AI-Drievn Layout-Aware RTL Optimization Loop Pipeline × Cognix

AI-Powered RTL Verification & Self-Optimizing

Cognix (ACI) | Hackathon 2026

https://github.com/Bhanuu01/AI-Drievn-Layout-Aware-RTL-Optimization-Loop.git

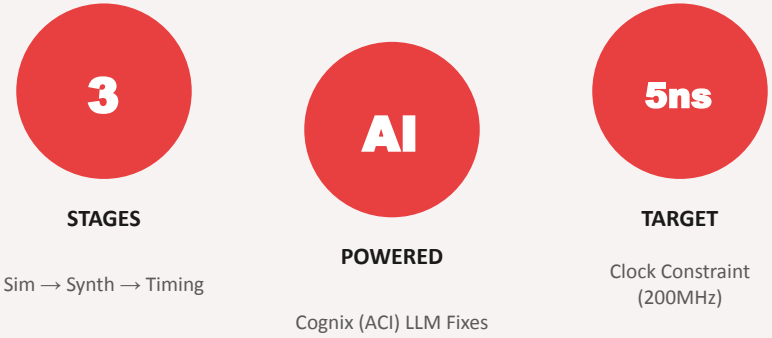
TEAM MEMBERS :

- **Bhanuja Karumuru**
- **Harshal Pimpalshende**
- **Abhishek Bharadwaj**
- **Saishruti Veda**

Table of Contents

About the Project

We build an AI-driven RTL verification and self-optimizing pipeline that automatically detects, diagnoses, and fixes hardware design bugs — reducing manual debug cycles from hours to seconds.



Verilator

Yosys

OpenSTA

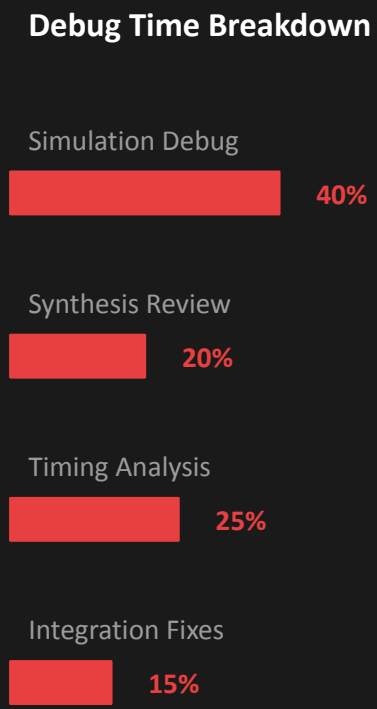
Cognix AI

The Problem

SLOW, MANUAL, ERROR-PRONE LAYOUT DESIGNING

RTL engineers spend 60–70% of project time on debugging. Traditional flows rely on manual review of Verilator logs, Yosys reports, and STA output — disconnected tools with no automated remediation. Subsequently, the RTL codes usually do not have design fixes for some parameters that can be optimized in the code before handover to layout engineer.

- No automated bug-fix loop
- Timing violations caught too late
- Context lost across tool boundaries
- Repeated manual iteration cycles



Our Solution

AI Self-Optimizing RTL Pipeline

An automated pipeline that ties Verilator simulation, Yosys pre-synthesis, and OpenSTA timing analysis into a single loop — with Cognix AI generating and applying targeted RTL fixes at each failure point.

- Auto-checks RTL + testbench files
- Runs full 3-stage verification
- Formats context-rich prompts for Cognix
- Watches for file save ⟶ auto-reruns
- Archives every successful run with logs

Sim Bugs Fixed

Verilator ⟶ Cognix loop

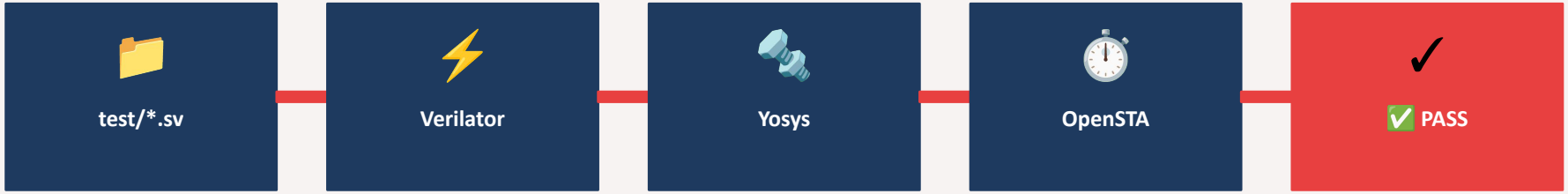
Synth Clean

Yosys latch/loop detection

Timing Met

STA WNS < −0.1 ns threshold

Pipeline Architecture



AI FIX LOOPS

SIM FAIL ⟶ Cognix formats prompt with Verilator + Yosys logs ⟶ Apply fix ⟶ Re-run

