eSim Semester Long Internship
2025

Project Report

On

AI Chatbot Integration in esim

Submitted by
Radhika Goyal

Department of Computer Science Engineering
Gitam Deemed to be University

Under the Guidance of
Prof. Prabhu Ramachandran
Principal Investigator
Department of Aerospace Engineering
Indian Institute of Technology Bombay

January 2026

Acknowledgement

I express my sincere gratitude to Prof. Prabhu Ramachandran for providing me with the opportunity to be part of the FOSSEE internship program and for his continued efforts in promoting open-source engineering tool development. His leadership and vision have been instrumental in fostering meaningful student participation in the open-source ecosystem.

I also acknowledge Prof. Kannan M. Moudgalya for his foundational role in establishing and strengthening the FOSSEE initiative. His contributions to open-source education and the development of the FOSSEE fellowship framework have been pivotal in creating the academic and organisational platform through which this internship was undertaken.

I am deeply grateful to my mentor, Mr. Sumanto Kar, for his consistent guidance, technical expertise, and mentorship throughout the course of this project. His insights, constructive feedback, and systematic approach were crucial in addressing challenges and ensuring the successful development and integration of the AI chatbot in eSim.

I extend my sincere appreciation to my internal mentors, Mr. Varad Patil, Mr. Aditya Minocha, and Ms. Shanthi Priya, for their coordination, technical inputs, and constructive feedback during the internship. Their support played an important role in maintaining clarity, direction, and steady progress in the execution of the work.

This internship provided valuable exposure to software development, artificial intelligence, and electronic design automation within a structured open-source environment. I also thank the FOSSEE team for their support and coordination, which facilitated smooth progress and the successful completion of the project.

Contents

Chapter 1

Introduction

1.1 Background of the FOSSEE–eSim Environment

The Free and Open Source Software for Education (FOSSEE) initiative is a nationally coordinated project headquartered at the Indian Institute of Technology Bombay (IIT Bombay). The project operates under the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education (MoE), Government of India. The primary objective of FOSSEE is to promote the adoption of free and open-source software within engineering education, academic research, and technological development across the country.

By encouraging the use of community-driven and freely accessible tools, the initiative aims to reduce dependency on proprietary software platforms and to establish sustainable alternatives that align with academic and institutional requirements. As part of this effort, FOSSEE has supported the development and dissemination of multiple domain-specific software solutions designed to facilitate learning, experimentation, and innovation across a wide range of science and engineering disciplines.

One of the key outcomes of the FOSSEE initiative is eSim, an open-source Electronic Design Automation (EDA) platform developed to support circuit design, simulation, and analysis. eSim provides an integrated working environment that combines schematic capture with simulation backends and waveform-based analysis tools. By integrating multiple open-source components within a unified workflow, the platform enables users to design, simulate, and evaluate analog, digital, and mixed-signal electronic circuits.

The eSim platform is designed to maintain compatibility with established SPICE-based simulation methodologies, allowing it to function as a viable alternative to commercial

circuit design and simulation tools. Its modular and extensible architecture supports transparency, adaptability, and customization, which are essential characteristics for educational and research-oriented environments.

eSim has been widely adopted by academic institutions for laboratory instruction, independent learning, and research-driven circuit development. Its open-source nature allows users to explore internal workflows, modify tool behavior, and extend functionality as required. However, this level of openness also requires users to engage more deeply with circuit modeling principles, simulator constraints, netlist-level conventions, and tool-specific operational guidelines.

As a result, while eSim provides a powerful and flexible platform for electronic design education, new and intermediate users often experience a steep learning curve. Challenges frequently arise during simulation setup, debugging, and interpretation of error messages, particularly when users lack prior exposure to SPICE-based simulation environments. These factors highlight the need for enhanced user support mechanisms within the eSim ecosystem.

1.2 Rationale for Intelligent Assistance in Circuit Simulation

Circuit simulation tools rely on formally defined mathematical representations derived from circuit topologies, device models, and simulation parameters. These representations typically involve systems of algebraic and differential equations that enable accurate prediction of circuit behavior under various operating conditions. While such frameworks offer precision and reliability, they also produce diagnostic warnings and error messages that are often low-level, domain-specific, and syntactically concise.

Interpreting these messages requires users to map simulator feedback to schematic-level design issues, such as incorrect connectivity, missing component models, invalid parameter values, or incomplete circuit definitions. For users without extensive experience in simulation semantics, this process can be challenging and time-consuming.

The complexity of circuit simulation workflows further compounds these difficulties during iterative design and debugging phases. Minor inconsistencies—such as incorrect node labeling, improperly grounded circuits, or unsupported device parameters—can prevent simulation convergence or result in execution failure. Identifying the root cause of such issues often necessitates manual inspection of generated netlists, simulator constraints, and detailed error logs, thereby increasing diagnostic overhead and reducing productivity.

These challenges emphasize the importance of structured assistance mechanisms that can reduce cognitive load and support systematic troubleshooting. Embedding intelligent support directly within the simulation environment can address these limitations by providing contextual explanations, design-level insights, and targeted guidance based on user queries and simulation context.

By correlating schematic information, simulation outputs, and known failure patterns, an intelligent assistant can facilitate more efficient debugging and improve accessibility for beginners. At the same time, such support enhances the overall learning experience by allowing users to focus on conceptual understanding and design intent rather than low-level diagnostic interpretation.

1.3 Aim and Scope of the Project

The primary aim of this project is to design and develop an AI-assisted chatbot to support users during circuit design and simulation activities within the eSim environment. The chatbot is intended to function as an interactive support system that assists users in understanding simulation behavior, interpreting error messages, and resolving common design and configuration issues encountered during circuit analysis.

The system is designed to provide contextual explanations and recommendations by processing user queries expressed in natural language. By complementing existing documentation and manuals, the chatbot enhances the troubleshooting process and reduces reliance on external resources. The assistant focuses on guiding users through error resolution rather than replacing the underlying simulation engine.

The scope of the project includes several key functional capabilities required for effective assistance in a circuit simulation context. These capabilities include natural language query processing, schematic and netlist analysis for extracting circuit-level information, and identification of common simulation errors such as connectivity faults, missing components, and configuration inconsistencies. Based on this analysis, the system generates corrective suggestions to assist users in resolving identified issues.

In addition, the chatbot is designed to support multimodal interaction, enabling users to interact using textual queries and visual inputs where applicable. A critical design objective of the system is offline operability, ensuring that all processing, retrieval, and inference tasks are executed locally. This feature allows the assistant to remain functional in environments with limited or no internet connectivity, such as academic laboratories and standalone installations.

Importantly, the chatbot operates as an advisory layer without introducing modifications to the core eSim simulation framework. This design choice preserves the integrity and reliability of the existing software while enhancing usability and user support through intelligent assistance.

Chapter 2

Functional Capabilities of the eSim Copilot

2.1 Offline Query Assistance

Offline query assistance constitutes a core functional capability of the eSim Copilot. The system is designed to process user queries locally by utilizing preloaded language models in conjunction with a structured knowledge base derived from eSim documentation, simulation manuals, and domain-specific troubleshooting resources. This design enables the chatbot to deliver meaningful guidance without dependence on external services or continuous internet connectivity.

The offline operational model ensures uninterrupted access to assistance in environments where network availability may be limited or restricted. In academic laboratories, institutional setups, and standalone installations, this capability allows users to continue circuit design, experimentation, and learning activities without disruption. As a result, the reliability and accessibility of the support system are significantly enhanced, particularly in educational contexts where consistent technical guidance is essential.

Local execution of query processing also improves system responsiveness and predictability. Since all computations are performed on the user's machine, latency associated with remote server communication is eliminated. Furthermore, offline processing ensures that user interactions, circuit information, and simulation-related data remain confined to the local environment, thereby enhancing privacy, data control, and operational stability.

2.2 Circuit Debugging Support

The eSim Copilot provides structured assistance for circuit debugging by identifying commonly encountered schematic-level and simulation-related issues during circuit development. The system analyzes user queries to detect references to known error conditions and maps them to predefined diagnostic categories, including connectivity errors, missing or incorrect model declarations, improper component specifications, and convergence-related simulation failures.

