Summer Fellowship Report

On

Bug Fixing and Feature Enhancement of eSim

Submitted by

Abhishek Soni

Under the guidance of

Sumanto Kar

IIT Bombay

July 17, 2025

Acknowledgment

I wish to express my heartfelt gratitude and appreciation to the entire FOSSEE team for having me as a part of their Summer Fellowship Internship program they offered. The experience and knowledge that I have gained during this internship period is hard to express in words. I have gained tremendous insights about various programming concepts and have met very talented and hardworking people as a part of this internship.
I would like express my heartfelt appreciation to my mentor Sumanto Kar sir who has helped me throughout this internship with great patience and diligence. From him I have gained greater insight about eSim and also on how to write better and effective code. Working under him communicating various ideas about the project has helped me gain better grasp at communicating abstract concepts. He has instilled knowledge and confidence throughout this endeavor.
I am very grateful to the whole eSim community for providing me the help needed throughout this internship phase. I am very much confident that the skills that I have gained through this internship will continue to benefit me throughout my career. Once again, I thank the entire FOSSEE team for the unforgettable opportunity.

Contents

1 Introduction **4**
- 1.1 About FOSSEE . . . 4
- 1.2 Overview of eSim and Its Tools . . . 4
- 1.3 Project Scope and Objective . . . 6

2 Key issues and Features **7**
- 2.1 Simulation Crash due to lint_off . . . 7
- 2.2 Missing Model Removal Option . . . 8
- 2.3 Project Folder Not Reopening . . . 8
- 2.4 Lack of Snapshot and Timeline . . . 8
- 2.5 Inconsistent File Sync . . . 9

3 Approach and Methodology **10**
- 3.1 Handling Simulation Crash from lint_off . . . 10
- 3.2 Enabling Model Removal in NGHDL . . . 10
- 3.3 Restoring Last Opened Folder on Restart . . . 11
- 3.4 Introducing Snapshot and Timeline Support . . . 11
- 3.5 Fixing File Sync in Project Explorer . . . 11

4 Implementation Details **13**
- 4.1 Handling Crash from lint_off . . . 13
- 4.2 Model Removal Logic in NGHDL . . . 13
- 4.3 Restoring Last Opened Project Folder . . . 14
- 4.4 Snapshot Creation and Timeline Integration . . . 15
- 4.5 Real-time File Synchronization in Project Explorer . . . 17

5 Results and Testing **18**
- 5.1 Visual Results After Fixes . . . 18
  - 5.1.1 Resolved Crash from lint_off Removal . . . 18
  - 5.1.2 NGHDL Model Deletion Feature . . . 18
  - 5.1.3 Last Opened Project Auto-Restore . . . 19
  - 5.1.4 Snapshot and Timeline Functionality . . . 20
  - 5.1.5 Real-time File Synchronization . . . 20
- 5.2 Stability and Functionality Improvements . . . 20
- 5.3 User Experience Enhancements . . . 21

6 Conclusion and Future Work **23**
6.1 Summary of Contributions 23
6.2 Key Challenges and Resolutions 23
6.3 Recommendations for Future Enhancements 24
6.4 Scope for Community Contribution 25

Chapter 1

Introduction

1.1 About FOSSEE

FOSSEE (Free/Libre and Open Source Software for Education) is a national initiative funded by the Ministry of Education (MoE), Government of India, and implemented by IIT Bombay. The project aims to promote the use and development of free and open-source software (FOSS) in academia and research. With the ever-increasing cost of proprietary software, many educational and research institutions face limitations in accessing advanced tools. FOSSEE addresses this gap by supporting open-source alternatives that are accessible, customizable, and cost-effective.

Through initiatives like tool development, student internships, content creation, and workshops, FOSSEE empowers students, educators, and professionals to adopt open-source tools in their workflows. It not only helps reduce software costs but also builds a community-driven ecosystem for innovation and collaboration. The internship program under FOSSEE allows students to contribute directly to impactful projects such as eSim, thereby gaining real-world experience in software development and problem-solving.

1.2 Overview of eSim and Its Tools

eSim (Electronics Simulation) is an open-source EDA (Electronic Design Automation) software developed by FOSSEE, IIT Bombay. It integrates multiple open-source tools to provide a complete environment for schematic creation, simulation (analog, digital, and mixed-signal), and PCB design. eSim is designed as a powerful, cost-effective alternative to proprietary EDA tools, making it suitable for educational, research, and small-scale industrial applications.

eSim brings together various open-source modules in a unified GUI. Each module plays a specific role in the workflow, from schematic design to simulation and PCB layout:

- **Eeschema:** A schematic editor originally from KiCad, Eeschema is used to create and manage electronic circuit schematics. It supports symbol libraries, annotation, electric rules checking (ERC), and netlist generation.

- **CvPcb:** This tool maps components in the schematic to their physical footprints. It simplifies footprint association through manual or automatic filtering and previewing of component footprints, including 3D visualization.
- **Pcbnew:** Pcbnew is the PCB layout editor used in eSim. It allows for the design of printed circuit boards with features like rats-nest display, design rule checks (DRC), auto-routing, and multi-layer layout support.
- **Ngspice:** Ngspice is the core simulator for analog, digital, and mixed-signal circuits in eSim. It supports DC, AC, and transient analysis and is capable of simulating a wide range of semiconductor devices.
- **KiCad to Ngspice Converter:** This converter bridges the gap between schematic design and simulation. It converts KiCad-generated netlists into Ngspice-compatible formats and allows for detailed configuration of sources, models, and simulation parameters.
- **Model Builder:** This tool enables users to define or modify device models such as diodes, BJTs, MOSFETs, and IGBTs, enhancing the accuracy and customizability of simulations.
- **Subcircuit Builder:** Used to create reusable subcircuits like op-amps or IC modules. It simplifies complex design by allowing hierarchical circuit modeling and easy integration into other projects.
- **NGHDL:** NGHDL integrates VHDL-based digital simulation using GHDL. It enables mixed-signal simulation by interfacing digital logic with analog components through Ngspice.
- **NgVeri:** A Verilog/SystemVerilog simulation module using Verilator and SandPiper SaaS. It supports digital simulation and mixed-signal co-simulation with Ngspice.
- **Makerchip-App:** A cloud-based Verilog IDE interfaced with eSim to allow users to write, simulate, and test Verilog code using the Makerchip platform, especially useful for digital design education.
- **OpenModelica:** Integrated with eSim for advanced system-level modeling and simulation using the Modelica language. It allows users to analyze and optimize complex electrical systems with OMEdit and OMOptim.

By combining these tools into a single platform, eSim provides a comprehensive and flexible workflow for electronics design and simulation, while maintaining full transparency and control through its open-source foundation.

1.3 Project Scope and Objective

The scope of the project undertaken during the internship revolves around improving the usability, functionality, and reliability of the eSim platform. As an evolving open-source tool, eSim continuously benefits from community contributions that enhance its features and fix limitations. The main objective of this project is to support that mission by addressing key usability challenges, enhancing interface elements, and streamlining user workflows within the tool.

