eSim Semester Long Internship Report

On

eSim 2.5 for Ubuntu, Windows and Docker

Submitted by

Jayanth Tatineni
B.Tech in CSE (Cyber Security), GITAM University

Under the guidance of

Sumanto Kar
IIT Bombay

July 4, 2025

Acknowledgment

I sincerely extend my gratitude to the FOSSEE team for providing me with this significant opportunity to contribute to an open-source project like eSim. I am truly honored to have been selected for this internship.

I would like to express my heartfelt appreciation to my mentor, Sumanto Kar Sir, for his invaluable guidance, patience, and support throughout this journey.

This experience has strengthened my motivation to contribute to open-source projects while enhancing my skills and career.

Contents

Chapter 1

Introduction

1.1 FOSSEE: Promoting Open-Source Software for Education

The Free/Libre and Open Source Software for Education (FOSSEE) project is an initiative by IIT Bombay under the National Mission on Education through Information and Communication Technology (ICT), funded by the Ministry of Education, Government of India. The project aims to reduce dependency on proprietary software in academic and research institutions by promoting the adoption of Free and Open Source Software (FOSS) alternatives. [1]

FOSSEE works toward this goal through multiple initiatives:

- Development and Enhancement: Creating new open-source tools and improving existing ones to meet industry and academic needs.
- Workshops and Training: Conducting training programs and workshops to encourage open-source software adoption.
- Collaborations: Partnering with institutions, researchers, and professionals to integrate FOSS into mainstream education.
- Translation and Documentation: Converting open-source software documentation into multiple regional languages to enhance accessibility.

Through these efforts, FOSSEE ensures that high-quality software remains freely available, eliminating the financial barriers associated with proprietary tools and empowering individuals to learn, contribute, and innovate.

1.2 eSim: An Open-Source EDA Tool

One of FOSSEE's most significant contributions to the open-source ecosystem is eSim, a free and open-source Electronic Design Automation (EDA) tool for circuit design, simulation, analysis, and PCB design. Developed by IIT Bombay, eSim integrates multiple FLOSS tools to provide a complete design and simulation environment for electrical and electronics engineers. [2]

1.2.1 Key Features of eSim

eSim is designed to offer similar functionalities to commercially available EDA tools such as OrCAD, Xpedition, and HSPICE, but without the associated licensing costs. Some of its key features include:

- Schematic Capture: Users can create detailed circuit schematics using an interactive graphical interface.
- Simulation Support: Integration with NgSpice, allowing transient, AC, and DC circuit analysis.
- VHDL and Verilog Simulation: Powered by GHDL and Verilator, enabling mixed-mode simulations.
- PCB Layout and Design: Seamless integration with KiCad, a powerful open-source PCB design tool.
- Microcontroller Support: Offers features for designing embedded systems by integrating microcontrollers into circuits.
- Open-Source Licensing: Released under GPL, ensuring unrestricted access and modification.

1.2.2 Components Integrated with eSim

eSim brings together multiple open-source tools to create a comprehensive EDA suite:

- KiCad: Used for schematic capture and PCB design.
- NgSpice: A general-purpose circuit simulation program for analyzing AC, DC, and transient circuits.
- GHDL: A VHDL simulator that allows digital circuit design verification.
- Verilator: A fast Verilog simulator widely used for hardware verification.

By leveraging these tools, eSim provides an affordable, flexible, and powerful alternative to proprietary EDA software, making it a preferred choice for students, researchers, and professionals in circuit design and simulation.

While eSim continues to be a powerful tool for circuit design and simulation, its installation process across different Ubuntu versions has presented challenges, necessitating improvements to ensure seamless user experience. These issues are explored in the following section.

Chapter 2

Problem Statement

To enhance the usability, portability, and consistency of eSim across multiple system environments and Ubuntu versions.

