eSim Internship 2023 Report

On

kiCad

Submitted by

Partha Singha Roy
B.Tech (Electronics and Communication Engineering)
KGEC,Kalyani

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

June 27, 2023

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 and Rahul Panikar 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 in the field of electronic circuit simulation. 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 community's 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

FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools to improve the quality of education in our country. It aims to reduce dependency on proprietary software in educational institutions. It encourages the use of FLOSS tools through various activities to ensure commercial software is replaced by equivalent FLOSS tools. It also develops new FLOSS tools and upgrade existing tools to meet requirements in academia and research.[1]

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

KiCad, an open source electronic design automation (EDA) suite, is widely used for schematic capture and PCB layout. With the release of version 6, KiCad introduced numerous enhancements, including improved performance, a more intuitive user interface, and enhanced compatibility with modern operating systems. However, to leverage these new features, it was essential to update eSim, an open-source circuit simulation tool, to work seamlessly with the latest version of KiCad.[2]

My task is porting KiCad v4 to v6 in eSim software for Windows and Linux operating systems. eSim is a free/libre and open source EDA tool for circuit design, simulation, analysis, and PCB design developed by FOSSEE, IIT Bombay. It is an integrated tool built using open source software such as KiCad, Ngspice, NGHDL, and VHDL. eSim released under GPL. Because of this, it has the necessary packages and tools to integrate into it.

The process of porting KiCad v4 to v6 in eSim required meticulous attention to detail and a comprehensive understanding of both software packages. It involved analyzing the differences between the two versions, identifying the areas requiring changes, and implementing the necessary modifications to ensure compatibility.

Throughout this endeavor, I had the opportunity to explore the intricacies of the KiCad software, gaining a deeper understanding of its underlying architecture, libraries, and functionalities. I also familiarized myself with the eSim platform,

learning about its simulation capabilities and how it integrates with KiCad for efficient circuit design and analysis.

Chapter 2

Why KiCad v6 ?

Unfortunately, KiCad 4.0.7 is not available in Ubuntu 20 and above which is a dependency of **eSim**. Also, KiCad has stopped providing support for 4.0.7.

2.1 KiCad v6.0.11 :

The most stable release of KiCad v6.0.11, offers critical bug fixes and minor improvements compared to the deprecated KiCad 4. Some notable fixes and enhancements in v6.0.11 include:

1. Fixing ERC crash: Addressed an issue that caused the ERC (Electrical Rules Check) to crash during circuit validation.
2. Resolving crash when importing Eagle schematic: Fixed a crash that occurred when attempting to import a schematic from Eagle CAD software.
3. Fixing crash during language change with the open simulator window: Resolved a crash that would happen when changing the language while the simulator window was open.
4. Restoring proper file extension filters for graphic import: Corrected broken file extension filters for graphic imports in certain locale settings.
5. Ensuring project rescue with mismatched symbol name cases: Improved the project rescue feature to handle cases where symbol names have different capitalization.
6. Enabling real-time connectivity by default: Real-time connectivity, which shows immediate connections between components, is now enabled by default for enhanced design visibility.

These updates in KiCad 6.0.11 contribute to a more stable and reliable PCB design experience, addressing critical issues and enhancing overall functionality.

Chapter 3

PCB Design

KiCad is a popular open-source EDA tool for designing printed circuit boards (PCBs). It offers a user-friendly interface and a wide range of features for creating professional-grade PCB designs. With KiCad, you can create schematics, place components, route traces, and generate manufacturing files. It supports 3rd party footprints and offers various import/export options. KiCad also includes a powerful 3D viewer and offers extensive documentation and community support. Whether you're a hobbyist or a professional, KiCad provides a robust platform for PCB design with its intuitive workflow and comprehensive toolset.

3.1 STM32 Board Design

The STM32 Board is designed using KiCad v6.0.11.[10]

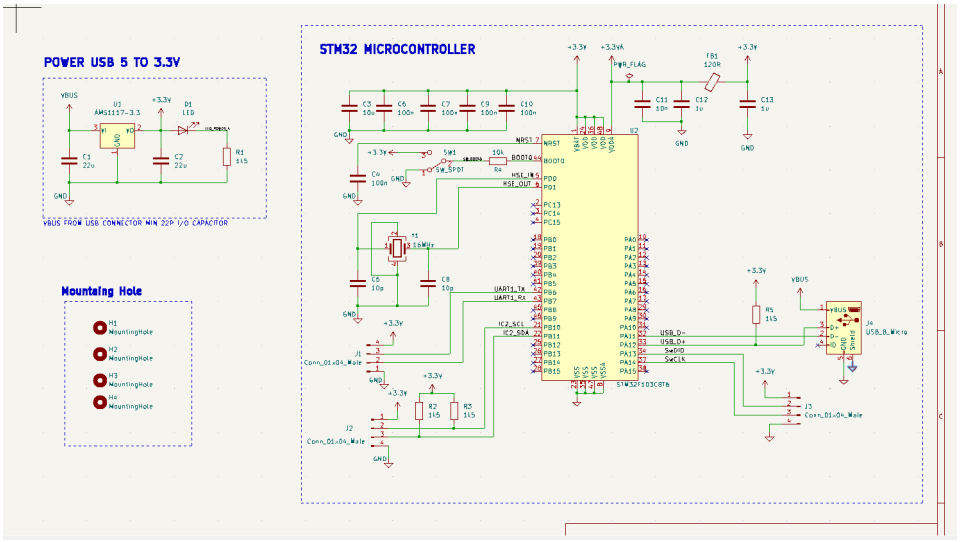

Figure 3.1: Circuit Diagram

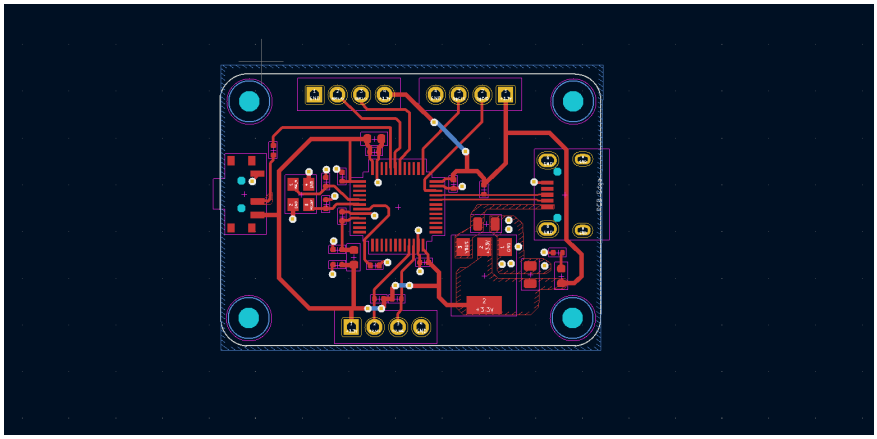

Figure 3.2: PCB layout

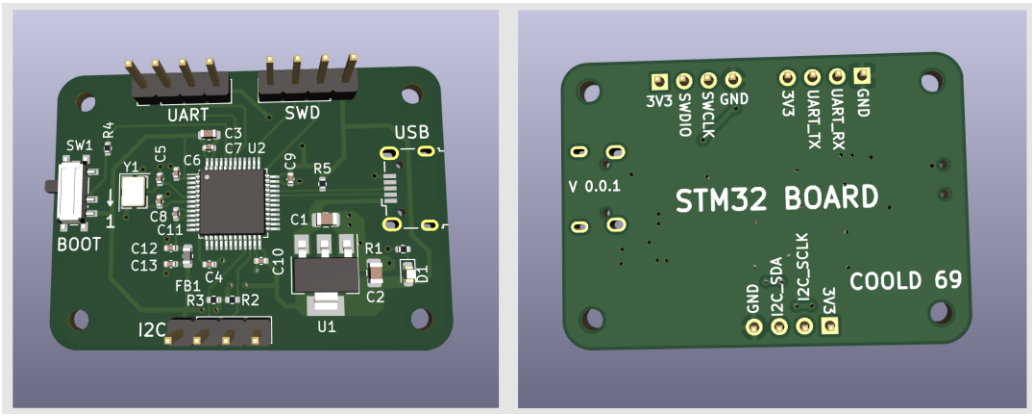

Figure 3.3: 3D View

Chapter 4

KiCad integrated Ngspice Simulation

KiCad has integrated the Ngspice circuit simulator, allowing users to perform accurate electrical simulations of their PCB designs. Here's a step-by-step guide to using KiCad's Ngspice simulation:

1. **Design your circuit:** Create the schematic of your circuit using KiCad's schematic editor, placing components and connecting the nets using wires.
2. **Assign spice models:** Assign Ngspice-compatible spice models to your components. These models define the electrical behavior of the components in the simulation.
3. **Configure simulation settings:** Define simulation parameters such as simulation type (DC, AC, transient, etc.), analysis type, simulation time, and desired outputs.
4. **Run the simulation:** Initiate the simulation in KiCad's integrated Ngspice simulator. Ngspice will analyze your circuit and generate simulation results based on the defined parameters.
5. **Analyze simulation results:** Once the simulation is complete, analyze the generated results, which may include voltage/current waveforms, frequency response, or other relevant data.

