User Guide for Circuit Tolerance Analysis Tool

Yichen Zhang
Supervisor:

5th August, 2021

Contents

1 Introduction

Good circuit design is more than simply calculating the "correct" component values, it is ensuring that the circuit behaves properly when it has been manufactured in large quantities! Part of that evaluation is ensuring that the circuit performs adequately within the bounds of production tolerances. Student projects, by their nature, are often one-off designs, and students have the ability to adjust component values to account for outputs that are not quite what was expected. In a production setting, this is not a feasible strategy.

This project will produce a front-end for the Spice circuit simulator that supports the testing of designs across the range of possible production values of components. The input to the simulator will be a Spice simulation file (previously tested using a package

such as LTSpice), and a set of component tolerances. The project will then proceed to run the simulation multiple times, for the different component tolerances, to test out the impact of these on the final output.

2 Installation

- **Requirement**

  ngspice = 34

  python3

  Module: numpy, scipy, PyQt5 $\geqslant$ 5.15, matplotlib, pandas, django, quantiphy

  Optional Module: bs4, weasyprint, coloredlogs

- **Ngspice Configuration**

  Ngspice should be installed in directory *./Workspace*. This project has included a pre-compiled ngspice. If anything wrong is with it, you can compile it by yourself. It can be downloaded from sourceforge ngspice Git page.

  To compile ngspice34, make sure you enable the relative paths for spinit and code models.

  ```
  $ ./configure --enable-relpath
  ```

  The normal compile command for linux-64 bit installed is

  ```
  $ sudo apt install libxaw7-dev autoconf automake libtool
     libreadline6-dev
  $ ./autogen.sh
  $ mkdir release
  $ cd release
  $ ../configure --with-x --enable-xspice --enable-cider --with-
     readline=yes --enable-openmp --enable-relpath --prefix={Path
     -To-Workspace} --disable-debug CFLAGS="-m64 -O2" LDFLAGS="-
     m64 -s"
  $ make clean
  $ make install
  ```

  Where Path-To-Workspace is the complete path to the dirctory *Workspace*.

3 Flowchart

4 Project structure

This project is mainly written in python3 and for circuit analysis part, it is written in spice language.

4.1 main.py

Main function of the project.

Variable Description

- root: Root path of main.python3
- path: PATH variable of the system
- spiceinit: See Function init() in Logging.py for more information.

