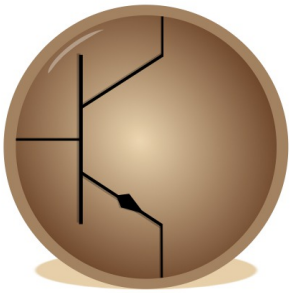
eSim

An open source EDA tool for circuit design, simulation, analysis and PCB design



eSim User Manual

Version 2.3

Prepared By:
eSim Team
FOSSEE, IIT Bombay

Indian Institute of Technology Bombay

September 2022

Contents

Acknowledgement

There have been many people contributing towards the software development and/or the electronic system design and simulation for eSim. The following people have contributed in some way.

Development:

- Fahim Khan
- Rahul Paknikar
- Saurabh Bansode
- Gloria Nandihal
- Sumanto Kar
- Athul George
- Gaurav Supal
- Kayva Manohar
- Komal Sheth
- Jay Mistry
- Manasi Yadav
- Digvijay Singh
- Inderjit Dhanjal
- Madhuri Kadam
- Nalinkumar S
- R.V.Rohinth Ram
- Charaan S
- Ashutosh Jha
- Shubhangi Mahajan
- Vadissa Yamini
- Bladen Martin
- Aamir Thekiya
- Neel Manilal Shah
- Padigepati Mallikarjuna Reddy
- Bhargav Katakam
- Anjali Jaiswal
- Mahfooz Ahmad
- Mudit Joshi
- Ashutosh Gangwar
- Akshay NH
- Athul MS
- Powai Labs Technology Private Limited
- Redwood EDA, LLC
- VLSI System Design Corporation Ltd.

Technical Guidance:

- Kannan Moudgalya
- Pramod Murali
- Madhav P. Desai
- Usha Vishwanathan
- Rupak Rokade
- Sunil Shetye

Financial Sponsorship:

- NMEICT, MoE, Govt. Of India
- MoE, Govt. Of India
- Indian Institute of Technology Bombay

If someone helped in the development/simulation and has not been inserted in this list, then this omission was unintentional. If you feel you should be on this list, then please feel free to contact us at contact-esim@fossee.in.

Chapter 1

Introduction

Electronic systems are an integral part of human life. They have simplified our lives to a great extent. Starting from small systems made of a few discrete components to the present day integrated circuits (ICs) with millions of logic gates, electronic systems have undergone a sea change. As a result, design of electronic systems too have become extremely difficult and time consuming. Thanks to a host of computer aided design tools, we have been able to come up with quick and efficient designs. These are called `Electronic Design Automation` or `EDA` tools.

Let us see the steps involved in EDA. In the first stage, the specifications of the system are laid out. These specifications are then converted to a design. The design could be in the form of a circuit schematic, logical description using an HDL language, etc. The design is then simulated and re-designed, if needed, to achieve the desired results. Once simulation achieves the specifications, the design is either converted to a PCB, a chip layout, or ported to an FPGA. The final product is again tested for specifications. The whole cycle is repeated until desired results are obtained

A person who builds an electronic system has to first design the circuit, produce a virtual representation of it through a schematic for easy comprehension, simulate it and finally convert it into a Printed Circuit Board (PCB). There are various tools available that will help us do this. Some of the popular EDA tools are those of `Cadence`, `Synopys`, `Mentor Graphics` and `Xilinx`. Although these are fairly comprehensive and high end, their licenses are expensive, being proprietary.

There are some free and open source EDA tools like `gEDA`, `KiCad` and `Ngspice`. The main drawback of these open source tools is that they are not comprehensive. Some of them are capable of PCB design (e.g. `KiCad`) while some of them are capable of performing simulations (e.g. `gEDA`). To the best of our knowledge, there is no open source software that can perform circuit design, simulation and layout design together. eSim is capable of doing all of the above.

eSim is a free and open source EDA tool. It is an acronym for **E**lectronics **Sim**ulation. eSim is created using open source software packages, such as KiCad, Ngspice and

Python. Using eSim, one can create circuit schematics, perform simulations and design PCB layouts. It can create or edit new device models, and create or edit subcircuits for simulation.

Because of these reasons, eSim is expected to be useful for students, teachers and other professionals who would want to study and/or design electronic systems. eSim is also useful for entrepreneurs and small scale enterprises who do not have the capability to invest in heavily priced proprietary tools.

This book introduces eSim to the reader and illustrates all the features of eSim with examples. The software architecture of eSim is presented in Chapter 2 while Chapter 3 gives the user step by step instructions to install eSim on a typical computer system. Chapter 4 gets the user started with eSim. It takes them through a tour of eSim with the help of a simple RC circuit example. Chapter 5 illustrates how to create the circuit schematic in esim and Chapter 6 explains simulating the circuit schematic. The advanced features of eSim such as Model Builder and Subcircuit Builder are covered in Chapter 7 and Chapter 8 respectively. Additional features in eSim like NGHDL, Makerchip-NgVeri and OpenModelica are covered in Chapter 9, Chapter 10 and Chapter 11 respectively. Chapter 12 illustrates how to use eSim for solving circuit simulation problems. The last chapter, Chapter 13 explains how eSim can be used to do PCB layout.

The following convention has been adopted throughout this manual.All the menu names, options under each menu item, tool names, certain points to be noted, etc., are given in *italics*. Some keywords, names of certain windows/dialog boxes, names of some files/projects/folders, messages displayed during an activity, names of websites, component references, etc., are given in `typewriter` font. Some key presses, e.g. `Enter` key, `F1` key, `y` for yes, etc., are also mentioned in `typewriter` font.

Chapter 2

Architecture of eSim

eSim is a CAD tool that helps electronic system designers to design, test and analyse their circuits. But the important feature of this tool is that it is open source and hence the user can modify the source as per his/her need. The software provides a generic, modular and extensible platform for experiment with electronic circuits. This software runs on Ubuntu Linux LTS distributions 18.04 and 20.04, and Microsoft Windows 7, 8 and 10. It uses `Python 3`, `KiCad 4.0.7`, `Makerchip`, `GHDL`, `Verilator` and `Ngspice`.

The objective behind the development of eSim is to provide an open source EDA solution for electronics and electrical engineers. The software should be capable of performing schematic creation, PCB design and circuit simulation (analog, digital and mixed-signal). It should provide facilities to create new models and components. The architecture of eSim has been designed by keeping these objectives in mind.

2.1 Modules used in eSim

Various open-source tools have been used for the underlying build-up of eSim. In this section we will give a brief idea about all the modules used in eSim.

2.1.1 Eeschema

Eeschema is an integrated software where all functions of circuit drawing, control, layout, library management and access to the PCB design software are carried out. It is the schematic editor tool used in KiCad. Eeschema is intended to work with PCB layout software such as Pcbnew. It provides netlist that describes the electrical connections of the PCB. Eeschema also integrates a component editor which allows the creation, editing and visualization of components. It also allows the user to effectively handle the symbol libraries i.e; import, export, addition and deletion of library components. Eeschema also integrates the following additional but essential functions needed for a

modern schematic capture software:

1. Design rules check (DRC) for the automatic control of incorrect connections and inputs of components left unconnected. 2. Generation of layout files in POSTSCRIPT or HPGL format. 3. Generation of layout files printable via printer. 4. Bill of materials generation. 5. Netlist generation for PCB layout or for simulation.

This module is indicated by the label 1 in Fig. 2.1.

As Eeschema is originally intended for PCB Design, there are no fictitious components[1] such as voltage or current sources. Thus, we have added a new library for different types of voltage and current sources such as sine, pulse and square wave. We have also built a library which gives printing and plotting solutions. This extension, developed by us for eSim, is indicated by the label 2 in Fig. 2.1.

2.1.2 CvPcb

CvPcb is a tool that allows the user to associate components in the schematic to component footprints when designing the printed circuit board. CvPcb is the footprint editor tool in KiCad. Typically the netlist file generated by Eeschema does not specify which printed circuit board footprint is associated with each component in the schematic. However, this is not always the case as component footprints can be associated during schematic capture by setting the component's footprint field. CvPcb provides a convenient method of associating footprints to components. It provides footprint list filtering, footprint viewing, and 3D component model viewing to help ensure that the correct footprint is associated with each component. Components can be assigned to their corresponding footprints manually or automatically by creating equivalence files. Equivalence files are look up tables associating each component with its footprint. This interactive approach is simpler and less error prone than directly associating footprints in the schematic editor. This is because CvPcb not only allows automatic association, but also allows to see the list of available footprints and displays them on the screen to ensure the correct footprint is being associated. This module is indicated by the label 3 in Fig. 2.1.

2.1.3 Pcbnew

Pcbnew is a powerful printed circuit board software tool. It is the layout editor tool used in KiCad. It is used in association with the schematic capture software Eeschema, which provides the netlist. Netlist describes the electrical connections of the circuit. CvPcb is used to assign each component, in the netlist produced by Eeschema, to a module that is used by Pcbnew. The features of Pcbnew are given below:

[1]Signal generator or power supply is not a single component but in circuit simulation, we consider them as a component. While working with actual circuit, signal generator or power supply gives input to the circuit externally thus, doesn't require for PCB design.

- It manages libraries of modules. Each module is a drawing of the physical component including its footprint - the layout of pads providing connections to the component. The required modules are automatically loaded during the reading of the netlist produced by CvPcb.
- Pcbnew integrates automatically and immediately any circuit modification by removal of any erroneous tracks, addition of new components, or by modifying any value (and under certain conditions any reference) of old or new modules, according to the electrical connections appearing in the schematic.
- This tool provides a rats nest display, a hairline connecting the pads of modules connected on the schematic. These connections move dynamically as track and module movements are made.
- It has an active Design Rules Check (`DRC`) which automatically indicates any error of track layout in real time.
- It automatically generates a copper plane, with or without thermal breaks on the pads.
- It has a simple but effective auto router to assist in the production of the circuit. An export/import in `SPECCTRA` dsn format allows to use more advanced autorouters.
- It provides options specifically for the production of ultra high frequency circuits (such as pads of trapezoidal and complex form, automatic layout of coils on the printed circuit).
- Pcbnew displays the elements (tracks, pads, texts, drawings and more) as actual size and according to personal preferences such as:
  - display in full or outline.
  - display the track/pad clearance.

This module is indicated by the label 4 in Fig. 2.1.

2.1.4 KiCad to Ngspice converter

Analysis parameters, and the source details are provided through this module. It also allows us to add and edit the device models and subcircuits, included in the circuit schematic. Finally, this module facilitates the conversion of KiCad netlist to Ngspice compatible ones.
It is developed by us for eSim and it is indicated by the label 7 in Fig. 2.1. The use of this module is explained in detail in section (yet to be put).

2.1.5 Model Builder

This tool provides the facility to define a new model for devices such as, 1. Diode 2. Bipolar Junction Transistor (BJT) 3. Metal Oxide Semiconductor Field Effect Transistor (MOSFET) 4. Junction Field Effect Transistor (JFET) 5. IGBT and 6. Magnetic core. This module also helps edit existing models. It is developed by us for eSim and it is indicated by the label 5 in Fig. 2.1.

2.1.6 Subcircuit Builder

This module allows the user to create a subcircuit for a component. Once the subcircuit for a component is created, the user can use it in other circuits. It has the facility to define new components such as, Op-amps, IC-555, UJT and so on. This component also helps edit existing subcircuits. This module is developed by us for eSim and it is indicated by the label 6 in Fig. 2.1.

2.1.7 Ngspice

Ngspice is a general purpose circuit simulation program for nonlinear dc, nonlinear transient, and linear ac analysis. Circuits may contain resistors, capacitors, inductors, mutual inductors, independent voltage and current sources, four types of dependent sources, lossless and lossy transmission lines (two separate implementations), switches, uniform distributed RC lines, and the five most common semiconductor devices: diodes, BJTs, JFETs, MESFETs, and MOSFET. This module is indicated by the label 9 in Fig. 2.1.

2.1.8 NGHDL

NGHDL, a module for mixed signal circuit simulation, is also integrated with eSim. It makes use of VHDL code. It uses ghdl for digital simulation and the mixed signal simulation happens through Ngspice.

2.1.9 NgVeri

NgVeri, a module for mixed signal circuit simulation, is also integrated with eSim. It makes use of Verilog/System Verilog/Transaction-Level Verilog code. It uses SandPiper SaaS and Verilator for digital simulation and the mixed signal simulation happens through Ngspice.

2.1.10 Makerchip-App

Makerchip is a cloud based browser application developed by Redwood EDA to do digital circuit design. One can simulate Verilog/SystemVerilog/Transaction-Level Verilog code

in Makerchip. eSim is interfaced with Makerchip using a Python based application called Makerchip-App which launches the Makerchip IDE.

2.1.11 SandPiper SaaS

Sandpiper-saas is a tool developed by Redwood EDA which converts Transaction Level Verilog code to SystemVerilog code. It is used by NgVeri so that it can get the System Verilog code which can be further passed to the Verilator.

2.1.12 Verilator

Verilator is a Verilog/SystemVerilog simulator tool. It converts the Verilog/SystemVerilog code to C++ object files. These object files are linked with that of Ngspice thus enabling mixed signal simulation in eSim.

2.1.13 OpenModelica

OpenModelica (OM) is an open source modeling and simulation tool based on Modelica language. Two modules of OpenModelica, OMEdit, an IDE for modeling and simulation and OMOptim, an IDE for optimisation are integrated with eSim.

2.2 Workflow of eSim

Fig. 2.1 shows the work flow in eSim. The block diagram consists of mainly three parts:

- Schematic Editor
- PCB Layout Editor
- Circuit Simulators

Here we explain the role of each block in designing electronic systems. Circuit design is the first step in the design of an electronic circuit. Generally a circuit diagram is drawn on a paper, and then entered into a computer using a schematic editor. Eeschema is the schematic editor for eSim. Thus all the functionalities of Eeschema are naturally available in eSim.

Libraries for components, explicitly or implicitly supported by Ngspice, have been created using the features of Eeschema. As Eeschema is originally intended for PCB design, there are no fictitious components such as voltage or current sources. Thus, a new library for different types of voltage and current sources such as sine, pulse and square wave, has been added in eSim. A library which gives the functionality of printing and plotting has also been created.

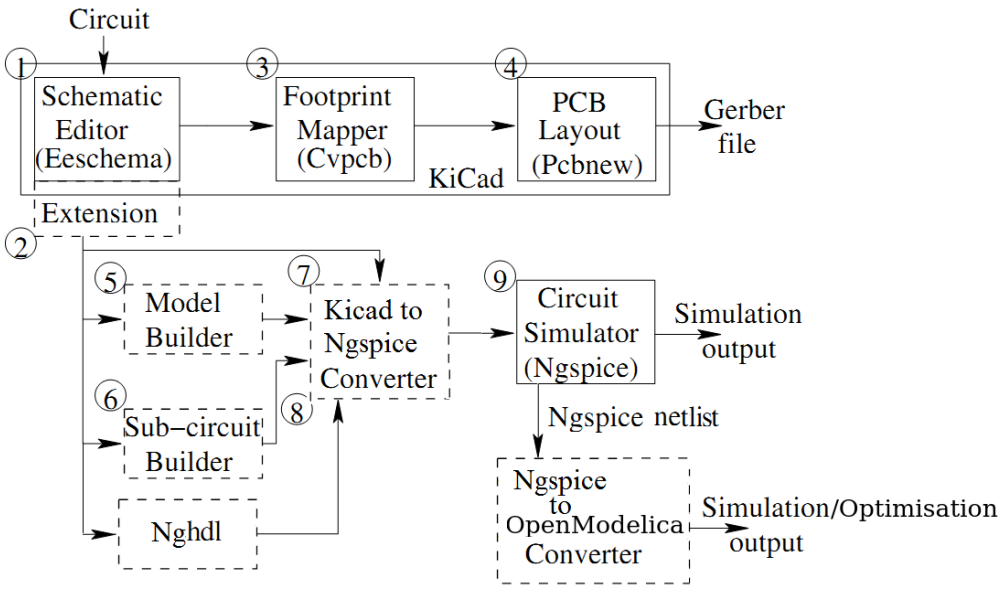


Figure 2.1: Work flow in eSim. (Boxes with dotted lines denote the modules developed in this work).

The schematic editor provides a netlist file, which describes the electrical connections of the design. In order to create a PCB layout, physical components are required to be mapped into their footprints. To perform component to footprint mapping, CvPcb is used. Footprints have been created for the components in the newly created libraries. Pcbnew is used to draw a PCB layout.

After designing a circuit, it is essential to check the integrity of the circuit design. In the case of large electronic circuits, breadboard testing is impractical. In such cases, electronic system designers rely heavily on simulation. The accuracy of the simulation results can be increased by accurate modeling of the circuit elements. Model Builder provides the facility to define a new model for devices and edit existing models. Complex circuit elements can be created by hierarchical modeling. Subcircuit Builder provides an easy way to create a subcircuit.

The netlist generated by Schematic Editor cannot be directly used for simulation due to compatibility issues. Netlist Converter converts it into Ngspice compatible format. The type of simulation to be performed and the corresponding options are provided through a graphical user interface (GUI). This is called KiCad to Ngspice Converter in eSim.

eSim uses Ngspice for analog, digital, mixed-level/mixed-signal circuit simulation. Ngspice is based on three open source software packages

- Spice3f5 (analog circuit simulator)
- Cider1b1 (couples Spice3f5 circuit simulator to DSIM device simulator)
- Xspice (code modeling support and simulation of digital components through an event driven algorithm)

It is a part of gEDA project. Ngspice is capable of simulating devices with BSIM, EKV, HICUM, HiSim, PSP, and PTM models. It is widely used due to its accuracy even for the latest technology devices.

Chapter 3

Installing eSim

3.1 eSim installation in Ubuntu OS

1. Download eSim installer from `http://esim.fossee.in/downloads` to a local directory and unpack it. You can also unpack the installer through the terminal. Open the terminal and navigate to the directory where the installer is located. Use the following command to unpack:

   ```
   $ unzip eSim-2.3.zip
   ```

2. To install eSim and other dependencies run the following command:

   ```
   $ cd eSim-2.3
   $ chmod +x install-eSim.sh
   $ ./install-eSim.sh --install
   ```

3. To run eSim from the terminal, type:

   ```
   $ esim
   ```

   or you can double click on `eSim` icon created on Desktop after installation.

3.2 eSim installation in Windows OS

1. Download **eSim-2.3_installer.exe** from `https://esim.fossee.in/downloads`
2. Disable the antivirus (if any).
3. If MinGW and/or MSYS is already installed in your machine, then remove it from the PATH environment variable as it may interfere with eSim and might not work

as intended.

4. Now, double click on the exe file to start the installation process. If a window appears, click `Yes` to complete the installation.

5. By default eSim will be installed in C drive, under an auto-generated FOSSEE Folder. Note that installation directory can neither be in "Program Files" nor can contain any spaces in its path.

6. **eSim** icon will be created on desktop. You can double click on the `eSim` icon created on the Desktop after installation.

Chapter 4

Getting Started

In this chapter we will get started with eSim. Referring to this chapter will make one familiar with eSim and will help plan the project before actually designing a circuit.

4.1 How to launch eSim?

After the installation of eSim, a shortcut to eSim is created on the Desktop. To launch eSim double click on the shortcut.
Alternately, for Ubuntu Linux users, one can also launch eSim from the terminal.
1. Go to terminal.
2. Type **esim** and press `Enter`.

The first window that appears is the workspace dialog as shown in Fig. 4.1.

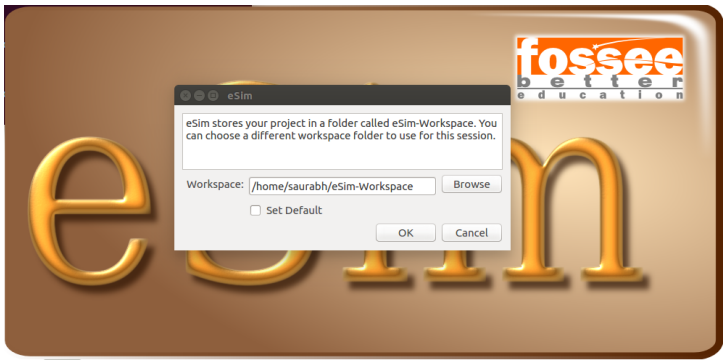


Figure 4.1: eSim-Workspace

1. The default workspace is eSim-Workspace under home directory. To select a new workspace location, use the `browse` option. Do not select a location which has a

space character or special character(s).

2. If you select the **set default** click-box, then the chosen location will be set as default workspace and the dialog box won't appear next time you launch eSim.

3. If you wish to change the default workspace location, then use the menu-bar from eSim's main Interface, which is explained in upcoming section.

4.2 eSim User Interface

The main graphic window of eSim is as shown in Fig. 4.2.

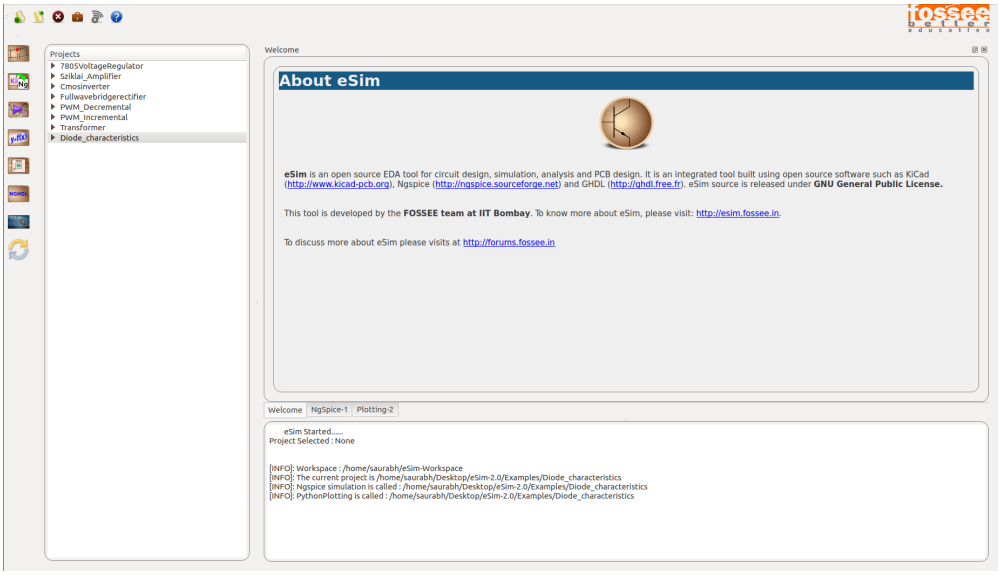


Figure 4.2: eSim Main GUI

The eSim window consists of the following sections.

1. Menubar
2. Toolbar
3. Project explorer
4. Dockarea
5. Console area

4.2.1 Menubar

- **New Project**: New projects are created in the eSim-Workspace. When this menu is selected, a new window opens up with `Enter Project name` field. Type the name of the new project and click on `OK`. A project directory will be created in eSim-Workspace. The name of this folder will be the same as that of the project created. *Make sure that the project name does not have any spaces in between.* This project is also added to the project explorer.

- **Open Project**: This opens the file dialog of default eSim-Workspace where the projects are stored. Select the required project and click on `Open`. The selected project is added to the project explorer.

- **Close Project**: This button closes the opened project.

- **Change workspace** : Clicking on this will open the window shown in Fig. 4.1. If you have chosen a default workspace location but wish to change it later on, launch eSim, click on this icon and do the necessary changes.

- **Mode Switch** : Using this feature user can decide whether to fetch latest footprints(refer Section :PCB Designing) from the internet or use the locally available footprints.
  Note : By switching to online mode, you will require a stable and high-speed internet connection, if it is not available to you then please always remain in the offline mode.

- **Help** : Clicking on this icon will launch the eSim user manual for that particular version of eSim.

4.2.2 Toolbar

The toolbar consists of the following buttons. See Fig. 4.3.

Open Schematic

The first button on the toolbar is the `Schematic Editor`. Clicking on this button will open the schematic editor. If a new project is being created, one will get a dialog box confirming the creation of a schematic. This is illustrated in See Fig. 4.4. However, if an already existing project is opened, the schematic editor window is opened. To know how to use the schematic editor to create circuit schematics, refer to Chapter 5.

When one right clicks on a particular project : three options appear, their functions listed below:

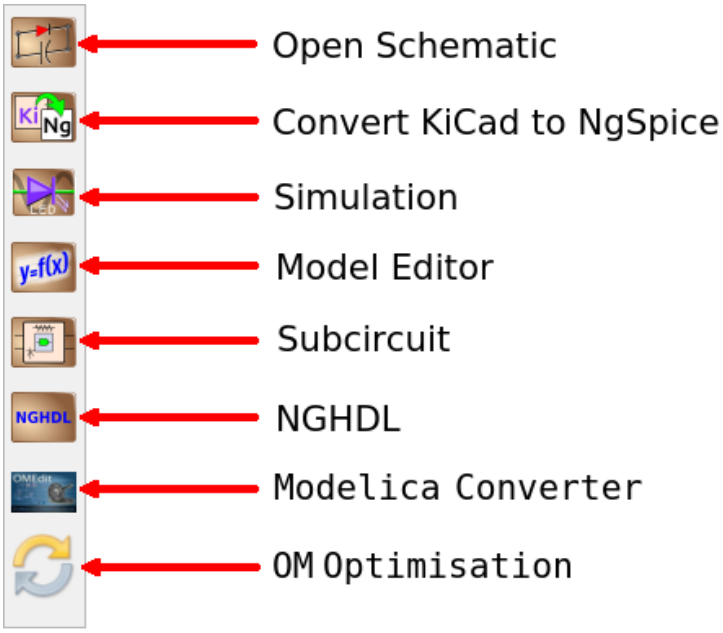


Figure 4.3: Toolbar

1. Rename Project : This will rename the project and the underlying files. Note that all project files must be closed from all running applications that access them before renaming a project.

2. Remove Project : This will remove the project selected from the `Project Explorer` list.

3. Refresh : At times, all the files under a project will not be displayed. In that case, selecting this option will update the latest files under a project.

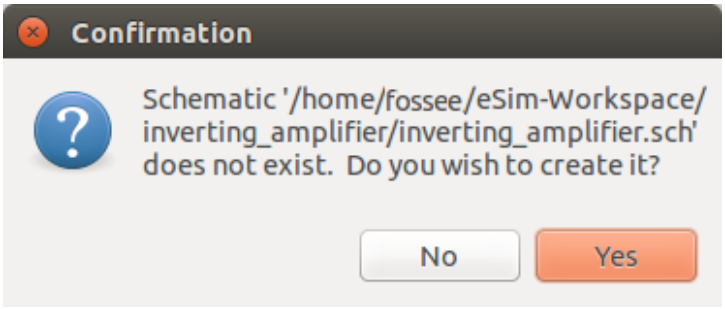


Figure 4.4: Confirmation for schematic creation

Convert KiCad to Ngspice

In the schematic editor window, after creating the schematic a netlist is to be generated which contains information about the components present in the schematic created and their values specified . Although this netlist is present, it cannot be directly fed to the simulator. Here the KiCad-to-Ngspice converter comes into play. This tool converts the netlist generated from the schematic into another netlist which is compatible with Ngspice, the simulator used in eSim. The **Convert KiCad to Ngspice** window consists of five tabs namely `Analysis, Source Details, Ngspice Model, Device Modeling and Subcircuits`. The details of these tabs are as follows.

- `Analysis:` This feature helps the user to enter the parameters for performing different types of analysis such as Operating point analysis, DC analysis, AC analysis, transient analysis, DC Sweep Analysis.

  It has the facility to select the

  - Type of analysis and
  - The simulation parameter values for analysis

- `Source Details:`eSim sources are added from `eSim_Sources` library in the schematic. Sources such as *SINE, AC, DC, PULSE, PWL* are in this library. The parameter values to all the sources added in the schematic can be given through 'Source Details' tab in the KiCad-To-Ngspice window.

- `Ngspice Model:`Ngspice has in-built model such as `basic logic gates, flip-flops, gain, summer, buffer, DAC and ADC block`etc. which can be utilised while building a circuit. eSim allows to add and modify Ngspice model parameter through Ngspice Model tab.

- `Device Modeling:`Devices like `Diode, JFET, MOSFET, IGBT, MOS` etc used in the circuit can be modeled using device model libraries. eSim also provides editing and adding new model libraries. While converting KiCad to Ngspice, these library files are added to the corresponding devices used in the circuit.

- `Subcircuits:`eSim allows you to build subcircuits. The subcircuits can again have components having subcircuits and so on. This enables users to build commonly used circuits as subcircuits and then use it across circuits. The subcircuits are added to the main circuits using this facility. We can also edit already existing subcircuits.
  Once the values have been entered, press the `Convert` button. This will generate the `.cir.out` file in the same project directory. Note that *KiCad to Ngspice Converter* can only be used if the KiCad spice netlist `.cir` file is already generated.

Simulation

The netlist generated using the `KiCad to Ngspice` converter is simulated using *Simulation* button on the eSim left toolbar. This will run the Ngspice simulation for current project. eSim have two options to see the simulation output. The first one is the Python plotting window which opens up in the dock area, as shown in Fig. 4.5. The second is the Ngspice window with the simulation data. The user can type in Ngspice commands to view the plots.

Note: If the user has used the plot components (available under eSim_Plot library) at various nodes in the circuit schematic the Ngspice plots are displayed automatically.

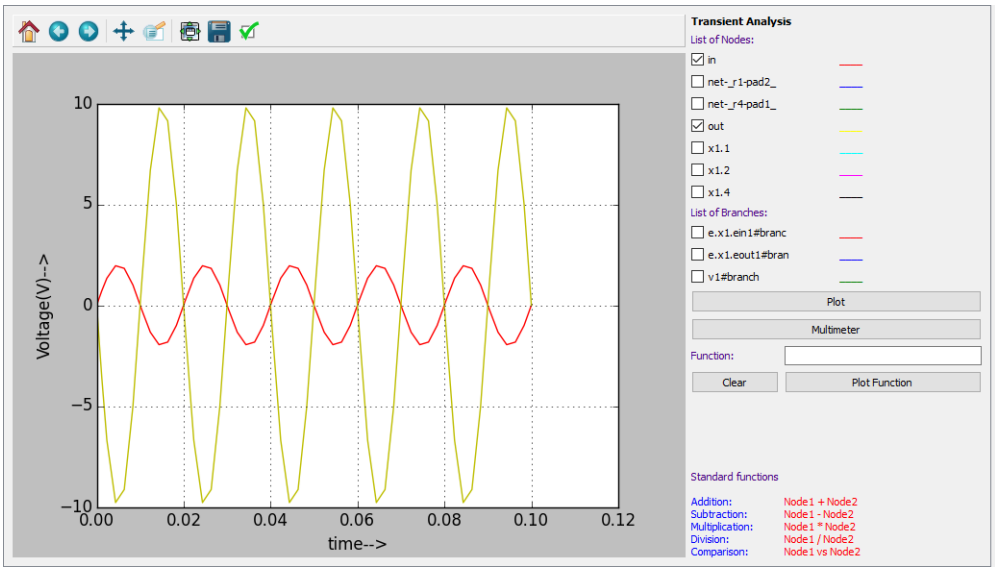


Figure 4.5: Simulation Output in Python Plotting Window

Model Editor

eSim also gives an option to re-configure the model library of a device. It facilitates the user to change model library of devices such as diode, transistor, MOSFET, etc. It also facilitates the user to load the spice library externally and use it for the existing or newly added models.To know more about Model Editor, refer to Chapter 7.

Subcircuit

eSim also allows the user to build subcircuits. The subcircuits can again have components having subcircuits and so on. This enables users to build commonly used circuits as subcircuits and then use it across circuits. For example, one can build an Op-Amp

as a subcircuit and then use it as just a single component across circuits without having to recreate it. Clicking on *Subcircuit Builder* tool will allow one to edit or create a subcircuit. To know how to make a subcircuit, refer to Chapter 8.

NGHDL

NGHDL is an add on to eSim for mixed signal circuit simulation. By using the foreign language interface of GHDL, NGHDL communicates with Ngspice and accomplishes mixed signal simulation. Using NGHDL, user can create custom digital model using VHDL language. From simple multiplexers, counters to microcontrollers and ASICs, any custom component in the digital domain can be realized using the NGHDL tool. The created digital model can be used in either mixed signal circuit or a standalone circuit operating in digital domain. NGHDL gives user the liberty to edit existing models supplied with eSim per their needs, either for experimenting new ideas or to change the model per their specific requirement.

Modelica Converter

OpenModelica (OM) is an open source modeling and simulation tool based on Modelica language. Modelica is an object oriented language. The Modelica Converter in eSim interface, converts the ngspice netlist to Modelica format. This facility will be only available if you have OpenModelica already installed in the system. More details on how to use this module is available in Chapter 10.

OM Optimisation

OMOptimisation (OMOptim) is a powerful and interactive tool for performing design optimisation. It has a good library of electrical components called Modelica Standard Library (MSL). OMOptim is stable and robust. It is very easy to add objective functions to the OMOptim interface.

4.2.3 Project Explorer

Project explorer contains the list of all the projects previously added to it. Select a project and double click on it, this will display all the files under this project. Right click on any displayed file to open it. To remove or refresh any project file from the project explorer, right click on the main project file.

4.2.4 Dockarea

This area is used to open the following windows.

1. KiCad to Ngspice converter

2. Ngspice plotting
3. Python plotting
4. Model builder
5. Subcircuit builder

Modules/Windows will appear here as per your selection.

4.2.5 Console Area

Console area provides the log information about the activity done during the current session.

Chapter 5

Schematic Creation

The first step in the design of an electronic system is the design of its circuit. This circuit is usually created using a `Schematic Editor` and is called a `Schematic`. eSim uses `Eeschema` as its schematic editor. Eeschema is the schematic editor of KiCad. It is a powerful schematic editor software. It allows the creation and modification of components and symbol libraries and supports multiple hierarchical layers of printed circuit design.

5.1 Familiarizing the Schematic Editor interface

Fig. 5.1 shows the schematic editor and the various menu and toolbars. We will explain them briefly in this section.

5.1.1 Top menu bar

The top menu bar will be available at the top left corner. Some of the important menu options in the top menu bar are:

1. File - The file menu items are given below:
   (a) New - Clear current schematic and start a new one
   (b) Open - Open a schematic
   (c) Open Recent - A list of recently opened files for loading
   (d) Save Schematic project - Save current sheet and all its hierarchy.
   (e) Save Current Sheet Only - Save current sheet, but not others in a hierarchy.
   (f) Save Current sheet as - Save current sheet with a new name.
   (g) Page Settings - Set preferences for printing the page.
   (h) Print - Access to print menu (See Fig. 5.2).
   (i) Plot - Plot the schematic in Postscript, HPGL, SVF or DXF format
   (j) Close - Close the schematic editor.

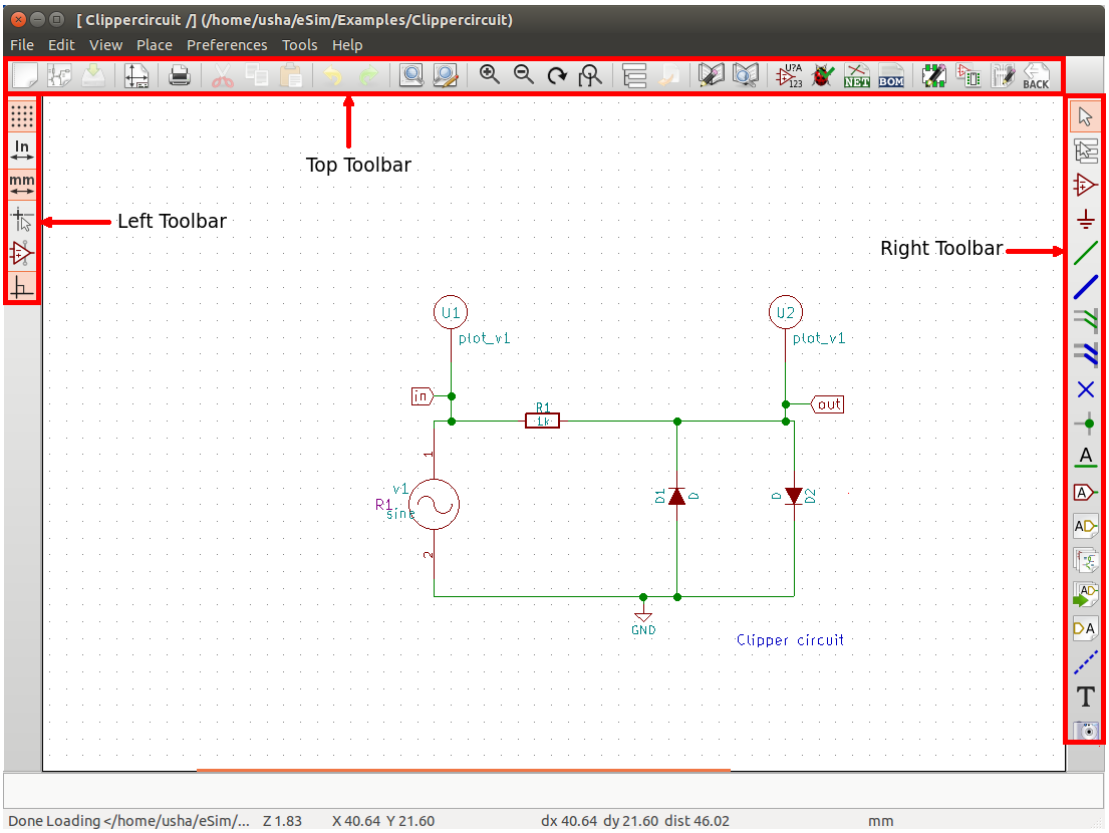


Figure 5.1: Schematic editor with the menu bar and toolbars marked

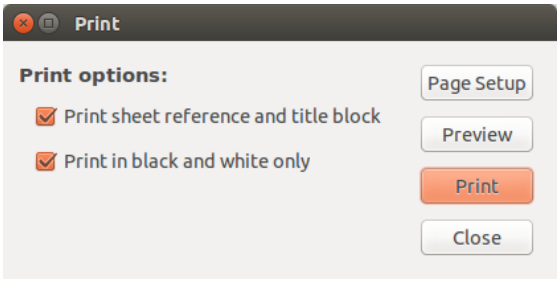


Figure 5.2: Print options

2. Place - The place menu has shortcuts for placing various items like components, wire and junction, on to the schematic editor window. See Sec. 5.1.5 to know more about various shortcut keys (hotkeys).
3. Preferences - The preferences menu has the following options:

(a) Component Libraries - Select component libraries and library paths. This enables the user to add the libraries, if the libraries are not loaded in the Eeschema.
(b) Schematic Editor Options - Select colors for various items, display options and set hot keys.
(c) Language - Shows the current list of available languages. Use default.
(d) Import and Export - Contain options to load and save preferences and import/ export hot key configuration files. See Sec. 5.1.5 to know about various hotkeys.

5.1.2 Top toolbar

Some of the important tools in the top toolbar are discussed below. They are marked in Fig. 5.3.

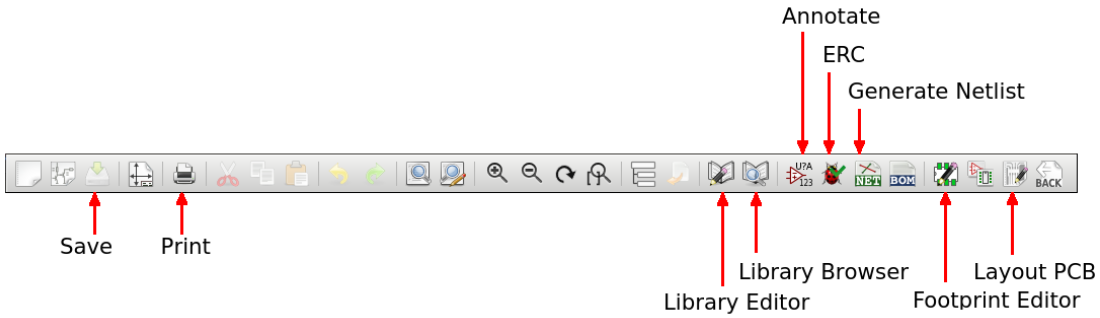


Figure 5.3: Toolbar on top with important tools marked

1. Save - Save the current schematic
2. Print - Print the schematic
3. Navigate schematic hierarchy - Navigate among the root and sub-sheets in the hierarchy
4. Library Editor - Create or edit components.
5. Library Browser - Browse through the various component libraries available
6. Annotate - Annotate the schematic
7. Check ERC - Do Electric Rules Check for the schematic
8. Generate Netlist - Generate a netlist for PCB design(`.net`) or for simulation(`.cir`).
9. Create BOM - Create a Bill of Materials of the schematic
10. Footprint editor - Map each component in the PCB netlist to a footprint
11. Layout PCB - Lay tracks between the footprints to get the PCB layout

5.1.3 Toolbar on the right

The toolbar on the right side of the schematic editor window has many important tools. Some of them are marked in Fig. 5.4. Let us now look at each of these tools and their

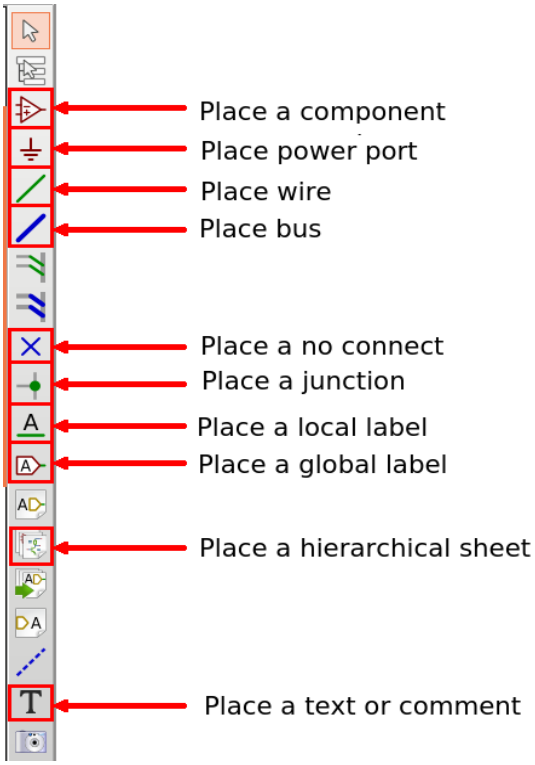


Figure 5.4: Toolbar on right with important tools marked

uses.

1. Place a component - Load a component to the schematic. See Sec. 5.3.1 for more details.
2. Place a power port - Load a power port (Vcc, ground) to the schematic.
3. Place wire - Draw wires to connect components in schematic.
4. Place bus - Place a bus on the schematic.
5. Place a no connect - Place a no connect flag, particularly useful in ICs.
6. Place a local label - Place a label or node name which is local to the schematic.
7. Place a global label - Place a global label (these are connected across all schematic diagrams in the hierarchy).
8. Create a hierarchical sheet - Create a sub-sheet within the root sheet in the hierarchy. Hierarchical schematics is a good solution for big projects.
9. Place a text or comment - Place a text or comment in the schematic.

