Summer Fellowship Report

On

Spice Simulations using Kicad nightly builds

Submitted by

Akshay NH and Athul MS

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

July 3, 2018

Acknowledgment

We are extremely thankful to Prof . Kannan Moudgalya for guiding and motivating us throughout the FOSSEE fellowship programme.We would also like to thank our mentors Mrs.Gloria Nandihal and Mr.Athul George for their immense support and advice.Moreover, we are grateful to our fellow friends Mudit Joshi and Ashutosh Gangwar for assisting us with their programming knowledge.Lastly,we extend our warm gratiutude to the managers and staff of FOSSEE for their co-operation and assistance.

Contents

Chapter 1

Introduction

1.1 Introduction

KiCad is an open-source software tool for the creation of electronic schematic diagrams and PCB artwork.KiCad can be considered mature enough to be used for the successful development and maintenance of complex electronic boards.KiCad does not present any board-size limitation and it can easily handle up to 32 copper layers, up to 14 technical layers and up to 4 auxiliary layers. KiCad can create all the files necessary for building printed boards, Gerber files for photo-plotters, drilling files, component location files and a lot more.Being open source (GPL licensed), KiCad represents the ideal tool for projects oriented towards the creation of electronic hardware with an open-source flavour.

Stable builds Stable releases of KiCad can be found in most distributions package managers as kicad and kicad-doc. If your distribution does not provide latest stable version, please follow the instruction for unstable builds and select and install the latest stable version.

Unstable (nightly development) builds Unstable builds are built from the most recent source code. They can sometimes have bugs that cause file corruption, generate bad gerbers, etc, but are generally stable and have the latest features.

Chapter 2

Structure of the Signal Processing toolbox

The toolbox has two types of files:

2.1 Visible Files

- loader.sce
- builder.sce
- demos
- etc
- jar
- cleaner.sce
- testfile.sce
- unloader.sce
- README.md
- help

The primary step that one has to do, to use this toolbox for the first time is that , one has to essentially go through the README.md file of the toolbox. This file has the text providing the guidance on how to use the toolbox.

2.1.1 loader.sce

Its basically a file that calls for the function FOSSEE_Signal_processing_toolbox.start present in the etc folder. This has to be executed first on opening the toolbox, using the command "exec loader.sce". in the scilab console.

2.1.2 builder.sce

This file builds the macros,help and the loader.sce files. Type the command "exec builder.sce"in the scilab console to execute this file. Both these have to executed every time, to load the toolbox for usage.

2.1.3 demos

This folder includes the examples taken from `https://in.mathworks.com/help/signal/examples/dft-estimation-with-the-goertzel-algorithm.html`, which is function used for DFT Estimation with the Goertzel Algorithm.

2.1.4 macros

Its the main folder of the toolbox, which has all the signal processing functions of the toolbox.

2.1.5 etc

Its a folder which consists of FOSSEE_Signal_processing_toolbox.start and FOSSEE_Signal_processing_toolbox.quit files. These files are used to start and exit files of the toolbox.

.start file It will run a script when loader.sce is run from root toolbox directory. It will run loader files in all of the directories and link to important library.

.quit file It will run a script when unloader.sce is run from root toolbox directory. It will unlink all of the important libraries.

2.1.6 jar

This folder has scilab_en_US.jar file. This file is basically a Java Archive package file format typically used to aggregate many Java

class files and associated metadata and resources (text,images etc.) into a file for distribution.

2.1.7 cleaner.sce

This file is generated by builder.sce. On executing this file, we actually delete the loader.sce and the unloader.sce files. One caution to the users is that **do not edit this file.**

2.1.8 testfile.sce

This file is used to validate all the functions present in the toolbox macros. This file is executed in the Travis CI (Continuous Integration).

2.1.9 unloader

It unloads the toolbox.

2.1.10 help

It contains the help files of all the functions present in the toolbox(macros) as .xml files.

2.2 Hidden Files

These files have to included in the git repository before pushing it to the github. To view these files in the git repository use the command: "ls -a"on the terminal in the git repository directory.

- .gitignore
- .travis.yml

2.2.1 .gitignore

This file enables the user to explicitly tell git to ignore certain files. In our toolbox, we have included just a statement "*~", indicating that all the file names having the sign "~"("tilde") in the end(or suffixed) are ignored by git (when intentionally gone untracked).

2.2.2 .travis.yml

This file is included to facilitate the builds we can trigger in Travis CI.
Travis CI actually provides a default build environment and a default set of steps for each programming language. We can customize any step in this process in .travis.yml. Travis CI uses .travis.yml file in the root of our repository to learn about our project and how we want our builds to be executed. .travis.yml can be very minimalistic or have a lot of customization in it.

```
1 language: scilab
2
3 before_install:
4  - sudo apt-get install scilab
5
6
7
8
9 script:
10 - scilab -nw -f testfile.sce
```

Figure 2.1: .travis.yml file

The above figure shows the different sections involved in the .travis.yml file.
So each time when a build is triggered by the user, tracis engine actually downloads scilab (as per before_install section) and then a new scilab instance is opened in the terminal without any gui (-nw flag) and executes testfile.sce in the toolbox (because of the -f flag used here).

Reference

- http://docs.kicad-pcb.org/stable/en/getting_started_in_kicad.pdf
- https://github.com/KiCad/kicad-source-mirror
- https://github.com/FOSSEE/eSIm-Kicad-Simulations
- https://forum.kicad.info/
- http://www.ecircuitcenter.com/Basics.htm
- http://www.ecircuitcenter.com/SPICEsummary.htm