Winter Internship Report

On

eSim Software Development

Submitted by

Vivek Kumar

Under the guidance of

Prof. Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

March 13, 2025

Chapter 1

Acknowledgment

I take this opportunity to express my sincere gratitude to FOSSEE, IIT Bombay for providing me with the platform to work on the development of eSim. This internship has been an enriching journey, offering me valuable exposure to open-source EDA tools and their significance in circuit simulation and design. The hands-on experience I have gained has deepened my understanding of circuit modeling, simulation, verification, desktop application development, and frontend and backend software development, preparing me for future challenges in the field of electronics and software development.

I extend my heartfelt appreciation to Prof. Kannan M. Moudgalya for his support. His emphasis on open-source innovation has been truly inspiring, and I feel privileged to have been a part of this initiative.

A special note of thanks to my mentor, Mr. Sumanto Kar, for his invaluable assistance, constructive feedback, and mentorship. His expertise and willingness to help at every stage ensured that I could effectively navigate challenges and successfully complete my project tasks. His patience and technical insights have played a crucial role in enhancing my problem-solving skills.

This internship has been a remarkable learning experience, helping me develop not just technical skills but also a deeper appreciation for collaborative development in open-source projects. As I aspire to build a career in the software development industry, the knowledge and skills gained here will serve as a strong foundation for my professional growth. I am immensely grateful for this opportunity and look forward to applying my learnings in meaningful ways in the future.

Contents

Chapter 2

Introduction

2.1 FOSSEE

FOSSEE (Free and Open Source Software for Education) is an initiative by IIT Bombay aimed at promoting the use of open-source software in educational institutions. The project focuses on providing free and open-source tools for various domains like electronics, mathematics, simulation, and engineering design. FOSSEE encourages the adoption of open-source alternatives to expensive proprietary software, making tools for learning, research, and development more accessible to students, educators, and professionals. The initiative supports a range of tools, such as eSim for circuit design, Scilab for numerical computations, and other software for system modeling, simulation, and visualization. Through its efforts, FOSSEE has contributed significantly to making high-quality educational resources available to a global community while fostering the use of open-source software.

2.2 eSim

eSim is an open-source Electronic Design Automation (EDA) tool developed by FOSSEE, IIT Bombay, designed to assist engineers, students, and researchers in creating, simulating, and analyzing electronic circuits. It integrates several open-source tools like Ngspice for circuit simulation, KiCad for PCB design, Scilab for numerical computation, and Python for scripting and automation. eSim also supports mixed-signal simulation through NGHDL (combining Ngspice with GHDL) and system-level modeling with OpenModelica. With its completely free and open-source nature, eSim provides an accessible and cost-effective alternative to proprietary EDA tools, making it an ideal platform for learning and research in electronics design.

2.3 NgSpice

Ngspice is an open-source, powerful circuit simulator based on the SPICE (Simulation Program with Integrated Circuit Emphasis) model, used for simulating and analyzing electronic circuits. It supports a wide range of simulations, including

DC,AC, transient, and noise analysis, making it suitable for both analog and mixed-signal circuit designs. Ngspice uses a text-based input format for defining circuit components and their connections, which is simple to understand and modify. It is highly customizable and can be extended with new models and components, making it a flexible tool for research and development. Ngspice integrates well with other open-source tools, such as KiCad and eSim, and is widely used in educational and professional environments due to its cost-free nature and reliability.

2.4 KiCad

KiCad is an open-source PCB design tool that is widely used for designing electronic circuits, including schematic capture and PCB layout. In eSim, KiCad plays a vital role by providing users with the ability to design and lay out printed circuit boards (PCBs). It offers features like component libraries, schematic design, automatic routing, and Gerber file generation, which are essential for PCB manufacturing. By integrating KiCad, eSim enables a seamless workflow where users can design circuits and move on to the PCB layout stage without needing separate software tools. This integration enhances the overall design process, especially for students and researchers working on electronic projects.

Chapter 3

Problem Statement

Internship Focus: During my internship, I fixed the eSim Installation problems in Windows 11, enhanced the tool chain, and created a tool manager to handle eSim.