5.1.4 Toolbar on the left

Some of the important tools in the toolbar on the left are discussed below. They are marked in Fig. 5.5.

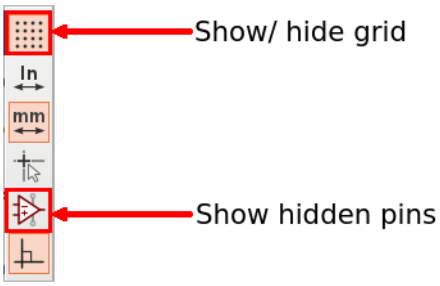


Figure 5.5: Toolbar on left with important tools marked

1. Show/Hide grid - Show or Hide the grid in the schematic editor. Pressing the tool again hides (shows) the grid if it was shown (hidden) earlier.
2. Show hidden pins - Show hidden pins of certain components, for example, power pins of certain ICs.

5.1.5 Hotkeys

A set of keyboard keys are associated with various operations in the schematic editor. These keys save time and make it easy to switch from one operation to another. The list of hotkeys can be viewed by going to Preferences in the top menu bar. Choose *Schematic Editor Options* and select *Controls* tab. The hotkeys can also be edited here. Some frequently used hotkeys, along with their functions, are given below:

- F1 - Zoom in
- F2 - Zoom out
- Ctrl + Z - Undo
- Delete - Delete item
- M - Move item
- C - Copy item
- A - Add/place component
- P - Place power component
- R - Rotate item
- X - Mirror component about X axis
- Y - Mirror component about Y axis
- E - Edit schematic component
- W - Place wire
- T - Add text
- S - Add sheet

Note: Both lower and upper-case keys will work as hotkeys.

5.2 eSim component libraries

eSim schematic editor has a huge collection of components. As Eeschema is meant to be a schematic editor to create circuits for PCB, Eeschema lacks some components that are necessary for simulation (e.g. probes(plot_v and current sources). A set of component libraries has been created with such components under the label *eSim_*.* These libraries are Ngspice compatible. If one is using eSim only for designing a PCB, then one might not need these libraries. However, these libraries are essential if one needs to simulate one's circuit. Hereafter, we will refer to these libraries as eSim libraries to distinguish them from libraries already present in Eeschema (Eeschema libraries) as shown in Fig. 5.6.

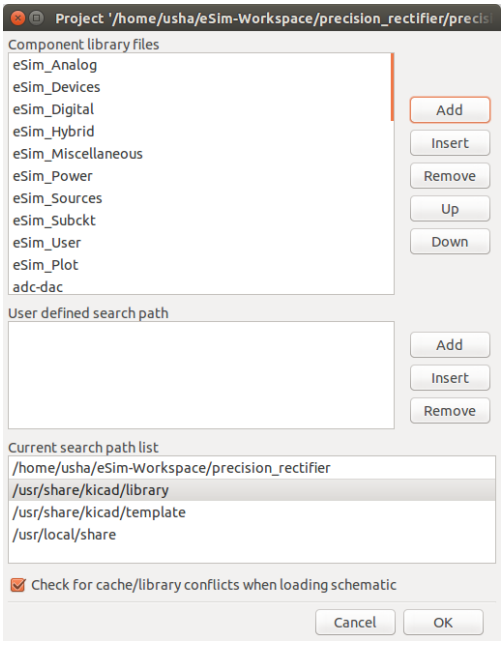


Figure 5.6: eSim-Components Libraries

The following list shows the various eSim component libraries.

- *eSim_Analog* - Contains Ngspice analog models such as aswitch(analog switch), summer(adder model), Transfo(Transformer), zener.
- *eSim_Devices* - Includes elementary components like resistor, capacitor, transistor, MosFet.
- *eSim_Digital* - Includes Ngspice digital models such as basic gates (AND, OR, NOR,NAND,XOR), filpflops (SR, D, JK), buffer, inverter.

- *eSim_Hybrid* - Includes components like ADC and DAC.
- *eSim_Miscellaneous* - Contains components like ic(used for giving initial conditions in circuit) and port(used in creating subcircuits).
- *eSim_Plot* - Contains plotting components like plot_v1 (plot voltage at a node), plot_v2 (plot voltage between 2 nodes), plot_i2 (plot current through branch), plot_log (plot logarithmic voltage at a node).
- *eSim_Power* - Includes power components like DIAC, TRIAC and SCR.
- *eSim_Sources* - Contains sources for the circuits like AC voltage source, DC voltage source, sine source and pulse source.
- *eSim_Subckt* - Contains subcircuit components like Op-Amp(UA 741), IC 555, Half adder and full adder.
- *eSim_User* - A repository for all user created components

5.3 Schematic creation for simulation

There are certain differences between the schematic created for simulation and that created for PCB design. We need certain components like plots and current sources for simulation whereas these are not needed for PCB design. For PCB design, we would require connectors (e.g. DB15 and 2 pin connector) for taking signals in and out of the PCB whereas these have no meaning in simulation. This section covers schematic creation for simulation. Refer to Chapter 13 to know how to create schematic for PCB design.

The first step in the creation of circuit schematic is the selection and placement of required components. Let us see this using an example. Let us create the circuit schematic of an RC filter given in Fig. 5.10c and do a transient simulation.

5.3.1 Selection and placement of components

We would need a resistor, a capacitor, a voltage source, ground terminal and some plot components. To place a resistor on the schematic editor window, select the *Place a component* tool from the toolbar on the right side and click anywhere on the schematic editor. This opens up the component selection window. This action can also be performed by pressing the key A. Choose the *eSim_Devices* library and click on the arrow near it. This will open the *eSim_Devices* library and the resistor component can be found here. Fig. 5.7 shows the selection of resistor component. Click on `OK`. A resistor will be tied to the cursor. Place the resistor on the schematic editor by a single click. To place the next component, i.e., capacitor, click again on the schematic editor. The capacitor component is also found under *eSim_Devices* library. Select it and then click on `OK`. Place the capacitor on the schematic editor by a single click.

Let us now place a sinusoidal voltage source. This is required for performing transient analysis. On the component selection window, choose the library *eSim_source*. Select

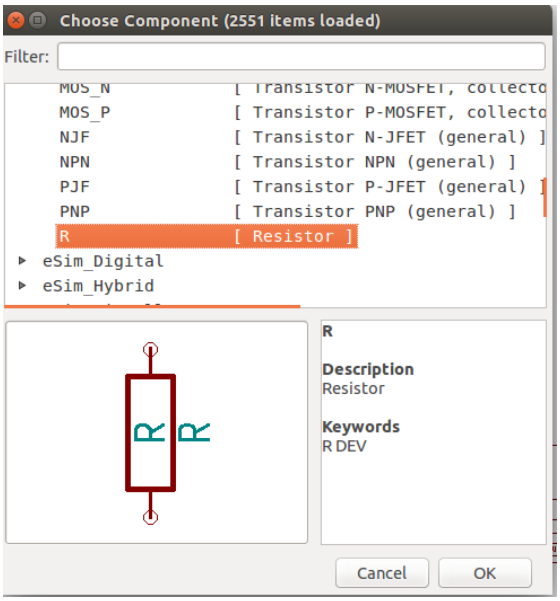


Figure 5.7: Placing a resistor using the Place a Component tool

the component `SINE` and click on `OK`. Place the sine source on the schematic editor by a single click. Similarly select and place `gnd`, a ground terminal from the `power` library.

The plot components can be found under the eSim_Plot library. Select the plot_v1 component and place the component. Once all the components are placed, the schematic editor would look like as in Fig. 5.8.

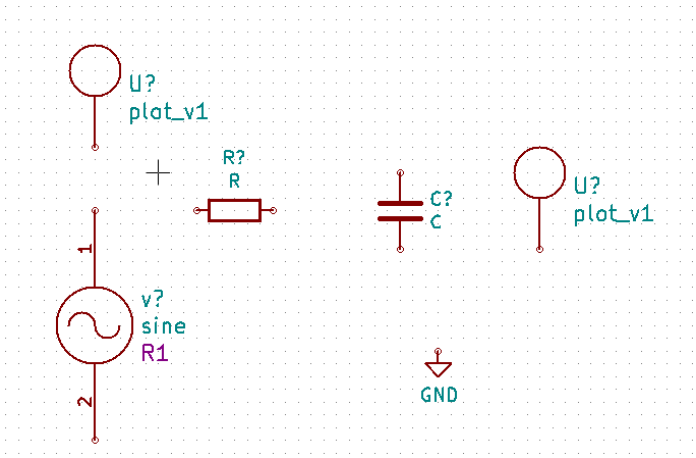


Figure 5.8: All RC circuit components placed

Let us rotate the resistor to complete the circuit. To rotate the resistor, place the

cursor on the resistor as shown in Fig. 5.9 and press the key `R`. This applies to all components.

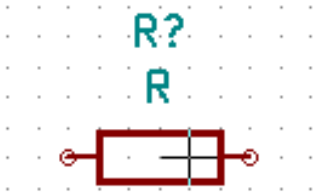
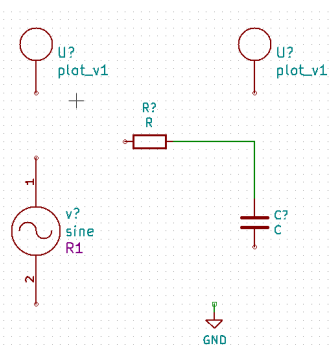


Figure 5.9: Placing the cursor (cross mark) on the resistor component

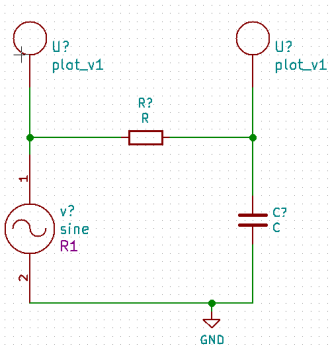
If one wants to move a component, place the cursor on top of the component and press the key `M`. The component will be tied to the cursor and can be moved in any direction.

5.3.2 Wiring the circuit

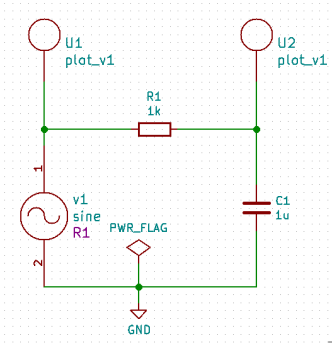
The next step is to wire the connections. Let us connect the resistor to the capacitor. To do so, point the cursor to the terminal of resistor to be connected and press the key `W`. It has now changed to the wiring mode. Alternately, this can also be done by selecting the *Place wire* tool on the right side toolbar. Move the cursor towards the terminal of the capacitor and click on it. A wire is formed as shown in Fig. 5.10a. Similarly connect



(a) Initial stages



(b) Wiring done



(c) Final schematic with PWR_FLAG

Figure 5.10: Various stages of wiring

the wires between all terminals and the final schematic would look like Fig. 5.10b.

5.3.3 Assigning values to components

We need to assign values to the components in our circuit i.e., resistor and capacitor. Note that the sine voltage source has been placed for simulation. The specifications of sine source will be given during simulation. To assign value to the resistor, place the cursor above the letter `R` (not `R?`) and press the key `E`. Choose *Field value*. Type `1k` in the *Edit value field* box as shown in Fig. 5.11. 1k means $1k\Omega$. Similarly give the value `1u` for the capacitor. 1u means $1\mu F$.

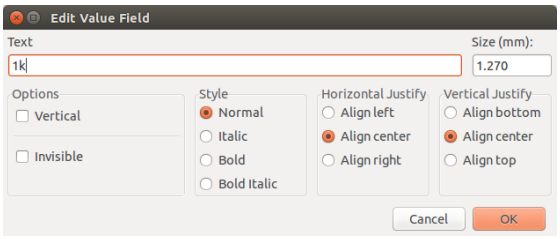


Figure 5.11: Editing value of resistor

5.3.4 Annotation and ERC

The next step is to annotate the schematic. Annotation gives unique references to the components. To annotate the schematic, click on *Annotate Schematic Components* tool from the top toolbar. Click on `Annotate`, then click on `OK` and finally click on Close as shown in Fig. 5.12. The schematic is now annotated. The question marks next to component references have been replaced by unique numbers. If there are more than one instance of a component (say resistor), the annotation will be done as R1, R2, etc.

Let us now do `ERC` or `Electric Rules Check`. To do so, click on *Perform electric rules check* tool from the top toolbar. Click on *Run* button. The error as shown in Fig. 5.13 may be displayed. Click on close in the test erc window.

There will be a green arrow pointing to the source of error in the schematic. Here it points to the ground terminal. This is shown in Fig. 5.14.

This error is due ti the GND pin. The GND pin is a power input pin and Eeschema gives this error as there is no power line connected. To correct this error, a flag has to be placed indicating that there will be an external power line connected to it. Place a `PWR_FLAG` from the Eeschema library *power*. Connect the power flag to the ground terminal as shown in Fig. 5.10c. Repeat the ERC. Now there are no errors. With this we have created the schematic for simulation.

5.3.5 Netlist generation

To simulate the circuit that has been created in the previous section, we need to generate its netlist. `Netlist` is a list of components in the schematic along with their connection

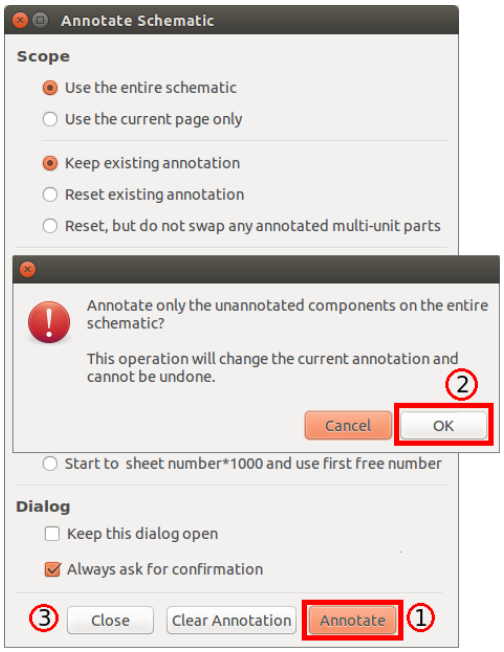


Figure 5.12: Steps in annotating a schematic: 1. First click on Annotation then 2. Click on OK then 3. Click on close

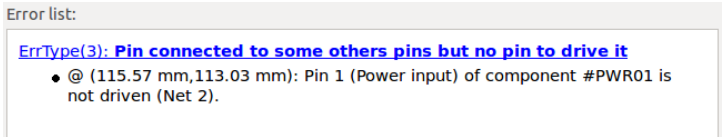


Figure 5.13: ERC error

information. To do so, click on the *Generate netlist* tool from the top toolbar. Click on Spice from the window that opens up. Check the option `Default Format`. Then click on *Generate*. This is shown in Fig. 5.15. Save the netlist. This will be a `.cir` file. Do not change the directory while saving. Now the netlist is ready to be simulated. Refer to [1] or [2] to know more about Eeschema.

5.4 Tools for creating the PCB layout

The Eeschema top toolbar also has two important tools which can help the user to generate the PCB layout of the created schematic.

Figure 5.14: Green arrow pointing to Ground terminal indicating an ERC error

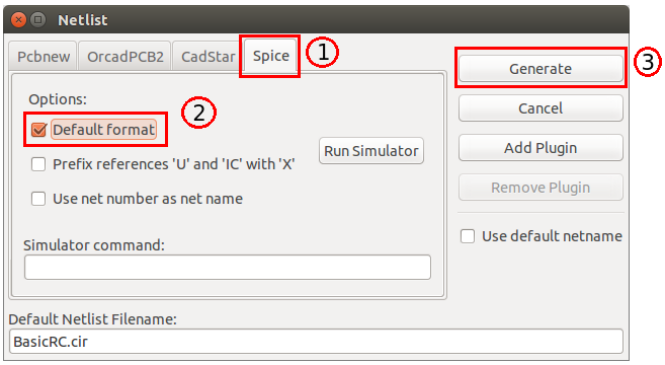


Figure 5.15: Steps in generating a Netlist for simulation: 1. Click on Spice then 2. Check the option `Default Format` then 3. Click on Generate

5.4.1 FootPrint Editor

Clicking on the *Footprint Editor* tool will open the `CvPcb` window. This window will ideally open the .net file for the current project. So, before using this tool, one should have the netlist for PCB design (a .net file). To know more about how to assign footprints to components, see Chapter 13.

5.4.2 PCB Layout

Clicking on the *Layout Editor* tool will open `Pcbnew`, the layout editor used in eSim. In this window, one will create the PCB. It involves laying tracks and vias, performing optimum routing of tracks, creating one or more copper layers for PCB, etc. It will be saved as a `.brd` file in the current project directory. Chapter 13 explains how to use the *Layout Editor* to design a PCB.

Chapter 6

Simulation

Circuit simulation uses mathematical models to replicate the behaviour of an actual device or circuit. Simulation software allows to model circuit operations. Simulating a circuit's behaviour before actually building it can greatly improve design efficiency. eSim uses `Ngspice` for analog, digital and mixed-level/mixed-signal circuit simulation. The various steps involved in simulating a circuit schematic in eSim are explained in the sections below:

6.1 Kicad to Ngspice Conversion

In the chapter on schematic creation, we have learnt to generate the netlist from circuit schematic. The generated netlist is not compatible with Ngspice. eSim uses Ngspice to simulate the circuit schematic. Hence the netlist i.e. `.cir` file generated should be converted in to a Ngspice compatible file. The *Convert KiCad to Ngspice* tool on eSim left toolbar is used to do this. Let us now see the various tabs and their functions available under this.

6.1.1 Analysis

In order to simulate a circuit, the user must define the type of analysis to be done on the circuit. This tab is used to insert the type of analysis and value of the analysis parameters to the netlist. eSim supports three types of analyses: 1. *DC Analysis* (Operating Point and DC Sweep) 2. *AC Small-signal Analysis* 3. *Transient Analysis* These are explained below.

In the current example for simulating an RC circuit, select the analysis type as `transient` analysis and enter the values as shown in the Fig. 6.1.

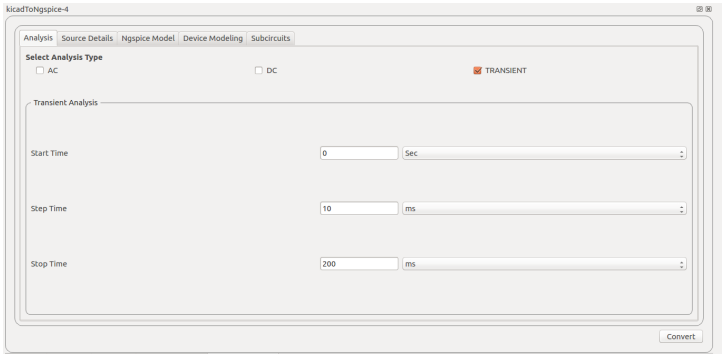


Figure 6.1: KiCad to Ngspice Window

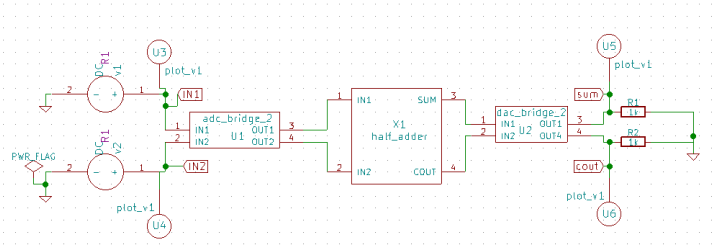


Figure 6.2: Half Adder Schematic

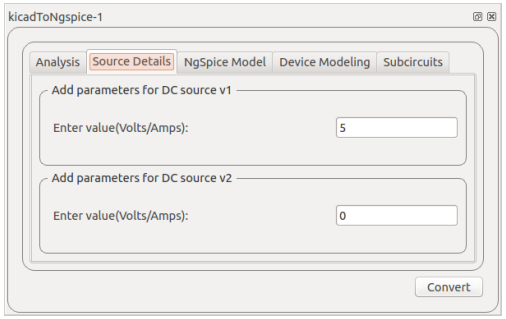
6.1.2 Source Details

The various parameter values of the sources added in the schematic can added using this tool. *Source details* is a dynamic tab, i.e. the fields are added as per the number of sources in the circuit. For example, consider a Half-Adder circuit as shown in Fig. 6.2 Here, we have used two DC input sources are used and hence the source detail GUI would be having two input fields as shown is Fig. 6.3a.

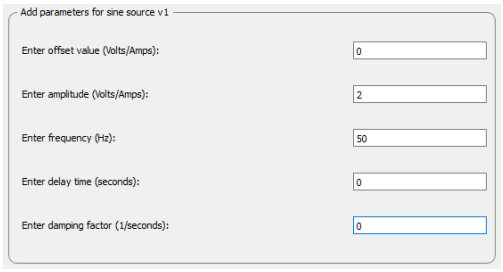
In the current example of the RC circuit, we have a single AC source. Fill in the details as shown in Fig. 6.3b.

6.1.3 Ngspice Model

The component libraries for components like DAC, ADC, transformer etc. which are used in the schematic are directly linked with the corresponding Ngspice models. The user can modify the parameter values using this tab, as shown in Fig. 6.4. If there are no modifications the default values are taken.



(a) Source Details of Half-Adder



(b) Source Details of RC circuit

Figure 6.3: Source details interface

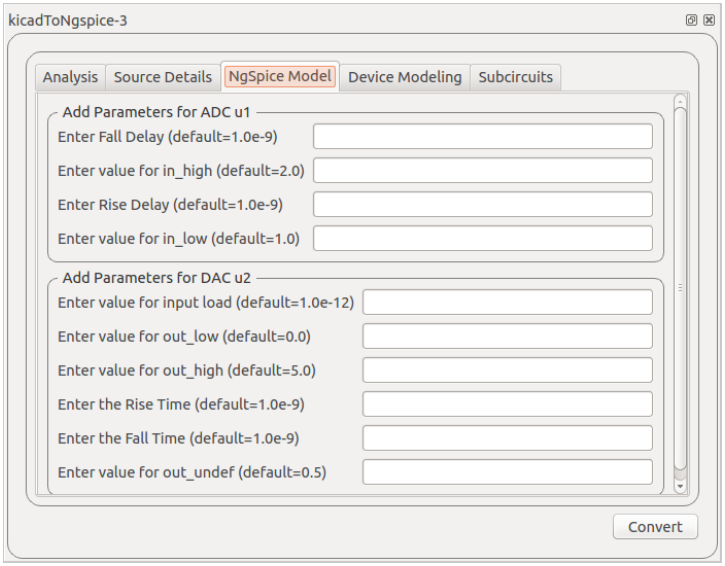


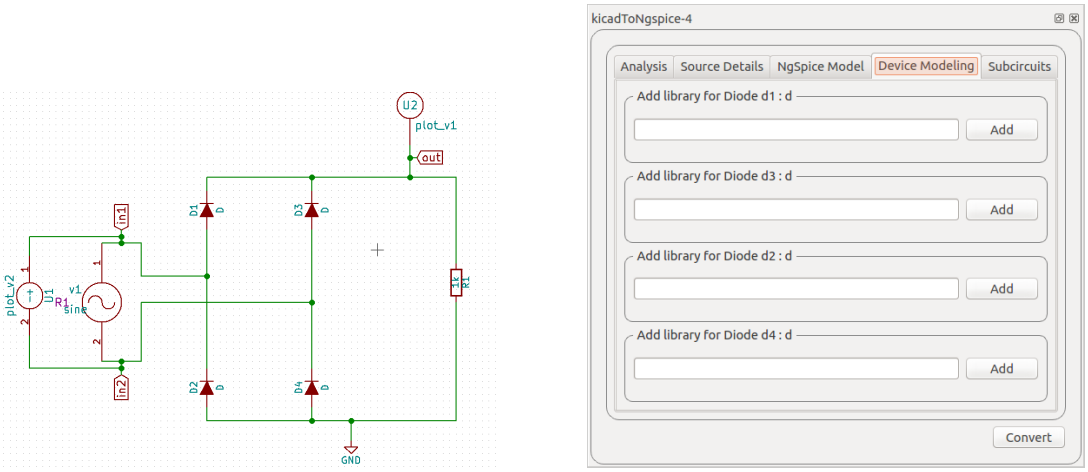
Figure 6.4: Half adder: Ngspice model

6.1.4 Device Modelling

Spice based simulators include a feature which allows accurate modeling of semiconductor devices such as diodes, transistors etc. Model libraries holds these features to define models for devices such as diodes, MOSFET, BJT, JFET, IGBT, Magnetic core etc.

The fields in this tab are added for each such device in the circuit and the corresponding model library is added. In the example of bridge rectifier as shown in Fig. 6.5a for four diodes library files are added as in Fig. 6.5b. Location for these libraries is as following :
library/deviceModelLibrary/Diode/ if you are using version 2.0 and above

src/deviceModelLibrary/Diode/ if you are using versions lower than 2.0



(a) Schematic (b) Adding device model

Figure 6.5: Bridge Rectifier

6.1.5 Sub Circuit

Subcircuit is a way to implement hierarchical modeling. Once a subcircuit for a component is created, it can be used in other circuits.
In the KiCadToNgspice conversion of example 7805VoltageRegulator, where a bridge rectifier is further connected to a voltage regulator LM7805, a subcircuit of this 7805 IC is used. These are located in library/SubcircuitLibrary if you are using version 2.0 and above and in src/SubcircuitLibrary if you are using versions lower than 2.0. The association is done as shown in Fig. 6.6.

After Filling up the values in all the above mentioned fields the convert button is pressed. The Ngspice netlist, `.cir.out` file is generated. A message box pops up, as shown in Fig. 6.7. Click on `OK`.

6.2 Simulating the schematic

6.2.1 Simulation

Once the Kicad to Ngspice conversion is successfully completed, press the *Simulation* button on the leftside toolbar on eSim interface. This will display the Python plot window along with the node names. Select the required nodes and click on `Plot`. The

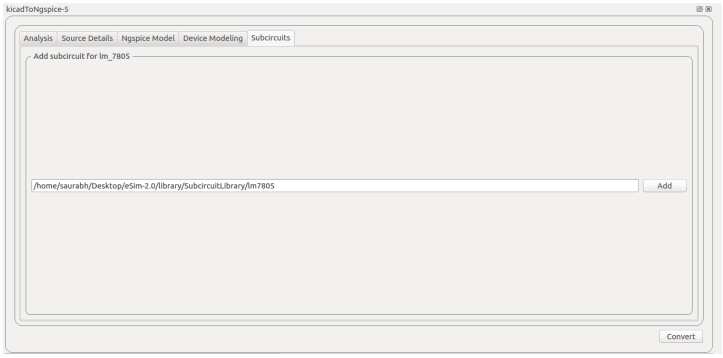


Figure 6.6: Assigning appropriate subcircuit file

Figure 6.7: Message after successful Ngspice netlist generation

simulations are displayed. In the present example of the RC circuit, the plot will be displayed as shown in Fig. 6.8. By changing the values of capacitor and resistor, the output can be varied.We can also use the option `Function` from the right side of the python plot window to plot combination of waveforms for example `plot V1+V2`.

Pressing the *Simulation* button also opens up the Ngspice terminal and plot windows. The Ngspice plots for all the nodes (where we have used the plot components in

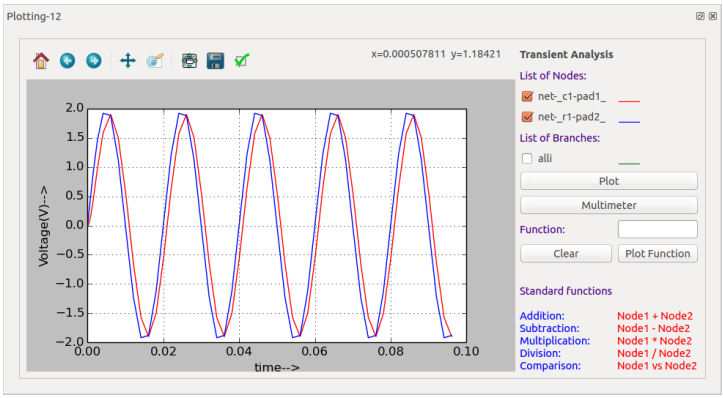


Figure 6.8: Pythonplot for RC circuit

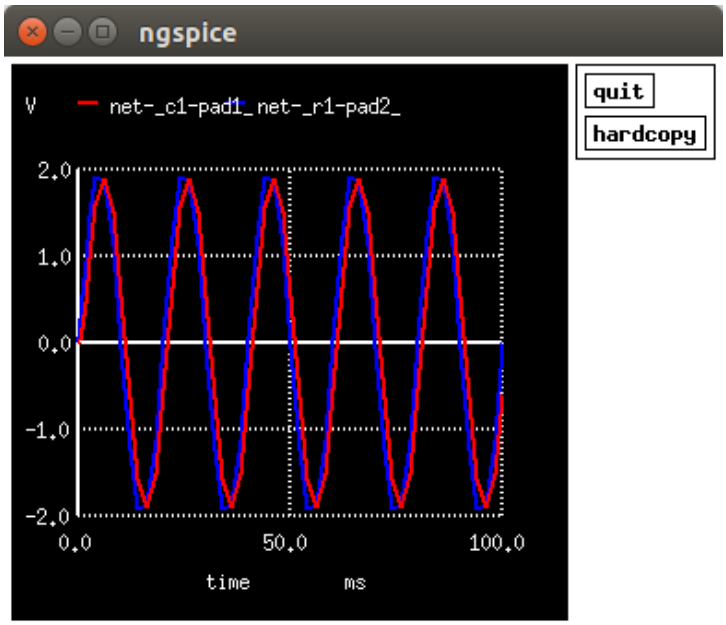


Figure 6.9: Ngspice voltage simulation for RC circuit

the schematic) will be displayed. In the current example, we have used two plot components and the Ngspice simulations for these two nodes are displayed as in Fig. 6.9.

If the *Plot components* are not used in the schematic, the simulations are not displayed automatically. To see the Ngspice simulations, type the following commands in the Ngspice terminal window.

- **plot allv** - Plots all the voltage waveforms.
- **plot v(node-name)** - Plot a waveform of the node-name voltage source e.g. **plot v(out)** will plot the voltage at node **out**
- **plot v(node-one) v(node-two)** - Plots waveforms of voltages at node-one and node-two on a single graph. Multiple nodes can be observed in a single graph, though scaling has to be kept in mind.
- **plot alli** - Plots all the current waveforms.

  You can refer the ngspice commands from `https://esim.fossee.in/ngspicecmd`

6.2.2 Multimeter

Multimeter is another feature that is available in eSim. Using this facility the user can view the voltage and current values in various nodes and branches respectively. To use

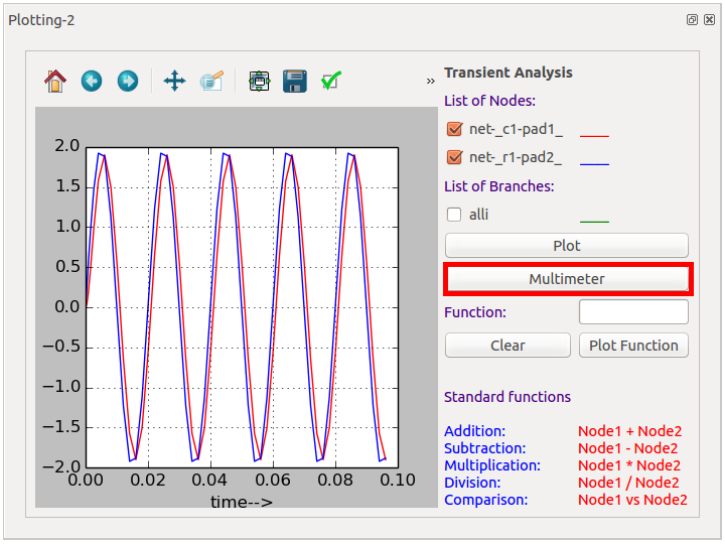


Figure 6.10: Multimeter feature in eSim

the multimeter select the required nodes from the plot window and press `Multimeter` button, shown in Fig. 6.10. Windows equal to the number of selected nodes will open. Now open the schematic window and place these pop up windows near the appropriate nodes on the schematic to get the voltage of each node. Similarly current through each branches in the schematic can also be found using the multimeter facility.

Chapter 7

Model Editor

Spice based simulators include a feature which allows accurate modeling of semiconductor devices such as diodes, transistors etc. eSim Model Editor provides a facility to define a new model for devices such as *diodes, MOSFET, BJT, JFET, IGBT, Magnetic core* etc. Model Editor in eSim lets the user enter the values of parameters depending on the type of device for which a model is required. The parameter values can be obtained from the data-sheet of the device. A newly created model can be exported to the model library and one can import it for different projects, whenever required. Model Editor also provides a facility to edit existing models. The GUI of the model editor is as shown in Fig. 7.1

7.1 Creating New Model Library

eSim lets us create new model libraries based on the template model libraries. On selecting `New` button the window is popped as shown in Fig. 7.2. The name has to be

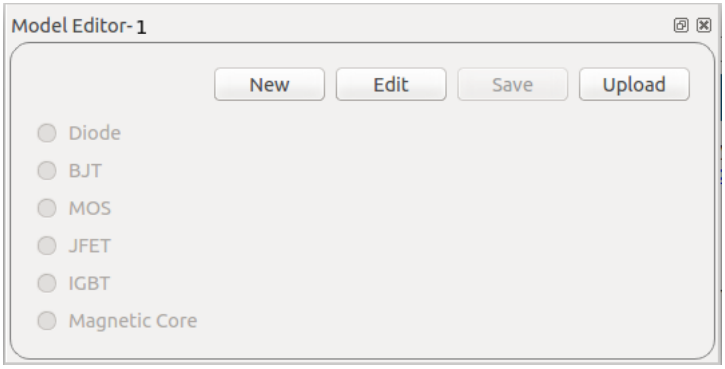


Figure 7.1: Model Editor

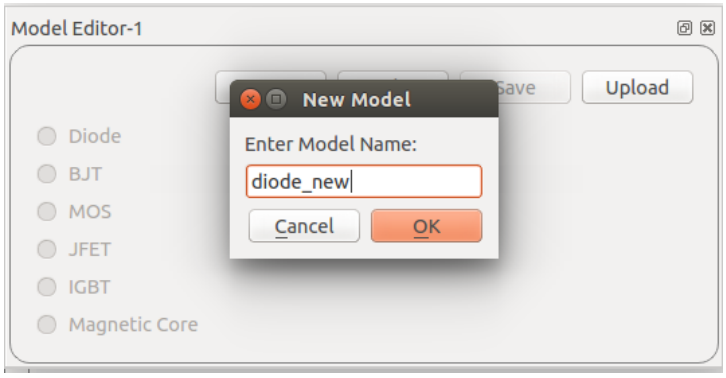


Figure 7.2: Creating New Model Library

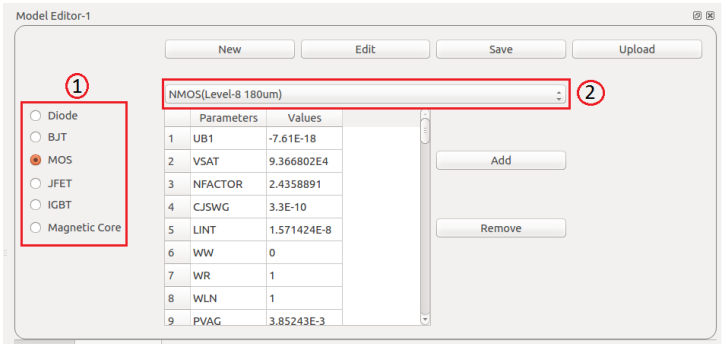


Figure 7.3: Choosing the Template Model Library

unique otherwise the error message appears on the window.

After the OK button is pressed the type of model library to be created is chosen by selecting one of the types on the left hand side i.e. `Diode, BJT, MOS, JFET, IGBT, Magnetic Core`. The template model library opens up in a tabular form as shown in Fig. 7.3

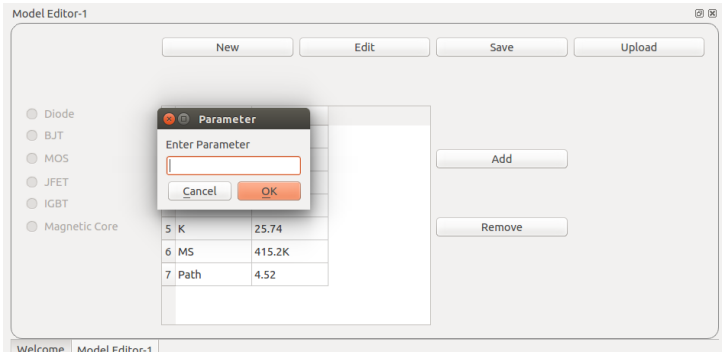


Figure 7.4: Adding the Parameter in a Library

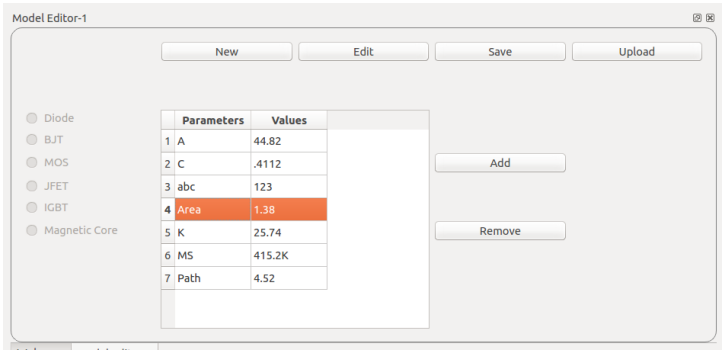


Figure 7.5: Removing a Parameter from a Library

New parameters can be added or current parameters can be removed using `ADD` and `REMOVE` buttons. Also the values of parameters can be changed in the table. Adding and removing the parameters in library files is shown in the Fig. 7.4 and Fig. 7.5

After the editing of the model library is done, the file can be saved by selecting the `SAVE` button. These libraries are saved in the *User Libraries* folder under *deviceModel-Library* directory.

7.2 Editing Current Model Library

The existing model library can be modified using `EDIT` option. On clicking the `EDIT` button the file dialog opens where all the library files are saved as shown in Fig. 7.6. You can select the library you want to edit. Once you are done with the editing, click on `SAVE` button.

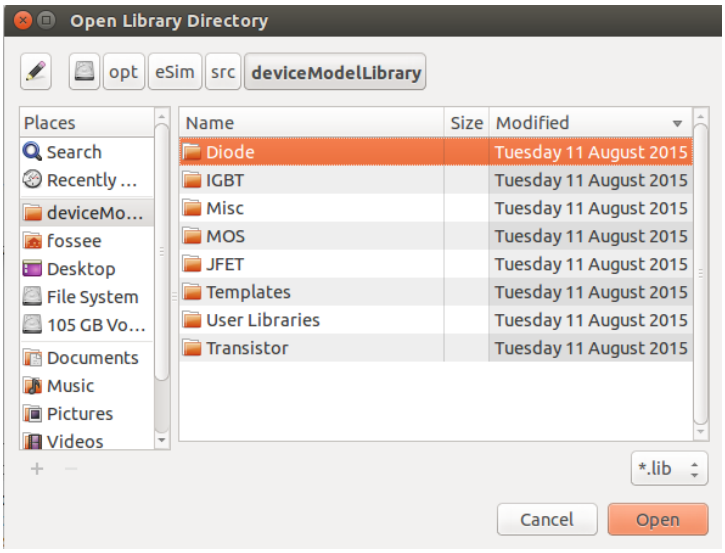


Figure 7.6: Editing Existing Model Library

7.3 Uploading external .lib file to eSim repository

User can also upload external spice **.model** library files. These .model libraries can be downloaded online. eSim directly cannot use the external .lib file. It has to be uploaded to eSim repository before using it in a circuit. eSim provides the facility to upload library files using the **U**pload option in the *Model Editor*. They are then converted into xml format, which can be easily modified from the eSim interface. On clicking `UPLOAD` button the library can be uploaded from any location. The model library will be saved with the name you have provided, in the *User Libraries* folder of repository *deviceModelLibrary*. Example: You can download any model of Schottky diode from Spice website and save it as .lib extension on the system. Click on `UPLOAD` option and give the path. The lib file along with XML file is created in the `eSim-<version>/library/deviceModeLibrary/UserLibraries` if you are using v2.0 and above versions; otherwise
`eSim-<version>/src/deviceModeLibrary/UserLibraries` if you are using versions lower than 2.0
. The uploaded library can be used for the existing part eSim_Diode or the user can create a new model (part). Refer Chapter 8 on how to create a new part library model in eSim.

Chapter 8

Subcircuit Builder

Subcircuit is a way to implement hierarchical modeling. Once a subcircuit for a component is created, it can be used in other circuits. eSim provides an easy way to create a subcircuit. The following Fig. 8.1 shows the window that is opened when the Subcircuit tool is chosen from the toolbar.

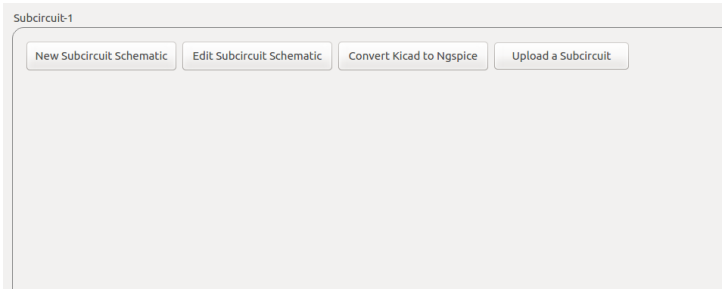


Figure 8.1: Subcircuit Window

8.1 Creating a Subcircuit

The steps to create subcircuit are as follows.

