FOSSEE Summer Fellowship 2025
Project Report

On

Offline Retrieval-Augmented Generation AI Architecture for eSim-Based Assisted Schematic Synthesis & Fault Diagnosis in Electronic Circuit Diagrams via Integrated Computer Vision Driven Analysis

Submitted by:

Aditya Bhattacharya
Rajiv Gandhi Institute of Petroleum Technology

Under the guidance of:

Mentor:
Mr. Sumanto Kar
Associate Project Manager, FOSSEE Project
Indian Institute of Technology, Bombay

Prof. Kannan M. Moudgalya
Chemical Engineering Department
Indian Institute of Technology, Bombay

August 2025

Acknowledgement

I would like to express my heartfelt and sincere gratitude to the **FOSSEE** team for granting me the privilege and trust to contribute to the enhancement of the open-source software **eSim**. Being part of this initiative has been an immensely rewarding experience, one that has not only expanded my technical skills but also deepened my appreciation for collaborative, open-source development. I feel truly honored to have been a part of such a meaningful project that aligns with my passion for innovation and knowledge sharing.

I am profoundly thankful to **Prof. Kannan M. Moudgalya** for his remarkable guidance, encouragement, and steadfast support throughout the FOSSEE Summer Internship program. His vision, commitment to excellence, and ability to inspire have had a lasting impact on my approach to learning and problem-solving. His guidance has been instrumental in shaping both my technical and professional growth during this journey.

My sincere appreciation extends to the entire eSim team for their generosity in offering assistance, for their readiness to answer questions, and for providing the resources that were essential for the successful completion of my project. I am especially grateful to **Mrs. Usha Viswanathan**, **Dr. Ritu Pahwa** for their constant encouragement, patience, and unwavering support, which motivated me to keep pushing forward even during challenging moments.

I owe a special debt of gratitude to my mentor, **Mr. Sumanto Kar**, whose continuous guidance, willingness to share his deep expertise, and collaborative spirit made my learning journey both enriching and enjoyable. His ability to explain complex concepts with clarity, combined with his constant encouragement, played a crucial role in enabling me to overcome obstacles and refine my work.

I am equally grateful to my fellow interns, whose enthusiasm, collaboration, and camaraderie transformed this internship into not just a period of hard work, but also one of shared learning, friendship, and mutual growth. Working alongside such talented and driven individuals has been an inspiring and motivating experience.

I close with deep appreciation for all who have been part of this journey. This internship has not only enriched my technical knowledge but has also instilled values of collaboration, perseverance, and continuous learning. I carry forward these lessons with the resolve to apply them meaningfully and contribute to creating solutions that inspire and make a lasting impact.

Contents

Chapter 1

Introduction

1.1 FOSSEE

The Free/Libre and Open Source Software for Education (FOSSEE) project, headquartered at IIT Bombay, is a distinguished initiative dedicated to promoting open-source software in education and research. Its core mission is to reduce dependence on proprietary tools by encouraging the adoption of cost-effective and accessible alternatives.
Through an extensive range of resources, including detailed documentation, structured tutorials, hands-on workshops, and training programs, FOSSEE empowers students, educators, and professionals to integrate open-source solutions into academic and professional work. By fostering a culture of collaboration and inclusivity, the project has significantly contributed to the democratization of technology, creating new avenues for innovation and lifelong learning.

1.2 eSim

eSim, developed under the FOSSEE project at IIT Bombay, is an advanced open-source Electronic Design Automation (EDA) platform for circuit design and simulation. It consolidates multiple open-source packages into a unified environment, enabling efficient schematic creation, simulation, and analysis.
This platform caters to students, academic institutions, and industry professionals seeking an affordable alternative to commercial design software. It features schematic creation tools, SPICE-based simulation, PCB design capabilities, and an extensive component library. A notable enhancement is its Subcircuit functionality, which facilitates the construction of complex designs by integrating smaller, reusable modules. eSim exemplifies FOSSEE's vision of encouraging innovation, knowledge sharing, and accessibility within engineering education and practice.

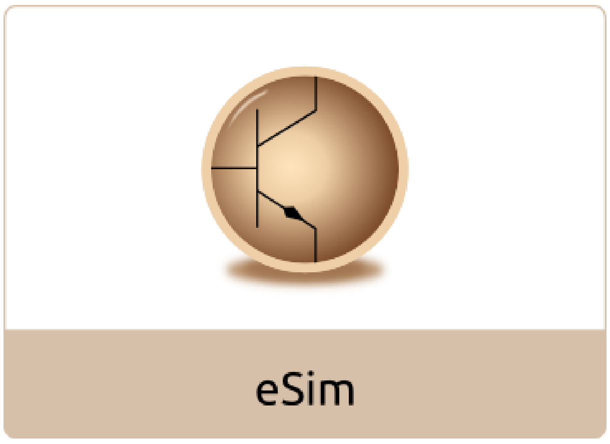


Figure 1.1: eSim-FOSS Software

1.3 NgSpice

NgSpice is a widely adopted open-source simulator based on the SPICE framework, designed for the analysis of analog, digital, and mixed-signal circuits. It supports a diverse range of components, including transistors, diodes, passive elements, transmission lines, and other semiconductor devices.

Figure 1.2: Ngspice

Its capabilities extend from simulating basic logic gates to complex mixed-signal systems. A comprehensive library of device models is available, sourced from community contributions, semiconductor manufacturers, and foundries. Circuit descriptions are provided in the form of netlists, and simulation outputs can be visualized as waveforms or exported for further study, enabling precise evaluation of circuit performance.

1.4 Makerchip

Makerchip is a versatile platform for digital circuit design, simulation, and debugging, offering both browser-based and desktop environments. It supports Verilog, SystemVerilog, and Transaction-Level Verilog, integrating a combination of open-source and proprietary tools to deliver a comprehensive development experience. Within the eSim ecosystem, Makerchip is

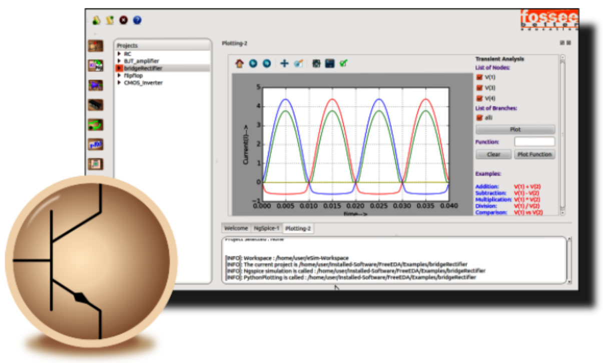
Figure 1.3: Makerchip

linked through a Python-based Makerchip-App, enabling seamless access to the Makerchip IDE. The platform's intuitive interface, coupled with efficient workflows, simplifies the design process for beginners while providing advanced features for experienced developers.
While various open-source tools provide valuable individual capabilities, few offer a complete, integrated EDA solution. eSim addresses this gap by combining schematic creation, simulation, and PCB design into a single coherent package.

Chapter 2

Features of eSim

The development of eSim aims to deliver a complete open-source EDA solution for electronics and electrical engineers. It supports schematic creation, PCB layout design, and simulation of analog, digital, and mixed-signal circuits. It also offers tools for developing custom models and components. The principal features include:



```
[INFO]: Workspace : /home/user/eSim-Workspace
[INFO]: The current project is /home/user/Installed-Software/FreeEDA/Examples/bridgeRectifier
[INFO]: Ngspice simulation is called : /home/user/Installed-Software/FreeEDA/Examples/bridgeRectifier
[INFO]: PythonPlotting is called : /home/user/Installed-Software/FreeEDA/Examples/bridgeRectifier
```

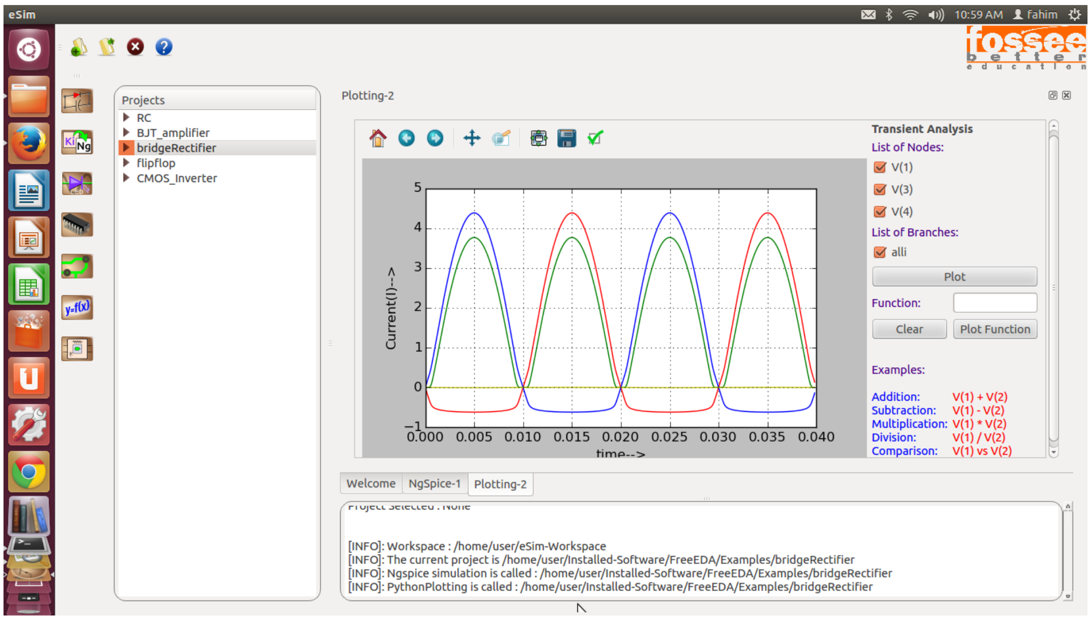
Figure 2.1: Dashboard: eSim Software

* **Schematic Creation** – A user-friendly graphical interface enables quick and precise circuit drafting. Components from an extensive library can be placed directly onto the schematic, with comprehensive editing tools for movement, rotation, and labeling.

* **Circuit Simulation** – SPICE-based simulation allows transient, AC, and DC analyses, offering insight into circuit behavior across time and frequency domains. An integrated waveform viewer assists in interpreting results and verifying design accuracy.

* **PCB Design** – The layout editor provides precise placement of components and routing of connections. Built-in design rule checks ensure compliance with manufacturing and electrical standards. Gerber file generation is supported for fabrication purposes.

* **Subcircuit Functionality** – Modular design is facilitated through reusable subcircuits, allowing smaller building blocks to be integrated into larger systems. This approach enhances design efficiency and maintainability.



```
[INFO]: Workspace : /home/user/eSim-Workspace
[INFO]: The current project is /home/user/Installed-Software/FreeEDA/Examples/bridgeRectifier
[INFO]: Ngspice simulation is called : /home/user/Installed-Software/FreeEDA/Examples/bridgeRectifier
[INFO]: PythonPlotting is called : /home/user/Installed-Software/FreeEDA/Examples/bridgeRectifier
```

Figure 2.2: Interface

* **Integration of Open-Source Tools** – eSim incorporates packages such as KiCad, NgSpice, and GHDL to provide a complete EDA environment without the cost constraints of proprietary software.

Chapter 3

Problem Statement

3.1 Motivation and Identified Challenges

FOSSEE eSim is widely used in academic and research settings for designing and simulating electronic circuits. Despite its utility, it lacked any form of built-in assistance for users, resulting in several challenges that limited both usability and learning outcomes. Designing and debugging electronic circuits is inherently complex, requiring precise connections, correct component values, and a clear understanding of simulation results. Without guidance, users, particularly students and novices, often resorted to trial-and-error methods, which were time-consuming, frustrating, and prone to repeated mistakes.

An **agentic AI**-powered assistant can address this gap by automating error analysis and providing real-time, intelligent feedback. By leveraging machine learning and natural language processing **(NLP)** techniques, such a system can parse error logs, classify error types, and map them to likely corrective actions. Beyond error detection, it can provide concise and actionable troubleshooting suggestions tailored to the specific circuit design context, thereby significantly reducing the manual effort required from users.

3.2 Causes for Developing an Intelligent Offline Assistance System

Several key factors made the development of a dedicated offline AI support system necessary:

* **Absence of Circuit Debugging Support:** – Users had no automated way to detect design errors, incorrect component connections, or simulation issues. Even small mistakes in schematic design could result in incorrect outputs, but the system provided no mechanisms to flag or explain these errors.

* **No Query Resolution or Context-Aware Assistance:** – Users could not ask the system questions such as, ***"Why is my RC filter not functioning?" or "Which component values are causing this error?"*** There was no AI-powered mechanism to interpret queries, analyze the circuit, and provide actionable guidance

* **Manual Troubleshooting Burden:** – Every error had to be identified manually, requiring extensive checking of each connection and component. This repetitive process often discouraged users and reduced the educational value of experimentation.

* **Lack of Learning Support:** – The system did not facilitate understanding or improvement. Beginners struggled to comprehend why a circuit failed or how to optimize designs, limiting the software's potential as a learning tool.

* **No Retention of Context or Iterative Guidance:** – Users could not build on previous queries. Each analysis started afresh, preventing iterative learning and increasing the cognitive load for complex designs.

* **Limited Multimodal Interaction:** – There was no integration of visual circuit analysis, textual explanations, or voice feedback. Users could not leverage multiple modalities simultaneously to understand errors and suggested improvements, which reduced engagement and comprehension.

* **Accessibility and Usability Challenges:** – The software did not provide intuitive guidance or support for users with different levels of expertise. Students, educators, and researchers had to navigate the platform without structured help, reducing efficiency and user confidence.

3.3 The Need for an Offline Intelligent Solution

These limitations highlighted the necessity for a comprehensive, offline, AI-driven support system capable of:

* Assisting users in detecting faults and errors in electronic circuits.

* Providing context-aware, actionable guidance for iterative design improvements.

* Supporting multimodal interactions, combining visual diagram analysis, textual explanations, and voice-based assistance.

* Maintaining chat history and caching to retain previous queries, allowing continuous learning and follow-ups.

* Enhancing educational outcomes by providing explanations that reinforce circuit theory and practical design principles.

* Improving user experience and accessibility, making the software intuitive for both beginners and advanced users.

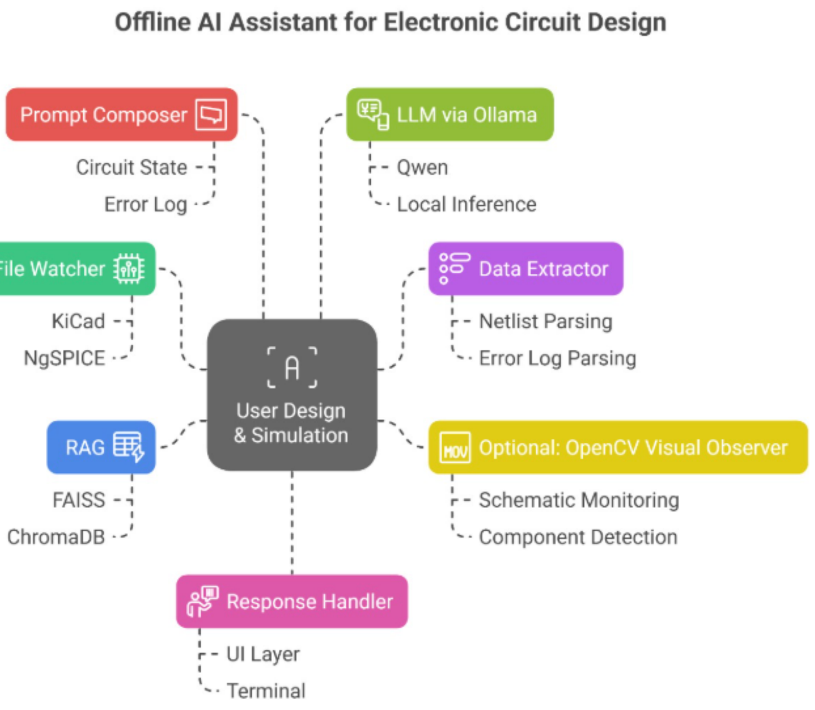


Figure 3.1: eSim-bot: Architecture Design

The development of this system is crucial in two key areas:

* **Design Productivity:** – By automating error detection and providing immediate, intelligent feedback, the assistant minimizes downtime caused by troubleshooting, thereby streamlining the design and simulation workflow. .

* **Learning Support :** – By offering interactive explanations and guided assistance, the system lowers the barrier for beginners, making circuit design more accessible while promoting self-paced learning.

Chapter 4

Concepts Theories of RAG

4.1 Introducing Retrieval-Augmented Generation (RAG)

To address the inherent limitations of purely generative AI models, researchers have proposed advanced hybrid approaches, among which Retrieval-Augmented Generation (RAG) has gained significant prominence. The concept was first formalized in the seminal paper *"Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" by Lewis et al. (2020),* published by Facebook AI Research.

RAG combines the generative capabilities of large language models (LLMs) with retrieval mechanisms, thereby enhancing both the accuracy and contextual relevance of generated outputs. Unlike traditional generative models, which rely exclusively on static knowledge learned during training, RAG introduces an intermediate retrieval step. In this process, the model queries an external knowledge base such as document repositories, curated databases, or the web before producing a response. This architecture allows the system to incorporate up-to-date and domain-specific information, making it particularly effective in knowledge-intensive tasks where factual correctness is essential.

4.2 Key Components

The effectiveness of RAG arises from the seamless interaction of multiple components:

* **Retrieval Component:** – The retrieval module identifies and ranks documents relevant to a given query. It employs methods such as document indexing, embedding-based similarity search, and ranking algorithms to ensure that the most contextually appropriate references are selected from a corpus.

* **Generation Component :** – Once relevant information is retrieved, the generative model—typically

an LLM such as GPT processes the retrieved content alongside the original query to produce coherent, context-aware responses. This ensures that outputs are both linguistically fluent and factually grounded.

* **Interaction Loop :** – RAG often operates in an iterative loop. If the initial retrieval is insufficient, the generative model may reframe or expand the query, prompting additional retrieval passes. This feedback loop refines responses and improves answer completeness.

