



eSim Winter Internship Report

On

eSim for Ubuntu 23.04

Submitted by

Jayanth Tatineni

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

Under the guidance of

Sumanto Kar

IIT Bombay

February 23, 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

1	Introduction	4
1.1	FOSSEE: Promoting Open-Source Software for Education	4
1.2	eSim: An Open-Source EDA Tool	4
1.2.1	Key Features of eSim	5
1.2.2	Components Integrated with eSim	5
2	Problem Statement	6
2.1	Challenges in eSim Installation	6
2.1.1	Issues Encountered During Installation	6
2.1.2	Approach to Address These Challenges	7
3	Modifying install-eSim.sh	8
3.1	Handling NgVeri dependencies	8
3.1.1	Overview	8
3.1.2	Problem Faced	8
3.1.3	Virtual Environment Implementation	8
3.1.4	Dependency Installation within the Virtual Environment	9
3.1.5	Conclusion	10
3.2	Addressing Package Installation Issues Due to Outdated <code>sources.list</code>	10
3.2.1	Overview	10
3.2.2	Problem Faced	10
3.2.3	Implementation to Update <code>sources.list</code>	11
3.2.4	How the Solution Works	12
3.2.5	Conclusion	12
3.3	Additional Modifications	12
3.3.1	Updates to <code>createDesktopStartupScript</code>	12
3.3.2	Enhancements to the Uninstallation Process	13
3.3.3	Conclusion	13
4	Modifying install-nghdl.sh	14
4.1	Updating LLVM Version	14
4.2	Updating GHDL Installation	15
4.2.1	Previous Method	15
4.2.2	Updated Method	15
4.2.3	Advantages of the New Method	16
4.3	Fixing Verilator Compilation Error	16
4.4	Conclusion	17

5	Verification and Successful Installation	18
5.1	Overview	18
5.2	Verifying eSim Installation	18
5.3	Successful Installation Confirmation	19
5.4	Successful Uninstallation Confirmation	20
5.5	Conclusion	20
6	Conclusion and Future Scope	21
6.1	Conclusion	21
6.2	Future Scope	21
	Bibliography	23

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 update the eSim installer script to install eSim 2.4 on Ubuntu 23.04 (Lunar Lobster) without errors.

2.1 Challenges in eSim Installation

2.1.1 Issues Encountered During Installation

While attempting to install eSim 2.4 on Ubuntu 23.04, several challenges arose that prevented successful installation. These issues stemmed from dependency mismatches, outdated packages, and software incompatibilities. The key problems encountered were:

- **Improper Installation of NgVeri Dependencies:** The installation script failed to properly install certain dependencies required for NgVeri, a crucial component for simulation. This was primarily due to missing or improperly configured package sources.
- **Incompatibility of LLVM, GHDL, and Verilator:** The default installation script attempted to install older versions of LLVM, GHDL, and Verilator, leading to multiple compatibility issues. Specifically, the script was designed for an earlier version of Ubuntu, where LLVM 9 was available. However, LLVM 9 was no longer supported in Ubuntu 23.04, causing package resolution errors. Similarly, GHDL failed due to an unhandled LLVM version, and Verilator required additional modifications to build successfully.
- **Outdated Package Sources in sources.list:** The script relied on outdated package sources, which resulted in missing dependencies. The default `sources.list` file did not include all necessary repositories, leading to errors when attempting to fetch certain packages. This problem was compounded by the fact that some required dependencies had been moved to different repositories or were no longer maintained.

These challenges caused repeated failures in the installation process, requiring manual intervention at multiple stages. Resolving them required modifying the script to ensure compatibility with the latest Ubuntu release.

2.1.2 Approach to Address These Challenges

To overcome these installation issues, several strategic modifications were made to the script. The revised approach included the following key improvements:

- **Implementing a Virtual Environment for NgVeri Dependencies:** To avoid conflicts with system-wide dependencies, a virtual environment was set up for NgVeri. This ensured that all necessary dependencies were installed in an isolated environment without affecting the system's default package versions. The virtual environment allowed for better dependency management and minimized compatibility issues with other installed software.
- **Updating the Script to Install Newer Versions of LLVM and Verilator, and Manually Installing GHDL:** The script was modified to install a newer version of LLVM (version 15) instead of the outdated LLVM 9, which was no longer available in Ubuntu 23.04. Similarly, the Verilator installation was updated to fix build errors by ensuring proper inclusion of necessary C++ headers. GHDL, which previously failed due to an unhandled LLVM version, was manually downloaded and installed using `wget` to fetch the appropriate release from GitHub.
- **Checking and Updating `sources.list` if Needed:** Since outdated package sources were a major cause of missing dependencies, the script was modified to verify and update `sources.list` before proceeding with the installation. This step ensured that the correct repositories were available, preventing failures due to missing packages. Additionally, package lists were refreshed using `sudo apt update` before installation to avoid errors caused by stale metadata.