- After opening the Subcircuit tool, click on `New Subcircuit Schematic` button. It will ask the name of the subcircuit. Enter the name of subcircuit (without any spaces) and click `OK` as shown in Fig. 8.2.
- After clicking `OK` button it will open KiCad schematic. Draw your circuit which will be later used as a subcircuit. e.g the Fig. 8.3 shows the half adder circuit.
- Once you complete the circuit, assign a PORT to each open node of your circuit which will be used to connect with the main circuit. The port should match with

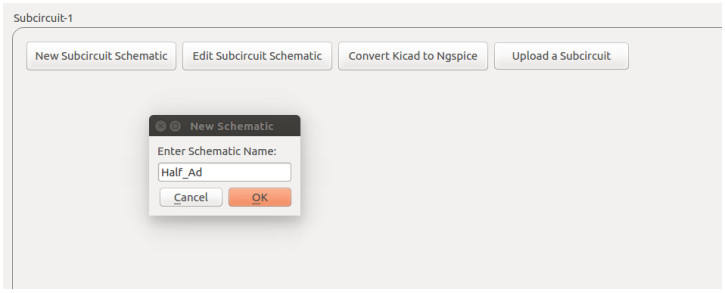


Figure 8.2: New Sub circuit Window

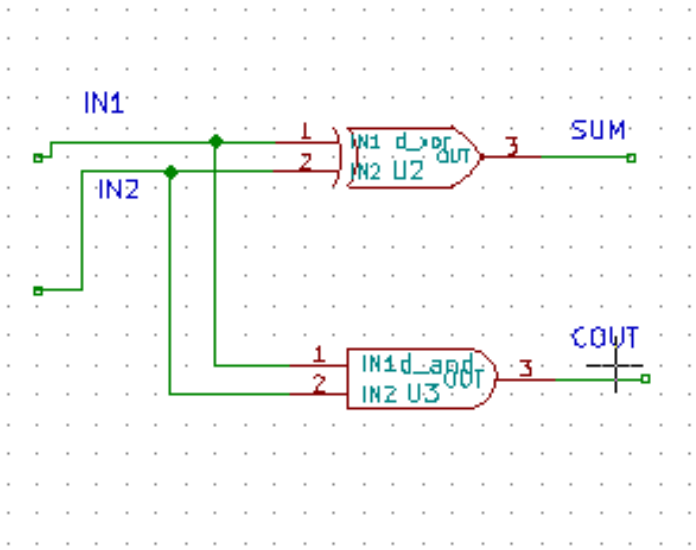


Figure 8.3: Inner circuit of the subcircuit

the number of input and output pin. The circuit will look like Fig. 8.4 after adding PORT to it. The PORT component can be found in the eSim_Miscellaneous library as shown in Fig. 8.5. Select a different port for each node (input or output), the PORT has 26 such components named alphabetically as Unit A, Unit B to Unit Z, meaning you can create a subcircuit up-to 26 pins(input, output combined).

- Next step is to save the schematic and generate KiCad netlist as explained in Chapter 5.
- To use this subcircuit in other schematics, create a block in the schematic editor by following steps given below as one should have a symbol corresponding to the newly created subcircuit that can be used in other schematics:

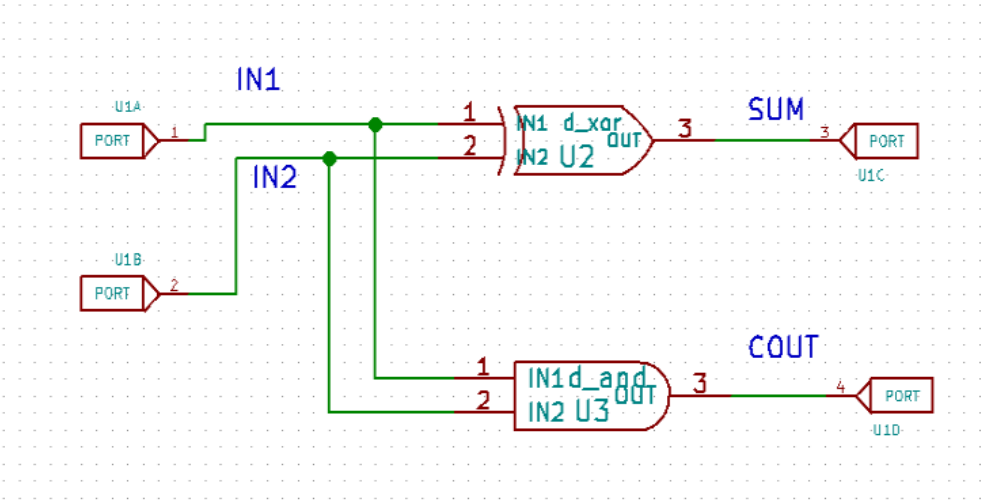


Figure 8.4: Half-Adder Subcircuit

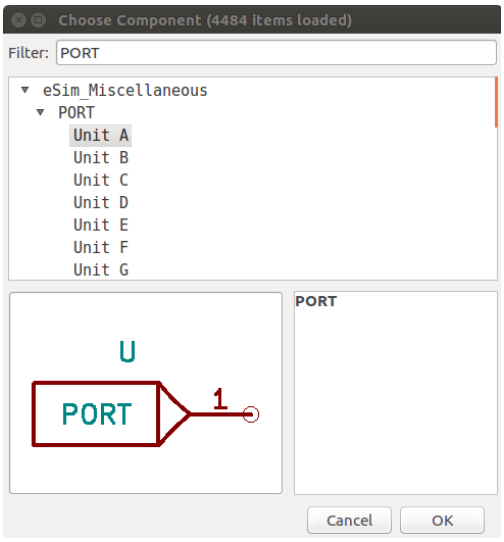


Figure 8.5: Selection of PORT component

1. Go to library browser of the schematic editor. It is an "open book with a pencil in its middle" icon on the top toolbar.
2. Select the Current Library as eSim_Subckt shown in Fig. 8.6

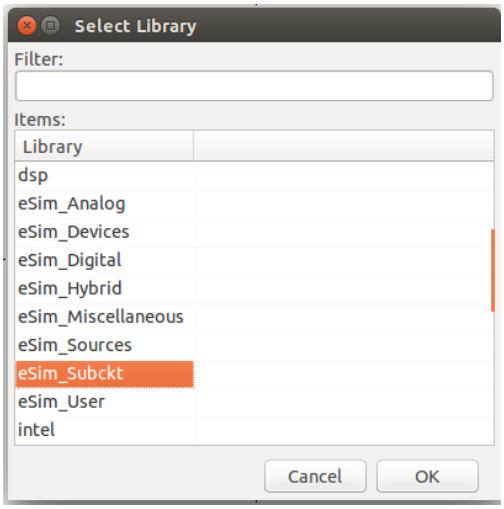


Figure 8.6: Selecting Working Library

3. Click on create a new component from the top toolbar.
4. Give the same name that was used for creating the new subcircuit's internal diagram, refer Fig. 8.2.
5. Choose designator as **X**. If any other reference designator other than X is used for **subcircuit**, your subcircuit will not be recognised during simulation.
6. Similarly, reference designator are as follows for different types of components. D is for diode, Q is for transistors, J is for FET. The user needs to choose the appropriate reference based on the library in which they wish to add a model.
7. Start drawing the subcircuit block by using the drawing tools from the right taskbar. Here we have used **Add graphic rectangle to component body**. You can start drawing with a point to point click on the editor.
8. To add pins select **Add pins to components** from the right taskbar. Give the `Pin Name` as `IN1` and `Pin Number as 1`. The pin number has to match with the `Port name`. Example Port A is mapped to pin 1. Select the **Orientation** as right or left accordingly. The **Electrical Type** has to be chosen as `Input` for nodes which will act as Input in the subcircuit you are creating. Similar logic is for output nodes. We would recommend to declare the ports as either Input, Output or Passive.
9. The final block of the subcircuit would look as shown in Fig. 8.8. Pins should be attached properly. Labels(Names to the PORTs) should be given such that it is intuitive and someone other than you should be able to understand and use that block with least amount of hassle.

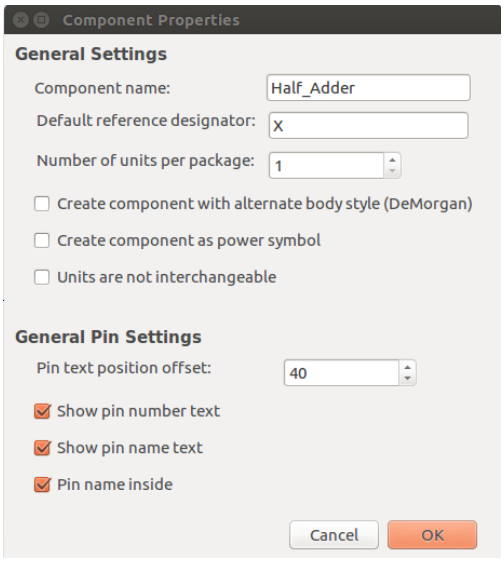


Figure 8.7: Creating New Component

10. In order to save this file, press **Ctrl+S** keys and click **yes** for confirmation purposes.
11. Note : A good practice to retain this created subcircuit would be to take a backup of this library. To do that, click on **File** from the library editor window and select the **Save Current Library as** option. A location needs to be selected, please select **eSim-Workspace** as the location for storing this file and give relevant name e.g. **eSim-Subckt-backup**. Later other users can use this in their circuits.

Specifying parameters for generating the .sub file

1. A **.sub** file is nothing but textual representation that is passed to the simulator which essentially informs the simulator about the nodes, and behavior of the subcircuit block. Remember the Fig. 8.4 circuit? It will be saved in a **.sub** file once we complete this process!
2. Switch to the eSim main window and click on **Convert KiCad to Ngspice button** in the **subcircuit builder tool**. as shown in Fig. 8.1
3. You need not assign any values in the transient parameters section.
   Assign the values to any voltage or current sources present in your internal circuit, if any.
   Add the appropriate device libraries or subcircuit libraries if you have used any Device Models or Subcircuits, if any.

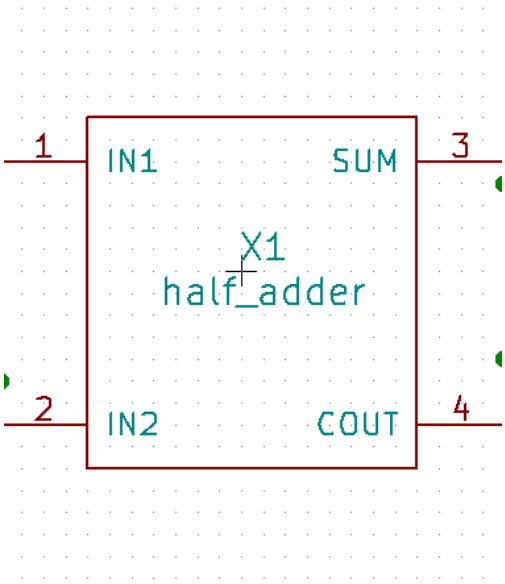


Figure 8.8: Half-Adder Subcircuit Block

4. Upon successful generation of the subcircuit file, an acknowledgement message will be displayed. To confirm, go to
   For Windows OS users :
   C:/FOSSEE/eSim/Library/SubcircuitLibrary if you are using v2.0 and above
   C:/FOSSEE/eSim/src/SubcircuitLibrary if you are using versions **lower than 2.0**

   For Ubuntu Linux Users
   ../eSim-2.0/Library/SubcircuitLibrary if you are using v2.0 and above
   ../eSim-1.1.3/src/SubcircuitLibrary if you are using versions **lower than 2.0**
   And make sure that the .sub file is present under the directory carrying the name of the subcircuit that you specified in step in Fig. 8.2.

8.2 Edit a Subcircuit

The steps to edit a subcircuit are as follows.

- After launching the Subcircuit tool, click on **Edit Subcircuit Schematic** button. It will open a dialog box where you can select any subcircuit for editing.

- After selecting the subcircuit it will open it in the schematic editor, where you can edit the subcircuit.
- Next step is to save the schematic and generate the .cir netlist.
- If you have edited the number of ports then you have to change the block exaplained in section Creating a Subcircuit accordingly.

Note:

- User can also import or append the schematic of different projects in the current page using the *Append Schematic Sheet* from the *File* menu. This will import(copy) the schematic that user has defined to the current schematic editor page.
- User can also import the model in the part library editor page using the option *Import Component* from the top toolbar.

8.3 Upload subcircuit

- Using this feature, one can import an existing subcircuit file into eSim environment. You necessarily need not create the schematic for this.
- Download the required subcircuit's .sub file from many online resources/repositories.
- Upload this file using the upload subcircuit feature.
- Upon uploading following checks will be made, and only and only if the checks are satisfied, the file will be uploaded. The checks are as following :
  1. The uploaded file should have the extension **.sub**
  2. The name of the file, say for example is **omega.sub**, then the content of the file must start with **.subckt omega** and end with **.ends omega**.
  Any line that starts with asterisk sign(*) is considered as a comment in these types of files. Hence, the file technically starts with **.subckt**.
- If above conditions are satisfied, then the file will be automatically placed in a folder that carries the same name as that of the .sub file will be created in ../SubcircuitLibrary/ directory.
- Once above steps are verified, proceed to create a block as shown in Fig. 8.8 and name should be same as that of the corresponding .sub file uploaded earlier. Pins of this block should match the number of pins stated in the .sub file.
  NOTE: ONLY AND ONLY THE OUTER BLOCK NEEDS TO BE CREATED. INTERNAL CIRCUIT IS NOT REQUIRED IF YOU ARE USING THE UPLOAD FEATURE.

Chapter 9

NGHDL: Mixed Signal Simulation

NGHDL feature facilitates creation of user-defined models for mixed-signal circuit simulation in eSim. By interfacing GHDL and Ngspice, we achieve mixed-signal simulation. Digital models are simulated using GHDL and XSPICE engine of Ngspice.

9.1 Introduction

Ngspice supports mixed-signal simulation, i.e. it can simulate both digital and analog component. It defines a `model` which has the functionality of the circuit component, which can be used in the netlist. For example you can create an `adder` model in Ngspice and use it in any circuit netlist of Ngspice.

However, it is not feasible to define complex digital models without a complete understanding of Ngspice and XSPICE architectures and is a time-consuming process. Also, most of the users are familiar with GHDL and can write the models using VHDL code with ease. Hence, NGHDL provides an interface to write VHDL code for a digital model and install it as model in Ngspice. So whenever Ngspice looks for that model, it will actually interface with VHDL code to get the result.

Fig. 9.1 shows the overview of NGHDL indicating its architecture at the abstract level. The values for the digital models present in the netlist are fetched from the GHDL side of the interface whereas the values of the analog part are fetched from Ngspice's spice3f5 engine. Digital and Analog components in Fig. 9.1 are connected to each other with the help of the hybrid ADC and DAC models provided by Ngspice. This helps in the signal level switching when simulation is performed. As analog signals

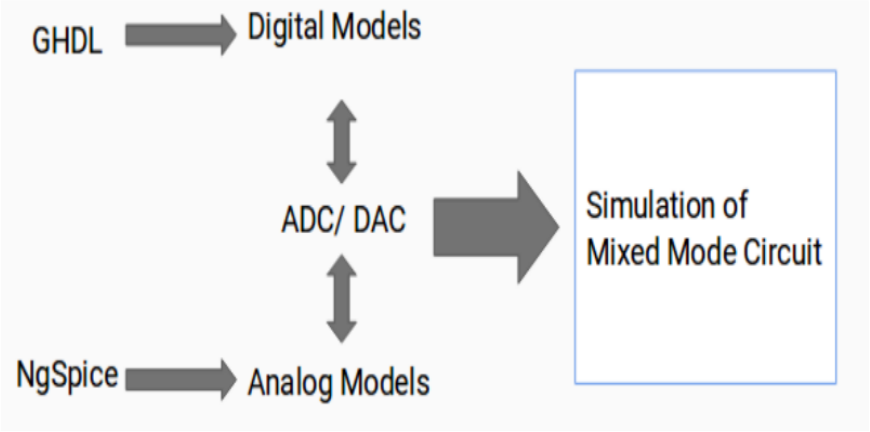


Figure 9.1: Overview of NGHDL

are in continuous time domain and Digital signals are in discrete time domain, hybrid components help bridge the gap. More information on the parameters of ADC and DAC present in Appendix : D.

9.2 Digital Model creation using NGHDL

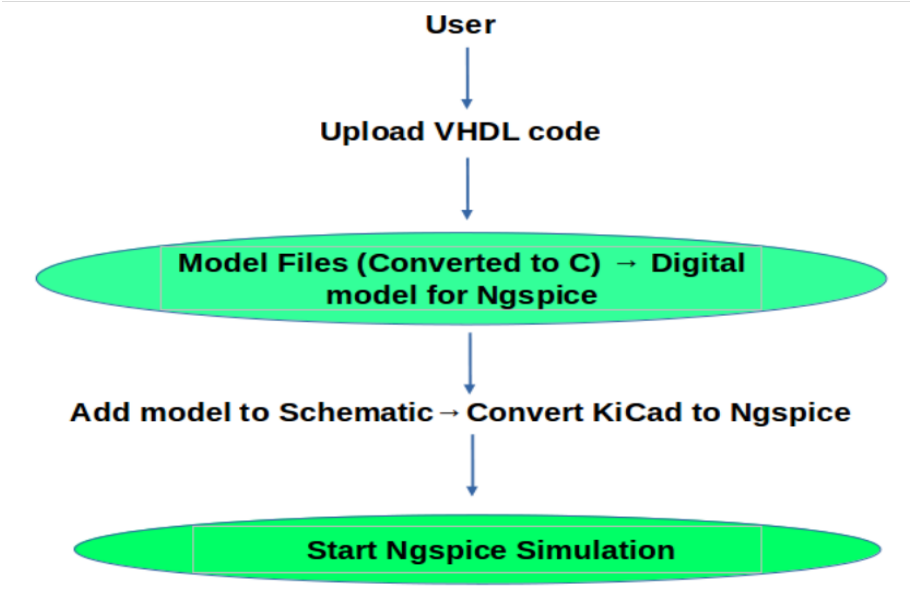


Figure 9.2: User flow for NGHDL

The steps to create digital models are given below:

1. Click on NGHDL button on left side pane of main window, the Ngspice Digital Model Creator window will appear as shown in Fig. 9.3

2. Now browse and locate the VHDL file to upload. Select the VHDL file and click on the Upload button. This process will create Ngspice model and corresponding component drawing inside the KiCad library (eSim_Nghdl.lib) of the VHDL block to be used in mixed-signal simulations. An acknowledgement message will appear upon sucessful processing of the VHDL code as shown in Fig. 9.4.

   Note : `"Add files"` option allow you to use a smaller entity / subpart / submodule to support the main VHDL file. That is, a digital model will be generated corresponding to that file that has been browsed. The file that has been `"added"` to Nghdl upload window will only be placed along with the model under model's DUTghdl folder to support the model.

   Hence, `"browsing"` one file and `"adding"` several files won't create that many number of models, but only one model will be created corresponding to the browsed file.

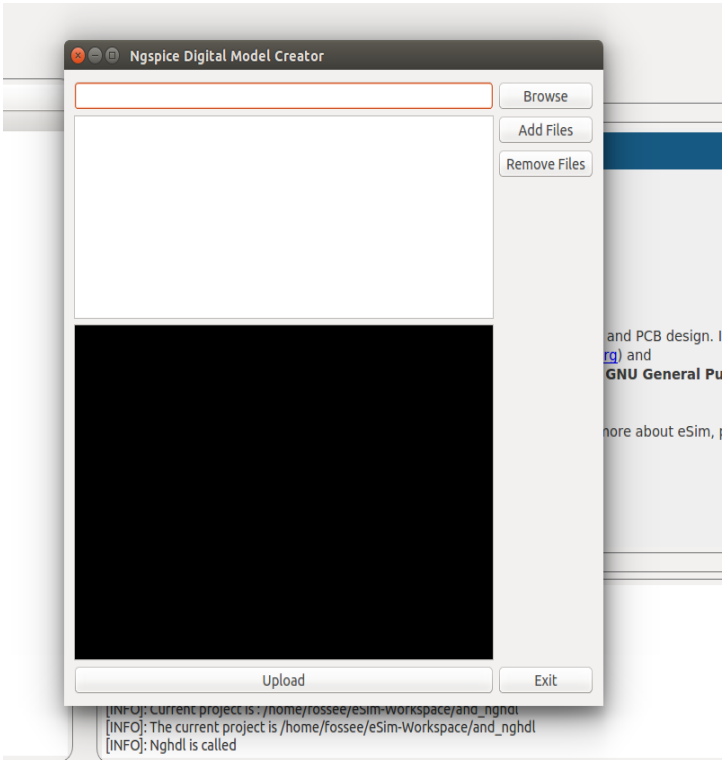


Figure 9.3: NGHDL interface

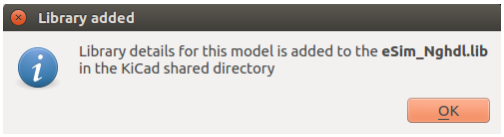


Figure 9.4: Uploading of digital model

9.3 Schematic Creation

Steps for schematic creation are as follows:

1. Click on New Project icon to create a new project as shown in Fig. 9.5, be careful of the naming conventions.

2. After successful upload of the model using the VHDL code, you can create the schematic of your design by clicking on `Open Schematic` button on the left pane of the eSim window. Then go to `Preferences` option on top of the schematic editor window and click on `Component Libraries` to add the library eSim_Nghdl.lib in KiCad. Following window will appear as shown in Fig. 9.6, where you will have to

Figure 9.5: Creation of a new project

click on *Add* button and select the `eSim_Nghdl` library. Refer Fig. 9.6 and Fig. 9.7.

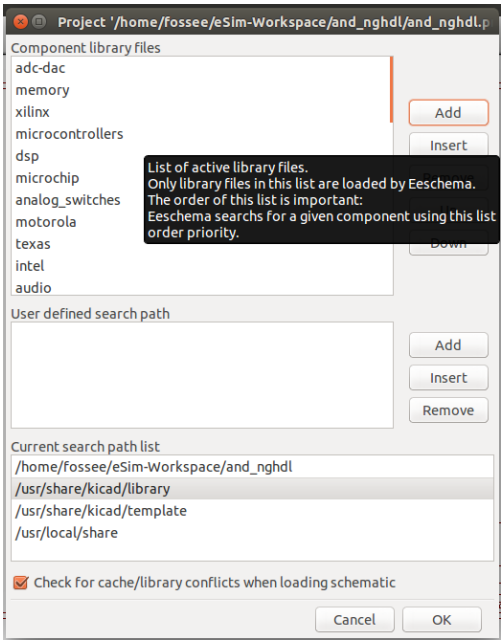


Figure 9.6: Adding the digital model library in KiCad

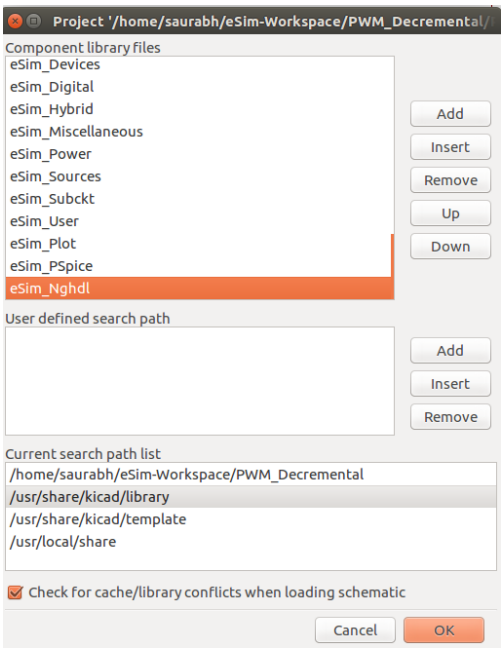


Figure 9.7: Selection of library

3. Next step is to locate the component in `eSim_Nghdl` library as shown in Fig. 9.8 and place it on the schematic editor as shown in Fig. 9.9.

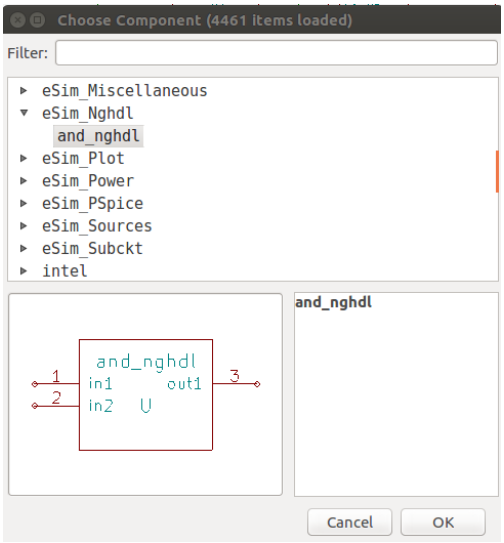


Figure 9.8: Locating the component in library

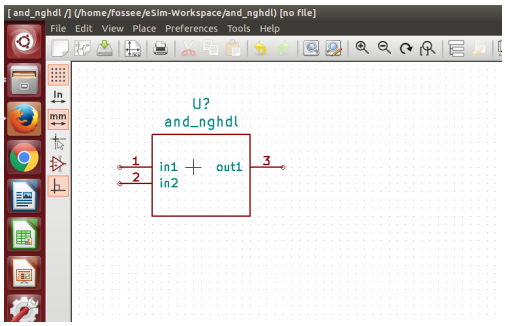


Figure 9.9: Placement of component on editor

4. Now create the schematic as shown in Fig. 9.10, annotate, perform ERC, create the netlist and save the schematic by following the steps given in Chapter 5.

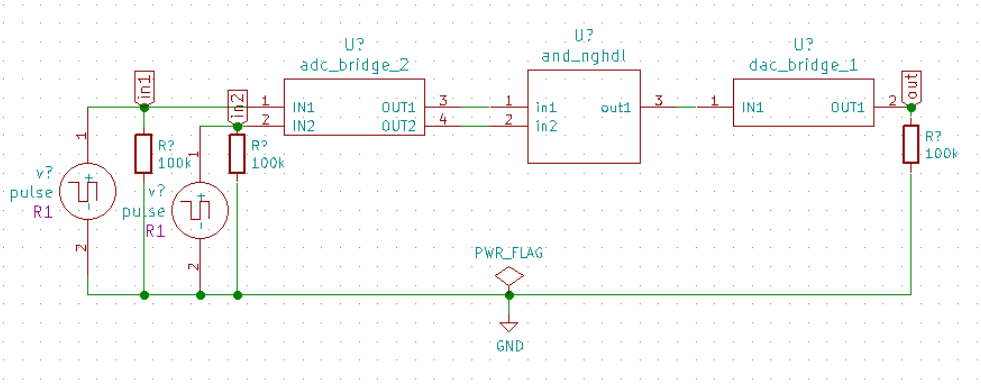


Figure 9.10: Example of an AND gate characteristics circuit

5. After creating the schematic, click on `KiCad-to-Ngspice converter` and select the type of analysis as transient as shown in Fig. 9.11 and set the start, step and stop time as shown in Fig. 9.12

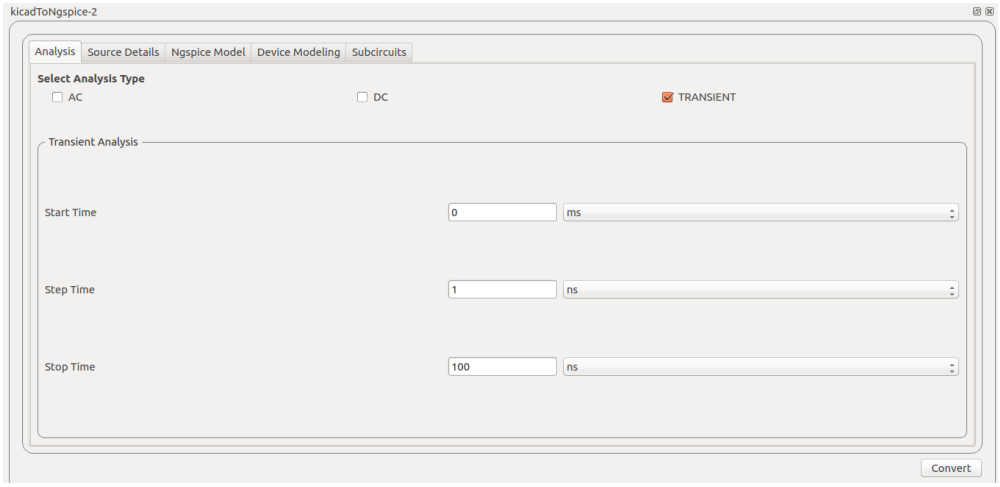


Figure 9.11: Analysis Part I

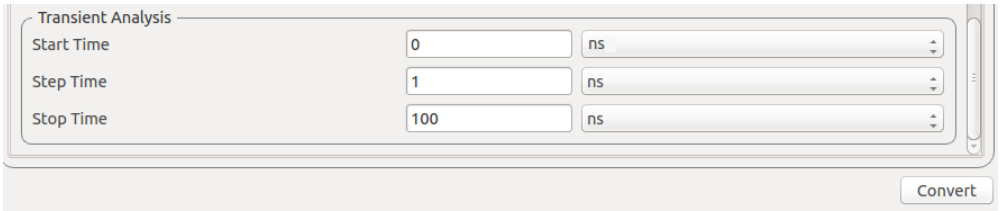


Figure 9.12: Analysis Part II

6. Now click on `Source Details` and enter the values for Source v1 and source v2 as shown in figure Fig. 9.13 and Fig. 9.14

7. Now select the option `Ngspice Model`, window as shown in Fig. 9.15 will appear. The values of the parameters listed can be changed per user's requirement. If you have used any semicnductor devices and Subcircuits in your design, then please specify the Spice models and subcircuits in the `Device Modeling` and `Subcircuits` tabs of the `KiCad-to-Ngspice converter` window. After that click on `Convert` button. This step will create the simulation compatible netlist.

8. Now click on `Simulation` button, it will display the following windows as shown in Fig. 9.16. This is the Ngspice terminal and Python plot window.

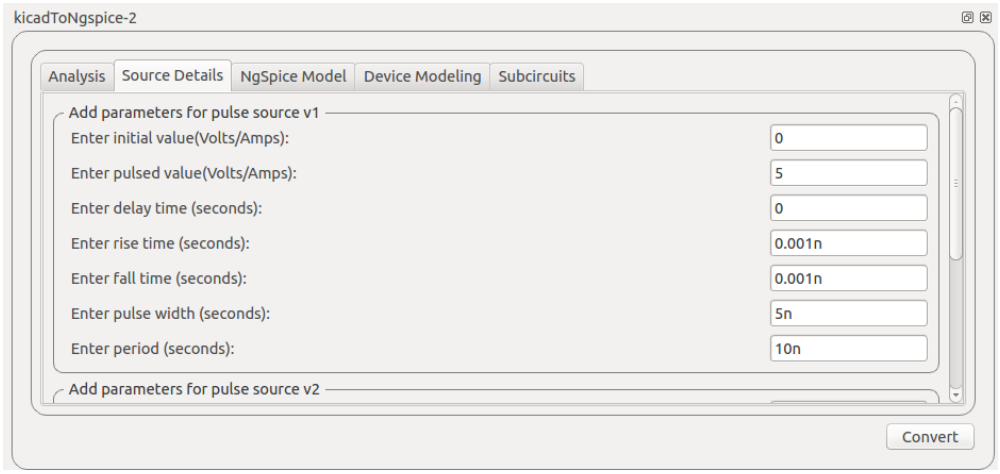


Figure 9.13: Value of Source v1

Figure 9.14: Value of Source v2

9. Now select the required nodes and click on `Plot` button. You can see the plots of input source v1, input source v2 and output as shown in Fig. 9.17, Fig. 9.18, and Fig. 9.19 respectively.

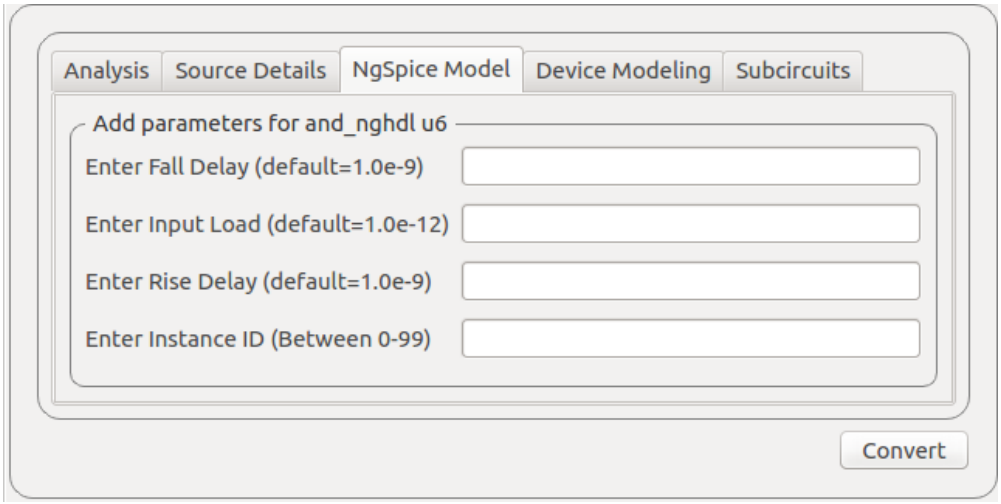


Figure 9.15: Model Parameters

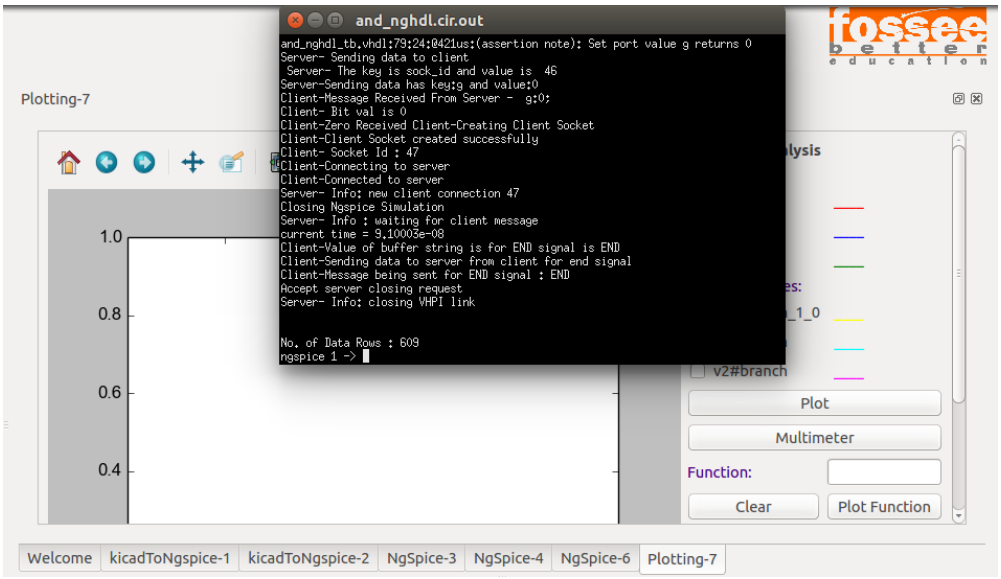


Figure 9.16: Simulation window

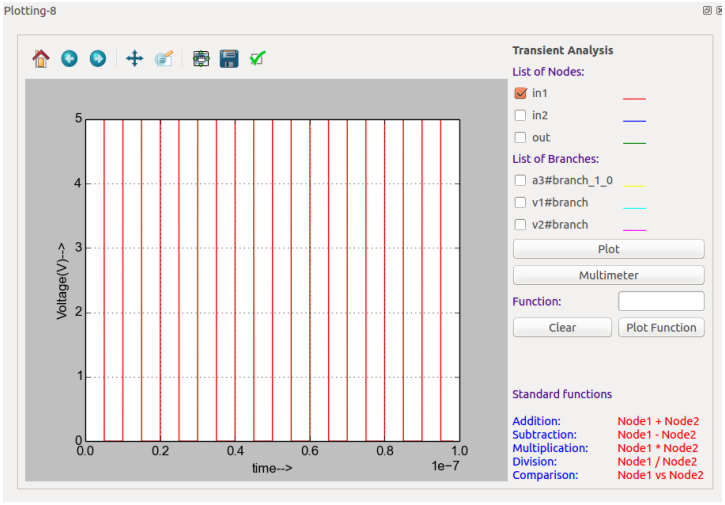


Figure 9.17: Plot of Source V1

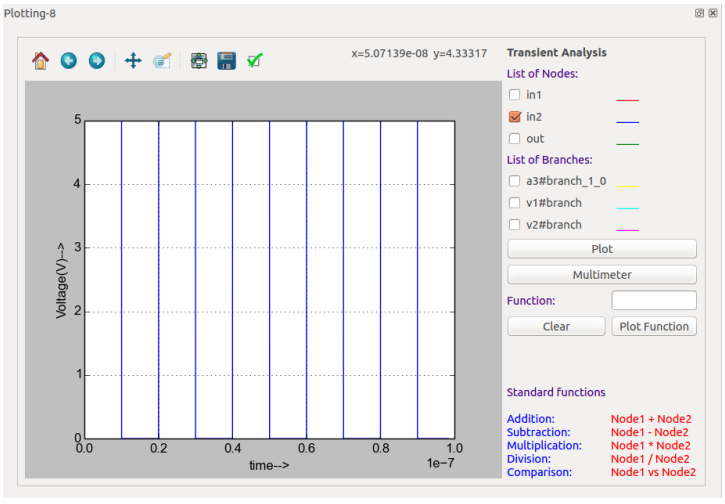


Figure 9.18: Plot of source V2

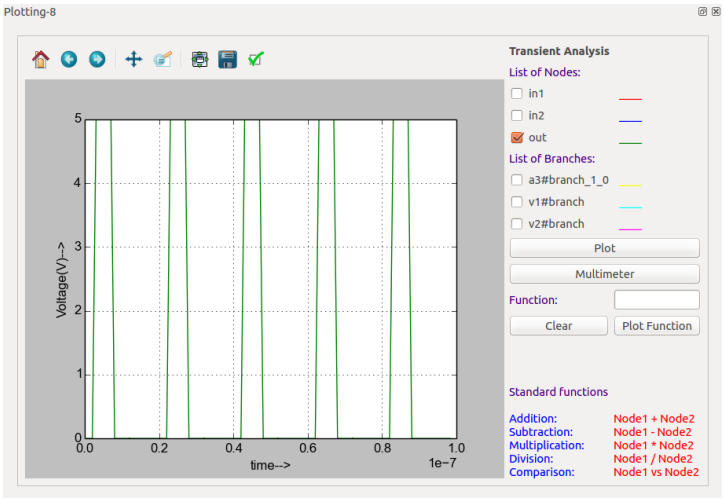


Figure 9.19: Plot of output

Chapter 10

Makerchip-NgVeri: Mixed Signal Simulation

NgVeri is a simulator in eSim which facilitates mixed-signal circuit simulation. Digital models are simulated using Verilator and analog models are simulated using XSPICE engine of Ngspice. NgVeri links Ngspice and Verilator to support mixed-mode simulations in eSim as shown in Fig. 10.1.

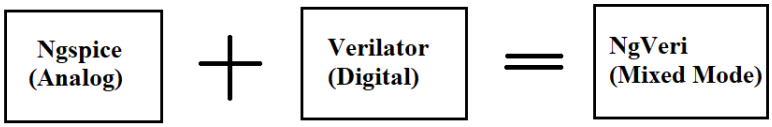


Figure 10.1: NgVeri Block

Makerchip is a web browser IDE to develop/simulate/debug Verilog, SystemVerilog and TL-Verilog Code developed by Redwood EDA, LLC. It provides seamless design experience by integrating code, debug window, block diagrams, waveforms together in one screen. Makerchip is integrated with NgVeri in the latest version of eSim (eSim 2.2). Makerchip provides the digital side of eSim's mixed-signal environment.

10.1 Familiarizing the Makerchip-NgVeri interface in eSim

In this section, we will explain Makerchip-NgVeri interface in eSim and the various menus and tabs.

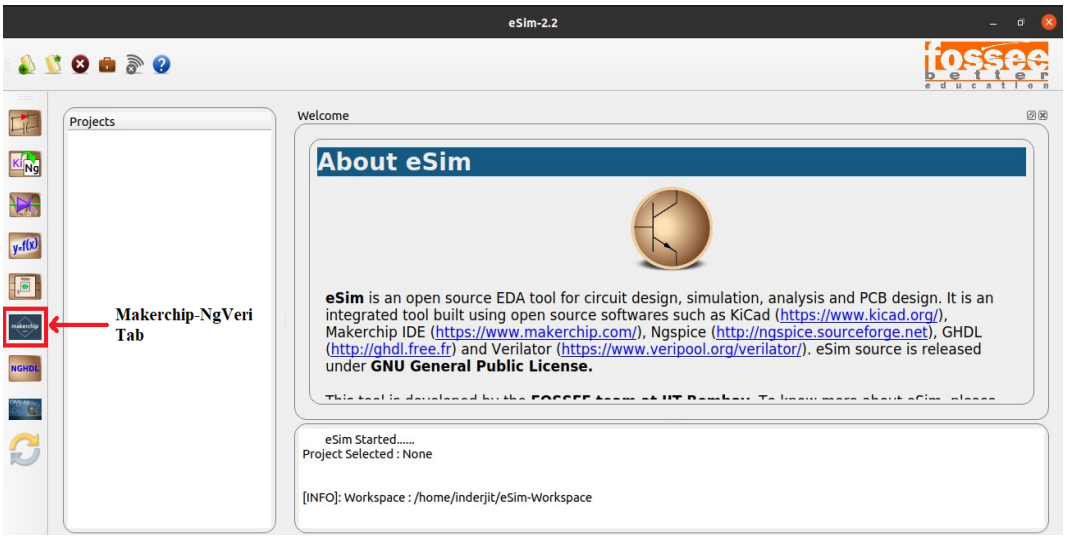


Figure 10.2: Makerchip NgVeri Tab

Fig. 10.2 shows the Makerchip-NgVeri tab (newly added in eSim 2.2). In order to open this tab click on the Makerchip button on LHS vertical bar. Makerchip interface opens up as shown in the Fig. 10.3.