SYNTH FAIL ⟶ Cognix gets Yosys blocking issue context ⟶ Fixes latches/loops ⟶ Re-run

TIMING FAIL ⟶ Cognix receives WNS/TNS + critical path ⟶ Restructures RTL ⟶ Re-run

Why Now

The convergence of LLMs + open EDA tools makes this possible today.

LLMs Can Read Code

Models like Cognix understand SystemVerilog syntax, timing constraints, and synthesis semantics well enough to generate targeted RTL fixes.

Open EDA Maturity

Verilator, Yosys, and OpenSTA are now stable, scriptable, and freely available — enabling a fully automated verification chain.

AI-Native Dev Tools

IDE-integrated AI assistants (Cognix ACI) now support 'Apply' workflows, closing the loop between AI suggestion and file modification.

Hardware Complexity

SoC complexity is doubling every 2 years. Manual RTL debug is no longer sustainable — automation is a necessity, not a luxury.

Key Features

Each feature purpose-built for real RTL design environments.

🔁

Communication Pipeline

Acts as a direct pipeline to communicate between RTL Engineer and Layout Engineer.

🔁

Project Duration

An optimized RTL code run through multiple iterations saves months of work.

🔁

Layout Efficiency

Provides better design quality to ensure layout efficiency is relatively higher at the initial placement phase.

