SentinelForgeAI: AI-Native Design Infrastructure for India’s Semiconductor Ecosystem

Abstract

SentinelForgeAI is an AI-powered platform designed to support India’s semiconductor ecosystem by enabling intelligent RTL code generation, verification, and policy simulation workflows. By combining large language models (LLMs), open-source EDA tools, and real-time infrastructure simulation, the platform aims to democratize chip design while enhancing security and verification standards. This paper presents the motivation, architecture, and early implementation of SentinelForgeAI as a foundational step toward India’s self-reliant silicon innovation roadmap.

Introduction

India’s growing ambitions in semiconductor manufacturing and electronics innovation have created an urgent need for accessible, intelligent design infrastructure. Despite government initiatives and increasing interest in local chip production, the Electronic Design Automation (EDA) ecosystem remains fragmented, costly, and heavily dependent on proprietary tools and expert-driven workflows. This presents a significant barrier for startups, universities, and early-stage designers to contribute to semiconductor R&D effectively.

At the same time, breakthroughs in large language models (LLMs) and open-source EDA tooling present a unique opportunity to reshape the RTL design and verification process. Inspired by state-of-the-art research in LLM-assisted EDA — such as AIVRIL and LLM4EDA — SentinelForgeAI proposes a cloud-native, AI-powered automation system for Verilog code generation, simulation, and verification.

The platform is designed to help democratize access to semiconductor design flows by reducing the complexity, tooling overhead, and human error typically associated with RTL workflows. Through multi-agent AI loops that can generate and refine RTL code from natural language prompts, SentinelForgeAI aims to build the foundational infrastructure that enables India's chip design sector to become self-reliant, agile, and innovation-ready.

Related Work

Recent advancements in the application of large language models (LLMs) to Electronic Design Automation (EDA) have laid the groundwork for intelligent, semi-autonomous chip design flows. Notable among these is **AIVRIL** (AI-Driven RTL Generation With Verification-in-the-Loop), which introduces a feedback-driven Verilog generation system using GPT-style models and syntax verification through open-source tools like Yosys. AIVRIL's core innovation lies in its iterative refinement of RTL code using error messages from synthesis tools to guide code corrections. This concept forms the technical backbone of SentinelForgeAI's code-review agent loop.

Another key development is **LLM4EDA**, a survey of the emerging ecosystem where LLMs assist in EDA flows ranging from synthesis to simulation and debugging. It outlines different architectural approaches — including single-agent assistants, chat-based debugging flows, and agent-based orchestration — many of which serve as inspiration for SentinelForgeAI's multi-agent approach. Unlike LLM4EDA's general exploration, SentinelForgeAI aims to deliver a focused SaaS platform that connects prompt-to-code-to-simulation in a minimal and usable loop.

While several proprietary tools (e.g., Synopsys DSO.ai, Cadence Cerebrus) explore AI-powered EDA optimization, their closed-source nature and high barrier to entry leave a wide gap for open, accessible, and India-first infrastructure. SentinelForgeAI occupies this space by marrying open-source EDA tools with LLMs in a looped, API-ready architecture — empowering early-stage teams to move fast with verifiable, AI-assisted hardware design.

System Architecture

SentinelForgeAI is built on a modular, multi-agent AI framework that replicates core RTL development workflows using language models and open-source EDA tools. The goal is to reduce the manual bottlenecks of Verilog generation, syntax debugging, simulation, and design refinement — especially for first-time or early-stage semiconductor engineers.

At its core, the platform is structured around three intelligent agents operating in a feedback loop:

---

1. Code Generation Agent (LLM)

- **Function**: Converts natural language prompts into Verilog RTL modules.

- **Model**: GPT-3.5 or 4-Turbo via OpenAI API (or open-source fallback like CodeLlama via Hugging Face).

- **Prompt Engineering**: Custom templates trained on HDL syntax expectations and design structure.