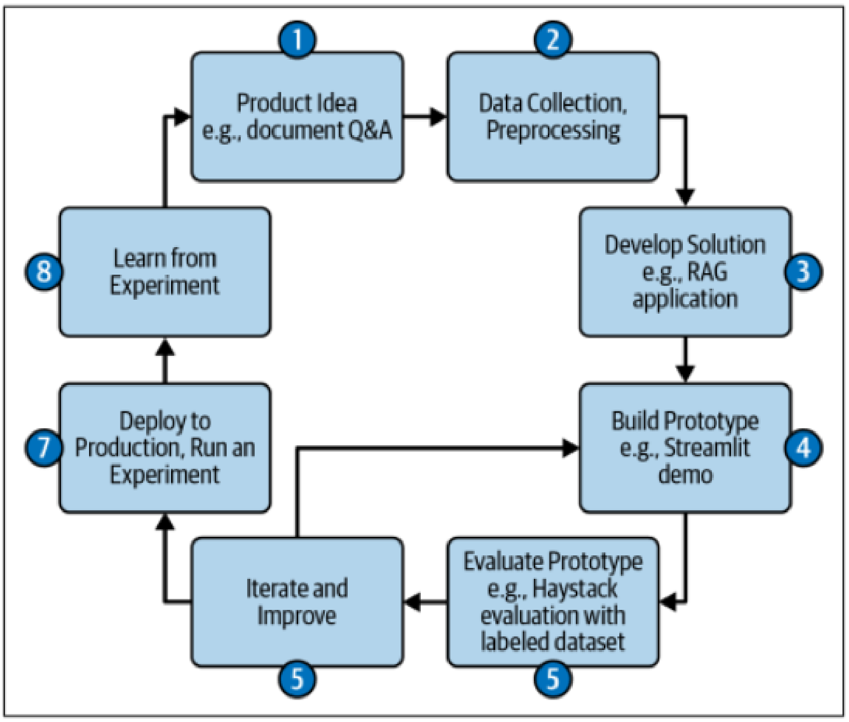

1
Product Idea e.g., document Q&A
2
Data Collection, Preprocessing
3
Develop Solution e.g., RAG application
4
Build Prototype e.g., Streamlit demo
5
Evaluate Prototype e.g., Haystack evaluation with labeled dataset
5
Iterate and Improve
7
Deploy to Production, Run an Experiment
8
Learn from Experiment

Figure 4.1: LLM application development lifecycle

* **Fine-Tuning :** – For domain-specific tasks, RAG models are frequently fine-tuned on specialized datasets. This adaptation enhances accuracy by aligning both retrieval and generation with the terminology and context of the target application domain.

* **Latent Space Representations :** – Queries and documents are transformed into vector representations in a shared latent space. Similarity metrics in this space allow efficient retrieval of semantically relevant passages, ensuring that the most appropriate content is provided to the generative model.

* **Attention Mechanisms :** – Attention layers are employed in both retrieval and generation stages. These mechanisms enable the system to focus selectively on the most informative parts of input documents and queries, improving contextual alignment and reducing irrelevant output.

4.3 Advantages in Question Answering

The RAG framework offers several distinct benefits over conventional generative or retrieval-only systems, particularly for question answering and knowledge-driven applications:

* **Access to Expanded Knowledge Sources:** – By incorporating retrieval, RAG can leverage vast repositories of documents beyond the LLM's static training data, thereby providing responses grounded in current and specialized knowledge.

* **Contextual Relevance :** – Generated responses are enriched with context extracted from retrieved documents, resulting in more precise and contextually aligned outputs.

* **Iterative Refinement :** – The interaction loop ensures that incomplete or ambiguous answers can be refined through repeated retrieval and generation cycles.

* **Adaptability to Complex Queries: :** – RAG handles multi-faceted or domain-specific queries more effectively than simple keyword-based retrieval systems.

* **Domain Specialization :** – Fine-tuning on targeted corpora (e.g., medical, legal, or engineering data) enables RAG to deliver domain-appropriate responses with high reliability.

4.4 RAG Architecture

At a structural level, Retrieval-Augmented Generation integrates two complementary modules into a unified framework:

* **Retrieval Model:** – The retrieval model is tasked with locating and ranking relevant information from large-scale datasets such as document corpora, knowledge bases, or domain-specific repositories. It ensures that the generative component operates on the most contextually meaningful data.

* **Generation Model :** – The generation model, typically a transformer-based LLM, synthesizes the retrieved content into coherent, contextually rich responses. By grounding its outputs in retrieved evidence, it avoids common issues of hallucination observed in standalone generative systems.

4.5 Building the Retrieval System

At a structural level, Retrieval-Augmented Generation integrates two complementary modules into a unified framework:

* The retrieval system forms the backbone of any Retrieval-Augmented Generation (RAG) pipeline. In the context of a chat-to-PDF application, it is responsible for extracting relevant information from document collections and ensuring that the generative model operates on precise, contextually relevant data. An effective retrieval system directly impacts both the accuracy and efficiency of responses, making its design a critical step in system implementation.

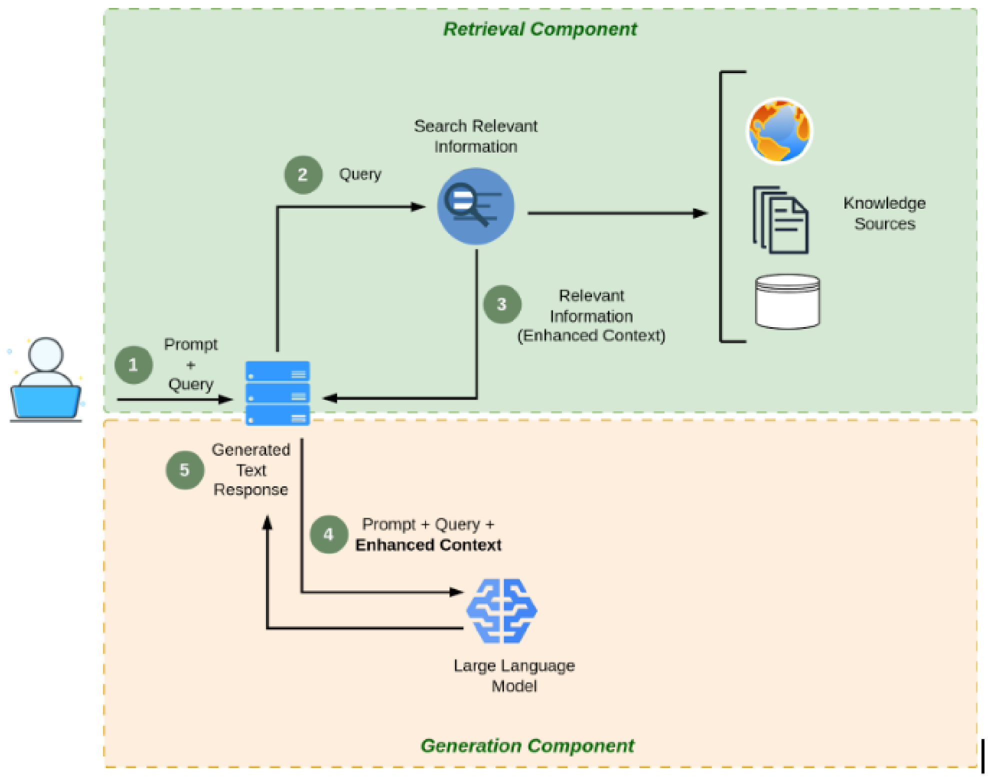

Retrieval Component
Search Relevant Information
2 Query
Knowledge Sources
3 Relevant Information (Enhanced Context)
1 Prompt + Query
5 Generated Text Response
4 Prompt + Query + Enhanced Context
Large Language Model
Generation Component

Figure 4.2: Retrieval & Generation Model

4.6 Selecting an Appropriate Retrieval Model

Choosing the right retrieval model is central to the performance of a RAG-enabled system. Retrieval models determine how queries are interpreted, how documents are ranked, and how efficiently relevant passages are identified. The choice depends on factors such as corpus size, query complexity, and application requirements.
Common retrieval approaches include:

* **TF–IDF *(Term Frequency–Inverse Document Frequency)*:** – A classical model that scores documents based on term frequency relative to their rarity across the corpus. While simple and lightweight, it is best suited for smaller or less complex datasets.

* **BM25 :** – An enhancement of TF–IDF that incorporates document length normalization and term saturation. It has become a robust baseline for modern information retrieval tasks.

* **Vector Space Models:** – These approaches represent documents and queries as vectors in a high-dimensional space, with similarity measured using metrics such as cosine similarity. Methods like Latent Semantic Analysis (LSA) or embedding-based models (e.g., **Word2Vec**) fall into this category.

* **Neural Ranking Models :** – Transformer-based retrieval models, such as BERT or its variants, capture deep semantic relationships between queries and documents. These models can be fine-tuned for domain-specific applications, making them highly effective in specialized contexts.

* **Hybrid Models :** – Combining traditional lexical approaches (e.g., **BM25**) with neural embedding-based methods can yield balanced performance, leveraging the precision of keyword-based retrieval with the adaptability of semantic search.

4.7 Key Considerations:

* **Domain and Corpus Size:** – Smaller, domain-specific collections may perform adequately with traditional methods, while large or diverse corpora often benefit from neural or hybrid approaches.

* **Scalability:** – The model must sustain consistent retrieval performance as the number of documents grows.

* **Resource Requirements:** – TNeural models, while more accurate, demand greater computational resources, which must be weighed against efficiency and cost

Chapter 5

Vector Embeddings for RAG

5.1 Embeddings and Vector Databases for RAG Retrieval

Beyond the choice of retrieval model, modern retrieval systems increasingly rely on vector embeddings and vector databases to achieve high-quality and scalable information access. Embeddings transform textual content into numerical vectors in a high-dimensional space, while vector databases enable efficient similarity search over these embeddings. Together, they form the foundation of semantic retrieval in RAG architectures.

5.2 Vector Embeddings: An Overview

Vector embeddings represent words, sentences, or entire documents as dense numerical vectors, where geometric relationships capture semantic similarity. In this representation, related concepts occupy nearby positions in the vector space, while unrelated concepts are farther apart.

5.2.1 Types of Embeddings:

* **Word Embeddings** ***(e.g., Word2Vec, GloVe)*** – Map individual words into vector space, preserving semantic relationships (e.g., “king” and “queen” are positioned closely).

* **Document Embeddings** ***(e.g., Doc2Vec, BERT-based encoders)*** – Encode longer texts such as paragraphs or full documents. Documents with similar themes produce embeddings that are proximal in the vector space.

5.3 Role of Vector Databases in Retrieval

While embeddings provide a semantic representation of data, vector databases or similarity search engines enable their efficient storage, indexing, and retrieval. These databases are optimized for large-scale similarity search, which is crucial for real-world deployments of RAG systems.

5.3.1 Advantages of Vector Databases in RAG::

* **Efficient Similarity Search:** – Indexing methods such as Approximate Nearest Neighbor (ANN) allow fast retrieval of semantically similar vectors, even in datasets containing millions of documents.

* **Scalability:** – Designed to handle large and growing corpora while maintaining low-latency search performance.

* **Flexible Similarity Metrics::** – Support for multiple similarity measures (e.g., cosine similarity, Euclidean distance, Jaccard index) allows fine-tuning relevance ranking to the application domain.

* **Integration with Retrieval Models:** – Vector databases can act as the first stage in retrieval pipelines, quickly narrowing down candidate documents for deeper ranking using neural models.

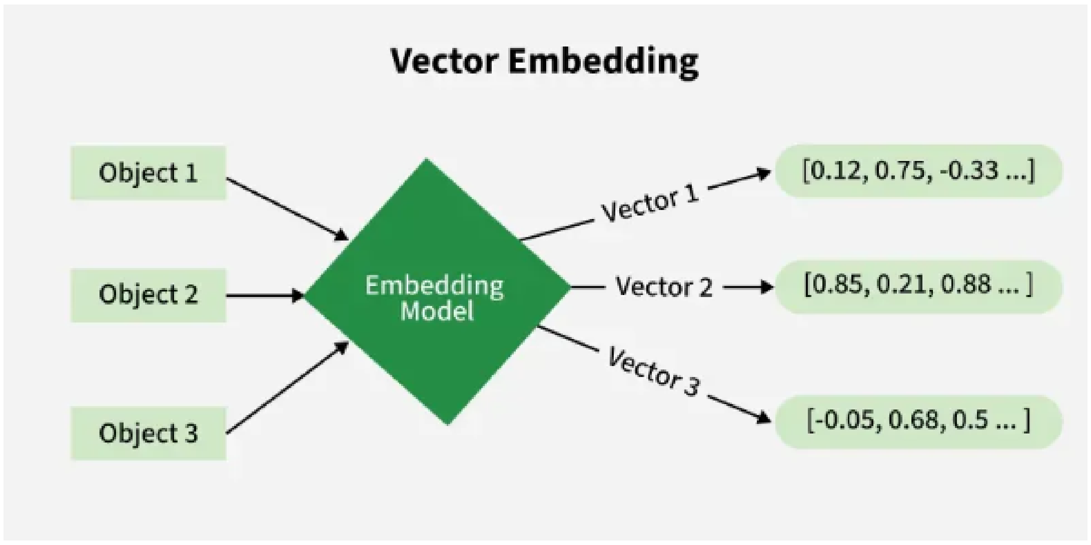


Figure 5.1: Vector Embeddings

5.3.2 Popular Vector Databases

* Several open-source and commercial vector databases are widely adopted in industry and research, including Qdrant, Weaviate, Pinecone, pgvector, Milvus, and Chroma. Selection should balance scalability, latency, cost-efficiency, and ecosystem integration, as this choice has long-term implications for the retrieval system's maintainability.

Pinecone

weaviate

Qdrant Chroma

Chapter 6

Data Extraction & Ingestion

6.1 RAG Data Ingestion Pipeline

Before a Chat-to-PDF application can effectively retrieve information from a vector database, it is necessary to preprocess raw PDF documents into a structured and searchable form. This preprocessing stage transforms unstructured content into clean, normalized, and indexed data, enabling precise and efficient retrieval. It can be compared to creating a well-organized library catalog without such preparation, even the most advanced retrieval models struggle to locate relevant information consistently. Although the focus here is on PDF documents due to their prevalence across industries, the methods described are broadly applicable to other file formats as well.

6.1.1 PDF Text Extraction

The first step in building the ingestion pipeline is extracting raw text from PDF files. Since PDFs may contain selectable text as well as embedded images, careful handling is required.

6.1.2 Key Steps in Text Extraction:

* **Text Extraction Libraries:** –such as *PyPDF2*, pdfminer.six, or pdf2txt can programmatically extract textual data from PDFs.

* **Handling Scanned PDFs :** – For documents that only contain images, *Optical Character Recognition (OCR)* tools such as *Tesseract* convert images into machine-readable text.

6.2 Handling Multi-Page Context:

PDFs typically span multiple pages, and retrieval performance depends on preserving logical continuity rather than treating pages as isolated fragments.

6.2.1 Strategies for Preserving Context:

* **Logical Segmentation:** – Divide the text into coherent units such as paragraphs, sections, or chapters, instead of page-level splits.
* **Metadata Extraction :** – Capture information such as document titles, authors, section headers, creation dates, and page numbers. This metadata provides valuable cues for indexing and improves retrieval accuracy.

6.3 Text Cleanup and Normalization:

Raw text from PDFs frequently contains artifacts that degrade retrieval quality. Normalization ensures that the text is standardized before being transformed into embeddings.

6.3.1 Key Normalization Processes:

* **Whitespace and Punctuation Correction:** – Remove unnecessary whitespace, excessive line breaks, and irregular punctuation.
* **Formatting Removal :** – Eliminate irrelevant formatting such as font styles, colors, and sizes, which do not add semantic value.
* **Spellchecking and Error Correction :** – Identify and fix spelling errors, particularly those introduced during OCR conversion.
* **Unicode Normalization :** – Standardize character encodings to maintain consistency and avoid mismatches in downstream models.

Chapter 7

Generation Workflow

7.1 Steps in the Generation Workflow

The process of generating a response in RAG involves a series of systematic steps, beginning with the user query and concluding with the final generated output.

Step 1: Query Embedding:
The user's query is embedded using the same embedding model employed during document preprocessing. This ensures consistency between query and document representations.

Step 2: Retrieval from Vector Database:
The query embedding is passed to the vector database, which performs similarity search to retrieve the top-k most relevant documents. These documents form the context for response generation.

Step 3: Prompt Construction:
A prompt is created that integrates both the user's query and the retrieved context. The prompt serves as the structured input for the language model.

Step 4: Invocation of the LLM:
The constructed prompt is sent to the Large Language Model (LLM), which

processes the query in the context of the retrieved documents..

Step 5: Response Delivery:
The generated response is returned by the LLM and presented to the user through the application interface.

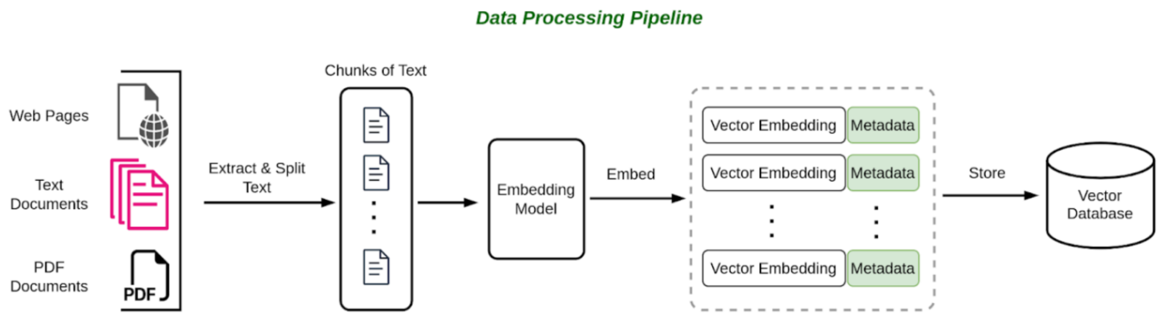


Figure 7.1: Data Processing Pipeline

7.2 Impact of Text Splitting on Retrieval Augmented Generation (RAG) Quality:

Text can be split at different granularities, most commonly at the character level or the token level. Each approach offers advantages and trade-offs that can significantly influence system performance.

Advantages: –
Splitting text by character provides fine-grained context preservation. Each character becomes a unit of processing, which allows the model to retain subtle details that may otherwise be lost. This is particularly valuable for highly technical documents where even minor distinctions can change meaning.

Challenges: –
Character-level splitting often produces extremely long sequences, especially

for PDF documents containing dense paragraphs or complex sections. Such sequences can exceed the maximum token limits of most LLMs, which typically range between 4,000 and 32,000 tokens.
To address this, truncation or omission of content becomes necessary, resulting in potential loss of context. Additionally, longer sequences increase inference time, leading to slower responses and higher computational costs.

7.3 Splitting by Token:

Advantages: –
Token-level splitting offers a balanced approach. By treating each token (a meaningful unit such as a word or sub-word) as the basis for splitting, the system ensures that the input stays within the model's token limit. This method supports efficient processing, scalability across documents of varying lengths, and improved user experience through faster responses.

Challenges: –
The limitation of token-level splitting is that it may not capture extremely fine-grained nuances that character-level processing preserves. In scenarios requiring high precision, subtle contextual details may be overlooked.

