               |                                                                                                                    |
|---------------|--------------------------------------------------------------------------------------------------------------------|
| Design        | top module <code>lockstep</code> , 2 properties                                                                    |
| File          | <code>10_lockstep.sv</code> (28 lines)<br>SHA-256 30a5f12b670bb4923d2e0835a52ddce4e894b9b524716b737203693c847753e1 |
| Reset         | <code>rst</code> active in the first clock cycle                                                                   |
| Issued to     | <b>Giolit Labs</b>                                                                                                 |
| Machine ID    | 5186e2bbb5474f2bb16093c28c3e092a                                                                                   |
| Machine       | 95bb1e6c01c1 — Ubuntu 24.04.5 LTS (kernel 7.2.9-orbstack-00391-g2b298d05fbd4); VirtualApple @ 2.50GHz              |
| Verified by   | 95bb1e6c01c1                                                                                                       |
| Engine        | Giolit SystemVerilog Assertion Verifier 1.0.0, build 29f9c97                                                       |
| Date and time | 2026-10-14 16:24:49 UTC (UTC+0000) (analysis 1.6 s)                                                                |
| Technique     | Inductive proof with refinement (CEGAR)                                                                            |
| Resilience    | not rated: no numeric bound in the proved assertions                                                               |
| License       | <b>Professional license No. GIOLIT-PRO-2026-000001</b> , licensed to Giolit Labs                                   |

## How your client can check this document

- Same design.** `sha256sum 10_lockstep.sv` must print `30a5f12b670bb4923d2e0835a52ddce4e894b9b524716b737203693c847753e1`.
- Licensed.** Issued under Giolit professional license No. **GIOLIT-PRO-2026-000001**, licensed to **Giolit Labs**.
- Same result.** `reproduce.sh` re-runs the verification; every counterexample replays in a simulator (evidence/).
- Ask us.** Write to [business.giolitlabs@gmail.com](mailto:business.giolitlabs@gmail.com) quoting document GIO-SVA-94B352394746.



## GIOLIT — Formal Verification Services

Mathematical proof that your RTL design meets its SystemVerilog assertions.

We offer formal verification as a service: proofs of the SystemVerilog assertions of RTL designs (protocols, data integrity, arbitration, liveness), formal sign-off evidence for safety standards (ISO 26262, DO-254, IEC 61508), assertion writing, and review of counterexamples with your engineers.

[www.giolit.com](http://www.giolit.com) [business.giolitlabs@gmail.com](mailto:business.giolitlabs@gmail.com)

[www.giolit.com](http://www.giolit.com)

## Properties and results

| # | Property         | Kind   | Line | Result   | Method or reason                        |
|---|------------------|--------|------|----------|-----------------------------------------|
| 1 | lockstep.a_same  | assert | 26   | ✓ proved | Inductive proof with refinement (CEGAR) |
| 2 | lockstep.a_limit | assert | 27   | ✓ proved | Inductive proof with refinement (CEGAR) |



## The proofs

For each proved property: the property as written, the method that proved it, and the proof in terms of the design's signals, simplified (relations implied by the signal widths are omitted). The complete machine-readable proofs are in `proof.json`.

### `lockstep.a_same` (line 26, proved)

```
a_same: assert property (a == b);
```

**Method:** Inductive proof with refinement (CEGAR). The design and its property checkers were encoded as constrained Horn clauses over the clock-cycle model and solved by bidirectional abstract interpretation with counterexample-guided refinement: every abstract counterexample was checked against the exact semantics and refuted.

**The proof.** Let `Inv` be the relation below over the state of the design at a clock edge (its registers and memories, the sampled values used by the assertions, and the states of the property checkers). The verifier established: (1) `Inv` holds at the first clock edge, for every start state; (2) if `Inv` holds at a clock edge, it holds at the next one, for every input; (3) in a cycle that starts in a state satisfying `Inv`, the property is not violated. So the property holds in every cycle of every run.

`Inv` = one of these cases:

**case 1** `$initstate = 1`

**case 2** `$initstate = 0 ∧ a = 0 ∧ b = 0`

**case 3** `$initstate = 0 ∧ 0 ≤ b ≤ 12000000 ∧ b = a`

where `$initstate` is 1 in the first clock cycle only. Every other name is a signal of the design, `x[k]` bit `k` of `x`.

### `lockstep.a_limit` (line 27, proved)

```
a_limit: assert property (b <= 24'd12_000_000);
```

**Method:** Inductive proof with refinement (CEGAR). **The proof** is the same invariant `Inv` as for `lockstep.a_same` above: no state in it violates this property either.

## Safety margins and the Giolit Safety Resilience Score

A proof says the assertion holds; the **safety margin** says by how much. For every numeric bound of a proved assertion — a comparison such as `count <= DEPTH`, or the upper end of a delay window such as `##[1:3]` — the verifier tightened the bound step by step and proved the tightened assertion again. The margin is the largest tightening that still holds: the distance between the values (or latencies) the design actually reaches, over all runs, and the closest value the assertion forbids. It is exact when tightening one step further fails with a counterexample; otherwise it is a lower bound ( $\geq$ ). A margin of 0 means the design reaches its bound — often by design, such as a FIFO that fills up — and that any change that moves the design toward the bound turns the verdict.

| Property             | Bound                               | Margin   | Allowed         | Relative | Score |
|----------------------|-------------------------------------|----------|-----------------|----------|-------|
| <code>a_limit</code> | <code>b &lt;= 24'd12_000_000</code> | $\geq 0$ | 12000001 values | —        | —     |

**The score.** Each bound is scored by its relative margin (margin / size of the allowed range): ★★★★★★ from 50%, ★★★★★☆ from 25%, ★★★★★ from 10%, ★★★★★ above 0, ★★★★★ when the design reaches the bound. The design's **Giolit Safety Resilience Score** is the score of its weakest bound: not rated.