- **Output**: A candidate `top.v` Verilog file.

---

2. Simulation & Syntax Validation Agent (Yosys Wrapper)

- **Function**: Automatically runs synthesis/simulation via [Yosys](https://github.com/YosysHQ/yosys).

- **Input**: The Verilog file generated by the LLM.

- **Validation**:

  - Parses Yosys logs for syntax errors, warnings, and unsupported modules.

  - Classifies issues into fixable categories (missing ports, wrong module syntax, etc.)

---

3. Review & Refinement Agent (LLM-Based)

- **Function**: Reads error messages and logs from Yosys, and suggests fixes to the original code.

- **Loop Logic**:

  - Takes failed logs and contextually re-prompts the Code Agent to refine its output.

  - Can iterate multiple times until a “pass” is returned from simulation.

Additional Components

| Component | Description |

|---|---|

| **Frontend UI** | Web prompt input (future phase) |

| **Cloud Orchestration** | Netlify for frontend, GitHub Actions or local Flask server for inference |

| **API Key Vault** | OpenAI key management via `.env` |

| **Export Module** | Verified RTL saved to repo, optionally passed to Verilator for logic simulation (Phase 2) |

Diagram (Conceptual)

Optional Add-ons (Future Integration Layers)

- **Simulator Integration**: Add Verilator or Icarus to run functional testbenches.

- **Coverage Analyzer**: Analyze code/test coverage using Verilator flags.

- **Testbench Generator**: Use LLM to generate behavioral testbenches.

- **Waveform Viewer**: Future phase UI to show outputs in GTKWave-like format.

Implementation Details

The initial implementation of SentinelForgeAI is structured as a modular, open-source MVP using Python, shell scripts, and standard EDA tools. The system is broken into logical directories, each representing a key component of the Code–Simulate–Fix loop.

🔹 Directory Structure

sentinelforgeai/

├── agents/

| ├── code_agent.py # Generates Verilog from NL prompts via OpenAI API

| ├── review_agent.py # Reads Yosys errors and suggests prompt refinements

├── sim/

| ├── run_yosys.sh # Shell script to call Yosys and write output logs

| ├── testbench.v # Placeholder functional testbench

├── main.py # Runs end-to-end flow: Prompt → Generate → Simulate → Refine

├── requirements.txt # Python dependencies (OpenAI SDK)

├── docs/

| ├── research_draft.md # This document

| └── images/ # Architecture diagram and assets

🔹 Agent Workflows

1. **Code Agent** (`agents/code_agent.py`)

   - Uses OpenAI's `gpt-3.5-turbo` model to generate Verilog code from a prompt

   - Prompt includes structural scaffolding to ensure syntactical accuracy

   - Output written to `top.v`

2. **Yosys Simulation Agent** (`sim/run_yosys.sh`)
   - Runs `yosys` on `top.v` to check for syntax errors and unsupported constructs
   - Logs output to `yosys_log.txt`

3. **Review Agent** (`agents/review_agent.py`)
   - Parses the Yosys log for error messages
   - Regenerates refined prompts to correct the Verilog using feedback loop

4. **Main Loop** (`main.py`)
   - Controls the full sequence: prompt input → generation → synthesis → correction
   - Current version supports single-pass logic; multi-iteration loop under development

---

🔹 Environment & Dev Stack

Component	Tool	Notes
Language Model	OpenAI GPT-3.5-Turbo	Swappable with HuggingFace
EDA Tool	Yosys	Syntax and netlist checker
Simulation	Verilator/Icarus (Planned)	Functional verification
Language	Python 3.8+	Orchestrator + agents
Frontend	HTML/CSS (Netlify)	Web layer under design

🔹 Output Artifacts

- ✅ `top.v` – LLM-generated Verilog code
- ✅ `yosys_log.txt` – Synthesis results and errors
- ✅ `wave.vcd` – Simulation file (optional, future)
- ✅ Loop metrics – Iteration count, fixed tokens, logs (planned)

This implementation validates the feasibility of using LLMs to automate RTL flows with open-source tools. Future versions will expand simulation, add testbench generation, and introduce cloud-based verification scaling.

