eSim Semester Long Internship Spring 2026

On

PCB Linter Plugin with Deterministic Annealing Clustering for KiCad using NGSpice

Submitted by

Prisha Bhatia

B.Tech CSE (AI and ML), 3rd Year

VIT Bhopal University

Under the guidance of

Prof. Prabhu Ramachandran

Principal Investigator

Department of Aerospace Engineering

Indian Institute of Technology Bombay

May 2026

Acknowledgment

I express my sincere gratitude to Prof. Prabhu Ramachandran for providing me with the opportunity to be part of the FOSSEE internship program. His leadership in promoting open-source engineering education has been an inspiration throughout this work.

I also acknowledge Prof. Kannan M. Moudgalya for his foundational role in establishing the FOSSEE initiative and for creating the academic platform through which this internship was made possible.

My sincere appreciation goes to my mentor, Mr. Sumanto Kar, for his continuous technical guidance, constructive feedback, and encouragement at every stage of the project. His insights were crucial in shaping both the plugin architecture and the theoretical approach.

I would also like to thank my internal mentors, Mr. Varad Patil and Ms. Shanthi Priya K, for their coordination, technical inputs, and timely reviews during the internship.

This work gave me an opportunity to study the Deterministic Annealing algorithm from a research paper, adapt it to the PCB domain, and implement it within a working KiCad plugin. The experience of reading academic literature and translating it into functional software has been deeply enriching.

I also thank the entire FOSSEE team for their support and coordination throughout the duration of this internship.

Prisha Bhatia
VIT Bhopal University
May 2026

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Electronic Design Automation (EDA) tools are essential to hardware engineering education. In the open-source academic ecosystem, KiCad is widely used for PCB layout and schematic capture, while eSim (developed by FOSSEE, IIT Bombay) provides SPICE-based simulation through NGSpice as the backend engine.

Despite being used together in academic settings, no automated tool existed to analyse the electrical characteristics of a completed KiCad PCB. After finishing a board design, a student or designer must manually inspect net voltages, identify power domains, and determine current paths. For complex PCBs with hundreds of nets, this process is tedious and error-prone.

The power network clustering problem in PCB design closely parallels the problem studied by Baranwal and Salapaka [1], who applied the Deterministic Annealing (DA) algorithm to identify coherent zones in electrical power grids. This internship adapts that approach to the PCB domain for the first time, extending it with current and power dissipation clustering as novel contributions.

1.2 Overview of eSim

eSim is a free and open-source EDA tool developed by FOSSEE, IIT Bombay [3], distributed under the GNU General Public Licence. The version used in this project, eSim 2.5, includes a SubcircuitLibrary containing over 647 subcircuit folders covering op-amps, voltage regulators, timers, and logic ICs. It also includes a deviceModelLibrary with model files for diodes, BJTs, MOSFETs, JFETs, and LEDs. These libraries were used in this project to enable automatic IC model injection into the generated SPICE netlist.

1.3 Objectives of the Project

The primary objectives of this project are listed below.

1. Develop a KiCad PCB Linter Plugin (v4.5) that automatically extracts nets and generates a SPICE netlist from a `.kicad_pcb` file.
2. Run NGSpice simulation and collect real node voltage and current data.
3. Apply Deterministic Annealing clustering to classify PCB nets into Voltage, Current, and Power Dissipation domains.
4. Integrate the eSim SubcircuitLibrary for automatic IC model injection.
5. Generate an interactive HTML report with quality score, attenuation matrices, PCB connectivity graph, and net-to-component map.
6. Support a Universal Import Mode (Mode 2) for clustering on any existing data file in DC, AC, Transient, or CSV format.

1.4 Novel Contributions

This project makes the following contributions that are not present in prior work.

- **DA Clustering on PCB Nets.** The algorithm from [1] is adapted from power grid analysis to PCB net analysis using $\alpha_{ij} = V_i/V_j$ as the similarity metric.
- **Current Clustering.** Per-net current is measured via the Rload technique and clustered using $\alpha_{ij} = I_i/I_j$.
- **Power Dissipation Clustering.** Per-node power $P_i = V_i \times I_i$ is computed and clustered using $\alpha_{ij} = P_i/P_j$.
- **eSim SubcircuitLibrary Integration.** The plugin automatically scans all 647 models and injects the correct `.include` directive.
- **Universal Parser.** DC, AC, Transient, and CSV formats are all supported for Mode 2 file import.

1.5 Methodology Overview

The project was executed in five phases. Phase 1 covered study of the KiCad API and the DA algorithm. Phase 2 involved implementing SPICE generation. Phase 3 covered the DA algorithm implementation and testing. Phase 4 dealt with SubcircuitLibrary integration. Phase 5 was testing on real KiCad demo boards and report writing.

Chapter 2

Literature Survey

2.1 SPICE and NGSpice

SPICE was developed at UC Berkeley in 1973 [7] and remains the global standard for analog circuit simulation. NGSpice [4] implements the full SPICE3 model set. Its batch mode (the `-b` flag) is used by this plugin for headless simulation, meaning NGSpice reads a netlist, executes the specified analysis, and writes results to files without any graphical interface.

2.2 KiCad Python Scripting API

KiCad version 8 exposes a Python module called `pcbnew` [5] that allows developers to register `ActionPlugin` subclasses and access footprints, pad nets, component values, and board geometry programmatically. This API is the entry point for the plugin and enables fully automatic net extraction from any `.kicad_pcb` file without user intervention.

2.3 Deterministic Annealing Algorithm

Rose (1998) [2] introduced Deterministic Annealing as a soft clustering approach derived from information theory and the rate-distortion framework. The key idea is to minimise the following Lagrangian over cluster assignment probabilities $p_{c|i}$ and cluster centres μ_c:

$$L = I(X;Z) + \beta \bar{D}(X,Z) \tag{2.1}$$

where $I(X;Z)$ is the mutual information between data and codewords, $\bar{D}$ is the expected distortion, and β is the annealing parameter. At low β, minimising L is

equivalent to minimising mutual information, which yields a convex problem with a unique solution. As β is increased, the problem gradually transitions to minimising distortion, and the soft assignments $p_{c|i}$ harden toward 0 or 1, recovering the original hard clustering problem. This avoids many poor local minima that afflict standard k-means.

The optimal association probabilities satisfy the Gibbs distribution:

$$p_{c|i} = \frac{q_c \exp(-\beta\, d_{ic})}{\sum_{c'} q_{c'} \exp(-\beta\, d_{ic'})} \tag{2.2}$$

where $q_c = \sum_i p_i p_{c|i}$ is the marginal probability of cluster c, p_i is the prior weight of node i, and d_{ic} is the distance from node i to the centre of cluster c. In each outer iteration, β is increased by a multiplicative factor $\gamma < 1$ (i.e. the temperature $T = 1/\beta$ is reduced), and the inner loop iterates until convergence of both $p_{c|i}$ and μ_c.

2.4 Clustering of Power Networks

Baranwal and Salapaka [1] applied the DA algorithm to the problem of identifying weakly coupled zones in electrical power transmission networks. The key insight in their work is the definition of the *attenuation matrix* $[\alpha_{ij}]$ as a measure of electrical similarity between buses:

$$\alpha_{ij} := \frac{\partial V_i / \partial Q_j}{\partial V_j / \partial Q_j} \tag{2.3}$$

This quantity measures the voltage fluctuation at bus i per unit voltage fluctuation at bus j, when a reactive power perturbation is applied at bus j. The normalisation by $\partial V_j / \partial Q_j$ ensures that the diagonal entries satisfy $\alpha_{ii} = 1$ for all i, giving equal importance to all nodes.

The electrical distance between two buses is then defined as:

$$d(i,j) = \|\alpha_i - \alpha_j\|_2^2 = \sum_{k=1}^{N} (\alpha_{ki} - \alpha_{kj})^2 \tag{2.4}$$

where α_i denotes the i-th column of the attenuation matrix. Two buses are considered electrically close when the influence of these buses over the entire network is similar. This formulation captures electrical coupling rather than only structural proximity.

2.5 Adaptation to PCB Net Clustering

In the original paper, the attenuation matrix is derived from reactive power sensitivity via load-flow computation. For PCB net analysis, real simulated node voltages from NGSpice are available directly. The attenuation matrix is therefore simplified to:

$$\alpha_{ij} = \frac{V_i}{V_j} \times \left(1 - 0.5 \cdot \frac{|V_i - V_j|}{V_{\max}}\right) \tag{2.5}$$

The ratio V_i/V_j captures voltage-level similarity, analogous to the original formulation. The difference penalty term reduces the similarity between nodes whose absolute voltages differ significantly, even when their ratio is moderate (for example, 5 V and 12 V have a ratio of 0.42, but their difference penalty pushes them further apart). This modification was introduced to address the specific case of adjacent power rails that share a moderate ratio but clearly belong to different domains.

For current clustering, $\alpha_{ij} = I_i/I_j$ directly. For power dissipation clustering, $P_i = V_i \times I_i$ and $\alpha_{ij} = P_i/P_j$.

The distance function in equation (2.4) and the Gibbs update in equation (2.2) are applied unchanged from the original algorithm.

2.6 NetworkX Graph Analysis

NetworkX [6] is used to compute degree centrality and betweenness centrality on the PCB component connectivity graph. Degree centrality identifies components with unusually high fanout (connected to many nets), while betweenness centrality identifies components that act as single points of failure in the connectivity graph. PyVis renders the resulting graph as an interactive HTML visualisation embedded in the report.

2.7 Existing Tools and Limitations

KiCad's built-in DRC focuses on physical layout rules such as clearance and trace width, with no electrical domain analysis. eSim's simulation converter requires manual schematic entry and does not operate on completed PCB files. No existing open-source tool performs voltage domain clustering on a `.kicad_pcb` file directly. This plugin fills that gap.

Chapter 3

Problem Statement

3.1 Problem Statement

A PCB designer finishing a board faces the following challenges.

1. **Manual net analysis.** With 100 or more nets, manually identifying voltage levels and power domains is impractical and error-prone.
2. **No voltage domain identification.** No open-source tool classifies nets into domains such as GND, 5V, 12V, and signal from real simulation data.
3. **No current-based guidance.** Trace width selection requires per-net current, which is unavailable without simulation.
4. **Manual IC model resolution.** SPICE models for ICs must be manually located and injected into netlists, which is time consuming.
5. **No power dissipation analysis.** PCB hot-spot identification requires per-node power computation, which no existing open-source plugin provides.

3.2 Approach

The proposed solution is a KiCad Python plugin that performs the following steps automatically.

- Automatic net extraction and SPICE generation from the KiCad PCB file using the `pcbnew` API.
- Smart voltage source assignment based on net naming conventions, with a user confirmation dialog on first run.
- NGSpice simulation and per-net current measurement using the Rload technique.
- DA clustering on real simulation data for voltage, current, and power dissipation domains.

- eSim SubcircuitLibrary auto-scan with `.include` injection.
- Dual-mode operation: Mode 1 for KiCad PCB and Mode 2 for data import.
- Interactive HTML report with quality score, attenuation matrices, connectivity graph, and net map.

The plugin is non-intrusive. It does not modify any KiCad project files or PCB layouts and operates purely as an analysis and reporting tool.

Chapter 4

Implementation

4.1 Plugin Architecture and Workflow

The plugin is implemented as a single Python module (`pcb_linter_main.py`, v4.5). Figure 4.1 shows the Mode 1 workflow and Figure 4.2 shows the Mode 2 workflow.

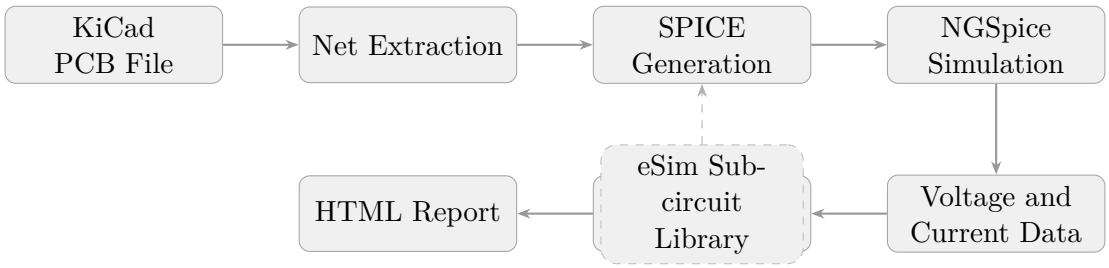


Figure 4.1: Mode 1: KiCad PCB plugin workflow

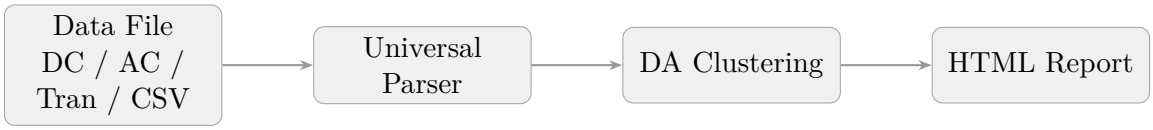


Figure 4.2: Mode 2: Universal import workflow

The mode selection dialog shown in Figure 4.3 is presented to the user when the plugin is invoked from the KiCad scripting console.