4.1 Example

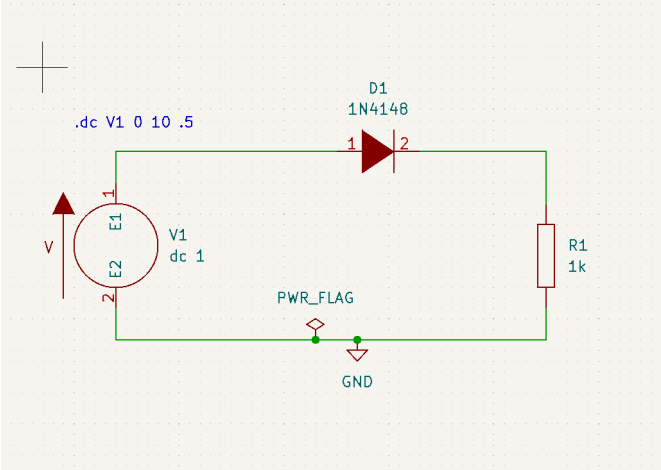

Figure 4.1: Circuit Diagram

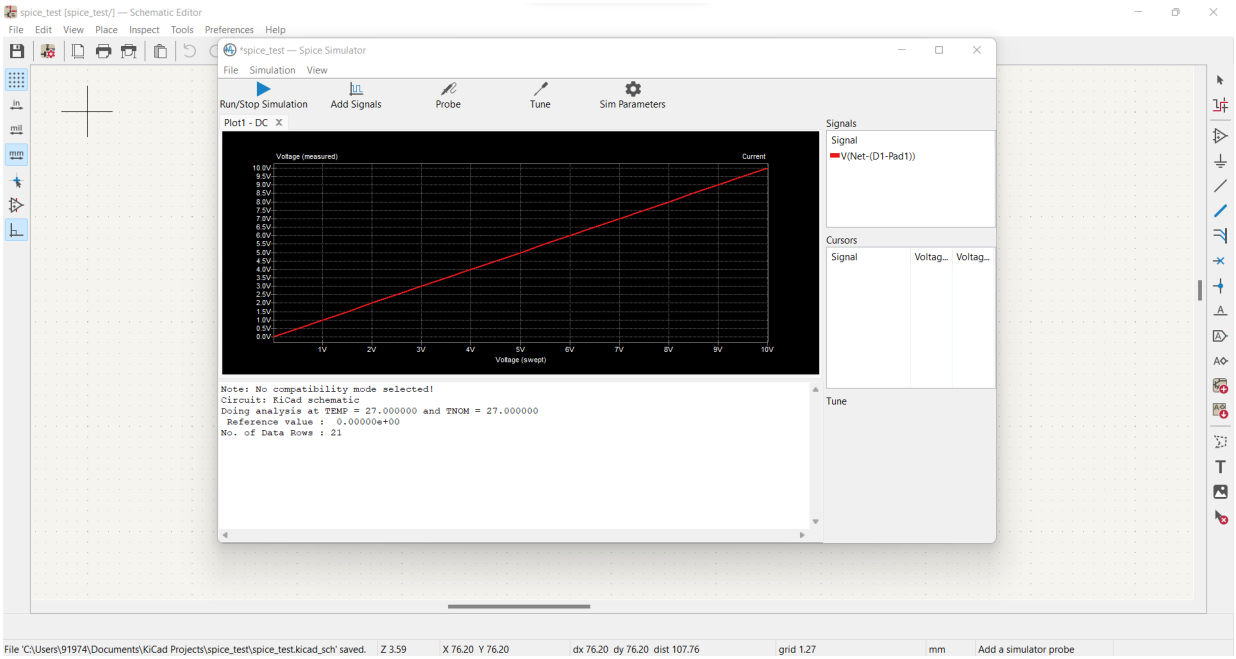

Figure 4.2: Circuit Simulation

Chapter 5

Differences between KiCad V4 and V6

Modern versions of KiCad can open files created in earlier versions, but can only write files in the latest formats. This means that in general, there are no special steps to migrate files from a previous version besides opening the files. In some cases, the file extension for a file has changed from one KiCad version to the next. After opening these files, they will be saved in the new format with the new file extension. The old files will not be deleted automatically.

In general, files created or modified by newer versions of KiCad cannot be opened by older versions of KiCad. For this reason, it is important to keep backup copies of your projects when testing a new KiCad release, until you are confident that you will not need to use the older KiCad version anymore.

5.1 New File Format

KiCad v6 is introducing a new format for the schematic files, which uses the S-Expressions notation. S-Expressions can represent data in a tree structure format as a nested list. The new file format makes schematic and layout files in human-readable form.

S-Expression notation is not new to KiCad. Layout (".kicad_pcb") and footprint files (".kicad_mod") have used SExpression notation in KiCad 4. But with KiCad 6, the transition is complete.

Here, The side by side comparison of the same schematic file section

KiCad 4 Eeschema file segment	KiCad 6 Eeschema file segment
Comp L power:GND PWR07 U 1 1 5EF25186 P 7150 2850 F 0 "PWR07" H 7150 2600 50 0001 C CNN F 1 "GND" H 7155 2677 50 0000 C CNN F 2 "" H 7150 2850 50 0001 C CNN F 3 "" H 7150 2850 50 0001 C CNN 1 7150 2850 1 0 0 -1 EndComp	(symbol(lib_id "power:GND")(at 181 72 0) (unit 1)(in_bom yes) (on_board yes) (uuid "186") (property "Reference" "PWR07" (id 0) (at 181.61 78.74 0) (effects (font (size 1.27 1.27)) hide)) (property "Value" "GND" (id 1) (at 181.737 76.7842 0)) (property "Footprint" "" (id 2) (at 181 72 0) (effects (font (size 1.27 1.27)) hide)) (effects (font (size 1.27 1.27)) hide)))

5.2 File Extension

KiCad v6 comes with new file extension. Here the comparism

KiCad 4 file extension	KiCad 6 file extension	Remarks
.sch	.kicad_sch	Kicad schematic file
.lib	.kicad_sym	KiCad symbol file
.pro	.kicad_pro	Kicad project file

5.3 Folders

KiCad v6 comes with new file extension. Here the comparism

KiCad 4 Folders	KiCad 6 Folders
models	footprints
library	symbols

These are some significant KiCad comparisons for porting KiCad 4 to 6. Additionally, it's worth noting that KiCad 4 uses "mm" for measurements, while KiCad 6 uses "mil".

Chapter 6

Conversion KiCad 4 to KiCad 6

6.1 .sch to .kicad_sch

With the introduction of KiCad v6.0, there is a change in the file extension for schematic files. The older ".sch" file extension has been replaced with the ".kicad_sch" extension.

To convert older KiCad schematic files in the ".sch" format to the new ".kicad_sch" format, you can follow these steps:

1. Open KiCad v6.0 or later.
2. Go to the "File" menu and select "Open."
3. Browse to the location of the older KiCad ".sch" schematic file.
4. Select the file and click "Open" to open it in KiCad.