Future Scope

As SentinelForgeAI matures, several key extensions are planned to increase system capability, developer utility, and real-world design coverage. These fall into three primary verticals: simulation depth, automation intelligence, and hardware readiness.

---

1. ⚙️ Verilator & Behavioral Simulation

Objective: Add Verilator support to run actual testbenches and validate functionality.

- Use Verilator or Icarus Verilog to simulate generated RTL against sample stimuli
- Parse waveforms (VCD files) to verify signal correctness
- Enable waveform analysis (e.g., waveform diff with golden ref)

✅ **Why**: Moves from "compilation correct" to "functionally correct"

---

2. 🧪 Auto Testbench Generator

Objective: Extend LLM agent to generate `testbench.v` automatically.

- Based on module name, inputs, and output ports
- Supports simple assertions and finite stimulus
- Eventually integrates property-based verification (SystemVerilog assertions)

✅ **Why**: Speeds up validation and enables looped simulation

---

3. 📊 Coverage Analysis

Objective: Add functional coverage via Verilator's `--coverage` and branching metrics.

- Provide line/branch coverage reports
- Integrate AI agent that can optimize stimulus for uncovered paths

✅ **Why**: Make simulation efficient, not brute-force

---

4. ☁️ Cloud-Based Simulation Loop

Objective: Containerize the entire loop (Code Agent → Sim → Review) for scalable backend.

- Use Docker + Python FastAPI + background workers

- Expose `/simulate` and `/generate` APIs

- Optional CI pipeline using GitHub Actions or Netlify functions

✅ **Why**: Enables true SaaS integration and multi-user backend

---

5. 🔐 Policy-Aware Design Assistant (Zscaler vision)

Objective: Integrate with security policies and simulate network/infra design for chip-infra alignment.

- Add NAT, ACL, segmentation DSL inputs

- Output policy-aware hardware modules (for SmartNIC or firewall chips)

✅ **Why**: Ties back into earlier SentinelForgeAI policy-simulation vision

---

6. 🧠 Foundation Model Alignment

Objective: Fine-tune custom LLMs (e.g., CodeLlama or Mistral) on Verilog + HDL design principles.

- Host models locally or on HuggingFace

- Tune on open-source Verilog repos, testbenches, and RTL snippets

✅ **Why**: Reduce OpenAI dependency and tailor model accuracy

---

7. 🚀 FPGA + Hardware Bring-Up (Long-Term)

Objective: Enable export-to-board for simple designs

- Output Vivado/Quartus-ready code
- Upload to AWS F1 or TinyFPGA boards
- Enable India-first open source hardware pipelines

✅ **Why**: Close the loop from code → chip → board

Conclusion

SentinelForgeAI represents a pragmatic leap forward in democratizing semiconductor design and verification through intelligent automation. By blending large language models, open-source EDA tooling, and feedback-driven refinement loops, the platform offers an accessible on-ramp for startups, researchers, and students to contribute to India's silicon innovation journey.

What begins as a lightweight prototype — transforming natural language into verifiable RTL — lays the foundation for a scalable, cloud-native design assistant. As the platform evolves, integration with simulation engines, testbench generation, coverage analysis, and FPGA support will enable full-stack design-to-deploy workflows previously gated behind proprietary toolchains and high expertise barriers.

In a world where software has been democratized through frameworks and APIs, SentinelForgeAI aims to do the same for silicon — by turning chip design into a programmatic, iterative, and AI-assisted process. This work marks the beginning of a new paradigm: infrastructure-aware, security-conscious, India-first chip development, made possible by the convergence of open source, cloud computing, and large-scale generative intelligence.