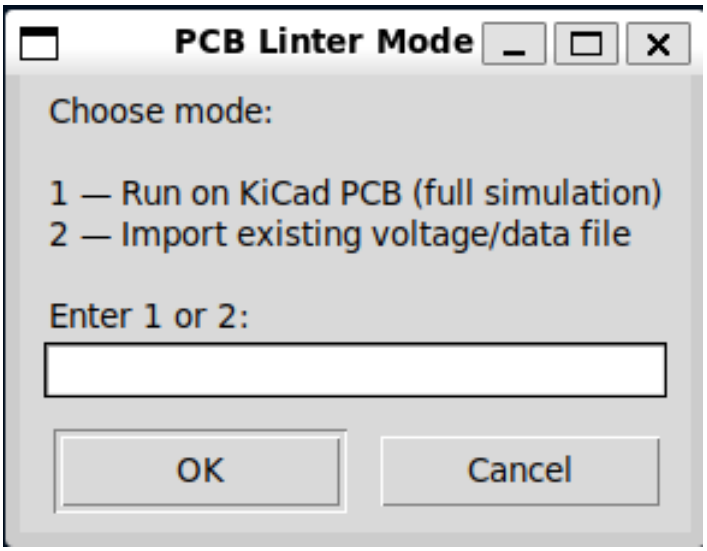


Figure 4.3: Plugin mode selection dialog presented at startup

Table 4.1 lists the key functions in the plugin and their roles.

Table 4.1: Key functions in the PCB Linter Plugin

Function	Role
`run()`	Main entry point; orchestrates the full pipeline
`_generate_spice()`	Builds SPICE netlist from KiCad footprints
`_infer_net_voltage()`	Infers supply voltage from net name
`_clean_spice_value()`	Normalises KiCad values for NGSpice
`_build_alpha()`	Constructs the similarity matrix
`_da_clustering()`	Runs the Deterministic Annealing algorithm
`_make_report()`	Renders the interactive HTML report
`_detect_floating_nets()`	Identifies unconnected signal nets
`_get_power_rail_config()`	Dialog for user voltage confirmation
`_compute_cluster_quality()`	Computes intra and inter cluster quality

4.2 DA Algorithm Implementation

4.2.1 Alpha Matrix

The similarity matrix is computed by `_build_alpha()` using equation (2.5). Two special cases are handled explicitly. When both nodes are at zero volts (GND), the similarity is set to 1.0 since they belong to the same domain. When the denominator node is at zero volts but the numerator is not, the ratio is capped at $R_{\max} = 10.0$ to prevent an unbounded value.

4.2.2 Cluster Count Selection

GND nodes (voltage below 0.01 V) are identified first and will form their own cluster. The number of clusters is then set to:

$$k = \min(5,\ |\mathcal{V}_{\text{unique}}| + \mathbf{1}[\text{GND nodes exist}]) \tag{4.1}$$

where $\mathcal{V}_{\text{unique}}$ is the set of distinct rounded node voltages among the non-GND nodes. Pre-grouping by rounded voltage is applied before DA, so that each distinct voltage level is guaranteed its own cluster regardless of the annealing outcome.

4.2.3 Algorithm Flow

Figure 4.4 shows the execution flow of the DA algorithm as implemented in `_da_clustering()`.

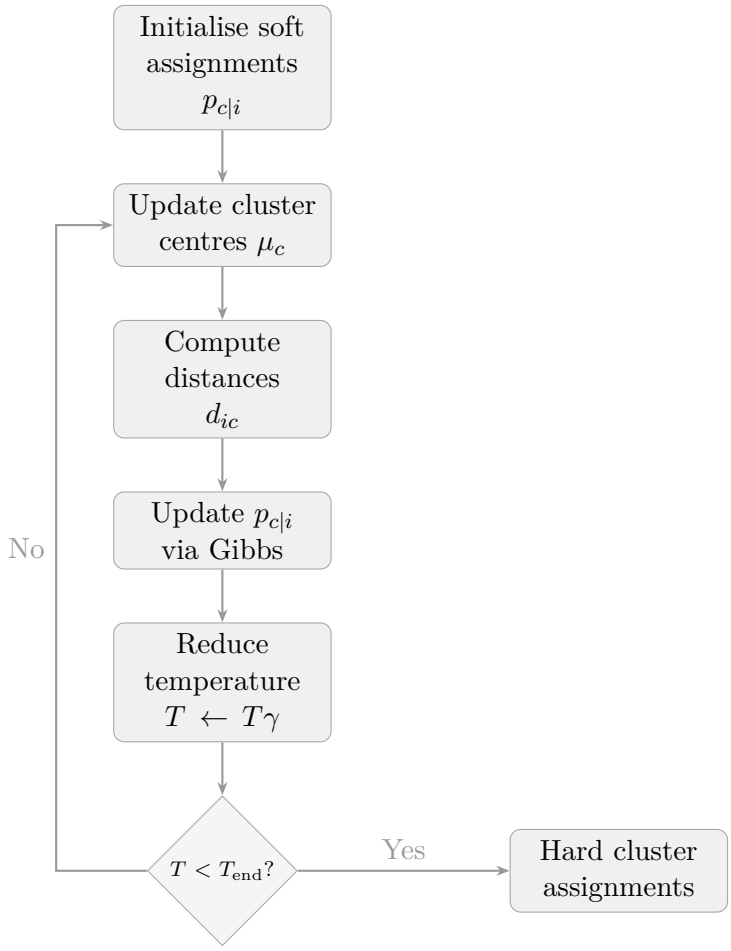


Figure 4.4: DA algorithm execution flow in `_da_clustering()`

The cooling schedule uses $\gamma = 0.92$, $T_{\text{start}} = 2.0$, and $T_{\text{end}} = 10^{-4}$, which gives approximately 119 iterations to convergence.

4.3 SPICE Generation Engine

The `_generate_spice()` function translates each KiCad footprint into a SPICE element. Table 4.2 shows the component prefix to SPICE element mapping.

Table 4.2: Component prefix to SPICE element mapping

Prefix	Component Type	Example SPICE Line
R	Resistor	`R1 n1 n2 10k`
C	Capacitor	`C1 n1 n2 100u`
L	Inductor	`L1 n1 n2 1m`
V	Voltage source	`V1 n1 0 DC 5`
D	Diode or LED	`D1 anode cathode Dmodel`
Q	BJT transistor	`Q1 c b e Qnpn`
M	MOSFET	`M1 d g s s Mnmos W=10u L=1u`
J	JFET	`J1 d g s Jmodel`
U	IC or subcircuit	`XU1 ... LM741`

KiCad net names often contain characters that are invalid in SPICE. The plus sign is replaced with the letter P, the minus sign with N, and the forward slash with an underscore. Component values are cleaned by the `_clean_spice_value()` function, which handles cases such as `22uF/25V` becoming `22u`, `100µF` becoming `100u`, and `5,1K` becoming `5.1k`.

4.4 Smart Voltage Source Assignment

Rather than placing a single arbitrary voltage source, the `_infer_net_voltage()` function examines each net name and assigns the most appropriate DC voltage. Table 4.3 shows the mapping for common patterns. A power rail confirmation dialog (Feature 1) is presented on the first run, and the user-confirmed values are saved to `pcb_voltages.json` for automatic reuse on subsequent runs.

Table 4.3: Net name patterns and inferred supply voltages

Net Name Pattern	**Inferred Voltage**
VCC, VDD, 5V, VBUS, VSYS	5.0 V
3V3, V3V3, AVDD, P3V3, VDDA	3.3 V
VIN, V12, VMOT, +12V	12.0 V
VPP, VPROG, VBOOST	13.0 V
1V8, VDDIO, DVDD	1.8 V
VBAT, LIPO	3.7 V
VEE, VNEG, −5V, −12V	Negative value
GND, AGND, DGND, VSS	0.0 V (reference)
Signal nets (no match)	100 kΩ load resistor

4.5 eSim SubcircuitLibrary Integration

Figure 4.5 shows the automatic subcircuit injection flow.

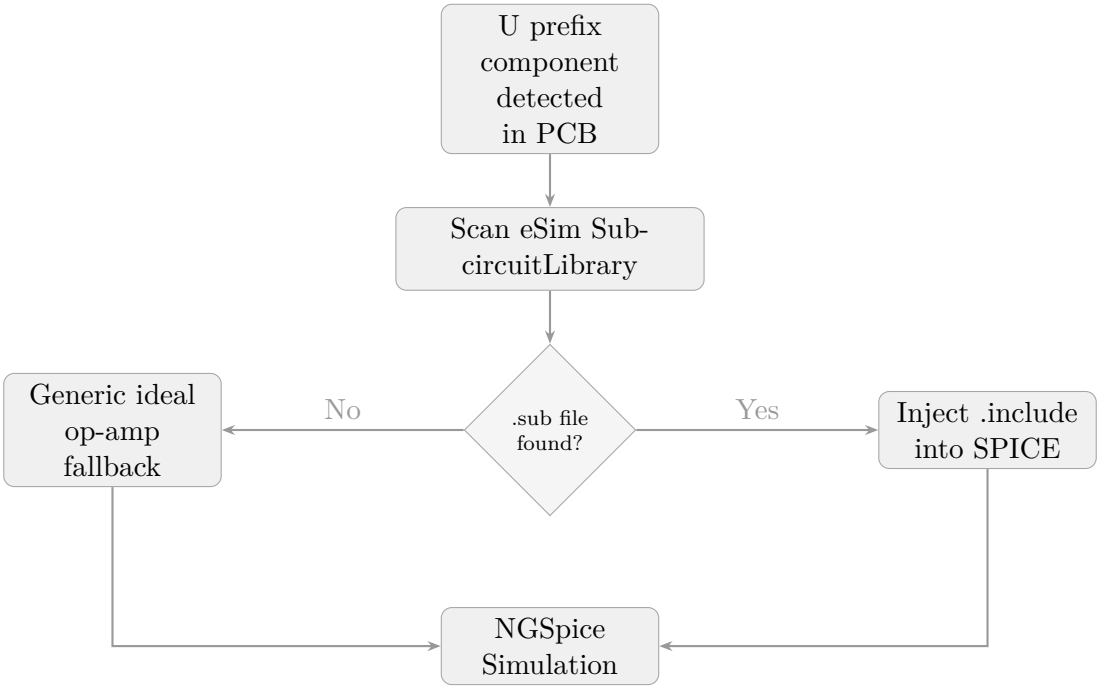


Figure 4.5: eSim SubcircuitLibrary injection flow

Table 4.4 lists the IC models supported through the eSim library.

Table 4.4: Supported eSim SubcircuitLibrary models

Category	Models
Op-Amps	LM741, LM358, LM393, LM386, LM833, LM124, LM1458
Voltage Regulators	LM7805, LM7812, LM7809, LM7905, LM7912, LM317
Timers	NE555, LM555
Comparators	LM311, LM301, LM302
Fallback	Generic ideal op-amp subcircuit

4.6 Cluster Quality Metric

A cluster quality score is computed using the ratio of inter-cluster separation to intra-cluster spread:

$$\text{Quality Ratio} = \frac{\overline{d_{\text{inter}}}}{\overline{d_{\text{intra}}} + 0.01} \tag{4.2}$$

where $\overline{d_{\text{inter}}}$ is the mean pairwise distance between cluster centroids and $\overline{d_{\text{intra}}}$ is the mean voltage spread within clusters. Table 4.5 shows the label thresholds.

Table 4.5: Cluster quality metric labels and thresholds

Quality Ratio	Label	Score
Greater than 10	Excellent	95
Greater than 5	Good	80
Greater than 2	Fair	60
2 or below	Poor	35

4.7 HTML Report Features

The generated HTML report is a self-contained file that works in any browser without an internet connection. Table 4.6 lists all sections included in the report.

Table 4.6: HTML report sections and their descriptions

Section	**Description**
Header and mode badge	Report type and data format indicator
Quality score bar	Score 0 to 100 with PASS / REVIEW / FAIL
Status block	Simulation result and data source
PCB Connectivity Graph	Interactive PyVis graph (Mode 1 only)
DA Clustering tabs	Voltage, Current, Power Dissipation
Voltage Attenuation Matrix	Colour coded $\alpha_{ij} = V_i/V_j$
Current Attenuation Matrix	Collapsible
Power Dissipation Matrix	Collapsible
Net to Component Map	All nets with their connected components
Issues and Recommendations	ERROR, WARNING, INFO with fix suggestions
Cluster Voltage Distribution	Bar chart per cluster
Simulation Coverage	Fully simulated and unmodeled component lists
Dark and Light Mode Toggle	Persistent across browser sessions

Chapter 5

Test Circuits and Results

5.1 Test Circuit 1: PIC Programmer PCB

5.1.1 Circuit Description

The PIC Programmer V03 is the KiCad demo PCB included with the KiCad installation. It implements a PIC microcontroller programmer with a USB interface, a 7805 voltage regulator for the 5 V supply, a VPP boost converter generating approximately 13 V for the MCLR programming pin, BC337 NPN transistors for VPP switching, and PIC sockets for 8, 18, 28, and 40-pin devices. This circuit is an ideal test case because it contains multiple voltage domains (0 V, 5 V, and 13 V) along with intermediate signal voltages at transistor bases and diode nodes.