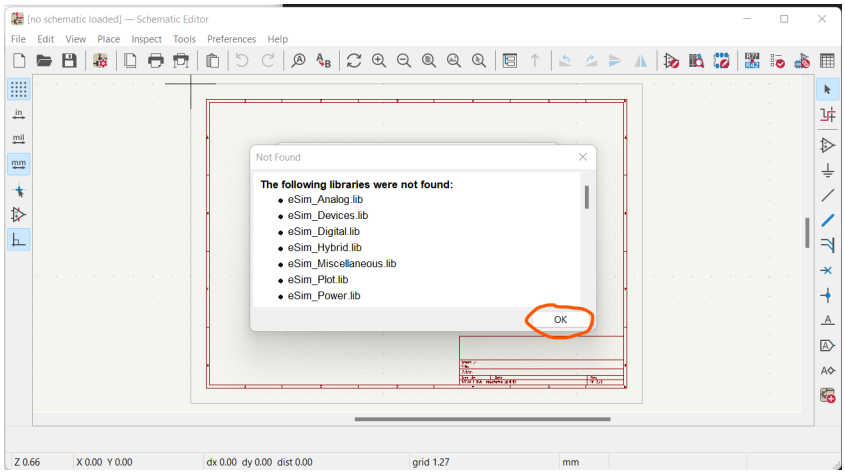

Figure 6.1: Opening .sch file, click OK

5. After clicking OK, now remap the symbols.

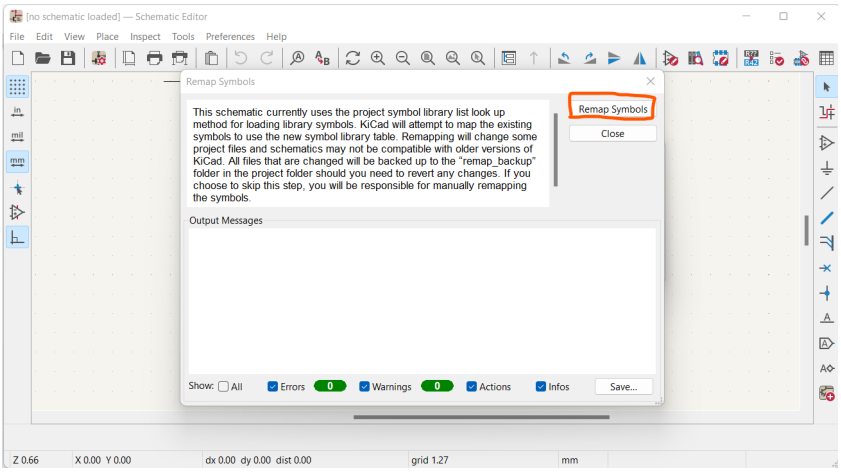

Figure 6.2: Remap all the symbols

6. Click OK.

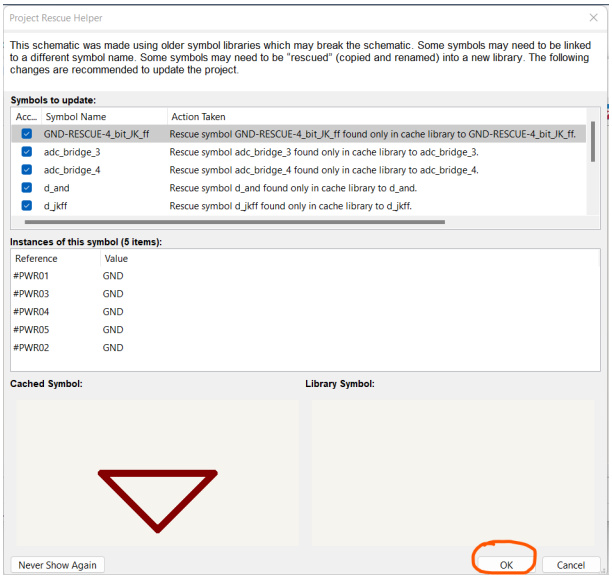

Figure 6.3: Click OK

7. Close the Tab

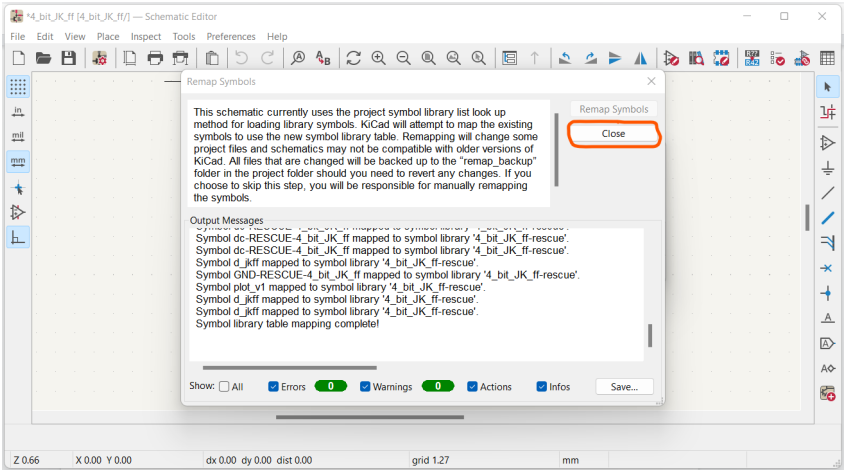

Figure 6.4: Close the tab

8. The schematic is now visible.

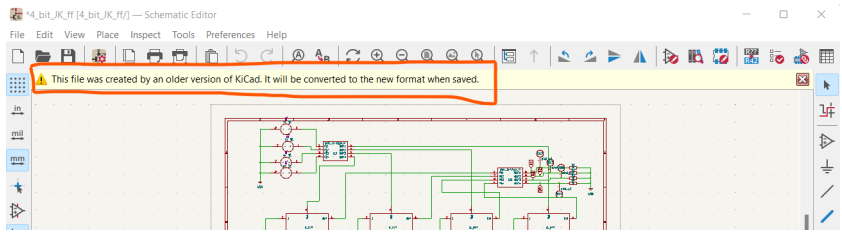

Figure 6.5: KiCad schematic

9. Press Ctrl+S to save the converted schematic file in the new ".kicad_sch" format.

By following these steps, you can convert your older KiCad ".sch" files to the new ".kicad_sch" format using KiCad version 6.0 or later. Also, in KiCad V6.0, the utilization of the cache.lib file is essential for remapping all symbols during the opening of a schematic (.sch) file. Without the cache.lib file, KiCad 6 would be unable to successfully open the schematic file and carry out the necessary symbol remapping process.

6.2 .lib to .kicad_sym

.lib is the older version of the KiCad schematic file extension. To convert your KiCad schematics to the new version, follow these steps:

1. Launch KiCad.
2. Go to "Tools" and select "Edit Schematic Symbols" from the menu.
3. Click on "Preferences" and choose "Manage Symbol Libraries" from the options.

4. In the "Global Libraries" tab, locate and click on the "Browse Libraries" button (represented by a small folder icon).

5. Browse your files and select the "*.lib" file that you want to convert. Then click "Open".

6. The library you imported will appear in the list. Click "OK" to confirm.

7. In the symbol library manager, you can now use the filter search field to locate the symbol you imported. Double-click on the symbol to open the file.

8. Once the symbol file is open, you can make any necessary modifications or updates to conform to the new KiCad version.

9. Save the updated symbol file with the ".kicad_sym" extension.

By using the KiCad symbol library management functionality, you can successfully convert and update your old libraries to the new format required by the current version of KiCad.

6.3 KiCad Display Setting

The presence of green lines in KiCad V6 is often associated with a known issue that is related to outdated Intel graphics drivers. It appears to be an internal bug within KiCad V6.[8][9]

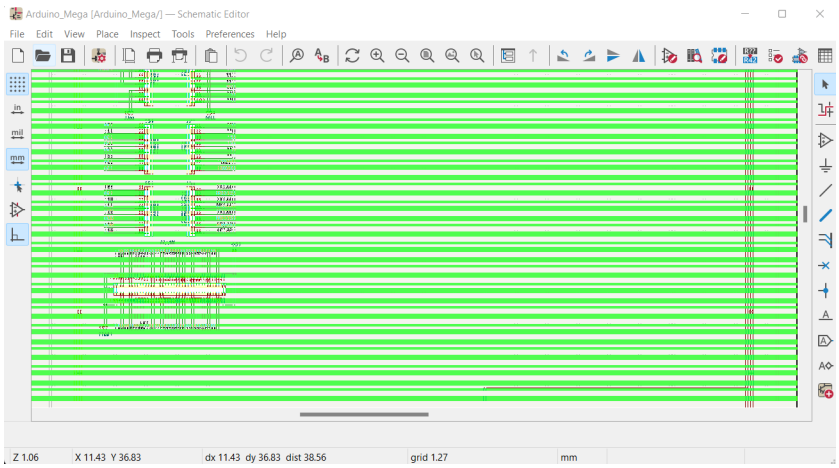

Figure 6.6: Green Line Display in KiCad 6

Recommended setting: Fallback graphics

Solution:

- I recommend downloading and installing the latest graphics driver for your system to address the issue. Upgrading to the most recent version of the graphics driver can often resolve compatibility problems and help mitigate the occurrence of the green lines in KiCad V6.

- To avoid encountering the green line problem in KiCad V6, you can follow these steps:

  1. Open the KiCad application.
  2. Go to the "Preferences" menu.
  3. Select "Display Settings" from the options.
  4. Look for the setting called "Fallback Graphics" for All Windows.
  5. enable or check this option.

  By enabling the "Fallback Graphics" for All Windows option, you can potentially prevent the occurrence of the green line issue in KiCad V6.

Accelerated graphics antialiasing: KiCad can use different methods to prevent aliasing (jagged lines) when rendering using a graphics card. Different methods may look better on different hardware, so you may want to experiment to find the one that looks best to you.[11]

Fallback graphics antialiasing: KiCad can also apply antialiasing when using the fallback graphics mode. Enabling this feature may result in poor performance on some hardware.[11]

6.4 Netlist Generation

In KiCad v4, the netlist generation option locates in the upper toolbar. You can find it as a separate button or as part of the toolbar options. It is represented by an icon that resembles a sheet of paper with a connection symbol. By clicking on this button or selecting the appropriate option from the upper toolbar, you can generate the netlist in KiCad v4.