These modifications significantly improved the reliability of the installation process, reducing manual intervention and ensuring compatibility with the latest system updates.

Chapter 3

Modifying install-eSim.sh

This chapter describes the modifications made to the ‘install-eSim.sh’ script to ensure compatibility with the latest system configurations and resolve errors encountered during installation. [6]

3.1 Handling NgVeri dependencies

3.1.1 Overview

NgVeri requires several dependencies for proper functionality. Managing these dependencies efficiently ensures that the software runs smoothly across different systems without conflicts. To achieve this, a virtual environment was implemented to isolate package installations from the system-wide Python environment.

3.1.2 Problem Faced

Initially, dependencies were installed globally using system package managers such as `apt`. This approach had several drawbacks:

- Conflicts between system-wide and project-specific package versions.
- Difficulty in maintaining consistent environments across different machines.
- Potential issues when upgrading or uninstalling packages.
- Risk of affecting other Python projects on the system.

3.1.3 Virtual Environment Implementation

To mitigate these issues, a Python virtual environment was implemented in the script. This prevents conflicts between system-wide and project-specific packages and allows controlled dependency management. The virtual environment setup follows these steps:

Creating the Virtual Environment

Before creating a virtual environment, the script checks if the directory `venv` already exists:

```
1 if [ ! -d "venv" ]; then
2     echo "Creating virtual environment..."
3     python3.11 -m venv venv
4 else
5     echo "Virtual environment already exists."
6 fi
```

If the directory does not exist, the command `python3.11 -m venv venv` creates a new virtual environment named `venv`. If the directory already exists, the script skips the creation step to avoid redundant setups.

Activating the Virtual Environment

Once the virtual environment is set up, it is activated using:

```
1 source ./venv/bin/activate
```

This ensures that all subsequent Python-related commands use the isolated environment rather than system-wide Python installations.

Ensuring Proper Deactivation

To ensure that the virtual environment is deactivated when the script exits, a trap is set:

```
1 trap 'if [[ -d "venv" ]]; then deactivate; fi' EXIT
```

This guarantees that when the script terminates, the virtual environment is exited cleanly, preventing potential conflicts in future commands executed in the terminal.

3.1.4 Dependency Installation within the Virtual Environment

Once the virtual environment is activated, dependencies are installed using `pip`. The package installation is handled as follows:

- **Using `pip`:** Packages such as Watchdog, PyQt5, and Matplotlib are installed inside the virtual environment:

```
1 pip install watchdog PyQt5 matplotlib
```

- **Using `pip3`:** Some dependencies, such as Hdlparse, Makerchip, and SandPiper SaaS, are installed using `pip3`:

```
1 pip3 install --upgrade https://github.com/hdl/
   pyhdlparser/tarball/master
2 pip3 install hdlparse
3 pip3 install makerchip-app sandpiper-saas
```

The use of pip3 ensures compatibility with Python 3 and avoids conflicts with system-wide installations.

3.1.5 Conclusion

By implementing a virtual environment for dependency management, the script ensures:

- Dependencies are installed in an isolated manner, avoiding interference with system-wide packages.
- The environment remains consistent across different machines, improving reproducibility.
- Proper cleanup via deactivation upon script exit, preventing persistent environment modifications.

3.2 Addressing Package Installation Issues Due to Outdated sources.list

3.2.1 Overview

Ubuntu releases have a defined support lifecycle, after which they reach End of Life (EOL). For instance, Ubuntu 23.04 (Lunar Lobster) reached EOL on January 25, 2024 [8]. Post-EOL, official package repositories are moved to an archive server, necessitating updates to the system's `sources.list` to maintain package management functionality.