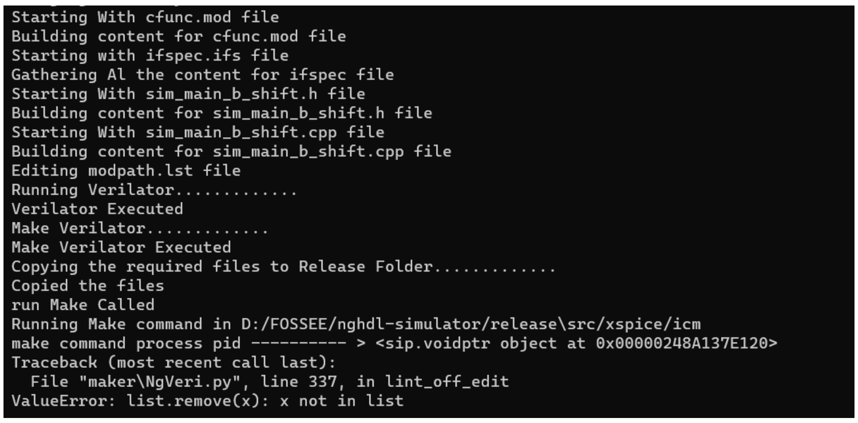
The broader impact of this work lies in strengthening eSim as a viable alternative to proprietary EDA tools, particularly in educational and research institutions where budget constraints are significant. Improvements made during the internship aim to reduce user friction, improve the stability of the tool, and provide a more intuitive and seamless user experience. This not only benefits current users but also encourages wider adoption of eSim in electronics education and small-scale hardware development. Ultimately, the project's goal is to contribute meaningfully to the open-source ecosystem by making high-quality design and simulation tools more accessible.

Chapter 2

Key issues and Features

2.1 Simulation Crash due to lint_off

A critical issue encountered during simulation in eSim was linked to the misuse and handling of the lint_off.txt file. This file contains a list of known linting warnings that the simulator should ignore during code analysis. While the intent is to allow users to selectively suppress warnings, problems arise when the file includes entries that are either misspelled, deprecated, or no longer relevant to the simulation context. During simulation, the system attempts to parse each line of the file and match it against known warning types. If it fails to find a match, the backend fails to handle the mismatch gracefully resulting in a crash that terminates the simulation process without explanation. The problem worsens when users try to modify this list using the graphical interface in eSim. Specifically, when attempting to remove the last entry in the lint_off.txt list using the Remove lint_off button, an unhandled exception occurs. This error suggests that the UI list and the underlying Python list used for tracking entries become unsynchronized. Instead of handling the edge case, the application crashes, leaving the user with no clear understanding of what went wrong. This not only disrupts the workflow but can also discourage less-experienced users from interacting with lint controls.



```
Starting With cfunc.mod file
Building content for cfunc.mod file
Starting with ifspec.ifs file
Gathering Al the content for ifspec file
Starting With sim_main_b_shift.h file
Building content for sim_main_b_shift.h file
Starting With sim_main_b_shift.cpp file
Building content for sim_main_b_shift.cpp file
Editing modpath.lst file
Running Verilator............
Verilator Executed
Make Verilator............
Make Verilator Executed
Copying the required files to Release Folder............
Copied the files
run Make Called
Running Make command in D:/FOSSEE/nghdl-simulator/release\src/xspice/icm
make command process pid ---------- > <sip.voidptr object at 0x00000248A137E120>
Traceback (most recent call last):
  File "maker\NgVeri.py", line 337, in lint_off_edit
ValueError: list.remove(x): x not in list
```

Figure 2.1: Unhandled exception while removing last lint_off entry

2.2 Missing Model Removal Option

Within the NGHDL (Next Generation Hardware Description Language) interface of eSim, there was no built-in option to remove or delete user-added models. While the interface provided functionality for uploading and selecting custom VHDL models for digital simulation, users were left without a direct way to manage or delete these models once they were no longer needed. This limitation often led to a cluttered workspace and confusion, especially for users experimenting with multiple model versions.

The lack of a model removal feature also posed usability and maintenance issues. Users either had to manually locate and delete the model files from internal foldersrisking accidental deletion of essential filesor continue working with an overloaded interface containing obsolete or incorrect entries. This missing feature reduced the efficiency of the workflow, particularly in academic or development environments where iterative testing of models is common.

2.3 Project Folder Not Reopening

Another notable issue was that eSim failed to reopen the previously accessed project folder upon restarting the application. Each time users relaunched eSim, they were required to manually browse to and open their working directory. This disrupted the user experience, particularly for those working on large or long-term projects where frequent reopening of the software was necessary.

This lack of persistent session memory proved inconvenient and time-consuming. It also increased the chances of users mistakenly working in unintended directories or losing track of their last active project. For new users, this behavior could lead to the false assumption that their project data was lost or not saved properly, creating a usability barrier in an otherwise streamlined development tool.

2.4 Lack of Snapshot and Timeline

eSim initially lacked a dedicated feature to take snapshots of the project state or view historical versions in a timeline. This became problematic for users who frequently made changes to circuit designs and simulations, especially when they needed to revert to a previous working version. Without such functionality, users had to rely on external backups or manually copy folders, which is inefficient and error-prone.

The absence of a snapshot and timeline mechanism made version control cumbersome and discouraged experimentation. Users working on academic or collaborative projects often needed to track progress, roll back recent changes, or compare different circuit versions. Without this capability, managing iterations and maintaining a reliable workflow within the tool was a significant challenge.

2.5 Inconsistent File Sync

In the Project section of eSim, a usability issue was observed where file changes made externally (outside of eSim) were not reflected in real-time. For instance, when a file was deleted or modified directly from the operating system, the changes did not immediately appear in the file tree unless the user manually collapsed and re-expanded the project folder. This static file view created a mismatch between the actual project state and what was shown in the interface.

This lack of dynamic file synchronization was especially disruptive for users who frequently switched between code editors and eSim or used scripts to modify files. It not only delayed updates but also introduced a risk of working with outdated files unknowingly. For larger projects with many files, this inconsistency made file management inside eSim less reliable and intuitive.

Chapter 3

Approach and Methodology

3.1 Handling Simulation Crash from lint_off

To resolve the simulation crash caused by the lint_off.txt file, I focused on the part of the backend code responsible for removing specific entries from this file. The crash occurred when the system attempted to remove an entry that did not exist in the underlying list, especially when users tried to delete the last remaining item. This led to an unhandled exception and abrupt termination of eSim.

To fix this, I introduced a try except block around the code segment handling entry removal. This ensured that any unexpected failure during the deletion process such as attempting to remove a non-existent item would be gracefully handled. Instead of crashing, the application now shows a dialog box with the message *"Could not remove lint_off entry"*, informing the user of the issue without disrupting the session. This approach improved both stability and user experience.

3.2 Enabling Model Removal in NGHDL

Previously, NGHDL in eSim allowed users to upload digital models, but lacked any functionality to remove them through the interface. This limitation made it difficult for users to manage outdated or unused models, especially in iterative testing environments. To resolve this, the model storage structure and internal reference system were analyzedspecifically across directories like ghdl, and files such as modpath.lst, KiCad symbol files, and associated XML configurations.

After mapping out the lifecycle of a model across the system, implemented a model deletion workflow that could be triggered via a new button in the NGHDL dialog. The backend logic was designed to remove all relevant components of the selected model, including its directory, entry in the modpath.lst file, corresponding XML metadata, symbol blocks in KiCad libraries, and any residual files in the DUTghdl testbench folder. To ensure robustness, added a confirmation dialog for users before deletion, and surrounded all file operations with exception handling. If any step in the removal process fails, an error message is displayed: *"An error occurred while deleting the model"*, preventing unexpected crashes.

3.3 Restoring Last Opened Folder on Restart

The absence of persistent session handling in eSim created a disruption in user workflow, as the last accessed project folder would not be restored upon restarting the application. Previously, the project path was maintained only as a temporary variable in the backend during runtime. Once eSim was closed, this variable was reset to None, resulting in a loss of context when the application was reopened.

To address this limitation, a persistent storage mechanism is introduced using a lightweight configuration file named last_project.json. During application closure, the path of the most recently opened project is written to this file. Upon the next launch, eSim reads the stored path from last_project.json and automatically loads the corresponding project into the backend. This enhancement restores continuity, saves time, and improves the overall user experience by eliminating the need to manually browse and reopen the last project.

