Winter internship Report

On

Upgradation of eSim installer to Ubuntu 22.04(Seamless Upgradaion)

Submitted by

Sangati shiva krishna reddy

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

February 19, 2025

Acknowledgment

I would like to express my heartfelt gratitude and appreciation to the entire FOSSEE eSim team for providing me with the incredible opportunity to be a part of their esteemed organization as an intern. My experience with the FOSSEE eSim internship program has been nothing short of exceptional, and I am immensely grateful for the valuable knowledge and skills I have gained during this period.

I would like to express my special appreciation to my mentors Sumanto Kar Sir, whose guidance and mentorship have been instrumental in shaping my internship experience. Their patience, knowledge, and encouragement have helped me develop a deeper understanding of eSim and its applications. Working under their supervision has not only enhanced my technical skills but also instilled in me a sense of confidence to tackle complex challenges.

I am also grateful to the entire FOSSEE eSim community for their collaborative spirit and the welcoming environment they foster. Interacting with fellow interns and contributors has been an enriching experience, allowing me to exchange ideas, seek advice, and engage in meaningful discussions. The communitys commitment to open-source values and the shared goal of making quality education accessible to all has been truly inspiring.

I am immensely grateful for the invaluable learning experiences, the supportive community, and the opportunities for personal and professional growth that the FOSSEE eSim internship has provided me. I am confident that the skills and knowledge I have acquired will continue to benefit me in my future endeavors. Once again, thank you to the entire FOSSEE eSim team for this incredible opportunity.

Contents

Chapter 1

Introduction

The Free/Libre and Open Source Software for Education (FOSSEE) project promotes the use of FLOSS tools to enhance the quality of education in India. Its primary objective is to reduce dependency on proprietary software in educational institutions by advocating for equivalent open-source alternatives.

FOSSEE achieves this through various initiatives, including:

- Encouraging FLOSS adoption in academia and research.
- Developing and upgrading open-source tools to meet educational and industrial requirements.
- Conducting training programs, workshops, and collaborative projects to spread awareness.

The FOSSEE project is a part of the National Mission on Education through Information and Communication Technology (ICT), Ministry of Human Resource Development (MHRD), Government of India.

eSim is a free and open-source Electronic Design Automation (EDA) tool developed by the FOSSEE team at IIT Bombay. It serves as an alternative to proprietary EDA tools by integrating multiple open-source software packages, creating a unified environment for circuit design, simulation, analysis, and PCB development.

eSim incorporates the following open-source tools to enhance its functionality:

- KiCad for schematic capture and PCB layout design.
- Ngspice for analog circuit simulation.
- GHDL for digital circuit simulation using VHDL.
- OpenModelica for system modeling and analysis.
- Verilator for high-performance digital verification.
- Makerchip for cloud-based HDL simulation and verification.
- SkyWater SKY130 PDK for open-source semiconductor design.

Importance in Circuit Design and Simulation.
eSim provides a comprehensive EDA environment, enabling users to:

- **Design Schematics:** Create detailed circuit diagrams using KiCad.
- **Simulate Circuits:** Perform simulations with Ngspice for analog circuits and GHDL for digital circuits.
- **Analyze Performance:** Utilize OpenModelica for system modeling and analysis.
- **Design PCBs:** Develop printed circuit board layouts and generate manufacturing files.

By combining these functionalities, eSim streamlines the workflow from circuit design to PCB production, enhancing efficiency and reducing errors.

Common Use Cases and User Base
Due to its open-source nature and feature-rich environment, eSim is widely adopted by:

- Students for hands-on learning in circuit design and simulation.
- Researchers to prototype and validate new circuit concepts.
- Engineers for designing and testing circuits before moving to production.

eSim empowers individuals and institutions to engage in electronics design without the constraints of proprietary software, making it a valuable tool for education, research, and industry.

Chapter 2

Problem Statement and Approach

With the release of Ubuntu 22.04 (Jammy Jellyfish), various software packages and dependencies have undergone significant updates. eSim, an open-source EDA tool for circuit design and simulation, was initially developed and tested on older versions of Ubuntu. However, attempting to install and run eSim 2.4 on Ubuntu 22.04 results in compatibility issues due to outdated dependencies and missing package support.