2.1 Challenges in eSim

2.1.1 Issues Encountered Across Versions

Following the resolution of compatibility issues in Ubuntu 23.04, broader challenges impacting the overall user experience of eSim were observed. These challenges were primarily associated with version-specific behaviors, environmental inconsistencies, and the need for updated packages for the 2.5 release. The key issues identified were:

- **Version-Specific Script Limitations:** The existing installation scripts were tightly coupled to individual Ubuntu releases. Attempts to reuse them on other versions — such as 22.04.4, 22.04.5, 23.04, and 24.04 — resulted in package conflicts, deprecated dependencies, and execution failures.
- **Lack of Automation for Version Detection:** There was no unified mechanism to dynamically handle version-specific installation flows. Manual selection or script modification was required depending on the system version, making the process inefficient and error-prone.
- **Need for a Containerized Version of eSim:** Installation inconsistencies across different Ubuntu versions, due to environment-specific factors and package availability, highlighted the need for a portable solution. A containerized approach was required to ensure consistent setup and execution regardless of host system configuration.
- **Requirement for Updated Installers for eSim 2.5:** While eSim 2.4 was available, the release of eSim 2.5 introduced several changes that necessitated updated packaging workflows. Dedicated installers for both Ubuntu and Windows were required to streamline deployment for end users.

These challenges restricted the ease of installation, maintenance, and accessibility of eSim. Addressing them required a more generalized, version-agnostic, and user-friendly approach to deployment.

2.1.2 Approach to Address These Challenges

To overcome the issues outlined, the following solutions were developed and implemented:

- **Creation of a Centralized Controller Script:** A unified main script was developed to detect the host Ubuntu version and automatically execute the appropriate version-specific installer. This automation removed the need for manual intervention and improved overall usability.
- **Development of a Docker-Based Deployment:** A Docker image was created to encapsulate the eSim environment, providing a consistent and portable installation independent of the host system's configuration. This image was verified across multiple Ubuntu versions, including 25.04.
- **Packaging of Installers for eSim 2.5:** Updated and tested installers were created for eSim 2.5. These included a packaged Ubuntu installer as well as a Windows installer, significantly reducing the setup complexity for users and ensuring compatibility with the latest software version.

These improvements collectively enhanced the reliability, portability, and accessibility of eSim, making it easier to install and use across a wide range of system environments.

Chapter 3

Version Handling Installation Scripts for Ubuntu

3.1 Overview

To ensure smooth and consistent installation of eSim across different Ubuntu versions, a version-aware script was implemented. This script is responsible for detecting the system version and automatically invoking the appropriate version-specific installation script. This approach eliminates manual steps, reduces compatibility issues, and improves overall reliability. [3]

3.2 Version Detection Mechanism

The script uses a combination of `lsb_release` and `/etc/os-release` to determine both the base and full version of the operating system:

```
VERSION_ID=$(grep "^VERSION_ID" /etc/os-release | cut -d '"' -f 2)
FULL_VERSION=$(lsb_release -d | grep -oP '\d+\.\d+\.\d+')
```

Based on the detected version, the script selects the appropriate installer. The logic is structured using a case statement:

```
case $VERSION_ID in
  "22.04")
    if [[ "$FULL_VERSION" == "22.04.4" ]]; then
        SCRIPT="install-eSim-22.04.sh"
    else
        SCRIPT="install-eSim-23.04.sh"
    fi
    ;;
  "23.04")
    SCRIPT="install-eSim-23.04.sh"
    ;;
  "24.04")
```

```
    SCRIPT="install-eSim-24.04.sh"
    ;;
  *)
    echo "Unsupported Ubuntu version: $VERSION_ID ($FULL_VERSION)"
    exit 1
    ;;
esac
```

3.3 Supported Versions

This version-handling script has been tested and confirmed to work on the following Ubuntu releases:

- Ubuntu 22.04.4 LTS
- Ubuntu 22.04.5 LTS
- Ubuntu 23.04
- Ubuntu 24.04

3.4 Error Handling

The script performs input validation and provides user-friendly error messages in the following cases:

- Invalid or missing arguments (e.g., no `--install` or `--uninstall`)
- Unsupported Ubuntu versions
- Missing or misnamed install scripts

All critical failures are handled gracefully, with informative messages printed to the terminal.