References

[1] Zhao, Zihao et al. "AIVRIL: AI-Driven RTL Generation With Verification-in-the-Loop." arXiv preprint arXiv:2403.07687 (2024).

[2] Jiang, Chuan et al. "LLM4EDA: Large Language Models for Electronic Design Automation." arXiv preprint arXiv:2312.02237 (2023).

[3] Wu, Mengzhao et al. "OpenEDA: Towards Open-Source EDA Benchmarks for Verilog Code Generation." arXiv preprint arXiv:2404.05233 (2024).

[4] OpenAI, "OpenAI Python SDK", https://github.com/openai/openai-python

[5] YosysHQ, "Yosys Open SYnthesis Suite", https://github.com/YosysHQ/yosys

[6] Verilator, "Verilog Simulator", https://verilator.org

[7] HuggingFace, "Code Llama and Mistral Models", https://huggingface.co

Acknowledgements

This project would not have taken shape without the continuous support of the open-source hardware community, the authors of pioneering LLM4EDA research, and the creators of foundational tools like Yosys and Verilator.

Special thanks to:

- OpenAI for the foundational API access

- GitHub and Netlify for hosting early infrastructure

- [Dhiraj Jadhav](mailto:contact@sentinelforgeai.com) for driving the India-first semiconductor vision and execution of SentinelForgeAI

- AI-based guidance by OpenAI’s GPT, adapted into practical design with human oversight

Mentorship and moral encouragement were provided by peers and faculty from the ENTC-2016 batch and AI research ecosystem — this is just the beginning.

Show Diff

SentinelForgeAI: AI-Native Design Infrastructure for India’s Semiconductor Ecosystem

Abstract

SentinelForgeAI is an AI-powered platform designed to support India’s semiconductor ecosystem by enabling intelligent RTL code generation, verification, and policy simulation workflows. By combining large language models (LLMs), open-source EDA tools, and real-time infrastructure simulation, the platform aims to democratize chip design while enhancing security and verification standards. This paper presents the motivation, architecture, and early implementation of SentinelForgeAI as a foundational step toward India’s self-reliant silicon innovation roadmap.

Introduction

India’s growing ambitions in semiconductor manufacturing and electronics innovation have created an urgent need for accessible, intelligent design infrastructure. Despite government initiatives and increasing interest in local chip production, the Electronic Design Automation (EDA) ecosystem remains fragmented, costly, and heavily dependent on proprietary tools and expert-driven workflows. This presents a significant barrier for startups, universities, and early-stage designers to contribute to semiconductor R&D effectively.

At the same time, breakthroughs in large language models (LLMs) and open-source EDA tooling present a unique opportunity to reshape the RTL design and verification process. Inspired by state-of-the-art research in LLM-assisted EDA — such as AIVRIL and LLM4EDA — SentinelForgeAI proposes a cloud-native, AI-powered automation system for Verilog code generation, simulation, and verification.

The platform is designed to help democratize access to semiconductor design flows by reducing the complexity, tooling overhead, and human error typically associated with RTL workflows. Through multi-agent AI loops that can generate and refine RTL code from natural language prompts, SentinelForgeAI aims to build the foundational infrastructure that enables India's chip design sector to become self-reliant, agile, and innovation-ready.

Related Work

Recent advancements in the application of large language models (LLMs) to Electronic Design Automation (EDA) have laid the groundwork for intelligent, semi-autonomous chip design flows. Notable among these is AIVRIL (AI-Driven RTL Generation With Verification-in-the-Loop), which introduces a feedback-driven Verilog generation system using GPT-style models and syntax verification through open-source tools like Yosys. AIVRIL’s core innovation lies in its iterative

refinement of RTL code using error messages from synthesis tools to guide code corrections. This concept forms the technical backbone of SentinelForgeAI's code-review agent loop.