To analyze and resolve these issues, Ubuntu 22.04 was installed using an ISO file on Oracle VirtualBox. This provided an isolated environment for testing, debugging, and upgrading eSim 2.4 without affecting the host operating system.

Some key challenges encountered during this process include:

- **Dependency Issues:** Several libraries and tools required by eSim 2.4 are no longer available in the standard repositories of Ubuntu 22.04.
- **Version Mismatches:** Software such as KiCad, LLVM, GHDL,Verilator and Python packages have undergone major updates, making older versions incompatible.
- **Error Resolution:** Errors related to package installation, missing repositories, and broken dependencies need to be addressed to ensure a seamless installation and execution process.
- **Performance and Security Enhancements:** Updating to newer software versions ensures better performance, stability, and security.

2.1 Approach Taken

To resolve the above challenges and successfully upgrade eSim 2.4 for Ubuntu 22.04, the following approach was taken:

2.1.1 Installing Ubuntu 22.04 in a Virtual Environment

To ensure a controlled testing environment, Ubuntu 22.04 was installed on **Oracle VirtualBox** using an ISO file. This setup allowed for:

- Testing software compatibility without affecting the primary operating system.

- Easily reverting to snapshots if any installation caused critical failures.
- Debugging installation issues systematically in an isolated environment.

2.1.2 Analyzing Software Dependencies

A detailed study of the required dependencies was conducted to identify outdated and missing packages. Compatibility with Ubuntu 22.04 was verified for each required component.

2.1.3 Identifying and Resolving Package Installation Issues

For packages that were no longer available in the standard repositories, alternative installation methods were used, such as:

- Installing via Personal Package Archives (PPAs) where available.
- Using tarball installations for unsupported software.
- Creating virtual environments for Python package dependencies.

2.1.4 Version Upgrades and Functional Changes

A structured approach was followed to upgrade essential components of eSim to ensure compatibility with Ubuntu 22.04:

- **KiCad Upgrade:** Transitioning from version 6.0 to 8.0 to support newer schematic capture and PCB layout features.
- **LLVM Upgrade:** Upgrading from LLVM 9 to LLVM 13 to resolve missing repository issues.
- **GHDL Upgrade:** Moving from GHDL 0.37 to 4.1.0, installed via tarball since older repositories were unavailable.
- **Verilator Upgrade:** Upgraded from Verilator 4.210 to Verilator 5.032 to improve simulation performance, enhance SystemVerilog support, and resolve compatibility issues with newer toolchains.
- **Python Dependencies:** Managing missing modules such as `python3-distutils` by using virtual environments.

2.1.5 Error Handling and Debugging

During the upgrade process, multiple errors were encountered, including:

- **Repository Not Found:** Packages like KiCad 6.0 were unavailable in Ubuntu 22.04 repositories, requiring manual source installation.
- **Package Installation Failures:** Errors such as missing `llvm-9` and `llvm-9-dev` required an upgrade to LLVM 13.

- **Broken Dependencies:** The removal of `python3-distutils` from the default Ubuntu repositories led to installation failures, which were resolved using virtual environments.

2.1.6 Testing and Verification

After successful upgradation, rigorous testing was conducted to verify the proper functionality of eSim. The following aspects were tested:

- **Successful Installation:** Ensuring all components installed without errors.
- **Compatibility Check:** Verifying that eSim runs properly with upgraded components.
- **Simulation Accuracy:** Checking that circuit simulations produce expected results.

2.2 Outcome

By following the above approach, eSim 2.4 was successfully upgraded and made fully compatible with Ubuntu 22.04. The key outcomes include:

- All missing dependencies were installed using alternative methods.
- Essential software components were upgraded to the latest supported versions.
- Compatibility issues were resolved, and eSim runs without errors.
- Performance and stability improvements were achieved with the latest software updates.

This report provides a comprehensive analysis of the upgrade process, detailing the challenges encountered, solutions implemented, and the successful transition to Ubuntu 22.04.

Chapter 3

Version Upgrades

3.1 KiCad

KiCad is a free and open-source electronics design automation (EDA) suite. It features schematic capture, integrated circuit simulation, printed circuit board (PCB) layout, 3D rendering, and plotting/data export to numerous formats. KiCad also includes a high-quality component library featuring thousands of symbols, footprints, and 3D models. KiCad has minimal system requirements and runs on Linux, Windows, and macOS.