Based on this classification, the chatbot generates targeted recommendations that guide users toward appropriate corrective actions. These suggestions are aligned with established practices in eSim and NgSpice, ensuring that the guidance remains consistent with simulator constraints and expected workflows. Through this approach, users receive actionable advice that is both context-aware and technically relevant.

In addition to proposing corrective steps, the system emphasizes conceptual understanding by explaining the underlying reasons for observed issues. This explanatory approach promotes systematic debugging practices and supports deeper comprehension of simulation behavior. By helping users understand not only how to fix errors but also why they occur, the chatbot reduces the likelihood of repeated mistakes during iterative circuit design.

2.3 Schematic Interpretation

Schematic interpretation enables the eSim Copilot to reason about circuit structure, component relationships, and node connectivity within a designed schematic. By analyzing schematic representations, the system can infer electrical interactions between components and identify how signals propagate through the circuit topology.

This capability allows the chatbot to respond effectively to queries related to circuit configuration and functional intent. It assists users in verifying whether a schematic aligns with expected operational behavior and helps identify regions that may require closer examination or modification. Such interpretation is particularly valuable when users seek clarification about circuit functionality or design correctness.

Furthermore, schematic interpretation supports early detection of design inconsistencies prior to simulation execution. By highlighting potential issues at the schematic level, the system helps users reduce debugging overhead and improves overall design efficiency. Early identification of design flaws contributes to smoother simulation workflows and

more effective learning outcomes.

2.4 Netlist Evaluation Features

Netlists serve as the formal textual input to SPICE-based simulators and represent schematic designs in a structured format suitable for simulation execution. Although essential for accurate simulation, netlists are often difficult for users to interpret due to their compact syntax, directive-based structure, and low-level abstraction.

The eSim Copilot evaluates netlists by examining component declarations, simulation directives, parameter assignments, and structural constraints. Through this analysis, the system identifies potential issues such as missing specifications, improper references, unsupported statements, or conflicting instructions that may hinder successful simulation or affect numerical stability.

By providing structured explanations of netlist elements and their roles within the simulation workflow, the chatbot helps users understand the relationship between schematic design decisions and simulator-level execution behavior. This capability improves interpretability, facilitates troubleshooting, and enhances user confidence when working with SPICE-based simulation environments.

2.5 Multimodal Interaction Support

The eSim Copilot supports multimodal interaction by accepting both textual and visual inputs during user engagement. In addition to natural language queries, users can provide schematic images that allow the system to incorporate visual circuit context into its analysis and reasoning process.

This functionality is particularly useful in situations where textual descriptions alone are insufficient to accurately convey circuit topology, component placement, or connectivity. Visual input enables the chatbot to interpret structural relationships more effectively and generate responses that are closely aligned with the user's specific design configuration.

Multimodal interaction enhances accessibility and flexibility by accommodating diverse user workflows and communication preferences. By enabling richer forms of interaction between users and the assistant, the system contributes to a more intuitive, efficient, and effective troubleshooting experience within the circuit simulation environment.

Chapter 3

Problem Definition

3.1 Complexity of Circuit Design Workflows

Circuit design workflows comprise multiple interdependent stages, including schematic creation, component selection and parameterization, netlist generation, simulation execution, and result analysis. Each stage requires strict adherence to design conventions, accurate specification of parameters, and compliance with constraints imposed by the underlying simulation environment. Errors introduced at an early stage may propagate through subsequent phases, ultimately affecting the correctness, stability, and interpretability of simulation results.

As circuit designs increase in functional scope and structural complexity, interactions between components, subcircuits, and higher-level system blocks become increasingly difficult to track. Dependencies involving analog and digital elements, feedback paths, and hierarchical design structures can introduce subtle faults that are not immediately apparent during schematic construction. In many cases, such issues surface only when simulation is initiated, at which point identifying their origin becomes challenging due to indirect, low-level, or non-intuitive diagnostic feedback provided by the simulator.

In addition to technical challenges, circuit design workflows often require users to switch between multiple representations of the same design, such as graphical schematics, textual netlists, simulator logs, and waveform visualizations. Managing these transitions imposes significant cognitive load, particularly on students and beginners who are still developing familiarity with modeling conventions and simulation semantics. This complexity increases the likelihood of oversight and underscores the need for structured assistance throughout the design, simulation, and debugging process.

3.2 Common User Errors in Simulation

Users frequently encounter simulation failures arising from incomplete, inconsistent, or incorrect circuit specifications. Common examples include missing or improper ground references, floating or unconnected nodes, incompatible or unrealistic component values, and undefined or unsupported device models. Such issues can prevent simulations from executing successfully, lead to convergence problems, or produce results that do not reflect the intended circuit behavior.

Although simulation engines typically generate warning messages and error outputs when failures occur, these messages are often expressed in concise, technical formats that emphasize internal solver conditions rather than actionable guidance. Effective interpretation of such feedback requires familiarity with SPICE syntax, circuit modeling assumptions, numerical solver behavior, and simulator-specific constraints. For new and intermediate users, particularly in educational and laboratory settings, acquiring this level of expertise can be difficult and time-consuming.

Identifying recurring error patterns and understanding their underlying causes is therefore essential for improving the usability of circuit simulation tools. By systematically recognizing common sources of simulation failure, support mechanisms can better relate simulator feedback to meaningful corrective actions. This approach has the potential to improve troubleshooting efficiency, reduce frustration, and enhance conceptual understanding among users.

3.3 Gaps in Existing Tool Assistance

While eSim and similar circuit simulation tools provide reference manuals, user documentation, and basic diagnostic outputs, these resources are predominantly static and externally oriented. Users are generally expected to consult documentation independently, interpret error messages manually, and determine corrective steps based on prior knowledge or trial-and-error experimentation. This approach creates a disconnect between simulator feedback and practical resolution strategies during active circuit development.

The absence of interactive, context-aware assistance significantly limits the effectiveness of existing support mechanisms. When errors arise, users must independently identify relevant information, interpret low-level technical messages, and apply fixes without guided support. This process can be particularly discouraging for beginners, often leading to inefficient debugging cycles and reduced confidence in using the tool.

These limitations highlight the need for intelligent assistance that is integrated directly within the circuit design and simulation workflow. Context-aware support systems can reduce cognitive overhead by improving the interpretability of error messages, guiding users toward resolution strategies informed by schematic and simulation context, and encouraging more systematic troubleshooting practices. Such integration enhances usability and contributes to improved learning outcomes within open-source electronic design automation environments.

Chapter 4

System Architecture

4.1 Overview of the eSim AI Chatbot Architecture

The eSim Chatbot is designed as a modular, extensible system that integrates seamlessly with the existing eSim simulation environment while operating as an independent advisory layer. The architectural design emphasizes usability, offline functionality, and compatibility with the current eSim workflow, ensuring that the core simulation engine remains unmodified.

At a high level, the system architecture consists of a user interaction layer, a query processing and coordination layer, an intelligence layer responsible for reasoning and response generation, and supporting modules for knowledge retrieval and multimodal input handling. These components collectively enable the Copilot to interpret user queries, analyze relevant circuit context, and generate meaningful assistance in response to simulation-related challenges.

The architectural separation of concerns ensures maintainability and scalability. Each module performs a well-defined role and communicates with other components through structured interfaces, allowing future enhancements without disrupting existing functionality.