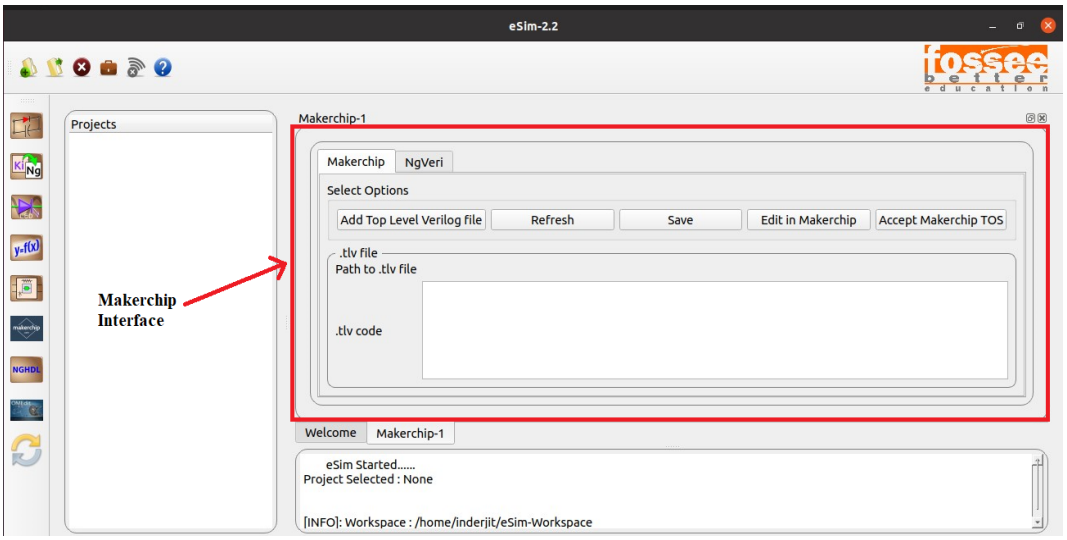


Figure 10.3: Makerchip Interface

10.2 Makerchip Interface

Makerchip interface working space is shown in the Fig. 10.3. Various buttons of Makerchip interface are shown in the Fig. 10.4

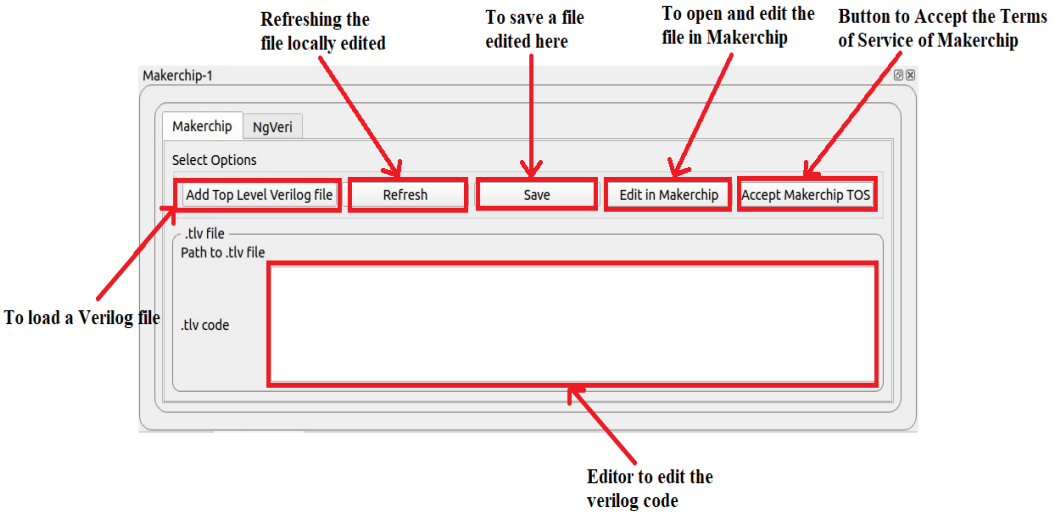


Figure 10.4: Makerchip buttons

The components of the Makerchip tab are:

1. Add Top Level Verilog file: This button helps the user to load the verilog file.
2. Refresh: It enables refreshing of the file locally edited.
3. Save: It saves the edited file.
4. Edit in Makerchip: It opens the Makerchip App and the user can edit and simulate the verilog file in Makerchip IDE.
5. .tlv Code Editor: After loading the top level verilog file, the verilog code appears in this editor. The user can then edit the verilog code within this editor workspace.
6. Path to .tlv file: The path or the directory where the verilog file exists locally appears here after uploading the verilog file.
7. Accept Makerchip TOS: This button when pressed accepts the Terms of Service of Makerchip. After accepting this, the button vanishes in the rerun of eSim.

10.3 NgVeri Interface

NgVeri interface is shown in the Fig. 10.5. Various components of NgVeri interface are shown in the Fig. 10.6

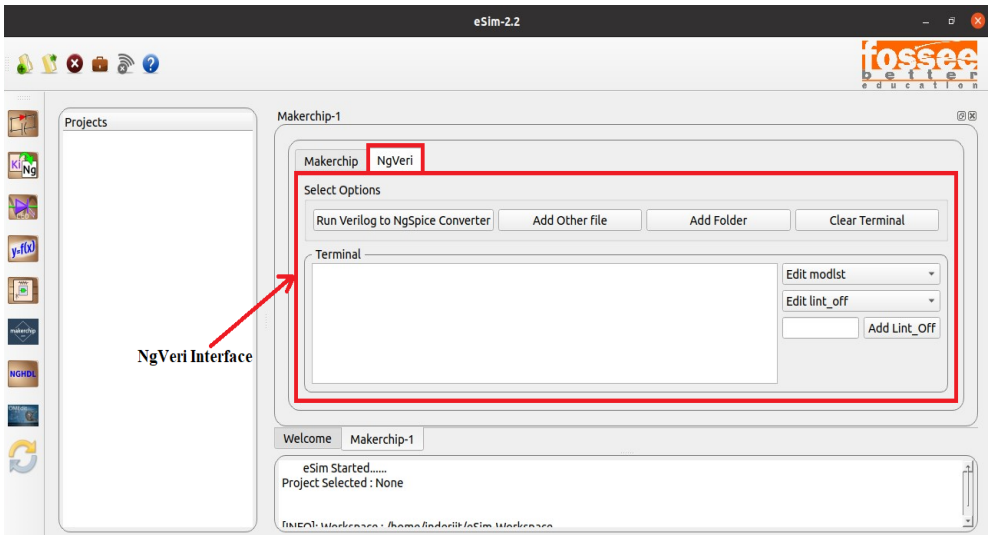


Figure 10.5: NgVeri Interface

The important components of NgVeri are:

1. Terminal : This is the terminal where the user can view all the commands and processes running.
2. Run Verilog to Ngspice Converter: This button when pressed run NgVeri and builds the model for Ngspice.
3. Add Other file: Using this option, the user can add all the dependency files which are needed by the Top level verilog file in Makerchip.
4. Add Folder: Using this option, the user can add all the dependency folders which are needed by the Top level verilog file.
5. Clear terminal: This button when pressed erases the content of the terminal i.e it clears the terminal.
6. Edit modlst: Using this option, the user can see the existing models present and the user can remove the models from the Model list.

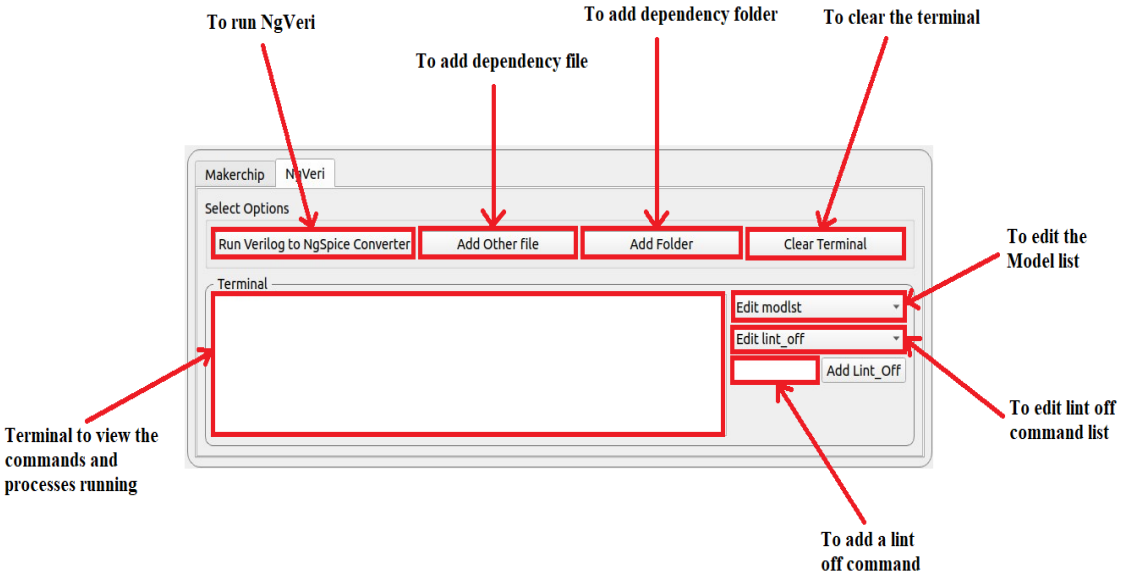


Figure 10.6: NgVeri buttons

7. Edit lint off: When this button is pressed, the user can see all the lint-off commands list and the user can remove the lint-off from the list.

8. Add Lint off: The user can add lint-off command by using this button. This lint-off command is used if user gets a lint-off error in the terminal.

10.4 Counter example using Makerchip and NgVeri in eSim

In this section, a digital 8 bit counter example using NgVeri and Makerchip tabs in eSim is discussed. Kindly note that steps shown over here are carried out with eSim 2.2 in Ubuntu version 20.04. Steps remain same for eSim in Windows version.

10.4.1 Makerchip steps in eSim

Steps for verilog code compilation and verification using Makerchip IDE are as follows:

1. Open eSim and select the default eSim workspace by clicking on **OK** as shown in Fig. 10.7. Its format is /home/<username>/eSim-Workspace. User-name can be user specific but be careful of the naming conventions(space is not allowed). If the user wants to select they can chose the same by using the **Browse** button in the dialog box.

Figure 10.7: eSim workspace

2. Open a blank Text Editor and write verilog code of 8 bit counter and save the file as either filename.v in the eSim-Workspace as shown in Fig. 10.8.

3. **Please Note**: The filename should be the same as top-level module name, otherwise eSim will throw an error as discussed in Common error sections.
   The file extensions allowed are:

   - .v for Verilog
   - .sv for SystemVerilog
   - .tlv for Transaction-Level Verilog

Figure 10.8: Saving verilog file in eSim workspace

4. The verilog code for 8 bit counter is written inside the Text Editor with .v file as extension saved in eSim workspace is shown in Fig. 10.9

Figure 10.9: 8 bit Counter verilog code

5. Click on Makerchip-NgVeri button on the left toolbar of eSim window which opens Makerchip interface in eSim as shown in Fig. 10.10.

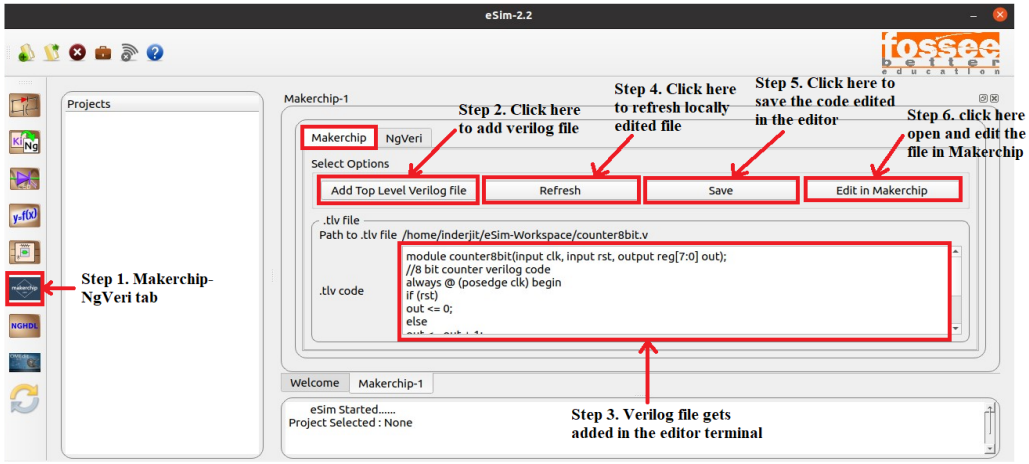


Figure 10.10: Adding verilog file in Makerchip editor

6. Click on the button shown in Step 2 as shown in Fig. 10.10 to add the verilog file for 8 bit counter. The path to top level verilog file along with the verilog code get added in the Makerchip editor terminal shown in Step 3 of Fig. 10.10

7. User can refresh locally edited verilog file by clicking on the **Refresh** button as shown in Step 4 of Fig. 10.10. If the file is edited using some other editor and also loaded in Makerchip Tab at the same time, the **Refresh** button starts toggling. The user must click on the Refresh button to get the new version of the file in the editor.

8. User can save the verilog code edited in the Makerchip editor terminal by clicking on the **Save** button shown in Step 5 of Fig. 10.10

9. For the first run, the user needs to accept the Term of Service by clicking on **Accept Makerchip TOS** as shown in Fig. 10.11. This button will disappear after rerun of eSim as shown in Fig. 10.10.

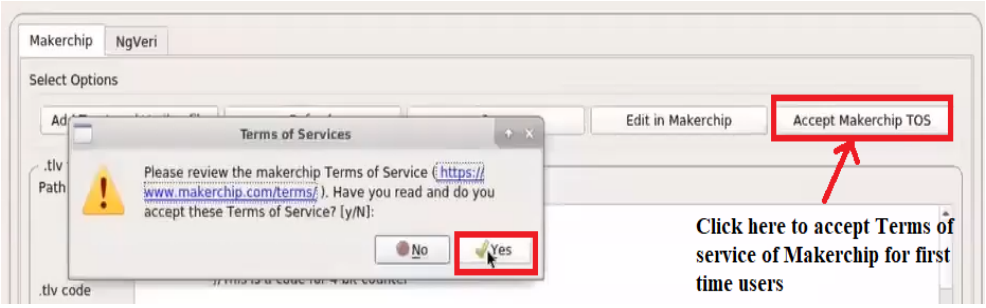


Figure 10.11: Accepting Terms of Service of Makerchip

10. Now click on **Edit in Makerchip** button to open and edit the verilog code in Makerchip IDE. A pop-up window will appear as shown in Fig. 10.12. By clicking on **Yes**, a top level verilog file i.e .tlv file will be created in the same directory of current verilog file and the Makerchip IDE will be run in a new web browser as shown in Fig. 10.13. By clicking on **No**, the current raw verilog file i.e .v file will open up in Makerchip IDE web browser.

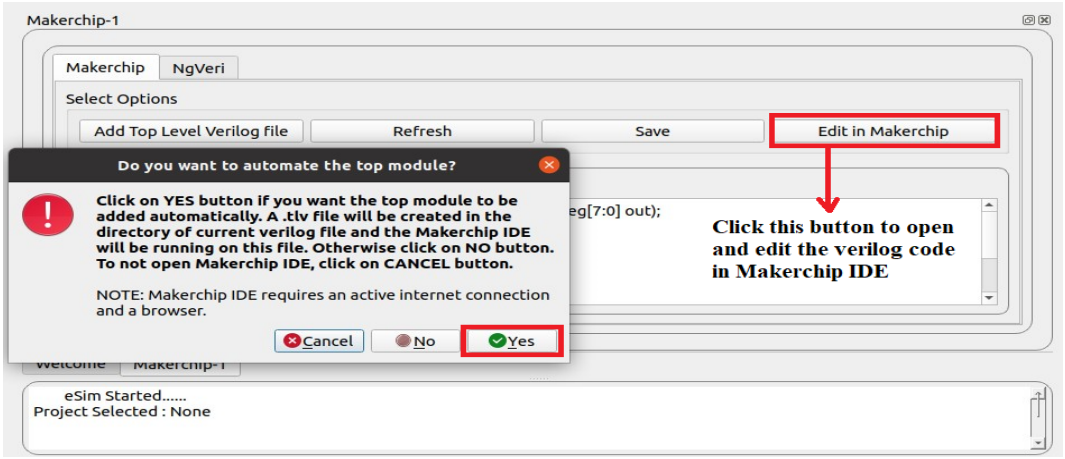


Figure 10.12: Opening Makerchip IDE

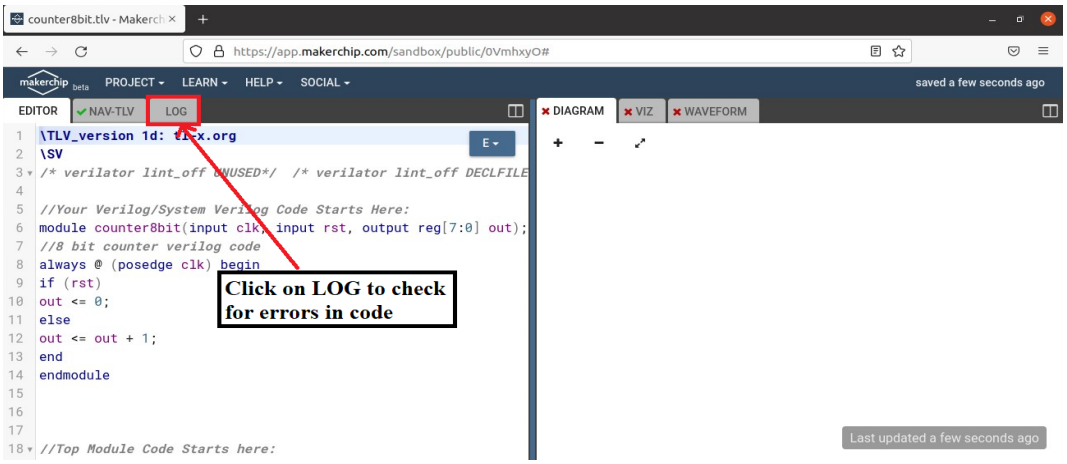


Figure 10.13: Makerchip IDE interface

11. User can check for errors by clicking on **LOG** tab in Makerchip IDE interface as shown in the Fig. 10.13. Following window will appear as shown in Fig. 10.14, where you can check the Verilator lint-off errors.

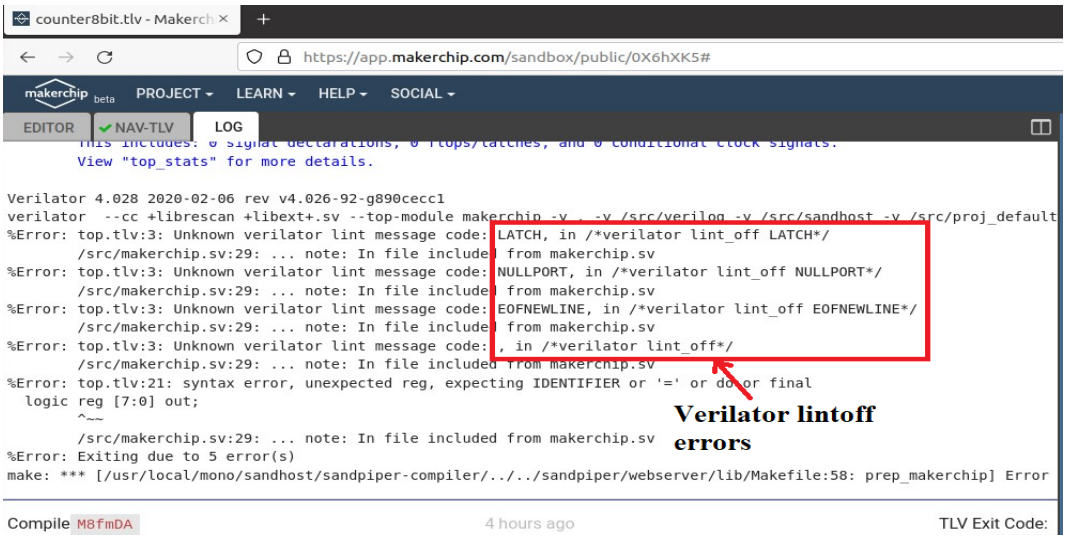


Figure 10.14: Verilator lintoff errors

12. As shown in the Fig. 10.14, verilator lint-off commands which caused error in the Log section needs to be removed from Line 3 in the editor window. Two such commands which needs to be removed are shown in the Fig. 10.15.

Figure 10.15: Removal of lintoff errors

13. Next, click on the **Editor** tab in the editor as shown in Fig. 10.16, then click on **Compile/Sim** button to compile the verilog code.

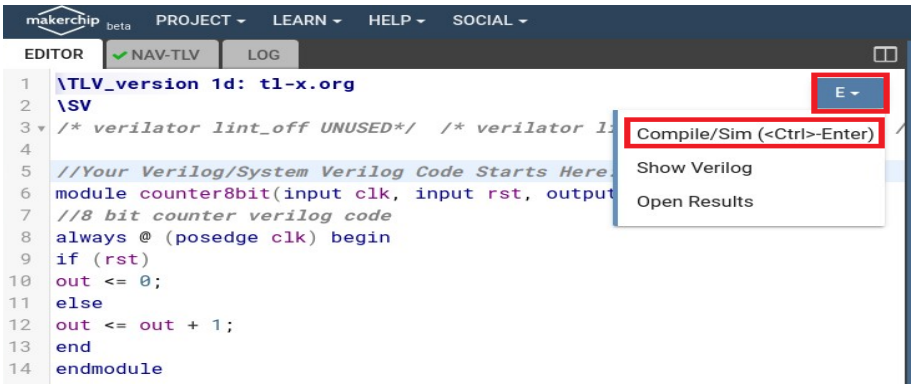


Figure 10.16: Compiling after removing lintoff errors

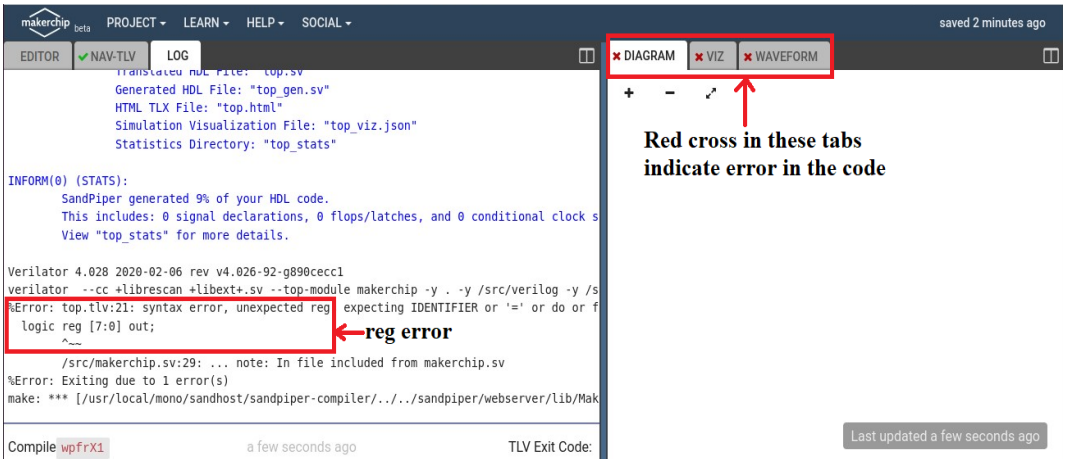


Figure 10.17: Unexpected reg error in the code

14. After compiling, **Unexpected reg error** remains in the Log section (Refer Fig. 10.17) which can be rectified by removing *reg* from top level verilog code as shown in Fig. 10.18. By clicking **Yes** button in Fig. 10.12, a Top Level Verilog (TLV) code automatically gets added in the editor. Refer Fig. 10.18. The value of *rst* value is by default as random value and it can be replaced if user wants to assign values. We have assigned *rst* with *reset* in this counter example.

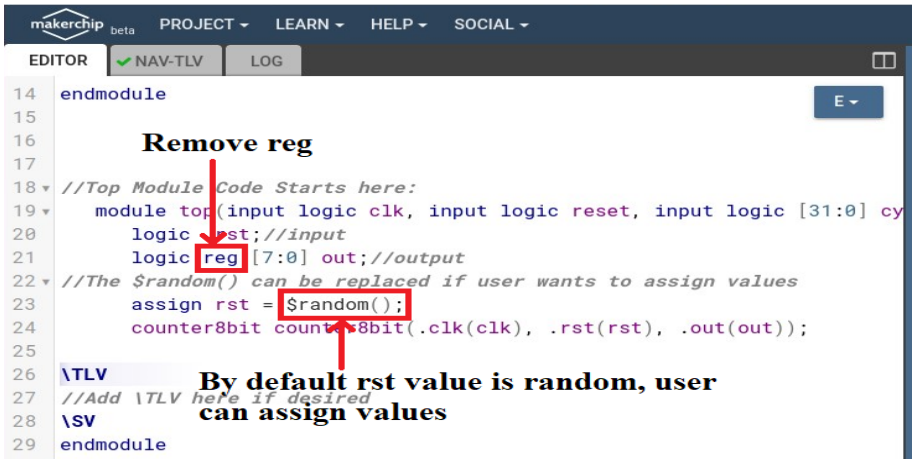


Figure 10.18: Top level verilog code

15. Now click on the **Editor** tab and then click on **Compile/Sim** button in the editor as shown in Fig. 10.16, it will display the following waveform window as shown in Fig. 10.19. This means that the verilog code of 8 bit Counter is successfully

compiled. Kindly, note that in Fig. 10.19, inputs and output waveform are in the compressed form.

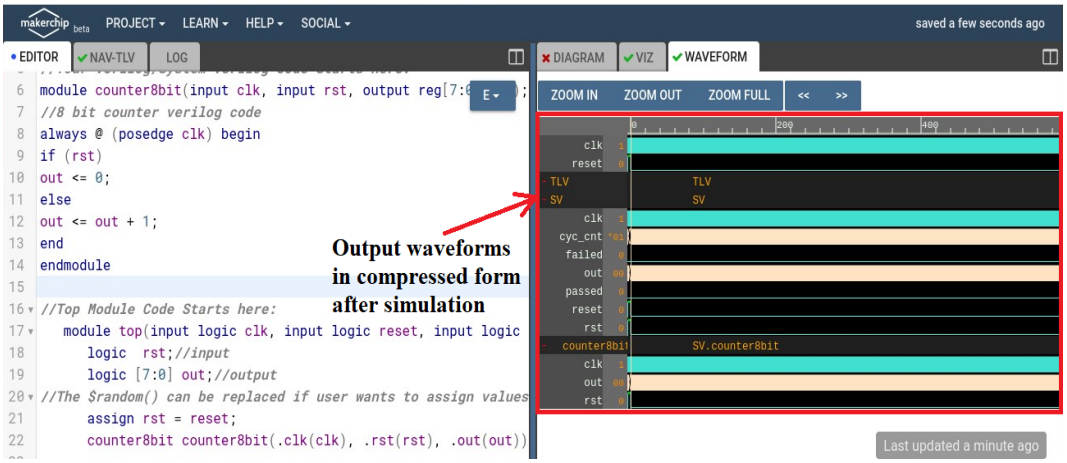


Figure 10.19: Output waveforms in compressed form of 8 bit Counter

16. Fig. 10.20 shows the expanded view of inputs and output waveforms of 8 bit counter. The waveforms in Fig. 10.19 can be expanded by clicking on **Zoom In** button as shown in Fig. 10.20.

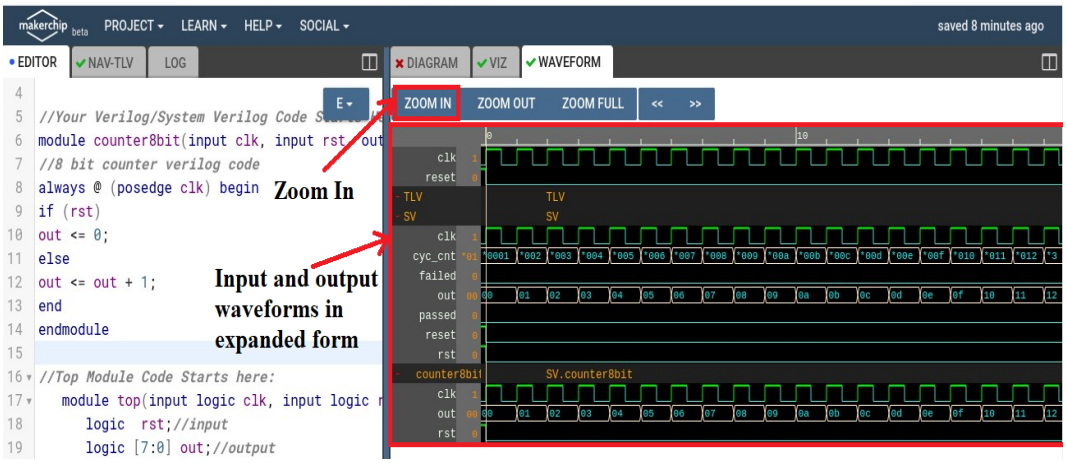


Figure 10.20: Output waveforms in expanded form of 8 bit Counter

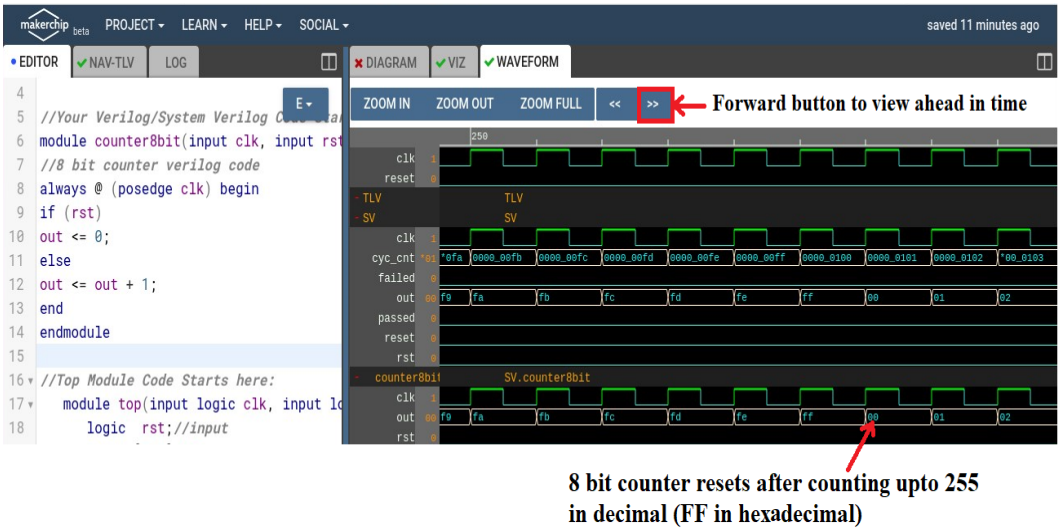


Figure 10.21: 8 bit Counter counting from 0 to 255

17. To view the output waveform forward in time, click on forward button as shown in Fig. 10.21. It can be clearly seen, the 8 bit counter counts from to 0 to 255 in decimal (FF in hexadecimal) and it resets after counting up to 255.

18. Next, close the Makerchip window. This completes the 8 bit counter verilog code compilation and verification process using Makerchip IDE.

10.4.2 Steps to run NgVeri in eSim

Steps for schematic creation and simulation using NgVeri in eSim are as follows:

1. The verilog code named counter8bit.v is loaded in editor space of Makerchip tab as shown in Fig. 10.10.

2. Click on **NgVeri** button and then click on **Run Verilog to Ngspice converter** button as shown in Fig. 10.22. This step will convert 8 bit counter's verilog file to Ngspice model. The commands that run in the NgVeri terminal leads to creation of Ngspice(XSPICE) model of 8 bit counter are mentioned in following steps.

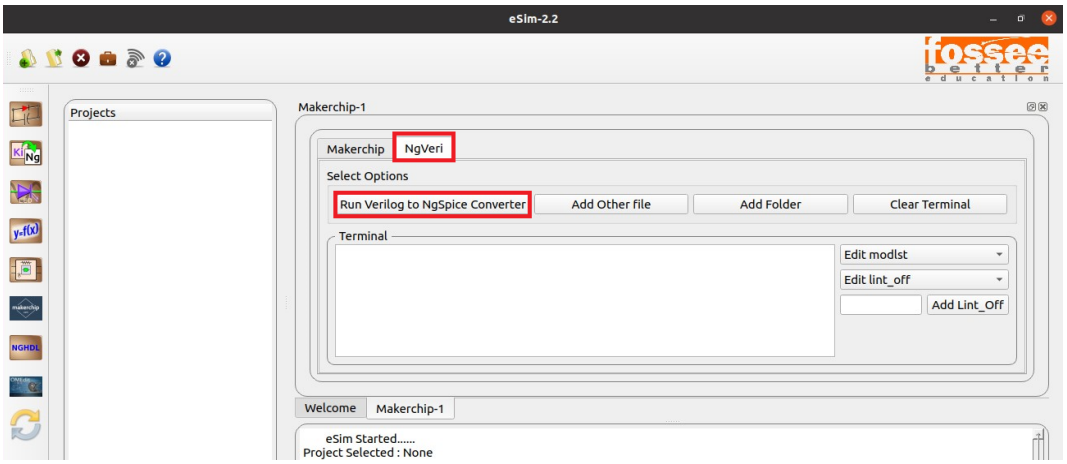


Figure 10.22: NgVeri interface buttons

3. First command after clicking **Run Verilog to Ngspice Converter** button in NgVeri terminal is **RUN VERILATOR** highlighted as 1 in red color in Fig. 10.23.

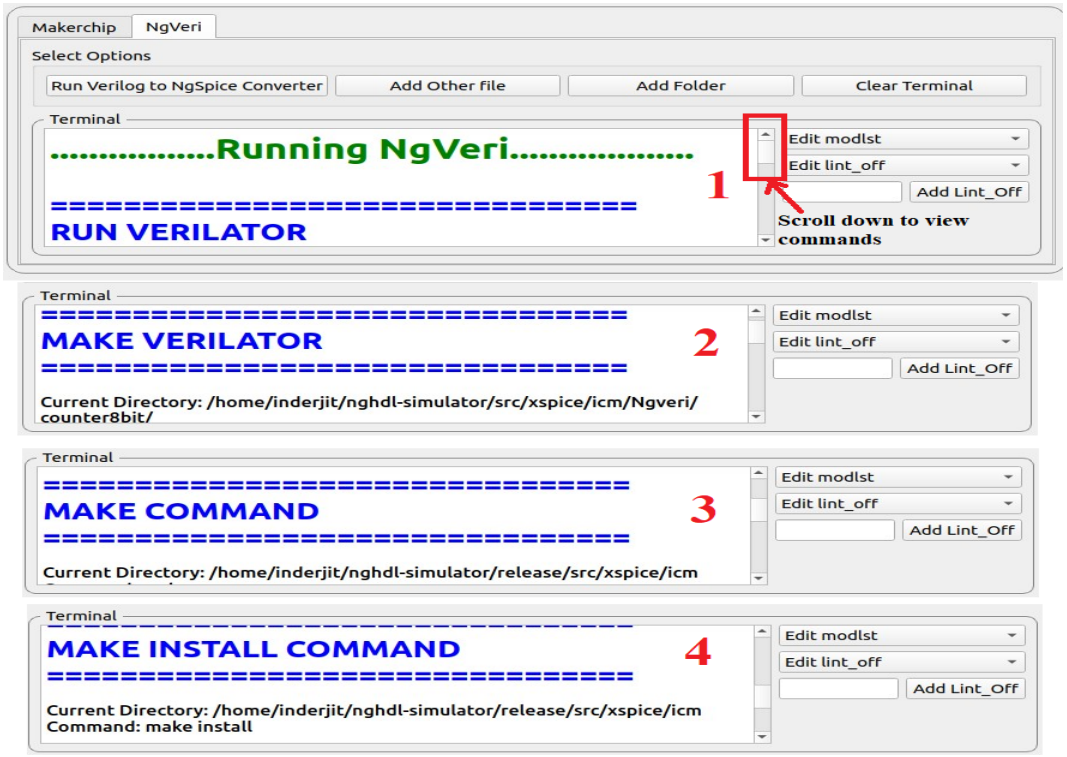


Figure 10.23: Commands in NgVeri

4. Verilator converts the verilog file into C++ objects. Next command is **MAKE VERILATOR** for obtaining the C++ objects. Next is **MAKE COMMAND**, which links the verilator's C++ objects and Ngspice objects. **MAKE INSTALL COMMAND** is the last command in the process before Ngspice model creation.

5. When the **Model Created Successfully** message appears in the terminal as shown in green color in Fig. 10.24, it indicates that the Ngspice model is created.

6. An optional command **RUN SANDPIPER-SAAS** gets run Transaction-Level Verilog Code(.tlv) before the **RUN VERILATOR** command. This command in run to convert the Transaction-Level Verilog Code to the SystemVerilog Code which is easily accessible to the verilator.

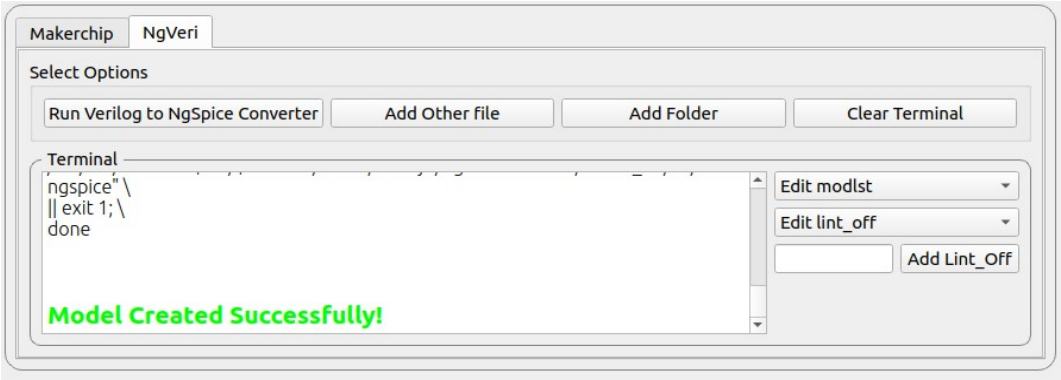


Figure 10.24: NgVeri Model created

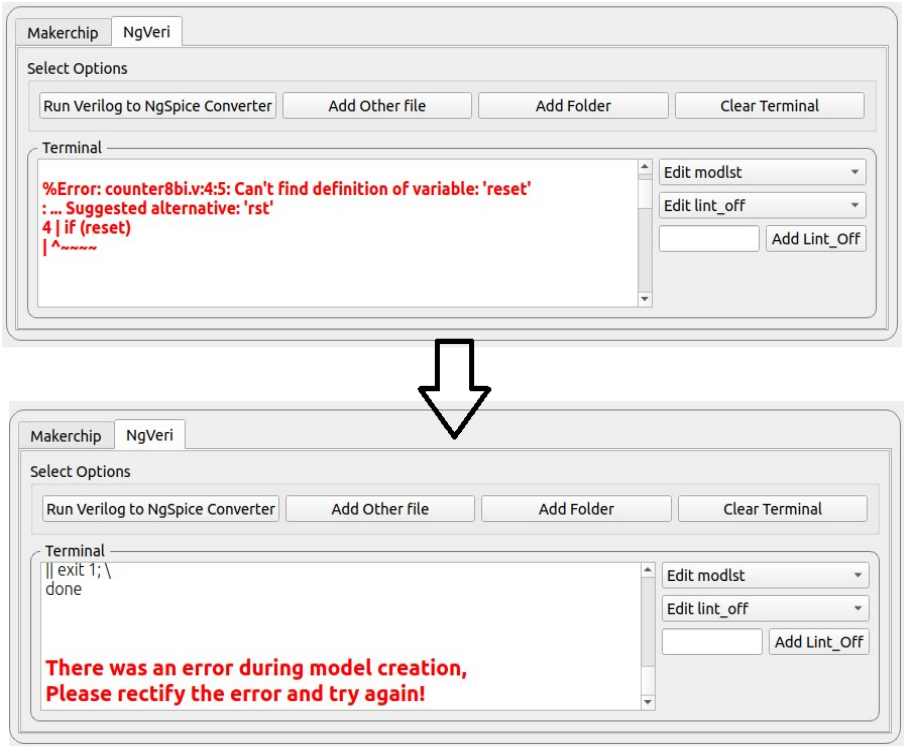


Figure 10.25: NgVeri Model creation error

7. In case of any error encountered in NgVeri terminal after running Verilog to Ngspice converter, then those errors will be displayed in the terminal in Red color in any of the commands discussed above. Debug the errors as per the messages

displayed in the terminal. In Fig. 10.25, the error message is displayed in Red color in NgVeri terminal.

10.4.3 Creating of Schematic of 8 bit counter

In this section, we will create a new project in eSim, create the schematic and run the simulation for 8 bit counter. Steps for schematic creation are as follows:

1. Click on **New Project** icon to create a new project as shown in Fig. 10.26, be careful of the naming conventions.

2. After successful creation of NgVeri model using the Verilog code, you can create the schematic of your design by clicking on **Open Schematic** button on the left pane of the eSim window as shown in Fig. 10.27. A confirmation pop-up window appears. Click on **Yes** to create a new schematic. A blank schematic created is shown in Fig. 10.28

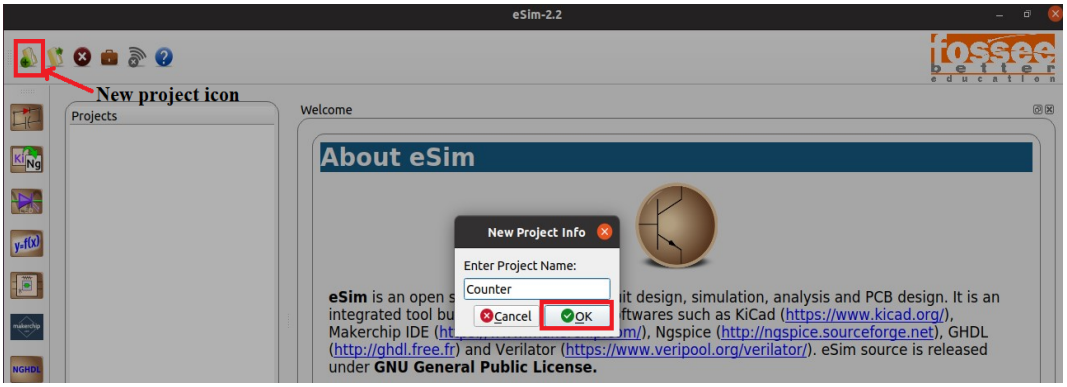


Figure 10.26: Creation of new project in eSim

3. Next step is to click on **Place component** icon to place the components in the schematic editor as shown in Fig. 10.28.

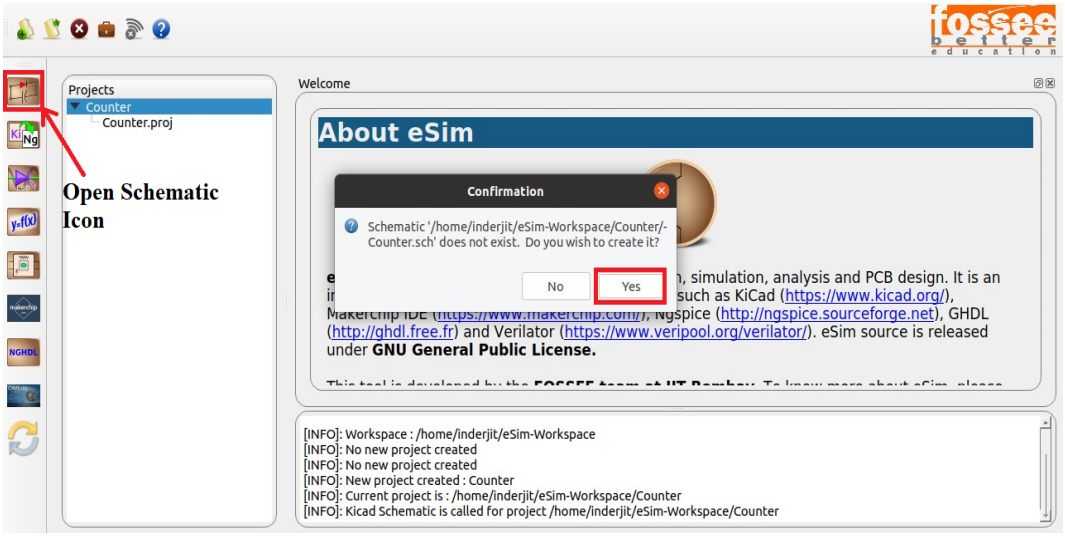


Figure 10.27: Creation of new schematic in eSim

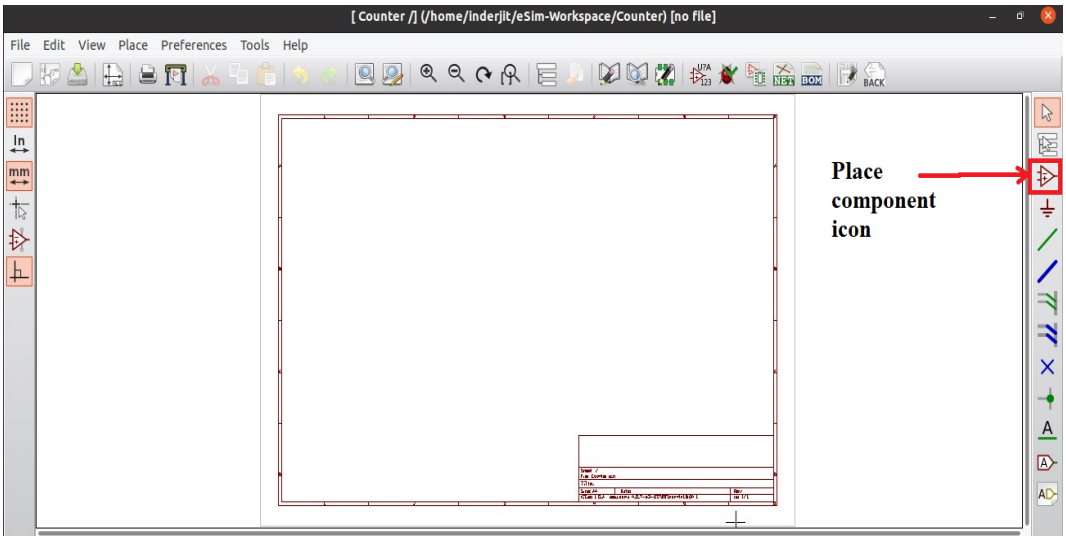


Figure 10.28: Blank schematic created in eSim

4. To locate the components in eSim library, click on **Place component** icon as shown in Fig. 10.28. A pop-up window named **Choose Component** appears. User can type component name in the space as shown in Fig. 10.29.