🔁

Auto-Detect Files

Scans test/ for RTL + TB, identifies top module automatically

🧠

Cognix AI Fix

Formats rich diagnostic prompts; watches for Apply → auto-reruns

🔩

Yosys Synthesis

Latch, loop, and multi-driver detection with netlist export

⏱️

OpenSTA Timing

WNS/TNS analysis with critical path breakdown for targeted fix

📦

Run Archiving

Full logs, VCD, RTL snapshot archived per successful run

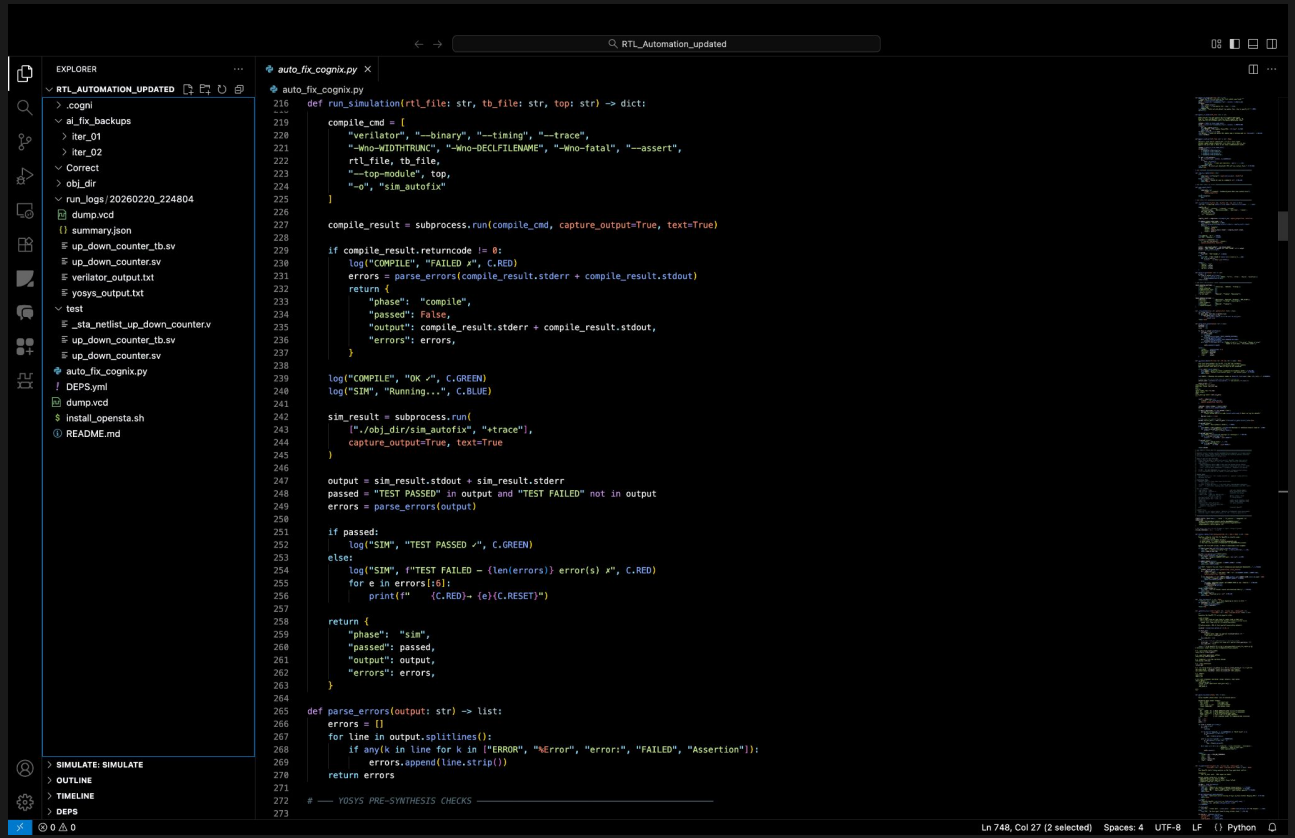
⚙️

CLI Configurable

--clock-period, --liberty, --skip-yosys and more flags

Demo & Results

Project Directories



Demo & Results

Terminal Process

```
bhanujakarumuru@10-20-121-90 RTL_Automation_updated % python3 auto_fix_cognix.py
[22:48:04][DETECT] RTL file : up_down_counter.sv
[22:48:04][DETECT] TB  file : up_down_counter_tb.sv
[22:48:04][DETECT] Top module (TB): tb_up_down_counter
[22:48:04][DETECT] RTL module (Yosys/STA): up_down_counter
[22:48:04][DETECT] Clock port heuristic: 'clock'
[22:48:04][STA] OpenSTA not found — skipping.
[22:48:04][STA] Build: https://github.com/parallaxsw/OpenSTA  |  brew install opensta

  RTL AUTO-FIX  ×  COGNIX  +  YOSYS  +  OpenSTA

  RTL          : up_down_counter.sv
  Testbench    : up_down_counter_tb.sv
  Top (sim)    : tb_up_down_counter
  RTL module   : up_down_counter
  Clock port   : clock
  Clock period : 5.00 ns  (200 MHz target)
  WNS threshold: -0.1 ns
  Max iters    : 5
  Timeout      : 300s
  Yosys        : enabled ✓
  OpenSTA      : disabled

  ITERATION 1 / 5

[22:48:04][SIM] Compiling up_down_counter.sv + up_down_counter_tb.sv...
[22:48:04][COMPILE] FAILED ✗
[22:48:04][YOSYS] Running pre-synthesis checks on up_down_counter.sv (top: up_down_counter)...
[22:48:04][YOSYS] Pre-synthesis: CLEAN ✓
[22:48:04][YOSYS] 7 warning(s):
    ⚠ /Users/bhanujakarumuru/Downloads/RTL_Automation_updated/test/up_down_counter.sv:16: Warning: Identifier '\reset' is implicitly declared.
    ⚠ /Users/bhanujakarumuru/Downloads/RTL_Automation_updated/test/up_down_counter.sv:18: Warning: Identifier '\enable' is implicitly declared.
    ⚠ /Users/bhanujakarumuru/Downloads/RTL_Automation_updated/test/up_down_counter.sv:16: Warning: Identifier '\reset' is implicitly declared.