4.2 User Interaction Layer

The user interaction layer serves as the primary interface between the user and the eSim Copilot. This layer is responsible for capturing user inputs and presenting generated responses in an accessible and intuitive manner. Users can interact with the system

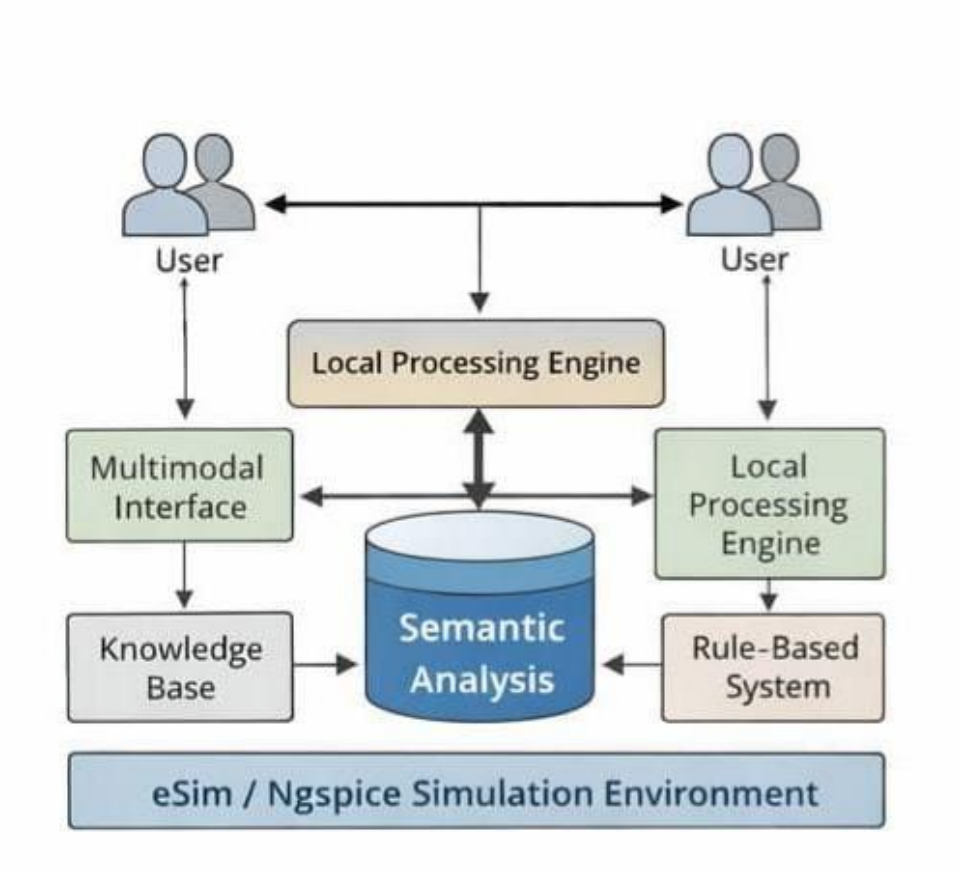

Figure 4.1: System architecture of the AI Chatbot integrated with eSim

through natural language queries and, where applicable, visual inputs such as schematic images.

The interaction layer is designed to integrate smoothly with the eSim graphical user interface, allowing users to seek assistance without leaving the simulation environment. By embedding the Copilot within the workflow, the system minimizes context switching and supports real-time guidance during circuit design, simulation, and debugging activities.

The responses generated by the Copilot are presented in a structured and readable format, emphasizing clarity and relevance. This design choice ensures that users can quickly understand suggested actions and apply them effectively within their design workflow.

4.3 Query Processing and Control Module

The query processing and control module functions as the coordination unit of the system. It receives user inputs from the interaction layer and performs initial preprocessing tasks such as query normalization, intent identification, and contextual tagging. This step en-

sures that the system accurately interprets user requests and determines the appropriate processing pathway.

Once the query is analyzed, the control module routes it to relevant downstream components, such as the knowledge retrieval subsystem or the language model inference engine. The module also manages the aggregation of intermediate results and ensures that responses are coherent, context-aware, and aligned with the user's original intent.

By centralizing query management, this module enables consistent handling of diverse input types and supports smooth coordination between multiple architectural components.

4.4 Knowledge Base and Retrieval Mechanism

The knowledge base forms a critical component of the eSim Copilot architecture. It consists of curated documentation, simulation guidelines, frequently encountered error patterns, and domain-specific troubleshooting information derived from eSim and NgSpice resources. This information is stored in a structured format to support efficient retrieval during query resolution.

A retrieval mechanism is employed to identify relevant knowledge fragments based on the processed user query. By matching query context with stored documentation and known issue patterns, the system ensures that responses are grounded in authoritative and tool-specific information. This approach reduces the likelihood of generic or misleading guidance and enhances the accuracy of the assistance provided.

The integration of a retrieval mechanism with the intelligence layer allows the Copilot to combine factual knowledge with contextual reasoning, resulting in responses that are both informative and practically applicable.

4.5 Intelligence and Reasoning Layer

The intelligence layer is responsible for generating explanations, recommendations, and guidance based on user queries and retrieved knowledge. This layer utilizes locally deployed language models to interpret input queries, synthesize relevant information, and construct coherent responses that align with circuit design and simulation practices.

The reasoning process incorporates both user-provided context and retrieved documentation to ensure that generated responses are technically accurate and contextually appro-

priate. By operating entirely offline, the intelligence layer supports reliable performance in environments with limited or no internet connectivity.

This layer plays a crucial role in transforming low-level simulator feedback into user-friendly explanations, thereby bridging the gap between technical diagnostics and actionable understanding.

4.6 Support for Multimodal Inputs

To enhance flexibility and usability, the system architecture includes support for multimodal inputs. In addition to text-based queries, the Copilot can process visual inputs such as schematic images. These inputs provide supplementary context that aids in understanding circuit topology, component placement, and connectivity.

Multimodal input handling enables the system to generate more precise and context-aware responses, particularly in scenarios where textual descriptions alone are insufficient. This capability supports diverse user workflows and accommodates different modes of interaction, thereby improving accessibility and effectiveness.

The modular design of the multimodal input subsystem allows for future expansion, such as improved image analysis or additional input modalities, without requiring architectural restructuring.

4.7 Offline Operation and Integration Considerations

A defining characteristic of the eSim Copilot architecture is its offline operational capability. All major components—including query processing, knowledge retrieval, and response generation—are executed locally on the user's system. This design ensures consistent availability of assistance regardless of network conditions and is particularly beneficial in academic and laboratory environments.

The Copilot is implemented as an auxiliary layer that interfaces with eSim without altering its internal simulation mechanisms. This non-intrusive integration preserves the stability and reliability of the existing software while enhancing its usability through intelligent assistance.

Overall, the architectural design of eSim Copilot prioritizes robustness, extensibility, and user-centered support, providing a scalable foundation for future enhancements and con-

tinued integration with open-source electronic design automation tools.

Chapter 5

Workflow and Control Flow of the Chatbot

5.1 Overview of the Operational Workflow

The workflow of the eSim Copilot defines the sequence of operations through which user queries are processed, analyzed, and resolved. The control flow is designed to ensure systematic handling of user interactions while maintaining responsiveness, contextual accuracy, and alignment with the eSim simulation environment. The workflow integrates multiple functional modules, including input acquisition, query interpretation, knowledge retrieval, reasoning, and response generation.

The overall process begins when a user encounters a design-related query or simulation issue during interaction with eSim. Rather than relying on external documentation or manual debugging, the user invokes the Copilot to obtain assistance. The Copilot then orchestrates a structured sequence of steps to interpret the query and generate an appropriate response.

This modular and sequential workflow ensures consistency in response generation and enables efficient coordination between the different architectural components described in the previous chapter.

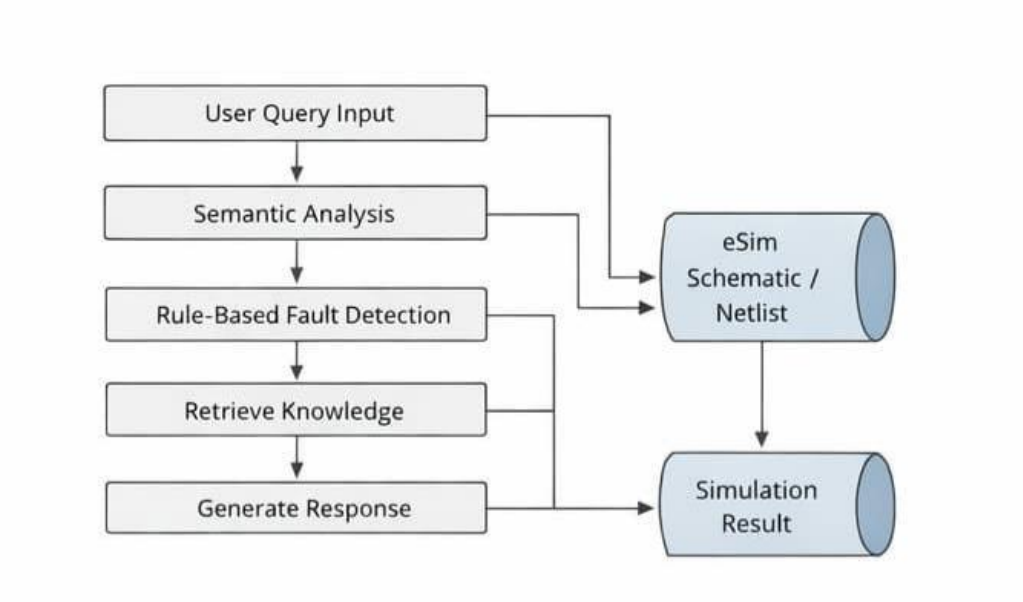

Figure 5.1: Workflow and control flow of the AI Chatbot

5.2 User Input Acquisition and Preprocessing

The first stage of the workflow involves acquiring input from the user through the Copilot interface. User inputs may be provided in natural language text form or, where supported, as visual inputs such as schematic images. This flexibility allows users to express issues in a manner that best reflects their design context and workflow preferences.