- Problem Definition: The main challenge was to fix the installation problems in Windows. It consisted of repackaging eSim and ensuring that no other files need to be imported by the user after installation.Tool chain was enhanced by adding dual plotting.
- Tool manager: Tool manager is a tool to manage the resources of eSim such as installed packages, libraries, dependencies and others. It allows the user to fetch the newest version and then update them.
- Dual Plotting: In eSim 4.0 the primary plotting mechanism was python while earlier ngSPice was used, dual plotting was introduced to help the users take the advantage of both type of plotting.
- Objective: The primary objective was to make eSim installation error free and minimal for users. Introduce dual plotting for error free plotting and tool manager to manage resources.
- Skills Gained: This experience helped me develop a deeper understanding of software development, desktop application, packaging of software, user experience and user interface.

Chapter 4

ToolManager

4.1 General Description

The Tool Manager is designed to automate the installation, updating, and management of essential tools and dependencies required for eSim. This manager ensures that required tools are compatible with the system environment, handles version control, and provides a user-friendly interface for monitoring installed tools and updates. By streamlining dependency management, the Tool Manager simplifies the user experience and ensures seamless integration of all necessary software components.

4.2 Problem Statement

The primary goal of the Tool Manager is to automate and simplify the management of essential tools required for eSim. The key challenges addressed by this module include:

- Reducing the manual effort required for installing and updating tools.
- Ensuring compatibility across different operating systems.
- Maintaining consistency in tool versions to prevent compatibility issues.
- Providing a user-friendly interface for better accessibility and management.

4.3 Features and Functionalities

4.3.1 Tool Installation Management

- **Automated Installation:** The Tool Manager automates the downloading and installation of essential tools and dependencies, including:
  - KiCad (for schematic capture and PCB design)
  - NgSpice (for circuit simulation)

  - Additional tools and dependencies specified in the configuration script

- **System Compatibility:** The Tool Manager ensures compatibility across multiple operating systems, including:
  - Linux (Ubuntu, Debian, Fedora, Arch, etc.)
  - Windows (Windows 10 and above)

- **Version Control:** The Tool Manager maintains a record of installed tool versions and ensures version consistency across multiple installations to avoid conflicts and errors.

4.3.2 Update and Upgrade System

- **Update Checks:** The Tool Manager periodically checks for updates to external tools and libraries, ensuring users have access to the latest stable versions.

- **Upgrade Functionality:** Users can update tools and dependencies seamlessly with a single command, eliminating the need for manual intervention.

- **Rollback Capability:** If an update causes issues, the Tool Manager allows rolling back to a previous version to maintain system stability.

4.3.3 User Interface

- **Logging:** All actions taken by the Tool Manager are logged, allowing users to review changes and troubleshoot issues if necessary.

- **Interface Options:** The Tool Manager provides both CLI and GUI options to cater to different user preferences and expertise levels.
  - CLI (Command Line Interface) for advanced users who prefer command-based interactions.
  - GUI (Graphical User Interface) for users who prefer a visual representation of installed tools and available updates.

- **Tool Management Overview:** Users can view installed tools, their versions, and any available updates through an intuitive dashboard.

4.4 How the Tool Manager Works

4.4.1 Modules and Their Responsibilities

The Tool Manager consists of several core modules, each responsible for a specific function:

- **main.py:** The central coordination script that orchestrates all other modules.

- **dependencies.json:** Stores the list of required dependencies and their respective versions.
- **dependencychecker.py:** Checks if the required dependencies are already installed on the system and retrieves their versions.
- **installmanager.py:** Handles the automated installation of tools and dependencies.
- **updatemanager.py:** Manages update checks and upgrades tools when newer versions are available.

4.4.2 Module Call Flow

The Tool Manager operates in the following sequence:

main.py → dependencies.json → dependencychecker.py → installmanager.py / updatemanager.py

4.5 Implementation Details

4.5.1 Installation Process

The Tool Manager follows these steps during installation:

1. Reads the list of required tools and dependencies from `dependencies.json`.
2. Checks if each tool is already installed using `dependencychecker.py`.
3. If a tool is missing, `installmanager.py` downloads and installs it automatically.
4. Verifies successful installation and logs the results.

4.5.2 Update Process

When an update is triggered, the Tool Manager performs the following steps:

1. Checks for available updates for installed tools.
2. Compares the current version with the latest available version.
3. If an update is found, prompts the user for confirmation before proceeding.
4. Downloads and installs the updated version.
5. Logs the update details and verifies successful installation.