3.2.2 Problem Faced

After Ubuntu 23.04 reached EOL, its standard repositories were relocated to `old-releases.ubuntu.com`. Systems continuing to use the default `sources.list` encountered errors during package installations and updates due to inaccessible repositories.

```
E: The repository 'http://ft.archive.ubuntu.com/ubuntu lunar-backports Release' no longer has a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
E: The repository 'http://ft.archive.ubuntu.com/ubuntu lunar-security Release' no longer has a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
Installing virtualenv.....
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-distlib python3-filelock python3-platformdirs python3-wheel-whl
The following NEW packages will be installed:
  python3-distlib python3-filelock python3-platformdirs python3-virtualenv python3-wheel-whl
0 upgraded, 5 newly installed, 0 to remove and 127 not upgraded.
Need to get 403 kB of archives.
After this operation, 1,742 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Err:1 http://ft.archive.ubuntu.com/ubuntu lunar/universe amd64 python3-distlib all 0.3.6-1
404 Not Found [IP: 185.125.190.82 80]
Err:2 http://ft.archive.ubuntu.com/ubuntu lunar/universe amd64 python3-filelock all 3.9.0-1
404 Not Found [IP: 185.125.190.82 80]
Err:3 http://ft.archive.ubuntu.com/ubuntu lunar/universe amd64 python3-platformdirs all 3.0.0-1
404 Not Found [IP: 185.125.190.82 80]
Err:4 http://ft.archive.ubuntu.com/ubuntu lunar/universe amd64 python3-wheel-whl all 0.38.4-1
404 Not Found [IP: 185.125.190.82 80]
Err:5 http://ft.archive.ubuntu.com/ubuntu lunar/universe amd64 python3-virtualenv all 20.19.0+ds-1
404 Not Found [IP: 185.125.190.82 80]
Failed to fetch http://ft.archive.ubuntu.com/ubuntu/pool/universe/d/distlib/python3-distlib_0.3.6-1_all.deb 404 Not Found [IP: 185.125.190.82 80]
Failed to fetch http://ft.archive.ubuntu.com/ubuntu/pool/universe/p/python-filelock/python3-filelock_3.9.0-1_all.deb 404 Not Found [IP: 185.125.190.82 80]
Failed to fetch http://ft.archive.ubuntu.com/ubuntu/pool/universe/p/platformdirs/python3-platformdirs_3.0.0-1_all.deb 404 Not Found [IP: 185.125.190.82 80]
Failed to fetch http://ft.archive.ubuntu.com/ubuntu/pool/universe/w/wheel/python3-wheel-whl_0.38.4-1_all.deb 404 Not Found [IP: 185.125.190.82 80]
Failed to fetch http://ft.archive.ubuntu.com/ubuntu/pool/universe/p/python-virtualenv/python3-virtualenv_20.19.0+ds-1_all.deb 404 Not Found [IP: 185.125.190.82 80]
Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?

Error! Kindly resolve above error(s) and try again.
Aborting Installation...
```

Figure 3.1: Error due to outdated sources.list

3.2.3 Implementation to Update sources.list

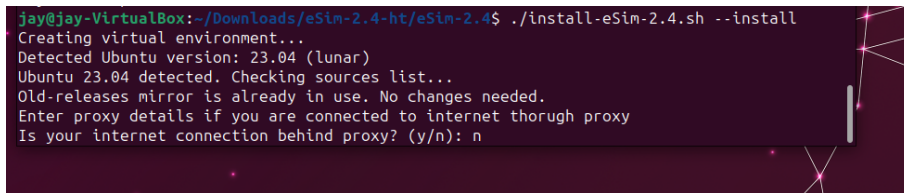
To resolve this, a script was implemented to detect the Ubuntu version and, if necessary, update the `sources.list` to point to the archived repositories:

```
1 # Update Sources list if necessary
2 # Get Ubuntu version
3 UBUNTU_VERSION=$(lsb_release -rs)
4 UBUNTU_CODENAME=$(lsb_release -cs)
5
6 echo "Detected Ubuntu version: $UBUNTU_VERSION (
7     $UBUNTU_CODENAME)"
8
9 # Check if running Ubuntu 23.04 (Lunar)
10 if [[ "$UBUNTU_CODENAME" == "lunar" ]]; then
11     echo "Ubuntu 23.04 detected. Checking sources list..."
12
13     # Check if the old-releases mirror is already set
14     if grep -q "old-releases.ubuntu.com" /etc/apt/sources.
15     list; then
16         echo "Old-releases mirror is already in use. No
17         changes needed."
18     else
19         echo "Switching to old-releases mirror..."
20
21         # Backup existing sources.list (only if not backed up
22         before)
23         if [ ! -f /etc/apt/sources.list.bak ]; then
24             sudo cp /etc/apt/sources.list /etc/apt/sources.
25             list.bak
26         fi
27
28         # Replace standard mirrors with old-releases
29         sudo sed -i 's|http://\(.*\).ubuntu.com/ubuntu|http
30         ://old-releases.ubuntu.com/ubuntu|g' /etc/apt/
31         sources.list
32
33         echo "Updated sources.list to use old-releases.
34         Running apt update..."
35
36         # Update package lists
37         sudo apt update -y && sudo apt upgrade -y
38     fi
39 else
40     echo "Not running Ubuntu 23.04, no changes needed."
41 fi
```