Once received, the input undergoes preprocessing to ensure suitability for downstream analysis. Textual inputs are normalized to remove ambiguity, standardize terminology, and identify key technical references. In the case of visual inputs, basic validation is performed to confirm compatibility and relevance. This preprocessing stage ensures that the subsequent interpretation and analysis stages operate on clean and well-structured input data.

5.3 Query Interpretation and Intent Identification

Following preprocessing, the system performs query interpretation to determine the underlying intent of the user's request. This stage involves analyzing the semantic content of the query to identify whether the user is seeking error explanation, debugging assistance, conceptual clarification, netlist evaluation, or general guidance related to circuit design and simulation.

By identifying intent, the Copilot can route the query to appropriate processing pathways and select relevant knowledge sources. This classification step is essential for ensuring that responses are focused, context-aware, and aligned with user expectations. It also enables the system to differentiate between conceptual questions and problem-specific troubleshooting requests.

5.4 Contextual Data Association

After intent identification, the Copilot associates the user query with available contextual data. This may include schematic information, netlist content, recent simulation logs, or previously established interaction context. By correlating user input with relevant design artifacts, the system gains a deeper understanding of the problem environment.

Contextual association enhances the precision of assistance by grounding responses in the user's current design state. This step is particularly important for resolving simulation errors, as it allows the Copilot to relate diagnostic feedback to specific schematic or netlist elements rather than providing generic explanations.

5.5 Knowledge Retrieval and Reasoning Process

Once sufficient context has been established, the system initiates the knowledge retrieval and reasoning phase. Relevant documentation fragments, known error patterns, and troubleshooting guidelines are retrieved from the local knowledge base based on query intent and contextual cues.

The intelligence layer then synthesizes this retrieved information with the user's query context to generate coherent explanations and recommendations. This reasoning process ensures that responses are both technically accurate and practically applicable. By combining structured knowledge with language-based reasoning, the Copilot delivers guidance that is tailored to the specific problem scenario.

5.6 Response Generation and Presentation

The final stage of the workflow involves generating and presenting the response to the user. The Copilot constructs responses in clear and structured language, emphasizing

actionable steps, explanatory clarity, and relevance to the user's design task. Where appropriate, responses may include conceptual explanations, step-by-step debugging suggestions, or references to relevant design principles.

The generated response is displayed within the Copilot interface in a format that is easy to read and apply. This presentation strategy minimizes cognitive load and enables users to quickly integrate the suggested guidance into their workflow. By maintaining consistency in response structure and tone, the system fosters a reliable and user-friendly assistance experience.

5.7 Iterative Interaction and Control Flow Continuity

The eSim Copilot supports iterative interaction, allowing users to refine queries, seek clarification, or address follow-up issues based on previous responses. Each interaction cycle builds upon earlier context, enabling progressive problem resolution and deeper understanding.

This iterative control flow ensures continuity across interactions and supports complex troubleshooting scenarios that require multiple steps. By maintaining contextual awareness and structured control flow, the Copilot facilitates efficient and systematic assistance throughout the circuit design and simulation lifecycle.

Chapter 6

Knowledge Base and Retrieval Framework

6.1 Role of the Knowledge Base in the AI Chatbot

The knowledge base forms the foundation of the AI Chatbot's assistance capabilities. It serves as the primary source of structured information used to generate accurate, relevant, and context-aware responses to user queries. In the context of circuit design and simulation, the knowledge base encapsulates domain-specific information related to eSim usage, SPICE-based simulation principles, common error patterns, and troubleshooting guidelines.

Unlike generic conversational systems, the AI Chatbot is designed to operate within a specialized technical domain. Therefore, the quality and organization of its knowledge base directly influence the usefulness and reliability of the assistance provided. By grounding responses in curated and tool-specific information, the system ensures that guidance remains aligned with established eSim workflows and simulation constraints.

The knowledge base supports offline operation by storing all required information locally, enabling uninterrupted access to assistance in environments with limited or no internet connectivity.

6.2 Knowledge Sources and Content Organization

The knowledge base is constructed using multiple authoritative sources relevant to the eSim simulation environment. These sources include official eSim documentation, NgSpice manuals, simulation guidelines, frequently encountered error descriptions, and domain-specific best practices derived from educational and practical usage scenarios.

To facilitate efficient retrieval, the collected information is organized into semantically coherent units. Each unit represents a distinct concept, procedure, or error pattern and is associated with descriptive metadata. This structured organization allows the system to identify and retrieve the most relevant information fragments in response to a given query.

By maintaining modular and well-labeled content segments, the knowledge base supports scalability and ease of maintenance. New documentation updates or additional troubleshooting cases can be incorporated without disrupting existing retrieval mechanisms.

6.3 Query-to-Knowledge Mapping Mechanism

When a user submits a query, the AI Chatbot initiates a query-to-knowledge mapping process. This process involves analyzing the semantic content of the query to determine its relevance to specific knowledge base entries. Key terms, contextual cues, and inferred intent are used to establish associations between the user query and stored knowledge fragments.

This mapping mechanism enables the system to narrow down the search space and focus on information that is most likely to address the user's concern. By prioritizing relevance over breadth, the AI Chatbot avoids generating overly generic responses and instead delivers targeted guidance that reflects the user's specific problem context.

Effective query-to-knowledge mapping is essential for maintaining response accuracy, particularly when dealing with ambiguous or partially specified user queries.

6.4 Retrieval-Augmented Response Generation

The AI Chatbot employs a retrieval-augmented approach to response generation, in which retrieved knowledge fragments are combined with language-based reasoning to construct meaningful explanations. Rather than relying solely on generative capabilities, the system

grounds its responses in retrieved documentation and known issue patterns.

This approach enhances factual accuracy and reduces the likelihood of incorrect or misleading guidance. By integrating retrieved information directly into the response generation process, the AI Chatbot ensures that explanations remain consistent with simulator behavior and documented practices.

Retrieval-augmented response generation also improves transparency, as users receive explanations that can be traced back to established documentation and simulation principles.

6.5 Handling of Domain-Specific Error Patterns

A significant portion of the knowledge base is dedicated to domain-specific error patterns commonly encountered in circuit simulation. These include connectivity issues, missing or invalid component definitions, improper grounding, unsupported device models, and numerical convergence problems.

Each error pattern is associated with descriptive explanations, potential causes, and recommended corrective actions. When the AI Chatbot detects a query related to such errors, it retrieves the corresponding knowledge entries and generates structured guidance tailored to the identified issue.

This targeted handling of error patterns supports efficient troubleshooting and helps users develop a deeper understanding of simulator behavior and circuit modeling principles.

6.6 Offline Retrieval and Performance Considerations

All retrieval operations are performed locally to maintain offline functionality and predictable performance. The knowledge base is indexed in a manner that supports efficient lookup and minimizes response latency. By avoiding dependence on external services, the system ensures consistent availability and responsiveness across different deployment environments.

Local retrieval also enhances data privacy and security, as user queries and circuit-related information remain confined to the user's system. This design choice aligns with the broader objective of supporting secure and self-contained educational and research workflows.

Overall, the knowledge base and retrieval framework play a central role in enabling the AI Chatbot to deliver accurate, context-aware, and reliable assistance within the eSim simulation environment.

Chapter 7

Semantic Representation and Matching

7.1 Importance of Semantic Understanding in the AI Chatbot

Effective assistance in circuit design and simulation requires the ability to interpret user queries beyond surface-level keyword matching. Users often describe problems using informal language, partial descriptions, or domain-specific terminology that may not directly correspond to documentation phrasing. Therefore, semantic understanding plays a critical role in enabling the AI Chatbot to accurately capture the intent and context of user queries.