4.6 Code Repository and Collaboration

Repository: https://github.com/092vk/eSim/tree/toolManager/toolManager
Collaborator: Pyae Sone

4.7 Future Enhancements

Several enhancements are planned for future versions of the Tool Manager:

- **Cross-Platform Support:** Improved support for macOS.
- **Dependency Graph:** Visualization of dependency relationships and conflicts.
- **Advanced Error Handling:** Better handling of installation failures and dependency conflicts.
- **Background Updates:** Automatic background update checks with optional notifications.
- **Plugin System:** Allowing third-party plugins for additional functionality.

4.8 Current Status

The Tool Manager is currently in the testing phase before its official integration into eSim. The testing process involves verifying installation workflows, update mechanisms, and system compatibility to ensure a smooth user experience. Feedback and improvements from testing will be incorporated before merging into the main eSim project.

4.9 Conclusion

The Tool Manager significantly simplifies tool management for eSim users by automating installation, updates, and version control. With a user-friendly interface and robust backend functionality, it enhances usability while ensuring software consistency. Ongoing development and future enhancements will further improve its capabilities, making eSim more accessible and efficient for all users.

Chapter 5

Dual Plotting

5.1 Introduction

Since version 2.4, eSim has been using Python's Matplotlib for generating simulation plots automatically once the netlist is created. However, in previous versions, eSim used NgSpice for plotting, which many users found reliable and preferable in certain scenarios. Due to resource constraints, the NgSpice-based plotting method was deprecated in favor of the Python-based approach.

Despite this change, several users have reported issues where the Python plotter fails or does not provide the desired output, leading to a demand for reinstating the NgSpice plotting method as an alternative. To address this concern, a solution was developed to reintroduce NgSpice plotting as an optional feature through a user-friendly toggle mechanism.

5.2 Problem Statement

The transition from NgSpice-based plotting to Python Matplotlib-based plotting in eSim 2.4 created the following challenges:

- The Python plotter occasionally fails to generate plots correctly.
- Some users prefer the NgSpice-based plotting method due to familiarity and better visualization capabilities.
- Earlier versions supported both methods, but the NgSpice method was removed to optimize resources.
- There was no built-in mechanism for users to switch between the two plotting methods.

Due to these challenges, a feature was needed that would allow users to select their preferred plotting tool dynamically during simulation.

5.3 Solution Proposed

To enhance user experience and restore flexibility, a toggle button was introduced to allow users to choose between Python and NgSpice-based plotting. This feature ensures that users can access their preferred plotting method seamlessly.

5.3.1 Feature Overview

- A pop-up box appears when the user clicks on **Simulate**.
- The pop-up contains the prompt: *"Do you want NgSpice Plots?"*
- Two options are provided: **Yes** and **No**.
- Selecting **Yes** enables NgSpice plots alongside Python plots.
- Selecting **No** keeps Python plotting as the default method.

This implementation ensures that users can dynamically switch between the two methods without requiring configuration file modifications or additional commands.

5.4 Expected Behavior

The toggle feature should function as follows:

- The user clicks **Simulate** to initiate circuit simulation.
- A pop-up appears, prompting the user to choose between NgSpice and Python plots.
- If **Yes** is selected, both NgSpice and Python plots are generated.
- If **No** is selected, only Python plots are displayed.
- The decision persists for the current simulation session but can be changed upon reinitiating the simulation process.

This implementation provides users with a smooth and intuitive way to select their preferred plotting method without disrupting the workflow.

5.5 Implementation Details

To implement this feature, modifications were made to the eSim codebase, particularly in the simulation trigger mechanism. The following steps outline the implementation:

5.5.1 Step 1: Creating the Toggle Pop-up

A pop-up window was integrated into the GUI simulation process. This pop-up asks the user whether they want to enable NgSpice plotting. The window contains a message, a **Yes** button, and a **No** button.

5.5.2 Step 2: Modifying the Simulation Flow

The simulation logic was updated to handle user input from the toggle pop-up. Based on the user's selection:

- If **Yes** is selected, the system triggers both Python and NgSpice plots.
- If **No** is selected, only the default Python plotting mechanism runs.

5.5.3 Step 3: Updating Configuration and Resource Handling

Since reintroducing NgSpice plotting could impact resource usage, careful handling was implemented to ensure efficient resource allocation. The feature does not run both plotters simultaneously unless explicitly requested by the user.