In KiCad v6, the netlist generation option has been moved to a different location. Instead of the upper toolbar, you can now find the netlist generation option under the "File" menu. Here are the steps to access it:

1. Open KiCad version 6.
2. Go to the "File" menu at the top-left corner of the KiCad window.
3. Click on "Export" in the dropdown menu.
4. In the submenu that appears, select "Netlist".

By following these steps, you will be able to access the netlist generation option in KiCad v6.

6.5 The "??" mark Problem

In KiCad, if a symbol is no longer available or removed from the library, it can result in a "???" error when opening a schematic. A similar issue observes in the newer versions of KiCad, including version 6.

Instead of displaying "??" for missing symbols, KiCad provides a more informative message indicating that the symbols could not be found or loaded. This change helps users understand the issue more clearly.

To address this error and replace missing symbols with suitable alternatives, you can follow these steps in KiCad:

1. Open the schematic file (.sch) in KiCad.
2. Select the component with the missing symbol.
3. Press the "E" key on your keyboard or right-click on the component and choose "Edit Symbol" from the context menu.
4. In the symbol editor, try to find a similar component symbol that closely matches the missing one.
5. Assign the appropriate replacement symbol to the component by selecting it in the symbol editor.
6. Save the symbol changes and exit the symbol editor.
7. The component in the schematic should now display the newly assigned symbol without the "??" error.

By following these steps, you can address the issue of missing symbols in KiCad and ensure that your schematic functions as expected.

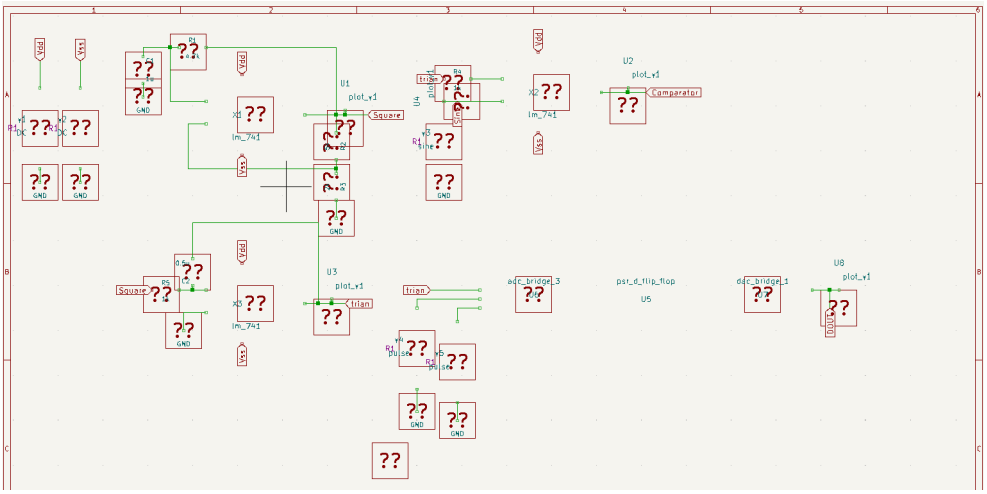

Figure 6.7: Question marks

Chapter 7

Coordinate Issue

The integrated KiCad 4 in eSim-2.3 has encountered a notable issue related to coordinate discrepancies of some symbols, presenting challenges in the seamless alignment between cursor positions and the actual positions of the symbols.

When attempting to place components, it becomes evident that the expected coordinates are not accurately reflected, there by hindering the precise positioning of symbols on the design canvas. This discrepancy issue persists throughout various symbol placements within eSim, with the "dvsd_8_bit_priority_encoder" component being among the notable examples affected this predicament.

Users have reported instances where the intended position of the cursor does not correspond to the actual placement of the symbol.

This coordinate problem has implications for the overall usability and efficiency of eSim, as it introduces additional complexity and potential errors in the design process. Addressing and rectifying these coordinate discrepancies would significantly enhance the user experience and streamline the design workflow within eSim, ensuring more accurate and consistent symbol placements throughout schematic creation and circuit layout endeavors.

7.1 Example

One of the components affected by this coordinate problem in eSim is the "dvsd_8_bit_priority_encoder". The red circle represents the actual symbol, while the blue circle represents the cursor position.

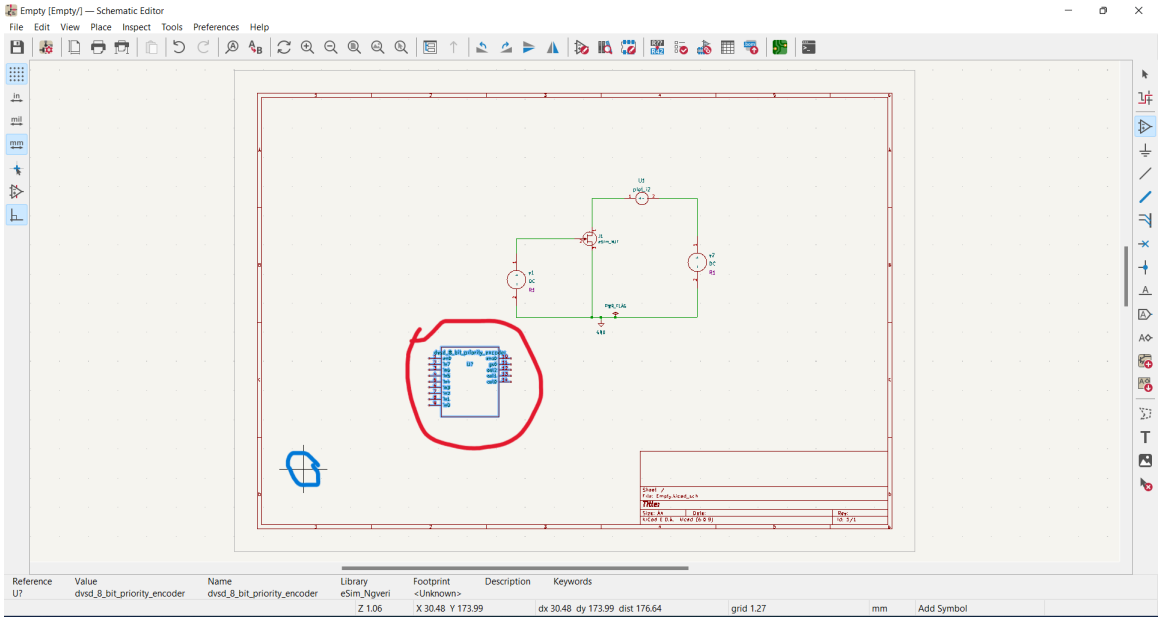

Figure 7.1: Coordinate Issue of "dvsd_8_bit_priority_encoder"

7.2 Reason

The coordinate issue is not a bug or problem specific to KiCad 4 or KiCad itself. Rather, it is an issue related to eSim, which occurs in two distinct scenarios.

- Firstly, The coordinate problem arises during the conversion of Verilog files to schematics within Makerchip. It seems that the schematic files generated through this process exhibit the issue.
- Secondly, the coordinate issue is also observed in the context of HGHDL.

In both cases, the coordinates of the components or symbols within the generated schematics or during the simulation may not align accurately with the expected cursor positions, leading to difficulties in precise placement and potential inconveniences in the design process.

7.2.1 KiCad Library Template

Drawing section:

```
kicad_lib_template = {
    "start_def": "DEF comp_name U 0 40 Y Y 1 F N",
    "U_field": "F0 \"U\" 2850 1800 60 H V C CNN",
    "comp_name_field": "F1 \"comp_name\" 2850 2000 60 H V C CNN",
    "blank_field": ["F2 blank_quotes 2850 1950 60 H V C CNN",
                    "F3 blank_quotes 2850 1950 60 H V C CNN"],
```

```
    "start_draw": "DRAW",
    "draw_pos": "S 2350 2100 3350 1800 0 1 0 N",
    "input_port": "X in 1 2150 2000 200 R 50 50 1 1 I",
    "output_port": "X out 2 3550 2000 200 L 50 50 1 1 O",
    "end_draw": "ENDDRAW",
    "end_def": "ENDDEF"
}
```

In the above template used for generating KiCad schematics using a Python script, the template itself is structured as a Python dictionary. Within this template, there is a specific section called "draw_pos" that holds values related to the positioning of elements within the schematic.

To modify the "draw_pos" value, you can directly access and update the corresponding key or keys . By adjusting the values within "draw_pos"; You can change the positioning of elements as desired.

Make sure to identify the specific key or keys within the "draw_pos" section that correspond to the elements you wish to reposition. Update the values associated with these keys according to the desired coordinates or positional adjustments.