Table 5.1 summarises the PCB statistics as reported by the plugin.

Table 5.1: PIC Programmer PCB statistics

Parameter	**Value**
Total Nets	111
Power Nets	5
Signal Nets	29
Unconnected	77
Fully Simulated	48
Unmodeled	15
Quality Score	100 / PASS

Figure 5.1 shows the KiCad PCB layout of the PIC Programmer.

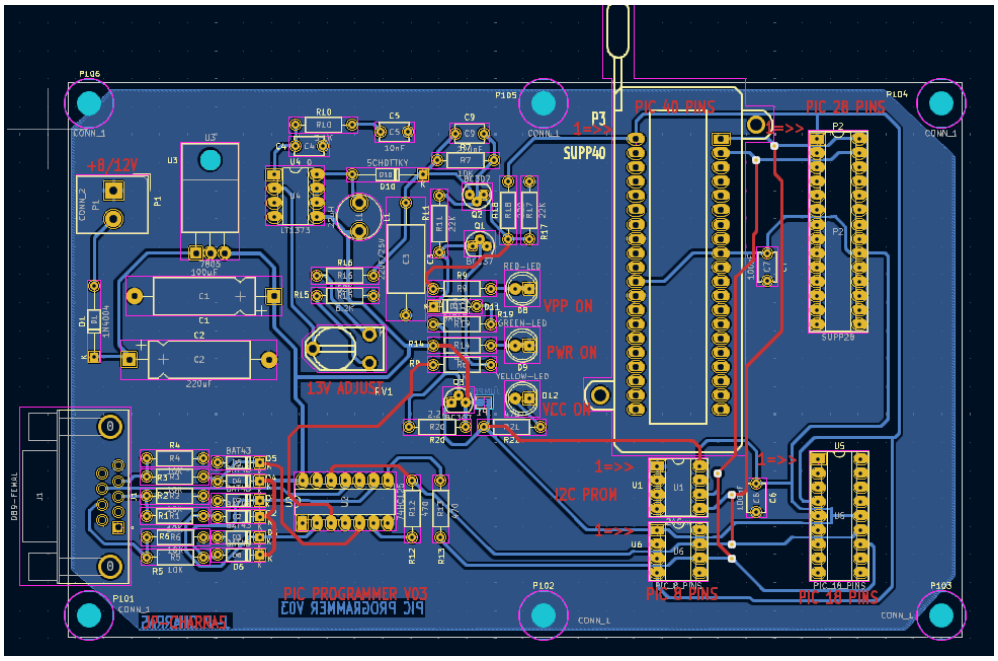


Figure 5.1: KiCad PCB layout of the PIC Programmer V03

5.1.2 Power Rail Confirmation

When the plugin is run on the PIC Programmer for the first time, it detects the power rail net names and presents the confirmation dialog shown in Figure 5.2. The user can confirm or modify the inferred voltages. The confirmed values are saved to `pcb_voltages.json` and loaded automatically on subsequent runs.

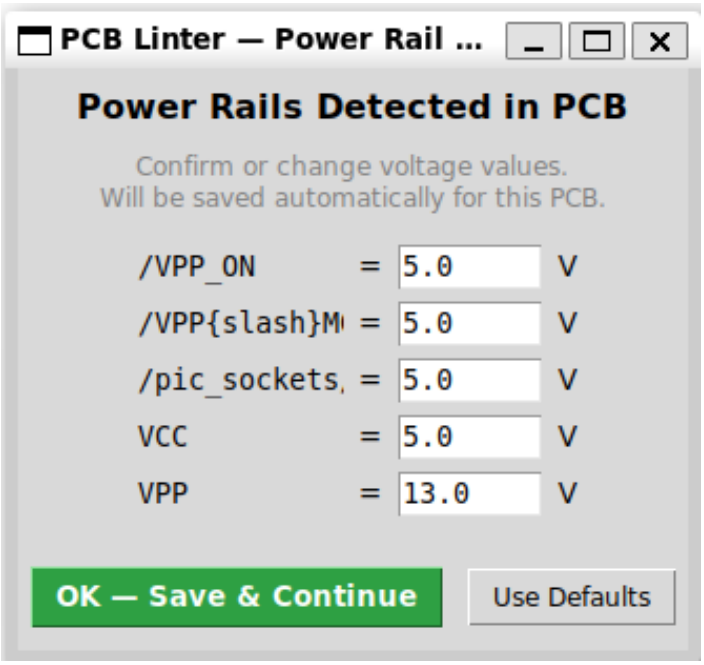


Figure 5.2: Power Rail Dialog: VCC = 5 V and VPP = 13 V detected for the PIC Programmer

5.1.3 Quality Score and Simulation Status

Figure 5.3 shows the report header for the PIC Programmer. The plugin assigned 5 voltage source nodes correctly and simulated 34 non-GND nets. The quality score of 100 indicates that no high-severity issues were found in the connectivity graph.

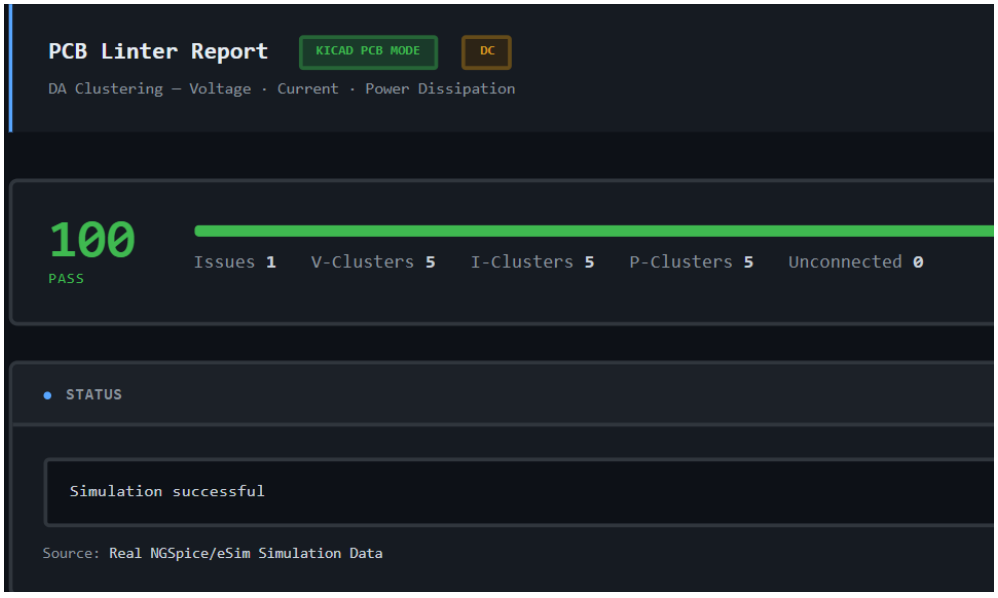


Figure 5.3: PIC Programmer: Quality Score 100/PASS with 5 voltage clusters, 5 current clusters, and 5 power clusters

5.1.4 PCB Connectivity Graph

Figure 5.4 shows the interactive connectivity graph generated by NetworkX and PyVis. Gold diamond nodes represent components connected to power rails. Blue circle nodes represent signal components. A red star node would indicate high fanout, but none was detected in this circuit.

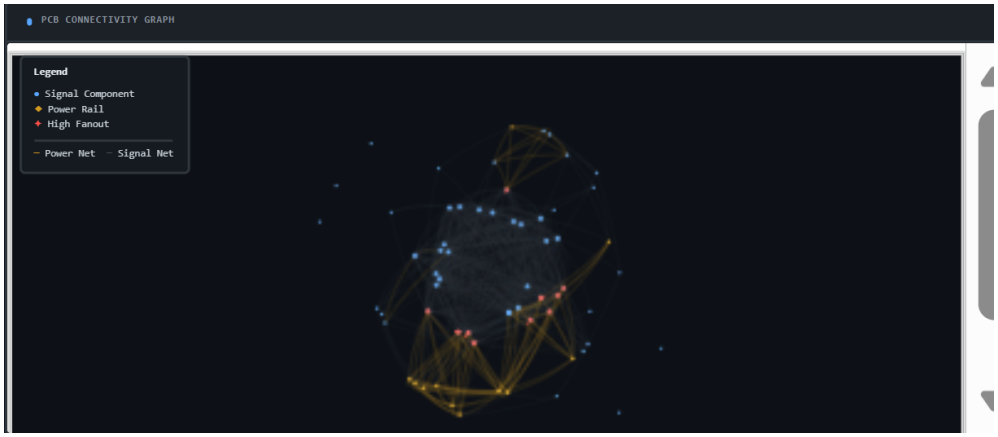


Figure 5.4: PIC Programmer PCB Connectivity Graph showing component-level connectivity

5.1.5 Voltage Clustering Results

Figure 5.5 shows the voltage clustering output. The DA algorithm correctly identified five distinct power domains corresponding to the physical voltage levels present in the PIC Programmer circuit.

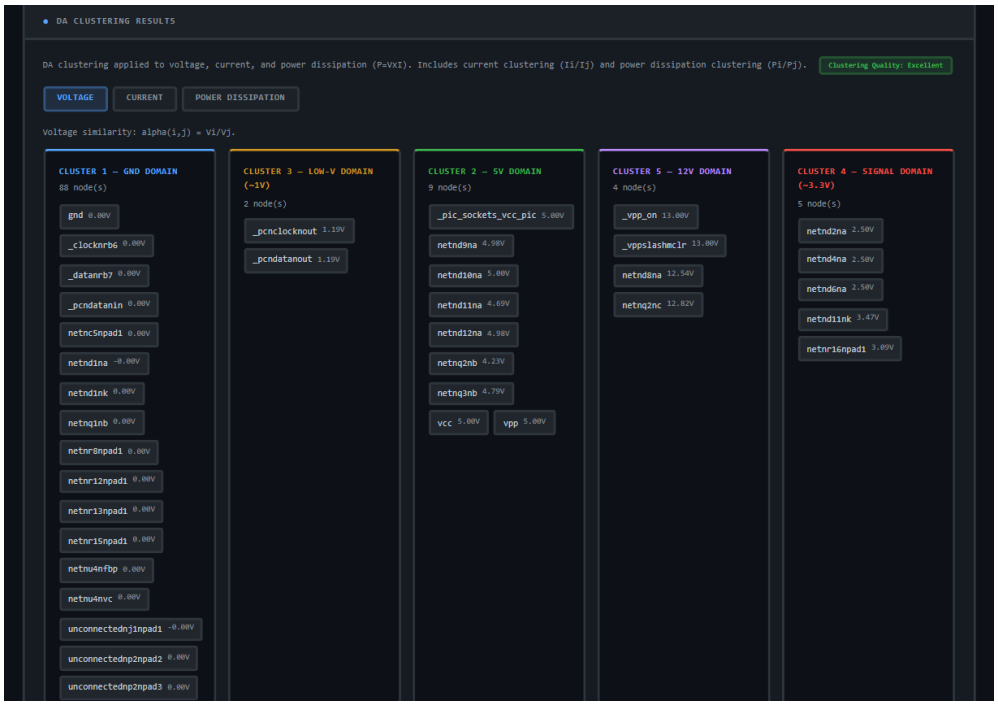


Figure 5.5: PIC Programmer: Voltage clustering showing 5 domains (GND, Low-V signal, 5 V, 12 V, and VPP 13 V)

Table 5.2 describes the five clusters and their physical interpretation.

Table 5.2: Voltage clustering results for the PIC Programmer circuit

Cluster	Domain	Voltage Range	Representative Nets
1	GND Domain	0 V	gnd, all signal nets at 0 V
2	PC Signal	1.0 to 2.0 V	_pc_clock_out, _pc_data_out
3	Diode nodes	2.0 to 3.0 V	net_d2_a, net_d4_a, net_d6_a
4	5 V Domain	4.7 to 5.3 V	vcc, _pic_sockets_vcc_pic
5	VPP Domain	10 to 13 V	vpp, net_q2_b

5.1.6 Current Clustering Results

Figure 5.6 shows the current clustering output. This is a novel contribution not present in prior work. The per-net currents are obtained by placing a 100 kΩ load resistor at each signal node and measuring the branch current through each resistor using NGSpice's `print @rload_N[i]` command. The rload-to-net mapping is stored in `rload_node_map.json` to allow meaningful cluster labelling.