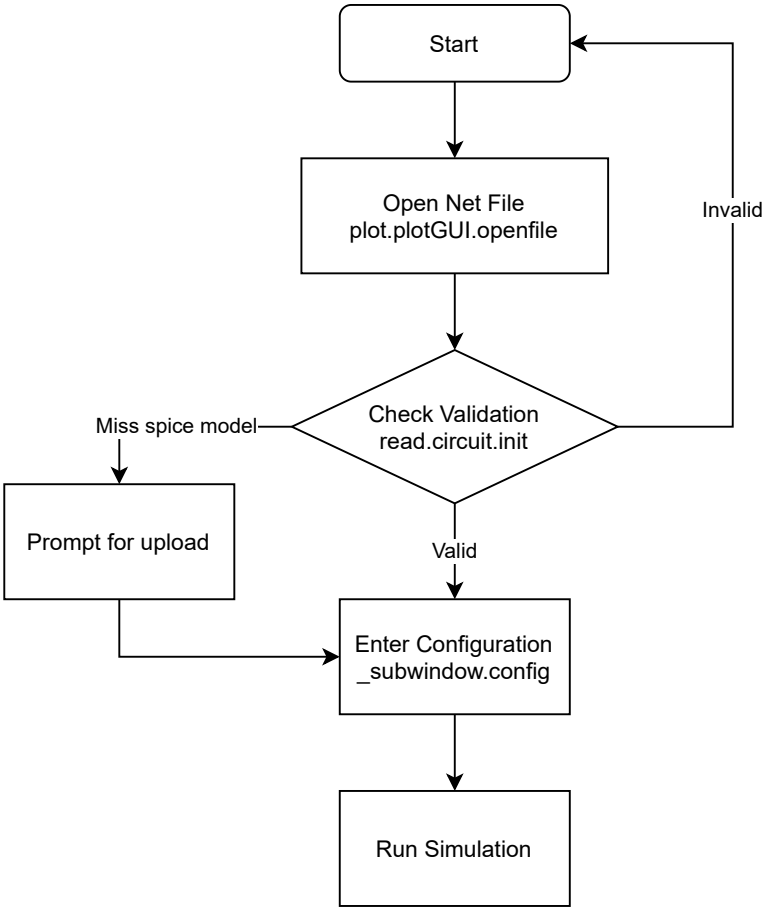


Figure 1

4.2 src/read.py

Read in a circuit file

- rm(*filename)
  Delete the given files. If the file does not exist, pass.
- Class C
  Capacitor.
  Attribute:
  - C.name: Capacitor name
  - C.c: Capacitance value
  - C.tol: Capacitor tolerance
- Class R
  Capacitor.
  Attribute:
  - R.name: Resistor name
  - R.r: Resistance value
  - R.tol: Resistor tolerance
- Class circuit
  Circuit class.
  Attribute:

- Initialize
  Give the file name of the netlist file. For example, circuit(TIA.cir)
- circuit.read()
  Read in the netlist and create two circuit files, *test.cir* and *run.cir*.

  Acceptable control command in netlist: ".model", ".subckt", ".global", ".include", ".lib", ".temp", ".ends", ".ac", ".probe"

  Caution: All commands between **.control** and **.endc** will be ignored.

  To run simulation correctly, the netlist must have a net named **"out"**!
- circuit.init()
  Initialize the circuit, check if any error in circuit file.

  It will call the ngspice to first perform an operating point analysis. During this process, it will give us all the component and net information. The first perform output data is stored in file *'op'* abd *'list'*.

  If the operation point analysis is passed, it will perform an AC, small-signal frequency response analysis and the output data is stored in file *'ac'*.

  Finally, it will read the information of the resistors and capacitors in the circuit from file 'list', generated during the first test. The components in the subcircuit will not be read.

  Return: two-element tuple (error_message, flag).

  flag=0: no error occurs or uncategorized error

  flag=1: include file error. Could not find include file

  flag=2: Unknown subcircuit error.

  If no error occur, both elements are 0.
- circuit.fixinclude(self, repl, mode)
  Fix the error, missing the include file or subcircuit file.

  Parameters:
  repl: String, Required. Uploaded include file
  mode: Int, Required. 1 or 2. 1 represents include file error, 2 represents subcircuit error.
- circuit.readnet()
  Read the nets of circuit of file 'op'. Nets in the subcircuit will not be read.

4.3 src/_write.py

Write control files. This is the sub-file for class circuit, defining new attributes.

- circuit.create_prerun()
  Create the control file *run_control _pre.sp*. It tests the circuit before the formal simulation. Measuring max and min gain does not require this function to run.
- circuit.create_sp(add=False)
  Create the main simulation control file *run_control.sp*
- circuit.create_wst()
  Create the control file *run_control_wst.sp* for the worst case simulation.
- circuit.create_step()
  Create the control file *run_control.sp* for the step mode.

– circuit.create_opamp()
  Create the control file *run_control.sp* for switching op amp.
– circuit.create_cmrr(add=False)
  Create the control file *run_control.sp* and *run_control_wst.sp* for CMRR mode.
  If add=True, control file *run_control_wst.sp* will not be created as it would have been run when first created.

4.4 src/_resultaly.py

Analyse simulation results and create report. This is the sub-file for class circuit, defining new attributes.

– circuit.resultdata(self, worst=False, add=False, mode=None)
  Analyse simulation results.
  Parameters:
  worst: Bool, Optional
    If true, it will analyse the worst case simulation data.
  add: Bool, Optional
    If true, this function read in data from last end read point.
  mode: Str, Optional
    This function would analyse data using importance sampling by default.
    Available choice: 'Opamp' and 'Step'
– f(x, miu, sigma, tol)
  Importance sampling function. It converts uniform distribution sampling data back to Gaussian distribution data, where *miu*(μ) is the mean of the distribution, *sigma*(σ) is the standard deviation and *tol* is the tolerance. This function calculates:

$$\frac{\exp\left[\frac{(x-\mu)^2}{2\sigma^2}\right]}{\frac{\mathrm{erf}(\frac{tol\cdot\mu}{\sqrt{2}\sigma\pi})-\mathrm{erf}(\frac{tol\cdot\mu}{\sqrt{2}\sigma\pi})}{2}\sqrt{2\pi}\sigma}$$

– circuit.report(mode=None)
  Generate html5 report. This function is run in another thread.