3.2.4 How the Solution Works

The script operates as follows:

1. **Version Detection:** Utilizes `lsb_release` to obtain the Ubuntu version and codename.
2. **EOL Check:** Verifies if the system is running Ubuntu 23.04 (codename "lunar").
3. **Mirror Verification:** Checks if the `sources.list` already points to `old-releases.ubuntu.com`.
4. **Backup Creation:** If not already backed up, creates a backup of the current `sources.list`.
5. **Source Update:** Replaces standard repository URLs with the archived ones.
6. **Package Update:** Runs `apt update` and `apt upgrade` to refresh package lists and upgrade installed packages.



```
jay@jay-VirtualBox:~/Downloads/eSim-2.4-ht/eSim-2.4$ ./install-eSim-2.4.sh --install
Creating virtual environment...
Detected Ubuntu version: 23.04 (lunar)
Ubuntu 23.04 detected. Checking sources list...
Old-releases mirror is already in use. No changes needed.
Enter proxy details if you are connected to internet through proxy
Is your internet connection behind proxy? (y/n): n
```

3.2.5 Conclusion

By implementing this script, systems running Ubuntu 23.04 can seamlessly transition to using the archived repositories post-EOL. This ensures continued access to package installations and updates without manual intervention.

3.3 Additional Modifications

3.3.1 Updates to createDesktopStartScript

the `createDesktopStartScript` function was modified to ensure seamless execution of eSim using the virtual environment. The previous implementation manually activated the virtual environment using the `source` command, whereas the new implementation directly references the Python interpreter inside the virtual environment.

```
1 # Generating new esim-start.sh
2     echo '#!/bin/bash' > esim-start.sh
3     echo "cd $eSim_Home/src/frontEnd || exit" >> esim-start.sh
4     echo "$eSim_Home/venv/bin/python Application.py" >> esim-start.sh
```

Previous implementation:

```
1 # Generating new esim-start.sh
2   echo '#!/bin/bash' > esim-start.sh
3   echo "cd $eSim_Home/src/frontEnd" >> esim-start.sh
4   echo "source $config_dir/env/bin/activate" >> esim-start.
5   sh
6   echo "python3 Application.py" >> esim-start.sh
```

The new approach simplifies execution by eliminating the need to explicitly activate the virtual environment, improving maintainability and reducing potential environment-related issues.

3.3.2 Enhancements to the Uninstallation Process

Additional cleanup steps were introduced in the uninstall section to properly remove dependencies and the virtual environment. This ensures a complete and clean uninstallation of eSim.

Newly added lines:

```
1 # Uninstalling pip3 dependencies
2 pip3 uninstall -y hdlparse makerchip-app sandpiper-saas
3
4 # Removing the virtual environment
5 if [[ -d "venv" ]]; then
6     deactivate
7     rm -rf venv
8 fi
```

These changes ensure that all installed dependencies are removed, and the virtual environment is properly deactivated and deleted, preventing residual configurations from interfering with future installations.

3.3.3 Conclusion

The modifications to `createDesktopStartupScript` and the uninstall process enhance the efficiency and maintainability of the eSim installation. The new approach ensures:

- A streamlined execution process by directly referencing the virtual environment's Python interpreter.
- A cleaner uninstallation by properly removing installed dependencies and deactivating the virtual environment.
- Reduced risk of conflicts with system-wide configurations.

Chapter 4

Modifying install-nghdl.sh

This chapter describes the modifications made to the ‘install-nghdl.sh’ script to ensure compatibility with the latest system configurations and resolve errors encountered during installation. [7]

4.1 Updating LLVM Version

The script originally attempted to install LLVM version 9 with the following command:

```
echo "Installing LLVM-9....."  
sudo apt install -y llvm-9 llvm-9-dev
```

However, this resulted in an error since `llvm-9` and `llvm-9-dev` were not available in the repositories for Ubuntu Lunar. The following error messages were encountered:

```
E: Unable to locate package llvm-9  
E: Unable to locate package llvm-9-dev
```

LLVM version 9 is outdated and has been removed from the default Ubuntu Lunar repositories. This led to a failure in the installation process, preventing other dependent tools from compiling correctly.