3.5 Advantages

This structure offers several key benefits:

- Avoids hardcoding installation steps into a single script
- Makes the installer modular and easier to maintain
- Reduces risk of user error during manual setup
- Enables better compatibility across a range of system configurations

3.6 Extension to NGHDL

A similar version-handling script was also implemented for `install-nghdl.sh`, following the same logic and structure to ensure compatibility with different Ubuntu versions. [4]

3.7 Conclusion

By using version-handling scripts to manage installer execution, eSim can now be deployed more reliably across supported Ubuntu versions. This modular and adaptive approach improves long-term maintainability and reduces manual effort during installation.

Chapter 4

Docker Image for eSim

4.1 Overview

To ensure consistent functionality and ease of deployment across different Ubuntu systems, a Docker image for eSim 2.5 was created. The containerized setup addresses compatibility issues, simplifies the installation process, and eliminates the need for users to handle dependencies manually. The final image was designed to support a plug-and-play experience for eSim users on Linux systems with Docker.

4.2 Motivation

Installing eSim on various Ubuntu versions often resulted in dependency conflicts, environment misconfigurations, or package mismatches. Additionally, newer Ubuntu releases (such as 25.04) introduced changes that were incompatible with the legacy scripts. To mitigate these problems, a Docker-based solution was adopted to:

- Provide a consistent, pre-configured runtime environment
- Eliminate the need for version-specific script adjustments
- Offer an easy and reproducible setup method

4.3 Dockerfile Structure

The Docker image was built using `ubuntu:22.04` as the base. The following additional tools and steps were included:

- All required packages and GUI libraries
- `PyQt5` installed via `pip3` for NGHDL support
- A dedicated non-root user (`esimuser`) for running the application
- PDF viewer (`evince`) to open the bundled user manual from within the container

- A prompt-free, automated installation script to support non-interactive Docker builds

Dockerfile:

Listing 4.1: Dockerfile for eSim 2.5

```
FROM ubuntu:22.04

# Step 1: Install system packages
RUN apt-get update && apt-get install -y \
    lsb-release curl wget sudo unzip \
    python3 python3-venv python3-pip \
    libx11-dev libxcb1 libx11-xcb-dev libxcb-util1 \
    libxcb-xinerama0 libxext6 libxrandr2 \
    libqt5gui5 libqt5core5a libqt5widgets5 \
    libglib2.0-0 libwayland-client0 libwayland-egl1 \
    libxrender1 dbus-x11 mesa-utils \
    software-properties-common xdg-utils \
    evince && apt-get clean && rm -rf /var/lib/apt/lists/*

# Step 2: Install PyQt5 for NGHDL
RUN pip3 install --no-cache-dir PyQt5

# Step 3: Create a non-root user
RUN useradd -m -s /bin/bash esimuser && \
    echo "esimuser:password" | chpasswd && \
    usermod -aG sudo esimuser && \
    echo "esimuser␣ALL=(ALL)␣NOPASSWD:␣ALL" >> /etc/sudoers

# Step 4: Switch to non-root user
USER esimuser
WORKDIR /home/esimuser

# Step 5: Copy eSim source files
COPY --chown=esimuser:esimuser ./esim /home/esimuser/esim/

# Step 6: Make script executable
RUN chmod +x /home/esimuser/esim/install-eSim.sh

# Step 7: Install eSim non-interactively
RUN /home/esimuser/esim/install-eSim.sh --install || echo "eSim␣
    install␣failed.␣Check␣logs␣during␣runtime."

# Step 8: GUI support
ENV DISPLAY=:0

# Step 9: Default shell
CMD ["bash"]
```

4.4 Build and Image Creation

The Docker image was built using the following command:

```
docker build -t esim-2.5 .
```

This generated a pre-installed, self-contained Docker image named `esim-2.5` that is ready to run out-of-the-box.

Figure 4.1: Building eSim Docker Image

4.5 Modifications for Container Usability

To ensure the image worked seamlessly in container environments, several application-level changes were made:

- The **installation script was modified** to run without user prompts, enabling unattended installation during image build time.
- The **default "Dev Docs" button** in the GUI was commented out, since containers typically lack a browser. Instead, a message was printed to the console with the documentation URL so users can open it externally.
- A **PDF viewer (Evince)** was included so that users can open the bundled eSim user manual directly from within the Docker container.
- The image was designed with a focus on simplicity and "plug-and-play" behavior — users can launch it without needing to perform any installation steps inside the container.