4.5 src/plot.py

Code for GUI interface.

Class plotGUI(QtWidgets.QMainWindow)

Main GUI interface.

– UI file: main.ui
  To edit this file, you can install module qt5-applications by

```
pip install qt5-applications
```

  Open the designer in Path-To-site-packages/qt5_applications/Qt/bin/designer and edit it.
  The varibles for the main widgets of this window is shown in the Figure 2 below.

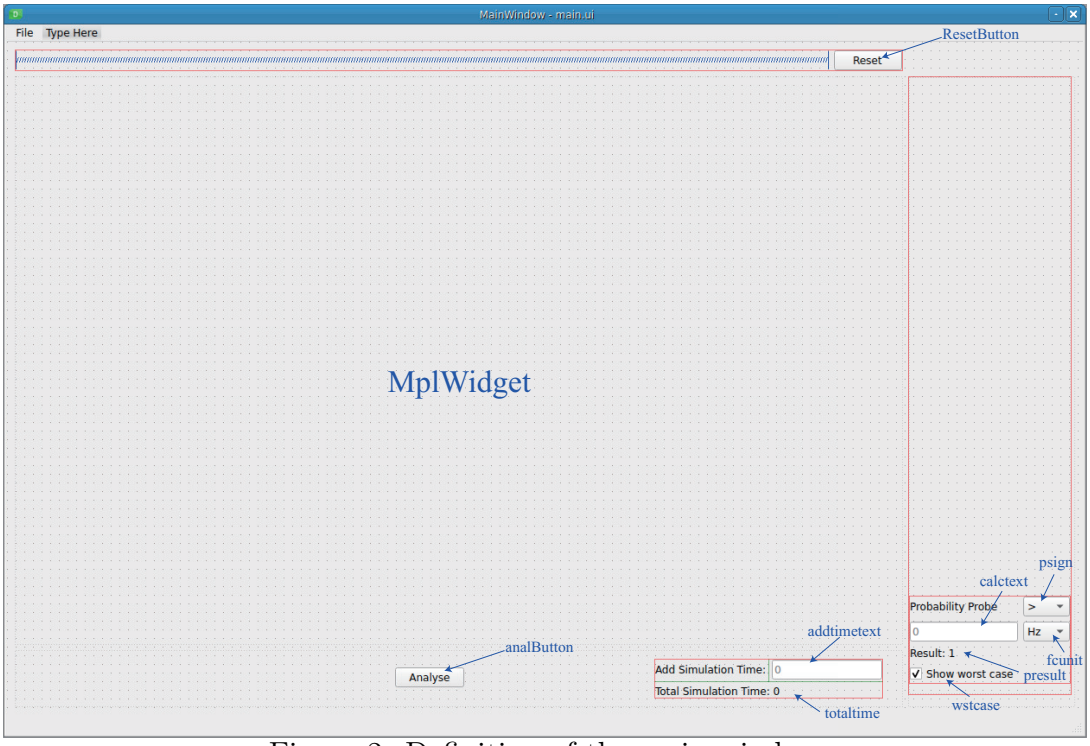


Figure 2: Definition of the main window

– openfile()
  Open a new netlist file. If there is a new opened, warn the user with a message box.
  It will first copy the selected file to a new folder, named with the filename and uploaded time. Then, call the function to read the file. If no error occur, call the function to initialize the circuit. If some error happens, it will pop out a window. Finally, show the configuration window.
– configCreate()
  Read in parameters from configuration window. Check the validity of the parameters and call specific function to create control files.
– configreject()
  If 'cancel' is clicked in the configuration window, this function will run if the signal is connected to this function.
– start_process(finishmode, runmode=0)
  Call ngspice to do simulation.
  Parameters:
    finishmode: String, Required
    Available options:
    runmode: int, optional
    runmode=0, run only run_control.sp
    runmode=1, run both run_control.sp and run_control_wst.sp
– kill()
  Kill the ngspice if *Cancel* is clicked while it is running.
– finishrun(mode)
  This function will be run when ngspice simulation is finished. It checks if any error occurs during ngspice processing. Then, it will call function postinit().

mode: String, Required. It determines which method the result data would be processed.