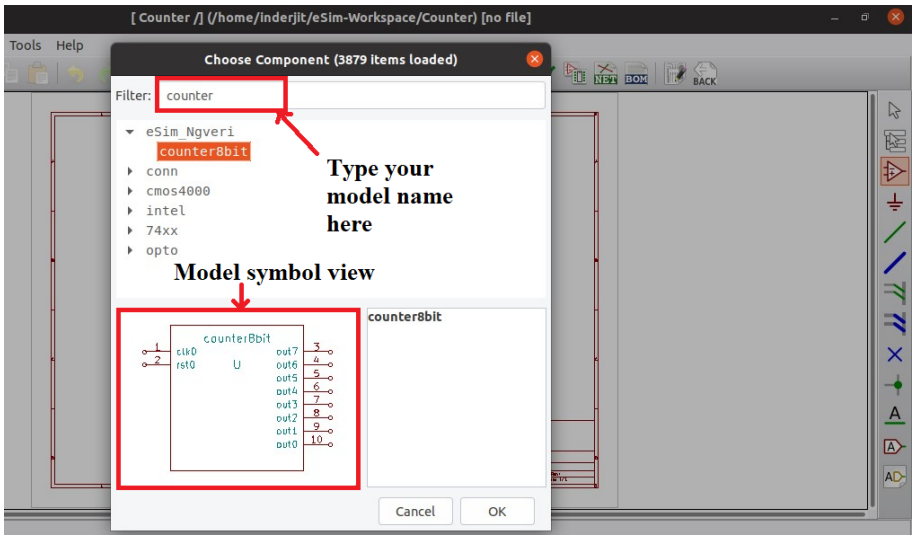


Figure 10.29: Locating the model in library

5. In this example, counter8bit model is created under eSim NgVeri library. The model view is visible with two input pins reset and clock and 8 output pins from out0 to out7 as shown in Fig. 10.29. Click on **OK** to place the component on the schematic editor as shown in Fig. 10.30.

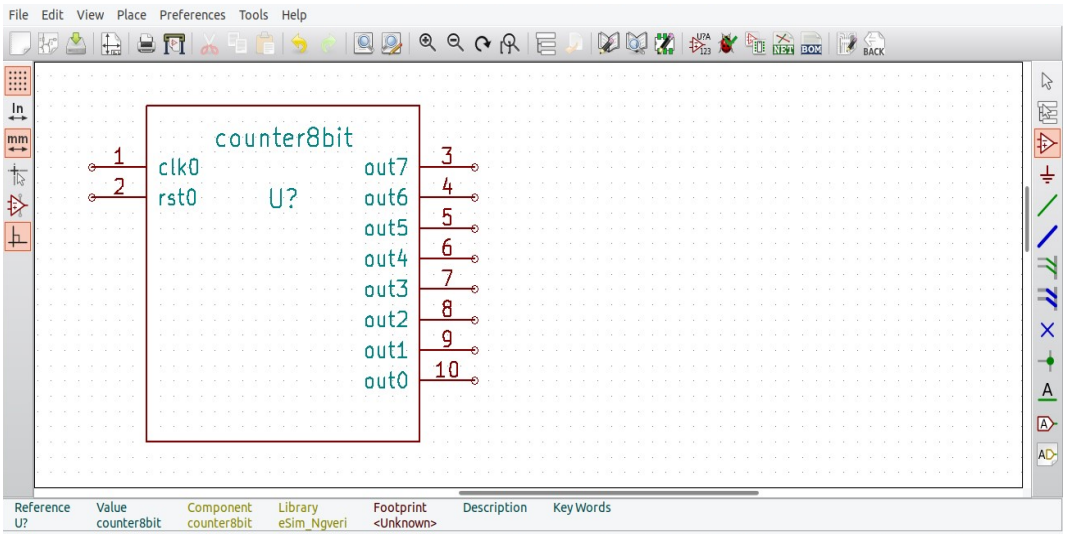


Figure 10.30: Placement of component in Schematic Editor

6. Now create the schematic as shown in Fig. 10.31, annotate, perform electrical rules

check (ERC), create the netlist and save the schematic by following the steps given in Chapter 5.

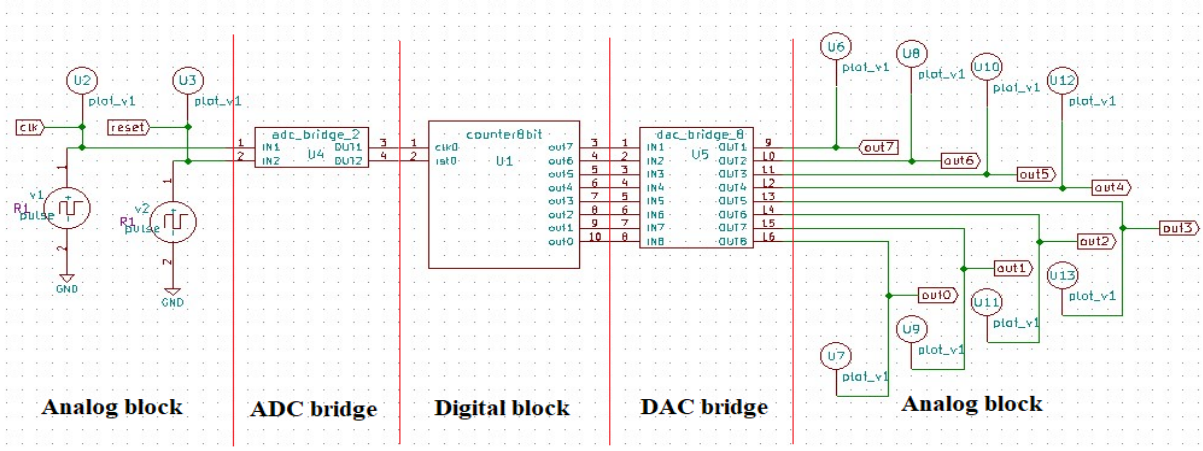


Figure 10.31: Example of a 8 bit Counter circuit in eSim

10.4.4 Ngspice Simulation of 8 bit counter

In this section, we will run the simulation and plot input-output waveforms for 8 bit counter. Steps for Ngspice simulation and plotting results are as follows:

1. After creating the schematic, click on **KiCad to Ngspice Converter** icon and select the type of analysis as transient and set the start, step and stop time as shown in Fig. 10.32

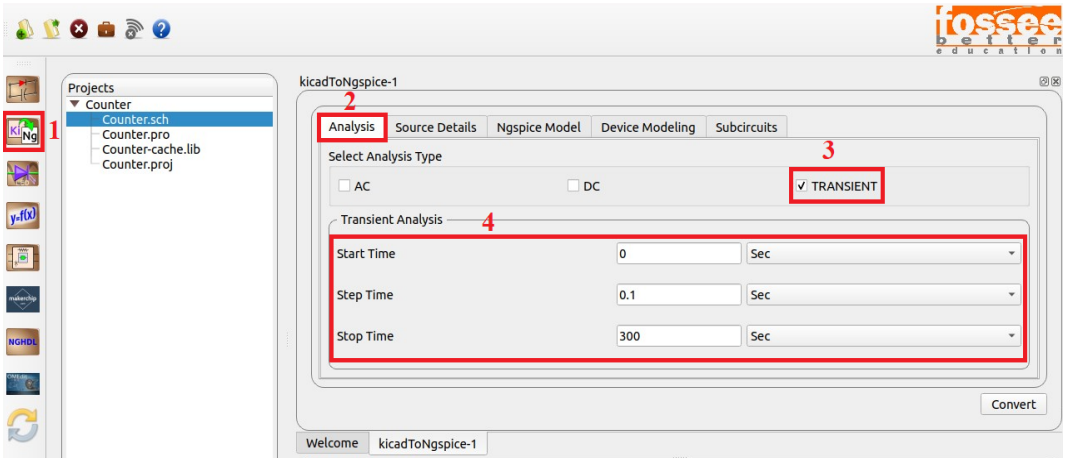


Figure 10.32: KiCad to Ngspice conversion steps

2. Now click on Source Details and enter the values for Source v1 and Source v2 as shown in Fig. 10.33 and Fig. 10.34

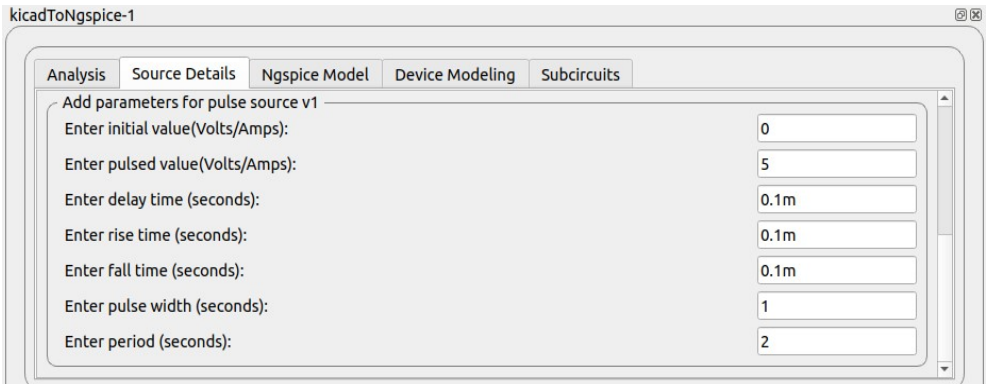


Figure 10.33: Values for Source v1

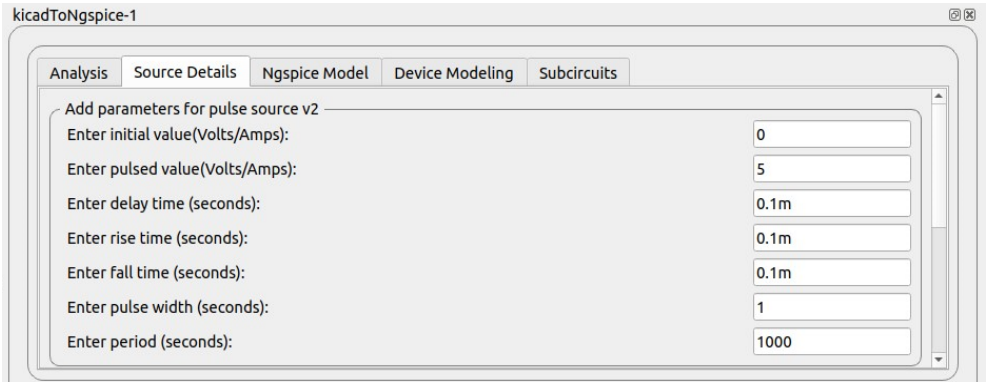


Figure 10.34: Values for Source v2

3. In this counter example, no device modeling and sub-circuit build up is required. Now click on the **Convert** button as shown in Fig. 10.35. This will convert KiCad schematic into Ngspice code i.e it creates the simulation compatible netlist.

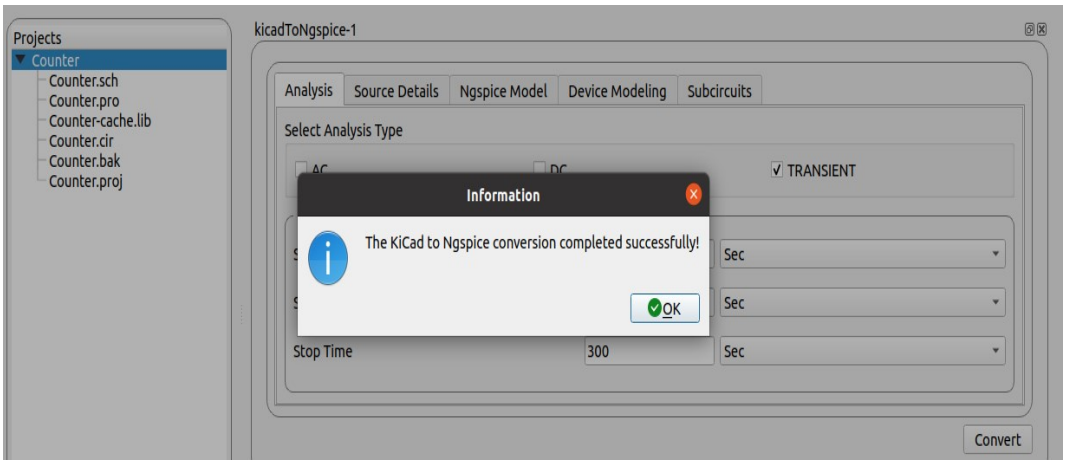


Figure 10.35: KiCad to Ngspice conversion

4. Next, right click on **Counter** and click on **Refresh** as shown in Fig. 10.36. It can be seen that Project_Name.cir.out file (Note: In this example, project name is Counter, so the file name is Counter.cir.out) is added after pressing refresh. Repeat this step for every rerun of KiCad to Ngspice conversion.

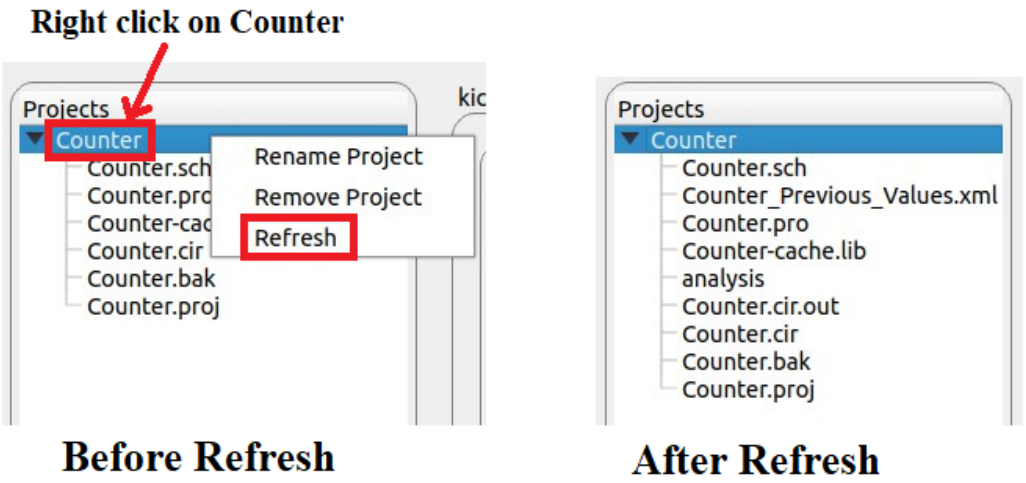


Figure 10.36: Cir.out file added after conversion

5. Next, right click on **Counter.cir.out** file and open it as shown in Fig. 10.37. This file consists of Ngspice netlist of 8 bit counter including information of source, input and output plot labels, transient analysis parameters and control and plot

statements. Refer Fig. 10.37 and Fig. 10.38

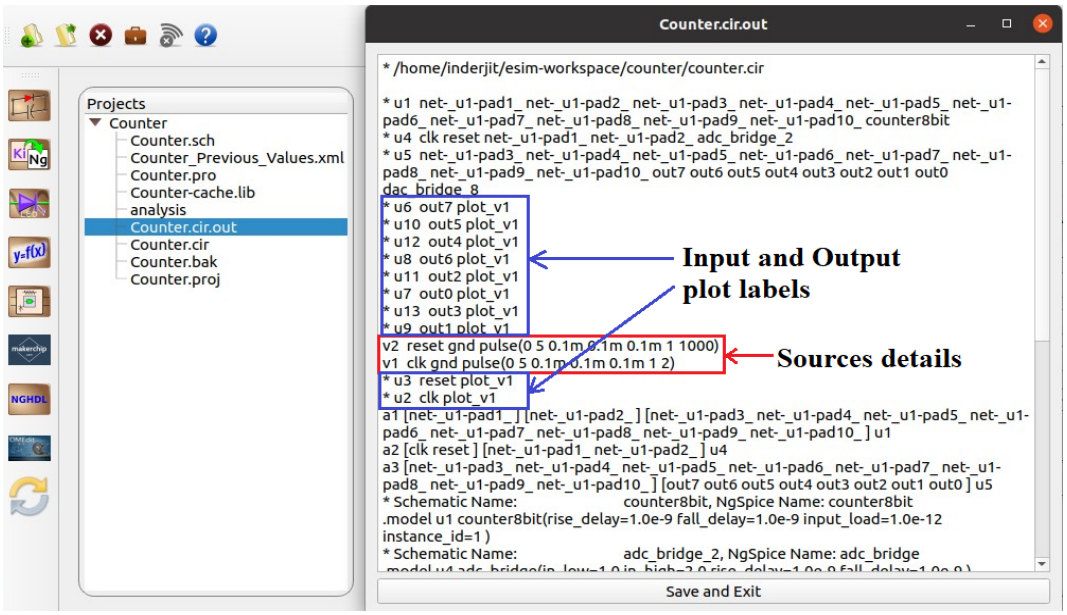


Figure 10.37: Cir.out file details Part 1

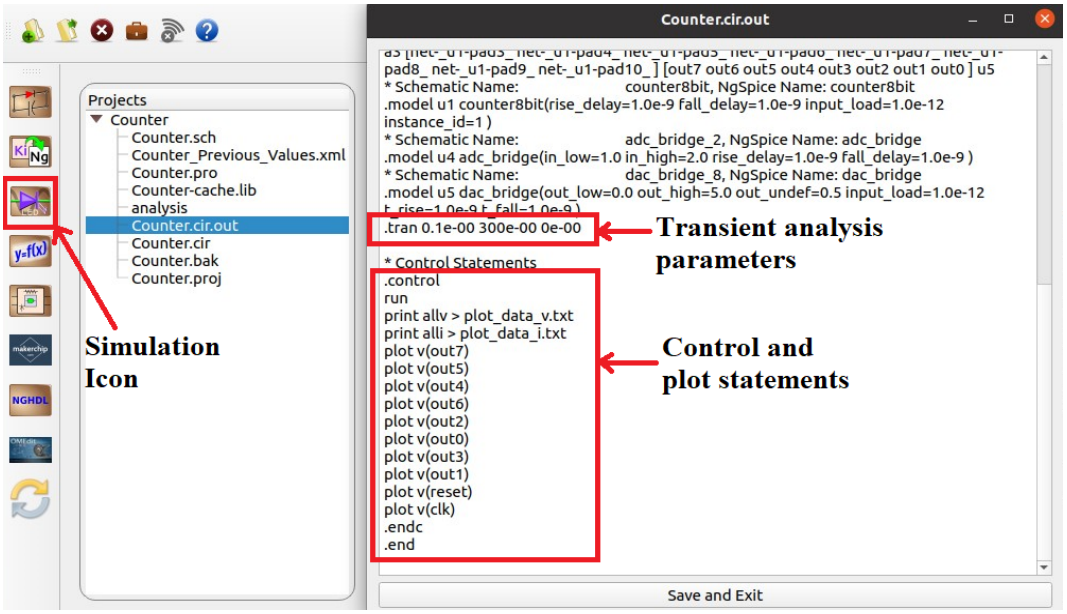


Figure 10.38: Cir.out file details Part 2

6. To run simulation click on **Simulation** icon as shown in Fig. 10.38. It will display input and output plots for 8 bit counter as shown in Fig. 10.39. You can see the plots of input clock, input reset and outputs out0 to out7. However, these plots are not in order.

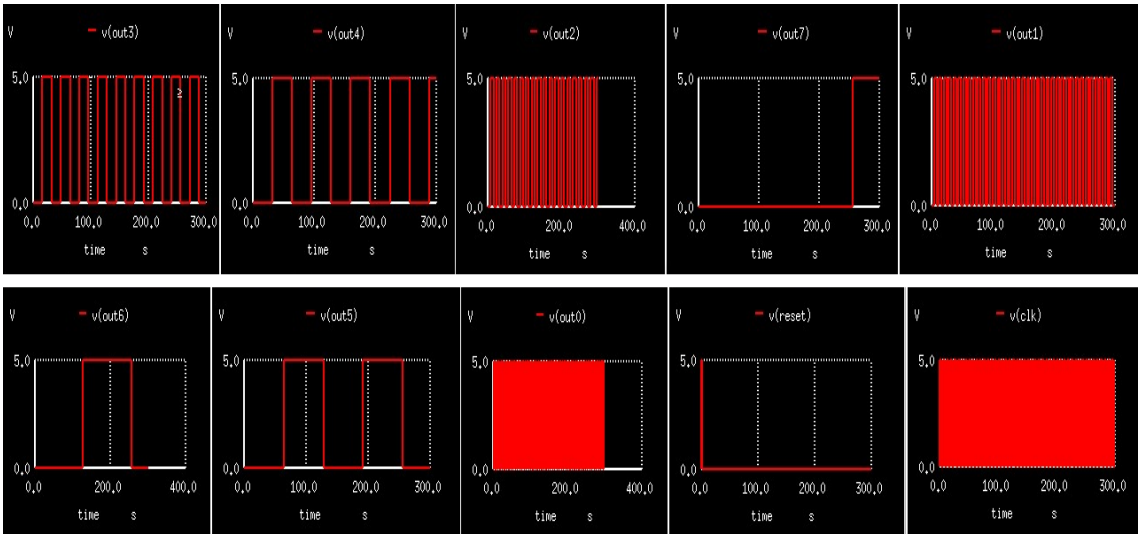


Figure 10.39: Input and Output simulation plots of 8 bit Counter

7. Close the simulations plots, and right click on **Counter** button and click on **Refresh** button as shown in Fig. 10.36 to update the plot data files in the Projects folder.

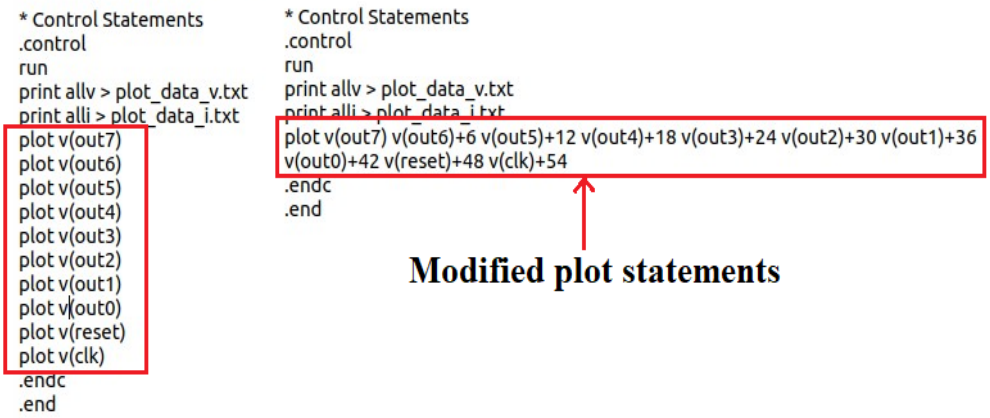


Figure 10.40: Editing the plots statements for stacked waveforms

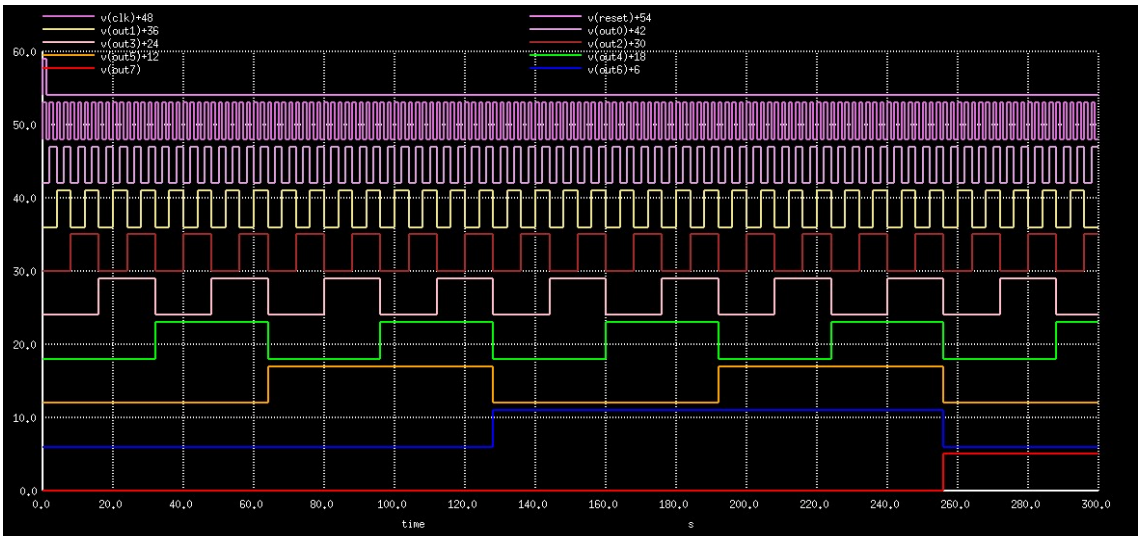


Figure 10.41: Stacked Input and Output Simulation plots of 8 bit counter

8. To view the plots in order and stacked manner, open Counter.cir.out file, do the changes in the plot statements as shown in Fig. 10.40.

9. Running simulation again will result in the output of 8 bit counter as shown in Fig. 10.41.

10.5 Common errors encountered in Makerchip-NgVeri:

This section describes the most common errors faced by the user while building projects using Makerchip-NgVeri feature in eSim.

1. In Makerchip tab, when the user clicks on Add Top Level Verilog button to load the file, the allowed file extension's should be either .v, .sv or .tlv as shown in Fig. 10.42. It will not accept any other file extension.

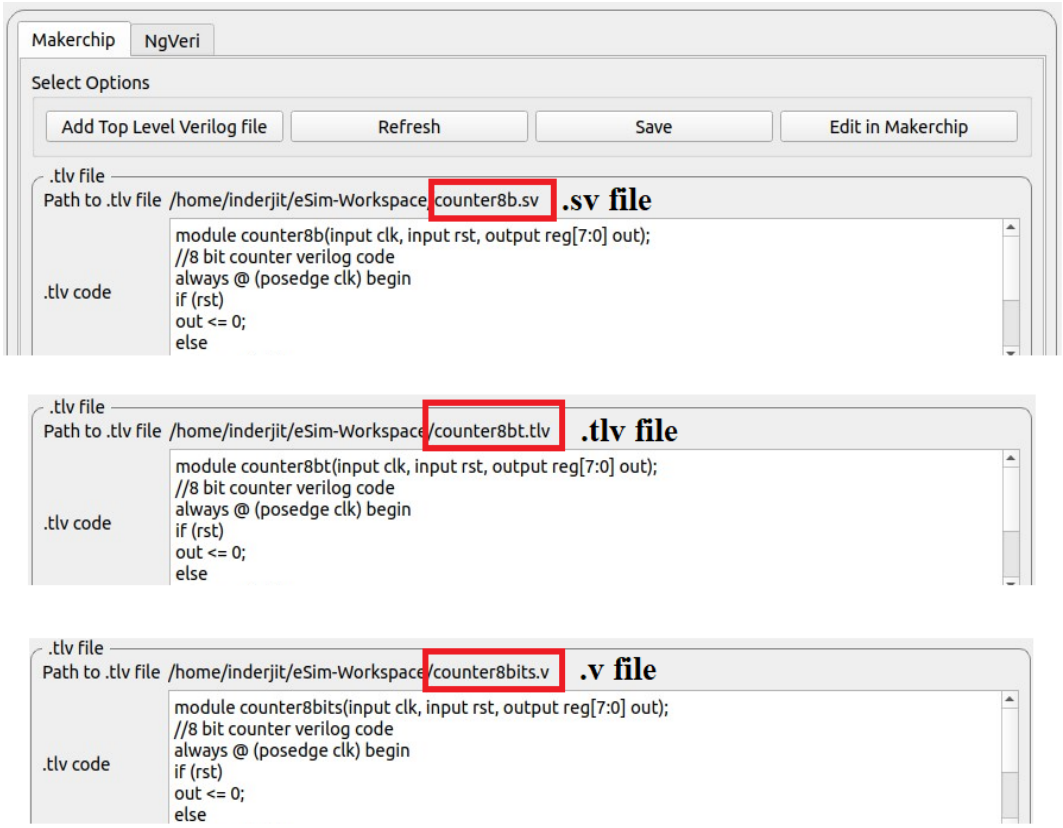


Figure 10.42: File extension allowed in NgVeri

2. User should avoid spaces or special characters in the path to file. Fig. 10.43 shows file path and workspace path without any spaces and special characters.

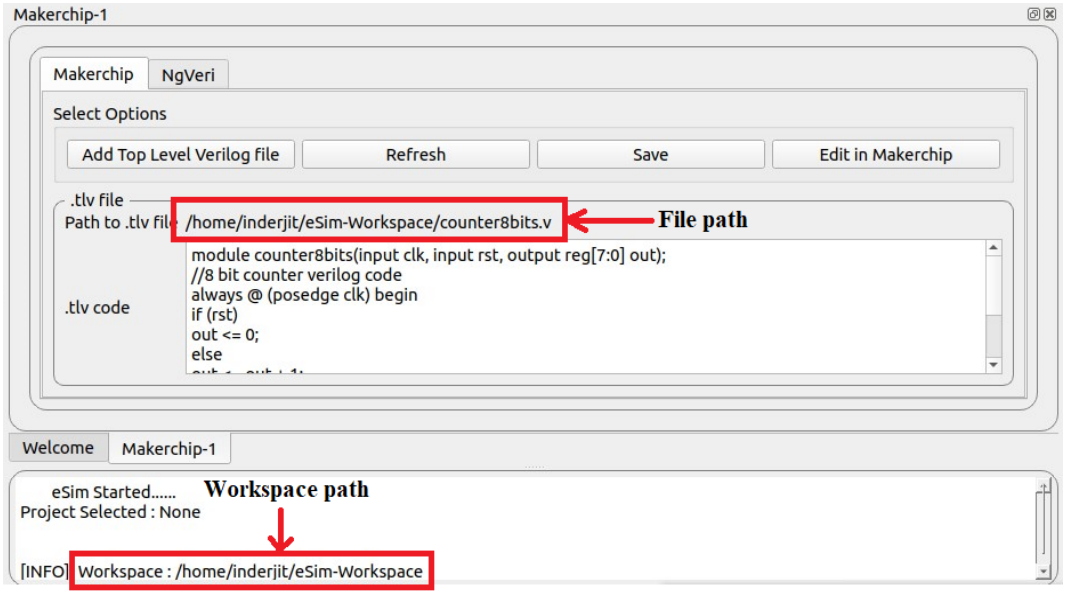


Figure 10.43: File and workspace path without spaces and special characters

3. The verilog file should have Read and write Permissions. To set read and write permissions of the verilog file, right click on verilog file and click on **Properties**. Next click on **Permissions** and select access as **Read and write** as shown in Fig. 10.44.

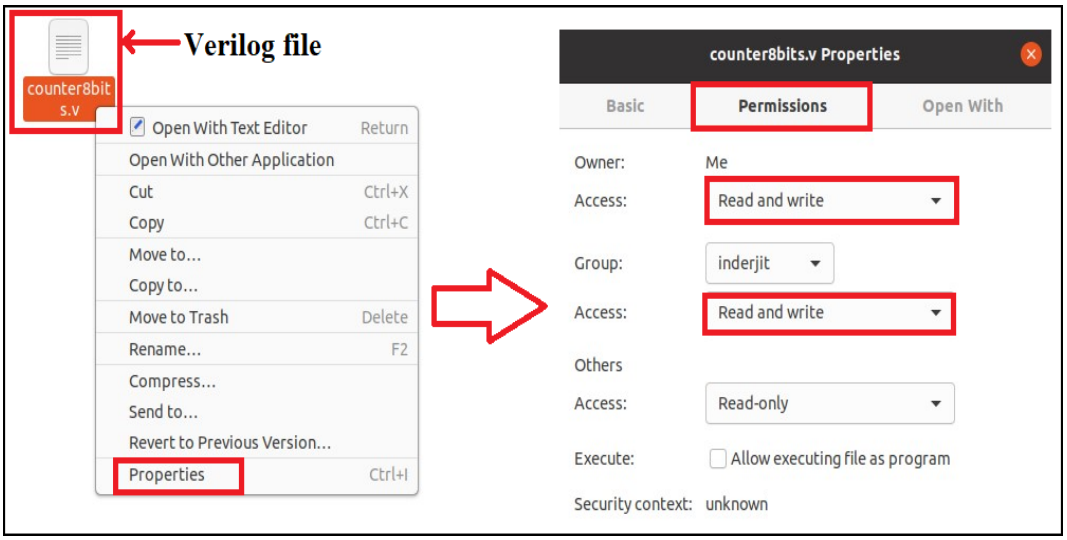


Figure 10.44: Read-Write Permissions for Verilog file

4. Do not delete or rename the verilog file while it is loaded in eSim.

5. If one gets an error in the NgVeri terminal, please clear the terminal and rerun.

6. Do not use delays in the verilog code (as verilator does not support it).

7. Verilog filename should be the same as top-level verilog module name, otherwise the following error message will appear as shown in Fig. 10.45. This error appears when a user tries to open TL verilog in Makerchip by clicking **Edit in Makerchip** button.

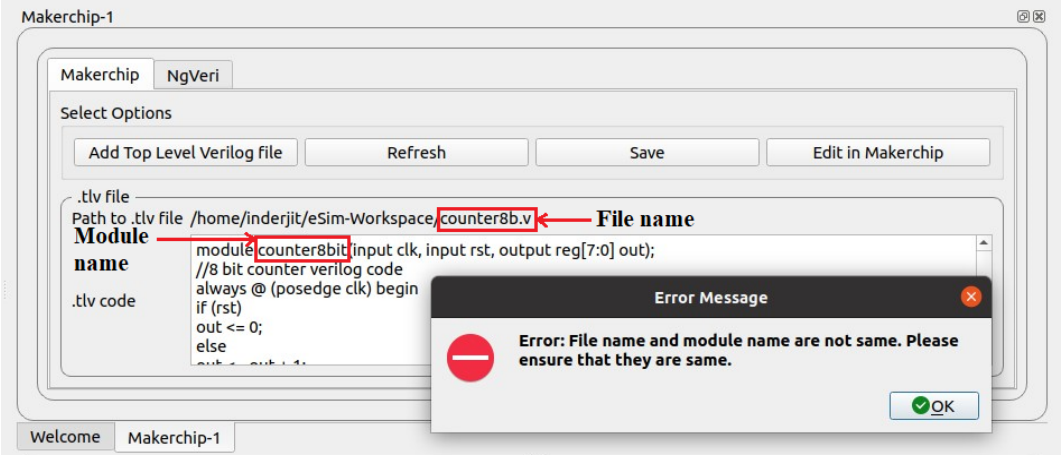


Figure 10.45: Verilog file and module name error

8. **Verilator lint-off errors:** Lint-off commands will create this error in verilog code in Makerchip IDE can be resolved by removing such lint-off commands as shown in Fig. 10.14 and Fig. 10.15. Details about dealing with removal of such errors are already discussed in Makerchip steps in eSim.

9. **Unexpected reg error:** This type of error can occur after compiling TL verilog code in Makerchip IDE visible in Log section (Refer Fig. 10.17). This error can be rectified by removing word *reg* from top level verilog code as shown in Fig. 10.18. More details are discussed in Makerchip steps in eSim.

Chapter 11

OpenModelica

11.1 Introduction

OpenModelica (OM) is an open source modeling and simulation tool based on Modelica language. Modelica is an object oriented language. As a result, it has all the features of an object oriented language such as inheritance. Models or circuits are defined in the form of classes, with in which there are components, functions, connection and placement information. The OM suite has the following major tools.

11.1.1 OMEdit

An IDE for modeling and simulation. It supports a lot of electrical components. It has a good graphical interface to drag and drop components and create the circuit. One can only do transient simulation using this interface. An attractive feature of OMEdit is the plotting interface. All the parameters in the circuit like voltages and currents through each component, parameters like frequency, delay etc. will be displayed as a list, after simulation. The user can choose the variables to be plotted in an interactive manner from this list. On choosing the variable to plot, it will be plotted on the plot window. One can also create multiple plot windows.

11.1.2 OMOptim

An IDE for optimisation. It lists all the variables in the given model. One can choose the variables to be optimised from the list. Multiple models can be loaded for a given optimisation problem. One can do multi objective optimisations as well. It supports various optimisation algorithms such as Particle Swarm Optimisation (PSO) and Simulated Annealing (SA). The results are displayed graphically.

11.2 OpenModelica in eSim

The above two functionalities can be accessed through the `Modelica Converter` and `OM Optimisation` tools on the eSim left toolbar. The two examples given below illustrates how to use OpenModelica in eSim.

Low Pass Filter circuit

Let us now see how to simulate a low pass filter in OpenModelica.

1. Open the schematic and create the circuit as shown in Fig. 11.1.

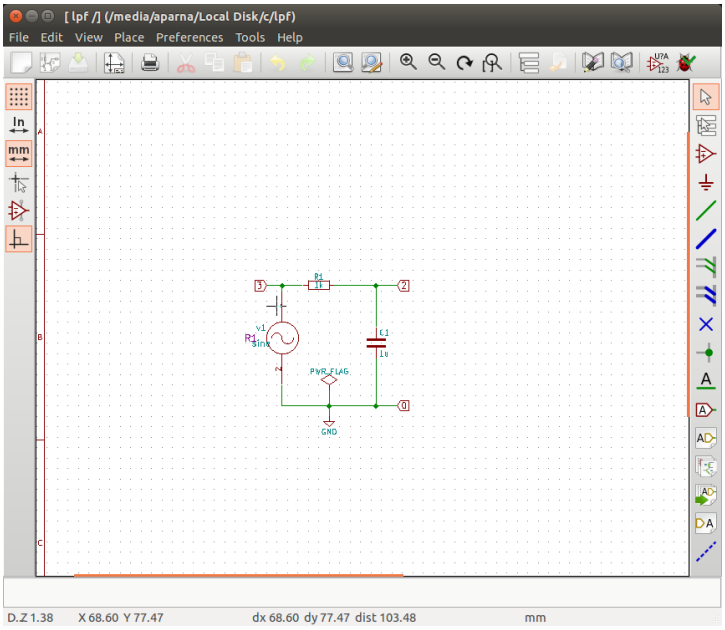


Figure 11.1: Circuit schematic: Low pass filter

2. Create the KiCad netlist. Now the analysis and analysis parameters are given as shown in Fig. 11.2.

3. The source details are given as in Fig. 11.3. The generated KiCad netlist is then converted to ngspice compatible netlist.

4. Simulate the ngspice netlist. The simulation curves are shown in Fig. 11.4.

5. Now to use OpenModelica, click on `Modelica Converter` in the bottom left of eSim left toolbar.*M*ake sure you have OpenModelica installed in the system. This converter converts the spice netlist to Modelica format. Click on the LPF in the

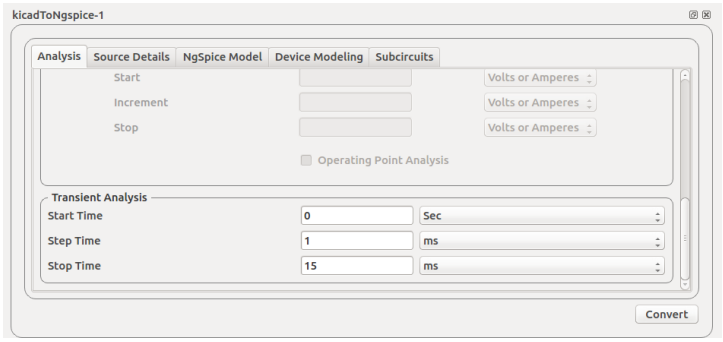


Figure 11.2: Analysis parameters: Low pass filter

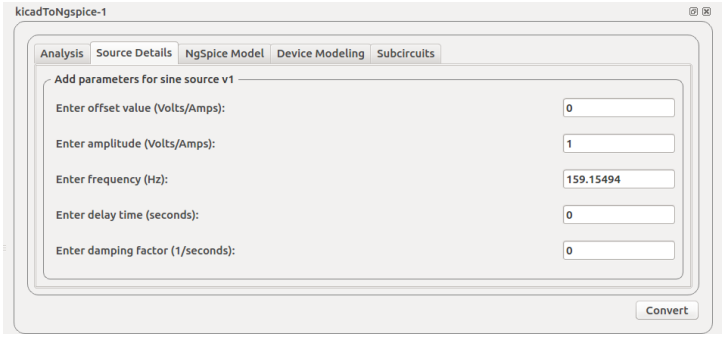


Figure 11.3: Source details: Low pass filter

left that is appended in OpenModelica main window. Make sure you are in text view to see the Modelica code as shown in Fig. 11.5 Figure shows that LPF circuit is being used as a model, the initialisation of sources and components are in the beginning followed by the connection information. n3, n0,n2 are the nodes.

Default Modelica libary is used for electrical sources and components. This has to imported so that it can be used in the current circuit. This is available in the left side of main window.

6. Click on Simulation Setup on the toolbar at the top. A window opens as shown in Fig. 11.6. Give start and stop time. Click `OK`.

