上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

本科课程项目报告

UNDERGRADUATE COURSE PROJECT REPORT

课程： 设计自动化引论

COURSE: INTRODUCTION TO DESIGN AUTOMATION

项目题目：集成电路仿真器软件设计

PROJECT TILE: SOFTWARE IMPLEMENTATION FOR INTEGRATED CIRCUIT SIMULATION

学生姓名：刘耀天

学号 ：519021910090

邮箱 ：henry_liu@sjtu.edu.cn

任课教师：施国勇 教授

学院(系)：电子信息与电气工程学院 (微纳电子学系)

开课学期：2022 年 (秋季)

报告成绩：

目录

摘要

本课程要求学生用 C++独立编写一个具有类 SPICE 仿真器功能的小型电路仿真器。在施国勇老师以及助教的指导下，成功实现了一个具有 GUI、文件操作、Parser，以及支持线性以及非线性电路的 DC 扫描、AC 频率响应以及 TRAN 瞬态效应的仿真。最终通过了所有作业的所有 Testbench（有一次作业是因为理解出现了问题，导致 tb4 结果有误）。

1 绪论

电子设计自动化（英语：Electronic design automation，缩写：EDA）是指利用计算机辅助设计（CAD）软件，来完成超大规模集成电路（VLSI）芯片的功能设计、综合、验证、物理设计（包括布局、布线、版图、设计规则检查等）等流程的设计方式。

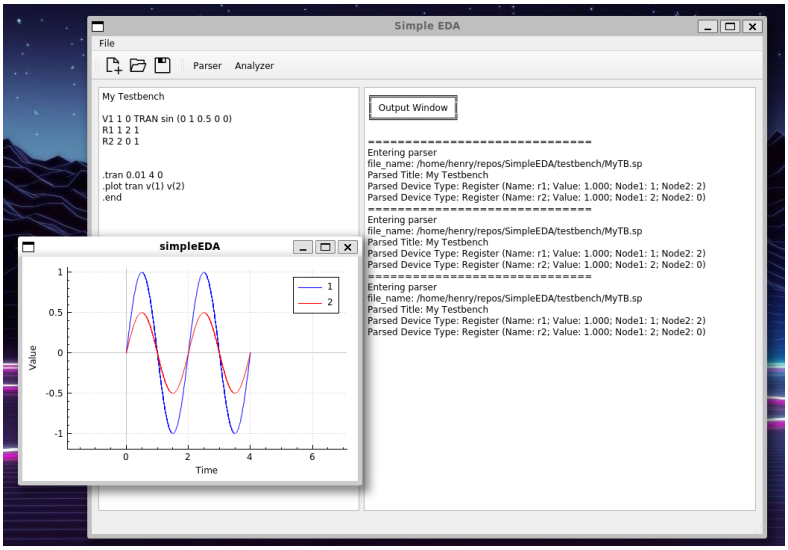
SPICE（Simulation program with integrated circuit emphasis）是最为普遍的电路级模拟程序，各软件厂家提供了 Vspice、Hspice、Pspice 等不同版本 spice 软件，其仿真核心大同小异，都是采用了由美国加州 Berkeley 大学开发的 spice 模拟算法。