3.1.1 Version Upgrade Details

- **Previous Version:** KiCad 6
- **Updated Version:** KiCad 8

3.1.2 Necessity of Upgrading to KiCad 8

The upgrade from KiCad 6 to KiCad 8 was essential for the following reasons:

- **Compatibility with Ubuntu 22.04:** KiCad 6 had dependency conflicts with newer Ubuntu 22.04, leading to installation and runtime issues.
- **Enhanced Performance:** KiCad 8 introduced better memory management, reducing crashes and improving stability.
- **Improved User Interface:** A more modern UI with refined toolbar arrangements, better component searching, and enhanced usability.
- **Advanced Simulation Features:** New SPICE simulation capabilities, making it more efficient for circuit validation.
- **Better PCB Design Features:** Improvements in routing, DRC (Design Rule Check), and footprint management.
- **Updated Library and Symbol Management:** More extensive library support with better component organization.

3.1.3 Key Improvements in KiCad 8

The major improvements in KiCad 8 that benefited the upgrade process include:

1. **Faster PCB Rendering:** KiCad 8 introduced an optimized graphics engine for better rendering performance.
2. **Multi-Threaded Processing:** Improved efficiency in schematic and PCB layout operations.
3. **Enhanced Gerber Exporting:** More accurate and streamlined export options for manufacturing.
4. **New Python API Features:** Better scripting support for automation in circuit design.

3.1.4 Impact on eSim Integration

- **Resolved Compatibility Issues:** KiCad 8 fixed issues that previously caused crashes in eSim when working with complex PCB layouts.
- **Improved Workflow:** Enhanced GUI and component libraries streamlined the design process.
- **Stability Improvements:** Reduced crashes and enhanced performance in handling large circuit projects.

3.2 LLVM

LLVM is a collection of modular and reusable compiler and toolchain technologies used for developing compiler frontends and backends. It is widely used in various programming environments, including eSim, for code optimization and compilation.

3.2.1 Version Upgrade Details

- **Previous Version:** LLVM 9
- **Updated Version:** LLVM 13

3.2.2 Necessity of Upgrading to LLVM 13

The upgrade from LLVM 9 to LLVM 13 was necessary due to the following reasons:

- **Incompatibility with Ubuntu 22.04:** LLVM 9 was not available in the default Ubuntu 22.04 repositories, causing installation failures.
- **Dependency Issues:** eSim requires newer LLVM versions for proper compilation and execution.

- **Performance Improvements:** LLVM 13 provides better optimization techniques, reducing compilation time.
- **Support for Newer C++ Standards:** LLVM 13 has improved support for C++17 and C++20, making the development process smoother.
- **Security Fixes and Stability Enhancements:** Bug fixes and security patches were applied in LLVM 13, reducing vulnerabilities.

3.2.3 Impact on eSim Integration

The upgrade to LLVM 13 provided the following benefits:

- **Resolved Installation Issues:** Allowed seamless installation on Ubuntu 22.04.
- **Improved Compilation Efficiency:** Faster compilation time and optimized code generation.
- **Better Debugging Support:** LLVM 13 enhanced debugging capabilities for eSim.
- **Future Compatibility:** Ensured long-term compatibility with upcoming eSim versions.

3.3 GHDL

GHDL is an open-source VHDL simulator that enables hardware designers to analyze, debug, and validate digital designs using VHDL. It is crucial for simulations in eSim.

3.3.1 Version Upgrade Details

- **Previous Version:** GHDL 0.37
- **Updated Version:** GHDL 4.1.0

3.3.2 Necessity of Upgrading to GHDL 4.1.0

The upgrade from GHDL 0.37 to 4.1.0 was necessary due to the following reasons:

- **Incompatibility with Ubuntu 22.04:** GHDL 0.37 was not available in the default Ubuntu 22.04 repositories.
- **Performance Improvements:** GHDL 4.1.0 provides faster simulation speeds and better optimization for VHDL designs.
- **Extended VHDL-2008 Support:** Improved support for the latest VHDL-2008 features.

- **Improved Debugging and Logging:** Better error handling and debugging tools.
- **Enhanced Compatibility with GCC and LLVM:** GHDL 4.1.0 includes better integration with modern compilers.