7. A plotting window opens. Click on the node at the right to display the waveform. The window is shown in Fig. 11.7.

11.2.1 OM Optimisation

Now let us explore how to use OpenModelica for optimisation through an example. Find the value of resistance R2 that maximises the power dissipated through it for the

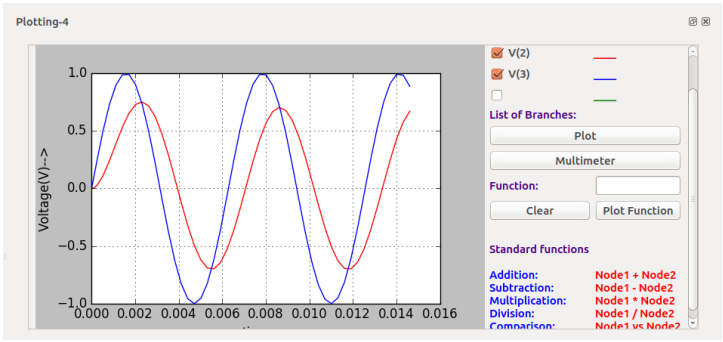


Figure 11.4: Simulation: Low pass filter

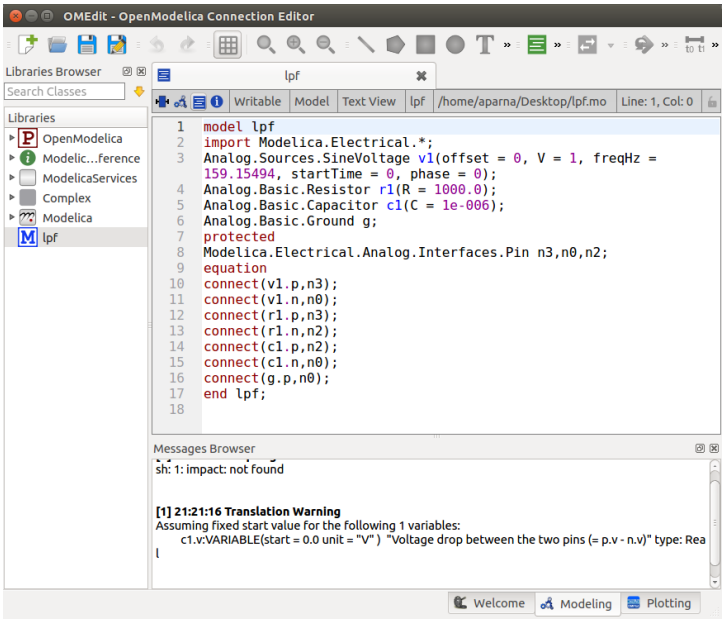


Figure 11.5: OpenModelica: Text view

circuit in Fig. 11.8. This is an illustration of the Maximum Power Transfer Theorem. The power is maximum when R2 = R1, i.e., when R2 = 100. So maximum power would be Pmax = 0.0625. Let us now see the steps to be followed find the value of R2 using eSim.

1. Follow all the steps as above and generate the Modelica model using the Ngspice to Modelica converter.

2. The objective function is $Power = i^2 \times R2$. To define the objective function, the line $power := i^2 \times R1 + i^2 \times R2$ is added under the keyword algorithm, in the

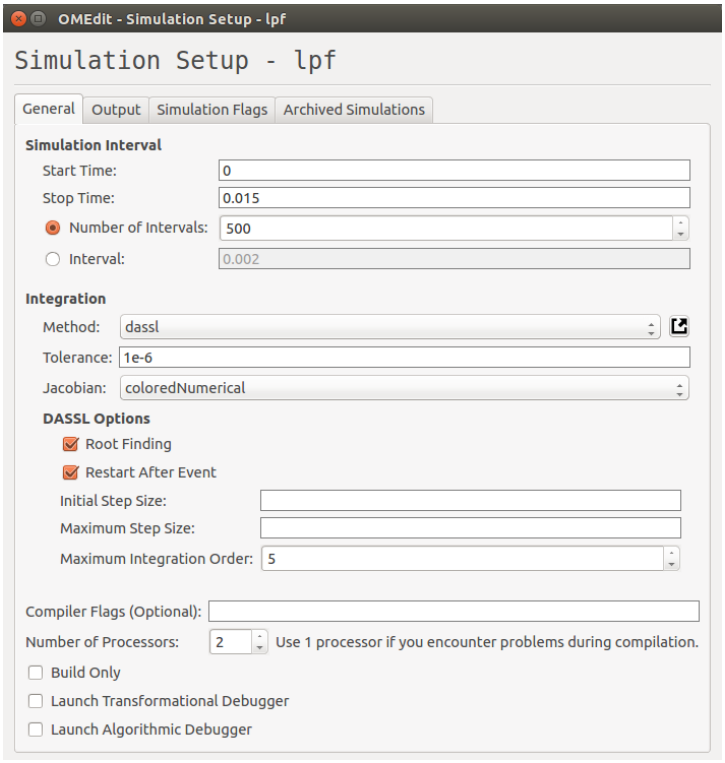


Figure 11.6: OpenModelica: Simulation setup

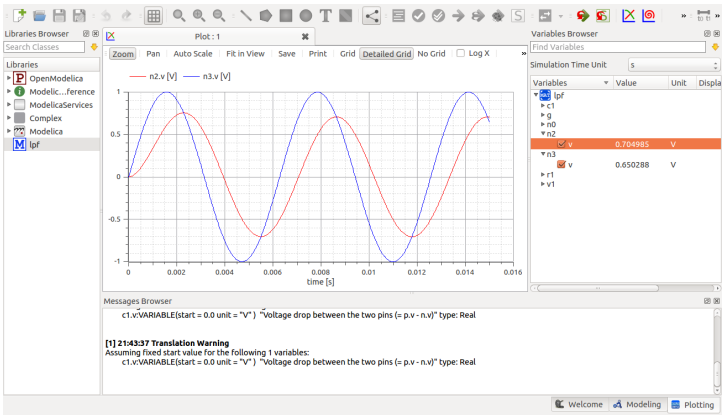


Figure 11.7: OpenModelica: Simulation

Modelica model file.

3. Select `OMOptim` from eSim left toolbar, in the displayed window click on `New`

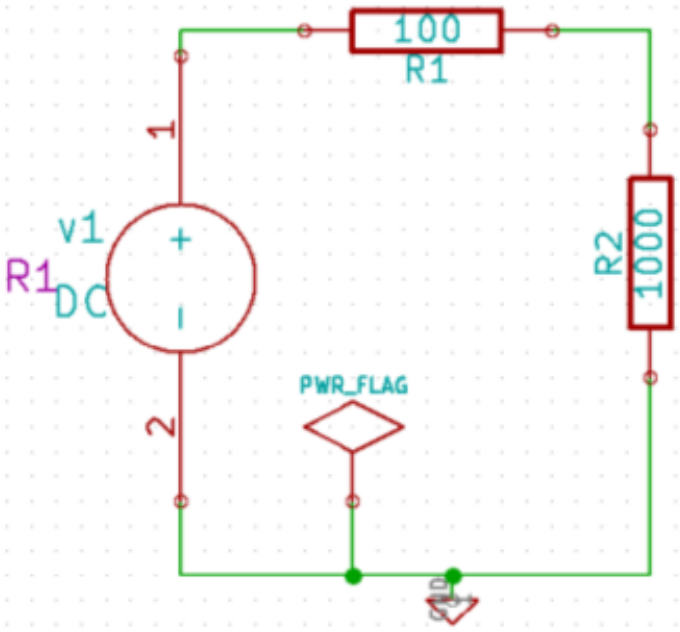


Figure 11.8: Circuit schematic for optimisation

`Project`. Then save the project. It is stored with an extension `.min`. Now select `Models` and then `Load Modelica Library`. Now select `Load mo file` under `Models`. It will be added on the left.

4. Click `Problems` and then `Optimisation`. Select the model to be optimised. *Note that for optimising, that model has to be loaded in OpenModelica as stated before.* Clicking blue turnover icon will display all the variables used in the model. Add details like optimsation variables and objective.

   The OMOptim project for this problem is given in Fig. 11.9. Power is the objective function that has to be maximized. `r2.R` is the variable that will be varied. `r2.R` is limited between 0 and 1000.

5. Click on Parameters tab to select the type of algorithm and its parameters. In this example, the optimisation algorithm used is PSO (Particle Swarm Optimisation). The various parameter values given are as follows: population size as 50, Inertia factor as 1, Learning factor: alpha and beta as 2, Population saving frequency was 1. Iteration limit is also specified. Select the .mo file to be simulated from `Files` tab. Click on `Launch`. The results of optimisation for various values of Iteration Limit are given in Fig. 11.10.

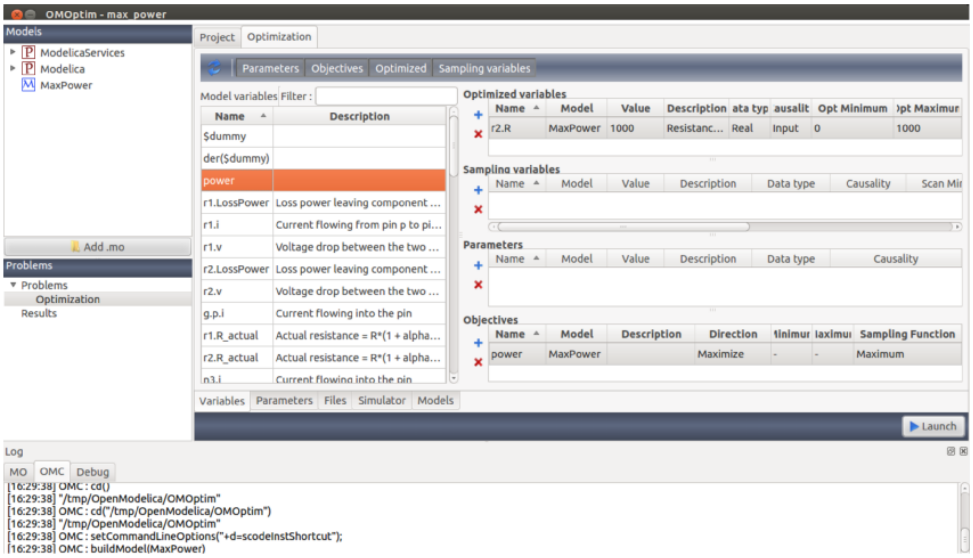


Figure 11.9: OMOptim project

Iteration Limit	$R_2(\Omega)$	**Power (W)**
100	102.652	0.0624893
200	99.9942	0.0625
300	99.9221	0.0625
400	98.6115	0.0624969
500	99.9923	0.0625
600	100.968	0.0624985
700	100.648	0.0624993

Figure 11.10: Optimisation values for various Iteration Limit

6. Depending on the type of algorithm, the time for optimisation varies. Optimised result is graphically displayed as shown in Fig. 11.11.

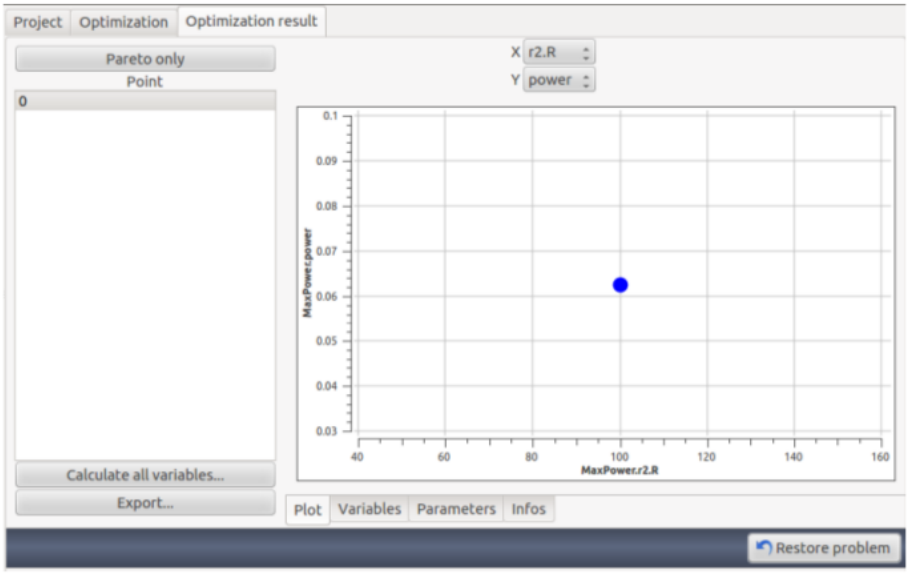


Figure 11.11: Optimised value of resistance for maximum power

Chapter 12

Solved Examples

12.1 Solved Examples

12.1.1 Basic RC Circuit

Problem Statement:

Plot the Input and Output Waveform of an RC circuit whose input voltage (Vs) is 50Hz, 3V peak to peak. The values of Resistor (R) and Capacitor(C) are $1k$ and $1uf$ respectively.

Solution:

- Creating a Project: The new project is created by clicking the `New` icon on the menubar. The name of the project is given in the pop up window as shown in Fig. 12.1.
- Creating the Schematic: To create the schematic, click the very first icon of the left toolbar as shown in the Fig. 12.2. This will open KiCad Eeschema.

  To create a schematic in KiCad, we need to place the required components. Fig. 12.3 shows the icon on the right toolbar which opens the component library.

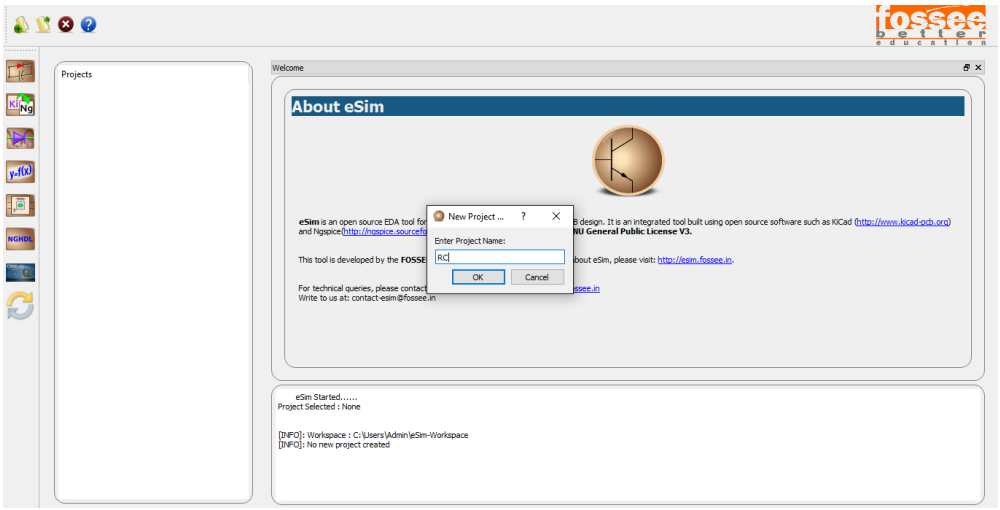


Figure 12.1: Creating New Project

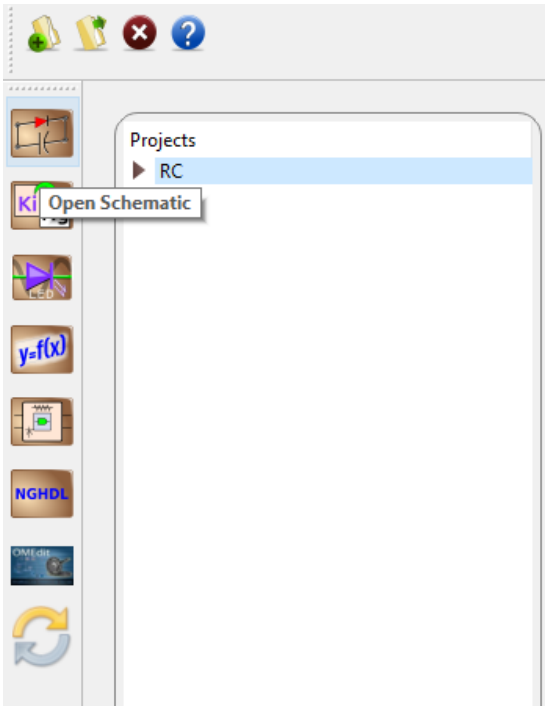


Figure 12.2: Open Schematic Editor

After all the required components of the simple RC circuit are placed, wiring is done using the `Place Wire` option as shown in the Fig. 12.4

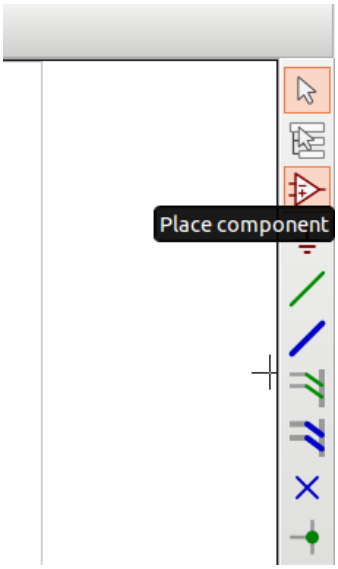


Figure 12.3: Place Component Icon

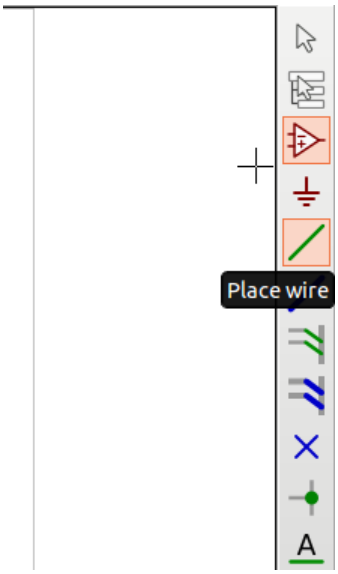


Figure 12.4: Place Wire Icon

Next step is `ERC (Electric Rules Check)`. Fig. 12.5 shows the icon for `ERC`. Fig. 12.6 shows the RC circuit after connecting the components by wire.

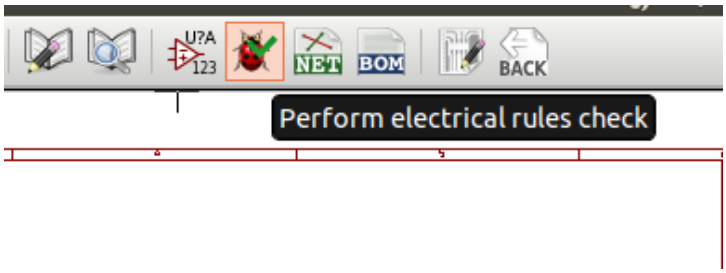


Figure 12.5: Electric Rules Check Icon

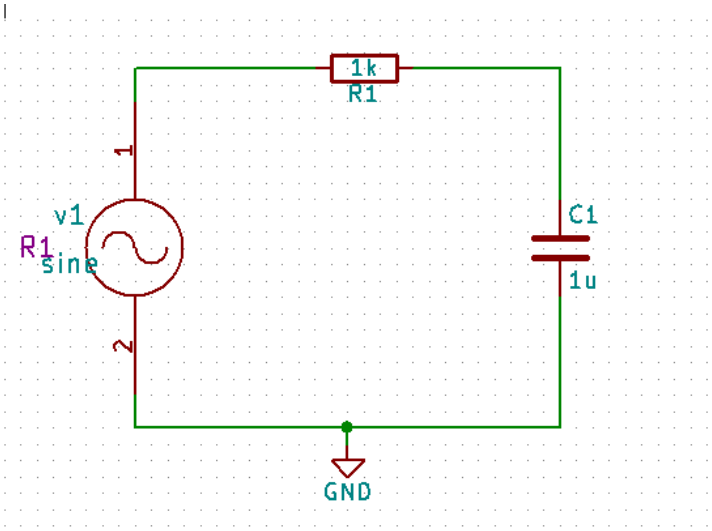
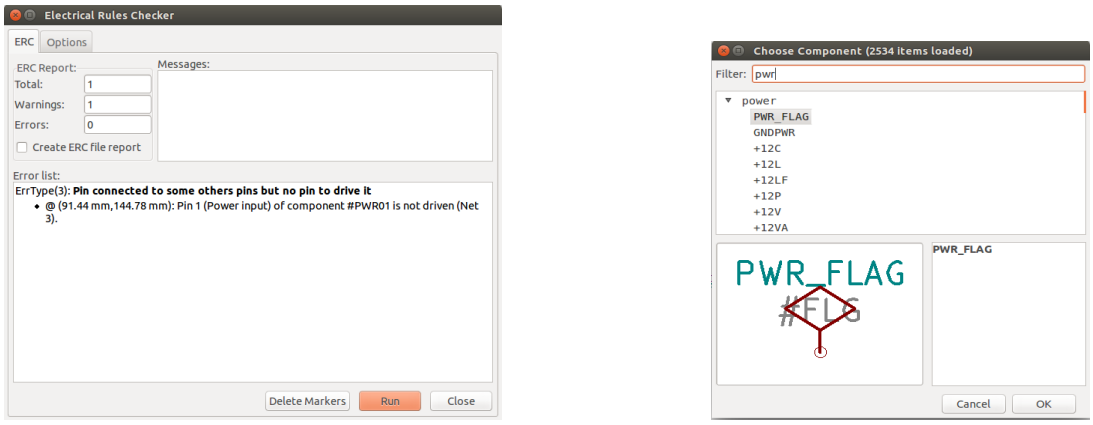


Figure 12.6: RC circuit

After clicking the `ERC` icon a window opens up. Click the `Run` button to run rules check. The errors are listed in as shown in Fig. 12.7a. This error is handled by adding `Power Flag` as shown in Fig. 12.7b.

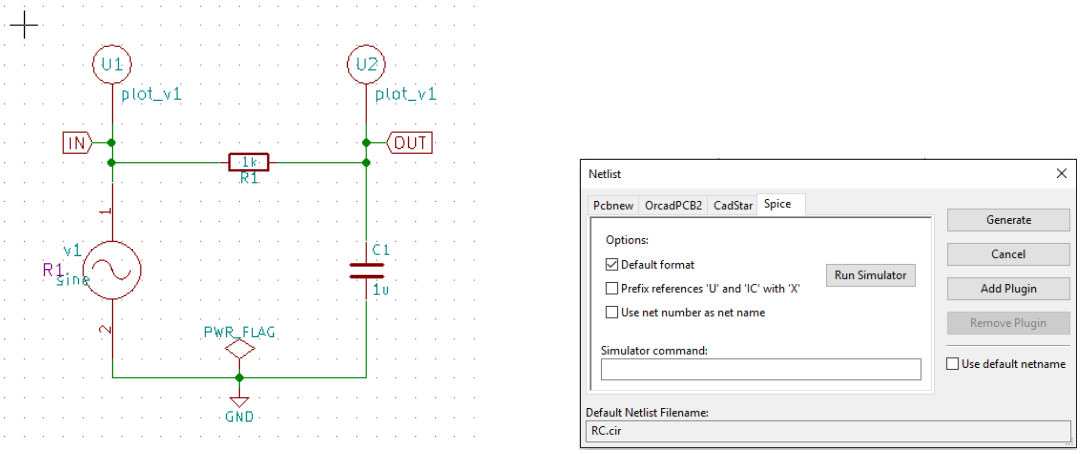
After adding the `Power Flag` the completed RC circuit is shown in Fig. 12.8a and the netlist is generated as shown in Fig. 12.8b.



(a) ERC Run

(b) Power Flag

Figure 12.7: ERC check and POWER FLAG



(a) Schematic of RC circuit

(b) Generating KiCad Netlist of RC circuit

Figure 12.8: RC Schematic and Netlist Generation

- Convert KiCad to Ngspice: To convert KiCad netlist of RC circuit to NgSpice compatible netlist click on KiCad to Ngspice icon as shown in Fig. 12.9.

  Now you can enter the type of analysis and source details as shown in Fig. 12.10a and Fig. 12.10b respectively.

  The other tab will be empty as RC circuit do not use any Ngspice model, device library and subcircuit.

  After entering the value, press the convert button. It will convert the netlist into Ngspice compatible netlist.

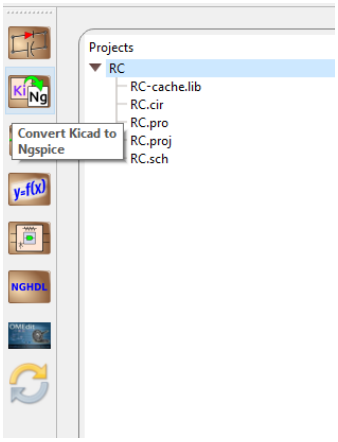
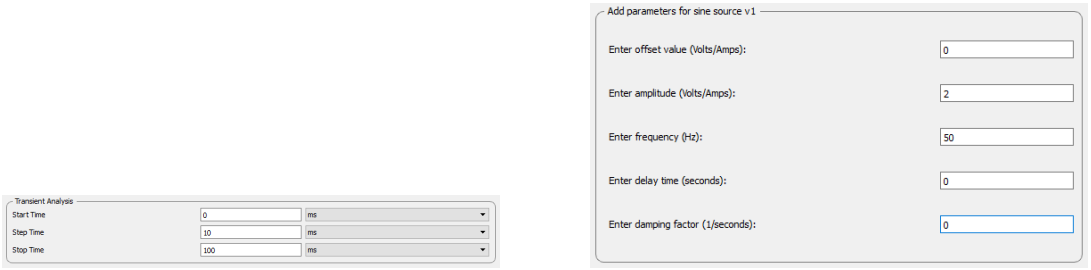


Figure 12.9: Convert KiCad to Ngspice Icon



(a) RC Analysis

(b) RC Source Details

Figure 12.10: RC Analysis and Source Detail

- Simulation: To run Ngspice simulation click the simulation icon in the tool bar as shown in the Fig. 12.11.

  In eSim, there are two types of plot. First is normal Ngspice plot and second is interactive python plot as shown in Fig. 12.12a and Fig. 12.12b respectively.

  In the interactive python plot you can select any node or branch to plot voltage or current across it. Also it has the facility to plot basic functions across the node like addition, substraction, multiplication, division and v/s.

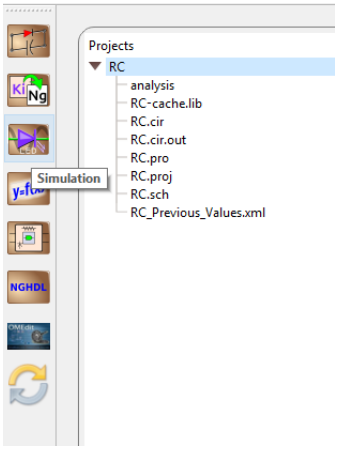
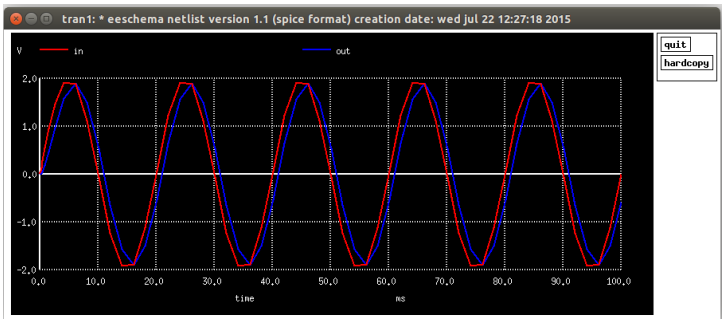
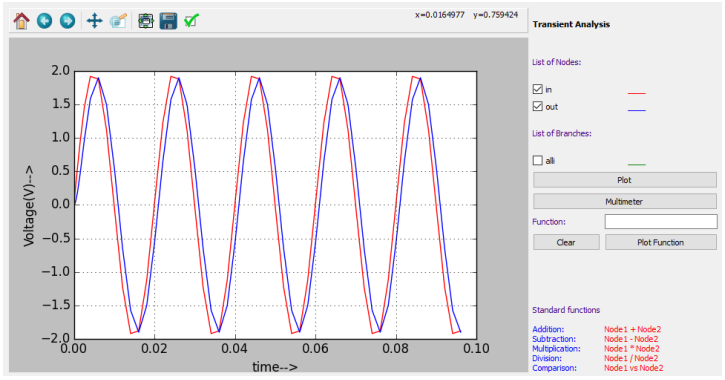


Figure 12.11: Simulation Icon



(a) Ngspice Plot of RC



(b) Python Plot of RC

Figure 12.12: Ngspice and Interactive Python Plotting

12.1.2 Half Wave Rectifier

Problem Statement:

Plot the Input and Output Waveform of Half Wave Rectifier circuit where the input voltage (Vs) is 50Hz, 2V peak to peak. The value for Resistor (R) is 1k.

Solution:

The new project is created by clicking the `New` icon on the menubar. The name of the project is given in the window shown in Fig. 12.1.

- Creating Schematic: To create the schematic, click the very first icon of the left toolbar as shown in the Fig. 12.2. This will open KiCad Eeschema.

  After the KiCad window is opened, to create a schematic we need to place the required components. Fig. 12.3 shows the icon on the right toolbar which opens the component library.

  After all the required components of the simple Half Wave rectifier circuits are placed, wiring is done using the `Place Wire` option as shown in the Fig. 12.4

  Next step is `ERC (Electric Rules Check)`. Fig. 12.5 shows the icon for `ERC`. After completing all the above steps the final Half Wave Rectifier schematic will look like Fig. 12.13.

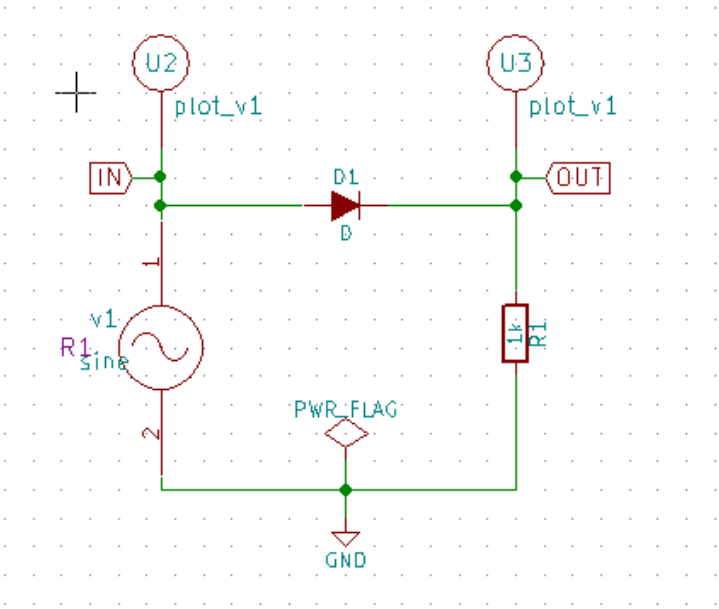


Figure 12.13: Schematic of Half Wave Rectifier circuit

KiCad netlist is generated as shown in the Fig. 12.14

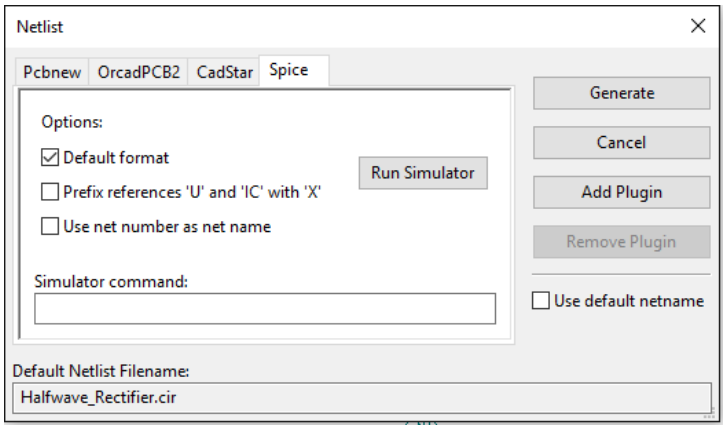
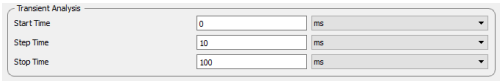
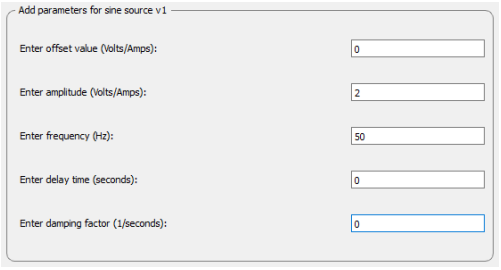


Figure 12.14: Half Wave Rectifier circuit Netlist Generation

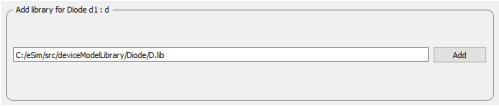
- Convert KiCad to Ngspice: After creating KiCad netlist, click on the `KiCad-Ngspice converter` button. This will open converter window where you can enter details of Analysis, Source values and Device library.



(a) Half Wave Rectifier Analysis



(b) Half Wave Rectifier Source Details

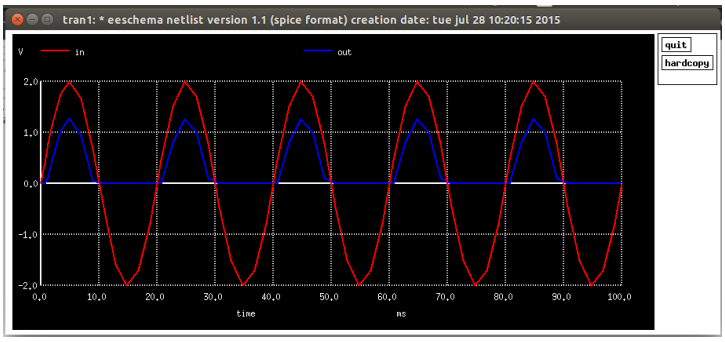


(c) Half Wave Rectifier Device Modeling

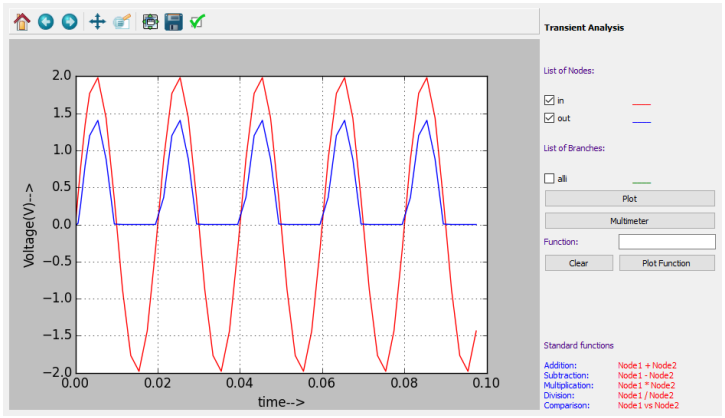
Figure 12.15: Analysis, Source and Device Tab

Under device library you can add the library for diode used in the circuit. If you do not add any library it will take default Ngspice model.

- Simulation: Once the KiCad-Ngspice converter runs successfully, you can run simulation by clicking the simulation button in the toolbar.



(a) Ngspice Plot of Half Wave Rectifier



(b) Python Plot of Half Wave Rectifier

Figure 12.16: Half Wave Rectifier Simulation Output

12.1.3 Inverting Amplifier

Problem Statement:

Plot the Input and Output Waveform of Inverting Amplifier circuit where the input voltage (Vs) is $50Hz$, $2V$ peak to peak and gain is 2.

Solution:

- Creating Schematic: To create the schematic, click the very first icon of the left toolbar as shown in the Fig. 12.2. This will open KiCad Eeschema.
  After the KiCad window is opened, to create a schematic we need to place the required components. Fig. 12.3 shows the icon on the right toolbar which opens the component library.
  After all the required components of the inverting amplifier circuit are placed,

wiring is done using the `Place Wire` option as shown in the Fig. 12.4.
Next step is `ERC (Electric Rules Check)`. Fig. 12.5 shows the icon for `ERC`.

The Fig. 12.17 shows the complete Precision Rectifier schematic after removing the errors.

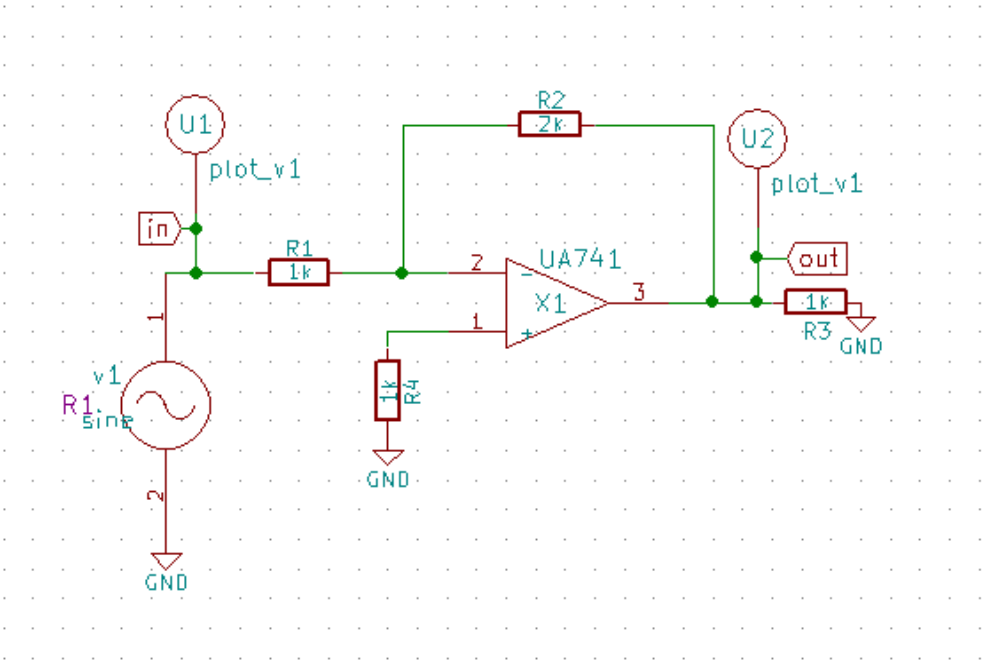


Figure 12.17: Schematic of Inverting Amplifier circuit

The KiCad netlist is generated as shown in Fig. 12.18.

- Convert KiCad to Ngspice: After creating KiCad netlist, click on KiCad-Ngspice converter button.

  This will open converter window where you can enter details of Analysis, Source values, Device library and Subcircuit.

  Subcircuit of Op-Amp is shown in Fig. 12.19d

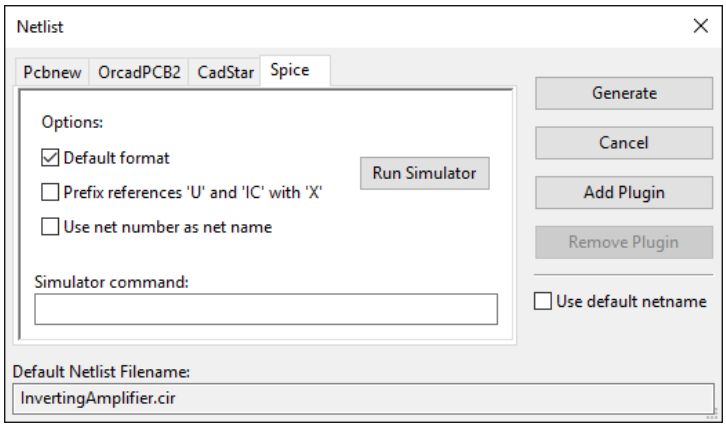
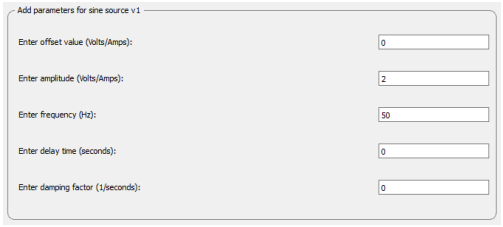


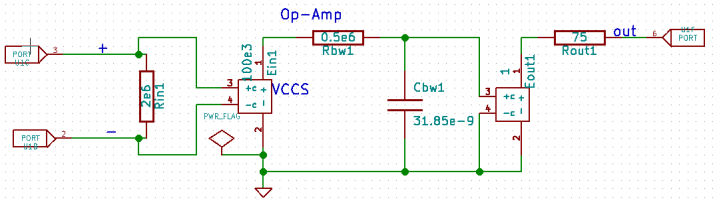
Figure 12.18: Inverting Amplifier circuit Netlist Generation

(a) Inverting Amplifier Analysis



(b) Inverting Amplifier Source Details

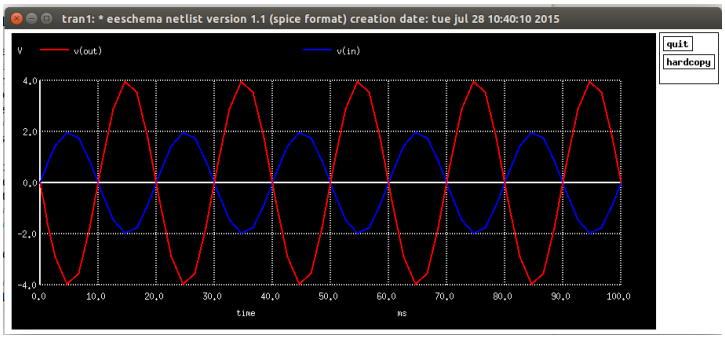
(c) Inverting Amplifier Subcircuit



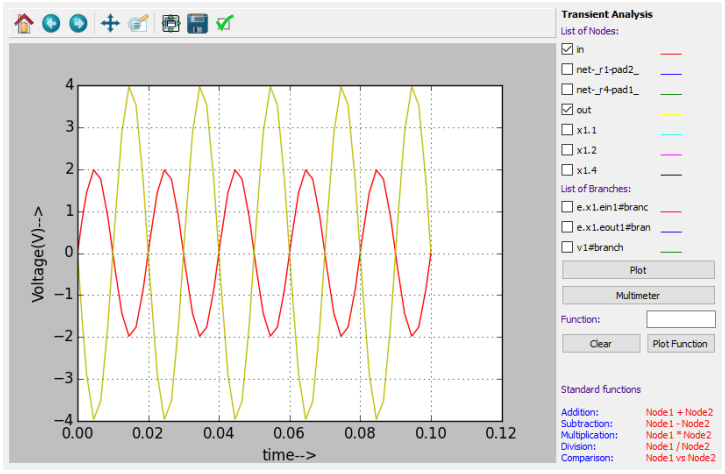
(d) Sub-Circuit of Op-Amp

Figure 12.19: Analysis, Source, and Subcircuit tab

Under subcircuit tab you have to add the subciruit used in your circuit. If you forget to add subcircuit, it will throw an error.



(a) Inverting Amplifier Ngspice Plot



(b) Inverting Amplifier Python Plot

Figure 12.20: Inverting Amplifier Simulation Output

- Simulation: Once the KiCad-Ngspice converter runs successfully, you can run simulation by clicking the simulation button in the toolbar.

12.1.4 Half Adder

Problem Statement:

Plot the Input and Output Waveform of Half Adder circuit.