By making these modifications to the "draw_pos" values in the template, you can effectively customize the positioning of elements within the generated KiCad schematic through the Python script.

According to the Eeschema documentation

S X1 Y1 X2 Y2 part dmg pen fill	Rectangle, from X1,Y1 to X2,Y2. Folders

To resolve the coordinate issue in the generated KiCad schematics, a Python script can be used to modify the values of X1, Y1, X2, and Y2. By targeting the affected components and accessing their "draw_pos" section, the script can adjust the coordinates to correct the misalignment. This solution provides an automated approach to address the coordinate issue, ensuring accurate placement of elements in the schematics.

Chapter 8

Porting KiCad 4 to KiCad 6

In the preceding chapters, I have discussed the process of porting KiCad 4 to KiCad 6. In this chapter, I will explore how eSim utilizes KiCad symbols for its Makerchip and NGHDL features.

eSim's Makerchip feature enables the conversion of Verilog/SystemVerilog files to ngspice-compatible format. As part of this conversion, eSim generates a corresponding KiCad symbol that represents the converted Verilog/SystemVerilog circuit. This KiCad symbol serves as a visual representation of the circuit within the schematic.

Similarly, eSim's NGHDL feature facilitates the conversion of VHDL files to spice-compatible format. During the conversion, eSim generates a KiCad symbol compared to the VHDL file. This KiCad symbol helps visualize the VHDL-based circuit in the schematic.

8.1 Makerchip

Inside the eSim repository, you'll find the 'src/maker' folder. This folder contains Python scripts responsible for converting Verilog files to ngspice-compatible format. These scripts enable seamless integration between Verilog designs and the ngspice simulation engine within eSim, facilitating accurate circuit simulation.

By utilizing these Python scripts, eSim provides users with a seamless pathway to convert Verilog designs into a format compatible with ngspice, allowing for accurate and efficient simulation of Verilog-based symbols.

To create new KiCad 6 symbols, the Kicad lib template needs to be modified to align with the updated Eeschema format, which now utilizes S-Expressions. The modified code may resemble the following structure:

```
kicad_sym_template = {
    "start_def": "(symbol \"comp_name\" (pin_names (offset 1.016)) (in_bom
    ↪  yes) (on_board yes)",
```

```
    "U_field":"(property \"Reference\" \"U\" (id 0) (at 12 15 0)(effects
     ↪  (font (size 1.524 1.524))))",
    "comp_name_field":"(property \"Value\" \"comp_name\" (id 1) (at 12 18
     ↪  0)(effects (font (size 1.524 1.524))))",
    "blank_field": [
                    "(property \"Footprint\" blank_quotes (id 2) (at 72.39
                     ↪  49.53 0)(effects (font (size 1.524 1.524))))",
                    "(property \"Datasheet\" blank_quotes (id 3) (at 72.39
                     ↪  49.53 0)(effects (font (size 1.524 1.524))))"
                    ],
    "draw_pos": "(symbol \"comp_name\"(rectangle (start 0 0 ) (end 25.40 3.6
     ↪  )(stroke (width 0) (type default) (color 0 0 0 0))(fill (type
     ↪  none))))",
    "start_draw":"(symbol",
    "input_port":"(pin input line(at -5.15 0.54 0 )(length 5.08 )(name \"in\"
     ↪  (effects(font(size 1.27 1.27))))(number \"1\" (effects (font (size
     ↪  1.27 1.27)))))",
    "output_port": "(pin output line(at 30.52 0.54 180 )(length 5.08 )(name
     ↪  \"out\" (effects(font(size 1.27 1.27))))(number \"2\" (effects (font
     ↪  (size 1.27 1.27)))))",
    "end_draw":"))"
}
```

By utilizing this modified Kicad lib template structure, the Python script dynamically generates accurate symbols based on the modified template, ensuring alignment with desired specifications for seamless integration into KiCad 6.

For more details about Python Script here.

8.2 NGHDL

Inside the NGHDL repository, you'll find the 'src' folder. This folder contains Python scripts responsible for converting VHDL files to ngspice-compatible format.

By utilizing these Python scripts, eSim provides users with a seamless pathway to convert VHDL designs into a format compatible with ngspice, allowing for accurate and efficient simulation of Verilog-based symbols.

To create new KiCad 6 symbols, the Kicad lib template needs to be modified to align with the updated Eeschema format, which now utilizes S-Expressions. The modified code may resemble the following structure:

```
    # KiCad V6 Symbol Template
kicad_sym_template = {
    "start_def": "(symbol \"comp_name\" (pin_names (offset 1.016)) (in_bom
     ↪  yes) (on_board yes)",
```

```
    "U_field":"(property \"Reference\" \"U\" (id 0) (at 6 7 0)(effects (font
     ↪  (size 1.524 1.524))))",
    "comp_name_field":"(property \"Value\" \"comp_name\" (id 1) (at 8 10
     ↪  0)(effects (font (size 1.524 1.524))))",
    "blank_field": [
                    "(property \"Footprint\" blank_quotes (id 2) (at 72.39
                     ↪  49.53 0)(effects (font (size 1.524 1.524))))",
                    "(property \"Datasheet\" blank_quotes (id 3) (at 72.39
                     ↪  49.53 0)(effects (font (size 1.524 1.524))))"
                    ],
    "draw_pos": "(symbol \"comp_name\"(rectangle (start 0 0 ) (end 15.25 2
     ↪  )(stroke (width 0) (type default) (color 0 0 0 0))(fill (type
     ↪  none))))",
    "start_draw":"(symbol",
    "input_port":"(pin input line(at -5.15 0.54 0 )(length 5.08 )(name \"in\"
     ↪  (effects(font(size 1.27 1.27))))(number \"1\" (effects (font (size
     ↪  1.27 1.27)))))",
    "output_port": "(pin output line(at 20.38 0.54 180 )(length 5.08 )(name
     ↪  \"out\" (effects(font(size 1.27 1.27))))(number \"2\" (effects (font
     ↪  (size 1.27 1.27)))))",
    "end_draw":"))"
}
```

By utilizing this modified Kicad lib template structure, the Python script dynamically generates accurate symbols based on the modified template, ensuring alignment with desired specifications for seamless integration into KiCad 6.

For more details about Python Script here.

8.3 Makerchip and NGHDL generated symbols

On the left side, we have the symbol generated by Makerchip, which represents the "test" component. This symbol is created through the Verilog/SystemVerilog to ngspice conversion process. It visually represents the Verilog/SystemVerilog circuit within the schematic.

On the right side, We have the symbol generated by NGHDL, which represents the "decade counter" component. This symbol generates during the VHDL to ngspice conversion process. It provides a visual representation of the VHDL-based circuit within the schematic.

Both symbols serve the purpose of visually representing the corresponding circuits within eSim. They enable seamless integration between the hardware description languages (Verilog/SystemVerilog and VHDL) and the ngspice simulation engine, facilitating accurate and efficient circuit simulation within the eSim environ-

ment.

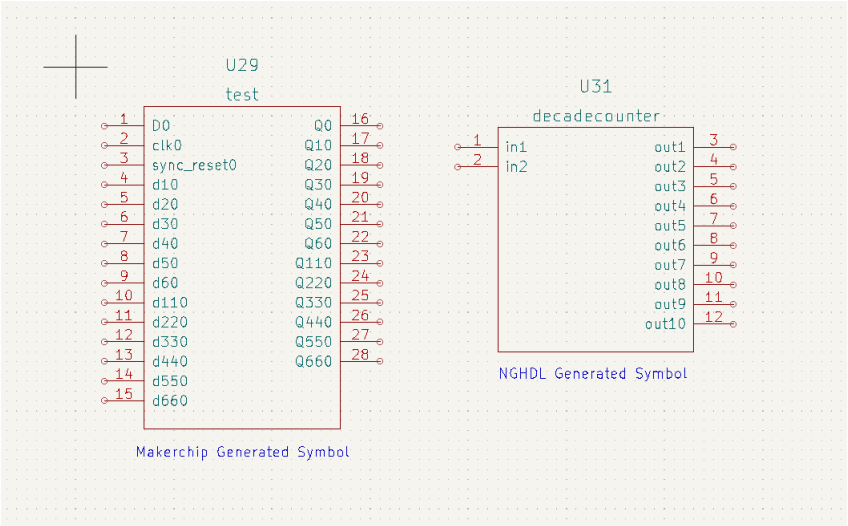

Figure 8.1: Example of makerchip and nghdl generated symbols

Also the newer symbols are more compact, the left one represents an older symbol generated by eSim, while the right one represents a newer version. The major difference between the two symbols is the increased compactness of the newer symbol.