本次课程项目也是在 SPICE 的基础上，实现一些基础的功能，

2 功能介绍

2.1 GUI

GUI 作为软件和人交互的接口，是非常重要的。本次课程项目使用 Qt5 实现了基本的文件读写、文件显示、按钮交互等功能。

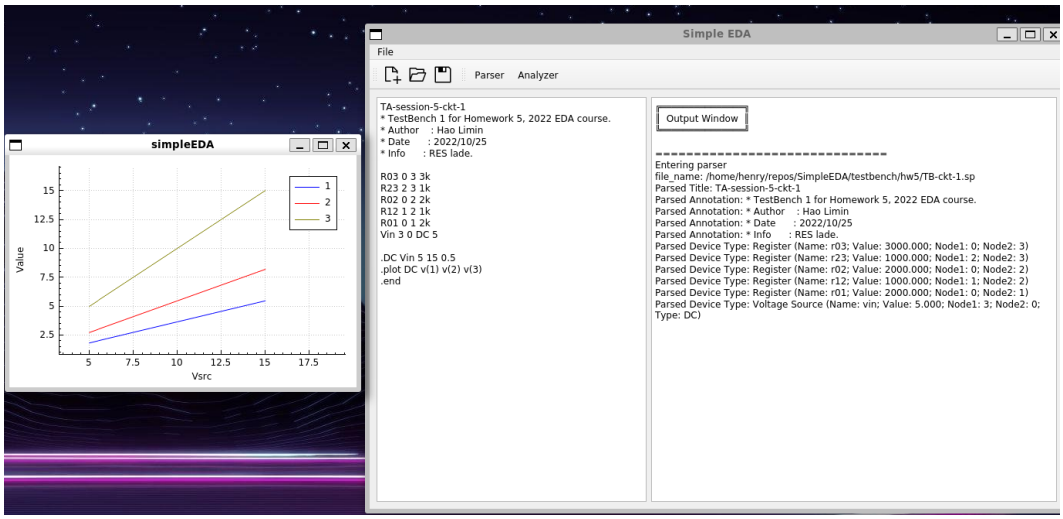


如上图所示，左侧边框为读写的文件，可修改并保存。右侧栏为 Parser 的输出栏。对

于.plot 命令，也能根据要求在对电路分析后给出响应的图像。同时图像支持使用鼠标键盘进行拖动和缩放。

2.2 DC 扫描

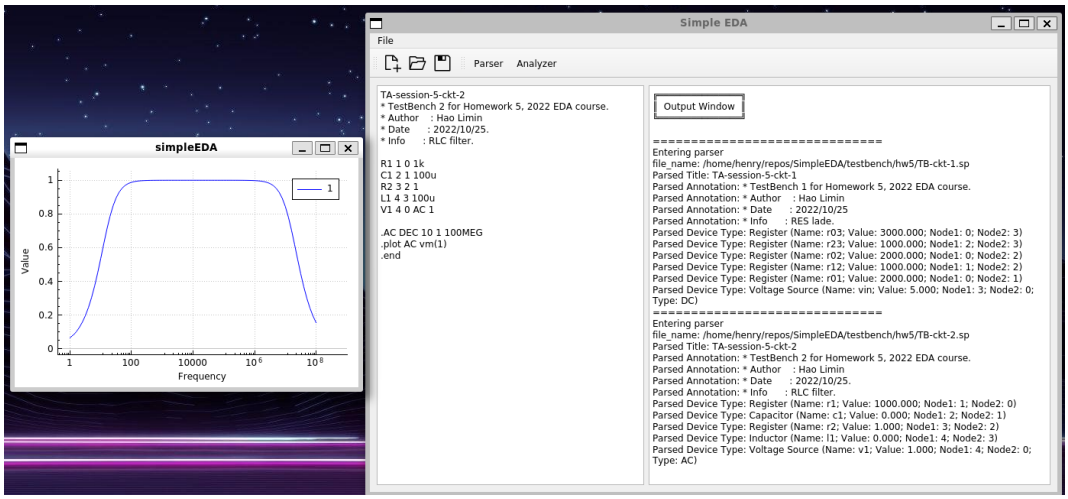
本程序支持对电路进行 DC 扫描。通过了助教提供的 DC 相关 testbench。