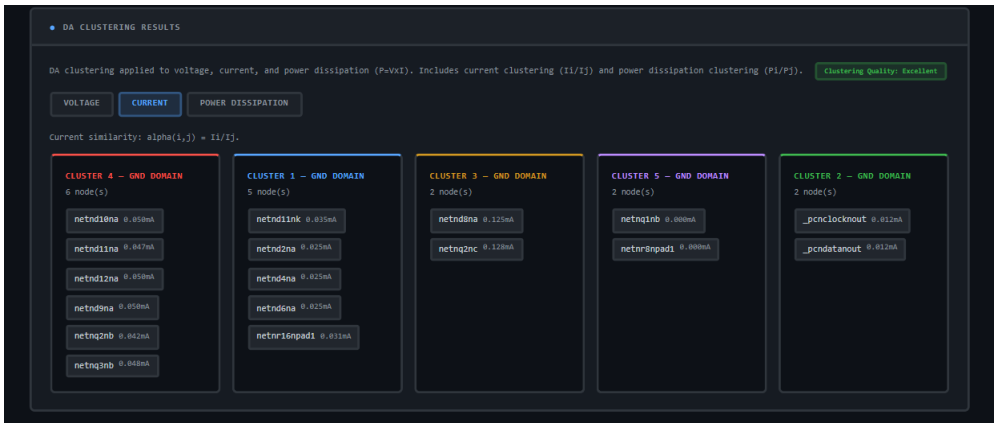


Figure 5.6: PIC Programmer: Current clustering showing 5 current domains based on $\alpha_{ij} = I_i/I_j$

5.1.7 Power Dissipation Clustering Results

Figure 5.7 shows the power dissipation clustering output. Per-node power is computed as $P_i = V_i \times I_i$ at each node where both voltage and current data are available. High-power nodes such as the VPP transistor paths are correctly separated from low-power signal nodes.

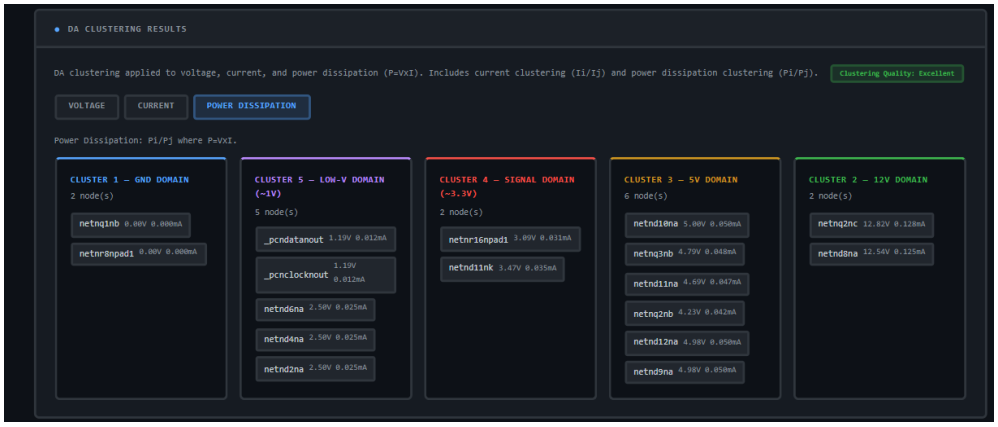


Figure 5.7: PIC Programmer: Power dissipation clustering based on $P_i = V_i \times I_i$

5.1.8 Voltage Attenuation Matrix

Figure 5.8 shows the voltage attenuation matrix $\alpha_{ij} = V_i/V_j$ for all simulated nodes. Green cells indicate high similarity ($\alpha > 0.8$), yellow cells indicate moderate similarity ($0.4 < \alpha < 0.8$), and dark cells indicate nodes in different power domains ($\alpha < 0.4$). The block structure visible in the matrix confirms that the DA algorithm has identified the correct cluster groupings.

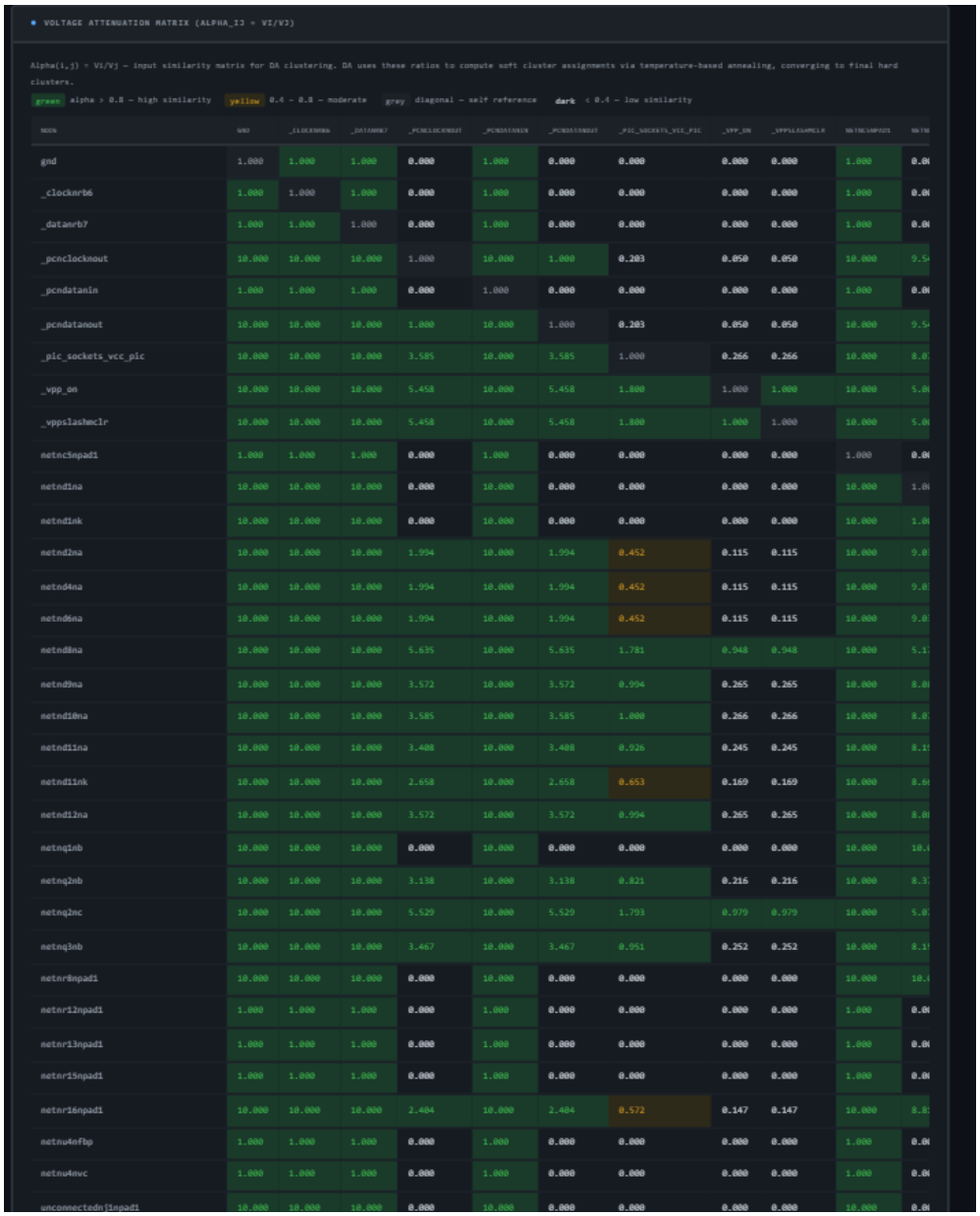


Figure 5.8: PIC Programmer: Voltage Attenuation Matrix $\alpha_{ij} = V_i/V_j$

5.1.9 Cluster Voltage Distribution

Figure 5.9 shows the bar chart section of the HTML report, which visualises the centroid voltage and voltage range of each cluster. The chart confirms that the five clusters span the full voltage range of the circuit from 0 V to 13 V.

Figure 5.9: PIC Programmer: Cluster Voltage Distribution bar chart

5.1.10 Issues and Net Map

Figure 5.10 shows the Issues and Recommendations section. The 77 unconnected pins on connectors and PIC sockets are correctly classified as INFO (intentional by design) and do not affect the quality score. Figure 5.11 shows the Net to Component Map, which lists all 111 nets and their connected components.

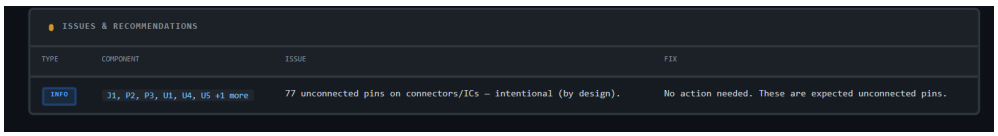


Figure 5.10: PIC Programmer: Issues and Recommendations table showing INFO badges for unconnected connector pins

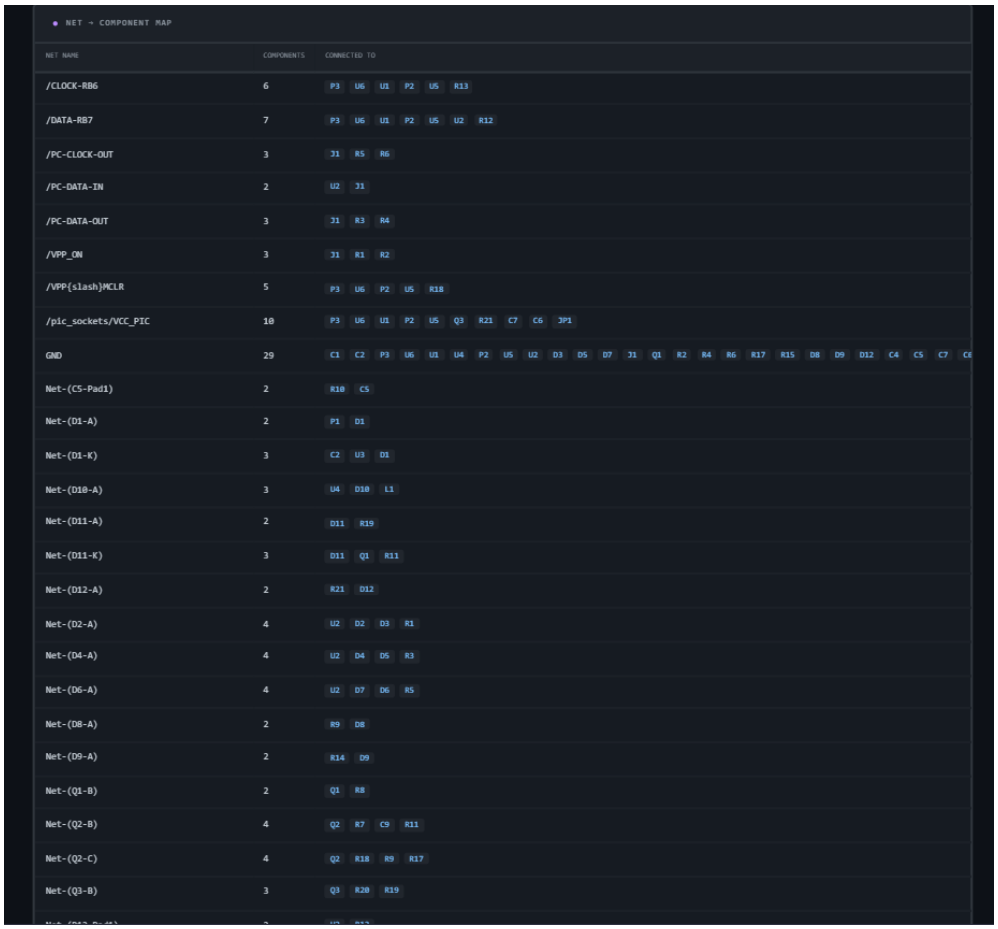


Figure 5.11: PIC Programmer: Net to Component Map for all 111 nets

5.1.11 Simulation Coverage and Dark Mode

Figure 5.12 shows the Simulation Coverage section, which lists which components were fully simulated and which were excluded due to missing models. Figure 5.13 shows the dark and light mode toggle functionality.