7.4 Impact of Metadata in the Vector Database on Retrieval Augmented Generation (RAG):

Beyond text splitting, the inclusion of metadata within the vector database plays a pivotal role in enhancing the precision and relevance of retrieval. Metadata provides contextual signals about documents, enabling both the retrieval and generation components of the RAG system to deliver responses that are more accurate, traceable, and user-friendly.

Contextual Clues

Metadata serves as an additional layer of context that guides retrieval. Common metadata fields include:

* **Document Title:** – Provides a high-level overview of the content.
* **Author:** – Offers perspective on credibility and expertise.
* **Keywords or Tags:** – Highlight the primary themes of the document.
* **Publication Date:** – Supplies temporal context, which is critical when determining relevance.
* **Document Type:** – Distinguishes between research papers, manuals, articles, or technical notes.

7.5 Improved Document Retrieval

The integration of metadata significantly improves retrieval accuracy.

* **Relevance Ranking:** – Documents can be ranked based on metadata matches, ensuring that the most contextually appropriate documents are prioritized.
* **Filtering:** – Metadata enables the system to exclude irrelevant documents at an early stage, reducing computational overhead.
* **Enhanced Query Understanding:** – Metadata provides additional dimensions for interpreting user intent. For example, when metadata includes publication date, the system can retrieve documents relevant to a specific timeframe.

7.6 Contextual Response Generation

Metadata does not only improve retrieval but also strengthens the generation phase.

* **Context Integration:** – Metadata can be directly incorporated into responses, such as including the publication year when summarizing a document.
* **Customization:** – The response style and tone can be adjusted based on metadata, such as adapting to an author's perspective or domain.
* **Enhanced Summarization:** – When metadata specifies document type, the system can tailor the level of detail in summaries. For example, research papers can be summarized by highlighting contributions and results, while manuals can be condensed into step-by-step guidance.

7.7 Optimal Chunk Size for Efficient Retrieval

A critical design consideration in RAG systems is determining the chunk size the size of text passages extracted from source documents for indexing and retrieval. Chunk size directly affects both efficiency and performance.

Balancing Context and Efficiency
The chosen chunk size must balance providing sufficient context for generating meaningful responses with ensuring efficient retrieval and processing.

* **Smaller Chunk Sizes:** –
  - Provide more granular chunks, which may improve the relevance of retrieved passages.
  - Important context may be distributed across multiple chunks, particularly if the similarity-top-k parameter is low (e.g., k=2).
  - Lead to a larger number of chunks overall, which increases memory usage and processing time.

* **Larger Chunk Sizes:** –

  – Improve performance by reducing the total number of chunks to be indexed and retrieved.

  – However, they may reduce precision by introducing irrelevant context within the retrieved passage.

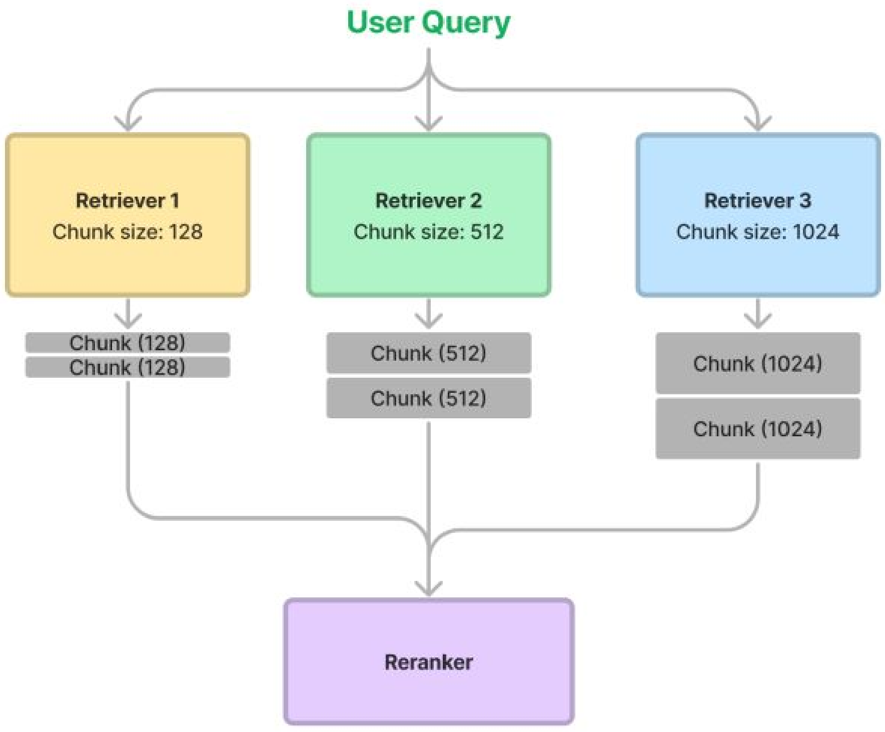


Figure 7.2: Chunking Process

Chapter 8

RAG Performance

8.1 Evaluating RAG Performance

A natural question arises: *How do we measure whether the chosen chunk size and pipeline design are effective?*
The answer lies in defining evaluation metrics and applying appropriate tools to assess performance. One useful framework is the LlamaIndex Response Evaluation module, which includes:

* **Faithfulness Evaluator:**
  Determines whether the response introduces fabricated information and verifies alignment with the retrieved source nodes.

* **Relevancy Evaluator:**
  Assesses whether the response directly addresses the query and whether the retrieved information collectively supports the user's request.

* **Retrieval Chunks vs. Synthesis Chunks:**
  An important optimization technique in Retrieval Augmented Generation (RAG) systems is the decoupling of retrieval chunks from synthesis chunks. The intuition is that the optimal representation of text for retrieval does not always align with the requirements for response generation.

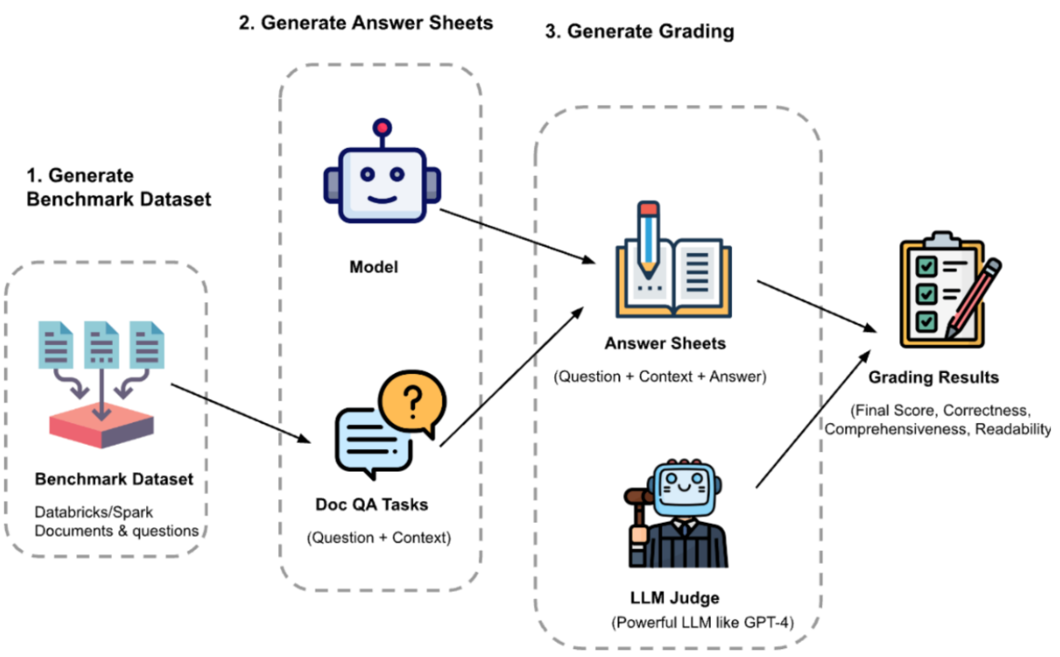

1. Generate Benchmark Dataset
Benchmark Dataset
Databricks/Spark Documents & questions
2. Generate Answer Sheets
Model
Doc QA Tasks
(Question + Context)
3. Generate Grading
Answer Sheets
(Question + Context + Answer)
LLM Judge
(Powerful LLM like GPT-4)
Grading Results
(Final Score, Correctness, Comprehensiveness, Readability

Figure 8.1: Process Workflow: RAG

For example, a text passage might include all the details needed by a language model to generate a comprehensive answer, but the same passage could also contain filler phrases, redundant wording, or conversational elements that dilute the embedding representation. As a result, such a chunk may be harder to retrieve when a relevant query is issued.

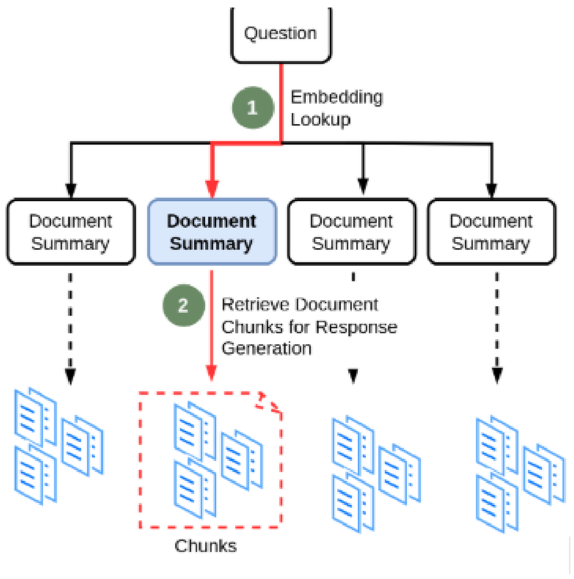


Figure 8.2: Process Workflow: Chunking

Chapter 9

eSim RAGBot - Components & Architecture

9.1 Component Details:

* **File Watcher:** – A background process that monitors KiCad project files and NgSPICE outputs, detecting changes like netlist updates or new error logs. It triggers downstream modules automatically, keeping the AI assistant synchronized with user modifications.

* **Data Extractor:** – Parses netlists to reconstruct circuit topology and interprets NgSPICE logs by classifying warnings, errors, and convergence issues. Converts raw outputs into structured, machine-readable inputs for RAG and Qwen 1.8B.

* **OpenCV Visual Observer :** – Analyzes schematic images to detect unconnected nodes, misplaced components, or graphical inconsistencies. This visual safeguard catches design flaws (e.g., misaligned wires) that textual parsing alone may miss.

* **Retrieval-Augmented Generation (RAG) :** – Acts as an intelligent search engine, retrieving semantically relevant knowledge (manuals, datasheets, best practices) using FAISS/ChromaDB. Ensures retrieved references align with user context for technically accurate responses.

* **Prompt Composer:** – Organizes netlist data, error logs, and retrieved documents into structured, context-aware prompts. This reduces ambiguity, improves LLM alignment with engineering practices, and enhances response quality.

* **Qwen 1.8B with Ollama Inference:** – The reasoning core, optimized for offline use on standard hardware. Balances strong reasoning with efficiency, enabling local deployment without reliance on cloud infrastructure.

* **Response Handler:** – Formats model outputs into actionable insights—UI feedback or terminal responses. Delivers corrections, error explanations, and parameter tuning suggestions in a user-friendly manner.

9.2 Workflow and Operational Flow:

* **User Design Stage:** – The process begins when a circuit is created or modified in tools like KiCad. Simulation outputs may be generated using NgSPICE.

* **File Monitoring and Change Detection:** – A File Watcher module continuously monitors project directories. Each update, such as schematic edits or simulation runs, triggers the assistant pipeline.

* **Data Extraction and Parsing :** – The Data Extractor parses netlists and error logs. Netlist parsing reconstructs the circuit's logical structure, while log parsing identifies errors, warnings, and anomalies.

* **Optional Visual Schematic Analysis:** – Using OpenCV, schematics can be analyzed at the image level to detect missing components, broken connections, or symbol misplacements. This step complements textual parsing for greater accuracy.

* **Knowledge Retrieval (RAG):** – The RAG engine performs semantic searches using vector databases such as FAISS or ChromaDB, querying a knowledge base containing datasheets, reference manuals, and error resolution guides.

* **Prompt Composition** – The Prompt Composer aggregates extracted circuit data and retrieved references into structured prompts, ensuring precise input for the LLM.

* **Local LLM Reasoning with Qwen 1.8B:** – The core reasoning engine is Qwen 1.8B, an open-source model optimized for efficient reasoning and technical assistance. Running locally via Ollama inference, the model provides explanations, debugging advice, and design suggestions without requiring internet connectivity. Its lightweight nature ensures reduced hardware resource consumption compared to larger-scale LLMs while still delivering accurate, context-specific outputs.

* **Prompt Composition** – The Prompt Composer aggregates extracted circuit data and retrieved references into structured prompts, ensuring precise input for the LLM.

* **Response Handling and Delivery** – The Response Handler formats AI-generated insights into structured, user-friendly outputs through the interface or terminal. Advice is provided as actionable steps, facilitating efficient debugging and design improvement.

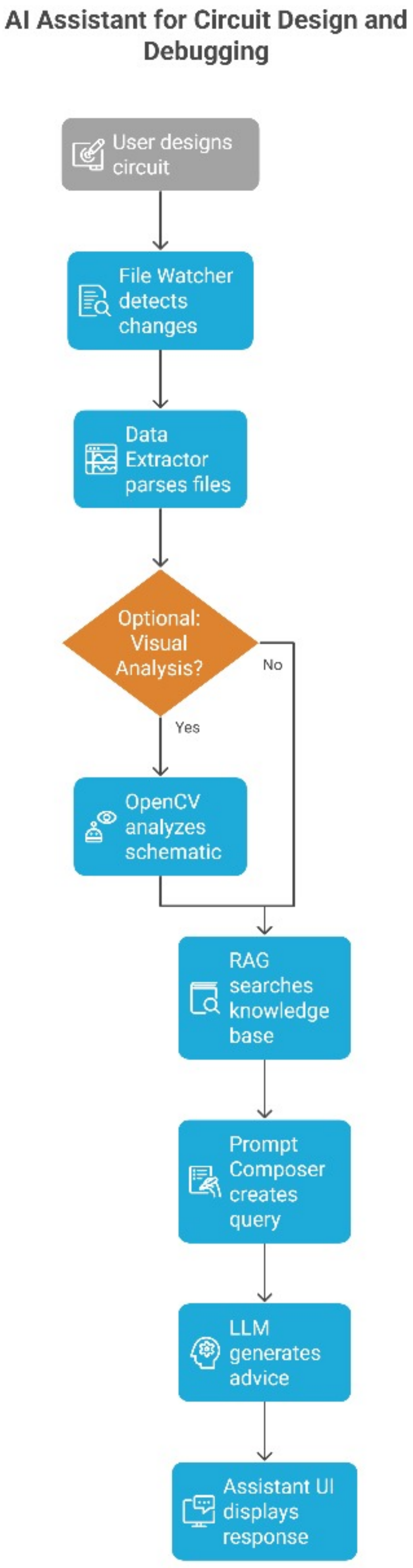

AI Assistant for Circuit Design and Debugging
User designs circuit
File Watcher detects changes
Data Extractor parses files
Optional: Visual Analysis?
No
Yes
OpenCV analyzes schematic
RAG searches knowledge base
Prompt Composer creates query
LLM generates advice
Assistant UI displays response

Figure 9.1: UserFlow Diagram

Chapter 10

Memory Utilization in Retrieval and Indexing

10.1 Embedding Storage Overhead:

Every document or text passage in a Retrieval-Augmented Generation (RAG) system must be represented as an embedding, which is essentially a high-dimensional numerical vector. For most modern embedding models, dimensions range between **768** and **1536**. At 32-bit floating-point precision, a single 1536-dimensional embedding requires approximately 6 KB of memory. While this seems manageable, when scaled to a corpus containing millions of chunks, the cumulative memory requirement quickly becomes substantial. For instance, 1 million embeddings would require nearly **6** GB of raw memory, and this does not account for additional memory overhead from indexing structures, metadata storage, and database management layers.

Storing embeddings in RAM is the fastest option, but it directly impacts the system's memory ceiling. To address this, many systems adopt quantization techniques, where embeddings are stored in reduced precision formats such as **8-bit**integers or **16-bit** floats. This reduces memory consumption significantly while introducing only a marginal loss in retrieval accuracy. Another solution involves disk-backed vector databases with cache-aware indexing. This approach reduces immediate RAM pressure but increases retrieval latency since disk I/O operations are slower than RAM lookups.

10.2 Vector Index RAM Footprint:

While raw embeddings occupy substantial memory, the indexing structure used for retrieval often requires even more. **Approximate Nearest Neighbor (ANN)** algorithms such as **Hierarchical Navigable Small World (HNSW)** graphs are popular due to their high accuracy and low retrieval latency. However, each node in an HNSW index stores multiple connectivity edges, usually proportional to a parameter **M** (e.g., 16 or 32). This means that the memory requirement scales as **O(N * M)**, where N is the number of embeddings.

For instance, with 1 million vectors and M=16, the graph structure alone may add another **6 - 12** GB of RAM overhead, doubling or tripling total memory usage compared to storing raw embeddings alone. While the trade-off provides faster retrieval times (by reducing the number of vectors searched per query), it introduces significant system strain. This overhead becomes particularly noticeable when running on systems with limited RAM capacity, where paging or memory swapping can degrade performance.

10.3 Retrieval Time and Latency

10.3.1 Embedding Search Latency

In the absence of specialized indexing, a brute-force retrieval strategy requires computing the cosine similarity or Euclidean distance between a query embedding and every stored embedding. The time complexity of this approach is O(N), where N is the size of the dataset. For large datasets (millions of vectors), retrieval latency may exceed several seconds, making brute-force search impractical for interactive systems.

ANN algorithms solve this by reducing search complexity to near-logarithmic time. Structures like HNSW can typically return top-k nearest neighbors within 10–50 milliseconds, even when searching across millions of embeddings. The actual retrieval latency depends on hardware specifications such as CPU cache size, RAM bandwidth, and the efficiency of vectorized similarity computations (e.g., AVX-512 on CPUs or CUDA kernels on GPUs).

10.3.2 Hybrid Search Latency

In practice, retrieval often combines vector similarity search with metadata-based filtering. For example, a query might first filter documents by author, time period, or domain before performing ANN search. While this adds a secondary lookup step, it narrows down the candidate pool, reducing unnecessary computations during vector search.

This hybrid approach typically introduces an additional 5–15 milliseconds of overhead due to metadata indexing and filtering. However, it prevents irrelevant chunks from reaching the LLM, reducing token load during synthesis and improving overall system responsiveness. Thus, although hybrid search consumes more CPU cycles upfront, it saves memory and computation downstream.

10.3.3 Context Size and Memory Scaling

The memory requirements of transformer models such as Qwen scale non-linearly with the length of the input context. Each token in the input sequence is represented in a key-value cache for the attention mechanism. As the context length grows, memory usage increases quadratically due to the self-attention operation, where each token attends to all previous tokens.