3.3.3 Impact on eSim Integration

The upgrade to GHDL 4.1.0 provided the following benefits:

- **Resolved Installation Issues:** Allowed smooth installation on Ubuntu 22.04.
- **Enhanced Simulation Performance:** Faster and more reliable processing of VHDL designs.
- **Improved Debugging and Error Handling:** Reduced runtime errors and better simulation logs.
- **Future Compatibility:** Ensured long-term support for upcoming eSim versions.

3.3.4 Verilator

Verilator is an open-source high-performance Verilog simulator widely used for functional verification of digital circuits. It converts Verilog code into C++ or SystemC, enabling fast and efficient simulation of large hardware designs. Due to its speed and accuracy, Verilator is extensively used in academia and industry for digital circuit modeling and verification.

3.3.5 Version Upgrade Details

- **Previous Version:** Verilator 4.210 (installed using tarball)
- **Updated Version:** Verilator 5.032 (installed from Verilator GitHub repository)

3.3.6 Necessity of Upgrading to Verilator 5.032

The upgrade from Verilator 4.210 to 5.032 was essential due to the following reasons:

- **Improved Installation Method:** Previously, Verilator was installed using tarball extraction, which required manual configuration. The updated installation directly pulls the latest stable release from the Verilator GitHub repository, ensuring seamless updates and version control.
- **Compatibility with Ubuntu 22.04:** Verilator 4.210 had certain dependency conflicts with Ubuntu 22.04, making installation and execution challenging.

- **Enhanced Performance and Speed:** Verilator 5.032 includes significant optimizations that improve simulation speed and memory efficiency.
- **Expanded Language Support:** Support for SystemVerilog constructs and enhancements in Verilog parsing for better compliance with IEEE standards.
- **Improved Multithreading:** New parallel simulation capabilities, reducing simulation runtime for large circuits.
- **Better Debugging Features:** Enhanced tracing and waveform generation capabilities for easier debugging of digital designs.
- **Bug Fixes and Stability Improvements:** Addressed several known issues in Verilator 4.210, leading to a more stable simulation environment.

3.3.7 Key Improvements in Verilator 5.032

The major improvements in Verilator 5.032 that benefited the upgrade process include:

1. **Faster Compilation and Execution:** Improved performance through better code generation and optimization techniques.
2. **Extended SystemVerilog Support:** Improved compatibility with advanced SystemVerilog features, including enhanced assertion handling.
3. **Optimized Memory Usage:** Reduced memory footprint when simulating large circuits, making it more efficient.
4. **Better Integration with Open-Source EDA Tools:** Improved interoperability with tools like GHDL, Yosys, and eSim.

3.3.8 Impact on eSim Integration

- **Seamless Simulation Workflow:** The upgrade ensures Verilator works efficiently with eSims digital simulation pipeline.
- **Reduced Simulation Time:** Faster execution allows users to verify digital circuits more quickly.
- **Enhanced Stability and Fewer Errors:** Verilator 5.032 minimizes unexpected crashes and improves debugging capabilities in eSim.
- **Improved Compatibility with Mixed-Signal Designs:** Ensures better interoperability when co-simulating digital and analog circuits within eSim.

Chapter 4

Changes Made in Script File

4.1 Updated Verilator Installation Script

The Verilator installation script has been updated to enhance reliability, improve error handling, and switch from tarball-based installation to repository-based installation.

Previous Script Version

The previous script installed Verilator using a tarball extraction approach, manually configuring and building it as follows:

```
function installVerilator
{
    echo "Installing $verilator......................."
    tar -xJf $verilator.tar.xz
    echo "$verilator successfully extracted"
    echo "Changing directory to $verilator installation"
    cd $verilator
    echo "Configuring $verilator build as per requirements"
    chmod +x configure
    ./configure
    make -j$(nproc)
    sudo make install
    echo "Removing the unessential verilator files........"
    rm -r docs
    rm -r examples
    rm -r include
    rm -r test_regress
    rm -r bin
    ls -1 | grep -E -v 'config.status|configure.ac|Makefile.in|verilator.1|configur
    echo "Verilator installed successfully"
    cd ../
}
```

Updated Script Version

The updated script improves upon the previous version by:

- Installing Verilator directly from the GitHub repository instead of using a tarball.
- Adding error handling for each installation step to prevent failures.
- Automating repository updates and selecting the appropriate Verilator version.
- Ensuring cleanup operations are handled safely and effectively.