To resolve this issue, the LLVM version was updated to 15:

```
llvm_version="15"  
echo "Installing LLVM- $\{llvm\_version\}$ ....."  
sudo apt install -y llvm- $\{llvm\_version\}$  llvm- $\{llvm\_version\}$ -dev
```

LLVM 15 is a more recent version with better optimizations, improved support for modern architectures, and compatibility with the latest toolchains. The update not only resolved package availability issues but also ensured better performance and stability for tools depending on LLVM.

Furthermore, updating to LLVM 15 helped maintain compatibility with GHDL, which previously encountered issues due to an unrecognized LLVM version. By making this change, the script successfully installed the necessary dependencies without errors, allowing GHDL and other components to function correctly.

Additionally, to prevent future issues, it was verified that the correct LLVM version was installed by running:

```
llvm-config --version
```

This command confirmed that the expected version (LLVM 15) was correctly installed and available for use in subsequent compilation steps.

4.2 Updating GHDL Installation

4.2.1 Previous Method

Initially, the script extracted `ghdl-0.37` from a local archive and configured the build process manually. The installation involved the following steps:

- Extracting the `ghdl-0.37` archive.
- Running the `configure` script to set up the build.
- Compiling GHDL from source using `make`.
- Installing it using `sudo make install`.

However, this method resulted in an error due to LLVM version incompatibility:

```
Unhandled version llvm 15.0.7
```

4.2.2 Updated Method

To resolve this issue, the installation process was modified. Instead of extracting a pre-existing archive, the script now downloads the latest compatible version directly from the official GHDL GitHub releases. The following changes were made:

- A new directory is created to store the GHDL files.
- The script fetches the pre-built GHDL binary using `wget`:

```
wget https://github.com/ghdl/ghdl/releases/download/v4.1.0/$ghdl.tgz
```

- The archive is extracted without requiring compilation.
- Instead of using `make install`, the necessary files are manually copied:

```
sudo cp bin/* /usr/local/bin/  
sudo cp -r include/* /usr/local/include/  
sudo cp -r lib/* /usr/local/lib/
```


4.2.3 Advantages of the New Method

- **Faster Installation:** Skips the compilation step, reducing installation time.
- **Avoids LLVM Compatibility Issues:** Uses pre-built binaries optimized for the target platform.
- **Reliability:** Ensures that the latest stable version of GHDL is used by fetching it dynamically.

The updated method successfully installs GHDL while maintaining compatibility with LLVM 15.

4.3 Fixing Verilator Compilation Error

During the installation of Verilator, the following compilation error occurred:

```
./V3Const.cpp:35:1: note: 'std::unique_ptr' is defined in header '<memory>';  
did you forget to '#include <memory>'?
```

To resolve this issue, the `#include <memory>` directive was added to `V3Const.cpp` using the `sed` command:

```
sed -i '/#include <algorithm>/a #include <memory>'  
./verilator-4.210/src/V3Const.cpp
```

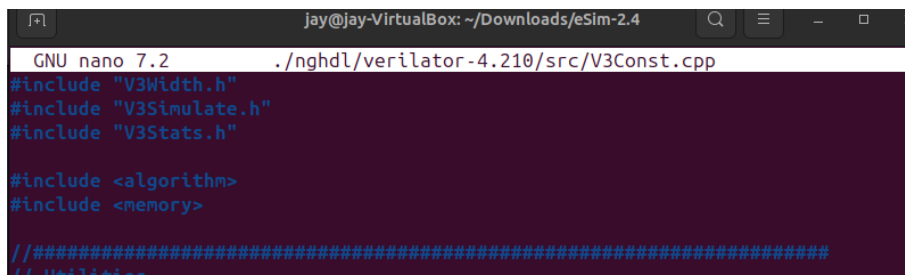
A screenshot of a terminal window showing the nano text editor. The window title is 'jay@jay-VirtualBox: ~/Downloads/eSim-2.4'. The editor is editing the file './nghdl/verilator-4.210/src/V3Const.cpp'. The visible code includes: '#include "V3Width.h"', '#include "V3Simulate.h"', '#include "V3Stats.h"', '#include <algorithm>', and '#include <memory>'. Below this, there are several lines of comments starting with '//' and 'Utilities'. The terminal background is dark purple.

Figure 4.1: modified file after execution of the command

This modification was applied within the `installVerilator` function to ensure the build process completes successfully. Instead of requiring users to manually edit files to fix compilation errors, the script automatically modifies the necessary source files before compilation.