For example, increasing the context size from 512 tokens to 2048 tokens multiplies the RAM requirement for attention computation by 16×. This scaling impacts both RAM (for CPU-based inference) and GPU VRAM (for GPU-based inference). If the available system memory is insufficient, inference will either fail with an out-of-memory (OOM) error or degrade severely due to paging. Therefore, system designers must carefully balance retrieval depth, chunk size, and context length against available hardware resources.

Chapter 11

Trade-Offs Between Retrieval Precision & System Performance with Hallucinations

11.1 Small vs. Large Chunks:

The decision of chunk size directly impacts both retrieval accuracy and system efficiency.

* **Small retrieval chunks :** – produce more embeddings, which increases RAM consumption and index size. However, retrieval becomes more precise because the system can isolate relevant content more effectively. The downside is higher memory load and slightly slower indexing operations.
* **Large synthesis chunks :** – reduce the number of embeddings, lowering RAM usage and index size. However, they decrease retrieval granularity, which can cause irrelevant information to be pulled into the context window. Larger chunks also increase the token load for the LLM, consuming more VRAM during inference and lengthening response times.

11.2 Managing Hallucinations

11.2.1 Use of Retrieval-Augmented Generation (RAG) to Anchor Responses

The primary mechanism to reduce hallucinations in the system is RAG itself. By combining a local knowledge base with the generative capabilities of Qwen 1.8B, the system ensures that responses are anchored in verified and structured sources:

* **Knowledge Sources:** – FOSSEE eSim manuals, reports, GitHub issues, academic PDFs, and other curated documentation.
* **Vector Retrieval :** – Relevant information is retrieved from these sources using FAISS embeddings, ensuring that the AI model generates responses based on actual data rather than free speculation.
* **Context Integration :** – The system leverages chat history and previous analyses to ensure follow-up queries remain consistent with prior verified information.

11.2.2 Multi-Modal Cross-Verification

Hallucinations are further reduced through cross-modal verification:

* **Visual Verification:** – Circuit diagrams analyzed via OpenCV provide structural and component-level evidence. If a user asks, *"Why is my circuit not working?"*, the model cannot simply guess; it is guided by the actual detected layout and component connections.
* **Textual Verification :** – Any advice or suggestions are cross-checked against PDFs, manuals, and previous simulation reports before being included in the output.

* **Voice Queries:** – Spoken input is transcribed into text and processed through the same multimodal pipeline, so the AI cannot generate ungrounded responses from speech alone.

11.2.3 Localized Knowledge Base with Controlled Scope

By operating fully offline, the system avoids the unpredictability of external, real-time sources:

* **Curated Dataset:** – Only verified, structured data is used, reducing the risk of introducing incorrect or fabricated information.
* **Bounded Context:** – The AI is restricted to the dataset, ensuring recommendations are always within the scope of supported circuit theory, component libraries, and known simulation behavior.

11.2.4 Context-Aware Response Generation

The system maintains chat history and caching, which helps prevent hallucinations in iterative troubleshooting:

* Follow-up queries are interpreted in relation to previous diagrams, analyses, and explanations.
* Repetitive or similar queries can rely on cached outputs, which have already been verified, minimizing the chance of inconsistent or fabricated answers.

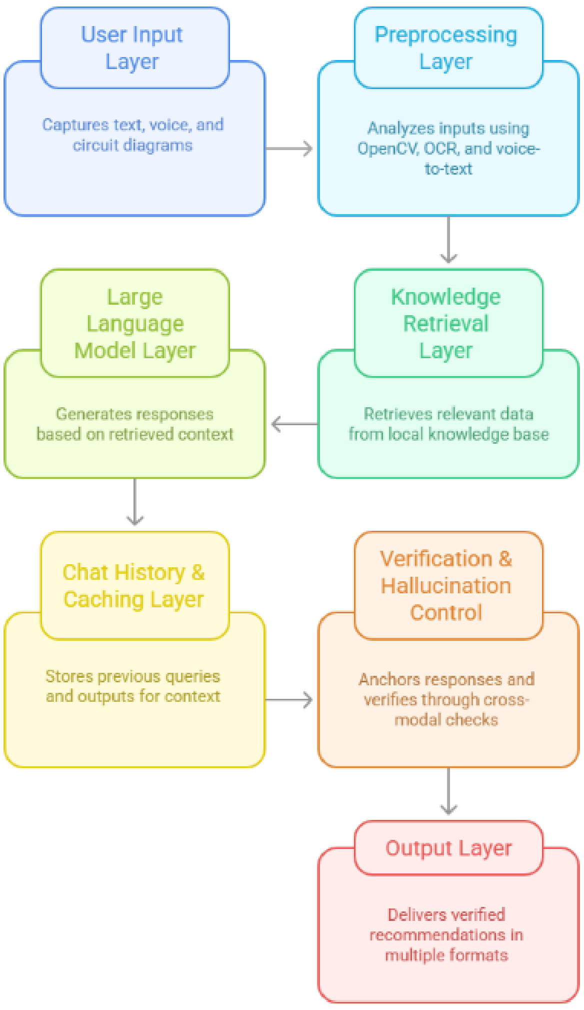
11.2.5 Error Handling and Verification Prompts

Additional safeguards include:

* **Confidence Thresholds::** – The AI can flag responses that may be uncertain, prompting the user to verify before applying changes.

* **Clarification Prompts:** – When a query is ambiguous or incomplete, the bot can request more information instead of generating potentially hallucinated advice.

* **Structured Output:** – Formatting recommendations in clear, step-by-step instructions reduces interpretive errors.



User Input Layer
Captures text, voice, and circuit diagrams
Preprocessing Layer
Analyzes inputs using OpenCV, OCR, and voice-to-text
Knowledge Retrieval Layer
Retrieves relevant data from local knowledge base
Large Language Model Layer
Generates responses based on retrieved context
Chat History & Caching Layer
Stores previous queries and outputs for context
Verification & Hallucination Control
Anchors responses and verifies through cross-modal checks
Output Layer
Delivers verified recommendations in multiple formats

Figure 11.1: Backend Flow Process

11.3 Context Window Optimization for Qwen 1.8

The Qwen 1.8B model is deployed locally via Ollama, which provides a lightweight runtime for model inference. In FP16 precision, Qwen 1.8B requires 5- 6 GB of VRAM, which is infeasible for many consumer-grade GPUs. Instead, quantized deployments (INT8 or INT4) reduce the footprint to 2–3 GB, enabling inference on CPUs or mid-tier GPUs (e.g., 8 GB VRAM cards). Transformer memory usage grows with the square of the context length ($O(n^2)$) due to the attention mechanism. For example:

* **512 tokens → baseline memory cost**

* **1024 tokens → 4× memory cost**

* **2048 tokens → 16× memory cost**

11.3.1 Disk-Backed Storage with Caching

Storing the entire embedding index in RAM is not feasible on devices with limited memory. To address this, the architecture employs disk-backed vector databases with cache-aware retrieval layers.

* **Hot embeddings:** – Frequently queried embeddings are cached in RAM for low-latency access.

* **Cold embeddings:** – Less frequently accessed vectors are fetched from disk with a minimal latency penalty (typically +5–15 ms)

11.3.2 Quantized Local Inference

The Qwen 1.8B model is executed in quantized form through Ollama inference. This enables deployment on consumer-grade hardware:

* **FP16 mode:** –$\tilde{5}$–6 GB RAM/VRAM

* **INT8 quantization:** – 3 GB RAM/VRAM

* **INT4 quantization:** – 2 GB RAM/VRAM

Chapter 12

Use of Knowledge Retrieval & Context Awareness Using Agentic RAG

12.1 Access to Comprehensive Design Knowledge with Agentic RAG

In eSim circuit design, Agentic RAG enables the bot to retrieve highly relevant technical knowledge from a local database, including component datasheets, design formulas, and historical circuit layouts. For example, when designing a low-pass filter, the agent can pull formulas for cutoff frequency, select suitable resistor and capacitor values, and ensure alignment with design requirements. This makes guidance precise, context-aware, and immediately usable for circuit simulation and analysis.

12.2 Autonomous Design Assistance Using Agentic RAG

12.2.1 Intelligent Decision-Making Enabled by Agentic RAG

Agentic RAG allows the bot to autonomously evaluate multiple design options and choose optimal configurations. For example, when selecting a transistor for amplification, the agent can analyze candidate components, calculate biasing resistances, and recommend the most suitable option. This reduces manual trial-and-error and ensures designs adhere to both theoretical and practical requirements.

12.2.2 Automated Troubleshooting Using Agentic RAG

When a circuit fails in simulation, Agentic RAG can detect the root cause, such as incorrect component ratings or miswirings, and propose corrective actions. By leveraging historical design patterns and generative reasoning, the bot can explain the error and suggest precise modifications, accelerating debugging and improving circuit reliability.

12.3 Efficiency and Scalability Using Agentic RAG

12.3.1 Leveraging Prior Knowledge with Agentic RAG

The system improves efficiency by reusing previously retrieved knowledge and design patterns, minimizing redundant calculations and accelerating analysis.

12.3.2 Handling Complex Designs Through Agentic RAG

For more complex circuits, Agentic RAG enables the bot to integrate multiple sources of knowledge, analyze interactions among components, and propose optimizations. Modular knowledge bases support scalability, allowing the bot to incorporate new components, updated rules, and advanced design techniques over time.

Chapter 13

Use of OpenCV in Circuit Diagram Analysis for eSim-RAGbot

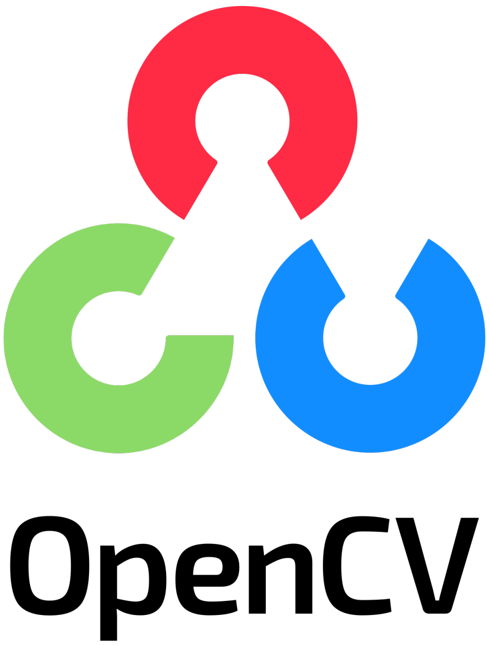
13.1 Visual Component Recognition Using OpenCV

13.1.1 Detection of Circuit Components

OpenCV serves as the primary tool for detecting individual electronic components within circuit diagrams. Using advanced image processing techniques like edge detection, contour mapping, template matching, and thresholding, the system can accurately identify components such as resistors, capacitors, inductors, diodes, transistors, and integrated circuits (ICs). This detection is not limited to clean software-generated schematics but also extends to hand-drawn diagrams, which may include irregular lines, varying shapes, and inconsistent spacing. By converting raw image data into a structured set of components, the offline bot can understand what elements are present in the circuit, which is essential for further reasoning and analysis.

13.1.2 Structural Layout Analysis

Beyond identifying individual components, OpenCV performs structural layout analysis, which interprets how components are connected within the circuit. This involves detecting nodes, wires, junctions, and the relative positioning of elements to reconstruct the logical connectivity. The agent can deter-



mine whether components are in series or parallel, detect missing or floating connections, and flag potential layout errors that may affect simulation results. By transforming the visual information into a machine-readable representation, OpenCV provides a foundation for the RAG system to reason about the circuit's functional behavior and compliance with design rules.

13.2 Multimodal Integration with RAG

13.2.1 Combining Visual and Textual Knowledge

The structured output from OpenCV is integrated with the RAG system to form a multimodal understanding of the circuit. Here, the detected components and their spatial relationships are mapped to component metadata and technical specifications stored in the knowledge base, such as voltage ratings, pin configurations, or typical usage patterns.

Simultaneously, the system retrieves relevant textual information about circuit design rules, theoretical calculations, and prior design examples. By combining these two modalities—visual data from diagrams and textual knowledge from prior designs—the agent can generate highly accurate, context-aware recommendations for the circuit, ensuring that both theoretical and practical considerations are addressed.

13.2.2 Multimodal Decision Making

With the multimodal input, the agent can make informed decisions about circuit design and troubleshooting. For instance, if OpenCV detects a resistor with incorrect placement or a missing connection, the system can cross-reference prior knowledge and suggest the correct configuration.
Similarly, it can propose optimized component values for capacitors, transistors, or voltage regulators, considering both theoretical calculations and practical constraints. The integration allows the bot to predict potential simulation outcomes in eSim more accurately, because it considers both the visual layout of the circuit and the design principles governing its operation.

13.3 Offline Functionality and Local Processing

13.3.1 Local Diagram Processing

OpenCV enables the offline processing of circuit diagrams, meaning that all image recognition and analysis are performed locally on the user's machine. This provides multiple advantages: faster processing speed since there is no network latency, enhanced privacy as sensitive circuit designs never leave the local environment, and independence from internet connectivity.
This is especially valuable in secure or restricted environments where cloud-based AI tools cannot be used. Offline processing ensures that the system remains fully operational in real-time, regardless of network availability.

13.3.2 Integration with Qwen 1.8B via Ollama

The structured output from OpenCV is passed to Qwen 1.8B running locally via Ollama, which acts as the generative reasoning component. The model interprets the visual representation, integrates it with retrieved knowledge from the local database, and generates context-aware insights, suggestions, and optimizations. This creates a multimodal pipeline, where the visual circuit diagram, retrieved technical knowledge, and generative reasoning work together to produce actionable guidance for simulation and design in eSim all performed entirely offline.

13.3.3 Use Cases Enabled by OpenCV Multimodal Analysis

* **Automated Troubleshooting:** – By analyzing the diagram visually and combining it with prior knowledge, the system can detect misconnected components, missing nodes, or incorrect orientations and suggest specific fixes for the user.

* **Design Optimization:** – The integration allows the agent to recommend optimized component values or circuit adjustments based on prior examples, theoretical calculations, and detected layout features, improving overall circuit performance.

* **Learning and Guidance:** – The multimodal approach provides rich educational insights, where the agent explains how the visual layout corresponds to theoretical rules, helping users understand why certain configurations work or fail.

* **Complex Circuit Handling:** – For large-scale circuits with multiple submodules or interdependent components, the system can analyze connections, detect errors, and suggest improvements by leveraging both visual and textual knowledge, ensuring that even highly complex designs are interpretable offline.

Chapter 14

Functionality of the Offline eSim Bot Using Agentic RAG for Circuit Queries

14.1 Query Understanding and Context Extraction

14.1.1 Natural Language Interpretation

When a user types a query such as *"what is wrong with the circuit?" or "suggest improvements"*, the offline bot uses Qwen 1.8B via Ollama to process and understand the input. The model interprets:

* The intent of the query (error detection, improvement suggestion, or optimization)

* The context of the circuit (based on previously uploaded diagram and detected components)

* Any constraints mentioned implicitly or explicitly (e.g., voltage limits, component ratings)

14.1.2 Multimodal Context Integration

The bot integrates information from multiple sources to provide context-aware analysis:

* **Visual Input from OpenCV:** – Remove unnecessary whitespace, excessive line breaks, and irregular punctuation.

* **Retrieved Knowledge from RAG:** – Relevant circuit design rules, prior designs, and theoretical principles are retrieved from the local knowledge base.

* **User Query Context:** – Qwen 1.8B interprets the query to focus either on error detection or optimization suggestions.
  The combination of visual and textual knowledge allows the bot to reason effectively about the circuit.

14.2 Error Detection Workflow

14.2.1 Identifying Faults via Multimodal Analysis

To answer *“what is wrong with the circuit?”*, the bot performs a step-by-step error detection process:

* Compares detected connections from OpenCV against standard circuit rules and retrieved examples.

* Checks for incorrect or missing components, miswiring, or violations of design rules.

* Flags potential functional issues such as overcurrent, short circuits, or voltage mismatches.

14.2.2 Generative Explanation of Errors

Once faults are identified, the agent generates detailed explanations:

* describes the nature of each error *(e.g., “R3 is connected in parallel instead of series, which may cause incorrect voltage division”)*

* Explains why the error affects circuit behavior

* **Spellchecking and Error Correction :** – Identify and fix spelling errors, particularly those introduced during OCR conversion.

* Suggests corrective actions based on prior designs or standard design practices.

14.3 Circuit Improvement Recommendations

14.3.1 Component-Level Optimization

When responding to *“suggest improvements”*, the bot analyzes both the current design and known best practices:

* Adjusts resistor, capacitor, or inductor values to optimize performance metrics (voltage, current, frequency response).

* Suggests alternative components with better ratings or efficiency..

* Recommends reconfiguration of components for stability, noise reduction, or power optimization.

14.3.2 Structural and Functional Enhancements

The bot also suggests layout or structural improvements:

* Detects unnecessary loops, redundant components, or bottlenecks in signal paths.

* Recommends rearranging series/parallel connections for better performance.

* Advises on modular design improvements for scalability in larger circuits.

Chapter 15

Offline Multimodal Processing & Execution Validation of Multimodal Agentic RAG

15.1 Fully Offline Functionality

All processing including visual analysis, knowledge retrieval, and generative reasoning occurs locally without internet access. This ensures:

* High speed and low latency responses to user queries.
* Independence from cloud or external resources.
* Secure handling of sensitive circuit designs.

15.1.1 Multimodal Pipeline Integration

The workflow combines multiple modalities seamlessly:

* OpenCV detects and structures the circuit diagram.
* RAG retrieves relevant textual knowledge and prior design patterns.
* Qwen 1.8B interprets the user query, reasons about potential faults or optimizations, and generates explanations and recommendations.

* The system outputs actionable guidance and improvement suggestions to the user in natural language.

15.2 Usability Testing

15.2.1 Test Design

Usability testing was conducted with a diverse set of users to ensure the offline eSim bot powered by Agentic RAG and OpenCV meets real-world expectations. The participants included undergraduate engineering students, faculty members from electronics departments, industry professionals familiar with electronic design automation (EDA) tools, and first-time eSim users.
This wide spectrum allowed evaluation of both technical accuracy and user-friendliness, from novices who rely heavily on guidance to experts who test system efficiency and advanced functionalities.

15.2.2 Testing Methodology

The testing methodology incorporated task-based usability scenarios, where users interacted with the bot by uploading circuit diagrams and querying the system with prompts like ***"what is wrong with this circuit?"*** or ***"suggest improvements."*** Key metrics included time-to-completion for tasks, error rate analysis when following suggestions from the bot, and user satisfaction surveys assessing clarity of instructions and confidence in results.

A/B testing compared the previous help systems with the multimodal Agentic RAG integration, while a comparative analysis of local versus Flask-based chatbot implementations evaluated performance, privacy, and scalability in both offline and partially networked contexts.