5.5.4 Key Code Modifications

- GUI logic was updated to include the toggle pop-up.
- Simulation scripts were modified to check for user selection before generating plots.
- NgSpice plotting functionality was reintroduced without affecting Python-based plotting.
- Error handling was improved to manage potential failures in either plotting method.

5.6 Implementation

5.7 Conclusion

The dual-plotting feature enhances the user experience by allowing users to choose their preferred plotting method dynamically. This implementation restores NgSpice plotting as an option without disrupting the default Python plotting mechanism. By providing a simple toggle, users gain flexibility while maintaining the benefits of modern visualization tools.

Future improvements could include:

- Remembering user preferences across sessions.

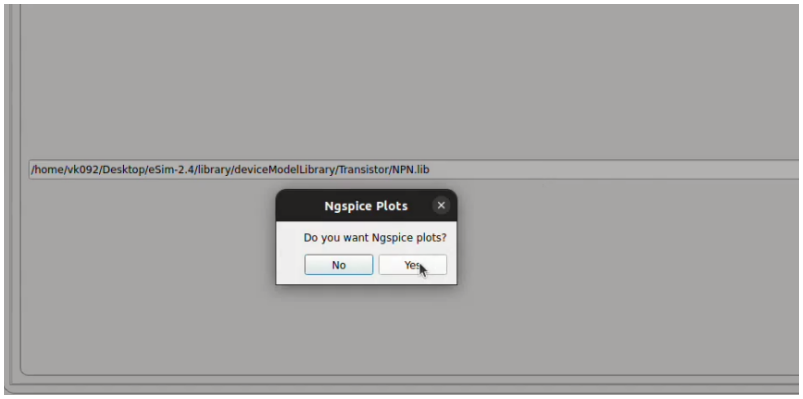

Figure 5.1: Dual Plotting Pop-up

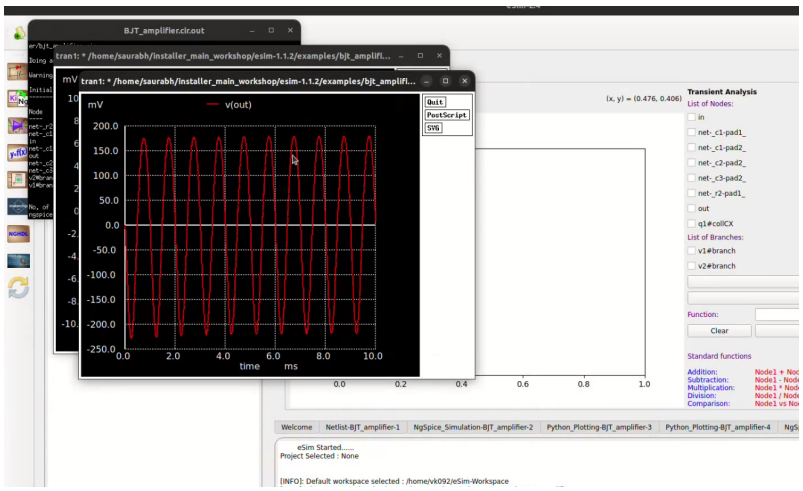

Figure 5.2: Dual Plotting

- Adding more customization options for plot formatting.
- Enhancing performance optimizations for dual plotting.

This feature marks a significant usability improvement in eSim, addressing user feedback and making circuit simulation more accessible and efficient.

Chapter 6

Fixing eSim Installation Problems for Windows

6.1 Introduction

eSim 2.4, when installed on Windows, encounters installation issues that require users to manually add essential files for proper functionality. These issues are documented in the installation instructions but cause inconvenience for users. The primary goal of this fix was to ensure a smooth installation process without requiring manual intervention.

6.2 Problem Statement

The installation of eSim 2.4 on Windows does not include all required files, leading to errors when using the software. Specifically, users need to manually download and add the following files:

- `TerminalUi.ui`
- `nghdl.exe`

These files are required for correct functionality but were not included in the packaged installer, causing confusion among users.

