DESIGN DOCUMENT

for

SRS for AC CIRCUIT SOLVER

Prepared by :

Abhishek Gupta(2016CSJ0012)

Akshit Kansra(2016CSJ0001)

March 19, 2018

Contents

1 Introduction

1.1 Basic Design

In this project, we plan to implement the AC Circuit Solver, which will take netlist code as input, and output the SVG Image as Output. We plan to have SVG Image interactable and zoomable, i.e, if you hover on some element in the SVG, the current and voltage values will appear along with the phase differences across the component.

2 Usage/Purpose of Tools

2.1 FLEX

Flex is a scanner, it checks if a particular string follows a regex or not. Hence, it classifies them as tokens. We will have different types of tokens, like Component Names, Nets, Values, etc. which will be detected by the FLEX, if they follow the correct regex. Hence, if any component is not following the required pattern, FLEX will flag it, and it is the first stage of error detection. It also passes them to Bison to Check for Semantics.

2.2 BISON

It is a parser, whic is used to define the semantics of the program. We define our context free grammer in the Bison. Flex passes tokens onto Bison. If the semantics are not followed, flag will be raised and error raised. Otherwise, the code is converted to relevant C++ code.

2.3 SVG

It is the library/tool using which we render our component. A `.svg` file is generated, which will can be viewed in browser.

3 Implementation

3.1 SVG Generation

For generation of SVG, we first need to have all the components. We keep on adding components to an array/vector of a `C` structure `Component`. We then find out an suitable arrangement of each of the Nets. We plan to arrange them in such a manner that at any moment, each net has two neighbours. We first draw the components in between the intermediate nets, the all the other components are drawn. We will make some C functions, which when provided with the end points, draw the component in between them. We will try to make sure that no two components co-incide, by taking some offset values, and updating them whenever we draw a new component. Now after the circuit has been drawn, we will need to find voltages and currents across different components.

3.2 Solving the Circuit

We know the basic laws of `KVL` and `KCL` can be used to solve most of the circuits. These will be use in our Project also. Let us suppose we have N nodes. We can write `KCL` for each of the node. Each node will have some components. Hence, we can get a system of equations.
This system of equation may be represented as an NxN matrix plus an N item array. Each equation is represented by a row in the matrix, with values on each row representing the coefficients of each variable. The right-hand side of each equation is stored in the separate array. Before solving the equations, one will know the net current flowing into each node (zero in this case), and the voltage difference between pairs of nodes connected by voltage sources. Solving the equations will yield the voltage at each node and the current flowing through each voltage source.