Solution:

- Creating Schematic: To create the schematic, click the very first icon of the left toolbar as shown in the Fig. 12.2. This will open KiCad Eeschema.
  After the KiCad window is opened, to create a schematic we need to place the required components. Fig. 12.3 shows the icon on the right toolbar which opens the component library.
  After all the required components of the Half Adder circuit are placed, wiring is done using the `Place Wire` option as shown in the Fig. 12.4.
  Next step is `ERC (Electric Rules Check)`. Fig. 12.5 shows the icon for `ERC`.

  The Fig. 12.21 shows the complete Half Adder schematic after removing the errors.

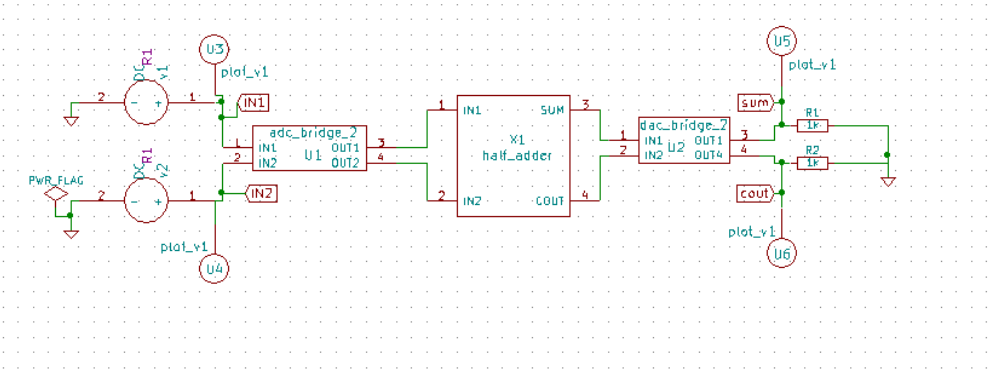


Figure 12.21: Schematic of Half Adder circuit

The KiCad netlist is generated as shown in Fig. 12.22.

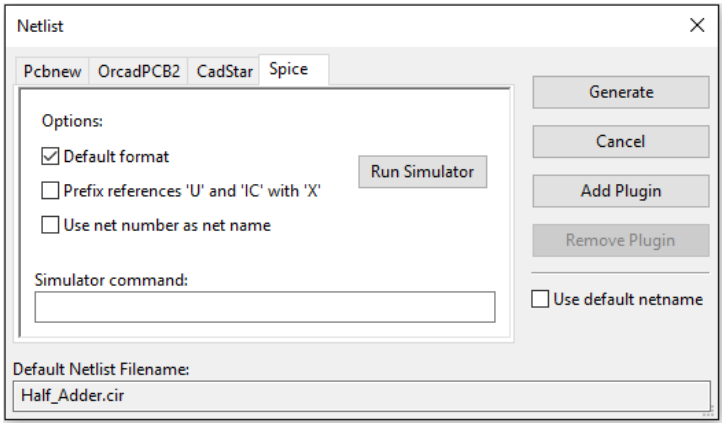
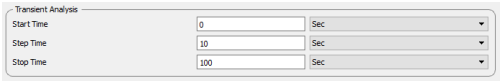


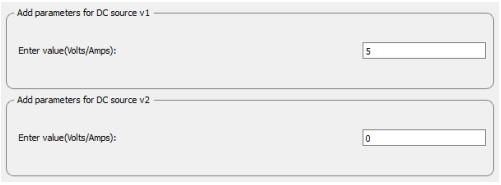
Figure 12.22: Half Adder circuit Netlist Generation

- Convert KiCad to Ngspice: After creating KiCad netlist click on KiCad-Ngspice converter button.

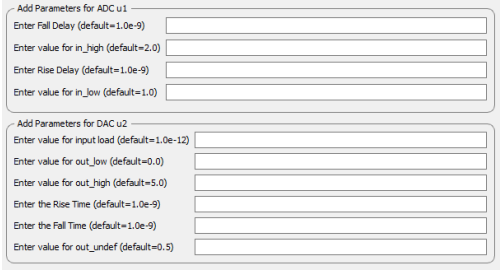
  This will open converter window where you can enter details of Analysis, Source values, Ngspice model and Subcircuit.



(a) Half Adder Analysis



(b) Half Adder Source Details



(c) Half Adder Ngspice Model

(d) Half Adder Subcircuit Model

Figure 12.23: Analysis, Source, Ngspice Model and Subcircuit tab

Subcircuit of Half Adder in Fig. 12.24

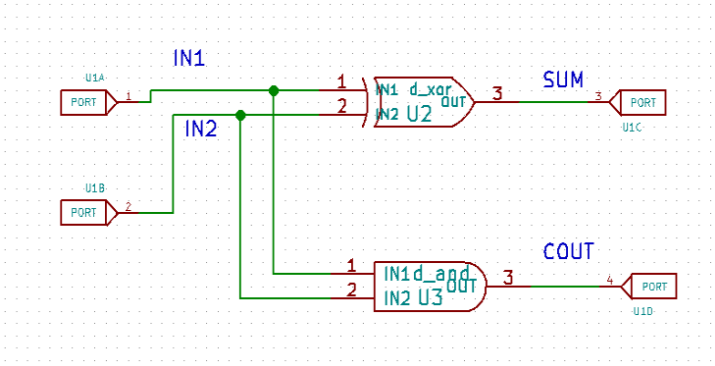
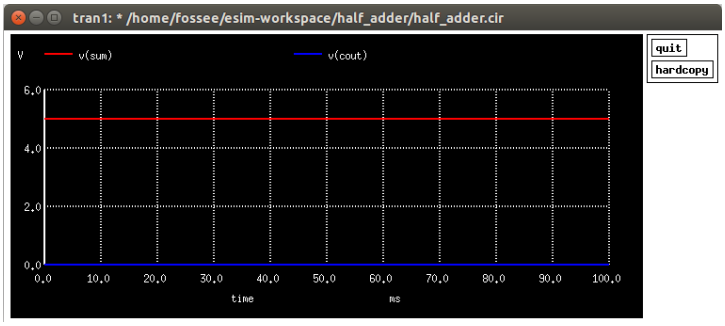
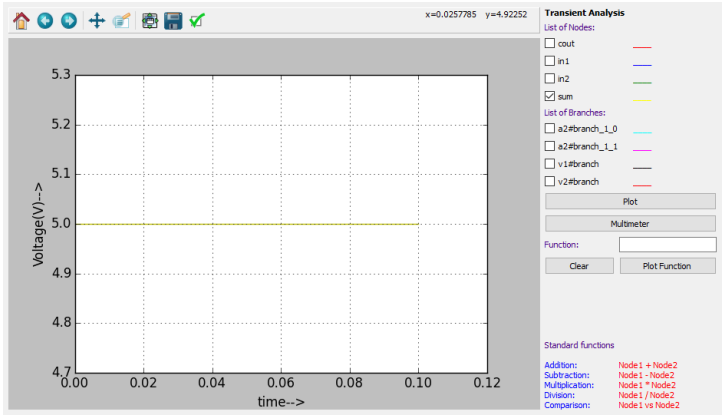


Figure 12.24: Half Adder Subcircuit

- Simulation: Once the KiCad-Ngspice converter runs successfully, you can run simulation by clicking the simulation button in the toolbar.



(a) Half Adder Ngspice Plot



(b) Half Adder Python Plot

Figure 12.25: Half Adder Simulation Output

12.1.5 Full Wave Rectifier using SCR

Problem Statement:

Plot the Input and Output Waveform of Full Wave Rectifier using SCR.

Solution:

- Creating Schematic: To create the schematic, click the very first icon of the left toolbar as shown in the Fig. 12.2. This will open KiCad Eeschema.
  After the KiCad window is opened, to create a schematic we need to place the required components. Fig. 12.3 shows the icon on the right toolbar which opens the component library.
  After all the required components of the Full Wave Rectifier using SCR circuit are placed, wiring is done using the `Place Wire` option as shown in the Fig. 12.4.
  Next step is `ERC (Electric Rules Check)`. Fig. 12.5 shows the icon for `ERC`.

  The Fig. 12.26 shows the complete Rectifier circuit using SCR after removing the errors.

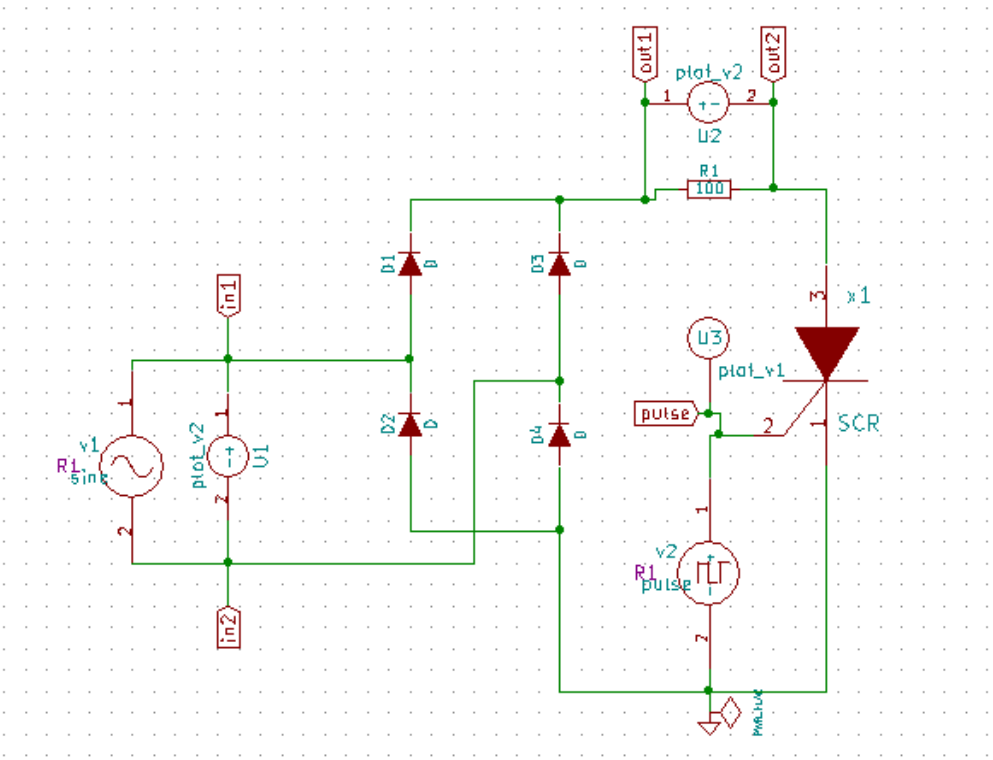


Figure 12.26: Schematic of Full Wave Rectifier using SCR

The KiCad netlist is generated as shown in Fig. 12.27.

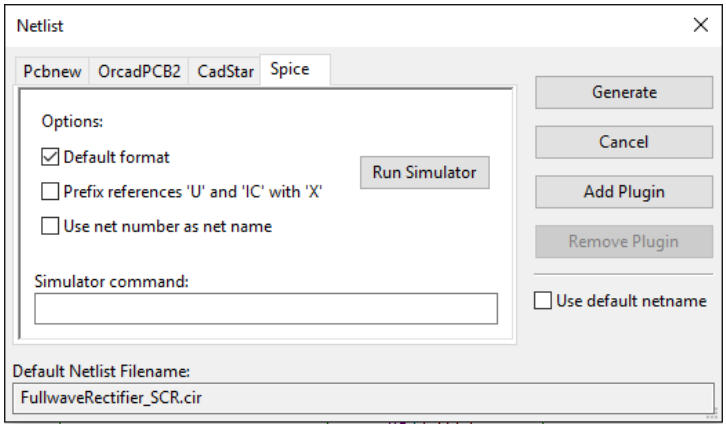
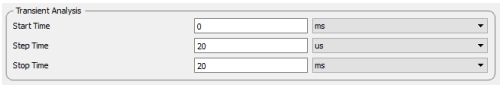


Figure 12.27: Full Wave Rectifier using SCR Netlist Generation

- Convert KiCad to Ngspice: After creating KiCad netlist click on KiCad-Ngspice converter button.

  This will open converter window where you can enter details of Analysis, Source values, Ngspice model and Subcircuit.



(a) Full Wave Rectifier using SCR Analysis

(b) Full Wave Rectifier using SCR Source Details

(c) Full Wave Rectifier using SCR Subcircuit Model

Figure 12.28: Analysis, Source and Subcircuit tab

Subcircuit of SCR in Fig. 12.29

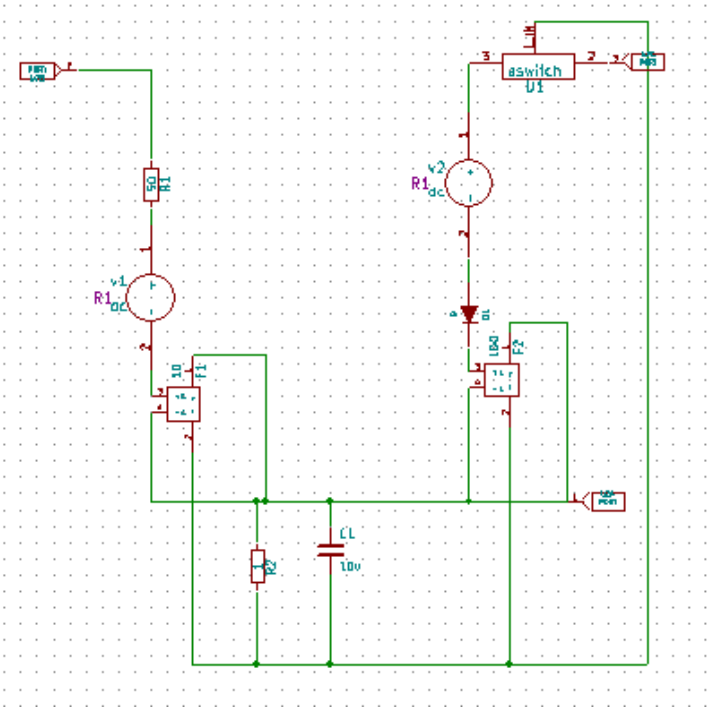
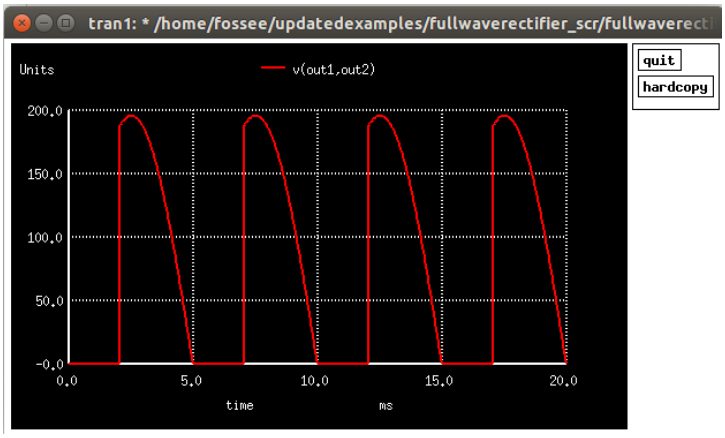
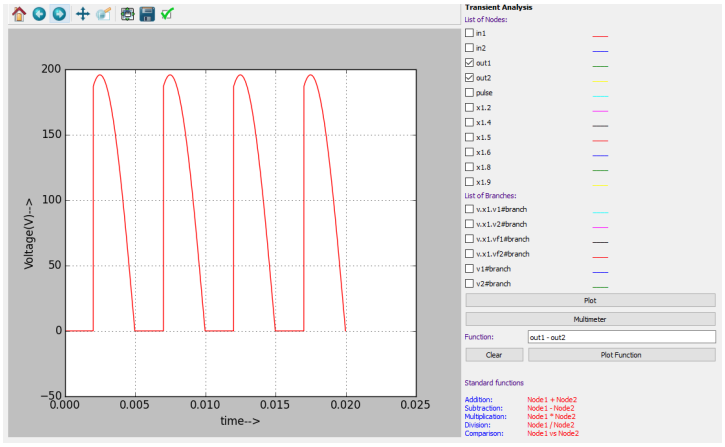


Figure 12.29: SCR Subcircuit

- Simulation: Once the KiCad-Ngspice converter runs successfully, you can run simulation by clicking the simulation button in the toolbar.



(a) Full Wave Rectifier using SCR Ngspice Plot



(b) Full Wave Rectifier using SCR Python Plot

Figure 12.30: Full Wave Rectifier using SCR Simulation Output

12.1.6 Oscillator Circuit

Problem Statement:

Plot the Oscillation Waveforms for Phase Shift Oscillator circuit.

Solution:

The new project is created by clicking the `New` icon on the menubar. The name of the project is given in the window shown in Fig. 12.1.

- Creating Schematic: To create the schematic, click the very first icon of the left

toolbar as shown in the Fig. 12.2. This will open KiCad Eeschema.

After the KiCad window is opened, to create a schematic we need to place the required components. Fig. 12.3 shows the icon on the right toolbar which opens the component library.

After all the required components of the Oscillator circuits are placed, wiring is done using the `Place Wire` option as shown in the Fig. 12.4
sss

Next step is `ERC (Electric Rules Check)`. Fig. 12.5 shows the icon for `ERC`. After completing all the above steps the Oscillator schematic will look like Fig. 12.31.

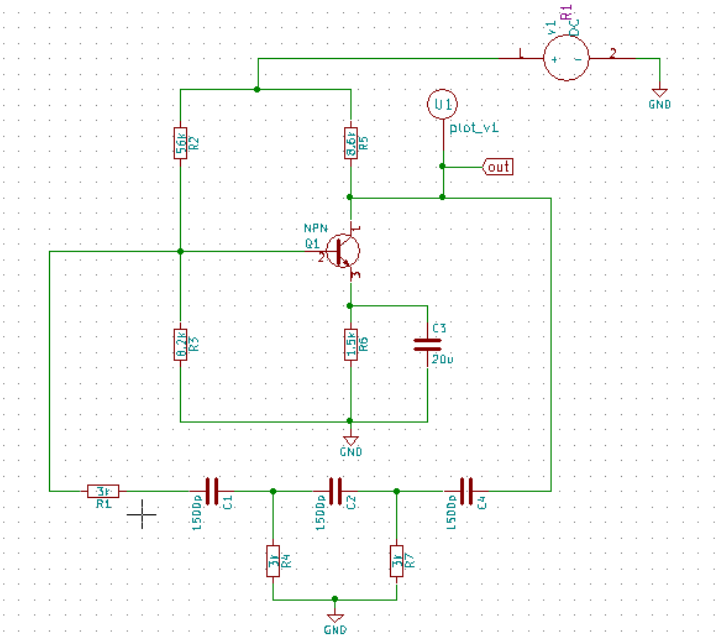


Figure 12.31: Schematic of Phase Shift Oscillator circuit

KiCad netlist is generated as shown in the Fig. 12.32

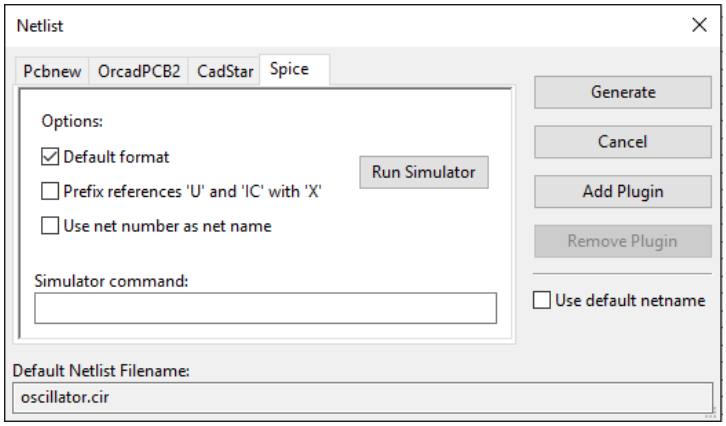


Figure 12.32: Phase Shift Oscillator circuit Netlist Generation

- Convert KiCad to Ngspice: After creating KiCad netlist, click on the `KiCad-Ngspice converter` button. This will open converter window where you can enter details of Analysis, Source values and Device library.

(a) Phase Shift Oscillator Analysis

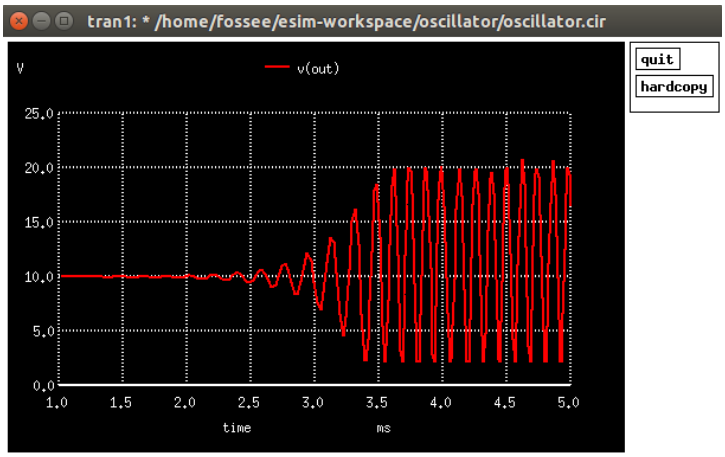
(b) Phase Shift Oscillator Details

(c) Phase Shift Oscillator Device Modeling

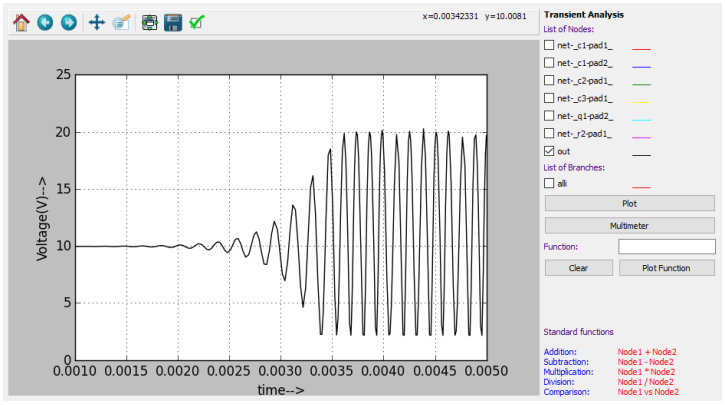
Figure 12.33: Analysis, Source and Device Tab

Under device library you can add the library for diode used in the circuit. If you do not add any library it will take default Ngspice model.

- Simulation: Once the KiCad-Ngspice converter runs successfully, you can run simulation by clicking the simulation button in the toolbar.



(a) Ngspice Plot of Phase Shift Oscillator



(b) Python Plot of Phase Shift Oscillator

Figure 12.34: Phase Shift Oscillator Simulation Output

12.1.7 Characteristics of BJT in Common Base Configuration

Problem Statement:

Plot Characteristics of BJT in Common Base Configuration.

Solution:

The new project is created by clicking the `New` icon on the menubar. The name of the project is given in the window shown in Fig. 12.1.

- Creating Schematic: To create the schematic, click the very first icon of the left toolbar as shown in the Fig. 12.2. This will open KiCad Eeschema.

After the KiCad window is opened, to create a schematic we need to place the required components. Fig. 12.3 shows the icon on the right toolbar which opens the component library.

After all the required components of the simple Half Wave rectifier circuits are placed, wiring is done using the `Place Wire` option as shown in the Fig. 12.4

Next step is `ERC (Electric Rules Check)`. Fig. 12.5 shows the icon for `ERC`. After completing all the above steps the BJT in CB Configuration schematic will look like Fig. 12.35.

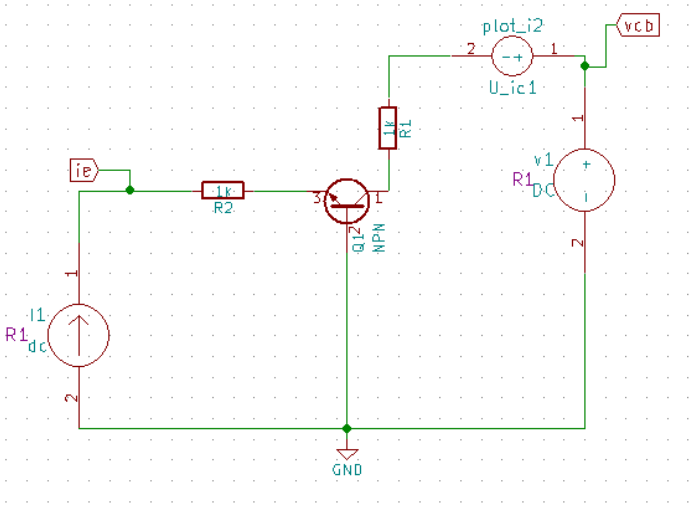


Figure 12.35: Schematic of BJT in CB Configuration circuit

KiCad netlist is generated as shown in the Fig. 12.36

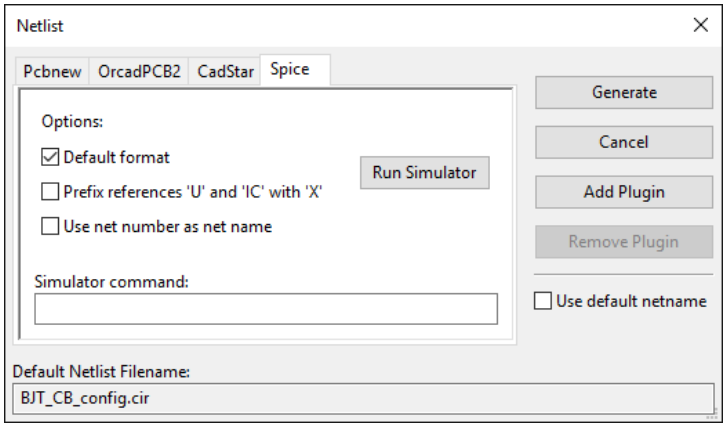
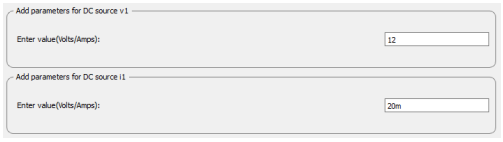


Figure 12.36: BJT in CB Configuration circuit Netlist Generation

- Convert KiCad to Ngspice: After creating KiCad netlist, click on the `KiCad-Ngspice converter` button. This will open converter window where you can enter details of Analysis, Source values and Device library.

(a) BJT in CB Configuration Analysis



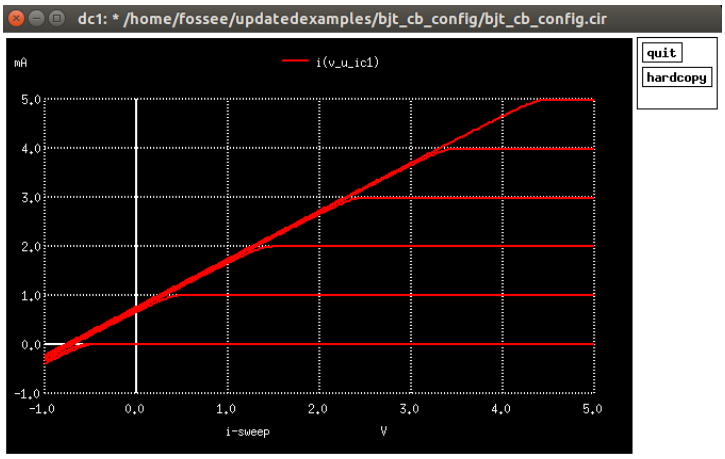
(b) BJT in CB Configuration Source Details

(c) BJT in CB Configuration Device Modeling

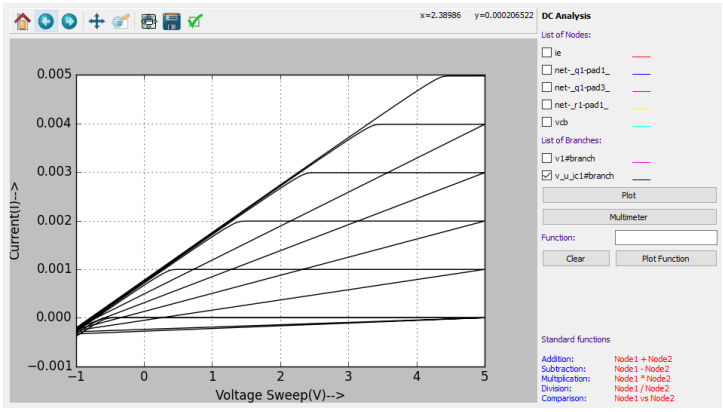
Figure 12.37: Analysis, Source and Device Tab

Under device library you can add the library for diode used in the circuit. If you do not add any library it will take default Ngspice model.

- Simulation: Once the KiCad-Ngspice converter runs successfully, you can run simulation by clicking the simulation button in the toolbar.



(a) Ngspice Plot of BJT in CB Configuration



(b) Python Plot of BJT in CB Configuration

Figure 12.38: BJT in CB Configuration Simulation Output

Chapter 13

PCB Design

Printed Circuit Board (PCB) design is an important step in electronic system design. Every component of the circuit needs to be placed and connections routed to minimise delay and area. Each component has an associated footprint. Footprint refers to the physical layout of a component that is required to mount it on the PCB. PCB design involves associating footprints to all components, placing them appropriately to minimise wire length and area, connecting the footprints using tracks or vias and finally extracting the required files needed for printing the PCB. Let us see the steps to design PCB using eSim.

13.1 Schematic creation for PCB design

In Chapter 5, we have seen the differences between schematic for simulation and schematic for PCB design. Let us design a PCB Layout for a 'constant 5V DC supply' circuit named as `7805VoltageRegulator`. First, we will simulate the circuit. Refer to Fig. 13.1 for the schematic used for simulation. After satisfying simulation results, we will move to PCB design. For this, we will remove the Source(s), Probes (plot_v , plot_db etc.), and global labels connected `solely` for the purpose of viewing simulation plots conveniently.
Connectors are the physical components that are used to interface/connect the board to external peripherals or sources.

13.1.1 Removing components required for simulation from the schematic

- We will remove the components which were placed for simulation purpose only.
- Components that will be placed on the board need to be added in the schematic.
- Modify the circuit schematic as shown in Fig. 13.2. The two pin female connector (`Conn_01x02_Female`) is placed for the taking the 5V output supply meanwhile a 2

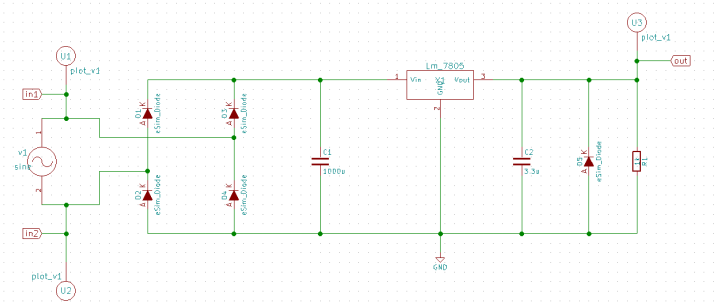


Figure 13.1: Schematic for simulation of the constant 5V DC supply circuit

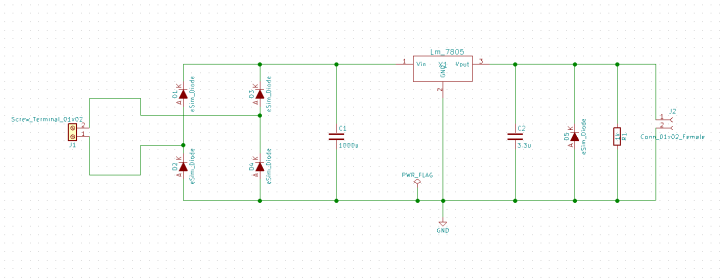


Figure 13.2: Schematic after adding connectors and removing the probes and sources.

pin Screw Terminal (`Screw_Terminal_01x02`) is used to transmit the input signal on board. Do the annotation and test for ERC. Refer to Chapter 5 to know more about basic steps in schematic creation.

13.1.2 Mapping of components using Cvpcb

- Once the schematic for PCB Design is created, one needs to map each component in the schematic to the appropriate footprint. The tool `Cvpcb` is used for this.
- Cvpcb can be launched by clicking the icon `Run Cvpcb to associate footprints and components` in Eeschema or by going under the `Tools` menu and selecting `Assign Component Footprint` option.

13.1.3 Familiarising with Cvpcb Window

- I. When one opens `Cvpcb` after annotating and running ERC on the schematic intended for PCB Design a window as shown in Fig. 13.3 will be obtained. The Toolbar for using Cvpcb will be available in the top-most left corner.
- II. The left pane has a list of all footprint libraries in the database.

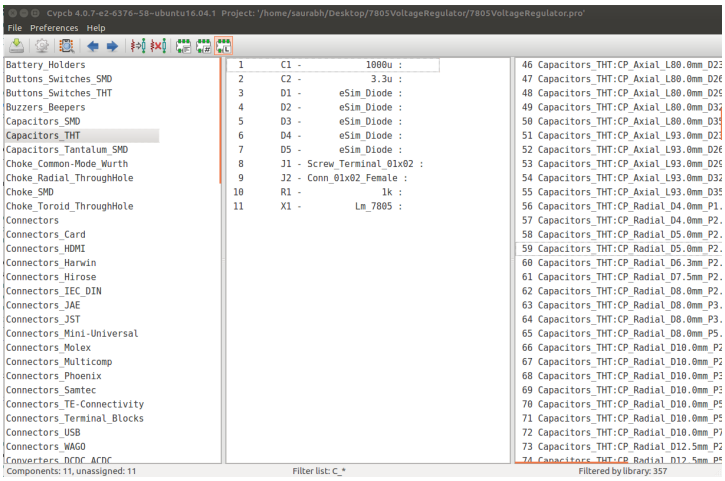


Figure 13.3: Cvpcb window

Figure 13.4: Cvpcb Toolbar

- III. The middle pane displays the list of components present in the schematic and if any footprint is assigned/associated to them.
- IV. The right pane has a list of available footprints for each component depending upon how of libraries.

Cvpcb Toolbar

Some of the important tools in the toolbar are shown in Fig. 13.4. They are explained below (Order of operation should ideally be from RIGHT to LEFT):

1. Filter footprints list by library : We recommend the use of only this as a filtering method if you are completely new to eSim and/or PCB Design as it narrows down footprints based on libraries of the type of the component. When a filter is selected, it's icon will be highlighted in light red color as seen in Fig. 13.4.
2. Filter footprints list by pin count : This will filter the footprints based on number of pins the footprint has. This can be used to narrow down your search after sorting footprints by their library type.
3. Filter footprints list by keyword - This filters the footprints in the database based on keywords.
4. Automatic footprint association - Perform footprint association for each component automatically. Footprints will be selected from the list of footprints available.

Note: This method of association is not recommended at all.

5. Delete all associations - Delete all the footprint associations made. This will erase all your association till now so be very careful in selecting this.
6. Select next unlinked component: Using this you can go to the next component in the list of components for associating a footprint.
7. Select previous unlinked component: Using this you can go to the previous component in the list of components for associating a footprint.
8. View selected footprint - View the selected footprint in 2D. See Sec. 13.1.4 for more details.
   Before clicking on this, make sure that a footprint is selected. Order of this operation should be
   1. Selection of footprint library from the left-most pane
   2. Selecting a footprint from the right-most pane
   3. Click on `View selected footprint`
9. Edit footprint library table - One should familiarize themselves with Cvpcb first and then only choose to use this. This impacts the footprints that you can choose, so be careful before making any severe changes.
10. Save netlist and footprint files - Save the netlist and the footprints that are associated with it. One ought to save the association after having assigned proper footprints to all the components.

13.1.4 Viewing footprints in 2D and 3D

- To view a footprint in 2D, select the component for which you wish to view the available footprints, then select the library from left-most pane and now from the right pane and click on the desired footprint and click on `View selected footprint` from the menu bar. Let us view a footprint for `C1` from the `Capacitors_THT` footprint library. Choose C1 from the middle pane as shown, click on `Capacitors_THT` in the left-most pane and select the *View selected footprint* tool. On clicking the **View selected footprint** tool, the `Footprint` window with the view in 2D will be displayed. 2D view of the footprint `CP_Radial_D5.0mm_P2.50mm` is shown in Fig. 13.6.

- Click on the `3D Display` icon in the `Footprint` window, as shown in Fig. 13.6. A top view of the selected footprint in 3D is obtained. Click on the footprint and rotate it using the computer mouse to get 3D views from various angles. One such view of the footprint in 3D is shown in Fig. 13.7.

13.1.5 Mapping of components in the circuit

1. Click on `C1` from the middle pane. Choose the footprint library *Capacitors_THT* from the left pane and locate the footprint

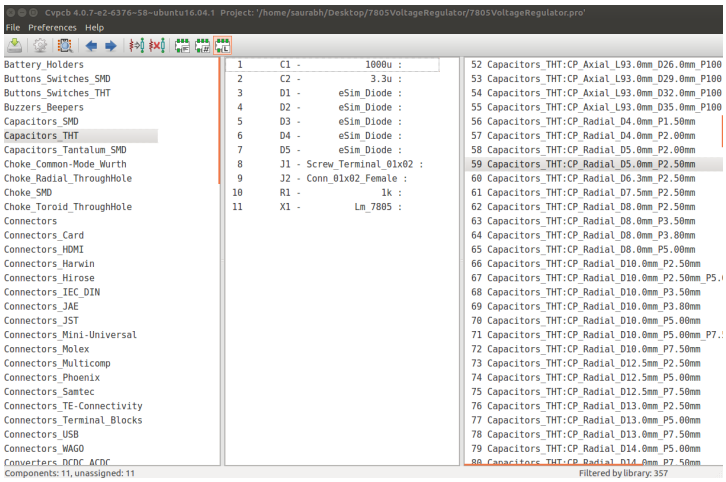


Figure 13.5: Viewing footprint for C1

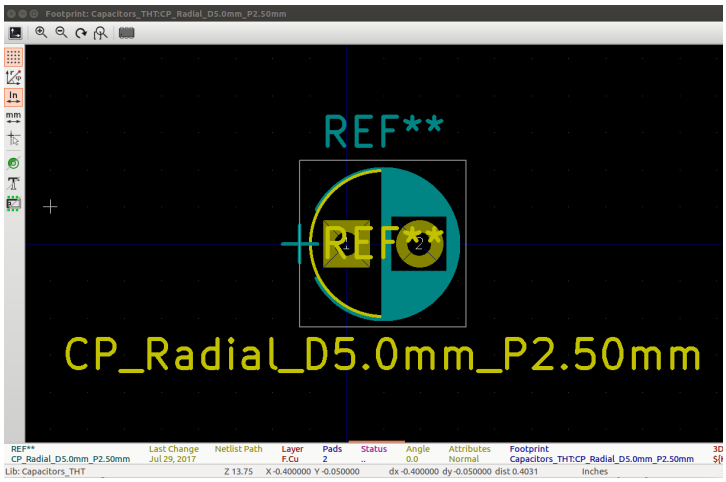


Figure 13.6: Footprint `CP_Radial_D5.0mm_P2.50mm`'s view in 2D.

`CP_Radial_D5.0mm_P2.50mm`. By double clicking on it, the said footprint will be assigned to `C1`.

2. Similarly choose the footprints per Fig. 13.8 where the footprint association for all the footprints is shown in Fig. 13.8. Save the footprint association by clicking on the `Save footprint association in schematic component footprint fields` tool from the `CvPcb toolbar`.

.

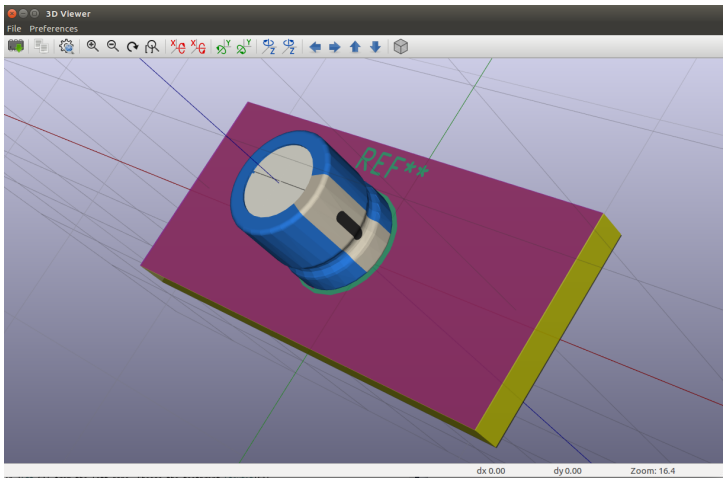


Figure 13.7: 3D view of the footprint

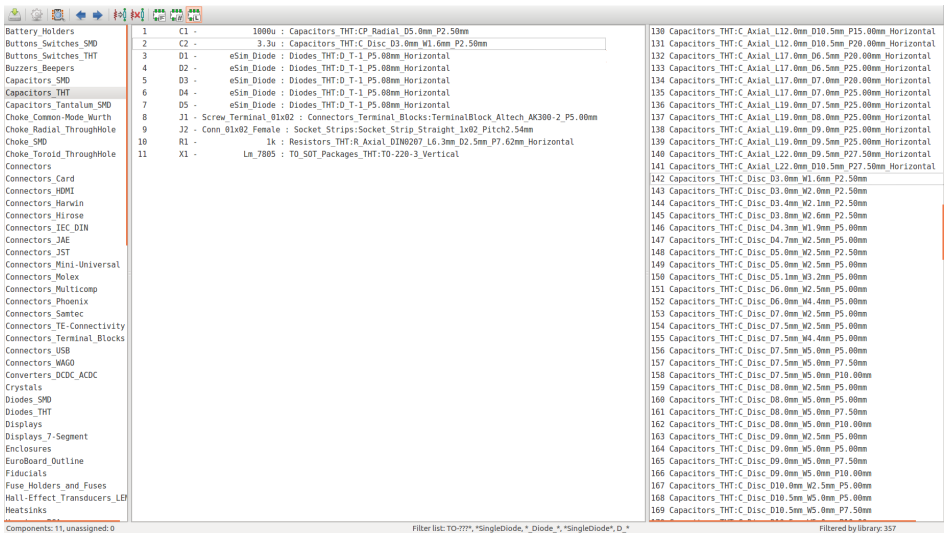


Figure 13.8: Footprint mapping completed for the circuit

13.1.6 Netlist generation for PCB

1. After having saved your footprint association, switch back to the Eeschema window and press Ctrl+S.
2. The netlist for PCB is different from that for simulation. To generate netlist for PCB, click on the *Generate netlist* tool from the top toolbar in Schematic editor.
3. In the `Netlist` window, under the tab *Pcbnew*, Select the option `Default format`. This is shown in Fig. 13.9. Click on `Generate` option.