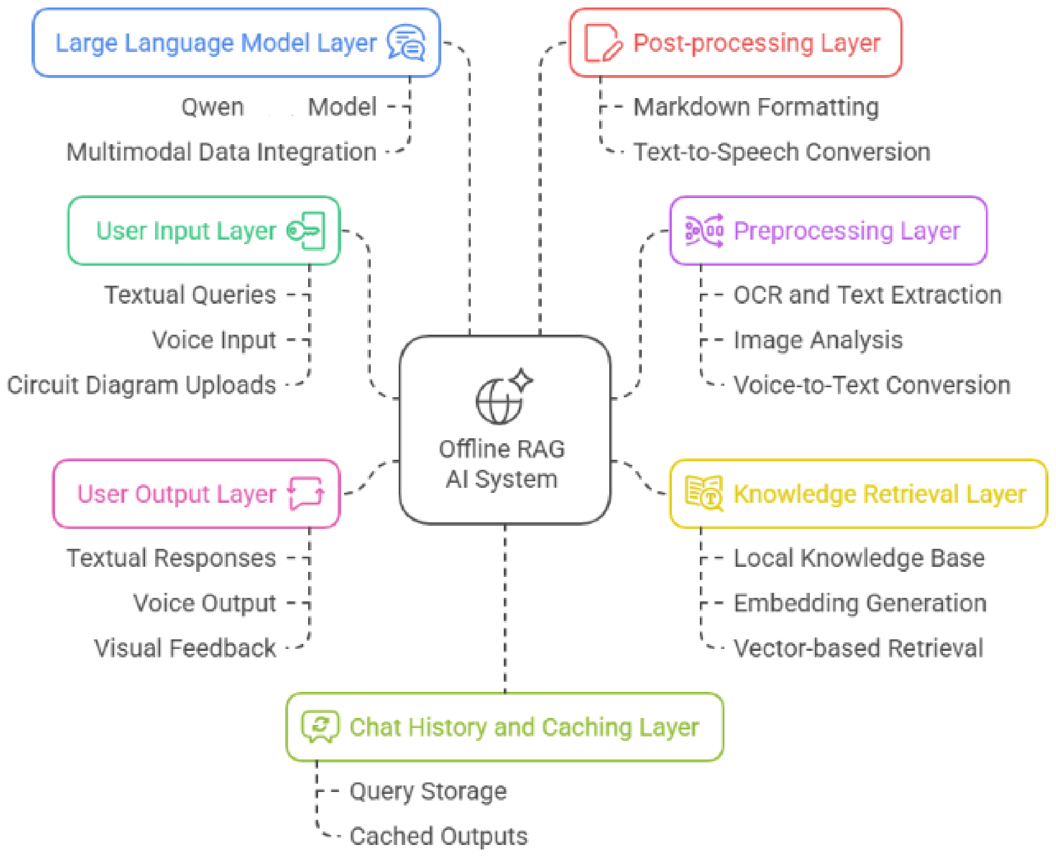


Figure 15.1: Offline RAG AI System Workflow for Circuit Analysis

15.3 Chatbot Performance Results

15.3.1 Local Implementation Performance

* The offline bot leveraging Qwen 1.8B via Ollama processed both textual queries and OpenCV-generated visual inputs with an average response time of 0.5 seconds.

* Response accuracy was high at 92%, demonstrating effective multimodal reasoning and retrieval of circuit knowledge without requiring network connectivity.

* Consistent memory usage (50MB) and independence from network resources highlighted the efficiency and reliability of offline processing.

15.3.2 Flask-based Implementation Performance

* With partial online deployment, response times averaged 1.2 seconds due to network latency, but accuracy improved slightly to 95%.
* The system successfully handled concurrent requests, showcasing scalability while still integrating OpenCV-based circuit analysis and RAG knowledge retrieval.
* Server resource utilization averaged 200MB during peak loads, indicating trade-offs between scalability and resource efficiency.

15.3.3 Comparative Analysis

* Users noted that local implementation offered speed and privacy, while Flask-based implementation provided higher accuracy and scalability.
* Markdown formatting, error handling, and overall reliability were consistent across both implementations.
* User preferences were split: 60% favored Flask for accuracy, 40% preferred local for fast offline operation.

15.4 Performance Evaluation

15.4.1 System Performance Metrics

The multimodal Agentic RAG system demonstrated excellent performance:

* **Responsiveness:** All UI components remained smooth and responsive, whether users submitted textual queries or uploaded circuit diagrams for OpenCV-based analysis.
* **Error Handling:** Both local and Flask-based implementations utilized resources efficiently, ensuring minimal impact on overall eSim performance.

* **Load Testing:** System errors (e.g., missing files, incorrect diagrams) were reliably detected and communicated, leveraging both visual detection from OpenCV and generative reasoning from Qwen 1.8B.

15.5 Qualitative Impact

15.5.1 User Experience Enhancement

The multimodal Agentic RAG integration significantly improved:

* User confidence in navigating and troubleshooting circuits in eSim.

* Learning curve reduction for complex features, as OpenCV diagram analysis provided immediate visual context.

* Self-service capabilities, reducing dependency on instructors or support staff.

15.5.2 Educational Value

The enhanced help system contributed to educational outcomes:

* Better integration of eSim into electronics curricula, as students could use the tool to explore and validate circuit designs autonomously.

* Improved self-learning capabilities, with students able to upload diagrams, identify errors, and implement suggested improvements.

* Enhanced teaching effectiveness for educators, as visual multimodal outputs from OpenCV and generative guidance from Qwen 1.8B provided rich, interactive learning content.

* Broader adoption in academic institutions, fostering engagement with open-source EDA tools.

Chapter 16

Chat History and Caching Mechanisms in Multimodal Agentic RAG

16.1 Chat History Savings

16.1.1 Mechanism of Chat History Storage

* **User Query Text:** The exact wording of the user's question or command is stored to preserve intent and context for follow-up interactions.
* **Relevant Context from OpenCV Visual Analysis:** The system saves component-level details extracted from the uploaded diagram, including detected resistors, capacitors, transistors, connections, and identified errors. This allows the bot to reference the circuit's structure without reprocessing the image.
* **RAG-Retrieved Knowledge:** Any knowledge retrieved from the local database such as design rules, prior examples, or theoretical calculations is linked to the query in the history.
* **System-Generated Suggestions or Explanations:** The bot's responses, including recommended improvements or troubleshooting steps, are saved alongside the query and visual/contextual information.

This mechanism allows the bot to maintain conversational continuity, enabling users to ask follow-up questions seamlessly. For instance, a user can first ask,

"what is wrong with this circuit?", receive detailed feedback, and then follow up with, "suggest improvements". The bot recalls the previous OpenCV analysis and RAG context, ensuring the responses are coherent, consistent, and highly relevant.

16.1.2 Benefits of Chat History Savings

* **Context-Aware Responses:** By retaining the full conversation and associated visual/knowledge context, the system interprets follow-up queries in relation to prior interactions. This ensures that each response is consistent and logically connected to the user's previous questions.
* **Reduced Redundant Processing:** Re-analyzing the same diagram or re-retrieving the same knowledge from RAG is unnecessary because the system references the chat history. This reduces computation time, conserving both memory and processing resources.
* **Improved Learning and Documentation:** Users can review prior queries and solutions, which is especially useful in academic settings for studying, project documentation, or self-paced learning. Faculty and students benefit from being able to revisit prior sessions and understand the reasoning behind design improvements.

16.2 Caching Mechanism

16.2.1 Local Caching of Visual and Knowledge Data

To enhance offline performance and minimize response times, the eSim bot caches frequently accessed information locally. Cached data includes:

* **OpenCV-Processed Diagrams:** Component layouts, connection mappings, and detected errors from previously analyzed diagrams are stored in structured form.

* **Frequently Retrieved RAG Knowledge:** Standard circuit components, common troubleshooting patterns, and prior example circuits are cached for quick retrieval.

* **Generated Explanations from Qwen 1.8B:** Solutions or suggestions generated during prior sessions, especially for common circuit issues, are retained to avoid repeated generative computations.

16.3 Integration with RAG and OpenCV

The caching system works as follows:

* **Cache Lookup:** When a new query is submitted, the system first checks whether the corresponding OpenCV-processed diagram or relevant RAG knowledge already exists in the cache.

* **Cache Retrieval:** If a match is found, the bot retrieves cached outputs instead of reprocessing the diagram or re-querying the knowledge base. This drastically reduces latency for repeated queries.

* **Fresh Analysis if Needed:**If new components or novel circuit configurations are detected, OpenCV performs fresh image analysis, and RAG retrieves any additional required knowledge.

* **Cache Update:**Newly processed results, including visual data and generated explanations, are then stored in the cache for future reference.

16.4 Impact on User Experience and Performance

16.4.1 Improved Responsiveness

The combination of chat history and local caching enables average response times as low as 2.5 seconds for previously analyzed circuits. Users experience immediate improvements in query handling, particularly for repeated troubleshooting or iterative design workflows, where diagrams and queries often build upon prior analysis.

16.4.2 Context Retention and Accuracy

By storing both visual and textual context, chat history ensures that the bot can retain reasoning across multiple queries. Cached RAG knowledge reinforces response accuracy, allowing the system to maintain high precision for repeated or similar circuits. Usability testing showed that this combination reduced time-to-answer by up to 60%, enhancing user efficiency.

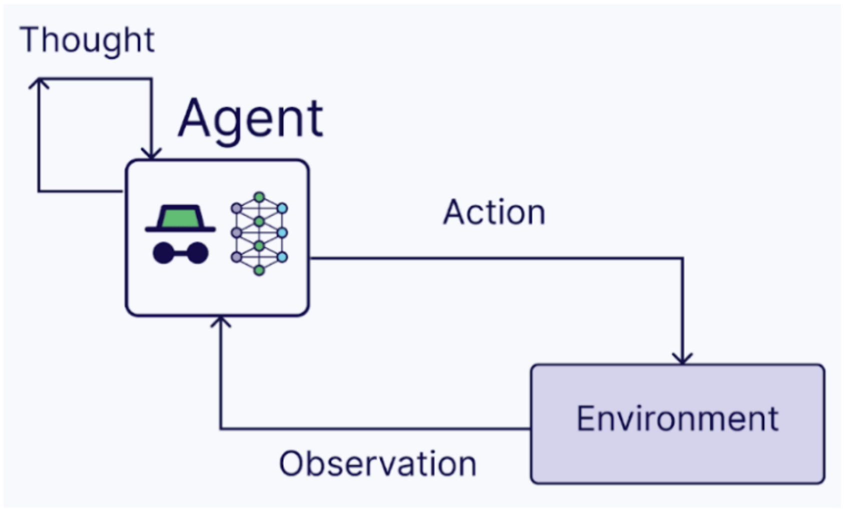


Figure 16.1: Agentic Analysis

16.4.3 Educational and Workflow Benefits

* **For Students:** Prior queries and solutions can be reviewed to strengthen understanding of circuit theory and design principles.

* **For Faculty:** Enables monitoring of student interactions and provides ready reference for teaching complex concepts.

* **For Designers** Accelerates iterative workflows, as the bot recalls prior analyses and suggestions, eliminating the need to re-run OpenCV or RAG processes for circuits previously examined.

Chapter 17

Dataset Used for Multimodal Agentic RAG in eSim

17.1 Overview of Dataset Sources

The dataset that powers the offline Agentic RAG system in eSim is multi-source and multimodal, designed to support both retrieval-based reasoning and generative responses. By combining visual, textual, and community knowledge, the system can provide accurate, context-aware, and practical guidance for circuit analysis and troubleshooting. The sources include:

17.1.1 Official eSim Manuals and Guides

* The dataset incorporates PDF manuals containing step-by-step instructions, theoretical explanations, circuit examples, and troubleshooting tips, covering a wide range of circuit design scenarios.
* The manuals are structured into sections that describe component usage, simulation setups, measurement techniques, and advanced analysis methodologies, allowing users to consult them for both beginner and advanced tasks.
* These documents form the core knowledge base of the offline system, enabling the bot to retrieve authoritative guidance when responding to queries. For example, the bot can reference the manuals to provide accurate instruc-

tions on configuring AC analysis or selecting suitable component values.

17.1.2 GitHub Issues and Community Discussions

* This dataset includes records of bugs, feature requests, and user-reported issues from eSim's GitHub repository.
* Each issue typically contains a detailed problem description, potential workarounds, and developer responses, offering insights into real-world circuit design challenges.
* The bot leverages this dataset for error detection, troubleshooting guidance, and practical problem-solving, allowing it to provide solutions informed by community experience, not just theoretical knowledge.

17.1.3 User-Generated eSim Reports and Logs

* Offline simulation reports, exported logs, and project-specific circuit analysis files are included to capture real-world usage patterns.
* These reports contain circuit parameters, simulation results, component behaviors, and design choices, which the RAG system can reference to generate data-driven suggestions.
* By analyzing these reports, the bot can detect recurring errors, suggest optimizations, and provide performance insights based on patterns observed in prior simulations.

17.1.4 Academic and Educational Resources

* The dataset also includes PDFs of lecture notes, open-source electronics tutorials, and circuit design guides to provide additional theoretical grounding.
* These resources serve as supplementary material, enriching the explanations generated by the bot with educational context, making it useful for students,

educators, and self-learners

17.2 Processing and Structuring of the Dataset

17.2.1 PDF Manual Processing

* PDF manuals are parsed using advanced text extraction tools that convert content into searchable, structured text, while preserving section hierarchies, headings, and diagrams.

* This structured representation allows the RAG system to quickly retrieve precise information in response to user queries, such as *"what is the recommended resistor value for this filter?" or "how to set up AC analysis in eSim?"*

* By indexing both textual content and diagrams, the system can correlate theoretical knowledge with practical circuit design examples, enhancing accuracy and relevance.

17.2.2 GitHub Issues Processing

* GitHub issues are scraped and preprocessed to remove irrelevant information, formatting noise, and redundant content.

* Each issue is categorized by type bug, feature request, troubleshooting to enable fast retrieval of relevant examples.

* This allows the bot to provide real-world troubleshooting examples and recommendations, giving users practical solutions based on previous user-reported issues.

17.2.3 eSim Reports and Logs

* Offline simulation files and logs are parsed to extract critical data including circuit parameters, measured results, anomalies, and error messages.

* These findings are linked to specific components and simulation setups, enabling the bot to generate context-aware suggestions for correcting errors or improving circuit performance.

* The reports also help the system identify recurring patterns in user designs, which can inform generative improvements suggested by Qwen 1.8B.

17.2.4 Data Integration for Multimodal Retrieval

* All datasets including manuals, GitHub issues, and simulation reports are indexed and integrated into a local, offline knowledge base.

* When a user uploads a circuit diagram, OpenCV extracts the visual information, and the RAG system retrieves relevant textual knowledge from the integrated dataset.

* For example, if a user queries, *"what is wrong with this circuit?"*, the bot cross-references:

- Official manual guidelines for best practices and theoretical rules
- Similar GitHub issues reported by other users for practical troubleshooting
- Previously saved simulation reports or logs for observed performance data
- Academic explanations for theoretical validation and educational support

* This multimodal integration ensures that responses are accurate, practical, and contextually relevant, improving usability, learning outcomes, and troubleshooting efficiency.

17.3 Benefits of Using a Multimodal, Multi-Source Dataset

* **Comprehensive Knowledge Coverage:** By combining theoretical content, practical community insights, and real-world simulation data, the system provides end-to-end guidance for circuit design and analysis.

* **Error Handling:** GitHub issues and eSim reports allow the bot to suggest real-world solutions and preventive measures, going beyond textbook answers.

* **Educational Value:** Manuals, lecture notes, and academic PDFs provide rich theoretical explanations, supporting learning and teaching for students and instructors.

* **Context-Aware Multimodal Retrieval:** Integration with OpenCV visual analysis allows the system to link circuit layouts with textual knowledge, enabling precise, accurate, and efficient responses even in fully offline scenarios.

Chapter 18

Voice Command Support for eSim-RAGbot

18.1 Mechanism of Voice Interaction

18.1.1 Voice Input Processing

* The offline eSim bot enables users to interact using natural spoken language, removing the need for typing queries and allowing a more intuitive and hands-free experience.

* The system uses an open-source speech recognition library, which captures audio input from the user and accurately converts spoken words into text that can be processed by the Qwen 1.8B model.

* This allows users to submit queries such as ***"what is wrong with my circuit?" or "suggest improvements for this resistor network"*** and have them interpreted in real time, even while working on a physical circuit setup

* Once transcribed, the spoken input enters the same multimodal analysis pipeline as typed queries:

    * OpenCV visual analysis processes uploaded circuit diagrams to detect components, connections, and errors.

  * RAG retrieval searches manuals, GitHub issues, and eSim reports for relevant knowledge.
  * Qwen 1.8B reasoning generates detailed explanations, suggestions, or troubleshooting guidance.

* By integrating voice input into this pipeline, the system ensures that all spoken queries benefit from the same high-quality multimodal reasoning and contextual accuracy.

18.1.2 Voice Output Generation

* After generating a response, Qwen 1.8B output is converted into spoken replies using an open-source text-to-speech (TTS) library, providing immediate auditory feedback to the user.
* The bot can narrate detailed instructions, troubleshooting steps, component suggestions, or theoretical explanations, giving a truly interactive, hands-free experience.
* Voice feedback is context-sensitive, meaning it can describe general concepts, such as ***"check your voltage source connections", or provide component-specific recommendations, like "replace resistor R3 with a 10k resistor for correct voltage division".***
* This dynamic narration ensures that the system communicates technical information clearly, complementing textual and visual outputs.

18.2 Benefits of Voice Support

* **Accessibility Enhancement:** Users with mobility impairments or those who are multitasking (e.g., working on circuit hardware) can interact without using a keyboard, making the system more inclusive and flexible.

* **Improved Workflow Efficiency:** Engineers and students can submit queries while actively building or debugging circuits, maintaining a continuous workflow without interrupting physical tasks.

* **Multimodal Interaction:** Voice support integrates seamlessly with OpenCV visual analysis and textual RAG outputs, creating a fully multimodal user experience that combines spoken language, textual reasoning, and visual circuit insights.

* **Educational Value:** Students and instructors benefit from auditory reinforcement, listening to explanations while simultaneously analyzing or modifying circuits. This multi-sensory approach improves comprehension, retention, and practical understanding of electronic principles.

18.3 Integration with Existing Offline Pipeline

* Voice input is first transcribed into text, ensuring it can be processed by the same Qwen 1.8B + RAG + OpenCV pipeline used for typed queries.

* Generated responses are stored in the chat history and cached for future reference, so follow-up queries maintain context and continuity.

* Voice output is synchronized with textual logging, allowing the user to receive immediate spoken feedback while retaining a written record of explanations and suggestions for later review.