By incorporating the `sed` command, the script inserts the required memory header into `V3Const.cpp` dynamically. This ensures that the build process does not fail due to missing headers, eliminating the need for manual intervention.

Additionally, the function automates the entire installation process, including extracting the archive, setting executable permissions, configuring, compiling, and installing Verilator. This streamlined approach not only ensures consistency across multiple installations but also reduces the risk of user errors when making manual changes.

```

function installVerilator
{
    echo "Installing $verilator....."
    tar -xJf $verilator.tar.xz
    sed -i '/#include <algorithm>/a #include <memory>'
    ./verilator-4.210/src/V3Const.cpp
    cd $verilator
    chmod +x configure
    ./configure
    make -j$(nproc)
    sudo make install
    echo "Verilator installed successfully"
    cd ../
}

```

These changes ensure that the installation process for ‘nghdl’ functions correctly on the target system.

4.4 Conclusion

In this chapter, we analyzed and modified the `install-nghdl.sh` script to address compatibility issues and improve its functionality. The primary changes included updating the LLVM version from 9 to 15 to resolve package availability issues, modifying the GHDL installation process to fetch the required version dynamically using `wget`, and implementing an automated fix for Verilator’s missing header dependency.

These modifications ensure that the script is fully compatible with modern Ubuntu distributions, eliminating the need for manual interventions during installation. By automating key aspects of the process, such as downloading and extracting necessary files, configuring builds, and applying necessary patches, the updated script streamlines the installation of NgHDL components.

Overall, the refined script enhances usability, reliability, and maintainability, making it more adaptable to future updates in dependencies.

Chapter 5

Verification and Successful Installation

5.1 Overview

After modifying the installation scripts to resolve compatibility issues with LLVM, GHDL, and Verilator, it is essential to verify whether eSim has been successfully installed. This chapter details the verification steps and confirms that the installation process now functions correctly.

5.2 Verifying eSim Installation

To ensure that eSim has been installed properly, the following verification steps were performed:

1. Checking Installed Components:

- Verified the installed LLVM version using:

```
llvm-config --version
```

- Confirmed that GHDL is installed and operational:

```
ghdl --version
```

- Checked the Verilator version to ensure compatibility:

```
verilator --version
```

2. Testing NgVeri and Verilog Simulations:

- Ran a sample Verilog simulation using NgVeri to confirm correct functionality.

- Ensured that the compiled Verilog designs produced the expected outputs.

3. Launching eSim:

- Started eSim from the terminal using:

```
esim
```

- Verified that the eSim GUI opened without errors.

5.3 Successful Installation Confirmation

After executing all verification steps, it was confirmed that eSim was successfully installed on Ubuntu 23.04. The modified script now ensures a smooth installation process by addressing dependency issues and using updated package versions.

