**ESIM DEBUGGER**

**GitHub link:** [**https://github.com/Snehartha/esim\_debugger**](https://github.com/Snehartha/esim_debugger)

**1. Tool’s Architecture and Workflow**

**Overview:**

This tool is designed to **analyse netlist files** and **parse error logs** to help identify design issues in circuit simulations. It provides **rule-based** as well as **AI-powered suggestions** for fixing problems related to:

* Incorrect or incomplete component connections.
* Floating nodes.
* Missing models or parameters.
* Unknown device types.

**i) Netlist Checker**

* Parses netlist files.
* Identifies:
  + Components with missing connections.
  + BJTs with incorrect pin count.
  + Nodes used only once (unconnected).
* Provides:
  + Rule-based suggestions.
  + Optional AI-generated suggestions via OpenAI API.

**ii) Error Log Parser**

* Analyses simulation output log files.
* Detects:
  + Unknown components.
  + Floating nodes.
  + Missing models.
  + Invalid parameters.
* Outputs error/warning messages with human-readable suggestions.

**Workflow:**

**Netlist Checker:**

1. **Input**: A .cir or .cir.out file .  
    change the file path in the main function. Create a .cir file in the folder then paste your netlist (which you want to parse) in the .cir file
2. **Parsing**:
   * Extract components and nodes.
   * Count how many times each node is used.
3. **Validation**:
   * Rule-based checks for issues.
   * (Optional) AI suggestions using GPT-3.5-Turbo.
4. **Output**:
   * Console logs of errors/warnings.
   * Fix suggestions.

**Error Parser:**

1. **Input**: A simulation log file (e.g., .log).
2. **Scanning**:
   * Reads each line.
   * Matches known error/warning patterns using regular expressions.
3. **Output**:
   * Specific error messages and corresponding fixes.

**2. AI/ML Utilization**

**Enabled via: --ai Flag**

When AI is enabled, the tool:

* Sends error messages or problem descriptions to **OpenAI's GPT-3.5-Turbo model**.
* Receives back **natural language suggestions** or solutions.

**AI Integration Details:**

* **API**: openai.chat.completions.create()
* **Model**: gpt-3.5-turbo
* **Usage Context**:
  + Suggests fixes for issues like "Node used only once" or "BJT has 2 pins instead of 3".
* **Fallback**: If API fails, tool gracefully handles the error with a custom message.

**3. User Guide**

**Installation**

**1) Prerequisites:**

* Python 3.7+
* OpenAI Python SDK
* .env file with your OpenAI API key:

OPENAI\_API\_KEY= your\_openai\_key\_here

for creating your open api key go to https://platform.openai.com/ then login with your account then go to the api section then create a new secrete key then paste in the .env file

**2) Dependencies:**

* Install required packages: pip install openai python-dotenv (This is used to get api key from .env file)

**Directory Structure**

esimdebugger/

|--.env

|--error\_parser.py

|--netlist\_parser.py

|--sample data

|--bjt\_amplifier.cir.out

|--sample\_log.log

|--package-lock.json

|--requirement.txt

|--rules

|--.vnev

**Usage Instructions**

**1)Run Netlist Checker**

* python netlist\_checker.py –ai  
  if you need ai suggestions and rule-based suggestions
* python netlist\_checker.py

if you need only rule-based suggestions

**Example Output:**

[✓] Netlist Loaded

[!] Warning: Node N003 is only used once.

Rule-Based Suggestion: Node N003 is only connected to one component...

AI Suggestion: Ensure N003 is connected to at least one other node...

**2) Run Error Log Parser**

python error\_parser.py

Modify file-path inside error\_parser.py , change the path to your .log file.

**Summary**

The above tool analyses netlist files and simulation logs to detect circuit issues. It provides both rule-based and optional AI-powered suggestions using OpenAI’s GPT-3.5 model. Errors like unconnected nodes, incorrect BJT connections, and floating nodes are identified. It runs via command line with customizable file input and AI support.