Another key development is LLM4EDA, a survey of the emerging ecosystem where LLMs assist in EDA flows ranging from synthesis to simulation and debugging. It outlines different architectural approaches — including single-agent assistants, chat-based debugging flows, and agent-based orchestration — many of which serve as inspiration for SentinelForgeAI's multi-agent approach. Unlike LLM4EDA's general exploration, SentinelForgeAI aims to deliver a focused SaaS platform that connects prompt-to-code-to-simulation in a minimal and usable loop.

While several proprietary tools (e.g., Synopsys DSO.ai, Cadence Cerebrus) explore AI-powered EDA optimization, their closed-source nature and high barrier to entry leave a wide gap for open, accessible, and India-first infrastructure. SentinelForgeAI occupies this space by marrying open-source EDA tools with LLMs in a looped, API-ready architecture — empowering early-stage teams to move fast with verifiable, AI-assisted hardware design.

System Architecture

SentinelForgeAI is built on a modular, multi-agent AI framework that replicates core RTL development workflows using language models and open-source EDA tools. The goal is to reduce the manual bottlenecks of Verilog generation, syntax debugging, simulation, and design refinement — especially for first-time or early-stage semiconductor engineers.

At its core, the platform is structured around three intelligent agents operating in a feedback loop:

1. Code Generation Agent (LLM)

Function: Converts natural language prompts into Verilog RTL modules.

Model: GPT-3.5 or 4-Turbo via OpenAI API (or open-source fallback like CodeLlama via Hugging Face).

Prompt Engineering: Custom templates trained on HDL syntax expectations and design structure.

Output: A candidate top.v Verilog file.

2. Simulation & Syntax Validation Agent (Yosys Wrapper)

Function: Automatically runs synthesis/simulation via Yosys.

Input: The Verilog file generated by the LLM.

Validation:

Parses Yosys logs for syntax errors, warnings, and unsupported modules.

Classifies issues into fixable categories (missing ports, wrong module syntax, etc.)

3. Review & Refinement Agent (LLM-Based)

Function: Reads error messages and logs from Yosys, and suggests fixes to the original code.

Loop Logic:

Takes failed logs and contextually re-prompts the Code Agent to refine its output.

Can iterate multiple times until a “pass” is returned from simulation.

Additional Components

Component Description

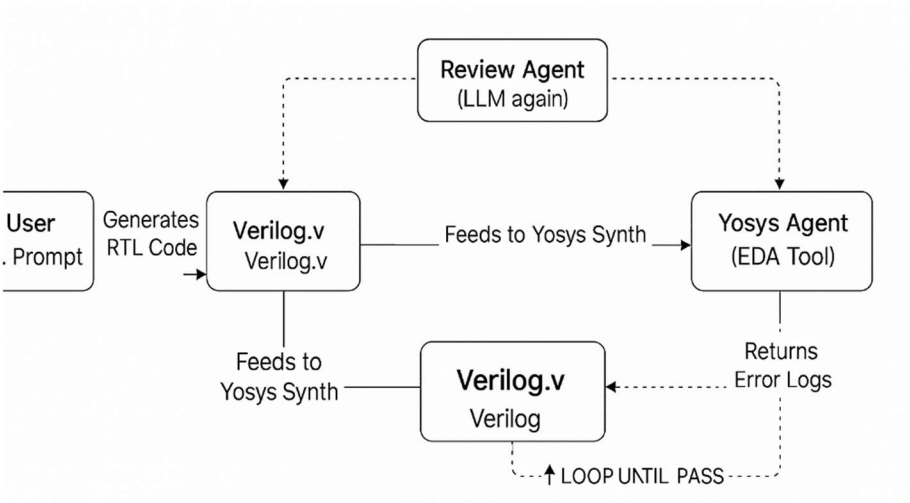
Frontend UI Web prompt input (future phase)

Cloud Orchestration Netlify for frontend, GitHub Actions or local Flask server for inference

API Key Vault OpenAI key management via .env

Export Module Verified RTL saved to repo, optionally passed to Verilator for logic simulation (Phase 2)

Diagram (Conceptual)



Optional Add-ons (Future Integration Layers)

Simulator Integration: Add Verilator or Icarus to run functional testbenches.