原始 testbench 只 plot 了 V(1)，这里为了测试同时 plot 多个结果，选择 plot 1，2，3 三个电压。

2.3 AC 仿真

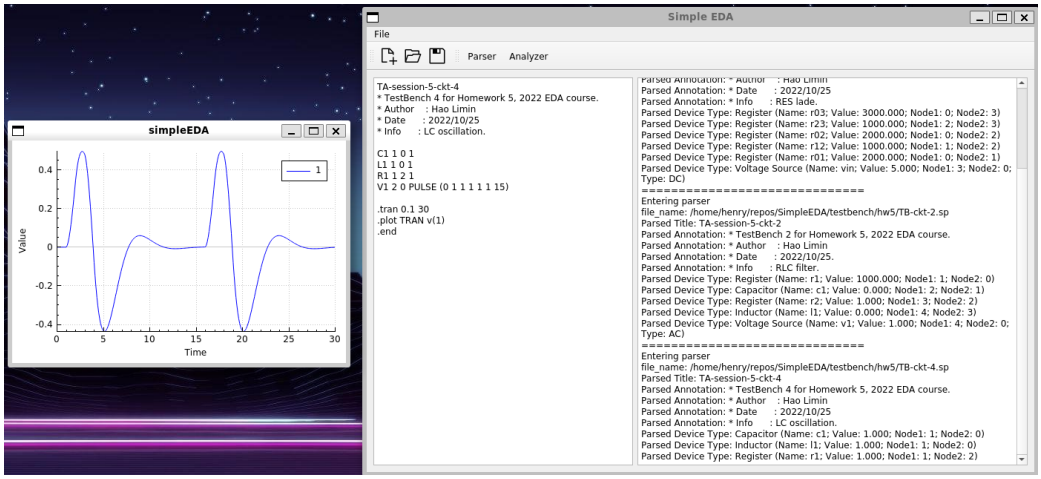
本程序支持对电路进行 AC 仿真。通过了助教提供的 AC 相关 testbench。



如图，是一个带通滤波器的 ac 仿真。

2.4 TRAN 瞬态仿真

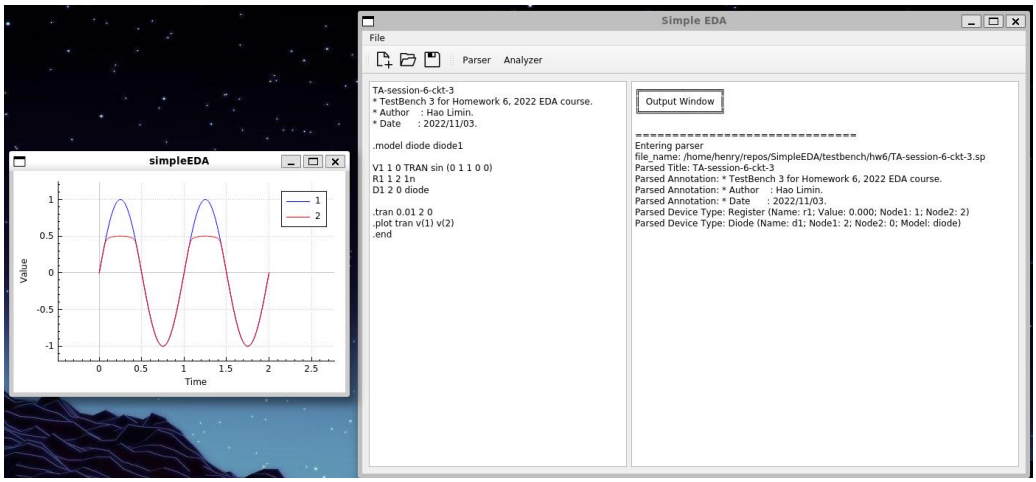
本程序支持对电路进行 TRAN 仿真。通过了助教提供的 TRAN 相关 testbench。



如上图，实现了对 RCL 振荡电路的仿真。该图像与助教提供吻合。

2.5 非线性器件——二极管仿真

该程序通过 NR 迭代来实现对二极管器件的仿真。并同时使用 Backward Euler 实现了对 Diode 的 tran 仿真。通过了助教提供的所有 3 个 testbench。



3 代码结构及实现细节

代码详见我的 GitHub repo：[Yaotian-Liu/SimpleEDA: A simple SPICE EDA written in C++ (github.com)]