Figure 4.2: eSim through Docker

4.6 Usage Instructions and GUI Setup

To use the eSim Docker container:

1. **Ensure Docker is installed and running** on the host system.
2. **Load the Docker image** (if not built locally):

```
docker load -i /path/to/esim-2.5.tar
```

3. **Allow Docker access to the host display** for GUI support:

```
xhost +local:docker
```

4. **Run the container using:**

```
docker run -it \
  --env DISPLAY=$DISPLAY \
  --volume /tmp/.X11-unix:/tmp/.X11-unix \
  esim-2.5
```

5. **Inside the container, launch eSim:**

```
esim
```

Note: These steps mount the X11 Unix socket and set the required `DISPLAY` variable, allowing the eSim GUI to run inside the container and display on the host system.

Storage: As of now, no persistent storage or project-sharing folder is configured. This can be implemented in the future using Docker volume mounts to allow users to save and retrieve projects outside the container.

4.7 Testing and Compatibility

The containerized version of eSim was tested on select Ubuntu host systems, and it successfully ran without requiring modifications to the host environment. While full testing was not conducted across all Ubuntu releases, the Docker image is expected to work reliably on the following distributions based on their compatibility with Docker and the image's base configuration:

- Ubuntu 22.04.4 LTS
- Ubuntu 22.04.5 LTS
- Ubuntu 23.04
- Ubuntu 24.04
- Ubuntu 25.04

Users on these systems should be able to run the image without issues, provided that Docker and GUI support (via X11 forwarding) are correctly configured.

4.8 Upgrade to eSim 2.5

The Docker image was later updated to include eSim 2.5. The upgrade process involved updating the copied source files inside the container and modifying the installation scripts to reflect the new build process and dependencies. This ensures that the image remains current with the latest official eSim release.

4.9 Benefits of Containerization

- Ensures consistent and reproducible installation across systems
- Eliminates host-level package conflicts and missing dependencies
- Simplifies deployment for academic use, testing, and demonstrations

- Requires no manual configuration or version-specific setup steps
- Supports GUI-based usage with minimal host configuration

4.10 Conclusion

The Docker-based deployment of eSim 2.5 provides a streamlined and reliable method to run the application across diverse system environments. With prompt-free installation, GUI compatibility, and built-in PDF viewer support, the image is designed for usability. Future additions such as persistent storage, volume mounting, and remote documentation access could further enhance its functionality and adoption.

Chapter 5

Installer Packaging for eSim 2.5

5.1 Overview

As part of the eSim 2.5 release, dedicated platform-specific installers were created for both Ubuntu and Windows. These installers were designed to simplify the user experience by automating the setup process, ensuring correct dependency management, and reducing installation errors. The final installers generated through this effort were included in the official eSim 2.5 release. [5]

5.2 Ubuntu Installer

5.2.1 Packaging Process

The Ubuntu installer was created by following the packaging instructions available in the official repository. It includes structured installation logic, version-aware detection, and dependency handling. The package provides both installation and uninstallation modes and supports multiple Ubuntu versions via internal scripts. [7]

5.2.2 Challenges and Observations

While the base packaging instructions provided in the repository served as a starting point, they required updates to align with the structural changes introduced in eSim 2.5. In particular:

- The official instructions only mention copying the main installer script (`install-eSim.sh`) into the `eSim-<version>` folder. However, starting from version 2.5, this script internally depends on a new directory named `install-eSim-scripts/`, which contains the version-specific installers.
- Omitting the `install-eSim-scripts/` folder leads to script failures during execution, as version handling is broken without these files.
- This discrepancy was identified during the packaging process. The packaging steps need to be updated accordingly to include both:

  - `install-eSim.sh` (the main entry point)
  - `install-eSim-scripts/` (required for version-specific logic)

- Additional minor adjustments were required in handling folder structure, permissions, and testing the install/uninstall flows.