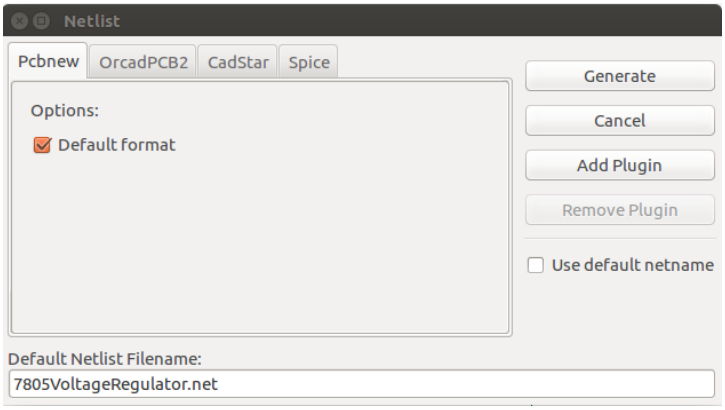


Figure 13.9: Netlist generation for PCB

4. Click on `Save` in the Save netlist file dialog box that opens up. Do not change the directory or the name of the netlist file. `Note that the netlist for PCB has an extension` *`.net`*`. The netlist created for simulation has an extension .cir`.

13.2 Creation of PCB layout

The next step is to place the footprints and lay tracks between them to get the layout. This is done using the *Layout Editor* tool. eSim uses `Pcbnew`, the layout creation tool in KiCad, as its layout editor.

13.2.1 Launching Pcbnew

1. To Launch the layout editor, `Pcbnew`, click on the `Run Pcbnew to layout printed circuit board` icon on the top right corner of the Schematic window.
2. Similarly, you can also click on `Tools`, and select the `Layout printed circuit board` option.
3. After doing either of the steps above, click **yes** on the *Confirmation Box* that will appear on the screen.

13.2.2 Familiarizing the Layout Editor tool

The layout editor with the various menu bar and toolbars is shown in Fig. 13.10.

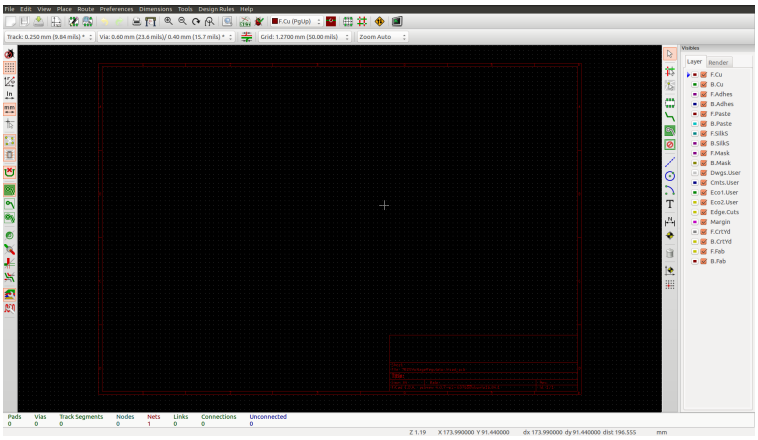


Figure 13.10: Layout editor with menu bar, toolbars and layer options

Top toolbar

Some of the important menu options in the top menu bar are shown in Fig. 13.10. They are explained below:

1. Save board - Save the printed circuit board
2. Plot - This enables users to import their design in Gerber, PDF, SVG and few more formats depending on the requirement.
3. Read netlist - Import the netlist whose layout needs to be created.
4. Perform design rules check - Check for design rules, unconnected nets, etc., in the layout.
5. Select working layer - Selection of working layer.
   Note: Selection of working layer can also be done from the `Layers toolbar` on the right-most side of the Pcbnew tool window.

13.2.3 Hotkeys

A list of few important hotkeys is given below:

1. F1 - Zoom in
2. F2 - Zoom out
3. Delete - Delete Track or Footprint
4. X - Add new track
5. V - Add Via
6. M - Move Item
7. F - Flip Footprint
8. R - Rotate Item
9. G - Drag Footprint
10. Ctrl+Z - Undo

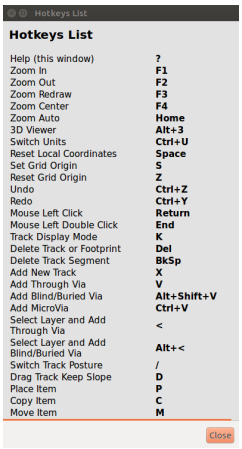


Figure 13.11: Default Hotkeys

11. E - Edit Item

The entire list of Hotkeys can be viewed by selecting *Preferences* from the top menu bar and choosing *List Current Keys* from the option *Hotkeys*.

13.2.4 Designing PCB layout for 7805VoltageRegulator circuit

Click on *Read Netlist* tool from the top toolbar of Pcbnew. Click on *Browse Netlist files* on the Netlist window that opens up. Select the `.net` file that was modified `after` assigning footprints. Click on *Open*. Now Click on *Read Current Netlist* on the Netlist window. The sequence of operations is shown in Fig. 13.12.

Arranging the footprints

1. After clicking on Read Netlist button and closing the Read Netlist window, the footprints that we assigned will appear on the Pcbnew screen in a cluttered fashion, stacked on top of each other as shown in Fig. 13.13
2. Let us separate these footprints and place them in proper orientation per the requirement of the circuit. Let us start with Screw_Terminal_01x02 footprint.
3. Right Click on the text `Screw_Terminal_01x02` in the Pcbnew window, and select `Footprint J1 on F.Cu.` and select the `Move` option from the list.
4. The footprint will be glued to the cursor and it can be placed in the location of one's choice by moving the cursor at the desired location and clicking once to place it there.
5. We have moved the Screw_Terminal_01x02 footprint to the left side of the Pcbnew window.
6. Once again right click on the text `Screw_Terminal_01x02` in the Pcbnew window,

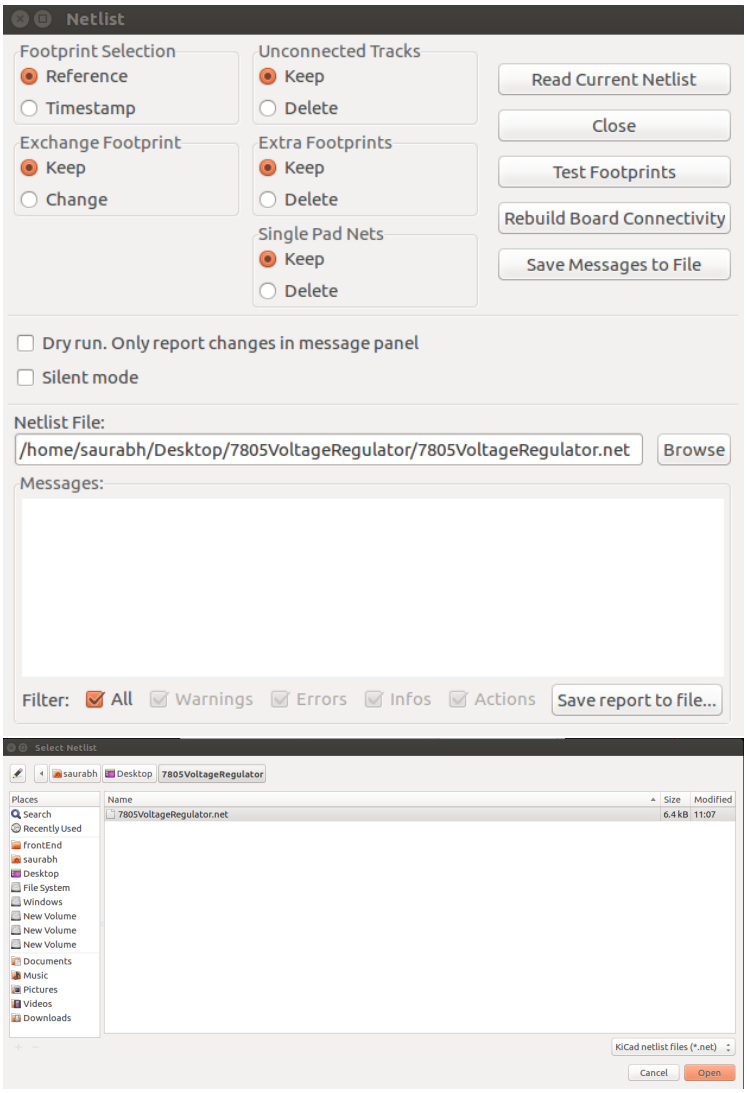


Figure 13.12: Importing netlist file to layout editor: 1. Browse netlist Files, 2. Choose the appropriate .net file, 3. Read Netlist file, 4. Close

and select `Footprint J1 on F.Cu.` and select the `Rotate +` option from the list as shown in Fig. 13.15

7. Using similar methods, we have moved and rotated all other footprints as shown in Fig. 13.16

Figure 13.13: Cluttered footprints

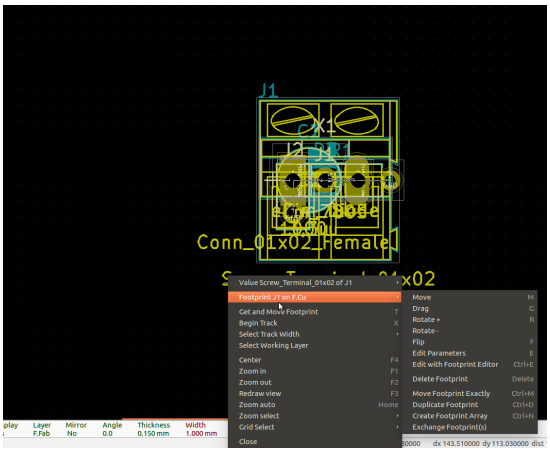


Figure 13.14: Moving the footprint

Setting Design Rules

1. Click on `Design Rules on top of the Pcbnew window, select Design Rules option from the drop down menu there.`
2. `Design Rules Editor` window will open, Under the `Net Classes Editor`, locate the `Track Width` box, erase the default value from the window placed under the `Track Width` box and enter 1.25 as the track width.
3. Click on `Global Design Rules`, under Minimum Allowed Values, locate `Min track width`, erase the default value and enter 1.25 in the data entry field on the right side of it as shown in Fig. 13.17. This will ensure that all tracks placed are of width 1.25mm and that when we perform Design Rules Check, the checks

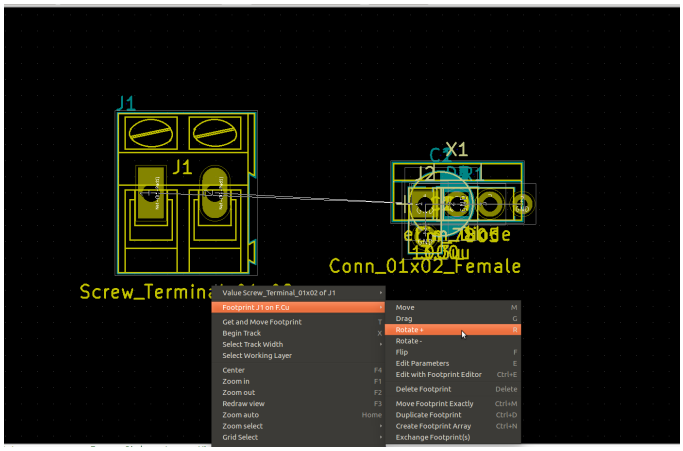


Figure 13.15: Rotating the footprint

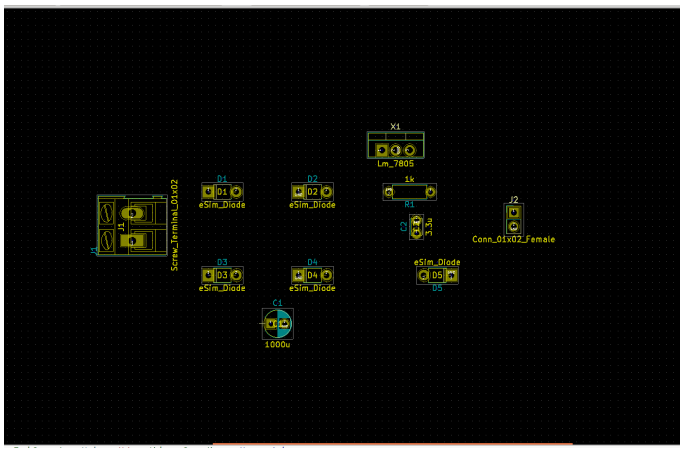


Figure 13.16: All footprints moved and rotated

will be made such that all tracks are of the width 1.25mm will be checked.

4. Click on Ok button and close the Design Editor Rules window.

Drawing the Board Outline

1. Board outline defines the physical dimensions of your board. After the fabricator is done placing tracks and other processes, he/she will cut your design from the copper cladded sheet or the material used per your choice, as per your board outline dimensions. Say, if your board outline is of rectangular shape with dimensions 80mmx50mm, the fabricator will cut the copper sheet of the said dimension. Its important to know that all the tracks(physical electrically conductive connections

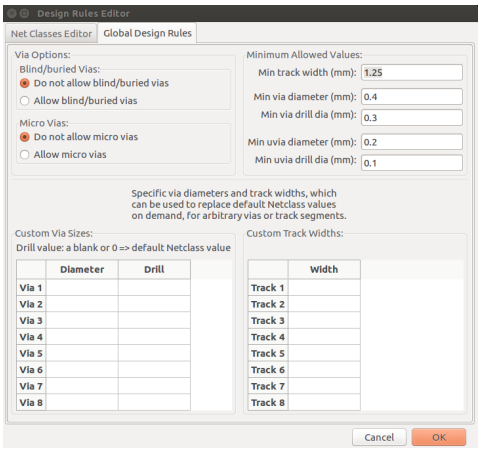


Figure 13.17: Design Rules Editor Window: Global Design Rules

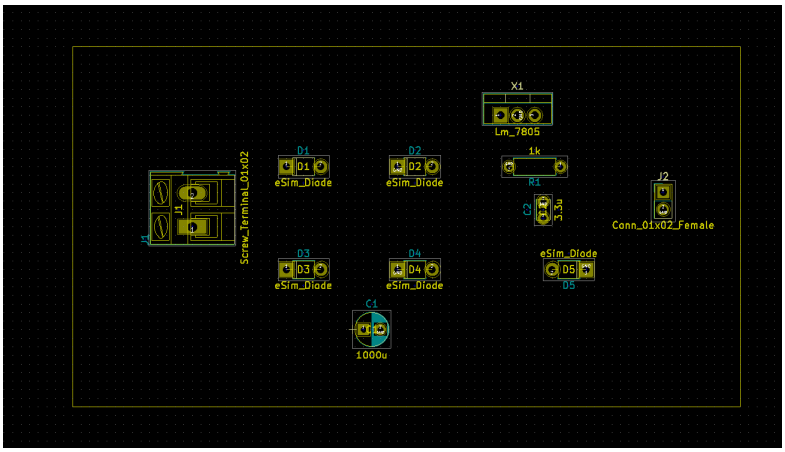


Figure 13.18: Drawing a board outline

between two nodes or points on a PCB) must lie inside this board outline.

2. We will also choose a board outline of rectangular shape. Select working layer as `Edge.Cuts` from the Layers menu on the far-right side of Pcbnew.
3. Click on `Place` from the top-left tool bar of the Pcbnew window and select `Line or Polygon`. A pencil icon will appear to be tied to the cursor.
4. Click once on the layout editor to start drawing the outline. Drag the cursor either horizontally or vertically. When it comes to the corner of the board, click once and drag the cursor in perpendicular direction. Do this till you reach the origin of the outline, and double click to finish drawing the rectangular outline. Completed outline is as shown in Fig. 13.18.

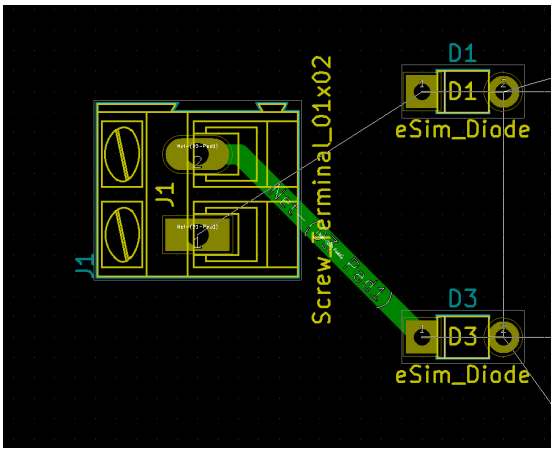


Figure 13.19: Track placed on B.Cu. between Screw_Terminal_01x02 connector and D3 Diode

Placing Tracks

1. Select the working layer as B.Cu from the Layers menu on the far-right side of Pcbnew.
2. Click on `Place` from the top-left tool bar of the Pcbnew window and select `Track`. A pencil icon will appear to be tied to the cursor.
3. The procedure to place a track between Node 2 of Screw_Terminal_01x02 to Node 1 of D3 diode, as shown in Fig. 13.18 is described in below steps.
4. Working layer is B.Cu., `Place Track tool` is selected earlier. Click the pencil icon tied to cursor on the Node 2 of Screw_Terminal_01x02 and drag the cursor till Node 1 of D3 diode and double click on Node 2 of D3. By double clicking the track will end at Node 2 of D3. Please refer Fig. 13.19 for the mentioned track being placed.
5. Similarly, all the tracks have been placed as shown in Fig. 13.20. Please note that tracks shown in green color are on B.Cu. layer whereas the tracks in red color are placed on the F.Cu. layer.

Performing Design Rules Check(DRC)

1. After tracks are placed, it is important that the design created by used should not violate any design rules set earlier.
2. Click on `Perform Design Rules check` button from the top menu bar of Pcbnew. DRC Control Window will pop-up as shown in Fig. 13.21.
3. Click on `Start DRC`, and observe if any messages/warnings appear in the `error messages` window at the bottom of the DRC Control window. If there are no

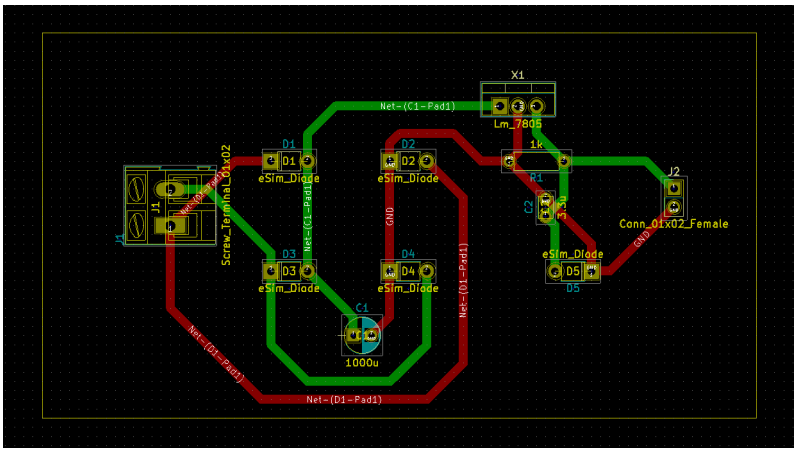


Figure 13.20: All tracks placed

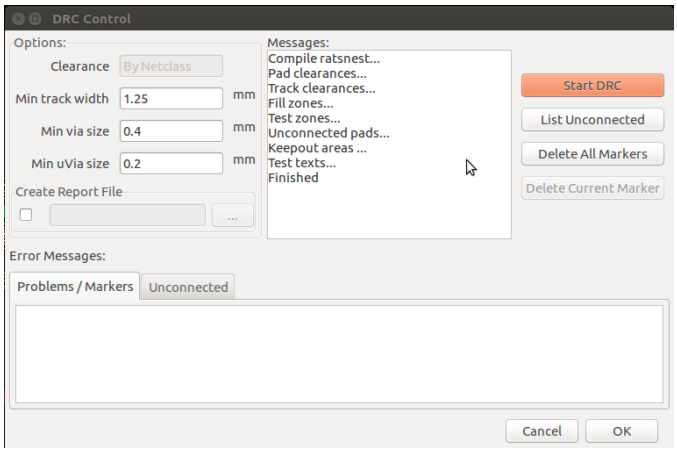


Figure 13.21: DRC Control Window

errors in the design present, there will not be any errors in the message window as shown in Fig. 13.21

Exporting the Design to Gerber format

1. Click on `Plot` tool from the top toolbar, select the Plot Format as `gerber` from the drop down menu of Plot Format.
2. Select the directory in which the user wishes to save the gerber files.
3. Select F.Cu, B.Cu, B.Silks, F.Mask, B.Mask, Margin and Edge.Cuts from the *Layers* on the left side of the Plot window as shown in Fig. 13.22.
4. After clicking on 'Plot' button, acknowledgment messages can be seen in the 'Mes-

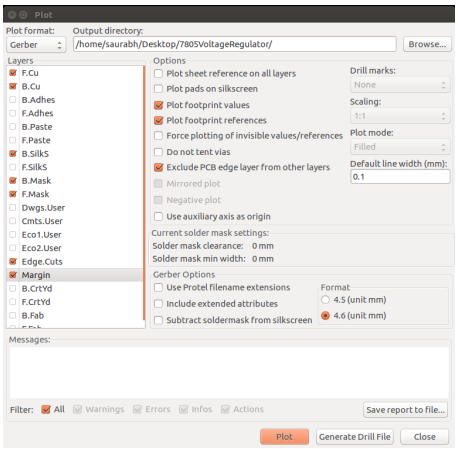


Figure 13.22: Plot Window before generating gerber files

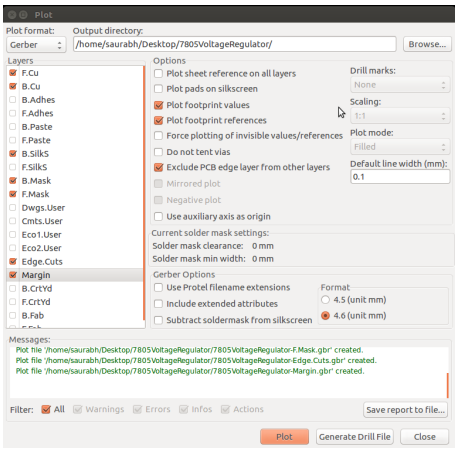


Figure 13.23: Plot window after generating gerber files

sages' window at the bottom of the `Plot` tool window as shown in Fig. 13.23.

Viewing the Gerber files generated

1. Open the terminal by Ctrl+Alt+T keys, and type `gerbview` and press enter. Gerbview tool of KiCad will open up. For windows OS users, search for `gerbview` application through Windows' search application option.
2. Click on File from top-left menu, and select Load Gerber File option.
3. Go to the directory where you have stored the earlier created gerber files as shown in Fig. 13.24 and click on Open.
4. The design created earlier will appear in the gerbview window as shown in Fig. 13.25.

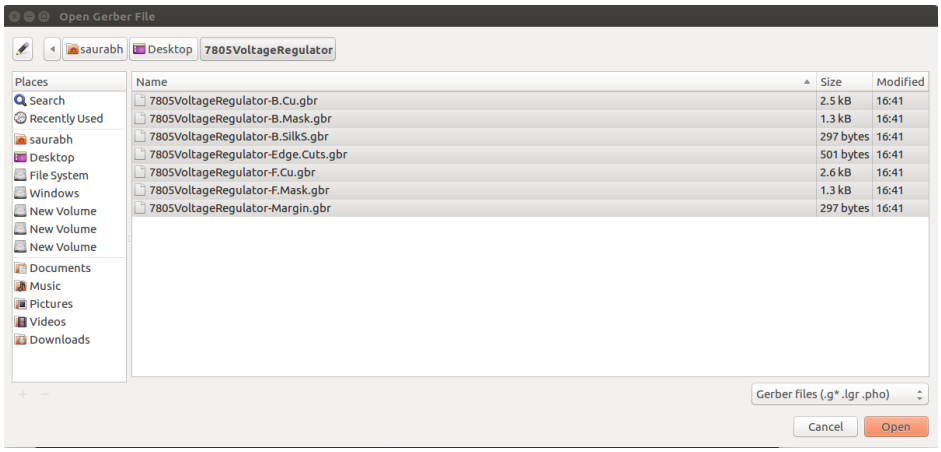


Figure 13.24: Loading gerber files in the gerbview

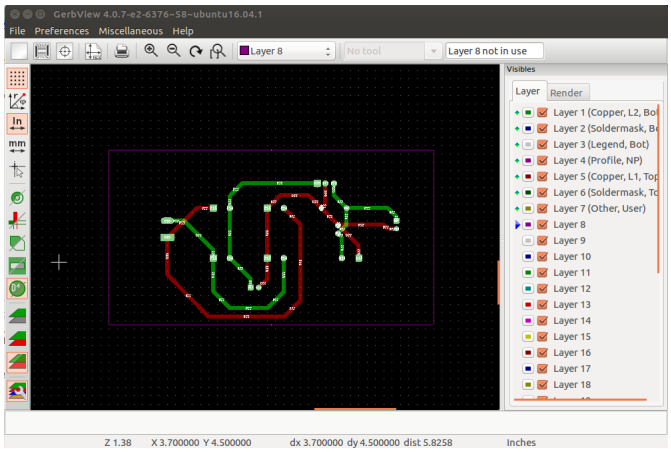


Figure 13.25: Observing the design in Gerber format

Chapter 14

Appendix

14.1 Appendix A

14.1.1 Backing up important data before uninstalling eSim

- Locate the folders : `SubcircuitLibrary` and `deviceModelLibrary` in the eSim installation directory, compress them in .zip or .rar format on your Desktop or some other location which does not contain eSim related files.
- The projects created and stored in eSim-Workspace will not be deleted when one uninstalls eSim, hence there is no need to backup these project files.
- Newly created subcircuits and device models should be backed up as explained above. A way to take backup of the subcircuit blocks (external outlook) which appears in the schematic editor (not to be confused with internal circuit of the subcircuit) is explained in the Subcircuit section of this manual.

14.1.2 Uninstalling eSim

For Windows OS users

- Locate the uninstaller `"uninst-eSim.exe"` from the directory where eSim is present, by default it is installed at C:/FOSSEE/eSim/
- Run the uninstaller executable and all the eSim related files and components **except** the projects created in **eSim-Workspace** will be deleted.

For Ubuntu Linux OS users

- Open terminal and go to the location where eSim is stored using the cd command.

- type the following and press enter :
  `$ ./install-eSim.sh --uninstall`

14.2 Appendix B

14.2.1 Pin types in KiCad

- **Input**: is a unidirectional input.
- **Output**: is a unidirectional output that can drive high or low.
- **Bidirectional**:can act as an input or an output. The I/O pins on microcontrollers are bidirectional.
- **Tri-state**:is an output that can drive high or low, but can also be placed in a high-impedance state where it floats. The 74125 is a chip with tri-state outputs (sometimes called three-state outputs: high, low, and high-impedance).
- **Passive**:is an unpowered connection, like resistors, capacitors and inductors.
- **Unspecified**: is, unspecified. User would use that when no other classification fits.
- **Power input**: is a pin where power comes in to a chip. Both the VCC and GND pins of chips would be classified as power inputs.
- **Power outputs**: are where power comes out of a chip. The outputs of voltage regulators are the most common example of this pin type.
- **Open collector output** pins: are outputs that can be driven low, but float otherwise. These are good for wired-AND connections where the output goes low if any of the attached open-collector pins goes low, but the output is pulled high by a pull-up resistor if all the pins are floating. The 7401 is a NAND gate chip with open-collector outputs.
- **Open emitter output pins**: can be driven high, but float otherwise. These are good for wired-OR connections where the output goes high if any of the attached open-emitter pins goes high, but the output is pulled low by a pull-down resistor if all the pins are floating.
- **Not-connected**:pins are pins which have no function. Think of these as package pins that are not bonded to the IC inside.

14.2.2 ERC Table

- User can actually decide what you want to be told about, by setting the table below accordingly.For instance if the user wanted the ERC to throw an error if an input was connected to an input then the user would change the topmost box from green (no message) to yellow (warning) or red (error) as shown in Fig. 14.1

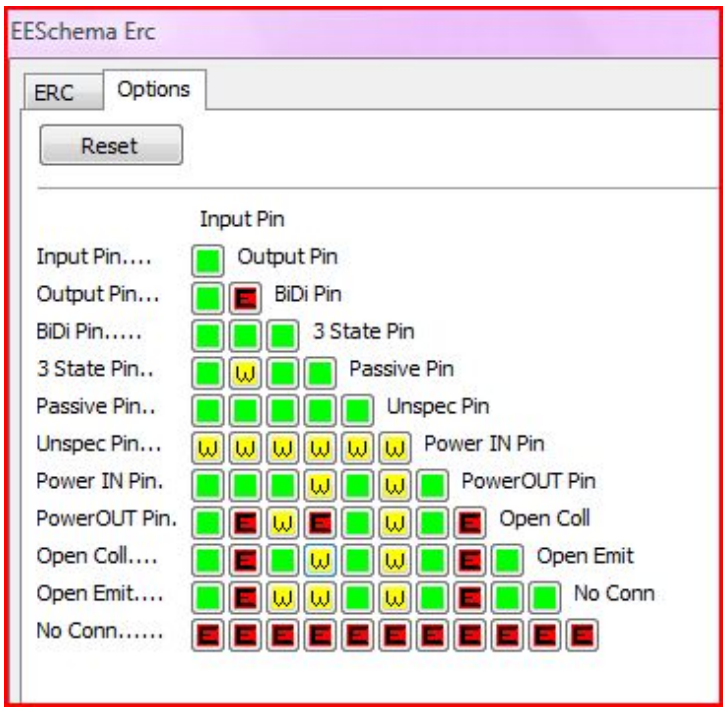


Figure 14.1: ERC Table

14.3 Appendix C

14.3.1 Shortcut keys in Schematic editor

- Open the schematic editor and press Shift and ? keys simultaneously. This will display the total shortcut keys, also called as Hotkeys. Please note that, if the shortcut key is related to a component, for example, changing its value or its orientation etc, then your cursor must be located on that component.
- **V**: This edits the Component's value.
- **R**: This rotates a component.

- **A**: This calls the place component tool through which you can add components in your schematic.
- **M**: This moves a component. After pressing M key, the component you chose will be tied to the cursor and you can place it anywhere in the schematic by clicking once on the schematic editor.
- **C**: This copies a component. After pressing C key, the component you chose will be tied to the cursor and you can place it anywhere in the schematic by clicking once on the schematic editor.
- **Ctrl + H**: This is to create a Global label. After pressing Ctrl + H keys simultaneously, the global label will be tied to the cursor and you can place it anywhere in the schematic by clicking once on the schematic editor.
- **W**: This calls the place wire tool through which you can add components in your schematic.

14.3.2 Shortcut keys in PCB editor

- The shortcuts that can be used in the Pcbnew Layout Editor .Open the Pcbnew Layout editor and press Shift and ? keys simultaneously.
- **X**: This calls the Place Track tool.
- **R**: This rotates a component.
- **B**: Fill or Refill All Zones.

14.4 Appendix D: ERC errors

- **ErrType(1)**: Duplicate sheet names within a given sheet. Review sheet names in hierarchy and remove duplicates.
- **ErrType(2)**: Pin not connected and no No Connect Symbol. Check the pin and wire overlaps or place No connect symbol if pin should be left unconnected.
- **ErrType(3)**: Pin connected to some others pins but no pin to drive it. This means there is a power input pin and there is no power connected to the pin. This is typically solved by adding a PWR FLAG symbol to the schematic.
- **ErrType(4)**: Conflict problem between pins. Severity: warning. Two pins connected but their function needs complementary signals (for ex. input -¿ output). You can have many power input pins connected to power output but no two power output pins should be connected together.

- **ErrType(5)**: Conflict Problem between pins, Severity: Error. This is because you can only have one output or power output on the same net.
- **ErrType(6)**: Mismatch between hierarchical labels and pins sheets. There is a mismatch between hierarchical label and existing pin sheet, try to re-import hierarchical pins to replace wrong pin sheet.
- **ErrType(7)**: A no connect symbol is connected to more than 1 pin. "No connect" symbol should be put at the end of the pin,and this pin should be left unconnected at all costs.
- **ErrType(8)**: Global label not connected to any other global label. There is a global label which has no pair in other sheet(s).
- **ErrType(9)**: Two labels are equal for case insensitive comparisons Review schematic labels to find possible duplicates, watch for similarity of large and small letters.
- **ErrType(10)**: Two global labels are equal for case insensitive comparisons. Review global labels in hierarchy to find possible duplicates, watch for similarity of large and small letters.

14.5 Appendix E: Checks to be done before Simulation in NGHDL

1. VHDL filename should be in small letters without any space or special characters (Underscore is allowed).
2. Entity name, architecture declaration and the VHDL file name should match exactly.
3. Port declaration can be either **std_logic** or **std_logic_vector**. No other declarations are allowed.
4. All **in** ports should be declared before **out** ports.
5. Maximum number of output ports that a VHDL entity under simulation can have is 64, provided the port names are not too lengthy to overflow the buffer.
6. Be extremely careful while dealing with Arithmetic Operations.
   (See **nghdl/Example/counter/counter.vhdl** for further reference)
7. If structural style is used, then the main entity of your VHDL code which is to be simulated, should be declared first in the file with inclusion of libraries for subsequent declaration of each supporting entity. See **nghdl/Example/full_adder/full_adder_structural.vhdl** for further reference.
8. Do not use `sudo` or root permissions while working with Mixed Signal Simulation and eSim.
9. If there are more than one VHDL file to be uploaded, then do it **one-by-one**, as described below:
   (a) Click on `NGHDL` button on eSim or type `nghdl/` in terminal. A new window will be opened.
   (b) Upload your first VHDL file and wait for the process to be completed.
   (c) Check if there are any Errors on the console. If possible, try to debug it and report your error and its solution to us. Otherwise, you can send the error to us. If there are no errors, upload your second model and repeat the steps mentioned above.
   You can upload as many models as required.
10. **Do not upload two or more VHDL files simultaneously :**
    `Add files` option allow you to use a smaller entity / subpart / submodule to support the main VHDL file. That is, a digital model will be generated corresponding to that file that has been browsed. The file that has been `added` to Nghdl upload window will only be placed along with the model under model's DUTghdl folder to support the model.
    Hence, `browsing` one file and `adding` several files won't create that many number models, but only model will be created corresponding to the browsed file.
11. Maximum number of NGHDL models that can be simulated at once is restricted to 512.
12. Next simulation should be started only after the completion of the previous/current

simulation. No two simulations should be executed at once.

13. While providing parameters to the adc-dac bridges that correspond to any NGHDL model, make sure that the **rise and fall delays of ADC and DAC bridges** are comparable to the simulation time parameters passed. Some examples are given below:

    (a) **Circuits involving both Analog and NGHDL Components** - If step time is 10 ms and simulation time is 200 ms, then DAC-ADC delays should be atleast 1 us.
    (b) **Circuits involving only NGHDL Components** - If step time is 10 ms and simulation time is 200 ms, then DAC-ADC delays should be atleast 500 us.
    (c) **CMOS inverter subcircuit [INVCMOS]** has a delay of 6 ms(can be changed by changing capacitor value inside the CMOS subcircuit). NGHDL model created, ADC and DAC bridges has rise and fall time 1 ns. You are simulating this circuit for 100 ms, it won't work. Increase the rise and fall delay for the ADC-DAC models from 1 nanosecond to 1 microsecond (1/1000th of the analog delay).

    In general **(a thumb rule to follow)**, delay value can be set to at least 1/500th part of the stop time.
14. If there is a need to use multiple instances of same NGHDL model in a given project and if even one of these instances need to have a different parameter value than the rest of the instances, then a separate NGHDL model needs to be generated with a different name.

14.6 Appendix F: Common Errors in NGHDL

Before concluding anything about NGHDL's working or about eSim's Mixed Signal Simulation, do check the console for outputs and logs. Kindly see the following errors on User's end that can be rectified there itself:

14.6.1 NGHDL Upload Errors

- **Error while opening NGHDL. Please make sure NGHDL is installed :**

  As the error indicates, NGHDL is not installed at all or there has been some error during installation which was not resolved effectively. However, if the installation was done correctly, then open a terminal and type :

  ```
  $ sudo ln -s <your_path_to_nghdl>/nghdl/src/ngspice_ghdl.py /usr/loc
  al/bin/nghdl
  ```

  Now type `nghdl` in terminal and check if the `NGHDL Digital Model Creator` window opens or not.

- **Error - select a *.vhdl file :**
  This type of error can occur due to a variety of underlying user issues. Check terminal from which eSim is launched for more detailed errors. However, following is a list of common mistakes that a user may face :

  – `Permission Denied` - Change the permission of `ngspice-nghdl` folder recursively to be used by all the users. To do so, open a terminal and type:

  ```
  $ sudo chown $USER:$USER -R $HOME/ngspice-nghdl/
  ```

14.6.2 Simulation Related Errors

- **GHDL compilation error :**
  The VHDL code itself is incorrect, as a result the corresponding model generated is incorrect and no simulation will be done. Kindly upload a correct VHDL code.

- **Warning : There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es) :**
  Such warnings are difficult to find in xterm outputs. However, it indicates that there is some logical error in your VHDL code regarding the data types / arithmetic operations / type conversions, etc. Kindly upload a logically correct VHDL code.

- **Unknown model name ⟨your_nghdl_model⟩ - Ngspice :**
  If this error occurs on xterm window while simulating with Ngspice, then check

whether the Ngspice engine used is of system built or that of the NGHDL. To check the same, follow the steps:

1. Open terminal and type : `$ whereis ngspice`
2. Go to the location where ngspice executable is found.
3. Now, check if this ngspice is a link to:
   `ngspice-nghdl/install_dir/bin/ngspice`
4. If it is not a link, then uninstall the system's installed ngspice and create a link to the above mentioned path by typing the command in terminal as:
   `$ sudo ln -s $HOME/ngspice-nghdl/install_dir/bin/ngspice/<previous_ngpsice_location>/ngspice`
5. If output of step 1 is empty, then run above command by replacing `<previous_ngpsice_location>` with `/usr/bin`

- **Unknown model name ⟨your_nghdl_model⟩ - KiCad-to-Ngspice :**

  If this error occurs during KiCad-to-Ngspice conversion, then check whether the uploaded VHDL file had any Uppercase characters or any other special characters in its filename. Note that only lowercase characters and underscores are allowed.

- **Warning : component instance `uut` of `xyz` is not bound :**

  This logical error is related to the VHDL code which is shown as a Warning and simulation plots may also appear. However, these plots are incorrect as the VHDL model itself is incorrect. Following are the examples that can cause this error:

  – If a vhdl file is saved as `dummycode.vhdl` and the entity and architecture are declared as for example, `codedummy` or `dummy` or something other than the name of the actual vhdl file itself, one will get the above error.

  – If structural style is being used and above error is seen, then kindly make the necessary changes by referring our example found at:
  `nghdl/Example/full_adder/full_adder_structural.vhdl`

  Kindly check with the GHDL documentation and online resources too are available regarding this GHDL error.

- **Warning : Too many analog/event-driven solution alternations**
  Warning : Convergence problems at node (net-*).
  . . . . . .
  Transient solution failed -
  doAnalyses: iteration limit reached :

  As the error says itself that if too many analog components are connected to digital model due to which there can be many analog alternations (or if too many

events are generated rapidly due to which the convergence would fail for those models), then the initial conditions won't converge and the transient solution will fail. Follow these steps to resolve it:

1. Open the corresponding project's cir.out file (`\*.cir.out"`).
2. Just below the **".control"** statement and above the **"run"** statement, insert a new line.
3. On this new line, type **"option noopalter"**.
4. Save the file and exit.
5. Run the simulation once again.

- **doAnalyses: TRAN : Timestep too small; time = ... , timestep = ...; trouble with node ... run simulation(s) aborted :**

  As the error says :

  – Check if the time-step is too small as compared to the stop-time.
  – If the time-step is of the same order(scalable) as that of the stop-time, then check the delays of the adc-dac bridges and make them comparable to simulation stop-time.
  – If the delays are comparable, then check the parameters provided to the digital and analog models and set them appropriately.

- **chmod : changing permissions of 'any_file': Operation not permitted :**

  This error, displayed on xterm by Ngspice, is similar to that of `Permission Denied` error. Kindly refer **section 13.4.1** for the same.

14.6.3 Model Deletion

If you want to delete a model from your file system or is deleted due to unavoidable circumstances, perform the following steps (to ensure consistency within the software) :

1. Delete the folders found at path :
   - `ngspice-nghdl/src/xspice/icm/ghdl/<your_model>/`
   - `ngspice-nghdl/release/src/xspice/icm/ghdl/<your_model>/`
2. Delete the name and description of that model from the files :
   - `ngspice-nghdl/src/xspice/icm/ghdl/modpath.lst`
   - `eSim_Nghdl.lib`

Realign the content of these files by removing any extra spaces or empty lines found between any two subsequent remaining model names after deletion.

3. Delete the file found at path :
`eSim-version/src/modelParamXML/Nghdl/<your_model>.xml`

14.7 Appendix G: References

1 A. S. Sedra and K. C. Smith, Microelectronic Circuits - Theory and Applications. Oxford University Press, 2009.

2 K. M. Moudgalya, "Spoken Tutorial: A Collaborative and Scalable Education Technology," CSI Communications, vol. 35, no. 6, pp. 10–12, September 2011, available at https://spoken-tutorial.org/CSI.pdf.

3 (2020, March). [Online]. Available: https://www.scilab.org/

4 (2020, March). [Online]. Available: https://scilab-test.garudaindia.in/scilab_in/ https://scilab-test.garudaindia.in/cloud

6 K. Kannan and K. Narayanan, "Ict-enabled scalable workshops for engineering college teachers in india," in Post-Secondary Education and Technology: A Global Perspective on Opportunities and Obstacles to Development (International and Development Education), R. Clohey, S. Austin-Li, and J. C. Weldman, Eds. Palgrave Macmillan, 2012.

7 (2020, March). [Online]. Available: https://esim.fossee.in

8 (2020, March). [Online]. Available: http://www.aakashlabs.org/

9 (2020, March). [Online].
Available: http://en.wikipedia.org/wiki/Electronic_design_automation

10 (2020, March) Synaptic Package Manager Spoken Tutorial. [Online]. Available: https://www.spoken-tutorial.org/tutorial-search/?search_foss=Linux&search_language=English

11 (2020, March). [Online]. Available: https://www.kicad-pcb.org/

12 (2020, March). [Online]. Available: ngspice.sourceforge.net/

13 (2020, March). [Online]. Available: http://scilab.in/

14 S. M. Sandler and C. Hymowitz, SPICE Circuit Handbook. New York: McGraw-Hill Professional, 2006.

15 J.-P. Charras and F. Tappero. (2020, March). [Online]. Available: https://docs.kicad-pcb.org/

16 Wayne Stambaugh (2020, March). [Online].
Available: https://launchpad.net/kicad/4.0/4.0.7

17 P. Nenzi and H. Vogt. (2020) Ngspice users manual version 31. [Online].
Available: http://ngspice.sourceforge.net/docs/ngspice-manual.pdf

18 K. M. Moudgalya, “LATEX Training through Spoken Tutorials,” TUGboat, vol. 32, no. 3, pp. 251–257, 2011.

19 (2020, March). [Online]. Available: https://www.spoken-tutorial.org/

References

[1] J.-P. Charras and F. Tappero. (2013, July). [Online]. Available: http://www.kicad-pcb.org/display/KICAD/KiCad+Documentation

[2] D. Jahshan and P. Hutchinson. (2013, July). [Online]. Available: http://bazaar.launchpad.net/~kicad-developers/kicad/doc/files/head:/doc/tutorials/