The new script is as follows:

```
function installVerilator {
    echo "Installing Verilator from repository..."

    sudo apt-get update
    sudo apt-get install -y git help2man perl python3 make autoconf g++ flex bison
        libgoogle-perftools-dev numactl perl-doc libfl2 libfl-dev \
        zlib1g zlib1g-dev || { echo "Error: Failed to install dependencies"; exit 1

    if [ ! -d "verilator" ]; then
        git clone https://github.com/verilator/verilator || { echo "Error: Failed t
    fi

    cd verilator || { echo "Error: Failed to enter Verilator directory"; exit 1; }

    git pull || { echo "Error: Failed to update Verilator repository"; exit 1; }

    git checkout stable

    unset VERILATOR_ROOT

    autoconf || { echo "Error: autoconf failed"; exit 1; }
    ./configure || { echo "Error: Configuration failed"; exit 1; }

    make -j $(nproc) || { echo "Error: Build failed"; exit 1; }
    sudo make install || { echo "Error: Installation failed"; exit 1; }
    verilator --version || { echo "Error: Verilator installation verification faile

    echo "Removing unessential Verilator files..."
    rm -rf docs examples include test_regress bin || { echo "Error: Failed to remov
    ls -1 | grep -E -v 'config.status|configure.ac|Makefile.in|verilator.1|configur

    echo "Verilator installed successfully!"
    cd ..
}
```

4.2 Changes Made in KiCad Installation and Library Copy Scripts

Updated KiCad Installation Script

The KiCad installation script has been updated to install the latest version (KiCad 8.0) instead of the older KiCad 6.0. The script also includes better repository checking and a streamlined installation process.

Previous Script Version

The previous script installed KiCad 6.0 and included additional KiCad libraries:

```
function installKicad
{
    echo "Installing KiCad........................."

    kicadppa="kicad/kicad-6.0-releases"
    findppa=$(grep -h -r "^deb.*$kicadppa*" /etc/apt/sources.list* > /dev/null 2>&1
    if [ -z "$findppa" ]; then
        echo "Adding KiCad-6 ppa to local apt-repository"
        sudo add-apt-repository -y ppa:kicad/kicad-6.0-releases
        sudo apt-get update
    else
        echo "KiCad-6 is available in synaptic"
    fi

    sudo apt-get install -y --no-install-recommends kicad kicad-footprints kicad-li
}
```

Updated Script Version

The updated script now installs KiCad 8.0, ensuring compatibility with the latest release and improving efficiency:

```
function installKicad {
    echo "Installing KiCad..."

    kicadppa="kicad/kicad-8.0-releases"
    findppa=$(grep -h -r "^deb.*$kicadppa*" /etc/apt/sources.list* > /dev/null 2>&1

    if [ -z "$findppa" ]; then
        echo "Adding KiCad-8 PPA to local apt repository..."
        sudo add-apt-repository -y ppa:kicad/kicad-8.0-releases
        sudo apt update
    else
        echo "KiCad-8 PPA is already added."
    fi

    echo "Installing KiCad and necessary libraries..."
    sudo apt install -y --no-install-recommends kicad
```

```
    echo "KiCad installation completed successfully!"
}
```

Key Improvements

- Upgraded KiCad installation from version 6.0 to 8.0.
- Simplified package installation by removing unnecessary libraries.
- Improved repository check and update mechanism.

Updated KiCad Library Copy Script

The KiCad library copy script has been enhanced to improve error handling, detect the latest KiCad version dynamically, and ensure smoother installation of custom libraries.

Previous Script Version

The older script used a fixed path for KiCad 6.0 configuration and manually copied files:

```
function copyKicadLibrary
{
    # Extract custom KiCad Library
    tar -xJf library/kicadLibrary.tar.xz

    if [ -d ~/.config/kicad/6.0 ]; then
        echo "kicad config folder already exists"
    else
        echo ".config/kicad/6.0 does not exist"
        mkdir -p ~/.config/kicad/6.0
    fi

    # Copy symbol table for eSim custom symbols
    cp kicadLibrary/template/sym-lib-table ~/.config/kicad/6.0/
    echo "symbol table copied in the directory"

    # Copy KiCad symbols made for eSim
    sudo cp -r kicadLibrary/eSim-symbols/* /usr/share/kicad/symbols/

    set +e      # Temporarily disable exit on error
    trap "" ERR # Do not trap on error of any command

    # Remove extracted KiCad Library - not needed anymore
    rm -rf kicadLibrary

    set -e      # Re-enable exit on error
    trap error_exit ERR
```

```
    # Change ownership from Root to the User
    sudo chown -R $USER:$USER /usr/share/kicad/symbols/
}
```