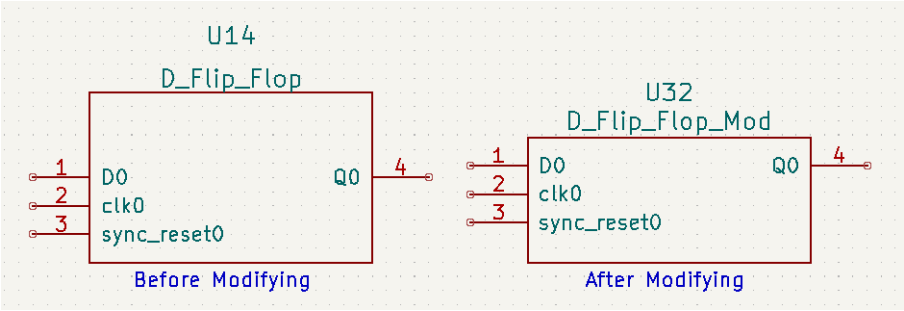

Figure 8.2: Before and After modified symbols

The newer symbol has undergone modifications to reduce its overall size and make it more space-efficient. This compactness allows for better utilization of the available area within the schematic, enabling a clearer representation of the component while minimizing any potential visual clutter.

By optimizing the symbol's design and layout, the newer version achieves a more streamlined appearance without sacrificing its functionality or readability. This enhancement not only enhances the visual aesthetics of the schematic but also improves the overall user experience when working with the symbol within the eSim environment.

Chapter 9

KiCad Installer

Creating a KiCad installer for Windows and Linux involves developing an installation package that simplifies the setup process.

For Windows, the installer can be an executable file that includes all the necessary components and libraries. On Linux, the installer can automate package management, handle dependencies, and ensuring a smooth installation experience.

9.1 Ubuntu 20.04

In Ubuntu, installing KiCad is a straightforward process. Open the Terminal and run the following commands:

The New KiCad 6 installation steps are:

```
# Adding KiCad-6 PPA to local apt-repository
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
 ↪  83FBAD2D910F124E
sudo add-apt-repository --yes "deb [trusted=yes]
 ↪  http://ppa.launchpad.net/kicad/kicad-6.0-releases/ubuntu focal main"
sudo touch /etc/apt/preferences.d/preferences
echo "Package: kicad" | sudo tee -a /etc/apt/preferences.d/preferences >
 ↪  /dev/null
echo "Pin: version 6.0.11*" | sudo tee -a /etc/apt/preferences.d/preferences >
 ↪  /dev/null
echo "Pin-Priority: 501" | sudo tee -a /etc/apt/preferences.d/preferences >
 ↪  /dev/null
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
 ↪  3B4FE6ACC0B21F32
sudo add-apt-repository --yes "deb http://archive.ubuntu.com/ubuntu/ focal main
 ↪  universe"

#Install KiCad V6.0.11
sudo apt-get install -y --no-install-recommends kicad
```

9.2 Windows

To build the KiCad executable for Windows, you have two options. The first option is to build it from the source code, which involves setting up the build environment and following the Windows-specific build instructions. The second option is using the steps provided by the eSim team. This version is designed to work with eSim and ensures compatibility and integration.

9.2.1 Build KiCad From its source

Building KiCad from its source code involves setting up the built environment on your system and following the instructions provided for the Windows platform. By building from source, you have more control over the customization and configuration options. This approach allows you to tailor KiCad to your specific needs and preferences, ensuring a personalized installation.

Unfortunately, I am unable to build KiCad from its source as I am facing difficulties installing or creating the KiCad installation environment.

9.2.2 KiCad Installer for eSim

The steps provided by the eSim Team for building KiCad from its source are generally straightforward. However, I have encountered an issue related to the "install.nsi" script, as there are differences between the file and folder structure of KiCad 4 and KiCad 6. These differences are extensively discussed in the "Differences between KiCad v4 and v6" section of the report. To successfully proceed with the installation, modifications need to be made to the "install.nsi" script to accommodate the specific file and folder structure of the desired version of KiCad.

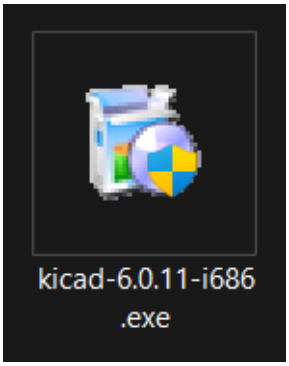

Figure 9.1: KiCad 6.0.11 Installer for windows

All the neccessary steps are in https://github.com/PSR0001/KiCad-eSim

Chapter 10

NGHDL Installer

NGHDL eSim is a powerful open-source electronic simulator integrated into the eSim software suite. It allows users to simulate and analyze digital electronic circuits using a hardware description language (HDL) approach.

10.1 NGHDL Executable

After completing all the necessary changes in the NGHDL code, the NGHDL executable is created using PyInstaller, as documented by the eSim team. PyInstaller is a popular tool for converting Python scripts into standalone executables. By utilizing PyInstaller, the NGHDL code can be packaged into a single executable file.

After following all the necessary steps

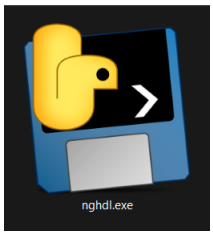

Figure 10.1: NGHDL executable

10.2 NGHDL Package

All the necessary components are added to the NGHDL package.

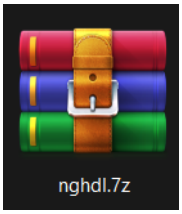

Figure 10.2: NGHDL ZIP

Chapter 11

eSim Installer

eSim Installer provides a convenient way to install eSim on Ubuntu and Windows operating systems. It simplifies the setup process by automating the installation and configuration steps. For Ubuntu, it ensures compatibility with the required dependencies and streamlines the installation. On Windows, it offers a user-friendly interface and takes care of the necessary dependencies. The eSim Installer significantly simplifies the installation process of eSim, making it more accessible for users on both platforms.

11.1 Ubuntu 20.04

Installing eSim with the new KiCad 6 version on Ubuntu using the eSim installer is a straightforward process. It involves modifying the "installKicad" and "copyKicadLibrary" functions in the "install-eSim.sh" bash script. By making these adjustments, the eSim installer can seamlessly handle the installation of KiCad 6 and copy the necessary KiCad libraries. These allow for a smooth and hassle-free installation experience of eSim with the latest KiCad version on Ubuntu.

InstallKicad function with KiCad 6

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

    #sudo add-apt-repository ppa:js-reynaud/ppa-kicad
    kicadppa="kicad/kicad-6.0-releases"
    findppa=$(grep -h -r "^deb.*$kicadppa*" /etc/apt/sources.list* > /dev/null
     ↪  2>&1 || test $? = 1)
    if [ -z "$findppa" ]; then
        echo "Adding KiCad-6 PPA to local apt-repository"
        if [[ $(lsb_release -rs) == 20.* ]]; then
            sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
 ↪  --recv-keys 83FBAD2D910F124E
```

```
            sudo add-apt-repository --yes "deb [trusted=yes]
             ↪  http://ppa.launchpad.net/kicad/kicad-6.0-releases/ubuntu focal
             ↪  main"
            sudo touch /etc/apt/preferences.d/preferences
            echo "Package: kicad" | sudo tee -a
             ↪  /etc/apt/preferences.d/preferences > /dev/null
            echo "Pin: version 6.0.11*" | sudo tee -a
             ↪  /etc/apt/preferences.d/preferences > /dev/null
            echo "Pin-Priority: 501" | sudo tee -a
             ↪  /etc/apt/preferences.d/preferences > /dev/null
            sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
             ↪  --recv-keys 3B4FE6ACC0B21F32
            sudo add-apt-repository --yes "deb http://archive.ubuntu.com/ubuntu/
             ↪  focal main universe"
        else
            sudo add-apt-repository ppa:kicad/kicad-6.0-releases
        fi
    else
        echo "KiCad-6 is available in synaptic"
    fi

    sudo apt-get install -y --no-install-recommends kicad
    if [[ $(lsb_release -rs) == 20.* ]]; then
        sudo add-apt-repository -ry "deb http://archive.ubuntu.com/ubuntu/ focal
↪   main universe"
    fi

}
```

CopyKicadLibrary function to copy all the KiCad 6 symbols and templates to usr/share/kicad folder.