3.4 Introducing Snapshot and Timeline Support

The absence of a snapshot or version control feature in eSim made it difficult for users to manage and recover previous states of individual project files. To overcome this limitation, a file tracking mechanism was implemented to allow users to manually save and manage snapshots of important files. This functionality was integrated directly into the Project Explorer for ease of access.

The mechanism works by enabling users to right-click on any file in the Project Explorer, where a new "Snapshot" option appears. Upon selection, the current state of that file is saved to a separate folder, while its name and the timestamp of the snapshot are logged and displayed in a newly added "Timeline" section under the project panel. The Timeline section includes two main actions: Restore and Clear. If a file is selected from the timeline, the user can either restore its saved state overwriting the existing file in the project directory or delete the snapshot from the separate folder. If no specific file is selected, clicking the buttons performs the respective operation on all snapshot entries at once. This setup provides a lightweight version control system that enhances project safety and flexibility during iterative development.

3.5 Fixing File Sync in Project Explorer

Previously, the Project Explorer in eSim did not reflect file changes made outside the application in real time. If a user deleted or modified a file directly from the operating system while the project was open and expanded in eSim, the interface continued to display outdated file listings until manually refreshed. This created inconsistencies between the actual project structure and what was shown in the UI.

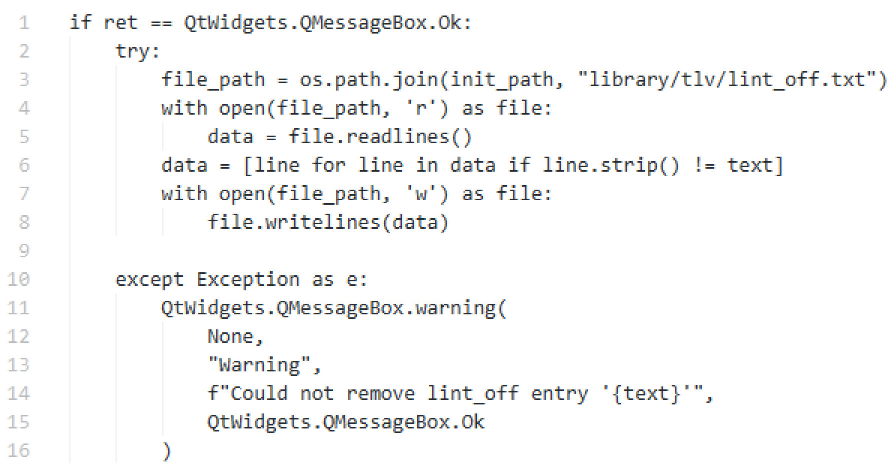
To address this, a file system monitoring mechanism was introduced to detect external file events. With this enhancement, whenever a file is deleted from the project directory through the operating system, the Project Explorer widget immediately updates to reflect the change even if the folder is currently expanded in the GUI. This ensures that the displayed file tree remains consistent with the actual directory structure, eliminating the need for manual user intervention and improving the overall responsiveness of the interface.

Chapter 4

Implementation Details

4.1 Handling Crash from lint_off

To prevent eSim from crashing when the last item is removed from the lint_off.txt list, error handling was added to the backend logic. The system now attempts to remove the selected entry using a try except block. If the entry does not exist in the internal list, a warning message is shown to the user instead of terminating the application. The following code was implemented in src/maker/NgVeri.py.

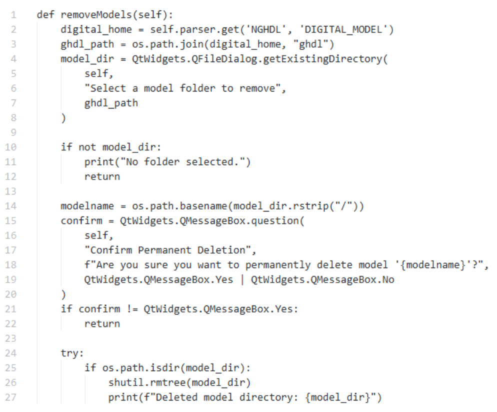


```
if ret == QtWidgets.QMessageBox.Ok:
    try:
        file_path = os.path.join(init_path, "library/tlv/lint_off.txt")
        with open(file_path, 'r') as file:
            data = file.readlines()
        data = [line for line in data if line.strip() != text]
        with open(file_path, 'w') as file:
            file.writelines(data)

    except Exception as e:
        QtWidgets.QMessageBox.warning(
            None,
            "Warning",
            f"Could not remove lint_off entry '{text}'",
            QtWidgets.QMessageBox.Ok
        )
```

Figure 4.1: Code to handle exception while removing lint_off entry

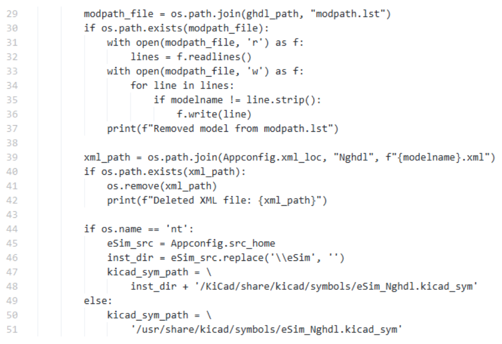
4.2 Model Removal Logic in NGHDL

To enable model deletion within the NGHDL interface, a backend function was implemented to remove all associated files of a selected model. This includes deleting the models directory, its XML file, references in the modpath.lst file, and the corresponding KiCad symbol. Additionally, a confirmation dialog was added to avoid accidental deletions, and exception handling was included to manage errors gracefully. The following code was implemented in nghdl/src/ngspice_ghdl.py.



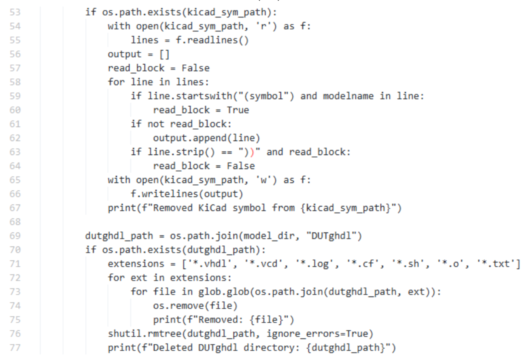
```python
def removeModels(self):
    digital_home = self.parser.get('NGHDL', 'DIGITAL_MODEL')
    ghdl_path = os.path.join(digital_home, "ghdl")
    model_dir = QtWidgets.QFileDialog.getExistingDirectory(
        self,
        "Select a model folder to remove",
        ghdl_path
    )

    if not model_dir:
        print("No folder selected.")
        return

    modelname = os.path.basename(model_dir.rstrip("/"))
    confirm = QtWidgets.QMessageBox.question(
        self,
        "Confirm Permanent Deletion",
        f"Are you sure you want to permanently delete model '{modelname}'?",
        QtWidgets.QMessageBox.Yes | QtWidgets.QMessageBox.No
    )
    if confirm != QtWidgets.QMessageBox.Yes:
        return

    try:
        if os.path.isdir(model_dir):
            shutil.rmtree(model_dir)
            print(f"Deleted model directory: {model_dir}")
```

(a)