Updated Script Version

The updated script dynamically detects the installed KiCad version, improves error handling, and enhances reliability:

```
function copyKicadLibrary {
    set -e  # Exit immediately on error
    trap 'echo "An error occurred! Exiting..."; exit 1' ERR

    echo "Extracting custom KiCad Library..."
    tar -xJf library/kicadLibrary.tar.xz -C library || { echo "Extraction failed!";

    # Detect the latest installed KiCad version
    kicad_config_dir="$HOME/.config/kicad"
    latest_version=$(ls "$kicad_config_dir" | grep -E "^[0-9]+\.[0-9]+$" | sort -V

    if [ -z "$latest_version" ]; then
        latest_version="8.0"  # Default to the latest known version
        mkdir -p "$kicad_config_dir/$latest_version"
        echo "Created KiCad config directory: $kicad_config_dir/$latest_version"
    else
        echo "Using existing KiCad version: $latest_version"
    fi

    kicad_version_dir="$kicad_config_dir/$latest_version"

    # Copy the symbol table for eSim custom symbols
    echo "Copying symbol table..."
    cp library/kicadLibrary/template/sym-lib-table "$kicad_version_dir/"

    # Ensure KiCad symbols directory exists
    kicad_symbols_dir="/usr/share/kicad/symbols"
    if [ ! -d "$kicad_symbols_dir" ]; then
        echo "Creating KiCad symbols directory..."
        sudo mkdir -p "$kicad_symbols_dir"
    fi

    # Copy custom symbols using rsync for better reliability
    echo "Copying eSim custom symbols..."
    sudo rsync -av library/kicadLibrary/eSim-symbols/ "$kicad_symbols_dir/"

    # Cleanup: Remove extracted KiCad Library (not needed anymore)
    echo "Removing extracted KiCad library..."
    rm -rf library/kicadLibrary
```

```
    # Change ownership from root to the user only if needed
    if [ "$(stat -c "%U" "$kicad_symbols_dir")" != "$USER" ]; then
        echo "Changing ownership of KiCad symbols directory..."
        sudo chown -R "$USER:$USER" "$kicad_symbols_dir"
    fi

    echo "KiCad Library successfully copied and configured!"
}
```

4.3 SKY130 PDK Installation Script

The SKY130 PDK installation script has been updated to enhance the copying process by using `cp -R` instead of `mv`, ensuring a more reliable copy operation without altering the source directory structure.

Previous Script Version

The previous script used the `mv` command to move the SKY130 PDK library to the system-wide location. The script steps were as follows:

```
function installSky130Pdk
{
    echo "Installing SKY130 PDK....................."

    # Extract SKY130 PDK
    tar -xJf library/sky130_fd_pr.tar.xz

    # Remove any previous sky130-fd-pdr instance, if any
    sudo rm -rf /usr/share/local/sky130_fd_pr

    # Copy SKY130 library
    echo "Copying SKY130 PDK......................."
    sudo mkdir -p /usr/share/local/
    echo "Directory created"
    sudo mv sky130_fd_pr /usr/share/local/
    # placing the SKY130 PDK in a standard, system-wide location.

    # Change ownership from root to the user
    sudo chown -R $USER:$USER /usr/share/local/sky130_fd_pr/
}
```

Updated Script Version

The updated script improves upon the previous version by:

- Replacing the `mv` command with `cp -R` for a safer and more robust copying process.

- Removing the source directory after a successful copy, ensuring a clean working environment.

The new script is as follows:

```
function installSky130Pdk
{
    echo "Installing SKY130 PDK....................."

    # Extract SKY130 PDK
    tar -xJf library/sky130_fd_pr.tar.xz

    # Remove any previous sky130-fd-pdr instance, if any
    sudo rm -rf /usr/share/local/sky130_fd_pr

    # Copy SKY130 library
    echo "Copying SKY130 PDK........................."
    sudo mkdir -p /usr/share/local/
    echo "Directory created"

    sudo cp -R sky130_fd_pr /usr/share/local/
    sudo rm -rf sky130_fd_pr

    # Change ownership from root to the user
    sudo chown -R $USER:$USER /usr/share/local/sky130_fd_pr/
}
```

4.4 Dependency Installation Script

The dependency installation script has been updated to enhance package management by introducing a Python virtual environment. This ensures better isolation of dependencies, reduces system-wide package conflicts, and allows for easier management of the installed packages.