    ⚠ /Users/bhanujakarumuru/Downloads/RTL_Automation_updated/test/up_down_counter.sv:18: Warning: Identifier '\enable' is implicitly declared.
    ⚠ Warning: Wire up_down_counter.\reset is used but has no driver.
    ⚠ Warning: Wire up_down_counter.\enable is used but has no driver.
[22:48:04][BACKUP] Saved originals → ai_fix_backups/iter_01/

  ■  PROMPT READY — PASTE INTO COGNIX

  ✓ Prompt copied to clipboard!

[22:48:04][WATCH] Watching for Cognix Apply... (timeout: 300s)
[22:48:04][WATCH] Watching: /Users/bhanujakarumuru/Downloads/RTL_Automation_updated/test/up_down_counter.sv
[22:48:04][WATCH] Will watch if created: /Users/bhanujakarumuru/Downloads/RTL_Automation_updated/up_down_counter.sv
[22:48:04][WATCH] Watching: /Users/bhanujakarumuru/Downloads/RTL_Automation_updated/test/up_down_counter_tb.sv
[22:48:04][WATCH] Will watch if created: /Users/bhanujakarumuru/Downloads/RTL_Automation_updated/up_down_counter_tb.sv

  →  Paste prompt in Cognix → Enter → click ✅ Apply

[22:49:16][WATCH] Change detected in: up_down_counter.sv ✓
[22:49:18][AUTO] Re-running simulation automatically...

  ITERATION 2 / 5

[22:49:18][SIM] Compiling up_down_counter.sv + up_down_counter_tb.sv...
[22:49:18][COMPILE] FAILED ✗
[22:49:18][YOSYS] Running pre-synthesis checks on up_down_counter.sv (top: up_down_counter)...
[22:49:18][YOSYS] Pre-synthesis: CLEAN ✓
[22:49:18][BACKUP] Saved originals → ai_fix_backups/iter_02/

  ■  PROMPT READY — PASTE INTO COGNIX
```

```
  ITERATION 3 / 5

[22:49:43][SIM] Compiling up_down_counter.sv + up_down_counter_tb.sv...
[22:49:47][COMPILE] OK ✓
[22:49:47][SIM] Running...
[22:49:47][SIM] TEST PASSED ✓
[22:49:47][YOSYS] Running pre-synthesis checks on up_down_counter.sv (top: up_down_counter)...
[22:49:47][YOSYS] Pre-synthesis: CLEAN ✓

  ✅  ALL CHECKS PASSED! SIM + SYNTHESIS + TIMING CLEAN.

  Fixed in 2 AI-assisted iteration(s) 🎉

  RUN LOG SAVED → run_logs/20260220_224804/

    → dump.vcd                    2,177 B
    → summary.json                581 B
    → up_down_counter.sv          1,294 B
    → up_down_counter_tb.sv       4,656 B
    → verilator_output.txt        521 B
    → yosys_output.txt            23,150 B

bhanujakarumuru@10-20-121-90 RTL_Automation_updated %
```