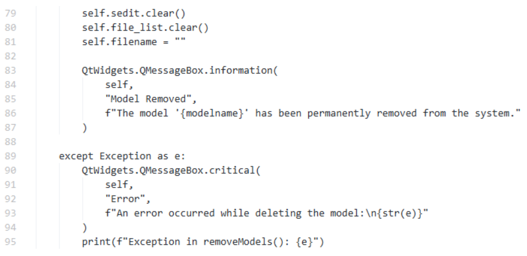
```python
        modpath_file = os.path.join(ghdl_path, "modpath.lst")
        if os.path.exists(modpath_file):
            with open(modpath_file, 'r') as f:
                lines = f.readlines()
            with open(modpath_file, 'w') as f:
                for line in lines:
                    if modelname != line.strip():
                        f.write(line)
            print(f"Removed model from modpath.lst")

        xml_path = os.path.join(Appconfig.xml_loc, "Nghdl", f"{modelname}.xml")
        if os.path.exists(xml_path):
            os.remove(xml_path)
            print(f"Deleted XML file: {xml_path}")

        if os.name == 'nt':
            eSim_src = Appconfig.src_home
            inst_dir = eSim_src.replace('\\eSim', '')
            kicad_sym_path = \
                inst_dir + '/KiCad/share/kicad/symbols/eSim_Nghdl.kicad_sym'
        else:
            kicad_sym_path = \
                '/usr/share/kicad/symbols/eSim_Nghdl.kicad_sym'
```

(b)



```python
        if os.path.exists(kicad_sym_path):
            with open(kicad_sym_path, 'r') as f:
                lines = f.readlines()
            output = []
            read_block = False
            for line in lines:
                if line.startswith("(symbol") and modelname in line:
                    read_block = True
                if not read_block:
                    output.append(line)
                if line.strip() == "))" and read_block:
                    read_block = False
            with open(kicad_sym_path, 'w') as f:
                f.writelines(output)
            print(f"Removed KiCad symbol from {kicad_sym_path}")

        dutghdl_path = os.path.join(model_dir, "DUTghdl")
        if os.path.exists(dutghdl_path):
            extensions = ['*.vhdl', '*.vcd', '*.log', '*.cf', '*.sh', '*.o', '*.txt']
            for ext in extensions:
                for file in glob.glob(os.path.join(dutghdl_path, ext)):
                    os.remove(file)
                    print(f"Removed: {file}")
            shutil.rmtree(dutghdl_path, ignore_errors=True)
            print(f"Deleted DUTghdl directory: {dutghdl_path}")
```

(c)



```python
        self.sedit.clear()
        self.file_list.clear()
        self.filename = ""

        QtWidgets.QMessageBox.information(
            self,
            "Model Removed",
            f"The model '{modelname}' has been permanently removed from the system."
        )

    except Exception as e:
        QtWidgets.QMessageBox.critical(
            self,
            "Error",
            f"An error occurred while deleting the model:\n{str(e)}"
        )
        print(f"Exception in removeModels(): {e}")
```

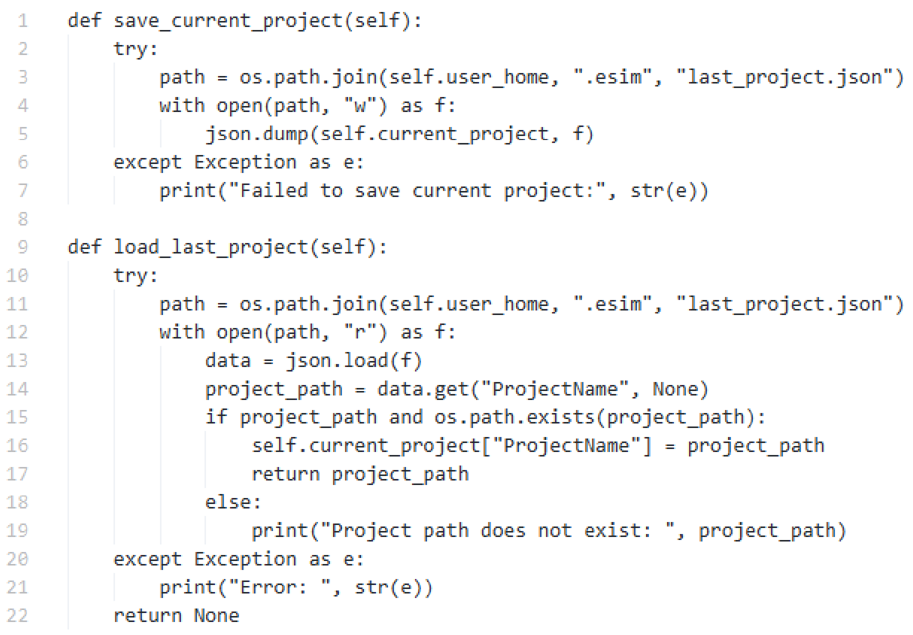
(d)

Figure 4.2: Code implementation for NGHDL model removal

4.3 Restoring Last Opened Project Folder

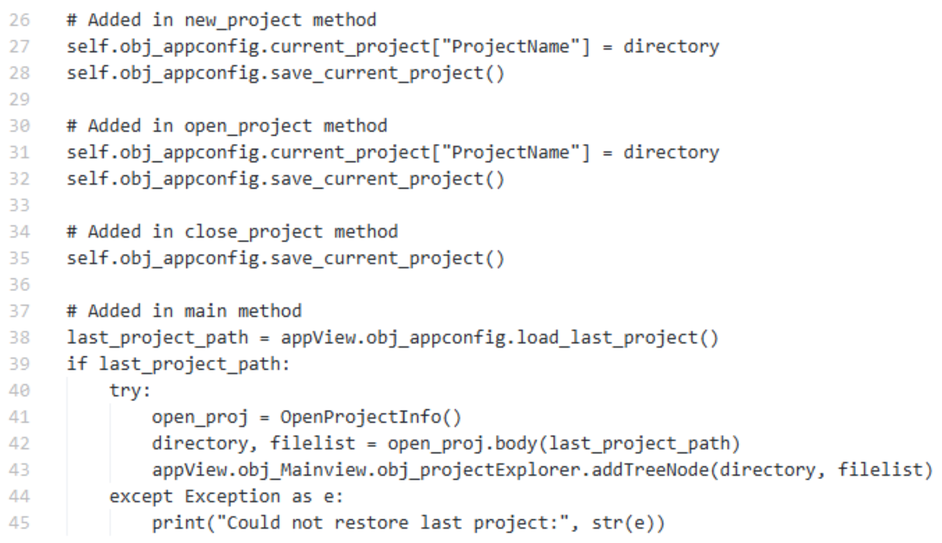
To improve user experience and maintain workflow continuity, functionality was added to automatically reopen the last accessed project folder when eSim is restarted. Previously, the project path was only stored in memory during runtime. Once the application was closed, this information was lost, requiring users to manually locate and open their project again upon each launch.

To address this, a lightweight persistence mechanism was implemented using a JSON file named last_project.json. The path of the last opened project is written to this file upon project load and read back when eSim starts. This ensures that the application can immediately reopen the same working directory without user intervention. The following code was implemented in src/configuration/Appconfig.py and src/frontEnd/Application.py.



```python
def save_current_project(self):
    try:
        path = os.path.join(self.user_home, ".esim", "last_project.json")
        with open(path, "w") as f:
            json.dump(self.current_project, f)
    except Exception as e:
        print("Failed to save current project:", str(e))

def load_last_project(self):
    try:
        path = os.path.join(self.user_home, ".esim", "last_project.json")
        with open(path, "r") as f:
            data = json.load(f)
            project_path = data.get("ProjectName", None)
            if project_path and os.path.exists(project_path):
                self.current_project["ProjectName"] = project_path
                return project_path
            else:
                print("Project path does not exist: ", project_path)
    except Exception as e:
        print("Error: ", str(e))
    return None
```

Figure 4.3: Code to save and retrieve last project path (Appconfig.py)