Previous Script Version

The previous script installed all dependencies globally without using a virtual environment. This could potentially lead to conflicts between project dependencies and system-wide Python packages.

```
function installDependency
{
    # Update apt repository
    echo "Updating apt index files..................."
    sudo apt-get update

    echo "Installing Xterm..........................."
    sudo apt-get install -y xterm
```

```
    echo "Installing Psutil........................."
    sudo apt-get install -y python3-psutil

    echo "Installing PyQt5........................."
    sudo apt-get install -y python3-pyqt5

    echo "Installing Matplotlib....................."
    sudo apt-get install -y python3-matplotlib

    echo "Installing Distutils......................"
    sudo apt-get install -y python3-distutils

    # Install NgVeri Dependencies
    echo "Installing Pip3..........................."
    sudo apt install -y python3-pip

    echo "Installing Watchdog......................."
    pip3 install watchdog

    echo "Installing Hdlparse......................."
    pip3 install --upgrade https://github.com/hdl/pyhdlparser/tarball/master

    echo "Installing Makerchip......................"
    pip3 install makerchip-app

    echo "Installing SandPiper Saas................."
    pip3 install sandpiper-saas

    echo "Installing Hdlparse again................."
    pip3 install hdlparse

    echo "Installing Matplotlib....................."
    pip3 install matplotlib

    echo "Installing PyQt5........................."
    pip3 install PyQt5
}
```

Updated Script Version

The updated script introduces the following enhancements:

- Virtual Environment: A virtual environment is created using `virtualenv` to isolate the projects Python packages.
- Package Upgrades: Pip is upgraded within the virtual environment for the latest package support.

- Environment Isolation: All Python dependencies are installed within the virtual environment, reducing the chances of conflicts with system-wide packages.

The updated script is as follows:

```
function installDependency
{
    set +e      # Temporarily disable exit on error
    trap "" ERR # Ignore errors temporarily

    # Update apt repository
    echo "Updating apt index files..................."
    sudo apt-get update

    set -e      # Re-enable exit on error
    trap error_exit ERR

    echo "Installing virtualenv......................."
    sudo apt install python3-virtualenv

    echo "Creating virtual environment to isolate packages"
    virtualenv $config_dir/env

    echo "Starting the virtual env..................."
    source $config_dir/env/bin/activate

    echo "Upgrading Pip..............................."
    pip install --upgrade pip

    echo "Installing Xterm............................"
    sudo apt-get install -y xterm

    echo "Installing Psutil..........................."
    sudo apt-get install -y python3-psutil

    echo "Installing PyQt5............................"
    sudo apt-get install -y python3-pyqt5

    echo "Installing Matplotlib......................."
    sudo apt-get install -y python3-matplotlib

    echo "Installing Distutils........................"
    sudo apt-get install -y python3-distutils

    # Install NgVeri Dependencies
    echo "Installing Pip3............................."
    sudo apt install -y python3-pip
```

```
    echo "Installing Watchdog......................."
    pip3 install watchdog

    echo "Installing Hdlparse......................."
    pip3 install --upgrade https://github.com/hdl/pyhdlparser/tarball/master

    echo "Installing Makerchip......................."
    pip3 install makerchip-app

    echo "Installing SandPiper Saas................."
    pip3 install sandpiper-saas

    echo "Installing Hdlparse again................."
    pip3 install hdlparse

    echo "Installing Matplotlib....................."
    pip3 install matplotlib

    echo "Installing PyQt5.........................."
    pip3 install PyQt5
}
```

Chapter 5

Errors Encountered and Resolutions

5.1 Python3-Distutils Package Installation Error

5.1.1 Error Description

During the installation process of eSim on Ubuntu 22.04, the following error occurred while trying to install `python3-distutils`:

```
Package python3-distutils is not available but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source.

E: Package 'python3-distutils' has no installation candidate.
```

This error occurs because `python3-distutils` is no longer available in the default Ubuntu 22.04 repositories. The package has been removed or replaced in newer versions of Ubuntu.