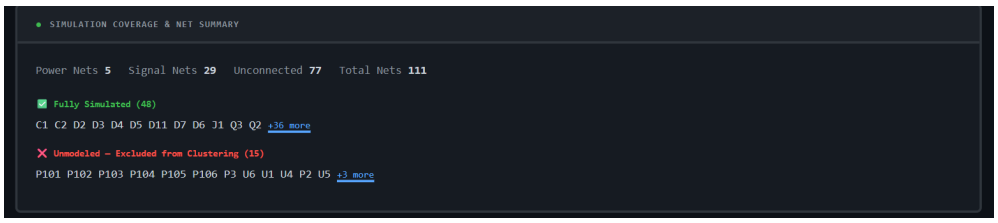


Figure 5.12: PIC Programmer: Simulation Coverage showing 48 fully simulated and 15 unmodeled components

(a) Toggle button (b) Report in light mode

Figure 5.13: Dark and light mode toggle, persistent across browser sessions

5.2 Test Circuit 2: Stickhub USB Hub PCB

5.2.1 Circuit Description

The Stickhub is a compact USB hub PCB included in the KiCad demo library. It contains a USB 2.0 hub controller IC (U1) with 7 downstream USB ports, diode protection, decoupling capacitors, status LEDs, and a crystal oscillator. The circuit has three supply rails: `+5V` (USB VBUS), `+1V8` (internal logic supply), and `VIN` (12 V input). The net names `+5V` and `+1V8` contain a plus sign, which tests the net name cleaning logic of the plugin.

Table 5.3 summarises the PCB statistics.

Table 5.3: Stickhub PCB statistics

Parameter	**Value**
Total Nets	47
Power Nets	5
Signal Nets	40
Unconnected	2
Fully Simulated	90
Unmodeled	4
Quality Score	85 / PASS

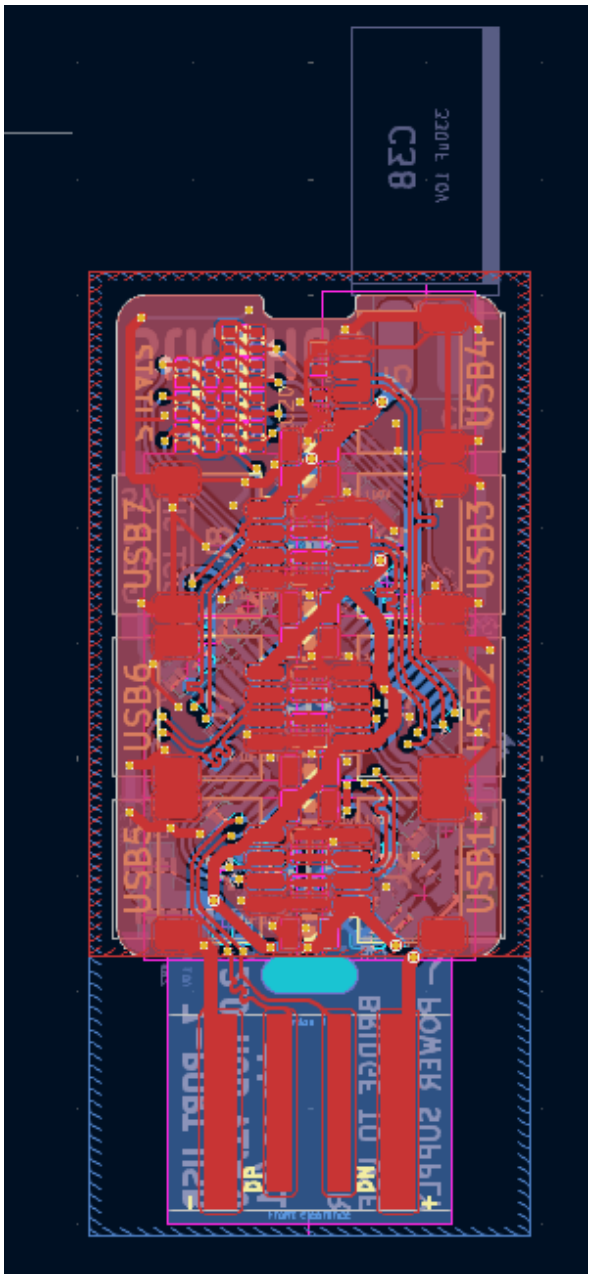


Figure 5.14: KiCad PCB layout of the Stickhub USB hub

5.2.2 Power Rail Confirmation and Net Name Handling

Figure 5.15 shows the power rail dialog for the Stickhub. The plugin detected four supply rails correctly. The net name `+5V` is internally converted to `P5V` and `+1V8` to `P1V8` to produce valid SPICE node names.

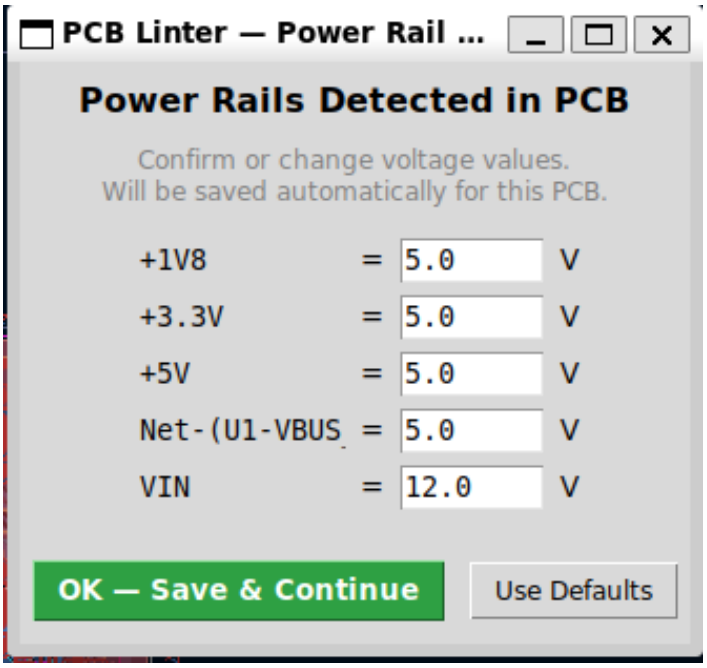


Figure 5.15: Power Rail Dialog for Stickhub: +1V8, +3.3V, +5V, and VIN = 12 V detected

5.2.3 Quality Score and Connectivity Graph

Figure 5.16 shows the report header and Figure 5.17 shows the connectivity graph. Component U1, the USB hub IC, shows high fanout (red star) due to its connections to all 7 USB port nets. This is a genuine design characteristic of a hub IC and is correctly flagged as a warning.

Figure 5.16: Stickhub: Quality Score 85/PASS

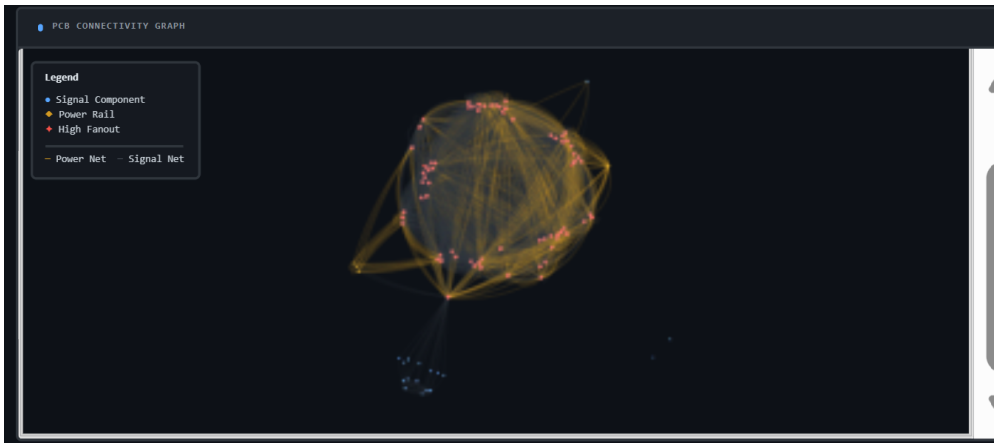


Figure 5.17: Stickhub Connectivity Graph: U1 shown as red star due to high fanout from 7 USB port connections

5.2.4 Clustering Results

Figures 5.18, 5.19, and 5.20 show the voltage, current, and power dissipation clustering results respectively. The voltage clustering shows five domains. The Clustering Quality label is Poor because the USB differential pair nets (D+ and D-) have very similar voltage values (1.65 V and 1.86 V), which produces a low inter-cluster separation relative to intra-cluster spread. This is a known limitation for circuits with closely spaced voltage rails.

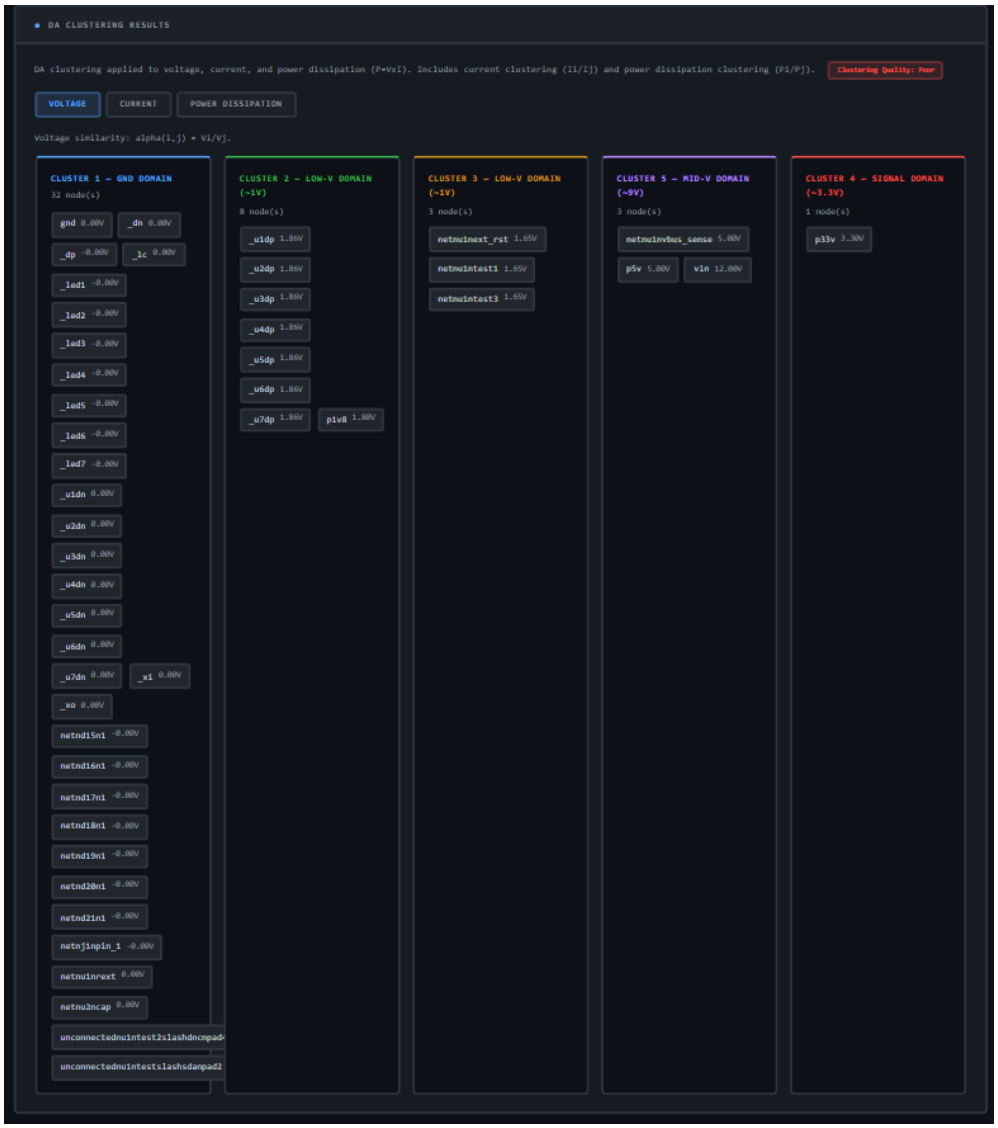


Figure 5.18: Stickhub: Voltage clustering showing 5 domains. Clustering Quality is Poor due to closely spaced USB differential pair voltages.