- postinit(mode=None)
  Do some post-initialization after running simulation. It will enable and disable certain buttons or widgets on the main based on the given mode.
- plot(mode=None)
  Plot the data on the main window based on the given mode.
- plotwst()
  Plot the worst case data.
- AddTime()
  Add more simulation time. This function will run when 'enter' is pressed.
- calcp()
  Calculate the y axis value. This function will run when 'enter' is pressed.
- analy()
  If the button analysis is clicked, show the configuration window.
- reset()
  If the reset button is clicked, clear the figure.

4.6 src/_subwindow.py

Define the Processing window and configuration window.

- Class processing(QtWidgets.QDialog)
  Show the processing dialog while ngspice simulation is running.
  ui file: src/processing.ui
- Class config(QtWidgets.QDialog)
  Configuration window definition.
  ui file: src/config.ui
  Variable definition of this window is shown in the figure 3 below.

4.7 src/MplWidget.py

The header file for the plotting area of both configuration window and main window.

4.8 src/Logging.py

Log file definition and some initialization of the tool.

- check_module(module_name)
  Check if the given module is installed. If the given module exists, return the module name. Otherwise, return none.
  Parameters:
  module_name: String, Required
- import_module(module_name, attr=None)
  Import the module. If attr is given, import the attribute from the module.
  Parameters:

module_name: String, Required

attr: String or list, Optional

- init(check=True)

  Do md5sum check of the software ngspice and check if the user defined configuration file for ngspice '.spiceinit' is correct.

  '.spiceinit' is located at the HOME dirctory typically. To edit the default content of this file, you can edit it in file Workspace/.spiceinit . Each time when the tool starts, it will check if these two file are the same.

  Parameters:

  check: Bool, Optional. Default value: True. Whether to perform md5sum check of ngspice.

  Return: string. The content in the file '.spiceinit' in the HOME dirctory. If the file does not exist, return False.

5 GUI Guide

5.1 Main Window

The interface of the main window is shown in figure 2.

The addtimetext line is enabled if the analysis is not step mode or Op amp mode.

The probability probe is also disabled under Op amp mode.

5.2 Configuration Window

The interface of the configuration window is shown in figure 3.

For tolerance analysis, you can enter the simulation options in the first tab and enter the tolerance of each component in the second tab *Tolerance*. Then click OK under either first or second tab.

For CMRR analysis, enter the simulation options under the third tab *CMRR* and confirm the tolerance under the second tab. Then click OK under the third tab. If you click under the tolerance tab, it will perform the simulation options of the first tab.

For Op amp alteration, enter the simulation options and click OK both under the fourth tab.

5.3 Report

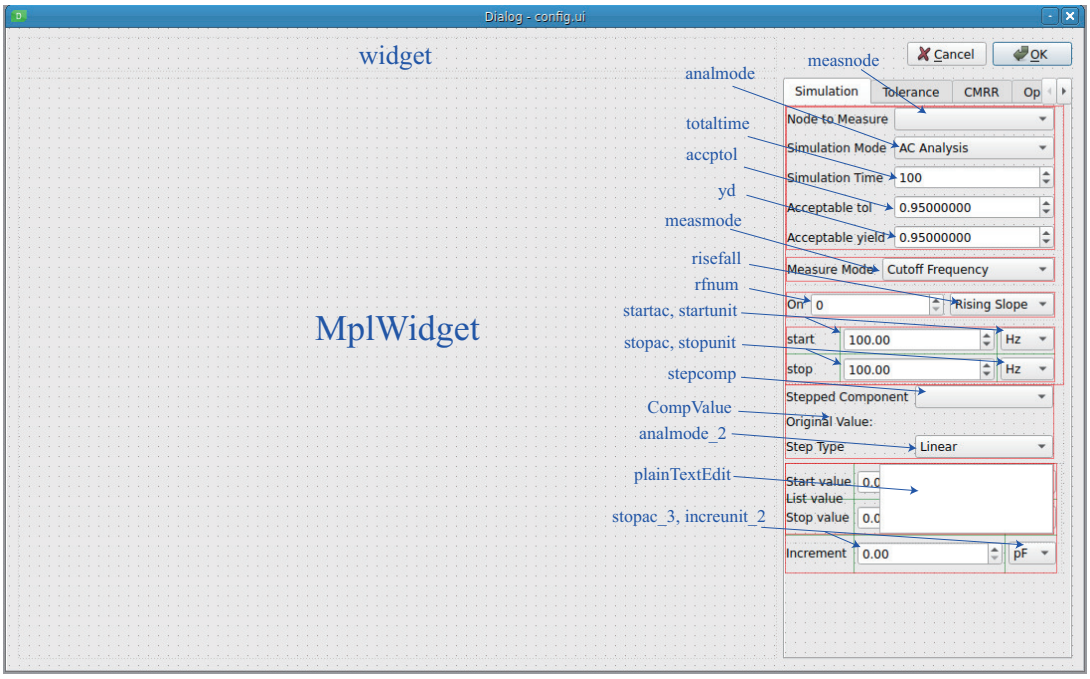
To see html5 report, under the directory src/report, run the below command.

```
python3 -m manage.py runserver
```