Coverage Analyzer: Analyze code/test coverage using Verilator flags.

Testbench Generator: Use LLM to generate behavioral testbenches.

Waveform Viewer: Future phase UI to show outputs in GTKWave-like format.

Implementation Details

The initial implementation of SentinelForgeAI is structured as a modular, open-source MVP using Python, shell scripts, and standard EDA tools. The system is broken into logical directories, each representing a key component of the Code–Simulate–Fix loop.

- Directory Structure

sentinelforgeai/ ├— agents/ | ├— code_agent.py # Generates Verilog from NL prompts via OpenAI API | ├— review_agent.py # Reads Yosys errors and suggests prompt refinements ├— sim/ | ├— run_yosys.sh # Shell script to call Yosys and write output logs | ├— testbench.v # Placeholder functional testbench ├— main.py # Runs end-to-end flow: Prompt → Generate → Simulate → Refine ├— requirements.txt # Python dependencies (OpenAI SDK) ├— docs/ | ├— research_draft.md # This document | └— images/ # Architecture diagram and assets

- Agent Workflows

Code Agent (agents/code_agent.py)

Uses OpenAI's gpt-3.5-turbo model to generate Verilog code from a prompt

Prompt includes structural scaffolding to ensure syntactical accuracy

Output written to top.v

Yosys Simulation Agent (sim/run_yosys.sh)

Runs yosys on top.v to check for syntax errors and unsupported constructs

Logs output to yosys_log.txt

Review Agent (agents/review_agent.py)

Parses the Yosys log for error messages

Regenerates refined prompts to correct the Verilog using feedback loop

Main Loop (main.py)

Controls the full sequence: prompt input → generation → synthesis → correction

Current version supports single-pass logic; multi-iteration loop under development

🔹 Environment & Dev Stack

Component	Tool	Notes
Language Model	OpenAI GPT-3.5-Turbo	Swappable with HuggingFace
EDA Tool	Yosys	Syntax and netlist checker
Simulation	Verilator/Icarus (Planned)	Functional verification
Language	Python 3.8+	Orchestrator + agents
Frontend	HTML/CSS (Netlify)	Web layer under design

🔹 Output Artifacts

✅ top.v – LLM-generated Verilog code

✅ yosys_log.txt – Synthesis results and errors

✅ wave.vcd – Simulation file (optional, future)

✅ Loop metrics – Iteration count, fixed tokens, logs (planned)

This implementation validates the feasibility of using LLMs to automate RTL flows with open-source tools. Future versions will expand simulation, add testbench generation, and introduce cloud-based verification scaling.

Future Scope

As SentinelForgeAI matures, several key extensions are planned to increase system capability, developer utility, and real-world design coverage. These fall into three primary verticals: simulation depth, automation intelligence, and hardware readiness.

1. ⚙️ Verilator & Behavioral Simulation

Objective: Add Verilator support to run actual testbenches and validate functionality.

Use Verilator or Icarus Verilog to simulate generated RTL against sample stimuli

Parse waveforms (VCD files) to verify signal correctness

Enable waveform analysis (e.g., waveform diff with golden ref)

✅ Why: Moves from "compilation correct" to "functionally correct"

2. 💉 Auto Testbench Generator

Objective: Extend LLM agent to generate testbench.v automatically.

Based on module name, inputs, and output ports

Supports simple assertions and finite stimulus

Eventually integrates property-based verification (SystemVerilog assertions)

✅ Why: Speeds up validation and enables looped simulation

3. 📊 Coverage Analysis

Objective: Add functional coverage via Verilator's --coverage and branching metrics.