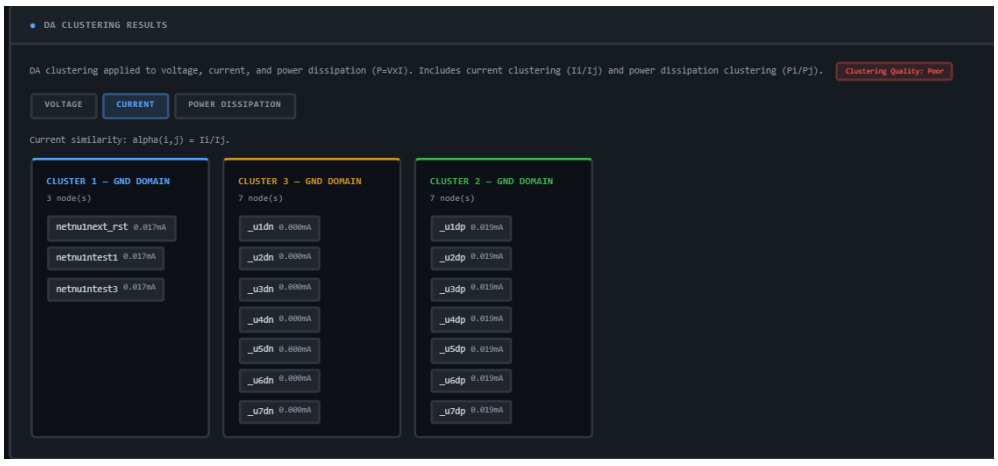


Figure 5.19: Stickhub: Current clustering showing 3 current domains

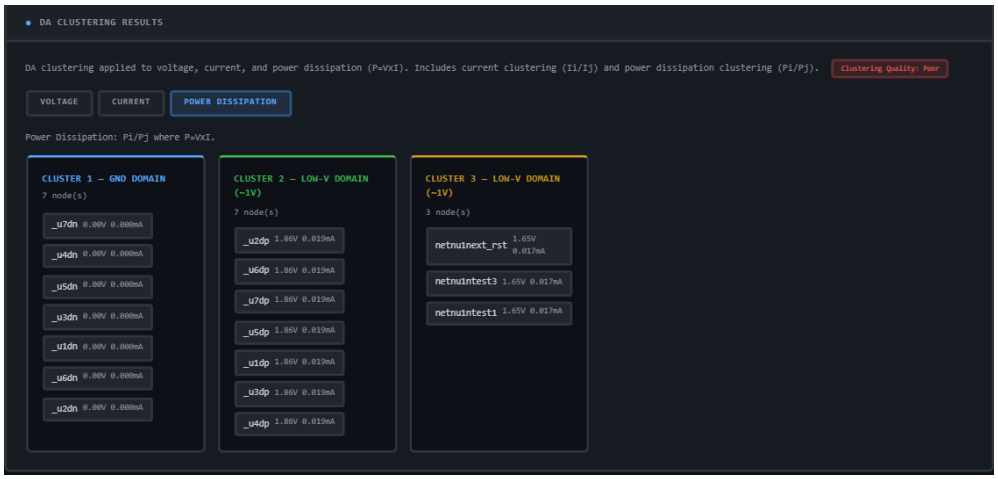


Figure 5.20: Stickhub: Power dissipation clustering showing 3 power domains

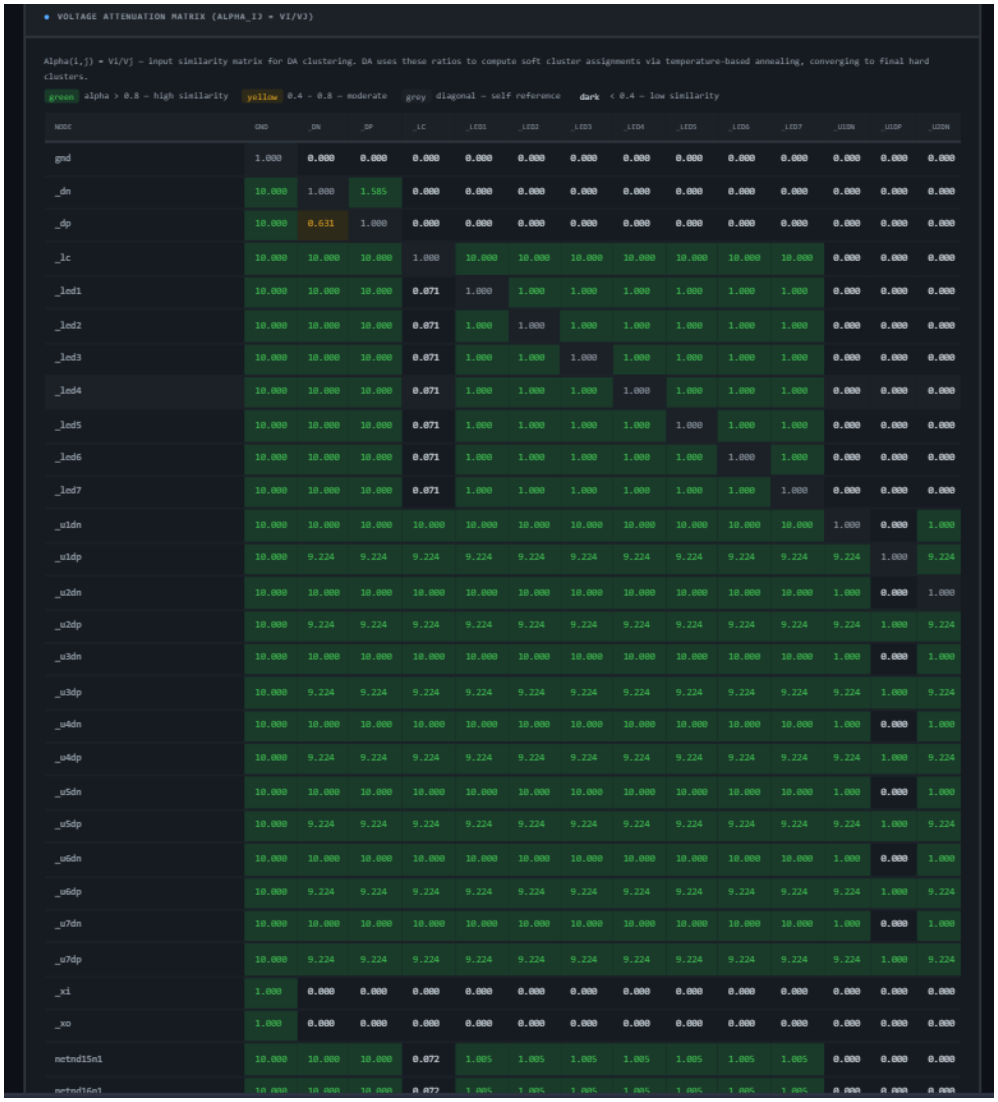


Figure 5.21: Stickhub: Voltage Attenuation Matrix

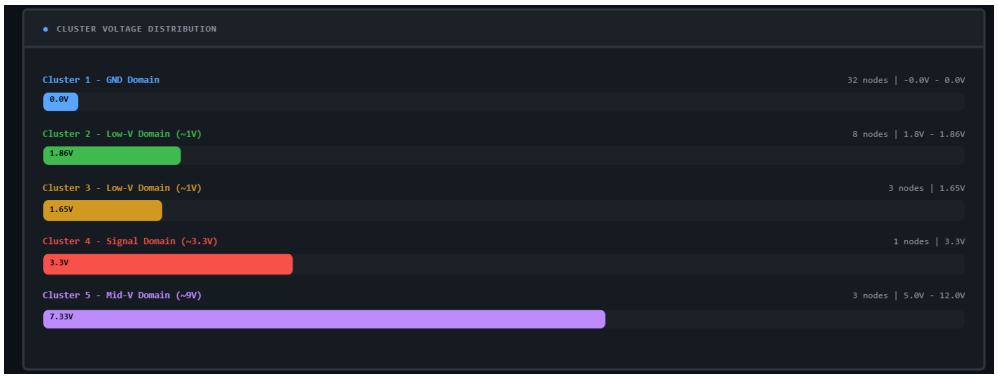


Figure 5.22: Stickhub: Cluster Voltage Distribution bar chart

Figure 5.23: Stickhub: Issues showing WARNING for U1 high fanout and INFO for 2 unconnected pins by design

Figure 5.24: Stickhub: Net to Component Map

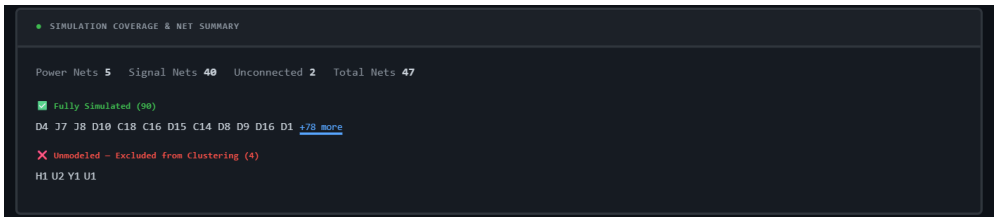


Figure 5.25: Stickhub: Simulation Coverage showing 90 fully simulated and 4 unmodeled components

5.3 Test Circuit 3: LM741 Subcircuit Injection Test

5.3.1 Circuit Description

A minimal inverting amplifier PCB was used to validate the eSim SubcircuitLibrary injection feature in isolation. The circuit contains one LM741 op-amp (U1), two 10 kΩ resistors (R1 for input and R2 for feedback), a positive supply VCC at 5 V, a negative supply VEE at −15 V, and an input voltage source VIN. This test was designed specifically to verify that the plugin correctly locates the `lm_741.sub` file in the eSim SubcircuitLibrary and injects it into the SPICE netlist automatically.

5.3.2 Subcircuit Injection Result

Figure 5.26 shows the terminal output confirming that the `.include lm_741.sub` directive was successfully injected. Without this injection, NGSpice would fail with a model not found error, and the LM741 would produce no useful simulation data. With the injection, the op-amp is fully simulated using its actual SPICE subcircuit model from eSim.

```
prisha_bhatia@LAPTOP-K53PLV9D:~/eSim-Workspace/pcb_linter_output
$ cat ~/eSim-Workspace/pcb_linter_output/pcblinter_sim.cir
* PCB Linter - Auto SPICE v4.5
* Prisha Bhatia, FOSSEE IIT Bombay

.include /home/prisha_bhatia/Downloads/eSim-2.5/library/Subcircu
itLibrary/lm_741/lm_741.sub
R1 VIN 0 10k
R2 VOUT VIN 10k
XU1 VCC VIN VEE VOUT 0 LM741
Vsrc_1 VCC 0 DC 5.0
Vsrc_2 VEE 0 DC -15.0
Vsrc_3 VIN 0 DC 5.0
Rload_1 VOUT 0 100k

.op

.control
run
echo 'GND = 0.0' > /home/prisha_bhatia/eSim-Workspace/pcb_linter
_output/plot_data_v.txt
print allv >> /home/prisha_bhatia/eSim-Workspace/pcb_linter_outp
ut/plot_data_v.txt
echo '' > /home/prisha_bhatia/eSim-Workspace/pcb_linter_output/p
lot_data_i.txt
print @rload_1[i] >> /home/prisha_bhatia/eSim-Workspace/pcb_lint
```

Figure 5.26: Terminal output confirming automatic injection of `.include lm_741.sub` from eSim SubcircuitLibrary

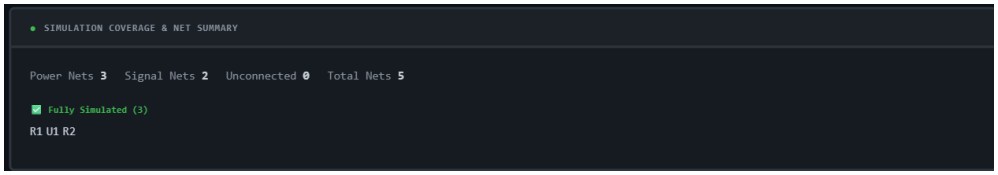


Figure 5.27: LM741 test: Simulation Coverage showing R1, U1, and R2 all fully simulated

5.3.3 Clustering Results

Figures 5.28 and 5.29 show the voltage and current clustering results. The negative supply VEE at $-15\,\mathrm{V}$ is correctly identified and placed in its own cluster, validating Feature 4 (negative supply support).