```
function copyKicadLibrary
{
    if [ -d ~/.config/kicad/6.0 ];then
        echo "kicad folder already exists"
    else
        echo ".config/kicad does not exist"
        mkdir -p ~/.config/kicad/6.0
    fi
    # Dump KiCad config path
    echo "$HOME/.config/kicad" >
     ↪  $eSim_Home/library/supportFiles/kicad_config_path.txt
    #Copy fp-lib-table for switching modes
    cp -r library/supportFiles/fp-lib-table ~/.config/kicad/6.0
    cp -r library/supportFiles/fp-lib-table-online ~/.config/kicad/6.0
```

```
    echo "fp-lib-table copied in the directory"

    # copy sym-lib-table for eSim-custom symbols
    cp -r library/supportFiles/sym-lib-table ~/.config/kicad/6.0
    echo "sym-lib-table copied in the directory"

    #Extract custom KiCad Library
    tar -xJf library/kicadLibrary.tar.xz

    #Copy KiCad libraries
    echo "Copying KiCad libraries....................."

    sudo cp -r kicadLibrary/symbols /usr/share/kicad/
    sudo cp -r kicadLibrary/footprints /usr/share/kicad/
    sudo cp -r kicadLibrary/template/* /usr/share/kicad/template/

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

    # Full path of 'kicad.pro file'
    KICAD_PRO="/usr/share/kicad/template/kicad.kicad_pro"
    KICAD_ORIGINAL="/usr/share/kicad/template/kicad.pro.original"

    if [ -f "$KICAD_ORIGINAL" ];then
        echo "kicad.pro.original file found"
        sudo cp -rv kicadLibrary/template/kicad.kicad_pro ${KICAD_PRO}
    else
        echo "Making copy of original file"
        sudo cp -rv ${KICAD_PRO}{,.original}
        sudo cp -rv kicadLibrary/template/kicad.kicad_pro ${KICAD_PRO}
    fi

    set +e      # Temporary 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/
}
```

11.2 Windows

Creating an installer for eSim is a complex task due to its numerous dependencies. It involves gathering and bundling all the required dependencies and configuring the NSI script accordingly. The NSI script handles the installation process, ensuring that all dependencies are properly installed. This comprehensive approach ensures that users can easily set up eSim on their systems, with the installer taking care of the intricate process of managing dependencies and providing a streamlined installation experience.

11.2.1 eSim Executable

eSim executable is created using pyinstaller a python library for creating the executable for windows. All the codes related to KiCad 6 is already updated.

The step by step guidance are in https://github.com/PSR0001/eSim

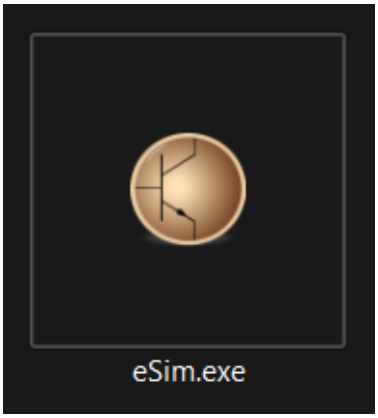

Figure 11.1: eSim Executable

11.2.2 Package eSim

Name	Date modified	Type	Size
eSim.7z	21-04-2023 11:32	WinRAR archive	87,047 KB
esim-setup-script-sky130.nsi	14-06-2023 16:36	NSI File	13 KB
ghdl.7z	05-10-2022 04:05	WinRAR archive	10,712 KB
kicad-6.0.11-i686.exe	05-04-2023 21:50	Application	1,24,913 KB
LICENSE.rtf	26-11-2022 20:20	Rich Text Format	80 KB
logo.ico	26-11-2022 20:20	Icon	98 KB
makerchip.7z	26-11-2022 20:20	WinRAR archive	6,345 KB
mingw64.7z	05-10-2022 04:05	WinRAR archive	62,109 KB
MSYS.7z	05-10-2022 04:05	WinRAR archive	16,560 KB
nghdl.7z	17-04-2023 17:04	WinRAR archive	15,253 KB
nghdl-setup-script.nsi	05-10-2022 04:05	NSI File	7 KB
nghdl-simulator.7z	05-10-2022 04:05	WinRAR archive	5,069 KB
sky130_fd_pr.7z	13-01-2023 13:02	WinRAR archive	48,524 KB
verilator.7z	05-10-2022 04:05	WinRAR archive	3,093 KB

Figure 11.2: Installer Folder

After executing the "esim-setup-script-sky130.nsi" file in NSIS (Nullsoft Scriptable Install System),

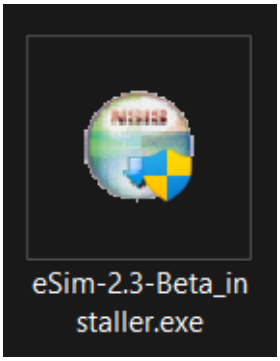

Figure 11.3: eSim Installer

Chapter 12

sym-lib-table

KiCad organizes symbols into symbol libraries, which hold collections of symbols. Each symbol in a schematic is uniquely identified by a full name that is composed of a library nickname and a symbol name.

12.1 Manage symbol library

KiCad uses a table of symbol libraries to map a symbol library nickname to an underlying symbol library on disk. Kicad uses a global symbol library table as well as a table specific to each project. To edit either symbol library table, use Preferences >Manage Symbol Libraries...

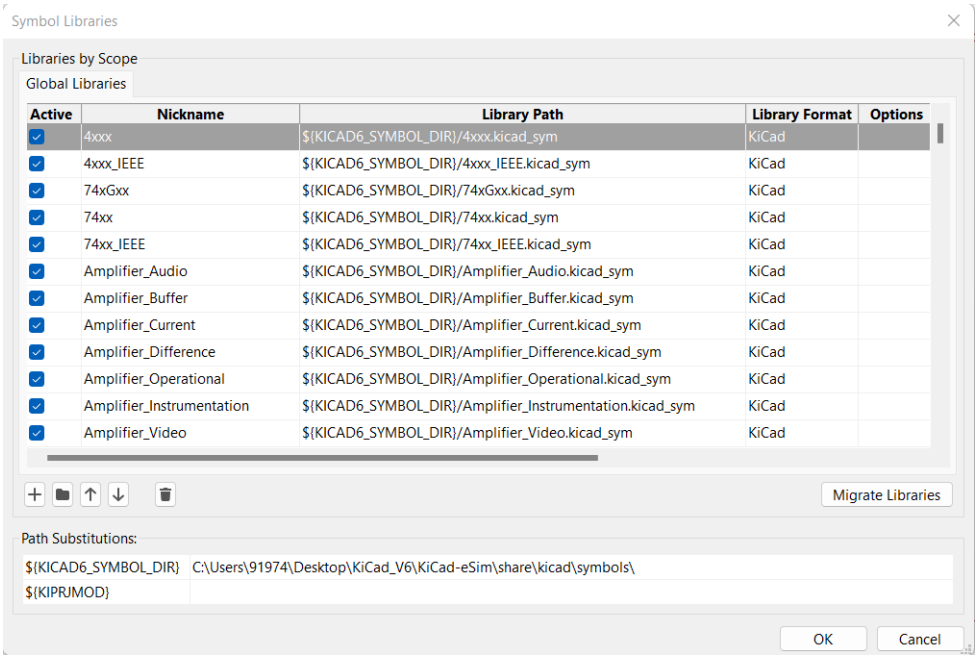

Figure 12.1: Manage symbols in KiCad 6

The global symbol library table contains the list of libraries that are always

available regardless of the currently loaded project. The table is saved in the file sym-lib-table in the KiCad configuration folder.

12.2 Configuration Folder

- **Windows**
  \AppData\Roaming\kicad\6.0
- **Ubuntu**
  ./config/kicad/6.0

12.3 Add eSim Custom Symbols to sym-lib-table

Please add the following code to the sym-lib-table:

```
(lib (name "eSim_Analog")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_Analog.kicad_sym")(options "")(
   descr ""))
(lib (name "eSim_Devices")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_Devices.kicad_sym")(options "")(
   descr ""))
(lib (name "eSim_Digital")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_Digital.kicad_sym")(options "")(
   descr ""))
(lib (name "eSim_Hybrid")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_Hybrid.kicad_sym")(options "")(
   descr ""))
(lib (name "eSim_Miscellaneous")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_Miscellaneous.kicad_sym")(
   options "")(descr ""))
(lib (name "eSim_Nghdl")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_Nghdl.kicad_sym")(options "")(
   descr ""))
(lib (name "eSim_Ngveri")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_Ngveri.kicad_sym")(options "")(
   descr ""))
(lib (name "eSim_Plot")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_Plot.kicad_sym")(options "")(
   descr ""))
(lib (name "eSim_SKY130")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_SKY130.kicad_sym")(options "")(
   descr ""))
(lib (name "eSim_SKY130_Subckts")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_SKY130_Subckts.kicad_sym")(
   options "")(descr ""))
(lib (name "eSim_Sources")(type "KiCad")(uri "${
   KICAD6_SYMBOL_DIR}/eSim_Sources.kicad_sym")(options "")(
   descr ""))
```