## Semantics and scope

- **Clock.** One step of the model is one clock cycle. Every clocked process is clocked by one global clock unless the run used `--multiclock` (exact edge semantics for several or derived clocks).
- **Inputs.** The top module's inputs are free in every cycle: the results cover *every* input sequence, constrained only by the design's assume properties.
- **Start state.** Registers without an initial value start with arbitrary values; the reset `rst` is active in the first cycle and free afterwards.
- **Values.** Two-state semantics: X and Z values are arbitrary 0/1 values; asynchronous resets act at the clock edge. Several drivers of a net that can release it (z) are checked for contention.
- **Assertions.** Concurrent assertions follow IEEE 1800-2017 (sampled values, `disable iff`, `$past`); liveness properties are proved by k-liveness or refuted by an infinite run (lasso).

## The design

### 10\_lockstep.sv

SHA-256 30a5f12b670bb4923d2e0835a52ddce4e894b9b524716b737203693c847753e1

```
1 // Two 24-bit counters that must stay in lock-step and stop at ten million: far too many states to
  enumerate, so the
2 // proof is an inductive invariant over the registers.
3 // Expected (sva-verify --reset=rst --report=lockstep 10_lockstep.sv): a_same and a_limit PROVED;
  a_limit has a safety
4 // margin of 2,000,000.
5 module lockstep (
6     input logic      clk,
7     input logic      rst,
8     input logic      en,
9     output logic [23:0] a,
10    output logic [23:0] b
11 );
12 localparam logic [23:0] STOP = 24'd10_000_000;
13
14 always_ff @(posedge clk)
15     if (rst) begin
16         a <= '0;
17         b <= '0;
18     end else if (en && a < STOP) begin
19         a <= a + 24'd1;
20         b <= b + 24'd1;
21     end
22
23 default clocking cb @(posedge clk); endclocking
24 default disable iff (rst);
25
26 a_same: assert property (a == b);
27 a_limit: assert property (b <= 24'd12_000_000);
28 endmodule
```

## Reproducing this result

The bundle delivered with this document contains:

- 10\_lockstep.sv — the verified design file, unchanged

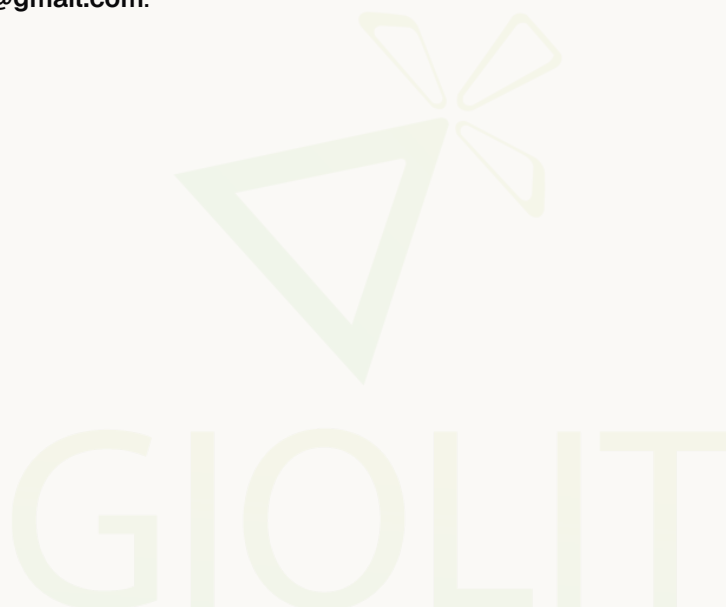
- `certificate.json` — the statement of this result, with the SHA-256 of every design and evidence file
- `proof.json` — the proofs (relations over the design state, and as stated by the engine) and the runs
- `results.json` — per-property results (machine-readable)
- `verifier-output.txt` — the verifier's console report
- `reproduce.sh` — the command that produced this result
- `certificate-password.txt` — the password that opens `certificate.pdf`

Command (run in the bundle folder):

```
sva-verify --reset=rst --timeout=900 --margins 10_lockstep.sv
```

## About Giolit

We offer formal verification as a service: proofs of the SystemVerilog assertions of RTL designs (protocols, data integrity, arbitration, liveness), formal sign-off evidence for safety standards (ISO 26262, DO-254, IEC 61508), assertion writing, and review of counterexamples with your engineers. Visit [www.giolit.com](http://www.giolit.com) or write to [business.giolitlabs@gmail.com](mailto:business.giolitlabs@gmail.com).



### Giolit — Formal Verification Services

Mathematical proof that your RTL design meets its SystemVerilog assertions.

We offer formal verification as a service: proofs of the SystemVerilog assertions of RTL designs (protocols, data integrity, arbitration, liveness), formal sign-off evidence for safety standards (ISO 26262, DO-254, IEC 61508), assertion writing, and review of counterexamples with your engineers.

 [www.giolit.com](http://www.giolit.com)

 [business.giolitlabs@gmail.com](mailto:business.giolitlabs@gmail.com)



[www.giolit.com](http://www.giolit.com)