Semantic representation allows the system to model the meaning of user inputs in a structured form that reflects conceptual relationships rather than exact lexical matches. By focusing on meaning, the AI Chatbot can recognize variations in phrasing, resolve ambiguity, and relate user queries to relevant knowledge base entries even when terminology differs. This capability is essential for delivering accurate and context-aware assistance within a technical domain such as circuit simulation.

7.2 Semantic Encoding of User Queries

When a user submits a query, the AI Chatbot converts the textual input into a semantic representation using locally deployed language models. This process involves encoding the query into a numerical or vector-based representation that captures semantic relationships

between words, phrases, and technical concepts.

Semantic encoding enables the system to preserve contextual information, such as the relationship between components, error descriptions, and simulation behavior. By representing queries in this form, the AI Chatbot can compare user input against stored knowledge in a manner that reflects conceptual similarity rather than strict textual correspondence.

This encoding process forms the foundation for effective matching and retrieval, particularly when dealing with diverse user expressions and complex technical terminology.

7.3 Representation of Knowledge Base Content

In parallel with query encoding, the content stored within the knowledge base is also represented semantically. Documentation fragments, error descriptions, and troubleshooting guidelines are processed to generate corresponding semantic representations. These representations capture the underlying meaning of each knowledge unit and its relationship to circuit simulation concepts.

By maintaining semantic representations for both queries and knowledge entries, the system establishes a common comparison space. This approach ensures that the AI Chatbot can identify relevant information even when there is no direct keyword overlap between the user query and stored content.

The semantic representation of knowledge base content is updated as new information is incorporated, ensuring consistency and relevance across retrieval operations.

7.4 Similarity Matching and Relevance Ranking

Once semantic representations are available, the AI Chatbot performs similarity matching to identify the most relevant knowledge base entries for a given user query. This involves computing similarity measures between the semantic representation of the query and those of stored knowledge units.

Based on computed similarity scores, the system ranks candidate knowledge entries in order of relevance. This ranking process ensures that the most contextually appropriate information is prioritized during response generation. By focusing on semantic similarity, the system reduces the likelihood of irrelevant or tangential responses.

Relevance ranking also supports efficient retrieval by limiting the number of knowledge entries considered during the reasoning phase, thereby improving response quality and performance.

7.5 Handling Ambiguity and Partial Queries

User queries may sometimes be incomplete, ambiguous, or underspecified, particularly when users are uncertain about the nature of the problem they are encountering. The semantic matching framework addresses this challenge by identifying multiple plausible interpretations and retrieving a small set of closely related knowledge entries.

In such cases, the AI Chatbot generates responses that either provide generalized guidance or request additional clarification from the user. This adaptive behavior ensures that the system remains helpful even when precise problem descriptions are unavailable.

By leveraging semantic relationships rather than exact matches, the system maintains robustness in the face of variability and uncertainty in user input.

7.6 Integration with Retrieval and Reasoning Workflow

Semantic representation and matching are tightly integrated with the knowledge retrieval and reasoning workflow described in earlier chapters. The matching process serves as the bridge between user input and knowledge retrieval, ensuring that relevant information is supplied to the reasoning layer for response generation.

This integration enables the AI Chatbot to combine semantic relevance with contextual reasoning, resulting in explanations that are both accurate and meaningful. By maintaining a cohesive semantic framework across system components, the chatbot delivers consistent and reliable assistance throughout the circuit design and simulation process.

Overall, semantic representation and matching form a foundational capability that enhances interpretability, flexibility, and effectiveness of the AI Chatbot within the eSim environment.

Chapter 8

Rule-Based Fault Identification

8.1 Motivation for Rule-Based Analysis

While semantic understanding and knowledge retrieval enable flexible interpretation of user queries, many circuit simulation errors arise from well-defined and recurring structural issues. Such issues can be effectively identified through deterministic rules derived from circuit design principles, simulator constraints, and established best practices. Rule-based fault identification therefore complements semantic reasoning by providing precise and reliable detection of common design and configuration errors.

In the context of eSim, rule-based analysis allows the AI Chatbot to systematically examine circuit representations and simulation-related information to identify faults that may lead to incorrect or failed simulations. By leveraging explicit rules, the system can generate consistent and explainable diagnostics that are aligned with simulator behavior.

8.2 Definition of Fault Identification Rules

Fault identification rules are formalized representations of known error conditions expressed as logical constraints or pattern-matching criteria. These rules are constructed based on common failure modes observed in circuit simulation workflows, such as missing reference nodes, incomplete component definitions, invalid parameter values, and unsupported device models.

Each rule specifies a set of conditions that must be satisfied for a particular fault to be detected. When these conditions are met, the rule triggers a corresponding diagnostic outcome. This structured representation enables the system to perform systematic checks

across different circuit artifacts and simulation inputs.

The use of clearly defined rules ensures transparency in fault detection and facilitates easier verification and maintenance of the rule set.

8.3 Application of Rules to Circuit Artifacts

The AI Chatbot applies fault identification rules to various circuit artifacts, including schematic representations, netlists, and simulation configuration data. By analyzing these artifacts, the system can detect inconsistencies and violations of design constraints before or during simulation execution.

For schematic-level analysis, rules are used to verify connectivity, component placement, and reference consistency. At the netlist level, the system examines component declarations, model references, and simulation directives to ensure compliance with expected syntax and simulator requirements.

This multi-level application of rules enables comprehensive fault detection and supports early identification of issues that might otherwise manifest as cryptic simulation errors.

8.4 Classification of Detected Faults

Detected faults are classified into predefined categories to facilitate structured analysis and response generation. Common categories include connectivity faults, parameter specification errors, model-related issues, and convergence-related problems. This classification allows the AI Chatbot to tailor its responses based on the nature and severity of the identified fault.

By categorizing faults, the system can prioritize critical issues that prevent simulation execution while also highlighting less severe warnings that may affect result accuracy. This structured classification supports more effective troubleshooting and improves user understanding of the underlying problem.

8.5 Generation of Corrective Recommendations

Once a fault is identified and classified, the AI Chatbot generates corrective recommendations based on the corresponding rule definition. These recommendations are formulated

to guide users toward resolving the detected issue in a systematic manner.

Corrective guidance may include suggestions to modify component parameters, establish missing connections, include required model definitions, or adjust simulation settings. In addition to actionable steps, the system provides explanatory context to help users understand why the fault occurred and how the proposed fix addresses the issue.

This combination of detection, explanation, and guidance reinforces learning and reduces the likelihood of recurring errors.

8.6 Integration with Semantic and Retrieval-Based Assistance

Rule-based fault identification operates in conjunction with semantic representation and retrieval-based assistance mechanisms. While rule-based analysis excels at detecting specific, well-defined faults, semantic reasoning supports interpretation of user queries and broader contextual understanding.

By integrating these complementary approaches, the AI Chatbot delivers robust and comprehensive assistance. Deterministic rules provide reliable fault detection, while semantic and retrieval-based components enrich explanations and adapt responses to user intent. This hybrid strategy enhances the overall effectiveness and reliability of the assistance provided within the eSim environment.

Overall, rule-based fault identification serves as a crucial component of the AI Chatbot, enabling precise diagnostics, structured guidance, and improved user support during circuit design and simulation activities.

Chapter 9

Static Schematic and Netlist Analysis

9.1 Purpose of Static Analysis in Circuit Simulation

Static analysis refers to the examination of circuit representations without executing a simulation. In the context of eSim, static schematic and netlist analysis plays a crucial role in identifying design inconsistencies, structural errors, and configuration issues prior to simulation execution. By detecting problems at an early stage, static analysis helps reduce simulation failures and improves overall design efficiency.

The AI Chatbot leverages static analysis to evaluate circuit correctness based on predefined rules, structural constraints, and modeling conventions. This approach enables proactive identification of faults that may otherwise surface only during simulation, where diagnostic feedback can be indirect or difficult to interpret.

9.2 Schematic-Level Static Analysis

Schematic-level analysis involves examining the graphical representation of a circuit to assess component placement, connectivity, and reference consistency. The AI Chatbot analyzes the schematic structure to verify that components are properly connected, essential reference nodes such as ground are present, and signal paths follow expected design conventions.