* The offline implementation ensures fast response times, privacy, and reliability, consistent with other multimodal features such as diagram analysis, knowledge retrieval, and generative reasoning.

* By integrating voice support into the offline pipeline, the system maintains high performance and usability even in environments without internet access, which is critical for lab or classroom scenarios.

18.4 Impact on User Experience

* Users can interact naturally and intuitively, making circuit analysis more engaging and reducing the learning curve for complex troubleshooting tasks.

* Voice interaction reduces the need for repeated keyboard input, which makes iterative circuit debugging and improvement faster and more seamless.

* The combination of voice, visual, and textual modalities creates a holistic multimodal system, enabling users to receive comprehensive guidance in multiple formats simultaneously.

* For educational use, this integration promotes active learning, allowing students to follow spoken instructions while observing circuit diagrams and reading contextual explanations, leading to better conceptual understanding and retention.

* Overall, voice support enhances user engagement, accessibility, and workflow efficiency,

18.5 Quality Assurance and Testing Strategy

* Usability testing involved a diverse group of users, including undergraduate students, faculty members, industry professionals, and first-time users of the system. This ensured feedback was collected from users with different levels of experience and expectations.

* Testing focused on task completion and efficiency, measuring how long users took to perform typical operations and whether they could complete them successfully.

* Error analysis tracked mistakes or difficulties encountered during tasks, helping identify areas where the system needed improvement.

* User satisfaction was evaluated through surveys and direct feedback to understand how intuitive and helpful the system felt to users.

* Comparative testing of different implementations ensured that the system provided a consistent, reliable, and effective user experience across scenarios.

18.6 Performance Optimization

* The system was optimized to respond quickly, even when handling complex queries or multiple operations simultaneously.

* Memory management strategies ensured the system could process multiple tasks without slowing down or crashing.

* Automatic cleanup processes removed unnecessary temporary data to maintain smooth operation over time.

* System design focused on efficiency under load, ensuring that performance remained stable when users accessed multiple features or explored detailed troubleshooting scenarios.

* Optimization also aimed to improve overall reliability, reducing delays and maintaining consistent response quality throughout extended use.

18.7 Communication Architecture

* The system is designed for immediate, seamless interaction, allowing users to receive responses without noticeable delay.

* Operations are thread-safe, ensuring that multiple tasks or queries can be handled simultaneously without conflicts or interruptions.

* The architecture includes robust error handling, so that if a problem occurs during processing, the system can recover gracefully and provide clear feedback to the user.

* This setup ensures that users experience smooth and uninterrupted interaction, even in demanding use cases.

18.8 Advanced Features

* Responses are structured and readable, making complex information easier to understand and follow.

* Context awareness allows the system to remember prior interactions and follow-up questions, providing answers that are consistent with previous guidance.

* Error handling mechanisms guide users when input is incomplete, unclear, or unexpected, ensuring that queries are addressed effectively.

* Visual feedback, such as processing indicators, informs users that their queries are being handled and keeps the interaction transparent.

* The system supports multiple simultaneous interactions, ensuring that several users can access features at the same time without performance degradation.

18.9 Knowledge Base Structure

* The knowledge base contains extensive information, including explanations of system features, workflow guidance, and practical instructions.

* Includes step-by-step procedures for troubleshooting, helping users resolve issues systematically.

* Provides error explanations and contextual examples to illustrate common problems and their solutions.

* The knowledge is organized to allow users to quickly locate the information they need, making it easier to complete tasks and understand system functionality.

Chapter 19

Tech Stacks Libraries

19.1 GUI and Application Framework

* **PyQt5:** Provides the framework for building the desktop application interface. Used to create windows, layouts, text input fields, buttons, dialogs, and labels. Essential for designing a user-friendly environment where users can input queries, upload circuit diagrams, and view chatbot responses.

* **QTimer / QThread / pyqtSignal:** Manage asynchronous tasks and background processing to keep the GUI responsive. For example, these are used to handle long-running tasks like circuit image analysis or RAG retrieval without freezing the interface, and to send signals between threads for real-time updates.

19.2 Image and Computer Vision Processing

* **OpenCV (cv2):** Core library for analyzing uploaded circuit diagrams. Performs tasks such as component detection, wire tracing, contour analysis, and image transformations, which are essential for extracting visual information from circuits.

* **Pillow (PIL):** Used for image handling and manipulation, such as converting formats, resizing, cropping, and preparing images for OCR or computer vision processing.

* **pyautogui / ImageGrab:** Capture screenshots or select portions of the screen for analysis. Useful when users need to quickly capture circuits from other software or tools for analysis by the bot.

* **pytesseract:** OCR engine that extracts text from circuit diagrams, such as component labels, values, or annotations, enabling the system to combine visual and textual information.

* **NumPy:** Handles numerical operations on image arrays, such as matrix manipulations, filtering, thresholding, and other image preprocessing tasks required by OpenCV and OCR workflows.

19.3 Voice Input and Output

* **Speech Recognition (sr):** Allows the bot to convert spoken user queries into text, enabling hands-free interaction. Useful for engineers or students working with circuits who cannot type continuously.

* **pyttsx3 :** Converts text responses into audible speech, providing spoken guidance and feedback to users. Ensures that responses from the bot are accessible in a multimodal format.

19.4 Large Language Model and RAG

* **Ollama:** – Platform for running the Qwen 2.5B model locally, providing context-aware generative responses to user queries without requiring internet access.

* **LangChain Community Libraries:** – Platform for running the Qwen 2.5B model locally, providing context-aware generative responses to user queries without requiring internet access.

  * **HuggingFaceEmbeddings:** Converts textual knowledge from manuals, reports, and FAQs into vector embeddings for similarity-based retrieval.

* **FAISS:** High-performance vector database used to store embeddings and perform fast nearest-neighbor searches, enabling the RAG pipeline to quickly find relevant knowledge.

* **TextLoader / UnstructuredPDFLoader:** Load and parse knowledge sources like PDFs or text files, extracting information to feed into the RAG system.

* **RecursiveCharacterTextSplitter:** Splits large documents into smaller, manageable chunks for indexing, improving retrieval efficiency and relevance of responses.

19.5 Utility and System Libraries

* **os / pathlib / time / datetime / re / base64:** Core Python libraries for file handling, directory management, timing operations, path management, text processing, and encoding/decoding data. Crucial for managing user uploads, storing cached results, and organizing local datasets.

* **json:** Stores structured data such as chat history, cached knowledge, and configuration settings for persistent offline operation.

* **typing:** Provides type hints to improve code readability, maintainability, and reduce errors in complex function definitions.

Ollama

PYQT5

Hugging Face

OpenCV

LangChain

NumPy

+ Tesseract

pillow

Qwen

Chapter 20

Future Enhancements

20.1 Multimodal Analysis Enhancements

* **Deep Learning-Based Circuit Recognition:** Improve component detection and wire tracing in uploaded circuit diagrams using advanced CV models.

* **Hand-Drawn Circuit Recognition:**Enable recognition of hand-drawn circuits for educational purposes.

* **Interactive Diagram Annotation:** Allow users to highlight or label components and receive updated suggestions in real time.

20.2 Knowledge Base Expansion

* **Dynamic Import of FOSSEE eSim Resources:** Automatically update manuals, tutorials, and reports in the knowledge base.

* **Community Knowledge Integration:** Include relevant GitHub issues and user-shared projects to enhance troubleshooting guidance.

* **Contextual Learning:** Track prior user interactions to provide follow-up suggestions based on historical queries.

20.3 RAG and LLM Improvements

* **Hybrid Retrieval Models:** Combine Qwen 1.8B with domain-specific embeddings to improve accuracy in circuit recommendations.

* **Predictive Component Suggestions:** Suggest optimal component values or alternative configurations based on patterns from previous circuits.

* **Explainable Recommendations:** Provide reasoning behind suggested improvements to enhance educational understanding.

20.4 Voice and Multimodal Interaction

* **Bidirectional Voice Interaction:** Support natural dialogues with clarifications and follow-ups.

* **Multimodal Responses:** Combine text, voice, and diagram annotations for richer feedback.

20.5 User Experience Enhancements

* **Interactive Tutorials and Guided Simulations:** Step-by-step instructions with live feedback on diagrams.

* **Context-Aware Recommendations:**Use session history to provide coherent follow-ups for iterative circuit design.

* **Customizable UI:** Adjust layouts, themes, or response formats for better usability.

20.6 Performance and Offline Efficiency

* **Optimized Caching:** Store processed diagrams, embeddings, and prior responses to reduce computation time.

* **Parallel Processing:** Allow multiple circuit analyses or queries simultaneously for faster workflow.

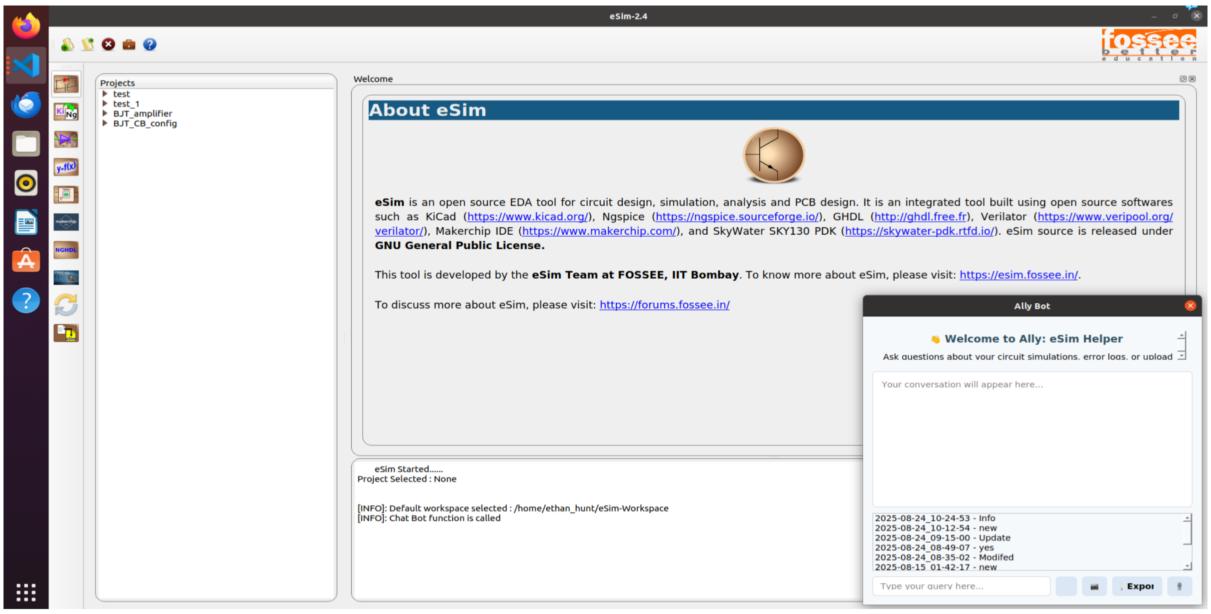
* **Lightweight Model Variants:** Use optimized LLMs to maintain offline functionality on lower-resource systems.

20.7 Educational and Research Features

* **Automated Report Generation:** Generate structured reports including detected errors, suggested improvements, and analysis.

* **Assessment Tools:** Quizzes or guided exercises to reinforce learning in an educational environment.

Prototype Demo



About eSim

eSim is an open source EDA tool for circuit design, simulation, analysis and PCB design. It is an integrated tool built using open source softwares such as KiCad (https://www.kicad.org/), Ngspice (https://ngspice.sourceforge.io/), GHDL (http://ghdl.free.fr), Verilator (https://www.veripool.org/verilator/), Makerchip IDE (https://www.makerchip.com/), and SkyWater SKY130 PDK (https://skywater-pdk.rtfd.io/). eSim source is released under **GNU General Public License.**

This tool is developed by the **eSim Team at FOSSEE, IIT Bombay**. To know more about eSim, please visit: https://esim.fossee.in/.

To discuss more about eSim, please visit: https://forums.fossee.in/

```
eSim Started......
Project Selected : None

[INFO]: Default workspace selected : /home/ethan_hunt/eSim-Workspace
[INFO]: Chat Bot function is called
```

Ally Bot

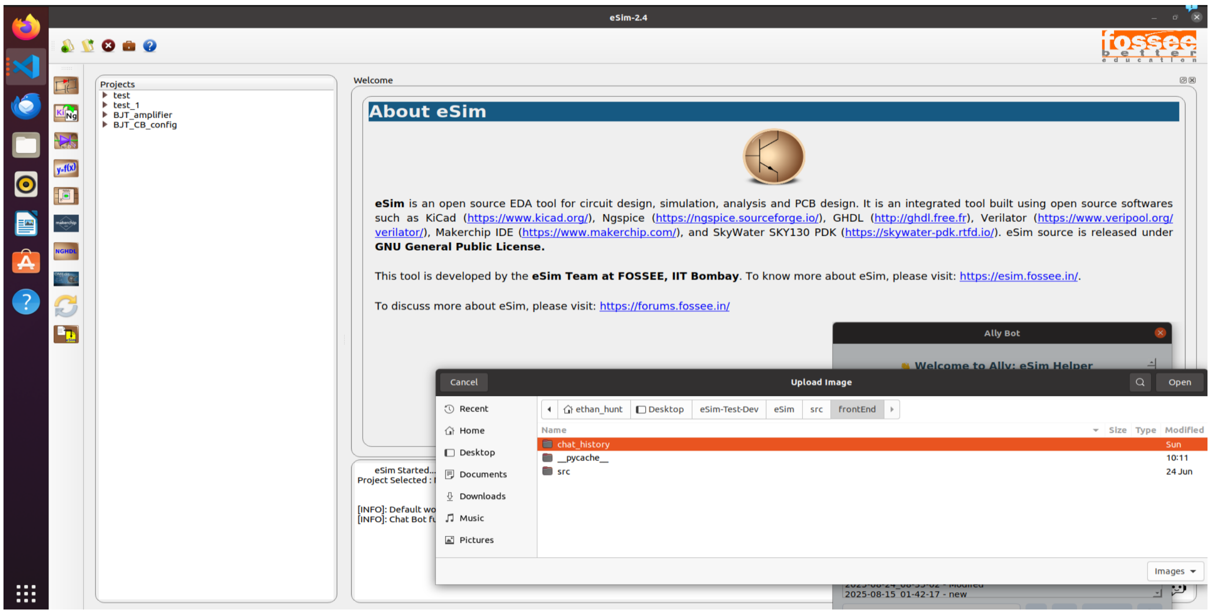
Welcome to Ally: eSim Helper

Ask questions about your circuit simulations, error logs, or upload

Your conversation will appear here...

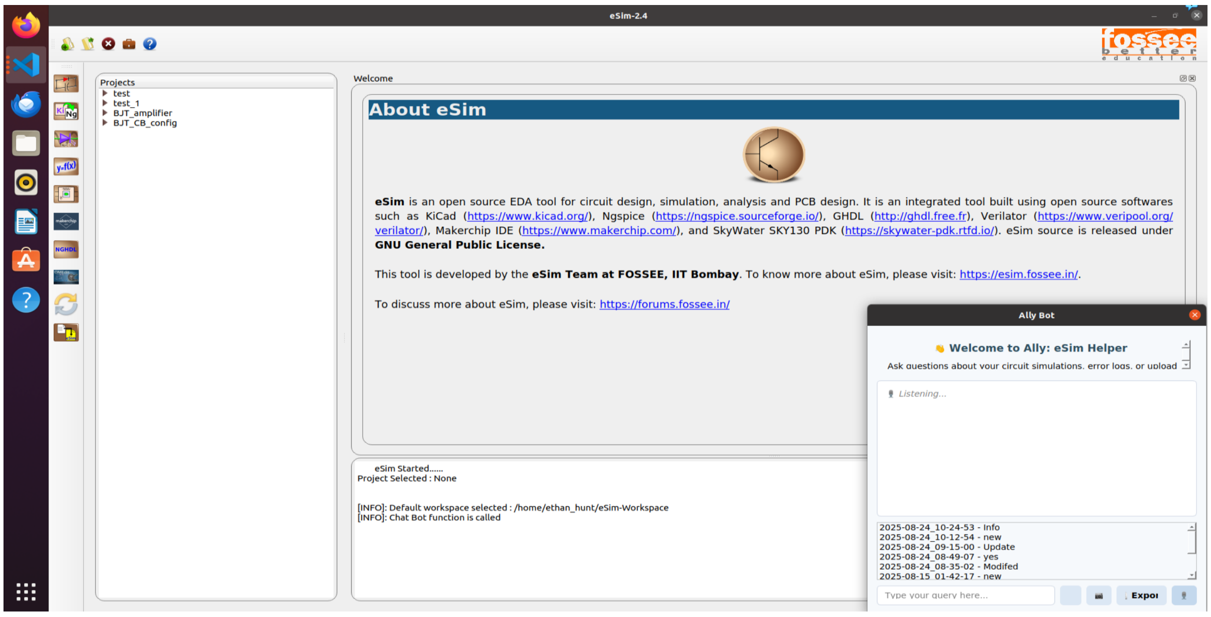
```
2025-08-24_10-24-53 - Info
2025-08-24_10-12-54 - new
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
```

Figure 20.1: Home Page



Name	Size	Type	Modified
chat_history			Sun
__pycache__			10:11
src			24 Jun

Figure 20.2: External Image Uploader


eSim-2.4

Projects
- test
- test_1
- BJT_amplifier
- BJT_CB_config

Welcome

About eSim

eSim is an open source EDA tool for circuit design, simulation, analysis and PCB design. It is an integrated tool built using open source softwares such as KiCad (https://www.kicad.org/), Ngspice (https://ngspice.sourceforge.io/), GHDL (http://ghdl.free.fr), Verilator (https://www.veripool.org/verilator/), Makerchip IDE (https://www.makerchip.com/), and SkyWater SKY130 PDK (https://skywater-pdk.rtfd.io/). eSim source is released under **GNU General Public License.**

This tool is developed by the **eSim Team at FOSSEE, IIT Bombay**. To know more about eSim, please visit: https://esim.fossee.in/.