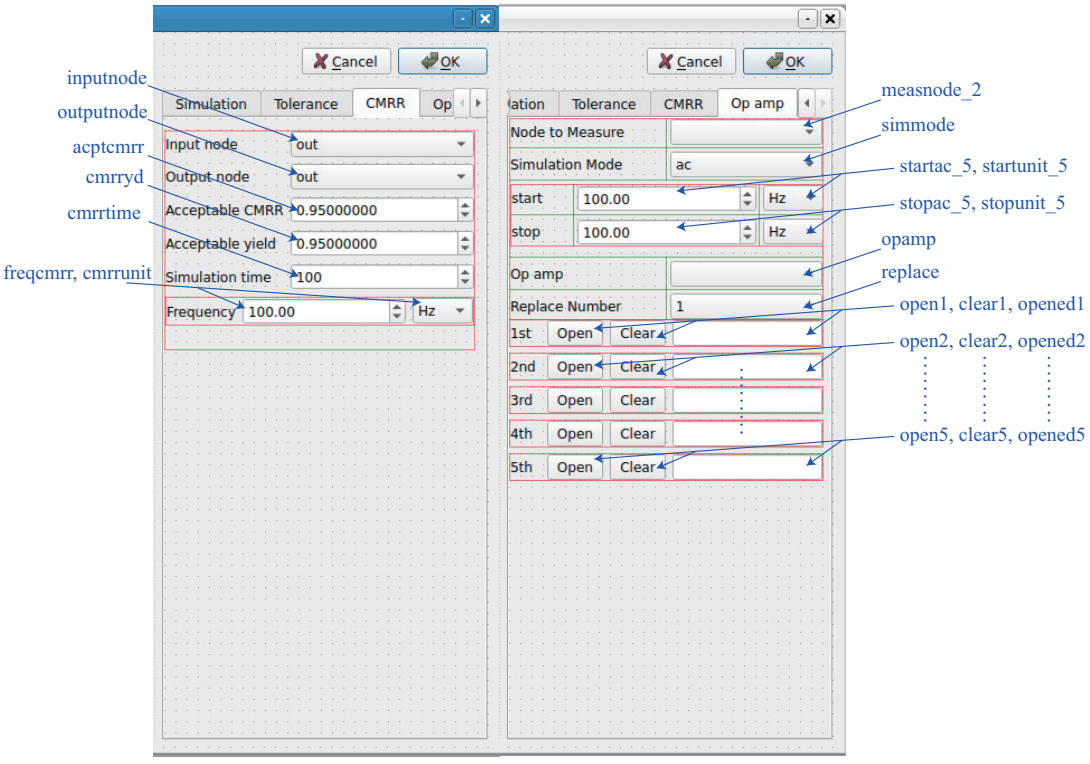
Then open 127.0.0.1:8000, you can see the result report.

Or you can find the html5 code file under src/htmlreport/templates/report.html

If you have the module weasyprint installed, you can find the pdf report under the circuit file base directory in folder Workspace.



(a) Simulation tab



(b) CMRR and Op amp tab

Figure 3: Configuration window definition