Through this analysis, the system can identify issues such as floating nodes, unconnected

pins, missing power references, and unintended short circuits. Detecting these faults at the schematic level allows users to correct design errors before they propagate to netlist generation and simulation execution.

Schematic-level static analysis also supports validation of circuit topology by ensuring that hierarchical blocks and subcircuits are interconnected correctly. This capability is particularly valuable for complex designs where visual inspection alone may not reveal subtle connectivity issues.

9.3 Netlist-Level Static Analysis

Netlist-level analysis focuses on the textual representation of the circuit that is provided to the SPICE-based simulation engine. Netlists encode component declarations, node assignments, model references, and simulation directives in a compact and formal syntax. While essential for simulation, netlists can be challenging for users to interpret directly.

The AI Chatbot performs static analysis of netlists by examining their structure and content to ensure compliance with simulator requirements. This includes verifying the presence of required component parameters, validating model definitions, checking node references, and identifying unsupported or conflicting directives.

By analyzing the netlist prior to simulation, the system can detect issues such as missing model files, incorrect parameter assignments, syntax inconsistencies, and ambiguous node naming. Early identification of such issues helps prevent simulation errors and reduces debugging effort.

9.4 Cross-Validation Between Schematic and Netlist

An important aspect of static analysis is cross-validation between schematic and netlist representations. The AI Chatbot compares schematic-level information with the generated netlist to ensure consistency between the two representations. This process helps identify discrepancies that may arise during schematic translation or netlist generation.

Cross-validation enables detection of mismatches such as missing components in the netlist, incorrect node mappings, or unintended alterations in circuit structure. By correlating schematic intent with netlist execution details, the system provides a more comprehensive validation of circuit integrity.

This alignment between schematic and netlist representations enhances reliability and ensures that the simulation accurately reflects the user's design intent.

9.5 Integration with Rule-Based Fault Identification

Static schematic and netlist analysis is closely integrated with the rule-based fault identification framework described in the previous chapter. Detected structural or syntactic issues are evaluated against predefined fault identification rules to classify errors and generate appropriate diagnostics.

This integration allows the AI Chatbot to not only identify faults but also explain their significance and suggest corrective actions. By combining static analysis with rule-based reasoning, the system delivers precise and actionable feedback that supports efficient troubleshooting.

9.6 Benefits of Static Analysis for Users

The inclusion of static schematic and netlist analysis within the AI Chatbot provides several benefits to users. Early detection of errors reduces the likelihood of simulation failure and minimizes time spent interpreting low-level diagnostic messages. Users gain clearer insight into design issues and can address them systematically before simulation execution.

From an educational perspective, static analysis promotes better understanding of circuit modeling principles and simulation requirements. By highlighting structural issues and their consequences, the system supports learning and encourages best practices in circuit design.

Overall, static schematic and netlist analysis enhances usability, reliability, and learning effectiveness within the eSim environment, reinforcing the AI Chatbot's role as a comprehensive support mechanism for circuit design and simulation.

Chapter 10

Multimodal Interaction

10.1 Motivation for Multimodal Interaction

Interaction with circuit simulation tools often involves multiple forms of representation, including textual descriptions, graphical schematics, and simulation outputs. Relying solely on text-based interaction can be limiting when users attempt to describe complex circuit topologies, component arrangements, or connectivity issues. Multimodal interaction addresses this limitation by enabling the AI Chatbot to accept and process multiple input modalities during user engagement.

By incorporating visual inputs alongside natural language queries, the system can better capture design context and reduce ambiguity in user communication. This approach enhances the accuracy and relevance of assistance provided, particularly in scenarios where textual descriptions alone are insufficient to convey circuit structure or intent.

10.2 Text-Based Interaction

Text-based interaction serves as the primary communication mode between users and the AI Chatbot. Users can submit queries in natural language to seek explanations, debugging assistance, or conceptual clarification related to circuit design and simulation. This mode of interaction is intuitive and accessible, allowing users to express problems without requiring formal syntax or predefined commands.

The AI Chatbot processes textual queries using semantic representation and intent identification mechanisms to interpret user intent and retrieve relevant knowledge. Text-based interaction supports iterative dialogue, enabling users to refine their queries and seek

follow-up clarification as needed. This conversational approach promotes ease of use and encourages exploratory learning.

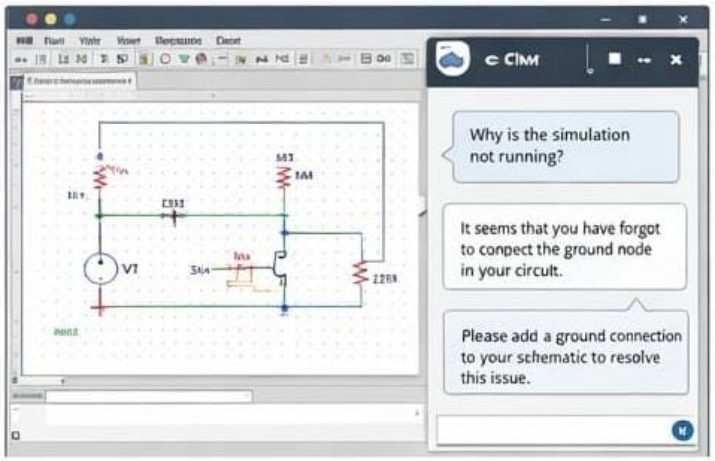

Figure 10.1: Text-based interaction with the AI Chatbot in eSim

10.3 Visual Input Processing

In addition to textual queries, the AI Chatbot supports visual inputs in the form of schematic images. Visual input processing allows the system to incorporate circuit structure and layout information into its analysis. By examining visual representations of schematics, the chatbot can infer component relationships, connectivity patterns, and overall circuit topology.

This capability is particularly valuable when users encounter difficulties describing circuit configurations through text alone. Visual inputs provide supplementary context that enables more precise interpretation of user intent and more accurate diagnostic guidance. The integration of visual context enhances the chatbot's ability to deliver targeted and design-aware assistance.

10.4 Integration of Multimodal Information

Effective multimodal interaction requires the integration of information obtained from different input modalities. The AI Chatbot combines insights derived from textual and visual inputs to form a unified representation of the user's problem context. This integrated representation supports more informed reasoning and response generation.

By correlating textual descriptions with visual schematic elements, the system can resolve ambiguities and generate responses that are better aligned with the user's design state. Multimodal integration thus improves the robustness and reliability of assistance, particularly for complex circuit designs involving multiple interacting components.

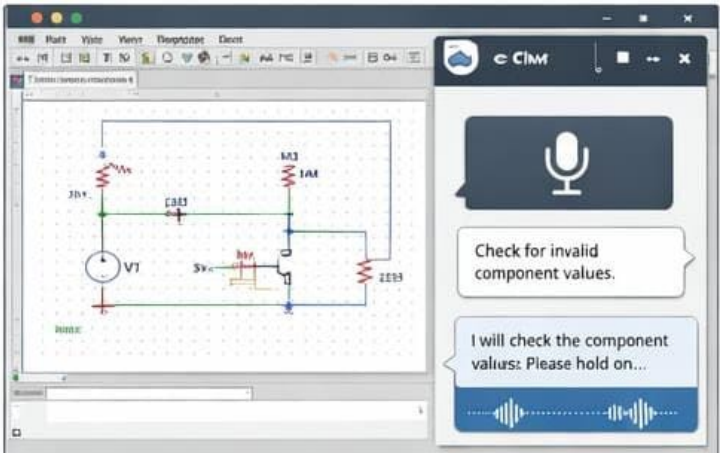

Figure 10.2: Voice-based interaction supported by the AI Chatbot

10.5 Use Cases and Interaction Scenarios

Multimodal interaction supports a range of practical use cases within the eSim environment. For example, users can submit a schematic image along with a textual query requesting verification of circuit connectivity or identification of potential design faults. The AI Chatbot can analyze the visual input in conjunction with the query to provide focused guidance.

Another common scenario involves troubleshooting simulation errors where visual context helps identify missing connections or incorrect component placement. By enabling such interaction scenarios, multimodal support enhances user productivity and reduces the time required to diagnose and resolve issues.

10.6 Impact on Usability and Learning

The inclusion of multimodal interaction significantly improves usability by accommodating diverse user preferences and workflows. Users can interact with the AI Chatbot using the modality that best suits their current task, thereby reducing communication barriers and cognitive effort.