To discuss more about eSim, please visit: https://forums.fossee.in/

```
      eSim Started......
Project Selected : None


[INFO]: Default workspace selected : /home/ethan_hunt/eSim-Workspace
[INFO]: Chat Bot function is called
```

Ally Bot

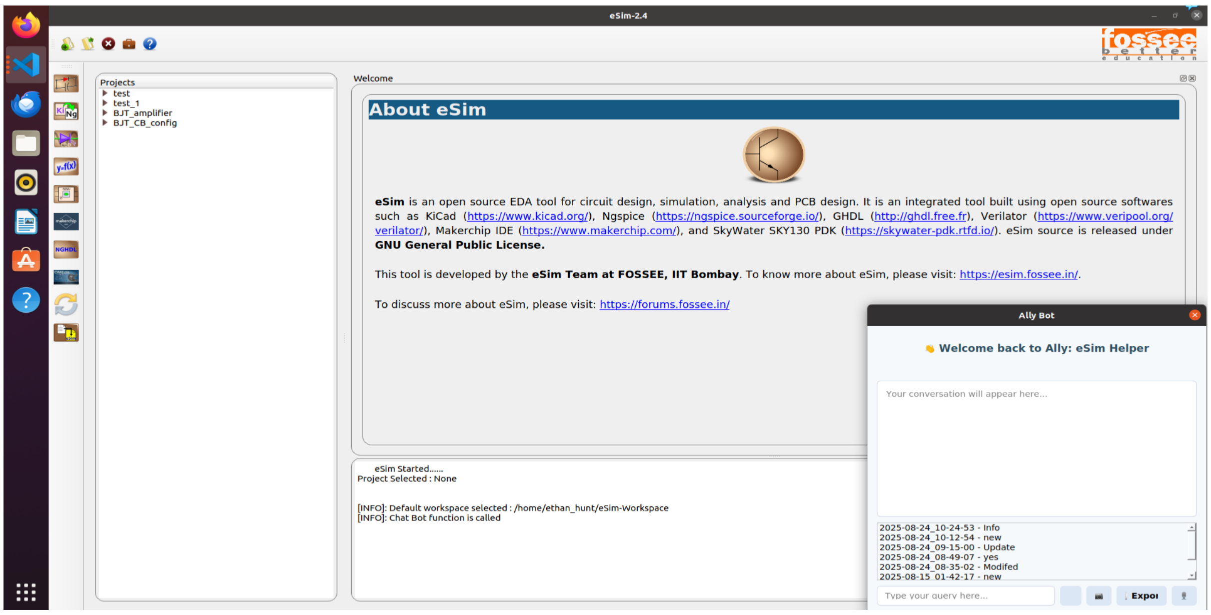
Welcome to Ally: eSim Helper

Ask questions about your circuit simulations, error logs, or upload

Listening...

```
2025-08-24_10-24-53 - Info
2025-08-24_10-12-54 - new
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
```

Figure 20.3: Audio Support


eSim-2.4

Projects
- test
- test_1
- BJT_amplifier
- BJT_CB_config

Welcome

About eSim

eSim is an open source EDA tool for circuit design, simulation, analysis and PCB design. It is an integrated tool built using open source softwares such as KiCad (https://www.kicad.org/), Ngspice (https://ngspice.sourceforge.io/), GHDL (http://ghdl.free.fr), Verilator (https://www.veripool.org/verilator/), Makerchip IDE (https://www.makerchip.com/), and SkyWater SKY130 PDK (https://skywater-pdk.rtfd.io/). eSim source is released under **GNU General Public License.**

This tool is developed by the **eSim Team at FOSSEE, IIT Bombay**. To know more about eSim, please visit: https://esim.fossee.in/.

To discuss more about eSim, please visit: https://forums.fossee.in/

```
      eSim Started......
Project Selected : None


[INFO]: Default workspace selected : /home/ethan_hunt/eSim-Workspace
[INFO]: Chat Bot function is called
```

Ally Bot

Welcome back to Ally: eSim Helper

Your conversation will appear here...

```
2025-08-24_10-24-53 - Info
2025-08-24_10-12-54 - new
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
```

Figure 20.4: Chat History

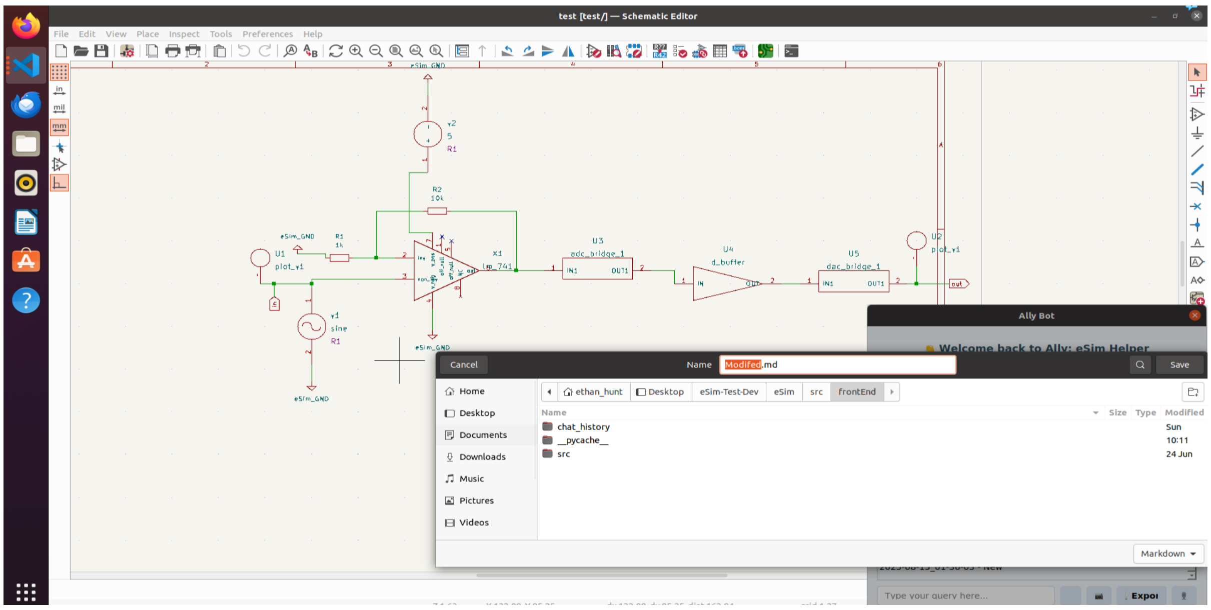
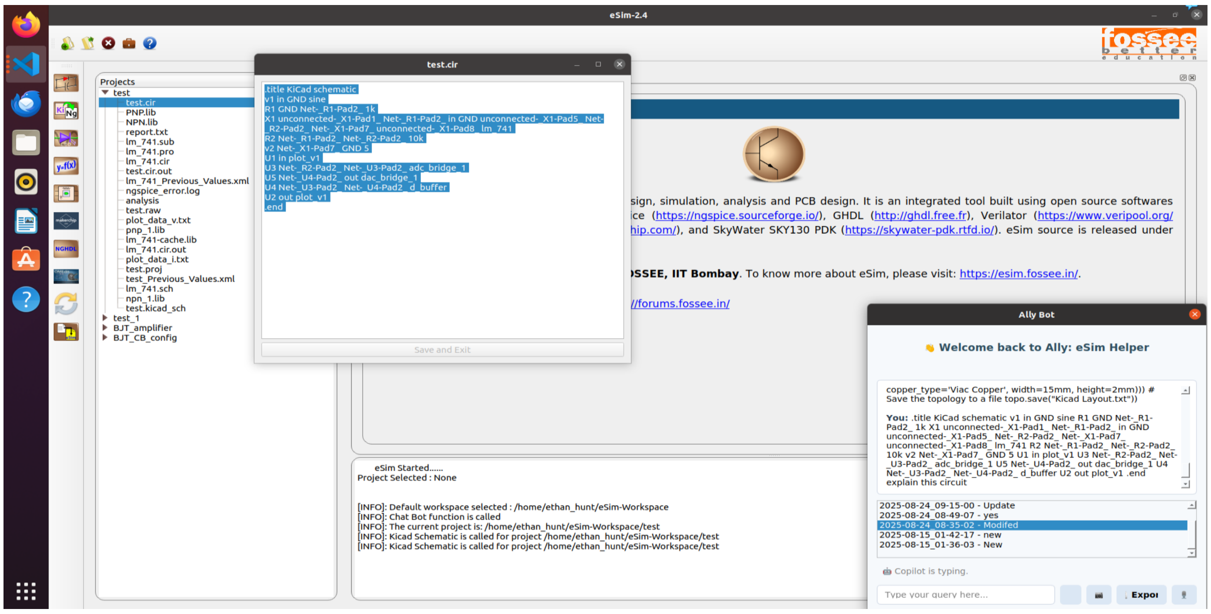


Figure 20.5: Chat Export: External Save



```
.title KiCad schematic
v1 in GND sine
R1 GND Net-_R1-Pad2_ 1k
X1 unconnected-_X1-Pad1_ Net-_R1-Pad2_ in GND unconnected-_X1-Pad5_ Net-
_R2-Pad2_ Net-_X1-Pad7_ unconnected-_X1-Pad8_ lm_741
R2 Net-_R1-Pad2_ Net-_R2-Pad2_ 10k
v2 Net-_X1-Pad7_ GND 5
U1 in plot_v1
U3 Net-_R2-Pad2_ Net-_U3-Pad2_ adc_bridge_1
U5 Net-_U4-Pad2_ out dac_bridge_1
U4 Net-_U3-Pad2_ Net-_U4-Pad2_ d_buffer
U2 out plot_v1
.end
```

sign, simulation, analysis and PCB design. It is an integrated tool built using open source softwares ice (https://ngspice.sourceforge.io/), GHDL (http://ghdl.free.fr), Verilator (https://www.veripool.org/ hip.com/), and SkyWater SKY130 PDK (https://skywater-pdk.rtfd.io/). eSim source is released under

OSSEE, IIT Bombay. To know more about eSim, please visit: https://esim.fossee.in/.

//forums.fossee.in/

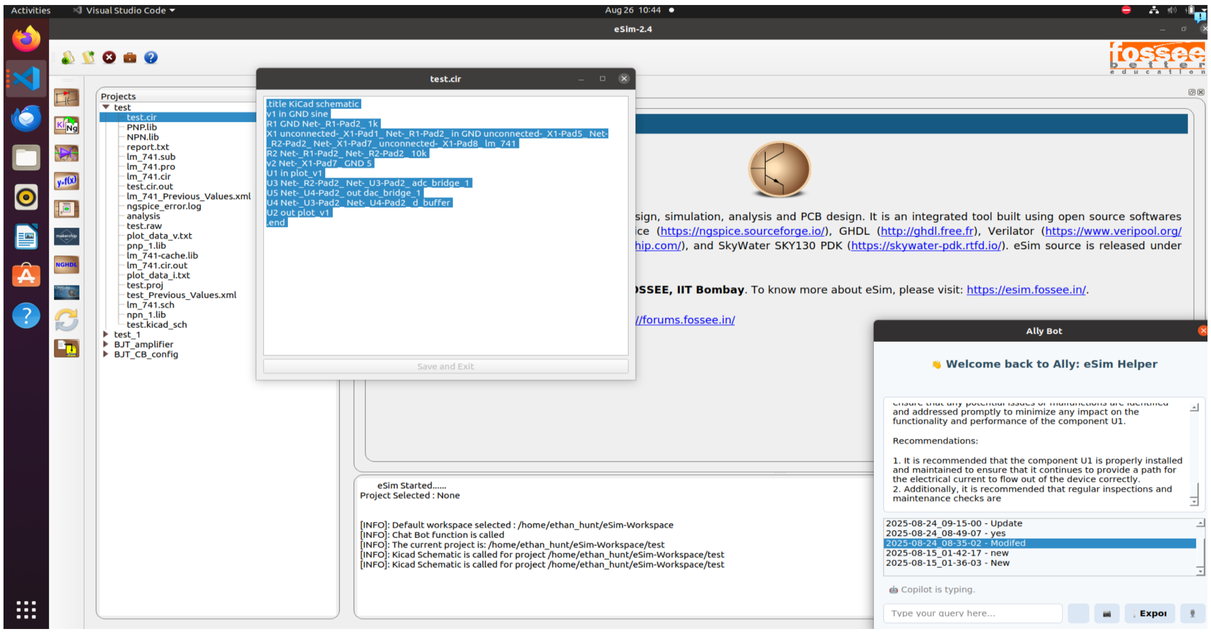
```
eSim Started......
Project Selected : None

[INFO]: Default workspace selected : /home/ethan_hunt/eSim-Workspace
[INFO]: Chat Bot function is called
[INFO]: The current project is: /home/ethan_hunt/eSim-Workspace/test
[INFO]: Kicad Schematic is called for project /home/ethan_hunt/eSim-Workspace/test
[INFO]: Kicad Schematic is called for project /home/ethan_hunt/eSim-Workspace/test
```

copper_type='Viac Copper', width=15mm, height=2mm))) # Save the topology to a file topo.save("Kicad Layout.txt"))

You: .title KiCad schematic v1 in GND sine R1 GND Net-_R1-Pad2_ 1k X1 unconnected-_X1-Pad1_ Net-_R1-Pad2_ in GND unconnected-_X1-Pad5_ Net-_R2-Pad2_ Net-_X1-Pad7_ unconnected-_X1-Pad8_ lm_741 R2 Net-_R1-Pad2_ Net-_R2-Pad2_ 10k v2 Net-_X1-Pad7_ GND 5 U1 in plot_v1 U3 Net-_R2-Pad2_ Net-_U3-Pad2_ adc_bridge_1 U5 Net-_U4-Pad2_ out dac_bridge_1 U4 Net-_U3-Pad2_ Net-_U4-Pad2_ d_buffer U2 out plot_v1 .end explain this circuit

```
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
2025-08-15_01-36-03 - New
```

Figure 20.6: test.cir Schematic Debugging



```
.title KiCad schematic
v1 in GND sine
R1 GND Net-_R1-Pad2_ 1k
X1 unconnected-_X1-Pad1_ Net-_R1-Pad2_ in GND unconnected-_X1-Pad5_ Net-
_R2-Pad2_ Net-_X1-Pad7_ unconnected-_X1-Pad8_ lm_741
R2 Net-_R1-Pad2_ Net-_R2-Pad2_ 10k
v2 Net-_X1-Pad7_ GND 5
U1 in plot_v1
U3 Net-_R2-Pad2_ Net-_U3-Pad2_ adc_bridge_1
U5 Net-_U4-Pad2_ out dac_bridge_1
U4 Net-_U3-Pad2_ Net-_U4-Pad2_ d_buffer
U2 out plot_v1
.end
```

sign, simulation, analysis and PCB design. It is an integrated tool built using open source softwares
ce (https://ngspice.sourceforge.io/), GHDL (http://ghdl.free.fr), Verilator (https://www.veripool.org/
hip.com/), and SkyWater SKY130 PDK (https://skywater-pdk.rtfd.io/). eSim source is released under

SSEE, IIT Bombay. To know more about eSim, please visit: https://esim.fossee.in/.

//forums.fossee.in/

```
    eSim Started......
Project Selected : None


[INFO]: Default workspace selected : /home/ethan_hunt/eSim-Workspace
[INFO]: Chat Bot function is called
[INFO]: The current project is: /home/ethan_hunt/eSim-Workspace/test
[INFO]: Kicad Schematic is called for project /home/ethan_hunt/eSim-Workspace/test
[INFO]: Kicad Schematic is called for project /home/ethan_hunt/eSim-Workspace/test
```

👋 Welcome back to Ally: eSim Helper

and addressed promptly to minimize any impact on the functionality and performance of the component U1.

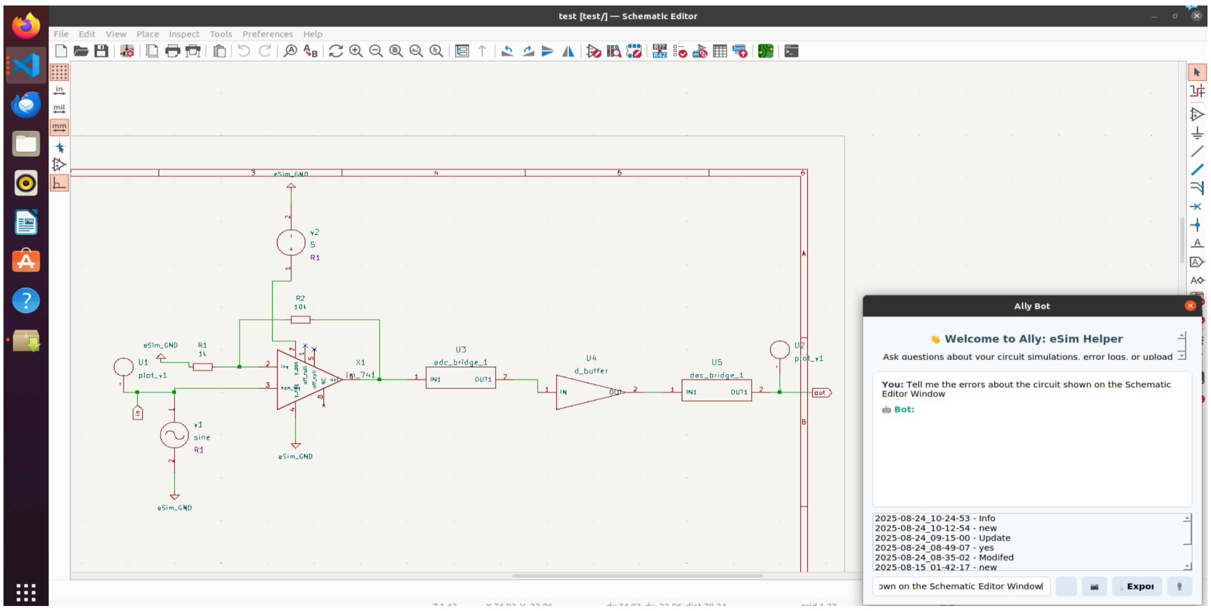
Recommendations:

1. It is recommended that the component U1 is properly installed and maintained to ensure that it continues to provide a path for the electrical current to flow out of the device correctly.
2. Additionally, it is recommended that regular inspections and maintenance checks are

```
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
2025-08-15_01-36-03 - New
```

🤖 Copilot is typing.