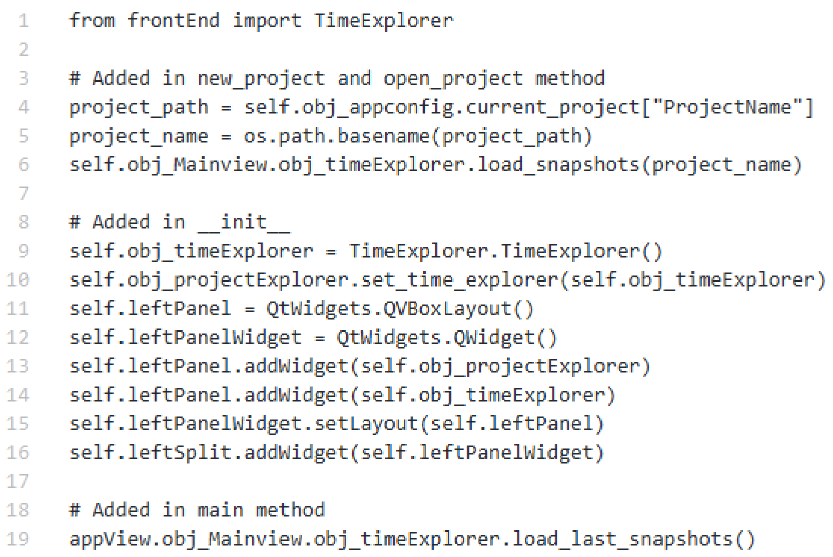
```python
# Added in new_project method
self.obj_appconfig.current_project["ProjectName"] = directory
self.obj_appconfig.save_current_project()

# Added in open_project method
self.obj_appconfig.current_project["ProjectName"] = directory
self.obj_appconfig.save_current_project()

# Added in close_project method
self.obj_appconfig.save_current_project()

# Added in main method
last_project_path = appView.obj_appconfig.load_last_project()
if last_project_path:
    try:
        open_proj = OpenProjectInfo()
        directory, filelist = open_proj.body(last_project_path)
        appView.obj_Mainview.obj_projectExplorer.addTreeNode(directory, filelist)
    except Exception as e:
        print("Could not restore last project:", str(e))
```

Figure 4.4: Code to auto-load last opened project on startup (Application.py)

4.4 Snapshot Creation and Timeline Integration

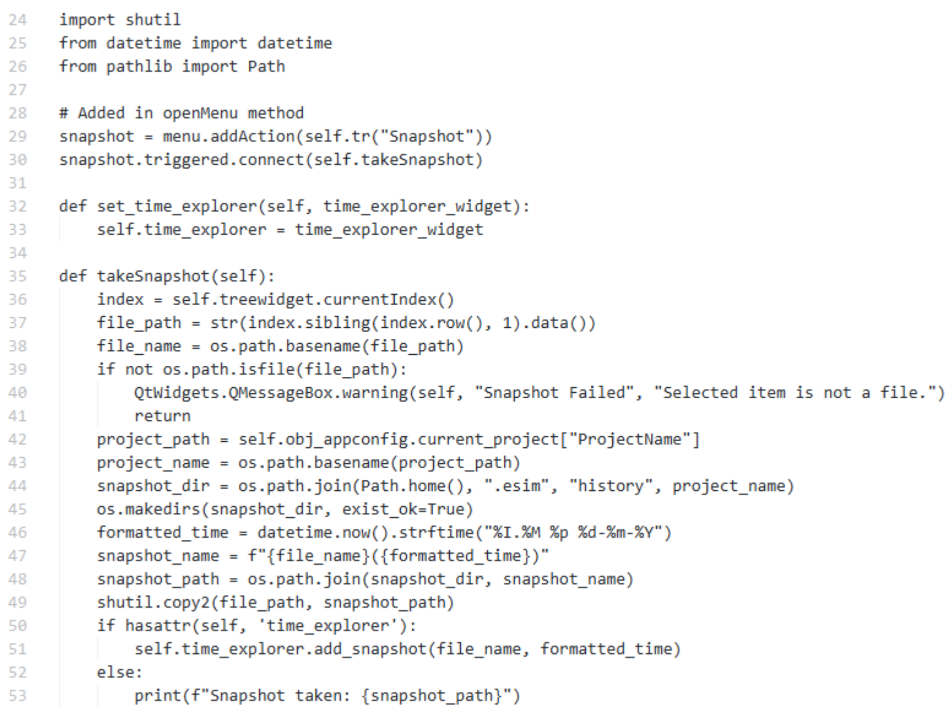
To provide a lightweight version control mechanism in eSim, a snapshot feature was introduced. This allows users to save the current state of any individual file at a specific moment, which can later be restored if needed. Users can right-click on any file in the Project Explorer and select the Snapshot option, which stores a copy of the file in a separate snapshot folder along with a timestamp. These saved snapshots are then displayed in a dedicated Timeline section under the Project panel.

To support this feature, two parts of the existing codebase were modified. Snapshot creation was triggered from the Project Explorer by connecting a context menu action to the snapshot saving logic. In addition, the main application file was updated to load the snapshot tracking system when a project is opened. The following code was implemented in src/frontEnd/Application.py and src/frontEnd/Project-Explorer.py.



```python
from frontEnd import TimeExplorer

# Added in new_project and open_project method
project_path = self.obj_appconfig.current_project["ProjectName"]
project_name = os.path.basename(project_path)
self.obj_Mainview.obj_timeExplorer.load_snapshots(project_name)

# Added in __init__
self.obj_timeExplorer = TimeExplorer.TimeExplorer()
self.obj_projectExplorer.set_time_explorer(self.obj_timeExplorer)
self.leftPanel = QtWidgets.QVBoxLayout()
self.leftPanelWidget = QtWidgets.QWidget()
self.leftPanel.addWidget(self.obj_projectExplorer)
self.leftPanel.addWidget(self.obj_timeExplorer)
self.leftPanelWidget.setLayout(self.leftPanel)
self.leftSplit.addWidget(self.leftPanelWidget)

# Added in main method
appView.obj_Mainview.obj_timeExplorer.load_last_snapshots()
```

Figure 4.5: Snapshots loader logic in Application.py



```python
import shutil
from datetime import datetime
from pathlib import Path

# Added in openMenu method
snapshot = menu.addAction(self.tr("Snapshot"))
snapshot.triggered.connect(self.takeSnapshot)

def set_time_explorer(self, time_explorer_widget):
    self.time_explorer = time_explorer_widget

def takeSnapshot(self):
    index = self.treewidget.currentIndex()
    file_path = str(index.sibling(index.row(), 1).data())
    file_name = os.path.basename(file_path)
    if not os.path.isfile(file_path):
        QtWidgets.QMessageBox.warning(self, "Snapshot Failed", "Selected item is not a file.")
        return
    project_path = self.obj_appconfig.current_project["ProjectName"]
    project_name = os.path.basename(project_path)
    snapshot_dir = os.path.join(Path.home(), ".esim", "history", project_name)
    os.makedirs(snapshot_dir, exist_ok=True)
    formatted_time = datetime.now().strftime("%I.%M %p %d-%m-%Y")
    snapshot_name = f"{file_name}({formatted_time})"
    snapshot_path = os.path.join(snapshot_dir, snapshot_name)
    shutil.copy2(file_path, snapshot_path)
    if hasattr(self, 'time_explorer'):
        self.time_explorer.add_snapshot(file_name, formatted_time)
    else:
        print(f"Snapshot taken: {snapshot_path}")
```