From a learning perspective, multimodal interaction supports deeper understanding by allowing users to connect visual circuit representations with textual explanations and conceptual reasoning. This integrated approach enhances comprehension and reinforces learning outcomes, particularly for students and novice users.

Overall, multimodal interaction strengthens the effectiveness of the AI Chatbot as an intelligent support system within the eSim environment by enabling richer, more intuitive, and context-aware user engagement.

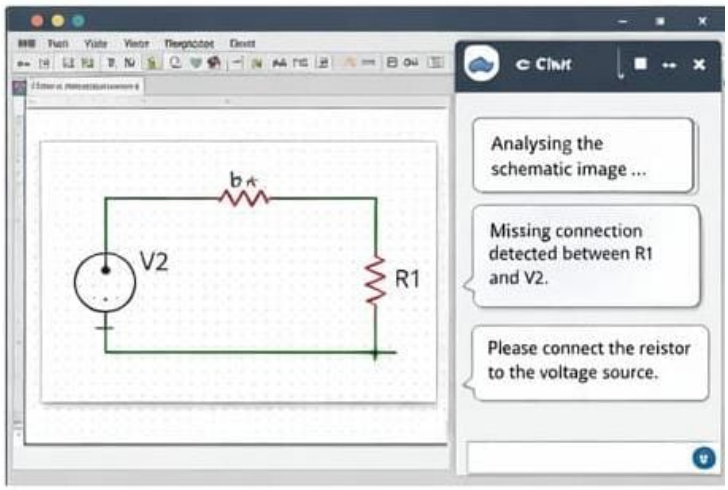

Figure 10.3: Image-based schematic input for analysis by the AI Chatbot

Chapter 11

Offline Operation and Performance Considerations

11.1 Rationale for Offline Operation

Offline operation is a fundamental design requirement of the AI Chatbot integrated with the eSim environment. In many academic institutions and laboratory settings, reliable internet connectivity cannot be assumed at all times. Dependence on cloud-based services or external APIs can therefore limit usability and disrupt learning or experimentation workflows.

By ensuring that all core functionalities operate without requiring network access, the AI Chatbot provides consistent and uninterrupted assistance. Offline capability supports equitable access to technical guidance, particularly in controlled environments where network usage may be restricted due to security or administrative policies. This design choice aligns with the broader objectives of the FOSSEE initiative, which emphasizes accessibility, openness, and self-reliance.

11.2 Local Execution of Core Components

All major components of the AI Chatbot are executed locally on the user's system. These components include query preprocessing, semantic representation, knowledge retrieval, rule-based fault identification, and response generation. Local execution eliminates dependency on external servers and ensures predictable system behavior.

Running all processes locally also reduces latency associated with remote communication. Users receive immediate feedback during debugging and design activities, which is critical for maintaining workflow continuity. Furthermore, local execution ensures that sensitive circuit designs and user interactions remain confined to the local environment, enhancing data privacy and security.

11.3 Resource Utilization and Efficiency

Offline operation necessitates careful consideration of system resource usage. The AI Chatbot is designed to operate efficiently within the computational constraints of typical academic and laboratory systems. Lightweight models, optimized retrieval mechanisms, and modular processing pipelines are employed to balance performance with resource consumption.

Memory usage is managed through structured storage of the knowledge base and efficient indexing techniques that support fast retrieval. Computational workloads are distributed across processing stages to avoid bottlenecks and ensure smooth interaction, even during extended usage sessions.

These efficiency considerations enable the AI Chatbot to function reliably on systems with modest hardware capabilities, thereby broadening its applicability across diverse deployment environments.

11.4 Response Time and User Experience

System responsiveness is a critical factor in user acceptance and usability. The AI Chatbot is designed to minimize response time by performing all inference and retrieval tasks locally and avoiding unnecessary computation. Preprocessing, semantic matching, and rule evaluation are optimized to ensure timely generation of responses.

Consistent and predictable response times contribute to a smoother user experience, particularly during iterative debugging workflows where users may submit multiple queries in succession. By maintaining low latency, the system supports natural conversational interaction and reduces cognitive interruption during design tasks.

11.5 Scalability and Maintainability Considerations

Although the AI Chatbot operates offline, its architecture is designed to support scalability in terms of functionality and knowledge content. The modular organization of system components allows new rules, documentation entries, and analysis capabilities to be incorporated without requiring architectural restructuring.

Maintainability is further supported by clear separation of concerns between knowledge storage, retrieval logic, semantic processing, and rule-based analysis. This design facilitates updates, debugging, and future enhancements, ensuring that the system can evolve alongside changes in the eSim platform and simulation practices.

11.6 Trade-offs and Design Implications

While offline operation offers significant advantages in accessibility and reliability, it also introduces certain trade-offs. Local execution places constraints on model complexity and storage capacity compared to cloud-based solutions. These limitations require careful selection and optimization of models and knowledge representations.

However, the design prioritizes robustness, transparency, and user control over reliance on external infrastructure. By accepting these trade-offs, the AI Chatbot achieves a balance between performance, usability, and independence, making it well-suited for educational and research-focused environments.

Overall, offline operation and performance optimization are integral to the effectiveness of the AI Chatbot within the eSim ecosystem. These considerations ensure that the system delivers reliable, efficient, and accessible assistance, reinforcing its role as a practical support tool for circuit design and simulation.

Chapter 12

Implementation Details

12.1 Overview of the Implementation Strategy

The implementation of the AI Chatbot was guided by the architectural and workflow principles discussed in earlier chapters. The primary objective was to develop a modular, maintainable, and offline-capable system that integrates seamlessly with the eSim environment. Emphasis was placed on preserving the integrity of the existing simulation framework while introducing intelligent assistance as an auxiliary support layer.

The implementation follows a layered approach in which individual components are developed independently and interconnected through well-defined interfaces. This strategy supports flexibility, ease of testing, and future extensibility.

12.2 Technology Stack and Development Environment

The AI Chatbot is implemented using open-source technologies to align with the philosophy of the FOSSEE initiative. The core logic is developed in Python, which offers extensive support for natural language processing, data handling, and integration with simulation tools.

Supporting libraries are used for text processing, semantic encoding, and rule-based analysis. Data storage for the knowledge base is implemented using lightweight file-based or embedded database formats to ensure efficient local access. The development and testing environment consists of standard desktop systems representative of typical academic and

laboratory setups.

12.3 Implementation of the Knowledge Base

The knowledge base is implemented as a structured collection of documentation fragments, error descriptions, and troubleshooting guidelines relevant to eSim and SPICE-based simulation. Each knowledge entry is stored in a modular format that facilitates indexing, retrieval, and update operations.

Metadata such as keywords, error categories, and contextual tags are associated with each entry to support efficient query-to-knowledge mapping. This structured representation enables scalable expansion of the knowledge base while maintaining consistency across retrieval operations.

12.4 Semantic Processing and Matching Implementation

Semantic representation of user queries and knowledge base entries is implemented using locally deployed language models. These models convert textual inputs into semantic representations that capture conceptual relationships between terms and technical constructs.

Similarity matching algorithms are employed to compare user queries with stored knowledge representations. Based on similarity scores, relevant knowledge entries are selected for further reasoning. This implementation ensures robust matching even when user queries differ in phrasing or terminology from documented content.

12.5 Rule-Based Fault Detection Mechanism

The rule-based fault identification module is implemented as a collection of logical rules derived from known circuit simulation error patterns. Each rule encodes conditions related to schematic structure, netlist syntax, or simulation configuration.

During analysis, circuit artifacts are evaluated against these rules to detect violations. When a rule is triggered, the corresponding fault category and corrective guidance are

recorded. This deterministic approach ensures reliable and explainable fault detection, complementing semantic reasoning.

12.6 Integration of Static Analysis Components

Static schematic and netlist analysis components are implemented to operate prior to simulation execution. These components parse circuit representations and validate structural and syntactic constraints without invoking the simulation engine.

By integrating static analysis results with rule-based fault detection, the system can identify and report issues early in the workflow. This integration enhances diagnostic precision and reduces reliance on post-simulation error messages.

12.7 Multimodal Input Handling

Support for multimodal interaction is implemented through separate processing pipelines for textual and visual inputs. Textual inputs are processed through the semantic and retrieval framework, while visual inputs are validated and analyzed to extract schematic-level information.