These updates ensure that the Ubuntu installer functions as expected for eSim 2.5 and above, and that it is aligned with the structural changes made to the installation framework.

5.2.3 Result

The Ubuntu installer for eSim 2.5 was successfully generated and contributed to the official release repository. It allows for simplified installation with minimal manual intervention across supported versions. [6]

5.3 Windows Installer

5.3.1 Packaging Process

The Windows installer was created using a two-step process:

1. **Application Bundling:** eSim was compiled into a standalone executable using `PyInstaller`.
2. **Installer Creation:** A user-friendly installer was built using `NSIS (Nullsoft Scriptable Install System)`.

This approach allowed the application and all necessary dependencies to be packaged into a single executable file, followed by the creation of a structured, guided installer for end users. [8]

5.3.2 Modifications to PyInstaller Setup

The instructions initially provided for generating the PyInstaller executable did not include all necessary files to support newer features in eSim 2.5. Specifically, support for the latest schematic converter required additional data paths to be bundled manually in the `.spec` file. The `datas` section was updated as follows:

Listing 5.1: Modified datas entry in eSim.spec

```
datas=[
  ('src/converter/schematic_converters/lib/PythonLib/*.py', '
     converter/schematic_converters/lib/PythonLib'),
  ('src/converter/LTSpiceToKiCadConverter/src/Windows/*','/
     converter/LTSpiceToKiCadConverter/src/Windows')
],
```

Without this modification, the PyInstaller build would complete successfully but result in missing runtime functionality for the converter feature.

Additionally, the `TerminalUi.ui` file, which was not mentioned in the official packaging instructions, had to be manually included when compressing the eSim directory. Omitting it caused runtime UI issues.

5.3.3 Fix for Welcome Page in Windows Executable

The `Welcome.py` script, responsible for rendering the welcome page in the eSim GUI, originally used a relative path assumption to locate the `welcome.html` file. This approach worked in source environments but failed when the application was packaged into a Windows executable using PyInstaller. [9]

To resolve this, the file path logic was updated to dynamically locate the directory of the running executable. The modified block used the `sys.executable` path (for bundled apps) or `__file__` path (during development) to construct the full absolute path to `welcome.html`.

The updated code is shown below:

Listing 5.2: Modified path handling in Welcome.py

```
# Original logic (simplified):
init_path = '../../'
if os.name == 'nt':
    init_path = ''
self.browser.setSource(QtCore.QUrl(init_path + "library/browser/
    welcome.html"))

# Modified logic:
base_path = os.path.dirname(sys.executable) if getattr(sys, 'frozen
    ', False) else os.path.dirname(__file__)
html_path = os.path.abspath(os.path.join(base_path, "library/
    browser/welcome.html"))
self.browser.setSource(QtCore.QUrl.fromLocalFile(html_path))
```

This change ensures that the welcome page displays correctly when eSim is launched from the packaged Windows executable, regardless of the user's working directory or environment.

5.3.4 NSIS Script Fixes and Enhancements

The NSIS script provided [10] for installer generation required multiple adjustments to support updated paths and file handling behaviors:

- The downloaded KiCad installer had to be renamed manually to `kicad-6.0.11-i686.exe` to match the script's expectations. This requirement was undocumented and initially caused installation failures.
- The installer logic was updated to correctly extract and place KiCad components. The `-InstallKiCad` section in the NSIS script was modified extensively to address runtime errors and improve behavior. Key updates included:

- Commenting out the failing ZIP extraction and using pre-extracted libraries instead
- Creating the KiCad configuration directory
- Copying essential template and configuration files to required paths
- Cleaning up redundant or non-functional lines

The modified NSIS section is shown below, with inline comments to highlight the changes made:

Listing 5.3: Modified NSIS Section -InstallKiCad

```
Section -InstallKiCad

  SetOutPath "$EXEDIR"
  File "kicad-6.0.11-i686.exe"          ; Renamed manually to
     match this name

  SetOutPath "$INSTDIR"
  SetDetailsPrint both
  DetailPrint "Installing: KiCad......"
  SetDetailsPrint listonly
  ExecWait '"$EXEDIR\kicad-6.0.11-i686.exe" /S /D=$INSTDIR\
     KiCad'
  SetDetailsPrint both

  Goto endActiveSync
  endActiveSync:

  ; Clean up unnecessary files
  Delete "$EXEDIR\kicad-6.0.11-i686.exe"
  Delete "$PROFILE\..\Public\Desktop\KiCad*.lnk"  ; Added to
     remove unwanted desktop shortcuts

  ; Set environment variable for KiCad binary path
  EnVar::SetHKLM
  EnVar::AddValue "Path" "$INSTDIR\KiCad\bin"
  Pop $0
  DetailPrint "EnVar::AddValue returned=|$0|"

  ; The following ZIP extraction was failing, so it was
     commented out
  ; ZipDLL::extractall "$INSTDIR\eSim\library\kicadLibrary.zip"
      "$INSTDIR\eSim\library\"

  ; These lines were commented due to inconsistent behavior
  ; CopyFiles "$INSTDIR\eSim\library\kicadLibrary\eSim-symbols
     \*" "$INSTDIR\KiCad\share\kicad\symbols\"

  ; Removed previous KiCad config directory if present (
     optional)
  ; RMDir /r "$PROFILE\AppData\Roaming\kicad\6.0\"

  ; Added to create the config directory manually
  CreateDirectory "$PROFILE\AppData\Roaming\kicad\6.0\"
```

```
37
38  ␣␣;␣Copy␣KiCad␣template␣files␣to␣user␣and␣KiCad␣directories
39  ␣␣CopyFiles␣"$INSTDIR\eSim\library\kicadLibrary\template\*"␣"
        $PROFILE\AppData\Roaming\kicad\6.0\"
40  ␣␣Delete␣"$INSTDIR\KiCad\share\kicad\template\sym-lib-table"
41  ␣␣CopyFiles␣"$INSTDIR\eSim\library\kicadLibrary\template\*"␣"
        $INSTDIR\KiCad\share\kicad\template\"
42
43  ␣␣;␣Remove␣extracted␣KiCad␣Library␣(used␣pre-extracted␣version)
44  ␣␣RMDir␣/r␣"$INSTDIR\eSim\library\kicadLibrary"
45
46  SectionEnd
```

- Few lines were commented out or replaced with manual alternatives due to inconsistencies in NSIS execution on some systems.

5.3.5 Additional Observations

- Placement instructions for `sky130_fd_pr.7z` within the installer folder were not clearly documented and required manual trial-and-error.
- The installer had to be rebuilt multiple times and tested across fresh Windows environments to verify its correctness.
- All changes were made with minimal deviation from the existing structure to ensure backward compatibility.

5.3.6 Outcome

The final Windows installer includes the full eSim 2.5 application, bundled dependencies, integrated KiCad, and relevant configuration files. It provides a clean graphical setup experience, sets environment variables, and ensures proper application startup post-installation.

This installer, along with the Ubuntu version, was contributed to the official eSim 2.5 release. [**?**]

5.4 Testing and Validation

Both installers underwent repeated test cycles on their respective platforms. The testing covered:

- Clean installations on fresh systems
- Proper execution of the application post-installation
- Verification of environment setup and GUI behavior
- Functional uninstallation and cleanup

5.5 Benefits of Platform-Specific Installers

- Reduces the barrier to entry for non-technical users
- Ensures consistent installation across different systems
- Minimizes dependency-related issues
- Supports distribution through official release channels

5.6 Conclusion

The successful packaging of eSim 2.5 into dedicated Windows and Ubuntu installers represents a major improvement in its usability and accessibility. The effort involved fixing outdated configurations, automating environment setup, and performing extensive testing to deliver reliable, easy-to-use installers. These installers now serve as the default installation method for eSim users across platforms.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The work carried out focused on significantly enhancing the installation, compatibility, and usability of eSim across a range of platforms and system configurations. Key contributions include:

- Development of version-handling installation scripts to support multiple Ubuntu versions without requiring manual modification.
- Creation of a Docker image to provide a consistent, portable runtime environment that works across host systems — including newer releases like Ubuntu 25.04.
- Packaging of eSim 2.5 installers for both Ubuntu and Windows platforms, enabling simplified installation with minimal effort from the user.
- Identification and resolution of system-level, packaging, and path-related issues during the installer creation process.
- Extensive testing of all installers and deployment methods to ensure reliability and proper functionality.