< 60s

Full Pipeline Run

Sim + Synth + Timing

5

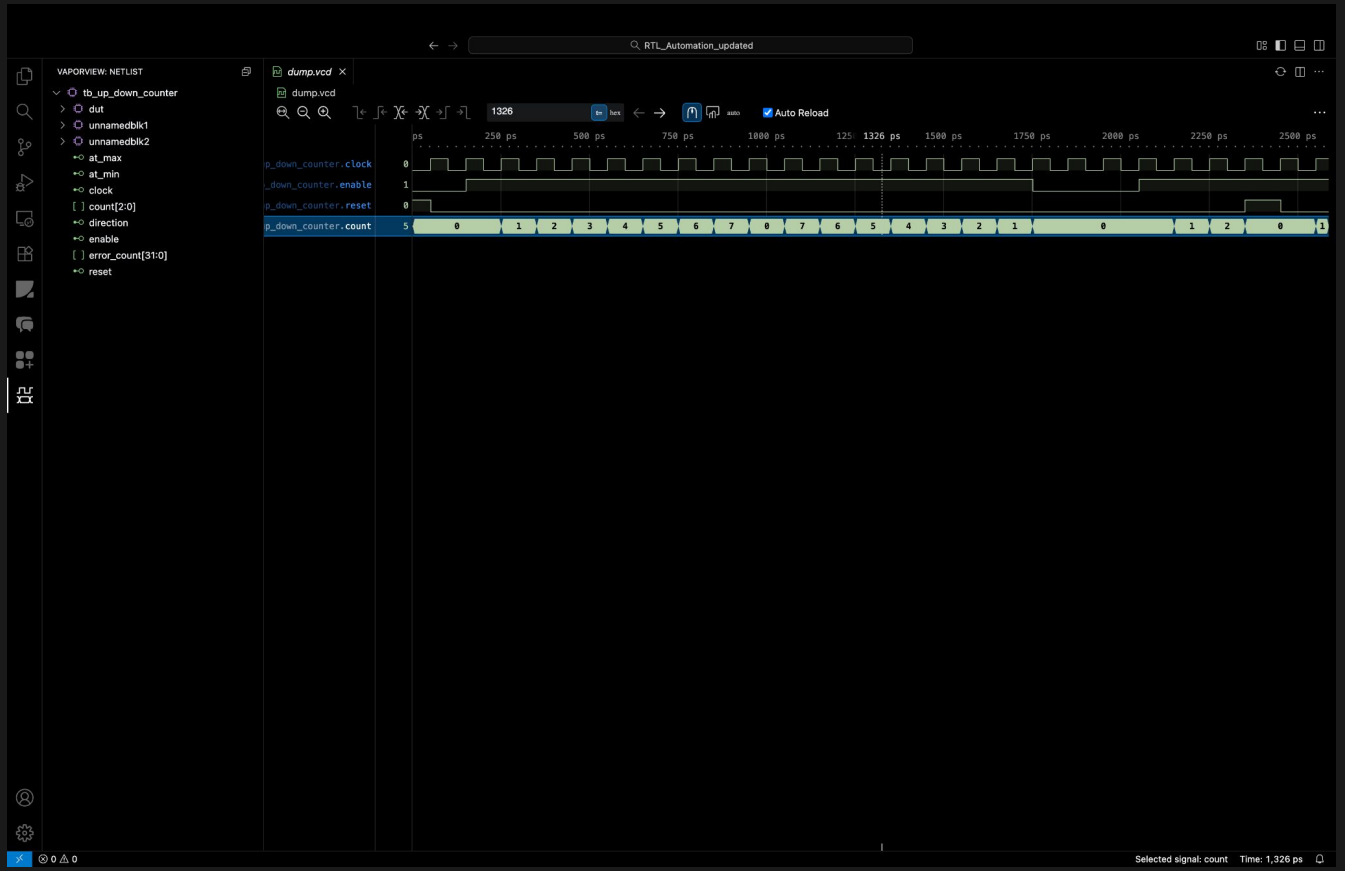
Max Fix Iterations

Configurable via --max

Demo & Results

WaveForm

WaveForm of “up_down_counter_tb”



Tech Stack

SIMULATION **Verilator**	`--binary --timing --trace | SystemVerilog | VCD output`	REQUIRED
PRE-SYNTHESIS **Yosys**	`read_verilog -sv | hierarchy -check | synth | check -assert`	Optional
TIMING STA **OpenSTA**	`NanGate45 Liberty (auto-dl) | WNS/TNS | Critical path report`	Optional
AI FIX ENGINE **Cognix (ACI)**	`Context-rich prompts | ACI Apply button | File-watch loop`	REQUIRED
LANGUAGE **Python 3.10+**	`subprocess | pathlib | watchdog | argparse | json logging`	REQUIRED
HDL **SystemVerilog**	`RTL + Testbench (.sv) | $dumpfile/$dumpvars | $finish`	REQUIRED

Future Scope

Phase 1

- If Cognix API is provided, the flow of RTL codes can be automated.
- Minimizes the necessity of manual labour which is tedious and significantly slower and less efficient.
- MCMM Timing Awareness - AI optimizes RTL across multiple PVT timing corners
- Design Space Exploration (DSE) - AI evaluates architectures for latency, area, throughput
- Explainable AI Debugging - Generates explanations for every RTL code fix
- Mixed-Signal Co-Simulation Support - Integrates Verilog-AMS/SystemC for SoC debugging
- CI/CD Hardware Integration - Automates RTL validation during design commits
- FPGA-in-the-Loop Validation - Tests AI-modified RTL on FPGA hardware

Phase 2

- Analysis tools shall be integrated into this to perform analysis for pre-layout quality checks.
- Thereby, partial layout quality improvement is done without even reaching layout phase.

Phase 3

- Cognix then becomes a smart assistant to layout engineers as they progress to generate netlist on tools such as Cadence Genus.
- Following that, Layout can be designed on tools such as Cadence Innovus.
- Further verification steps performed using tools such as Cadence Tempus, Pegasus and Quantus for full Signoff.

Phase 4

- Repeat the process on multiple designs.
- Run comparison tests proving how the addition of AI to semiconductor design is a massive boost to manufacturing thereby cutting the shortage globally.
- Subsequently figure out techniques to allow for layout engineers to verify and adopt possible designs using the help of AI using layout placement ideas.
- All these steps will require funding and support from Cognichip AI.

Thank You

AI-Driven Layout-Aware RTL Optimization Loop

Hackathon 2026