6.3 Major Issues

After installation, users must manually download and place these files in specific directories:

- **To the home directory (FOSSEE\eSim\folder):**
  - File: `TerminalUi.ui`
  - Download Link: https://github.com/FOSSEE/eSim/blob/master/src/frontEnd/Terminal

- **To the nghdl folder (FOSSEE\eSim\nghdl\src location):**
  - File: `nghdl.exe`
  - Download Link: https://drive.google.com/file/d/17MNCCq9cG6A7fnIH-4KMUMY-yb4rW9s4/view?usp=sharing

6.4 Steps to Reproduce

1. Install eSim 2.4 on Windows.
2. Attempt to generate plots or perform simulations.
3. Errors occur due to missing files.
4. Users have to manually download and add the missing files.

6.5 Expected Behavior

- eSim should install properly when the user runs the `.exe` file.
- No additional files should need to be manually added from an external source.
- All necessary dependencies should be included in the installation package.

6.6 Solution Implemented

The issue was resolved by repackaging eSim with all necessary files:

- **nghdl.exe:** Previously, the repackaging process did not include `nghdl.exe`, requiring users to manually add it. The issue was fixed by ensuring proper inclusion during packaging.
- **TerminalUi.ui:** This file was missing from the source package. Initially, the `spec` file was modified to include it, but later, it was bundled with other essential files such as `Readme.md` and `License.rtf` during packaging. This resolved the missing file issue.

After implementing these fixes, a newly packaged version of eSim was created that eliminates the need for manual intervention.

6.7 Future Plans

To ensure a stable release, the following steps will be taken:

- Conduct thorough testing of the newly packaged eSim to confirm the fixes.
- Deploy the updated version for production use.

- Update documentation to reflect the resolved issues and new installation process.

These measures will ensure a seamless installation experience for Windows users and prevent similar issues in future releases.

Chapter 7

Version Change Update

7.1 Introduction

During the upgrade from eSim 2.3 to 2.4, several configuration and frontend-related files were not updated. This resulted in inconsistencies where the version displayed within the software remained 2.3, leading to confusion among contributors. The issue was identified when contributors attempting to build eSim from the source noticed the outdated version information.

To resolve this issue, the required files were updated to reflect the correct version. These updates were crucial to maintain consistency in documentation, configuration, and frontend files, ensuring that all contributors have an accurate representation of the eSim version.

Note: These updates do not modify any functional aspects of eSim. The changes are strictly related to versioning and frontend consistency.

7.2 Problem Statement

The primary problem was that various configuration and frontend-related files were not properly updated during the transition from eSim 2.3 to 2.4. Due to this oversight, when repackaging eSim, the version still appeared as 2.3. This led to confusion among contributors and developers working with the source code. The inconsistency needed to be addressed to ensure all files correctly reflect the current version.

7.3 Files Updated

The following files were identified as requiring updates to reflect the new version:

- `master/version` - Updated the version string to 2.4.
- `master/conf.py` - Modified configuration settings to reflect the new version.
- `master/setup.py` - Ensured that package setup accurately represents version 2.4.

- `UserManual.py` - Updated documentation references to the latest version.
- `Appconfig.py` - Adjusted application configuration settings accordingly.
- `INSTALL` - Revised installation instructions to reference eSim 2.4.
- `README.md` - Corrected version references and updated general information.

7.4 Resolution and Implementation

A pull request was created to address the issue, updating all necessary files to reflect the correct version. These changes were thoroughly tested to ensure they did not impact the functional aspects of eSim. The verification process included:

- Checking the version displayed in the application.
- Ensuring all configuration files correctly referenced eSim 2.4.
- Confirming that documentation and installation instructions were aligned with the new version.

The issue was successfully resolved, and the changes were merged into the main repository.

7.5 Pull Request Details

The necessary updates were incorporated in the following pull request:

- Pull Request: https://github.com/FOSSEE/eSim/pull/297
- Author: 092vk
- Status: Merged

This pull request ensures that all required files are now consistent with the updated eSim 2.4 version, preventing any further confusion for contributors and developers.

7.6 Conclusion

By updating the version references across all relevant files, I have eliminated discrepancies related to versioning. These changes improve clarity for contributors and ensure that eSim 2.4 is properly represented in all documentation and configuration files. Future upgrades should follow a standardized checklist to prevent similar issues from occurring.

Chapter 8

Reference

- FOSSEE Project: https://fossee.in/
- eSim Documentation: https://esim.fossee.in/
- NgSpice: http://ngspice.sourceforge.net/
- KiCad: https://www.kicad.org/
- https://github.com/FOSSEE/eSim