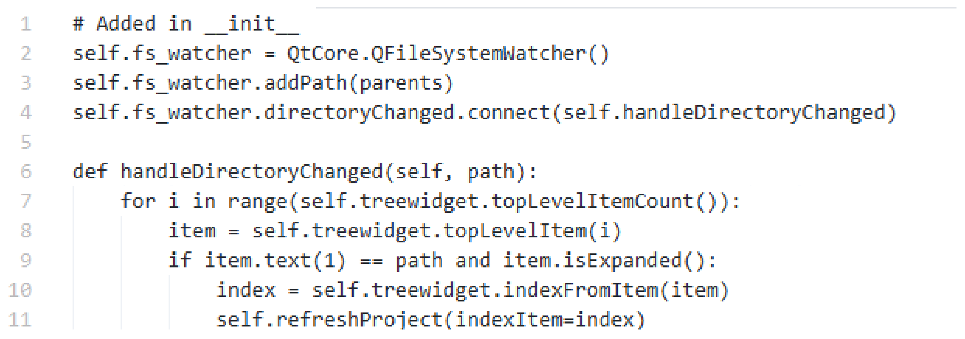
Figure 4.6: Snapshot trigger on right-click in ProjectExplorer.py

The core logic for managing, displaying, restoring, and clearing snapshots is encapsulated in a new custom file named src/frontEnd/TimeExplorer.py. This file creates a Timeline widget using PyQt that displays all snapshots in a list along with their timestamps. It provides Restore and Clear buttons. When a specific file is selected in the Timeline, the user can either restore it replacing the existing version in the project directory or delete it from the snapshot folder. If no item is selected, clicking the buttons allows bulk operations (restoring or deleting all snapshots). This file handles reading the last opened project from a configuration JSON, parsing and matching filenames, and ensuring robust error handling throughout the process.

4.5 Real-time File Synchronization in Project Explorer

Earlier, the Project Explorer in eSim did not automatically update when files were added, deleted, or modified from the operating system outside the application. This led to inconsistencies between the actual file system and what was shown in the GUI, particularly when a project folder remained expanded. Users were required to manually collapse and expand the folder to reflect the changes, which interrupted the workflow and introduced confusionespecially in large projects with frequent file changes.

To solve this, a real-time file synchronization mechanism was implemented. A file system watcher was integrated into the Project Explorer logic, which monitors the project directory. Whenever any change (e.g., file deletion, creation, or renaming) occurs outside the eSim interface, the Project Explorer immediately reflects the change in the UI even if the folder is currently expanded. This ensures seamless synchronization between the displayed files and the actual file system, improving responsiveness and usability. The following code was implemented in src/frontEnd/ProjectExplorer.py.



```python
# Added in __init__
self.fs_watcher = QtCore.QFileSystemWatcher()
self.fs_watcher.addPath(parents)
self.fs_watcher.directoryChanged.connect(self.handleDirectoryChanged)

def handleDirectoryChanged(self, path):
    for i in range(self.treewidget.topLevelItemCount()):
        item = self.treewidget.topLevelItem(i)
        if item.text(1) == path and item.isExpanded():
            index = self.treewidget.indexFromItem(item)
            self.refreshProject(indexItem=index)
```

Figure 4.7: Code for real-time file synchronization in ProjectExplorer.py

Chapter 5

Results and Testing

5.1 Visual Results After Fixes

This section showcases the visual outcomes after resolving each of the identified issues. For every feature or fix implemented, a corresponding screenshot is provided to demonstrate its effect and validate that the functionality is working as intended.

5.1.1 Resolved Crash from lint_off Removal

After implementing exception handling for invalid list removals, eSim now shows a proper warning message instead of crashing. This ensures smoother user interaction when managing entries in the lint_off.txt file.

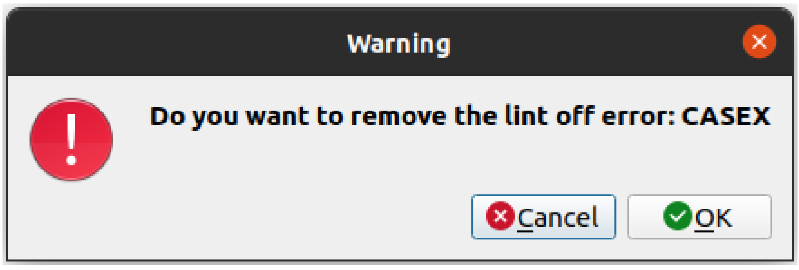
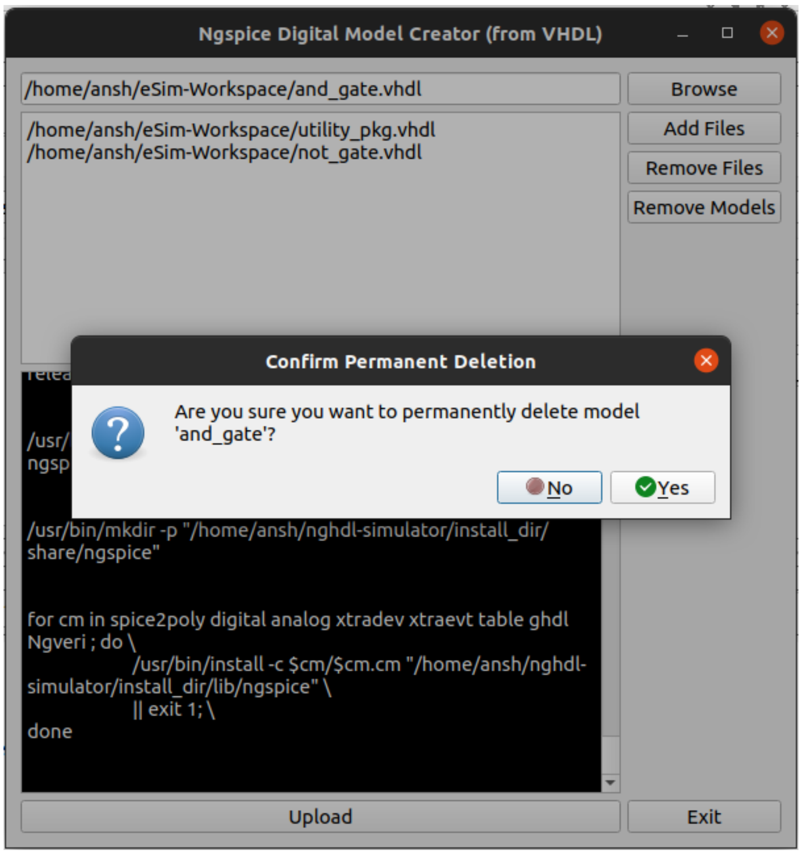

Do you want to remove the lint off error: CASEX

Figure 5.1: Warning message shown when trying to remove lint_off entry

5.1.2 NGHDL Model Deletion Feature

The NGHDL interface now includes a functional ”Remove Model” button that allows users to permanently delete uploaded models. The screenshot below shows the confirmation dialog that prevents accidental deletions.