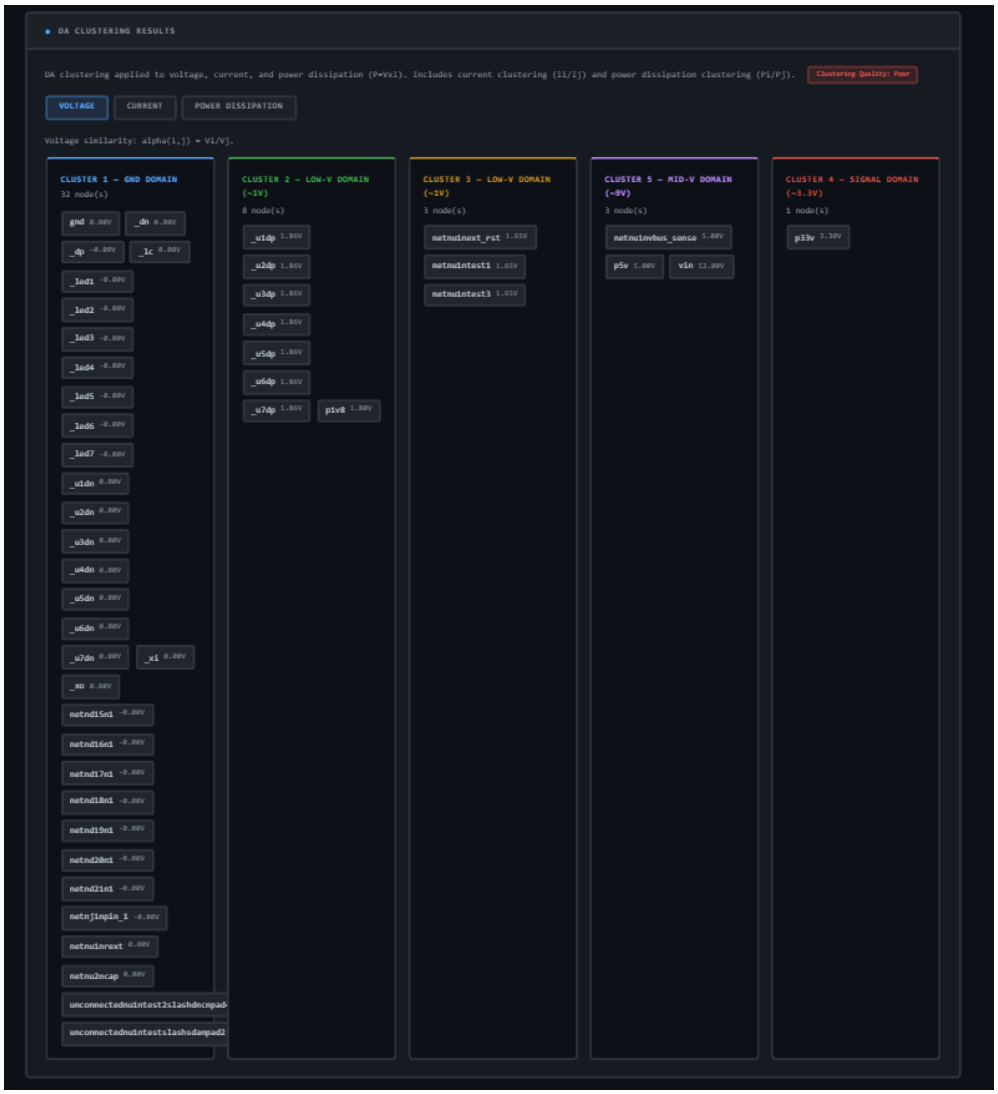


Figure 5.28: LM741 test: Voltage clustering results showing GND, negative supply, and positive supply domains

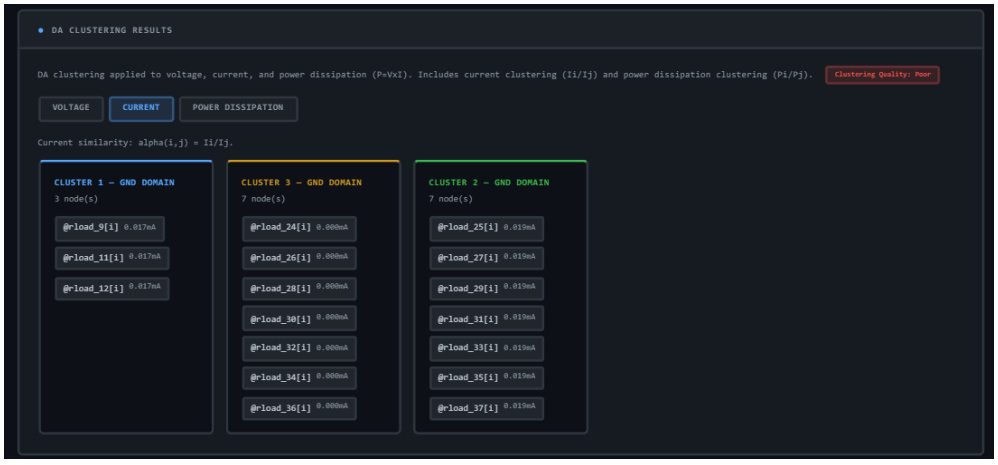
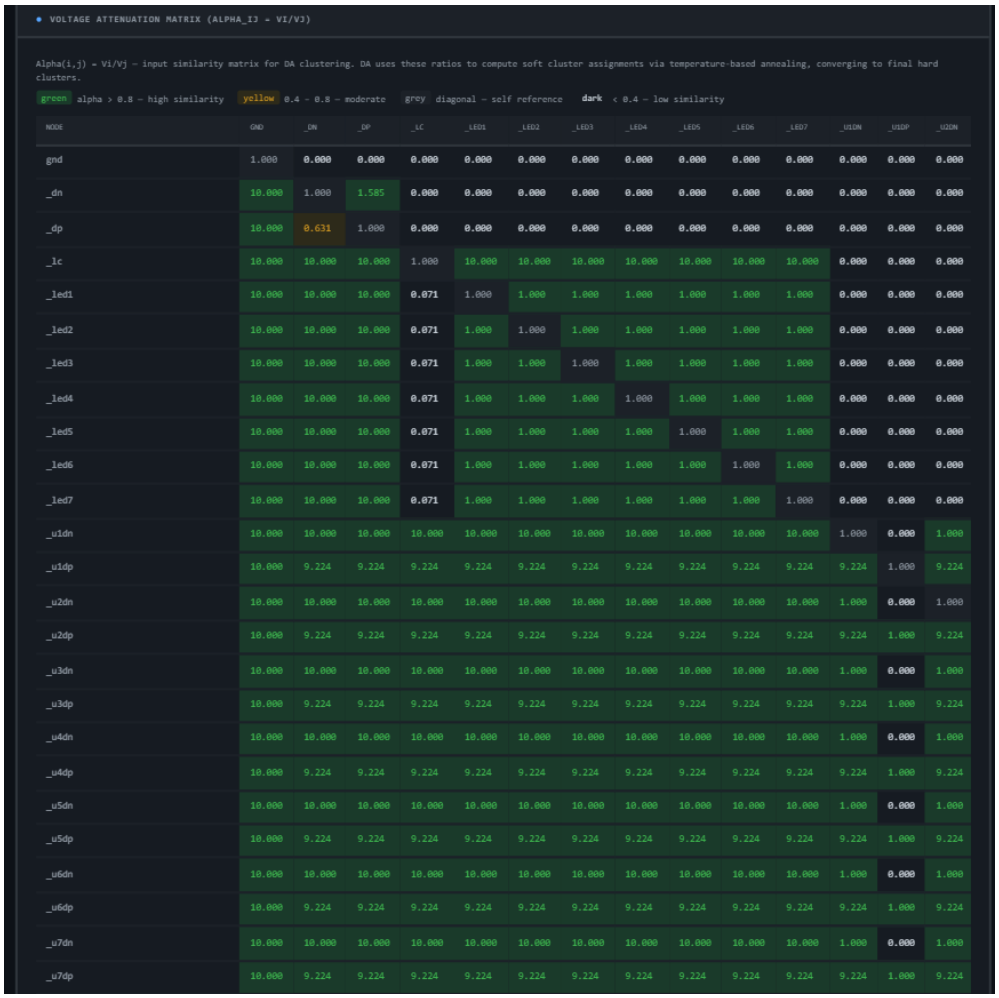


Figure 5.29: LM741 test: Current clustering results



VOLTAGE ATTENUATION MATRIX (ALPHA_IJ = VI/VJ)

Alpha(i,j) = Vi/Vj — input similarity matrix for DA clustering. DA uses these ratios to compute soft cluster assignments via temperature-based annealing, converging to final hard clusters.

green alpha > 0.8 — high similarity yellow 0.4 – 0.8 — moderate grey diagonal — self reference dark < 0.4 — low similarity

NODE	GND	_DN	_DP	_LC	_LED1	_LED2	_LED3	_LED4	_LED5	_LED6	_LED7	_U1DN	_U1DP	_U2DN
gnd	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
_dn	10.000	1.000	1.585	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
_dp	10.000	0.631	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
_lc	10.000	10.000	10.000	1.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	0.000	0.000	0.000
_led1	10.000	10.000	10.000	0.071	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
_led2	10.000	10.000	10.000	0.071	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
_led3	10.000	10.000	10.000	0.071	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
_led4	10.000	10.000	10.000	0.071	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
_led5	10.000	10.000	10.000	0.071	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
_led6	10.000	10.000	10.000	0.071	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
_led7	10.000	10.000	10.000	0.071	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
_u1dn	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	1.000	0.000	1.000
_u1dp	10.000	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	1.000	9.224
_u2dn	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	1.000	0.000	1.000
_u2dp	10.000	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	1.000	9.224
_u3dn	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	1.000	0.000	1.000
_u3dp	10.000	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	1.000	9.224
_u4dn	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	1.000	0.000	1.000
_u4dp	10.000	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	1.000	9.224
_u5dn	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	1.000	0.000	1.000
_u5dp	10.000	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	1.000	9.224
_u6dn	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	1.000	0.000	1.000
_u6dp	10.000	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	1.000	9.224
_u7dn	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	10.000	1.000	0.000	1.000
_u7dp	10.000	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	9.224	1.000	9.224

Figure 5.30: LM741 test: Voltage Attenuation Matrix

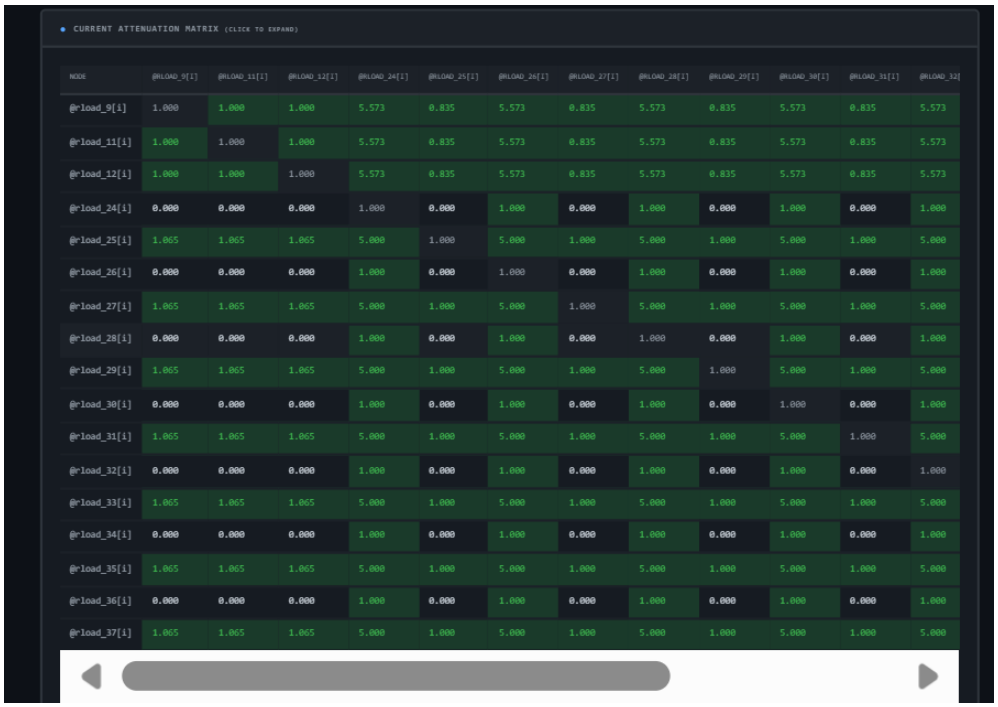


Figure 5.31: LM741 test: Current Attenuation Matrix

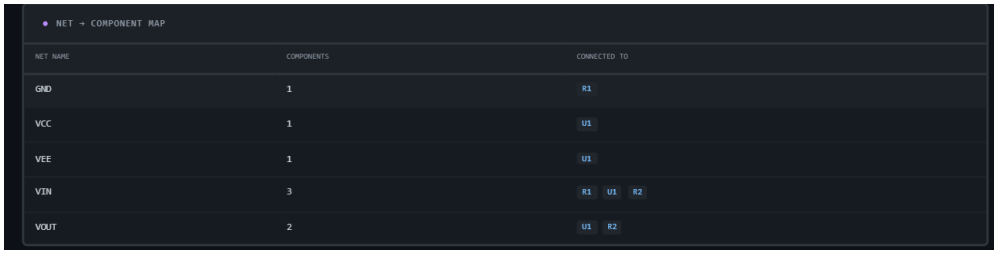


Figure 5.32: LM741 test: Net to Component Map showing GND, VCC, VEE, VIN, and VOUT

Figure 5.33: LM741 test: Issues showing INFO for 2 unconnected pins on U1 by design

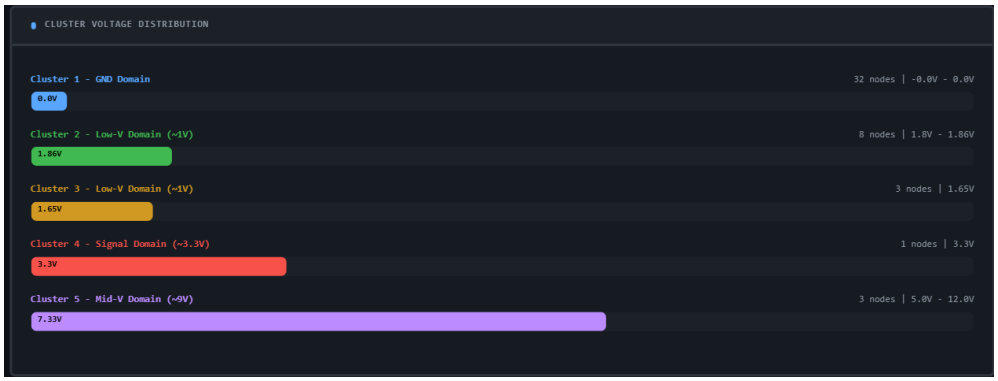


Figure 5.34: LM741 test: Cluster Voltage Distribution bar chart

5.4 Test Circuit 4: Mode 2 Universal Import

5.4.1 Description

Mode 2 allows clustering on any existing simulation data file without requiring a KiCad PCB file. To validate this, the transient simulation output of a Precision Rectifier circuit using an LM741 from the eSim Examples library was imported directly. This circuit has a negative supply rail at $-12\,\mathrm{V}$ and produces a rectified output voltage of approximately $3.8\,\mathrm{V}$ at the output node.