```
12  (lib (name "eSim_Subckt")(type "KiCad")(uri "${
       KICAD6_SYMBOL_DIR}/eSim_Subckt.kicad_sym")(options "")(
       descr ""))
13  (lib (name "eSim_User")(type "KiCad")(uri "${
       KICAD6_SYMBOL_DIR}/eSim_User.kicad_sym")(options "")(
       descr ""))
```

12.4 Edit sym-lib-table for eSim

Optimizing the sym-lib-table can significantly improve the execution efficiency by avoiding the unnecessary loading of all symbols. Since loading symbols consumes considerable time and CPU resources, it is advisable to select only the required symbols to include. By customizing the sym-lib-table to include specific symbols, you can streamline the process, reduce overhead, and enhance overall performance. This approach ensures a more efficient execution of the script, striking a balance between functionality and resource utilization. Ultimately, such optimization results in faster processing, improved user experience, and optimal utilization of computational power and time.

12.4.1 Example

Only custom eSim symbols are currently loaded here.

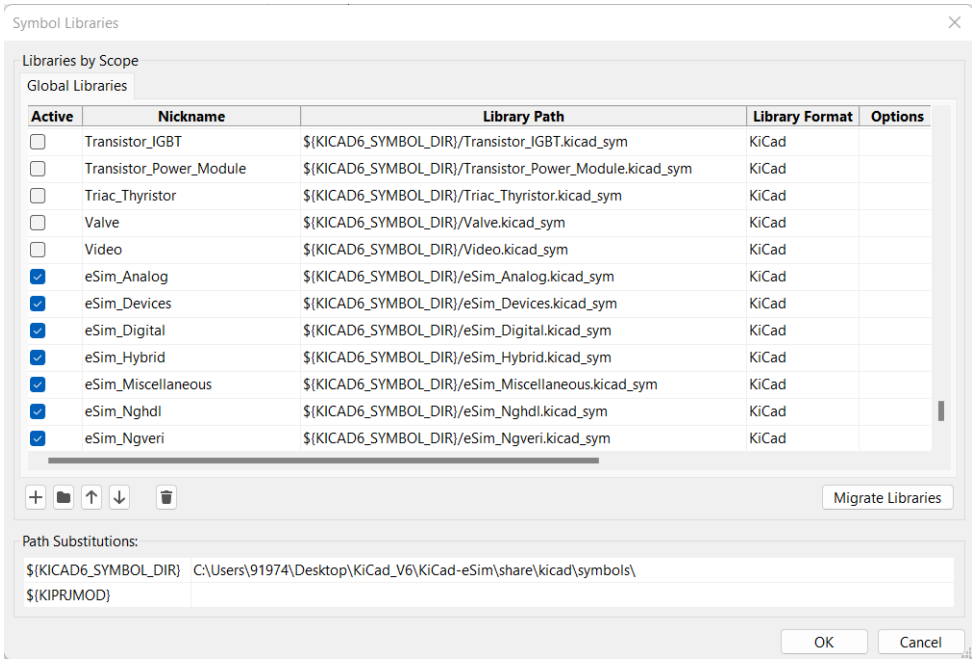

Figure 12.2: Custom sym-lib-table

12.5 Add sym-lib-table to eSim Installer

- Windows

```
    CopyFiles "$INSTDIR\eSim\library\supportFiles\
       sym-lib-table" "$PROFILE\AppData\Roaming\kicad
       \6.0\"
```

- Ubuntu

```
     # copy sym-lib-table for eSim-custom symbols
    cp -r library/supportFiles/sym-lib-table ~/.config/
       kicad/6.0
    echo "sym-lib-table copied in the directory"
```

Chapter 13

Bugs

13.1 Bug: eSim crash on schematic editor

An issue has been identified where eSim crashes when opening or editing a schematic without any project selection. This problem occurs when attempting to perform these actions without selecting a specific project in eSim.

Here the details `https://github.com/FOSSEE/eSim/issues/241`

13.2 Bug: Access the Verilog model's symbols

Users can access Verilog model symbols for KiCad even after deleting the model from Ngveri. It is expected that the model's symbol would no longer be accessible in KiCad after deleting it from Ngveri. However, inside the eSim/src/maker folder, there are no codes to delete that symbol.

Here the details `https://github.com/FOSSEE/eSim/issues/237`

13.3 Bug: Coordinate issue

The Coordinate issue is already detailed and explained in the dedicated chapter specifically addressing this problem.

13.4 Error: 0xc0000142

During the eSim installation, I encountered the error message "Application Unable to Start Correctly (Error: 0xc0000142)." This error was caused by the presence of a winAVR installation on my system.

13.4.1 Solution

Uninstalling winAVR, it is recommended to uninstall all C-program related compilers and utilities. This ensures a clean removal of any conflicting components that may be causing the error.

Here the details stackoverflow

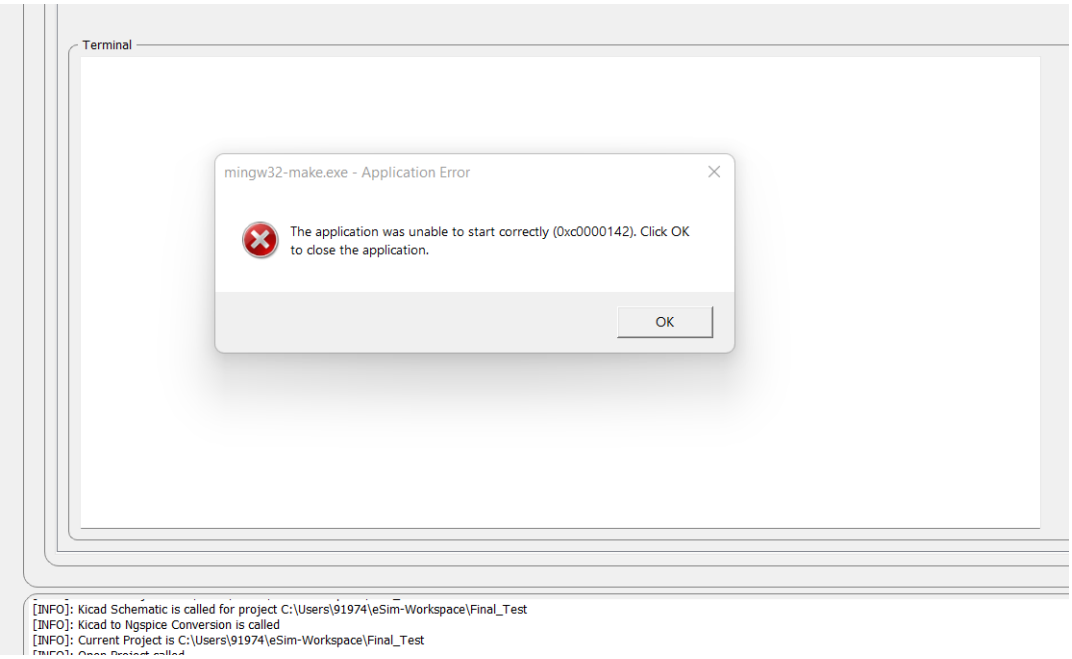

Figure 13.1: Error: 0xc0000142

Chapter 14

Conclusion and Future Scope

I successfully achieved the target of porting KiCad 4 to KiCad 6 and installed eSim with KiCad 6. I tested it with various circuits and encountered some bugs/issues during the testing period. One of the issues I faced was the Coordinate issue, which is clearly mentioned in this report.I successfully installed eSim in both Ubuntu and Windows, integrating it with KiCad 6. I found that the overall performance was good, and KiCad 6 was well synchronized with eSim. The integration between the two platforms worked seamlessly, allowing for a smooth and efficient circuit simulation experience.

The future work involves removing the integrated Ngspice simulation from eSim in KiCad 6 and improving the overall performance to make it more efficient. The objective is to separate the Ngspice integration from eSim while retaining its integration with KiCad 6. This step aims to optimize the performance and functionality of eSim, enhancing the circuit simulation experience.

Bibliography

[1] FOSSEE Official Website. 2023.
URL: https://fossee.in

[2] KiCad Offical Website. 2023.
URL: https://www.kicad.org

[3] Kicad Documentation.
URL: https://docs.kicad.org

[4] Kicad KLC(KiCad Library Convention).
URL: https://klc.kicad.org/

[5] Kicad Library Format.
URL: https://www.compuphase.com/electronics/LibraryFileFormats.pdf

[6] Kicad S-Expression.
URL: https://dev-docs.kicad.org/en/file-formats/sexpr-intro/

[7] Kicad File and Folders.
URL: https://docs.kicad.org/6.0/en/kicad/kicad.html#kicad_files_and_folders

[8] Green Display.
URL: https://www.reddit.com/r/KiCad/comments/ta44dq/green_lines_in_kicad_602/

[9] Horizantal Green Line.
URL: https://forum.kicad.info

[10] STM32 Board Desigm:
URL:

[11] Horizantal Green Line.
URL: