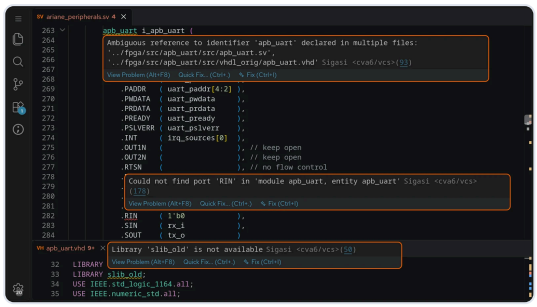
Sigasi Visual HDL

Built for engineers and AI agents.

Deterministic HDL analysis for ASIC and FPGA development. Deep semantic and project intelligence through MCP and CLI for your choice of coding agent, interactive design visualization in VS Code, and CI quality checks for seamless team collaboration.



SystemVerilog IEEE 1800-2023 · Verilog IEEE 1364-2005 · VHDL IEEE 1076-2019 · Mixed-language simultaneous parsing · VS Code · MCP · CLI · CI

OVERVIEW

Sigasi® Visual HDL™ is a deterministic semantic platform for professional ASIC and FPGA development. It parses SystemVerilog, Verilog, and VHDL simultaneously across mixed-language projects and builds a full semantic model of the design: types, scopes, libraries, interfaces, hierarchy, compile order, and dependencies with resolved names, references, and bindings.

That shared model updates incrementally as the design evolves, powering responsive analysis, navigation, visualization, and agent feedback.

Sigasi is not another AI coding assistant, and does not generate RTL. It is the project intelligence and validation layer that works together with the frontier coding agent that your team has already chosen and the engineers who write code, reuse IP, and review what that agent produces.

One engine, three ways in. Engineers work with it in VS Code through LSP. Coding agents in VS Code and Cursor, including Copilot, Claude Code, and Codex, call the engine through MCP. Sigasi exposes it as a set of MCP tools, so an agent can request diagnostics for a file or the whole project, read the project context, search symbols, and format code. CI pipelines and agents that run from the command line call the same engine through the CLI and its daemon.

Why it matters now

Generation is no longer the bottleneck. Frontier coding agents produce and modify HDL quickly. More generated code does not automatically mean more progress.

A model can read HDL sources, but it does not automatically know the exact library mappings, compile order, include paths, reused IP, language boundaries, or coding rules of a real project. Sigasi gives the agent those facts. For example: asked to fix a "library is not available" error, an agent reads Sigasi's diagnostic, adds the missing library mapping to the project configuration, and leaves the correct RTL untouched.

Fast enough for AI. Deterministic enough for silicon.

Trusted across

Aerospace, automotive, defense, energy, fintech, industrial, medical, semiconductor, and telecom. Sigasi has built professional HDL tooling since 2008. Ask for our reference list.

FROM GENERATED CODE TO A TRUSTED RESULT

AI / AGENT

Generate · Modify · Review

Copilot, Cursor, Claude Code, Codex, or the agent your team runs

→

SIGASI

Deterministic semantic layer

Full project context, real-time analysis, reproducible diagnostics

→

DOWNSTREAM TOOLS

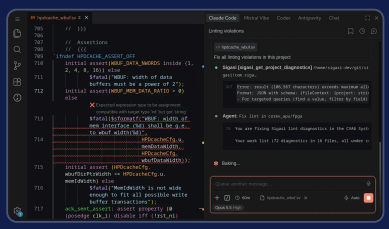
Synthesis · Simulation · Formal · CDC · Lint · DFT

Only reviewed, project-correct RTL moves forward

• SEMANTIC MODEL • REAL-TIME ANALYSIS • DETERMINISTIC RESULTS

AI generates. Sigasi grounds and validates. Engineers decide.

Model-neutral and toolchain-independent by design. Copilot, Claude Code, and Codex in VS Code and Cursor connect through MCP; agents that run from the command line call the same engine through the CLI. The engine itself can run fully air-gapped. Sigasi sits alongside whichever agent you run, not inside one.



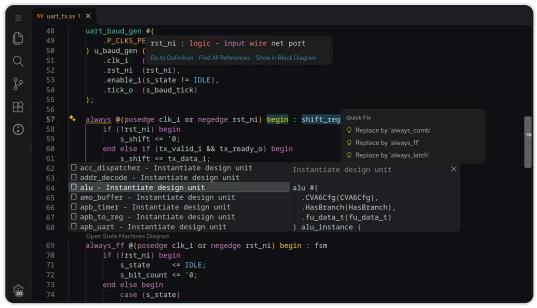
Claude Code fixing lint violations from Sigasi project diagnostics.

REAL-TIME FEEDBACK AS YOU TYPE OR GENERATE

Sigasi's incremental compiler reports syntax and semantic errors while the code is being written, whether an engineer is typing it or an agent is generating it. There is no need to invoke a simulator to find out that the project no longer compiles.

Diagnostics use the full project context, so an incorrect reference is distinguished from a missing library mapping, and an interface mismatch is distinguished from a typing error. Corresponding quick fixes are offered at the point of the problem.

- Syntax and semantic diagnostics across files and languages
- Configurable coding rules with per-rule severity levels
- Context-aware completion and auto-instantiation from the real project model, not from a guess
- Hovers, hover actions, and quick fixes on your code, written or generated
- Preprocessor power tools: autocomplete, expand, and inspect (hover)



Content assist instantiates a design unit with its full port map, resolved from the project model rather than from the open file.

NAVIGATION AND CODE UNDERSTANDING

Large hierarchies, reused IP, and mixed-language boundaries make a codebase hard to hold in your head. Sigasi Visual HDL maps out the design and makes it easy to navigate: jump to any declaration or reference across languages, inspect symbols, walk design and class hierarchies, search nets across the project, and more.

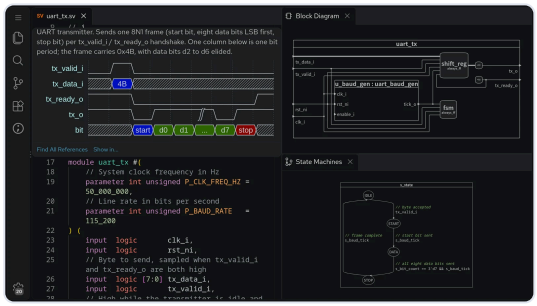
Rename refactoring is semantic rather than textual, so renaming a signal updates the declaration and every genuine use of it and leaves identically named but unrelated symbols alone. Formatting is applied deterministically from configured team rules, including Verible integration for SystemVerilog and Verilog.

DESIGN VISUALIZATION

Block diagrams, state machines, state-transition views, and dependency graphs are generated from your source code and update as the code changes. They show the design as it is now, not as somebody documented it once.

Move seamlessly between code and graphics: from a diagram to the declaration behind it, and from a transition clause to its place in the diagram. That makes an unfamiliar design readable, and a review much richer than a text diff, all the more important when the change under review was proposed by an agent.

- Block diagrams linked to source
- State machine diagrams and transition views
- File and target dependency views



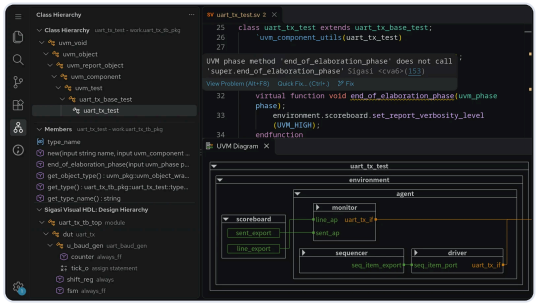
Block diagram and state machine views generated from the source, updated as the code changes.

VERIFICATION SUPPORT

Verification engineers need to understand both the design under test and the verification environment surrounding it. Sigasi applies the same real-time, project-aware analysis and visualization to verification code, with built-in support for industry-standard frameworks.

VUnit integration connects test execution right into VS Code. Workflows with OSVVM, UVVM, and cocotb are supported where the environment is configured for them. UVM-specific linting, UVM topology, and UVM diagrams help engineers understand and check class hierarchies and component relationships within the context of their project configuration.

> Sigasi supports verification development and downstream tool automation. It does not replace simulation, formal verification, coverage analysis, or sign-off.

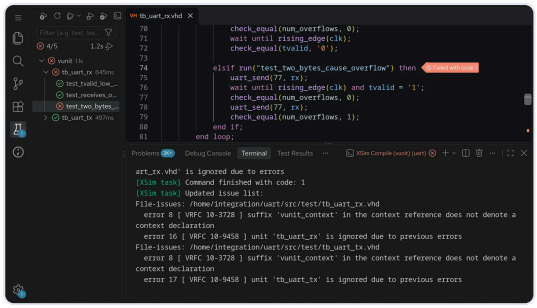


A UVM environment and its generated topology diagram, side by side with the testbench source.

AUTOMATION, CI, AND TOOLCHAIN FIT

The CLI runs the same semantic engine from the command line. Use it to check projects, produce machine-readable diagnostics with configurable severity thresholds, generate documentation and diagrams, export compilation order and recipes, and format code consistently across a team.

Sigasi runs alongside the editors, repositories, simulators, build scripts, and CI systems your team already uses. It fits into your existing workflow, catching issues early so costly downstream tools spend less time on avoidable errors. Modular Projects bring the power of software-proven modularity to HDL: reproducible, scriptable sources, libraries, dependencies, and targets, with documented Vivado and VUnit import workflows.

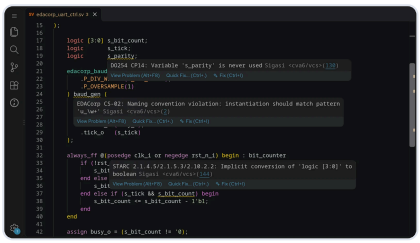


VUnit tests run from VS Code, with XSim compile output in the terminal: the existing toolchain reports in the same workspace as Sigasi's own diagnostics.

SHARED, ENFORCED CODING STANDARDS

Configurable rules let a team state its conventions once and have them applied identically to everyone's code, and to everything an agent writes. Deviations surface while the code is being written rather than at the review gate, and each rule carries its own severity.

This is the practice programs working to DO-254, ISO 26262, or STARC have to demonstrate rather than assert. Coding-rule checks are instrumental in supporting that process. They do not certify a design or guarantee compliance.



DO-254, STARC, and in-house rule violations, each with a quick fix.

10 REASONS TO CHOOSE SIGASI

01 Ground any coding agent in the real project: hierarchy, types, scopes, libraries, dependencies, and team rules, all there before it changes a line.

02 Check AI-generated RTL at interactive speed with the same deterministic engine your engineers already use to review it.

03 Act as a guardrail for your agent: Sigasi checks and lints AI-generated code as it is produced, through MCP in VS Code and Cursor, and through the CLI for agents that run from the command line.

04 Catch context errors such as unresolved dependencies, interface mismatches, and broken references, before they ever reach simulation.

05 Keep every AI-authored change reviewable against your team's own coding rules, including the practice expected in DO-254, ISO 26262, and STARC programs.

06 Use whichever agent or downstream EDA provider you want. Sigasi is independent of model provider, FPGA vendor, simulator, or synthesis tool.

07 Understand unfamiliar designs through block diagrams, state machines, and dependency views generated from your source.

08 Work with verification code as part of the engineering project: UVM topology, UVM diagrams, and methodology-specific linting.

09 Automate analysis, linting reports, documentation, and compilation recipes through the CLI and CI.

10 Analyze locally. HDL and project metadata are analyzed on your workstation or inside your company-controlled network.

WHAT THAT IS WORTH

ONE PROJECT, IN WEEKS	WITHOUT	WITH
RTL coding, review, and linting	12	6
Verification	16	12
Other project work	10	10
Total	**38**	**28**

10 weeks saved on one project, around a quarter of its duration

A typical illustration based on feedback from large customers, not a guarantee. Individual results vary by project. Run the numbers for your own team at sigasi.com/calculator.

CAPABILITIES AT A GLANCE

Navigate	Declarations and references across languages, hovers, outline, occurrence highlighting, design and class hierarchies, net search, preprocessor view
Edit	Content assist with auto-instantiation, semantic rename, quick fixes, formatting with Verible for SystemVerilog and Verilog
Check	Syntax and semantic diagnostics, configurable rules with severity per project, folder, or file
Visualize	Block diagrams, state machines with transitions, file and target dependencies, UVM topology
Verify	UVM linting, VUnit test execution in VS Code, OSVVM, UVVM, and cocotb workflows
Set up	Modular Projects, library mappings, include paths, dependencies such as UVM and Unisim, Vivado and VUnit import
Automate	CLI with machine-readable diagnostics and failure thresholds, documentation export, compile order and recipe export, daemon mode
Agents	MCP tools for diagnostics, project context, symbol search, and formatting

EDITIONS: DESIGNER, PROFESSIONAL, ENTERPRISE

One semantic engine, three depths of capability, from real-time analysis for the individual engineer to CLI-driven documentation and CI reporting for enterprise programs. Professional includes diagrams, UVM checks, and documentation generation in one license, with no separate product to buy. A complete feature list is available at sigasi.com/editions.

CAPABILITY	DESIGNER	PROFESSIONAL	ENTERPRISE
Project and external tools integration	✓	✓	✓
Hovers, hover actions, and quick fixes	✓	✓	✓
Full autocomplete and content assist	✓	✓	✓
Semantic and occurrence highlighting Color that reveals real intent, so mismatched types and wrong connections stand out.	✓	✓	✓
Real-time syntax error checking	✓	✓	✓
Real-time and cross-file linting	✓	✓	✓
Preprocessor autocomplete and hover	✓	✓	✓
New **Control AI-generated code in VS Code and Cursor through MCP** A guardrail and sparring partner: provides context to agents, and checks and lints generated code through MCP.	✓	✓	✓
Real-time and interactive diagrams Dependencies, FSMs, and block diagrams, generated from source.	—	✓	✓
State machine transitions view	—	✓	✓
UVM linting	—	✓	✓
UVM diagrams	—	✓	✓
Full project documentation generation and export Traceable documentation for audits and customer deliverables.	—	✓	✓
CLI linting reports	—	—	✓
CLI code formatting	—	—	✓

Security and deployment

Sigasi's engine and CLI run on your workstation or inside the company-controlled network; project files are not sent to Sigasi servers for semantic analysis. Commercial deployment supports a node-locked license file or an internal FlexNet server, and the engine can operate air-gapped. A connected coding agent is a separate component, with its own provider policy to review.

INTEGRATED SOLUTION

Sigasi Visual HDL is one engine behind VS Code, the CLI and its daemon, and CI, with Modular Projects for reproducible setup and documented Vivado and VUnit import paths for existing projects. Engineering teams in aerospace, automotive, defense, energy, fintech, industrial, medical, semiconductor, and telecom have used Sigasi since 2008.

Technical support

Our team answers technical questions, provides training, and helps scope how Sigasi fits into your workflow, with or without AI agents. Evaluate on a representative ASIC or FPGA project; a Trial Setup Session is available to get a real project configured and analyzed on the first day.

Contact Sigasi

Free trial	sigasi.com/trial
Support & evaluation	support@sigasi.com
Sales	sales@sigasi.com
Website	www.sigasi.com
Product manual	sigasi.com/manual

Sigasi analyzes HDL locally; project files are not sent to Sigasi servers for semantic analysis. Sigasi finds issues earlier and prepares engineering-ready RTL; it does not replace design review, simulation, formal verification, synthesis, or sign-off, and it does not certify a design or guarantee compliance. Engineers remain accountable for what moves downstream. Capabilities vary by edition and configuration. Sigasi® Visual HDL™ is a trademark of Sigasi nv. Copyright © 2026 Sigasi nv. All rights reserved. Information herein is subject to change without notice.