5.4.2 File Import Process

Figure 5.35 shows the file browser for selecting the voltage data file. Figure 5.36 shows the optional current file import dialog.

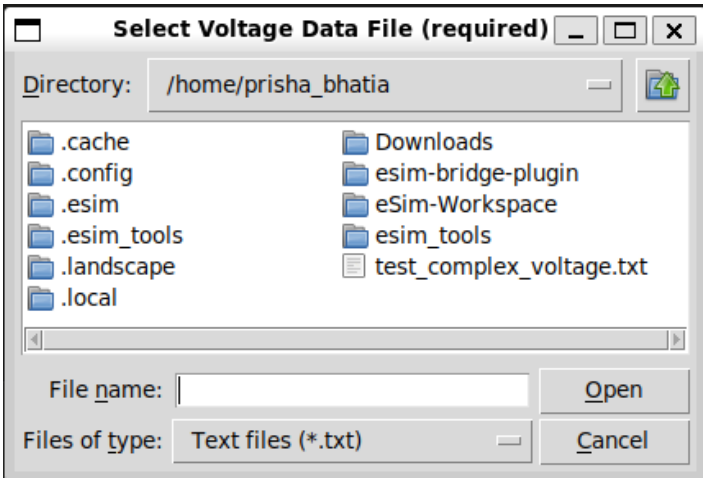


Figure 5.35: Mode 2: File browser for selecting the voltage data file

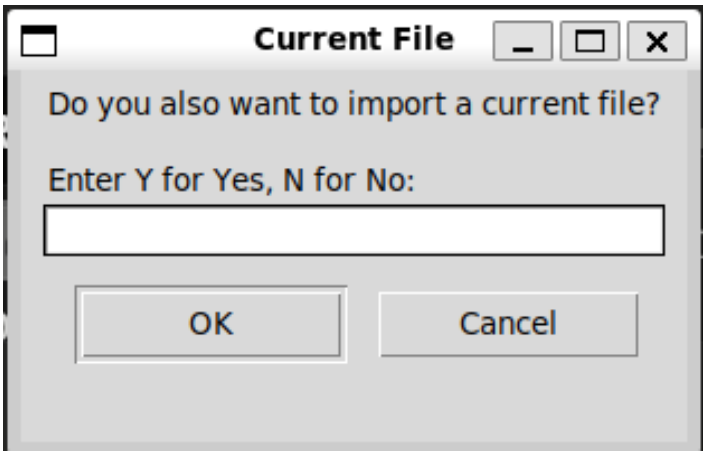


Figure 5.36: Mode 2: Optional current file import dialog

5.4.3 Results

Figure 5.37 shows the clustering result. The DA algorithm correctly identified four domains: GND (0 V), the negative supply (−12 V), the rectified output signal (approximately 3.8 V), and the 12 V input domain.

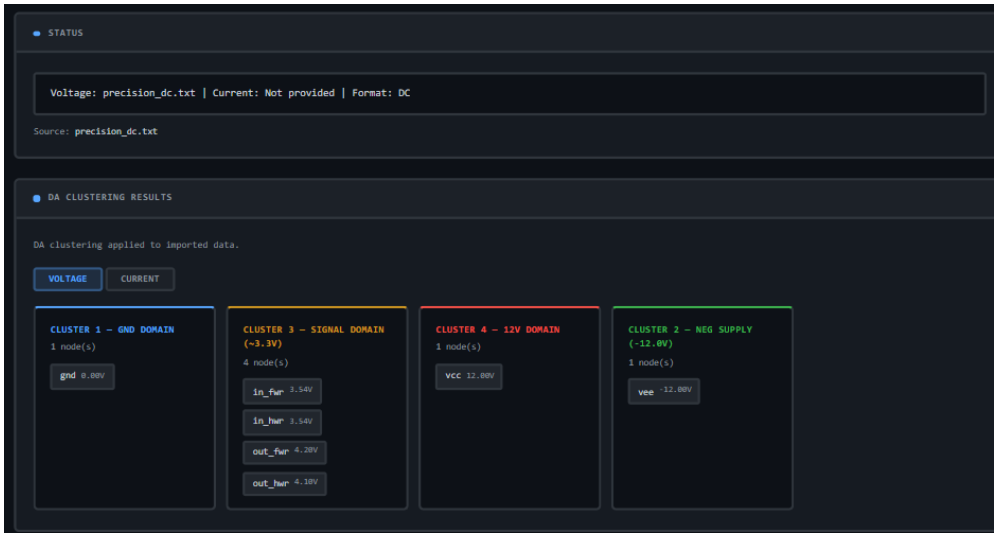


Figure 5.37: Mode 2: Clustering result showing 4 domains including negative supply at −12 V

Figure 5.38: Mode 2: Cluster Voltage Distribution bar chart for the Precision Rectifier data

Figure 5.39: Mode 2: Voltage Attenuation Matrix for the Precision Rectifier data

5.4.4 Results Comparison

Table 5.4 compares the results across all four test circuits.

Table 5.4: Summary of results across all test circuits

Circuit	Mode	Score	V-Cl.	I-Cl.	P-Cl.	Quality
PIC Programmer	1	100 / PASS	5	5	5	Excellent
Stickhub USB	1	85 / PASS	5	3	3	Poor
LM741 Test	1	100 / PASS	3	1	0	Good
Precision Rect.	2	N/A	4	0	0	N/A

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

This project developed a KiCad PCB Linter Plugin (v4.5) that automates simulation-based power domain analysis for any KiCad PCB. The plugin was motivated by the research of Baranwal and Salapaka [1] on clustering of power networks using the Deterministic Annealing algorithm. The core attenuation matrix formulation from that work was adapted to the PCB domain using directly simulated node voltages rather than sensitivity matrices derived from load-flow computation.

The plugin was validated on four different test cases. The PIC Programmer result confirmed correct identification of five voltage domains (GND, PC signal, diode nodes, 5 V, and 13 V VPP) with a quality score of 100. The Stickhub result demonstrated handling of non-standard net names with plus signs and correct high-fanout detection. The LM741 test confirmed successful eSim SubcircuitLibrary injection. The Mode 2 test confirmed correct handling of negative supply rails in imported data.

The current clustering and power dissipation clustering capabilities are novel contributions to the field of PCB analysis tools. No existing open-source plugin provides these features on a `.kicad_pcb` file.

6.2 Limitations

- For series circuits, NGSpice reports only a single total current, so per-net current clustering is not meaningful.
- ICs not present in the eSim SubcircuitLibrary are excluded from simulation and clustering.
- The Mode 2 transient import uses only the last time step value. Time- varying domain membership is not yet captured.

- The quality metric is Poor for circuits with closely spaced voltage rails such as USB differential pairs, because their voltage difference is small relative to the inter-cluster separation.

6.3 Future Scope

- **Schematic-based voltage reading.** Parse the `.kicad_sch` file to extract actual voltage source values rather than inferring from net names.
- **Transient clustering over time.** Apply DA at each time step to produce a time-series view of evolving domain membership.
- **Trace width recommendation.** Use current cluster centroids to suggest IPC-2221 compliant trace widths for each power domain.
- **KiCad PCB overlay.** Colour PCB traces in KiCad based on cluster membership using the `pcbnew` API directly.
- **Expanded model support.** Add power electronics devices and microcontroller models contributed by the FOSSEE community.

Bibliography

[1] M. Baranwal and S. M. Salapaka, "Clustering of Power Networks: An Information-Theoretic Perspective," *IEEE Conference on Decision and Control*, University of Illinois at Urbana-Champaign, 2015.

[2] K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2210–2239, 1998.

[3] FOSSEE Project, IIT Bombay. *eSim: Free and Open Source EDA Tool.* Available at: `https://esim.fossee.in` [Accessed May 2026].

[4] NGSpice Development Team. *NGSpice User Manual, Version 42.* Available at: `https://ngspice.sourceforge.io` [2024].

[5] KiCad EDA. *KiCad 8.0 Scripting Reference.* Available at: `https://docs.kicad.org/8.0/en/scripting/` [Accessed April 2026].

[6] A. Hagberg, P. Swart, and D. Schult, "Exploring Network Structure, Dynamics, and Function using NetworkX," *Proceedings of SciPy 2008*, pp. 11–15, 2008.

[7] L. W. Nagel and D. O. Pederson, "SPICE (Simulation Program with Integrated Circuit Emphasis)," *EECS Technical Report ERL-M382*, University of California, Berkeley, 1973.

Appendix A

Daily Work Log

Day	Date	Phase	Work Description
Wed	18/02/2026	Orientation	Attended FOSSEE SLI Spring 2026 orientation session.
Thu	19/02/2026	Exploration	Explored internship tasks; studied KiCad plugin scope.
Fri	20/02/2026	Exploration	Studied prior intern work and FOSSEE eSim documentation.
Mon	23/02/2026	Exploration	Researched KiCad scripting API and ActionPlugin registration.
Wed	25/02/2026	Meeting	Attended Google Meet with mentor; discussed plugin direction.
Fri	27/02/2026	Proposal	Prepared plugin proposal document for idea pitch.
Mon	02/03/2026	Development	Set up WSL2 Ubuntu; created plugin directory structure.
Thu	05/03/2026	Development	Implemented net extraction from KiCad PCB using pcbnew API.
Mon	09/03/2026	Development	Implemented SPICE generation for R, C, L, D, Q components.
Thu	12/03/2026	Development	Added power rail inference and Rload current measurement.
Sat	14/03/2026	Meeting	Attended Google Meet; discussed DA algorithm theory.
Mon	16/03/2026	DA Algorithm	Studied Baranwal and Salapaka; implemented alpha matrix.

Table A.1 continued

Day	Date	Phase	Work Description
Wed	18/03/2026	DA Algorithm	Implemented Gibbs distribution and temperature cooling.
Fri	20/03/2026	DA Algorithm	Added voltage clustering; tested on PIC Programmer PCB.
Sat	21/03/2026	Meeting	Attended Google Meet; discussed current clustering approach.
Mon	23/03/2026	DA Algorithm	Implemented current clustering using Rload measurement data.
Wed	25/03/2026	DA Algorithm	Implemented power dissipation clustering using P = V times I.
Sat	28/03/2026	Meeting	Discussed eSim SubcircuitLibrary integration approach.
Mon	30/03/2026	eSim Library	Studied SubcircuitLibrary structure; implemented library scanner.
Thu	02/04/2026	eSim Library	Implemented .include injection for LM741, LM7805, NE555.
Sat	04/04/2026	Meeting	Reviewed subcircuit injection results with mentor.
Mon	06/04/2026	Features	Implemented Power Rail Dialog with JSON auto-save.
Wed	08/04/2026	Features	Implemented floating net detection and cluster auto-naming.
Fri	10/04/2026	Features	Implemented cluster quality metric and negative supply support.
Mon	13/04/2026	HTML Report	Implemented HTML report with dark mode and quality score.
Wed	15/04/2026	HTML Report	Added PCB connectivity graph using NetworkX and PyVis.
Fri	17/04/2026	HTML Report	Added DA clustering tabs, bar chart, and net to component map.
Mon	20/04/2026	Mode 2	Implemented Universal Parser for DC, AC, Transient, CSV formats.
Wed	22/04/2026	Mode 2	Tested Mode 2 on Precision Rectifier NGSpice output data.

Table A.1 continued

Day	Date	Phase	Work Description
Mon	27/04/2026	Testing	Tested plugin on Stickhub PCB; debugged high-fanout detection.
Wed	29/04/2026	Testing	Validated LM741 subcircuit injection on test PCB.
Mon	04/05/2026	Refinement	Code cleanup; fixed duplicate bar chart function.
Wed	06/05/2026	Report	Drafted Chapters 1 to 3.
Thu	07/05/2026	Report	Drafted Chapter 4 with all tables and diagrams.
Fri	08/05/2026	Report	Drafted Chapter 5 with all screenshots.
Tue	12/05/2026	Report	Final review and corrections.
Wed	13/05/2026	Submission	Final report submitted.