Figure 20.7: test.cir Schematic Analysis



👋 Welcome to Ally: eSim Helper

Ask questions about your circuit simulations, error logs, or upload

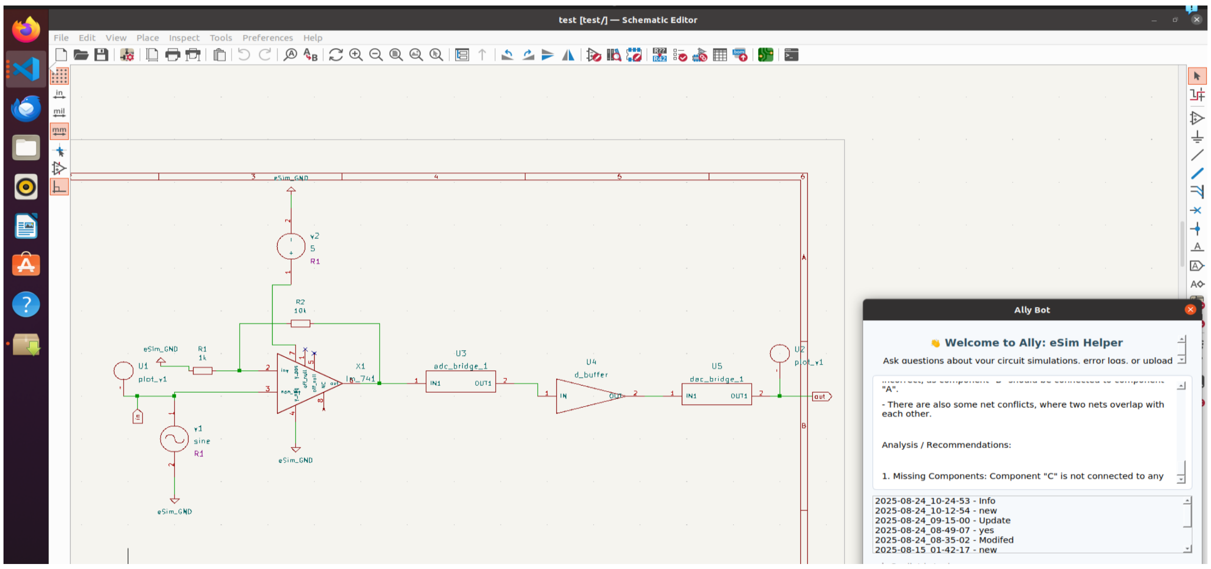
You: Tell me the errors about the circuit shown on the Schematic Editor Window

🤖 **Bot:**

```
2025-08-24_10-24-53 - Info
2025-08-24_10-12-54 - new
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
```

own on the Schematic Editor Window

Figure 20.8: OnPrompt Debugging with Image Analysis Support

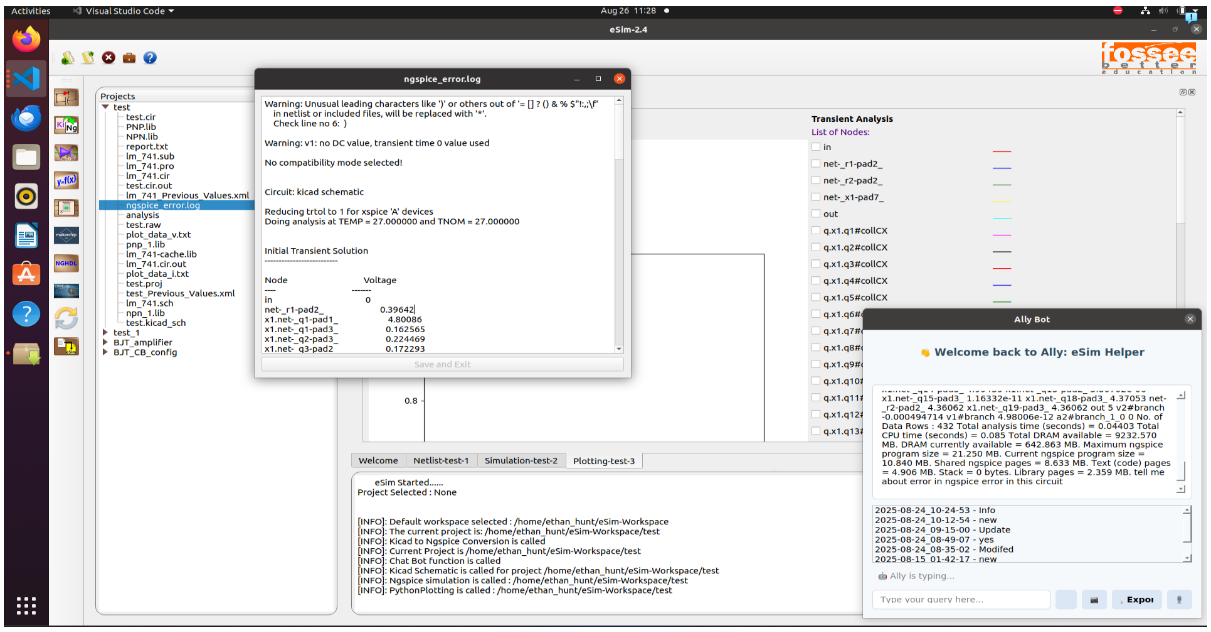


Ask questions about your circuit simulations, error logs, or upload

```
"A".

- There are also some net conflicts, where two nets overlap with
each other.

Analysis / Recommendations:

1. Missing Components: Component "C" is not connected to any
```

```
2025-08-24_10-24-53 - Info
2025-08-24_10-12-54 - new
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
```

Figure 20.9: Analysed Answer for Schematic Editor



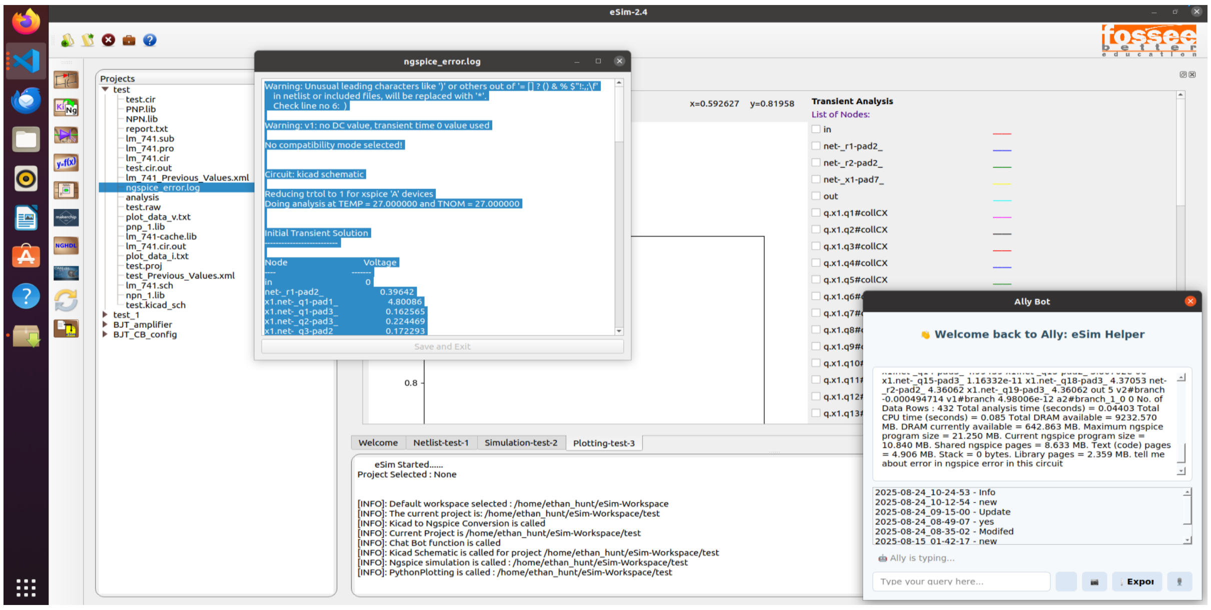
```
Warning: Unusual leading characters like ')' or others out of '= [] ? () & % $"!:,;\f'
   in netlist or included files, will be replaced with '*'.
   Check line no 6:  )

Warning: v1: no DC value, transient time 0 value used

No compatibility mode selected!


Circuit: kicad schematic

Reducing trtol to 1 for xspice 'A' devices
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000


Initial Transient Solution
--------------------------


Node                          Voltage
----                          -------
in                              0
net-_r1-pad2_                   0.39642
x1.net-_q1-pad1_                  4.80086
x1.net-_q1-pad3_                  0.162565
x1.net-_q2-pad3_                  0.224469
x1.net-_q3-pad2_                  0.172293
```

```
x1.net-_q15-pad3_ 1.16332e-11 x1.net-_q18-pad3_ 4.37053 net-
_r2-pad2_ 4.36062 x1.net-_q19-pad3_ 4.36062 out 5 v2#branch
-0.000494714 v1#branch 4.98006e-12 a2#branch_1_0 0 No. of
Data Rows : 432 Total analysis time (seconds) = 0.04403 Total
CPU time (seconds) = 0.085 Total DRAM available = 9232.570
MB. DRAM currently available = 642.863 MB. Maximum ngspice
program size = 21.250 MB. Current ngspice program size =
10.840 MB. Shared ngspice pages = 8.633 MB. Text (code) pages
= 4.906 MB. Stack = 0 bytes. Library pages = 2.359 MB. tell me
about error in ngspice error in this circuit
```

```
2025-08-24_10-24-53 - Info
2025-08-24_10-12-54 - new
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
```

```
      eSim Started......
Project Selected : None


[INFO]: Default workspace selected : /home/ethan_hunt/eSim-Workspace
[INFO]: The current project is: /home/ethan_hunt/eSim-Workspace/test
[INFO]: Kicad to Ngspice Conversion is called
[INFO]: Current Project is /home/ethan_hunt/eSim-Workspace/test
[INFO]: Chat Bot function is called
[INFO]: Kicad Schematic is called for project /home/ethan_hunt/eSim-Workspace/test
[INFO]: Ngspice simulation is called : /home/ethan_hunt/eSim-Workspace/test
[INFO]: PythonPlotting is called : /home/ethan_hunt/eSim-Workspace/test
```

Figure 20.10: Ngspice Error Log



ngspice_error.log

```
Warning: Unusual leading characters like ')' or others out of '= [] ? () & % $"!:,;\f'
   in netlist or included files, will be replaced with '*'.
   Check line no 6:  )

Warning: v1: no DC value, transient time 0 value used

No compatibility mode selected!


Circuit: kicad schematic

Reducing trtol to 1 for xspice 'A' devices
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000


Initial Transient Solution
--------------------------

Node                          Voltage
----                          -------
in                                  0
net-_r1-pad2_                 0.39642
x1.net-_q1-pad1_              4.80086
x1.net-_q1-pad3_              0.162565
x1.net-_q2-pad3_              0.224469
x1.net-_q3-pad2               0.172293
```

```
       eSim Started......
Project Selected : None


[INFO]: Default workspace selected : /home/ethan_hunt/eSim-Workspace
[INFO]: The current project is: /home/ethan_hunt/eSim-Workspace/test
[INFO]: Kicad to Ngspice Conversion is called
[INFO]: Current Project is /home/ethan_hunt/eSim-Workspace/test
[INFO]: Chat Bot function is called
[INFO]: Kicad Schematic is called for project /home/ethan_hunt/eSim-Workspace/test
[INFO]: Ngspice simulation is called : /home/ethan_hunt/eSim-Workspace/test
[INFO]: PythonPlotting is called : /home/ethan_hunt/eSim-Workspace/test
```

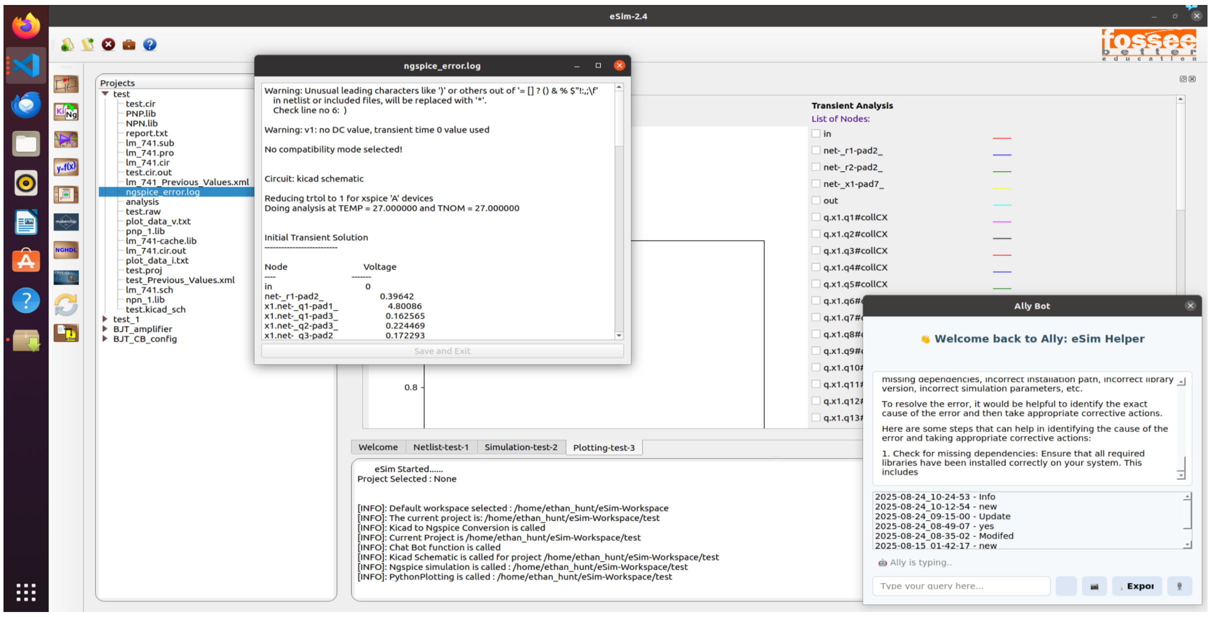
👋 Welcome back to Ally: eSim Helper

x1.net-_q15-pad3_ 1.16332e-11 x1.net-_q18-pad3_ 4.37053 net-_r2-pad2_ 4.36062 x1.net-_q19-pad3_ 4.36062 out 5 v2#branch -0.000494714 v1#branch 4.98006e-12 a2#branch_1_0 0 No. of Data Rows : 432 Total analysis time (seconds) = 0.04403 Total CPU time (seconds) = 0.085 Total DRAM available = 9232.570 MB. DRAM currently available = 642.863 MB. Maximum ngspice program size = 21.250 MB. Current ngspice program size = 10.840 MB. Shared ngspice pages = 8.633 MB. Text (code) pages = 4.906 MB. Stack = 0 bytes. Library pages = 2.359 MB. tell me about error in ngspice error in this circuit

```
2025-08-24_10-24-53 - Info
2025-08-24_10-12-54 - new
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
```

🤖 Ally is typing...

Figure 20.11: Error Log Analysis



ngspice_error.log

```
Warning: Unusual leading characters like ')' or others out of '= [] ? () & % $"!:,;\f'
   in netlist or included files, will be replaced with '*'.
   Check line no 6:  )

Warning: v1: no DC value, transient time 0 value used

No compatibility mode selected!


Circuit: kicad schematic

Reducing trtol to 1 for xspice 'A' devices
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000


Initial Transient Solution
--------------------------

Node                          Voltage
----                          -------
in                                  0
net-_r1-pad2_                 0.39642
x1.net-_q1-pad1_              4.80086
x1.net-_q1-pad3_              0.162565
x1.net-_q2-pad3_              0.224469
x1.net-_q3-pad2               0.172293
```

```
       eSim Started......
Project Selected : None


[INFO]: Default workspace selected : /home/ethan_hunt/eSim-Workspace
[INFO]: The current project is: /home/ethan_hunt/eSim-Workspace/test
[INFO]: Kicad to Ngspice Conversion is called
[INFO]: Current Project is /home/ethan_hunt/eSim-Workspace/test
[INFO]: Chat Bot function is called
[INFO]: Kicad Schematic is called for project /home/ethan_hunt/eSim-Workspace/test
[INFO]: Ngspice simulation is called : /home/ethan_hunt/eSim-Workspace/test
[INFO]: PythonPlotting is called : /home/ethan_hunt/eSim-Workspace/test
```

👋 Welcome back to Ally: eSim Helper

missing dependencies, incorrect installation path, incorrect library version, incorrect simulation parameters, etc.

To resolve the error, it would be helpful to identify the exact cause of the error and then take appropriate corrective actions.

Here are some steps that can help in identifying the cause of the error and taking appropriate corrective actions:

1. Check for missing dependencies: Ensure that all required libraries have been installed correctly on your system. This includes

```
2025-08-24_10-24-53 - Info
2025-08-24_10-12-54 - new
2025-08-24_09-15-00 - Update
2025-08-24_08-49-07 - yes
2025-08-24_08-35-02 - Modifed
2025-08-15_01-42-17 - new
```

🤖 Ally is typing..

Figure 20.12: Ally is Thinking

References

* *FOSSEE eSim Documentation and Manuals. Electronic Circuit Design and Simulation Using eSim.*
  https://fossee.in/eSim

* *FOSSEE eSim Community Reports and Case Studies. User Feedback and Simulation Reports for eSim. FOSSEE Project Repository.*
  https://fossee.in/reports

* *KiCad Documentation. Open Source PCB Design Software.*
  https://www.kicad.org/documentation/

* *NgSpice User Manual. Simulation and Analysis of Electronic Circuits.*
  http://ngspice.sourceforge.net/docs.html

* *Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela(2020).*
  Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.
  NeurIPS 2020.
  https://arxiv.org/abs/2005.11401

* *Chen, D., Fisch, A., Weston, J., Bordes, A. (2017). Reading Wikipedia to Answer Open-Domain Questions. ACL 2017.*
  https://arxiv.org/abs/1704.00051Chen, D., Fisch, A., Weston, & Bordes, A. (2017).
  Reading Wikipedia to Answer Open-Domain Questions. ACL 2017.
  https://arxiv.org/abs/1704.00051

* *Jurafsky, D., Martin, J. H. (2023). Speech and Language Processing (3rd edition draft). Pearson.*
  https://web.stanford.edu/ jurafsky/slp3/

* *He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. CVPR 2016.*
  https://arxiv.org/abs/1512.03385

* *Redmon, & Farhadi, A. (2018). YOLOv3: An Incremental Improvement.*
  https://arxiv.org/abs/1804.02767

* *OpenCV Documentation. Open Source Computer Vision Library.*
  https://opencv.org/

* *LangChain Documentation. Framework for Building RAG-Powered Applications.*
  https://www.langchain.com/docs/

* *OpenAI. Embeddings and Vector Databases for Retrieval.*
  https://platform.openai.com/docs/guides/embeddings

Final Remarks

This fellowship project has been an invaluable opportunity for me to explore the intersection of open-source EDA tools, AI-driven assistance, and multimodal interaction. Working on the offline multimodal Agentic RAG system for eSim allowed me to enhance the usability and accessibility of the software while gaining hands-on experience with modern technologies such as computer vision, retrieval-augmented generation, and voice-enabled interfaces.

Throughout this project, I have learned a great deal about designing and implementing a responsive, context-aware, and versatile support system. Integrating LLMs, knowledge retrieval frameworks, and offline processing pipelines has deepened my understanding of both software architecture and AI-driven educational tools. This experience also highlighted the importance of iterative development, user-centered design, and careful planning in delivering practical and effective solutions.

Working on this project has given me insight into the broader potential of open-source educational software and how thoughtful enhancements can improve user engagement, learning outcomes, and accessibility. While I am proud of the improvements achieved for eSim users, I also recognize that this work lays a foundation for future exploration, refinement, and expansion. It has inspired me to continue contributing to making educational tools more intuitive, effective, and accessible.

I hope that the work I have completed during this fellowship not only benefits eSim users immediately but also serves as a model for similar initiatives in other open-source educational projects. This experience has been both professionally enriching and personally rewarding, and I am grateful for the opportunity to learn, experiment, and grow through it.