Ngspice Digital Model Creator (from VHDL)
Browse
Add Files
Remove Files
Remove Models
Upload
Exit
Confirm Permanent Deletion
No
Yes

/home/ansh/eSim-Workspace/and_gate.vhdl

/home/ansh/eSim-Workspace/utility_pkg.vhdl
/home/ansh/eSim-Workspace/not_gate.vhdl

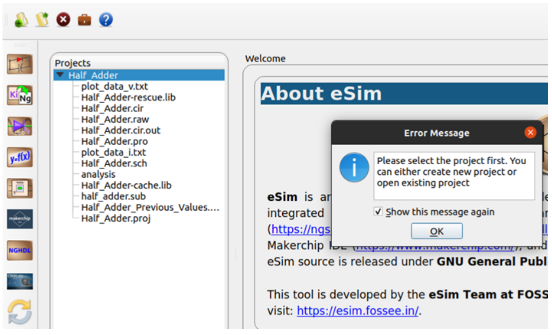
Are you sure you want to permanently delete model 'and_gate'?

```
/usr/bin/mkdir -p "/home/ansh/nghdl-simulator/install_dir/
share/ngspice"

for cm in spice2poly digital analog xtradev xtraevt table ghdl
Ngveri ; do \
            /usr/bin/install -c $cm/$cm.cm "/home/ansh/nghdl-
simulator/install_dir/lib/ngspice" \
            || exit 1; \
done
```

Figure 5.2: Confirmation dialog before deleting a model in NGHDL

5.1.3 Last Opened Project Auto-Restore

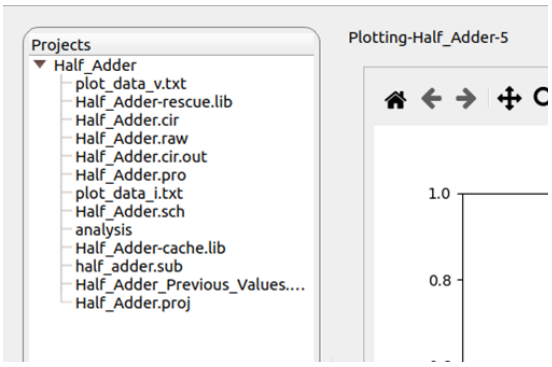
Previously, eSim did not retain the last accessed project after restarting. Users had to manually reselect the project folder, and none of the tools could be used until this step was completed. Now, the last opened project is automatically restored when eSim starts, allowing users to immediately begin working without manually selecting the project again.



Half_Adder
plot_data_v.txt
Half_Adder-rescue.lib
Half_Adder.cir
Half_Adder.raw
Half_Adder.cir.out
Half_Adder.pro
plot_data_i.txt
Half_Adder.sch
analysis
Half_Adder-cache.lib
half_adder.sub
Half_Adder_Previous_Values....
Half_Adder.proj

Please select the project first. You can either create new project or open existing project

(a) Previously: Tool unusable until project is selected



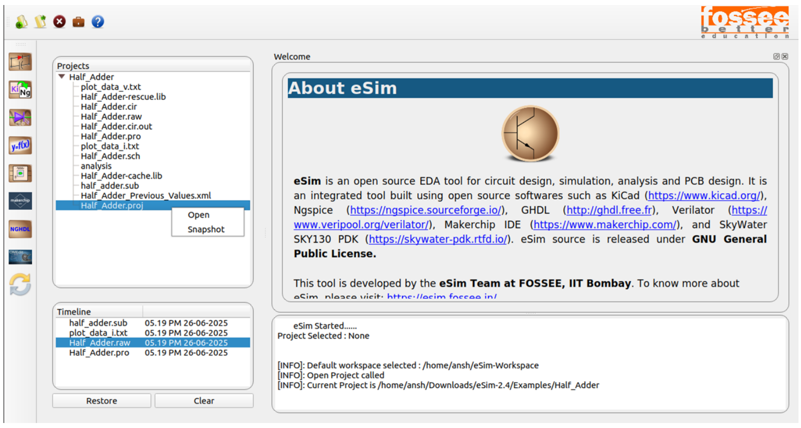
Half_Adder
plot_data_v.txt
Half_Adder-rescue.lib
Half_Adder.cir
Half_Adder.raw
Half_Adder.cir.out
Half_Adder.pro
plot_data_i.txt
Half_Adder.sch
analysis
Half_Adder-cache.lib
half_adder.sub
Half_Adder_Previous_Values....
Half_Adder.proj

(b) Now: Project auto-restored on startup

Figure 5.3: Before and after auto-restoration of last opened project in eSim

5.1.4 Snapshot and Timeline Functionality

The Project Explorer now includes a Snapshot option on right-click, and a Timeline section shows all saved snapshots with timestamps. Users can restore or delete snapshots easily from this interface.



```
Half_Adder
  plot_data_v.txt
  Half_Adder-rescue.lib
  Half_Adder.cir
  Half_Adder.raw
  Half_Adder.cir.out
  Half_Adder.pro
  plot_data_i.txt
  Half_Adder.sch
  analysis
  Half_Adder-cache.lib
  half_adder.sub
  Half_Adder_Previous_Values.xml
  Half_Adder.proj
```

```
half_adder.sub     05.19 PM 26-06-2025
plot_data_i.txt    05.19 PM 26-06-2025
Half_Adder.raw     05.19 PM 26-06-2025
Half_Adder.pro     05.19 PM 26-06-2025
```

eSim is an open source EDA tool for circuit design, simulation, analysis and PCB design. It is an integrated tool built using open source softwares such as KiCad (https://www.kicad.org/), Ngspice (https://ngspice.sourceforge.io/), GHDL (http://ghdl.free.fr), Verilator (https://www.veripool.org/verilator/), Makerchip IDE (https://www.makerchip.com/), and SkyWater SKY130 PDK (https://skywater-pdk.rtfd.io/). eSim source is released under **GNU General Public License.**

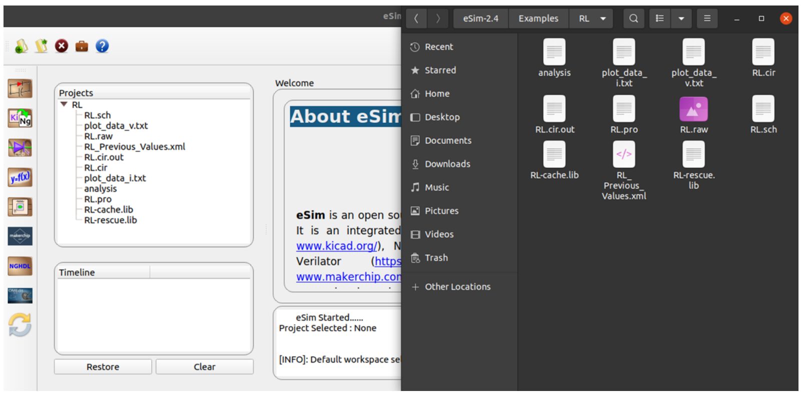
This tool is developed by the **eSim Team at FOSSEE, IIT Bombay**. To know more about eSim, please visit: https://esim.fossee.in/

```
    eSim Started......
Project Selected : None

[INFO]: Default workspace selected : /home/ansh/eSim-Workspace
[INFO]: Open Project called
[INFO]: Current Project is /home/ansh/Downloads/eSim-2.4/Examples/Half_Adder
```

Figure 5.4: Snapshot and Timeline integration in Project Explorer

5.1.5 Real-time File Synchronization

Any file that is deleted from the OS while the project folder is expanded in eSim now disappears instantly from the Project Explorer view, reflecting real-time synchronization.



```
RL
  RL.sch
  plot_data_v.txt
  RL.raw
  RL_Previous_Values.xml
  RL.cir.out
  RL.cir
  plot_data_i.txt
  analysis
  RL.pro
  RL-cache.lib
  RL-rescue.lib
```

eSim is an open so
It is an integrated
www.kicad.org/), N
Verilator (https
www.makerchip.con

```
    eSim Started......
Project Selected : None

[INFO]: Default workspace se
```

Figure 5.5: Project Explorer updating in real time after external file deletion

5.2 Stability and Functionality Improvements

After resolving each of the identified issues, systematic testing was conducted to ensure that the implemented features functioned correctly across typical and edge case scenarios. This section describes the verification process for each fix and highlights the resulting stability and reliability improvements in eSim.

The fix for the lint_off crash was verified by repeatedly adding and removing entries from the list, including attempting to remove non-existent items and emptying the entire list. It was confirmed that the system no longer crashes during these operations. Instead, a warning message is shown when an removal attempt is made. This behavior remained consistent across multiple test sessions, demonstrating that the issue was fully addressed and that simulation stability is now preserved.