These efforts collectively improved the distribution and adoption potential of eSim, particularly for academic, research, and individual users working in diverse environments.

6.2 Future Scope

While the current work has addressed major usability and deployment challenges across platforms, several areas remain for future improvement. These enhancements could further improve the robustness, portability, and accessibility of eSim:

- **Persistent Storage and Volume Mounting in Docker:** The current Docker setup does not support persistent project storage. Implementing shared volume mounts between the host and container would allow users to save and retrieve their eSim files across sessions.

- **Adaptations for Headless and Browser-Less Environments:** As the container environment lacks a graphical web browser, the DevDocs button in the GUI was disabled. Future solutions could include terminal-based documentation access or serving docs over a lightweight local webserver accessible from the host browser.

- **Cross-Platform GUI-Based Installer:** Developing a unified, graphical installer (e.g., using Qt or Electron) for both Windows and Linux systems would further simplify setup, especially for non-technical users unfamiliar with terminal commands.

- **Continuous Integration and Automated Testing Pipelines:** Incorporating CI tools such as GitHub Actions to automate building and testing of Docker images, Ubuntu installers, and Windows packages would improve release consistency, reduce manual effort, and catch regressions early.

- **Improved and Updated Packaging Documentation:** Several undocumented requirements were discovered during the packaging process for eSim 2.5, including critical paths, renamed files, and additional folders. Updating the official documentation to reflect these changes would support future contributors and reduce onboarding time.

- **Multi-Architecture and OS Support:** Future versions could extend support to ARM-based platforms (e.g., Raspberry Pi), macOS (via Docker or virtualization), and Linux distributions beyond Ubuntu, thereby broadening accessibility.

- **Automatic Update Mechanism:** An optional version checker or auto-updater could be introduced to notify users of new releases and guide them through upgrades.

- **Windows Path Handling and Portability Enhancements:** Improvements can be made to streamline platform-specific path handling logic (e.g., detecting relative paths to resources), especially in PyInstaller-built executables, to reduce hardcoding and increase portability.

Implementing these enhancements would elevate eSim's usability, maintainability, and platform reach—helping it mature as a reliable open-source EDA tool for both academic and professional use.

6.3 Final Remarks

The efforts undertaken during this phase laid a strong foundation for the sustainable packaging and deployment of eSim. By resolving compatibility barriers and introducing modular, tested, and maintainable installer solutions, the software is now better positioned for widespread usage across both Linux and Windows platforms.

Bibliography

[1] FOSSEE Official Website, 2020. Available at:
https://fossee.in/about

[2] eSim Official Website, 2020. Available at:
https://esim.fossee.in/

[3] eSim Ubuntu installer script, 2025. Available at:
https://github.com/FOSSEE/eSim/.../Ubuntu/install-eSim.sh

[4] nghdl Ubuntu installer script, 2025. Available at:
https://github.com/FOSSEE/nghdl/.../Ubuntu/install-nghdl.sh

[5] eSim-2.5 release, 2025. Available at:
https://github.com/FOSSEE/eSim/releases/tag/v2.5

[6] eSim-2.5 Downloads, 2025. Available at:
https://esim.fossee.in/downloads

[7] Ubuntu Installer Packaging documentation, 2022. Available at:
https://github.com/FOSSEE/eSim/.../Ubuntu/README.md

[8] Windows Installer Packaging documentation, 2023. Available at:
https://github.com/FOSSEE/eSim/.../Windows/README.md

[9] Welcome.py, 2025. Available at:
https://github.com/FOSSEE/eSim/.../src/browser/Welcome.py

[10] esim-setup-script.nsi, 2023. Available at:
https://github.com/FOSSEE/eSim/.../Windows/esim-setup-script.nsi