Provide line/branch coverage reports

Integrate AI agent that can optimize stimulus for uncovered paths

✅ Why: Make simulation efficient, not brute-force

4. ☁️ Cloud-Based Simulation Loop

Objective: Containerize the entire loop (Code Agent → Sim → Review) for scalable backend.

Use Docker + Python FastAPI + background workers

Expose /simulate and /generate APIs

Optional CI pipeline using GitHub Actions or Netlify functions

✅ Why: Enables true SaaS integration and multi-user backend

5. 🔐 Policy-Aware Design Assistant (Zscaler vision)

Objective: Integrate with security policies and simulate network/infra design for chip-infra alignment.

Add NAT, ACL, segmentation DSL inputs

Output policy-aware hardware modules (for SmartNIC or firewall chips)

✅ Why: Ties back into earlier SentinelForgeAI policy-simulation vision

6. 🧠 Foundation Model Alignment

Objective: Fine-tune custom LLMs (e.g., CodeLlama or Mistral) on Verilog + HDL design principles.

Host models locally or on HuggingFace

Tune on open-source Verilog repos, testbenches, and RTL snippets

✅ Why: Reduce OpenAI dependency and tailor model accuracy

7. 🚀 FPGA + Hardware Bring-Up (Long-Term)

Objective: Enable export-to-board for simple designs

Output Vivado/Quartus-ready code

Upload to AWS F1 or TinyFPGA boards

Enable India-first open source hardware pipelines

✅ Why: Close the loop from code → chip → board

Conclusion

SentinelForgeAI represents a pragmatic leap forward in democratizing semiconductor design and verification through intelligent automation. By blending large language models, open-source EDA tooling, and feedback-driven refinement loops, the platform offers an accessible on-ramp for startups, researchers, and students to contribute to India's silicon innovation journey.

What begins as a lightweight prototype — transforming natural language into verifiable RTL — lays the foundation for a scalable, cloud-native design assistant. As the platform evolves, integration with simulation engines, testbench generation, coverage analysis, and FPGA support will enable full-stack design-to-deploy workflows previously gated behind proprietary toolchains and high expertise barriers.

In a world where software has been democratized through frameworks and APIs, SentinelForgeAI aims to do the same for silicon — by turning chip design into a programmatic, iterative, and AI-assisted process. This work marks the beginning of a new paradigm: infrastructure-aware, security-conscious, India-first chip development, made possible by the convergence of open source, cloud computing, and large-scale generative intelligence.

References

[1] Zhao, Zihao et al. "AIVRIL: AI-Driven RTL Generation With Verification-in-the-Loop." arXiv preprint arXiv:2403.07687 (2024).

[2] Jiang, Chuan et al. "LLM4EDA: Large Language Models for Electronic Design Automation." arXiv preprint arXiv:2312.02237 (2023).

[3] Wu, Mengzhao et al. "OpenEDA: Towards Open-Source EDA Benchmarks for Verilog Code Generation." arXiv preprint arXiv:2404.05233 (2024).

[4] OpenAI, "OpenAI Python SDK", https://github.com/openai/openai-python

[5] YosysHQ, "Yosys Open SYnthesis Suite", https://github.com/YosysHQ/yosys

[6] Verilator, "Verilog Simulator", https://verilator.org

[7] HuggingFace, "Code Llama and Mistral Models", https://huggingface.co

Acknowledgements

This project would not have taken shape without the continuous support of the open-source hardware community, the authors of pioneering LLM4EDA research, and the creators of foundational tools like Yosys and Verilator.

Special thanks to:

OpenAI for the foundational API access

GitHub and Netlify for hosting early infrastructure

Dhiraj Jadhav for driving the India-first semiconductor vision and execution of SentinelForgeAI

AI-based guidance by OpenAI's GPT, adapted into practical design with human oversight

Mentorship and moral encouragement were provided by peers and faculty from the ENTC-2016 batch and AI research ecosystem — this is just the beginning.