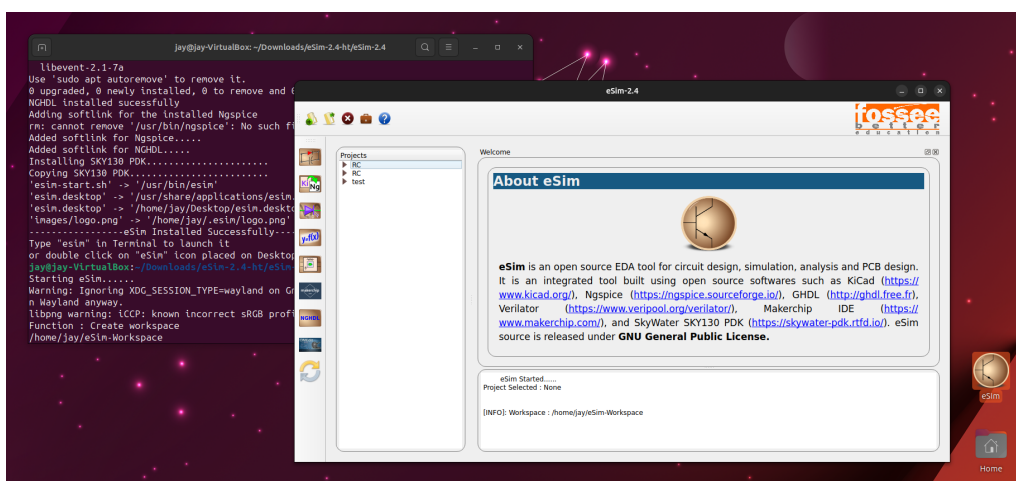
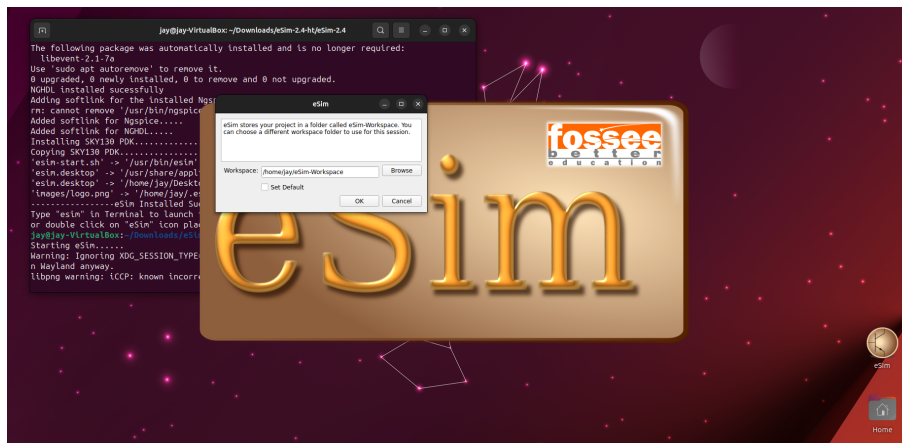
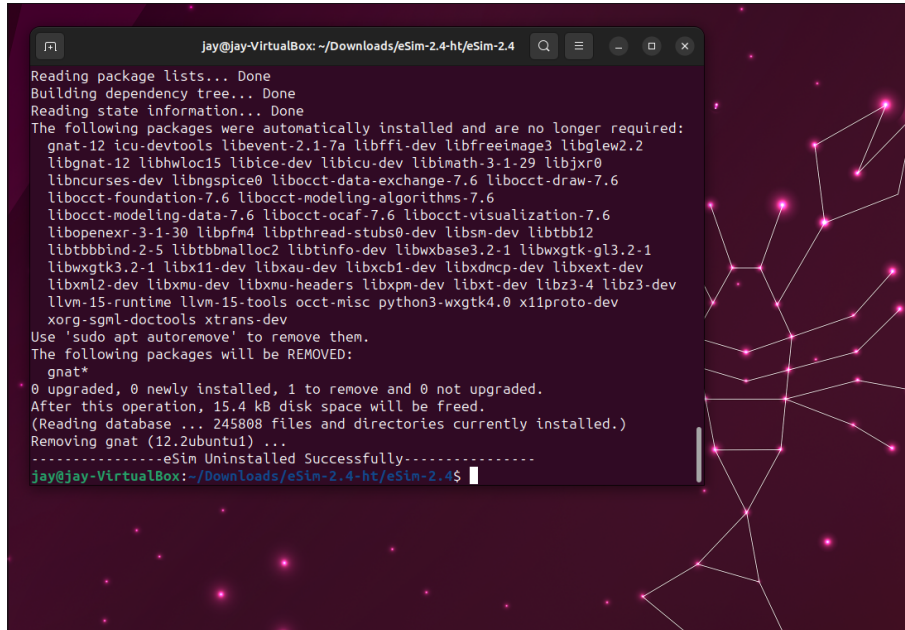


Figure 5.1: Successful launch of eSim after installation

5.4 Successful Uninstallation Confirmation

In addition to verifying the installation, it is important to ensure that eSim and its dependencies can be completely removed if required.

A terminal window titled 'jay@jay-VirtualBox: ~/Downloads/eSim-2.4-ht/eSim-2.4' displays the output of an 'apt autoremove' command. The output lists numerous packages that were automatically installed and are no longer required, such as 'gnat-12', 'icu-devtools', and 'libevent-2.1-7a'. It also shows that 'gnat*' is the only package to be removed. The terminal concludes with the message 'eSim Uninstalled Successfully' and a prompt for the user to enter a command.

```
jay@jay-VirtualBox: ~/Downloads/eSim-2.4-ht/eSim-2.4
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
gnat-12 icu-devtools libevent-2.1-7a libffi-dev libfreeimage3 libglew2.2
libgnat-12 libhwloc15 libice-dev libicu-dev libmath-3-1-29 libjxr0
libncurses-dev libngspice0 libocct-data-exchange-7.6 libocct-draw-7.6
libocct-foundation-7.6 libocct-modeling-algorithms-7.6
libocct-modeling-data-7.6 libocct-ocaf-7.6 libocct-visualization-7.6
libopenexr-3-1-30 libpfn4 libpthread-stubs0-dev libsm-dev libtbb12
libtbbbind-2-5 libtbbmalloc2 libtinfo-dev libwxbase3.2-1 libwxgtk-gtk3.2-1
libwxgtk3.2-1 libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libxext-dev
libxml2-dev libxmu-dev libxmu-headers libxpm-dev libxt-dev libz3-4 libz3-dev
llvm-15-runtime llvm-15-tools occt-misc python3-wxgtk4.0 x11proto-dev
xorg-sgml-doctools xtrans-dev
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
gnat*
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 15.4 kB disk space will be freed.
(Reading database ... 245808 files and directories currently installed.)
Removing gnat (12.2ubuntu1) ...
-----eSim Uninstalled Successfully-----
jay@jay-VirtualBox: ~/Downloads/eSim-2.4-ht/eSim-2.4$
```