5.1.2 Resolution

To resolve this issue, a virtual environment was created and used to install the required dependencies. The following steps were taken:

1. **Install Python Virtual Environment** (if not already installed):

   ```
   sudo apt install python3-venv
   ```

2. **Create a Virtual Environment:**

   ```
   python3 -m venv my_env
   ```

3. **Activate the Virtual Environment:**

```
source my_env/bin/activate
```

4. **Install Python3-Distutils inside the virtual environment:**

```
sudo apt install python3-distutils -y
```

5. **Verify Installation:**

```
python3 -m ensurepip
```

6. **Exit the Virtual Environment** when done:

```
deactivate
```

5.1.3 Outcome

After setting up the virtual environment and installing the required dependencies, the installation of eSim proceeded without any further issues.

5.2 LLVM 9 Installation Failure

5.2.1 Error Description

During the installation process of eSim on Ubuntu 22.04, the following error occurred while attempting to install LLVM 9:

```
Installing LLVM-9.....................................
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package llvm-9
E: Unable to locate package llvm-9-dev

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

Aborting Installation...
```

This error indicates that Ubuntu 22.04 does not include LLVM 9 in its package repositories by default, making installation impossible without additional configuration.

5.2.2 Resolution

To resolve this issue, the following steps were taken:

1. Removed any existing references to LLVM 9.
2. Added the official LLVM repository for Ubuntu 22.04:

```
sudo apt update && sudo apt upgrade
sudo apt install llvm-13 llvm-13-dev clang-13 lldb-13 lld-13
```

3. Updated eSim's build configuration to use LLVM 13.
4. Recompiled eSim with LLVM 13 and verified successful execution.

5.2.3 Outcome

After upgrading to LLVM 13, the installation proceeded without errors, and eSim successfully compiled and ran on Ubuntu 22.04.

5.3 GHDL 0.37 Installation Failure

5.3.1 Error Description

During the installation process of eSim on Ubuntu 22.04, the following error occurred when trying to install GHDL 0.37:

```
E: Unable to locate package ghdl-0.37
E: Package 'ghdl-0.37' has no installation candidate

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

Aborting Installation...
```

This error occurred because GHDL 0.37 was not available in Ubuntu 22.04's official package repositories. Since GHDL 4.1.0 was not available via standard package managers, it was installed manually using a tarball file.

5.3.2 Resolution

To resolve this issue, the following steps were taken:

1. Manually downloaded the official GHDL 4.1.0 tarball from the GitHub releases page and incorporated it into the workflow:

```
wget https://github.com/ghdl/ghdl/releases/download/v4.1.0/ghdl-4.1.0.tar.
```

2. Extracted the downloaded tarball:

```
tar -xvf ghdl-4.1.0.tar.xz
```

3. Verified the successful installation of GHDL 4.1.0:

```
ghdl --version
```

4. Incorporated the manually installed GHDL 4.1.0 into the eSim workflow by updating eSim's configuration files to reference the new GHDL version.

5.3.3 Outcome

After upgrading to GHDL 4.1.0, the installation proceeded without errors, and eSim successfully ran VHDL simulations with improved performance and compatibility.

Chapter 6

Conclusion

The upgradation of eSim 2.4 to be compatible with Ubuntu 22.04 was a necessary step to ensure seamless installation, execution, and stability of the software. The process involved resolving multiple dependency issues, upgrading key components such as KiCad, LLVM, and GHDL, and addressing errors related to outdated repositories and missing packages.

By leveraging alternative installation methods, including PPAs, manual package installations, and virtual environments, all compatibility issues were successfully resolved. The upgraded version of eSim now runs efficiently on Ubuntu 22.04, benefiting from enhanced performance, security, and long-term support. This report serves as a comprehensive guide for future upgrades and troubleshooting similar compatibility issues.

Chapter 7

References

The following resources were used for the upgradation process of eSim 2.4:

- GHDL installation details were obtained from GHDL Releases.
- KiCad version updates and installation were referenced from KiCad Ubuntu Download Page.
- Official eSim downloads and documentation were used from eSim Official Downloads.
- LLVM package updates were verified using LLVM Official Website.
- Ngspice and nghdl repository details were taken from nghdl GitHub Repository.