The results from both pipelines are combined to form a unified representation of the user's problem context. This integration enables the AI Chatbot to generate responses that account for both linguistic and visual information.

12.8 Response Generation and User Interface Integration

Response generation is implemented using a combination of retrieved knowledge, rule-based diagnostics, and semantic reasoning. Responses are structured to emphasize clarity, actionable guidance, and conceptual explanation.

The AI Chatbot interface is integrated with the eSim environment in a manner that allows users to access assistance without leaving the simulation workflow. Care is taken to present responses in a readable and organized format that supports efficient application of suggested guidance.

12.9 Testing and Validation Approach

The implementation is validated through a series of test scenarios representing common user queries and simulation errors. These tests assess the correctness of fault detection, relevance of retrieved information, and clarity of generated responses.

Validation is performed across different circuit types and complexity levels to ensure robustness and consistency. Feedback from testing is used to refine rules, improve knowledge entries, and enhance overall system performance.

Overall, the implementation details reflect a balance between technical rigor, usability, and maintainability, resulting in an AI Chatbot that effectively supports circuit design and simulation within the eSim environment.

Chapter 13

Results and Evaluation

13.1 Evaluation Objectives

The evaluation of the AI Chatbot focuses on assessing its effectiveness in supporting users during circuit design and simulation tasks within the eSim environment. The primary objectives of the evaluation are to determine the accuracy of fault identification, relevance of generated guidance, responsiveness of the system under offline operation, and its overall impact on usability and learning.

The evaluation is designed to reflect realistic usage scenarios encountered by students and practitioners, including debugging common simulation errors, interpreting circuit behavior, and resolving configuration issues. Emphasis is placed on qualitative and functional assessment rather than numerical benchmarking, in alignment with the educational objectives of the project.

13.2 Test Scenarios and Use Cases

To evaluate system performance, a set of representative test scenarios was constructed based on frequently observed issues in circuit simulation workflows. These scenarios include missing ground references, unconnected nodes, incorrect component parameter values, undefined device models, and syntactic errors in netlists.

Each scenario was executed by interacting with the AI Chatbot through natural language queries and, where applicable, schematic inputs. The chatbot's responses were examined to verify whether the detected issues aligned with the underlying fault and whether the suggested corrective actions were appropriate and actionable.

The selected use cases span varying levels of circuit complexity, ensuring that the evaluation captures system behavior across both simple and moderately complex designs.

13.3 Effectiveness of Fault Identification

The rule-based fault identification and static analysis components demonstrated reliable detection of common schematic-level and netlist-level issues. In cases involving structural faults such as missing connections or invalid references, the AI Chatbot consistently identified the root cause and classified the fault correctly.

The integration of semantic interpretation further enhanced detection accuracy by enabling the system to associate user-described symptoms with known error patterns. This combination of deterministic rules and semantic reasoning resulted in precise and explainable diagnostics, reducing ambiguity in system responses.

13.4 Quality of Generated Guidance

The quality of the chatbot's responses was evaluated based on clarity, relevance, and usefulness. Generated guidance typically included both corrective suggestions and explanatory context, enabling users to understand why an issue occurred and how to resolve it.

Responses were observed to align closely with eSim and SPICE simulation practices, ensuring technical correctness. The structured presentation of guidance supported efficient troubleshooting and minimized the need for external documentation consultation.

Overall, the AI Chatbot demonstrated the ability to provide meaningful assistance that supports both immediate problem resolution and longer-term learning.

13.5 Performance Under Offline Operation

Performance evaluation under offline conditions confirmed that the AI Chatbot remained fully functional without network connectivity. All core operations, including query processing, knowledge retrieval, semantic matching, and rule evaluation, were executed locally with consistent response times.

Offline operation did not introduce noticeable latency during typical interaction scenarios.

This confirms that the system meets its design objective of providing reliable assistance in constrained environments such as academic laboratories and standalone installations.

13.6 Impact on Usability and Learning

From a usability perspective, the AI Chatbot reduced the cognitive effort required to diagnose and resolve simulation issues. By presenting context-aware explanations and targeted recommendations, the system lowered entry barriers for new users and supported more systematic debugging practices.

In educational contexts, the chatbot encouraged exploratory learning by enabling users to ask conceptual questions and receive immediate clarification. This interactive support model promotes deeper understanding of circuit behavior and simulation principles compared to static documentation alone.

13.7 Limitations Observed During Evaluation

While the evaluation results were largely positive, certain limitations were observed. The effectiveness of assistance depends on the completeness of the knowledge base and the coverage of predefined fault identification rules. Rare or highly specialized simulation issues may require manual intervention or future extension of the system.

Additionally, while multimodal input enhances contextual understanding, the interpretation of complex schematics remains limited by the granularity of visual analysis. These limitations represent opportunities for future refinement rather than fundamental constraints.

13.8 Summary of Results

The evaluation demonstrates that the AI Chatbot effectively supports circuit design and simulation activities within the eSim environment. It provides accurate fault identification, clear and relevant guidance, and reliable offline performance. The system enhances usability, supports learning, and aligns with the objectives of open-source educational tool development promoted by the FOSSEE initiative.

Overall, the results validate the feasibility and practical value of integrating intelligent

assistance into open-source circuit simulation workflows.

Chapter 14

Conclusion and Future Work

14.1 Conclusion

This project focused on the design and development of an AI Chatbot to enhance user support within the eSim circuit simulation environment. The primary objective was to address the challenges faced by students and practitioners during circuit design, simulation, and debugging by providing intelligent, context-aware assistance that operates entirely offline.

Through a modular and extensible system architecture, the AI Chatbot integrates semantic understanding, knowledge base retrieval, rule-based fault identification, and static analysis of schematics and netlists. These capabilities enable the system to interpret user queries, identify common simulation issues, and generate clear and actionable guidance aligned with established eSim and SPICE simulation practices.

The results and evaluation demonstrate that the AI Chatbot effectively improves usability by reducing diagnostic effort, minimizing reliance on external documentation, and supporting systematic troubleshooting. Its offline operation ensures reliable access to assistance in academic laboratories and constrained environments, reinforcing the objectives of accessibility and self-reliance promoted by the FOSSEE initiative.

Overall, the project validates the feasibility and practical value of integrating intelligent assistance into open-source electronic design automation tools. The AI Chatbot serves not only as a debugging aid but also as a learning support mechanism that enhances conceptual understanding of circuit simulation workflows.

14.2 Future Work

While the current implementation provides robust and reliable assistance, several opportunities exist for future enhancement. Expanding the knowledge base to include a broader range of circuit types, advanced simulation scenarios, and specialized device models would further improve coverage and effectiveness. Continuous refinement of rule-based fault identification can also enhance detection of complex and less frequent error patterns.

Future work may explore deeper schematic and visual analysis techniques to improve interpretation of complex circuit layouts and hierarchical designs. Enhancements in multimodal processing could enable more detailed extraction of connectivity and component relationships from visual inputs.

Another potential direction involves adaptive learning mechanisms that allow the AI Chatbot to incorporate user feedback and evolving documentation into its knowledge base. This would support continuous improvement while maintaining offline operability.

Finally, integration of performance profiling, simulation result interpretation, and design optimization suggestions could extend the chatbot's role beyond troubleshooting to proactive design guidance. These enhancements would further strengthen the AI Chatbot as a comprehensive support system for circuit design education and research within the eSim ecosystem.

References

1. FOSSEE Project, Indian Institute of Technology Bombay. *Free and Open Source Software for Education (FOSSEE).* Available at: https://fossee.in

2. eSim Development Team. eSim, an open source electronicdesignautomation*eSim — Open Source Electronic Design Automation Tool.* Available at: https://esim.fossee.in

3. Ngspice Developers. *Ngspice User Manual and Reference Documentation.* Available at: https://ngspice.sourceforge.net

4. Ministry of Education, Government of India. *National Mission on Education through Information and Communication Technology (NMEICT).*

5. Sedra, A. S., and Smith, K. C. *Microelectronic Circuits.* Oxford University Press, 7th Edition.

6. Brown, S., and Vranesic, Z. *Fundamentals of Digital Logic with Verilog Design.* McGraw-Hill Education, 3rd Edition.