The NGHDL model removal functionality was tested by uploading several dummy models, verifying their presence in the NGHDL model directory, and then removing them one by one using the new "Remove Model" button. Each deletion correctly removed all associated files, including entries in modpath.lst and symbol files. A confirmation dialog also prevented accidental deletions. The feature was retested after application restarts to ensure that removed models did not reappear, confirming that the removal logic works persistently and reliably.

To confirm that the last opened project restoration worked correctly, several projects were opened, closed, and reopened in different sequences. It was consistently observed that upon restarting eSim, the last accessed project was automatically restored without any user input. Additional tests verified that this feature works across operating systems and does not interfere with manual project switching after startup, ensuring robustness in varied usage scenarios.

The snapshot and timeline integration was extensively tested by creating snapshots for different files in a project and verifying that each snapshot appeared in the timeline with the correct timestamp. Snapshots were then individually restored and cleared to validate both targeted and bulk operations. Edge cases such as restoring non-existent or corrupted snapshot files were handled gracefully, and the UI remained responsive throughout the process. This confirmed that the snapshot system was stable under both normal and unexpected conditions.

Real-time file synchronization was tested by performing file operations (creation, deletion, renaming) directly from the operating system while the corresponding project folder was open and expanded in eSim. It was observed that the Project Explorer updated instantly to reflect these changes without requiring manual refresh. Long test sessions with multiple concurrent file changes confirmed that the synchronization logic remained accurate and consistent, demonstrating a substantial improvement in user experience and interface reliability.

5.3 User Experience Enhancements

The features and fixes implemented during the project significantly improved the overall user experience in eSim. By eliminating crashes, reducing repetitive manual actions, and making the interface more responsive to user behavior, the tool has become more intuitive and efficient. The auto-restoration of the last opened project ensures that users can immediately resume their work without the interruption of navigating through directories. Similarly, the integration of the Snapshot and Timeline feature offers users the flexibility to manage file versions effortlesslyallowing them to experiment with circuit modifications and revert to previous states when necessary.

Other improvements, such as real-time file synchronization and model deletion from the NGHDL interface, directly enhance the day to day usability of the platform. Users can now rely on eSim to accurately reflect changes in their file system without needing to refresh the interface manually. The addition of confirmation dialogs, error handling messages, and persistent state management contribute to a more polished and predictable environment. Overall, these enhancements reduce friction, save time, and make eSim more accessible to both new and experienced users.

Chapter 6

Conclusion and Future Work

6.1 Summary of Contributions

The internship project focused on identifying and resolving key usability and functional issues in the eSim platform. The contributions made during this period spanned across interface enhancements, backend improvements, persistent state handling, and usability features. Each implemented fix was aimed at improving the stability, efficiency, and user-friendliness of eSim, ensuring that users could engage with the tool in a smoother and more intuitive manner.

Among the major contributions was the implementation of error handling in the lint_off.txt file parsing logic, which previously caused simulation crashes. The NGHDL interface was improved with the addition of a model removal feature, streamlining the digital simulation workflow. The addition of a persistent mechanism to restore the last opened project further enhanced workflow continuity, eliminating the need for repetitive project reloading.

Another notable feature was the integration of a Snapshot and Timeline system within the Project Explorer. This allowed users to create and restore file level backups, offering a lightweight version control mechanism without needing external tools. Lastly, real-time file system synchronization was introduced in the Project Explorer, ensuring that any file changes made externally were instantly reflected in the UI. Together, these enhancements represent a significant step forward in the overall user experience and reliability of eSim.

These contributions not only fixed specific bugs but also added long-term value to the project by introducing scalable features that align with user needs. The work has laid a solid foundation for future enhancements and set a precedent for maintaining usability standards across the platform.

6.2 Key Challenges and Resolutions

During the course of development, several technical and architectural challenges were encountered. One of the early issues involved diagnosing the crash related to the lint_off.txt file. Identifying the point of failure within the simulation backend and ensuring that invalid entries were handled without disrupting the simulation flow required a thorough understanding of the codebase and careful use of exception handling mechanisms.

The implementation of the NGHDL model removal feature involved dealing with multiple file dependencies, including XML files, KiCad symbols, and directory structures. Ensuring that all references were properly cleaned up without affecting other models required precise coordination between frontend prompts and backend file handling logic. Exception handling and confirmation dialogs were added to ensure safe execution and improve user control.

Introducing real-time file synchronization was another challenge, as it required monitoring external file changes without adding performance overhead. The solution involved integrating file system watchers into the Project Explorer logic in a way that balanced responsiveness with system resource management. Ensuring that changes were reflected instantly, especially in expanded folders, required in-depth testing and debugging.

The Snapshot and Timeline feature posed both design and functional challenges. Designing an intuitive UI, managing timestamped file versions, and implementing batch operations like restore and delete all within a limited interface space required iterative development and testing. Additionally, managing file paths across different operating systems and ensuring proper formatting in the Timeline required detailed attention to cross-platform compatibility.

6.3 Recommendations for Future Enhancements

While the implemented features significantly improve eSims usability, several areas remain open for further enhancement. One recommendation is to extend the Snapshot system to support folder level snapshots or project wide state saving. This would give users the ability to track entire project versions rather than just individual files.

Another improvement could be the introduction of more advanced error reporting and logging mechanisms. Currently, error messages are limited to dialogs or terminal prints. A unified logging dashboard within the GUI could help users and developers trace issues more effectively, especially in larger circuits and complex simulation scenarios.

Enhancing the NGHDL interface to support model versioning and categorization could also be beneficial. As users increasingly work with multiple models, offering features like tagging, search, and metadata display could improve model management. Integrating visual indicators for unused or orphaned models would further streamline the workflow.

Lastly, accessibility improvements, such as keyboard shortcuts for common tasks, better tooltip descriptions, and customizable UI themes, could enhance the overall user experience especially for users working on smaller screens or with accessibility needs. These additions would make eSim more inclusive and adaptable to various user contexts.

6.4 Scope for Community Contribution

Given eSim's open-source nature, there is ample opportunity for continued community involvement. Contributors can help by identifying bugs, suggesting improvements, or directly submitting patches for review. Clear documentation, well-structured issues, and modular design make it easier for new contributors to onboard and contribute meaningfully to the project.

There is also scope for building tutorials, guides, and example projects to demonstrate the new featuresespecially the Snapshot and Timeline system. Community driven educational content can help new users better understand the tools capabilities and encourage adoption across institutions.

Moreover, contributors with expertise in simulation engines, UI/UX design, or digital circuit modeling can bring specialized value to the project. Areas like model optimization, simulation speed-up, and support for additional languages (like Verilog/SystemVerilog) are open for exploration and would benefit from collaboration with domain experts.

Encouraging student led development sprints, hackathons, and mentorship programs could strengthen the contributor base. By promoting active engagement through open forums, workshops, and issue triaging, eSim can continue to grow as a robust, community driven EDA tool that supports both education and research.

References

- FOSSEE Official Website, 2020. Available at:
  https://fossee.in/about
- eSim Official Website, 2020. Available at:
  https://esim.fossee.in/
- GitHub FOSSEE eSim Repository, 2020. Available at:
  https://github.com/FOSSEE/eSim
- GitHub FOSSEE NGHDL Repository, 2020. Available at:
  https://github.com/FOSSEE/nghdl
- Qt Documentation Qt for Python (PyQt5), 2020. Available at:
  https://doc.qt.io/qtforpython/
- Python Official Documentation, 2020. Available at:
  https://docs.python.org/3/
- KiCad EDA Tool Suite, 2020. Available at:
  https://kicad.org/
- GHDL: Open-source VHDL Simulator, 2020. Available at:
  https://ghdl.github.io/ghdl/