Figure 5.2: Verification of successful uninstallation

5.5 Conclusion

This chapter confirmed that the modifications made to the installation script effectively resolved compatibility issues, ensuring that eSim and its dependencies function correctly. The successful execution of test cases validates that users can seamlessly install and utilize eSim for circuit simulation and verification.

Additionally, the uninstallation process was tested and verified to ensure that eSim and its dependencies can be removed cleanly without leaving unnecessary files or configurations. This provides flexibility for users who may need to reinstall eSim or switch between different versions, making the installation process more robust and user-friendly.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The installation of eSim 2.4 on Ubuntu 23.04 presented several challenges due to outdated dependencies, incompatibilities, and missing package sources. The primary issues included improper installation of NgVeri dependencies, the unavailability of LLVM 9 in the default repositories, GHDL failing due to LLVM version conflicts, and Verilator requiring additional configurations for successful compilation.

To address these challenges, the installation script was systematically modified. A virtual environment was implemented to ensure proper installation of NgVeri dependencies without affecting system-wide packages. The script was updated to install LLVM version 15 instead of LLVM 9, ensuring compatibility with Ubuntu 23.04. Additionally, the manual installation of GHDL using `wget` resolved its compatibility issues. Furthermore, necessary modifications were made to the Verilator installation process to prevent build failures. The script was also enhanced to verify and update `sources.list` when required, reducing dependency-related errors.

Through these modifications, the installation process became more reliable, reducing manual intervention and ensuring smooth deployment of eSim. This work highlights the importance of keeping installation scripts updated to match evolving system environments.

6.2 Future Scope

While the modified script successfully resolves the existing installation issues, further improvements can be explored to enhance its efficiency and adaptability:

- **Automated Dependency Resolution:** Future iterations of the script can include automated checks to detect missing dependencies dynamically and install appropriate versions based on the user's operating system.
- **Version Detection and Compatibility Checks:** Integrating version detection mechanisms can help the script adapt to different Ubuntu releases, ensuring that it selects compatible software versions automatically.

- **Precompiled Binary Packages:** Instead of manually compiling tools like GHDL and Verilator, precompiled binaries can be used to speed up installation and reduce potential build errors.
- **Docker-based Installation:** A containerized approach using Docker could be explored to provide a pre-configured environment for eSim, eliminating compatibility issues across different operating systems.
- **Continuous Integration and Testing:** Regular testing on different Ubuntu versions through CI/CD pipelines can help ensure that future updates do not introduce new compatibility issues.

By incorporating these improvements, the script can be made more robust, user-friendly, and adaptable to evolving software environments, further simplifying the installation process for future users.

Bibliography

- [1] FOSSEE Official Website, 2020. Available at:
<https://fossee.in/about>
- [2] eSim Official Website, 2020. Available at:
<https://esim.fossee.in/>
- [3] Ubuntu Official Webpage, 2020. Available at:
<https://ubuntu.com/tutorials/.../command-line-for-beginners>
- [4] GitHub - FOSSEE eSim Repository, 2020. Available at:
<https://github.com/FOSSEE/eSim>
- [5] GitHub - FOSSEE NGHDL Repository, 2020. Available at:
<https://github.com/FOSSEE/nghdl>
- [6] install-eSim.sh, 2024. Available at:
<https://github.com/FOSSEE/eSim/.../install-eSim.sh>
- [7] install-nghdl.sh, 2024. Available at:
<https://github.com/FOSSEE/nghdl/.../install-nghdl.sh>
- [8] Ubuntu 23.04 End of Life, 2024. Available at:
<https://fridge.ubuntu.com/.../ubuntu-23-04-lunar-lobster-reached-end-of-life>