由于本人觉得 qmake 有些繁琐，而且 qmake+make 的编译时间较长，最终选择了一个新

的开源程序 xmake，xmake 可以非常方便的管理项目并快速的编译（在本项目中可以实现编译时间减少 70%）。

3.1 代码结构

```
.
├── README.md
├── compile_commands.json
├── include
│   └── qcustomplot.h
├── lib
│   ├── libqcustomplot.so.2.1.0
│   └── libqcustomplotd.so.2.1.0
├── logs
├── simpleEDA
├── src
│   ├── analyzer
│   │   ├── acdc_analyzer.cpp
│   │   ├── analyzer.h
│   │   ├── analyzer_plot.cpp
│   │   ├── analyzer_type.h
│   │   ├── analyzer_utils.cpp
│   │   └── tran_analyzer.cpp
│   ├── main.cpp
│   ├── mainwindow
│   │   ├── images
│   │   ├── mainwindow.cpp
│   │   ├── mainwindow.h
│   │   └── mainwindow.qrc
│   ├── parser
│   │   ├── parser.cpp
│   │   ├── parser.h
│   │   └── parser_type.h
│   └── utils
│       ├── utils.cpp
│       └── utils.h
├── testbench
└── xmake.lua
```

其中 xmake.lua 为 xmake 的相关文件，其中包含了编译设置等。compile_commands.json 为 clangd 提供编译指导以正确的在 vscode 中实现代码的提示。

simpleEDA 为编译得到的二进制可执行文件。

include/ 和 lib/ 都为第三方（这里是 qcustomplot）相关的头文件以及动态链接库。

src/ 为主要的源码文件夹。其中主要有 main.cpp，负责 gui 以及主程序的 mainwindow/，实现 parser 的 parser/，实现 stamp、分析、求解、plot 的 analyzer/，以及一个 utils/。

程序主要通过 mainwindow 的按键操作唤起，然后去调用 parser 处理读取到的文件，然后再通过另一案件唤起 Analyzer，根据 SPICE 要求分析处理读入的电路。

3.2 实现细节

3.2.1 mainwindow

Mainwindow 主要由一个文件操作相关 Menu，两个 ToolBar，分别管理文件操作和 parser、analyzer，以及一个包含两个 TextEdit 的 QHBoxLayout 组成。其中一个 TextEdit 作为读入文件以及写文件的窗口，另一个 TextEdit 为只读，作为 parser 信息的输出。

3.2.2 parser

主要包含一个 Parser 类，由 QString 和 QRegularExpression 实现，其中负责处理读入到的每一行语句。对于每一个器件，都有一个 struct（事实上目前 c++的 struct 可以看作全为 public 的 class），这些 struct 都继承自一个 BaseDevice，包含器件的类别、器件名、器件的相关值以及器件相关的节点。Parser 就负责使用一个 Circuit 来存储所有读到的器件，这个 Circuit 由若干个 std::vector<Device-T>构成，可以较为方便的将所有电路信息传递到 Analyzer 进行处理。

同时 Parser 也要负责正确的解析并传递仿真控制信息，如.dc, .ac, .plot 等等，这些信息都作为 Parser 的私有成员保存，并在 Analyzer 初始化的时候传递。

Parser 也具有 debug 的能力，当检测到 SPICE 出现问题时，能给出相应的错误，目前支持的错误有:

1. 语法错误。如参数不够、参数过多等等。
2. 重复器件。若出现器件重复定义，会报错。
3. 不存在 GND。在本程序中，认为 “gnd”无论大小写等价与“0”节点。如果在.end 后，发现电路中不存在 GND，那么会弹窗报错。
4. 未规定仿真类型。若在未规定仿真类型的情况下使用相关语句，也会报错。

事实上我对目前的 Parser 并不太满意，如果能用 Grammar Tree 类似的状态机逻辑实现应该能够更舒服。目前的 Parser 在拓展性上不太强。

3.2.3 analyzer

Analyzer 是本次代码中最为重要的一个模块，负责解析电路生成对应 stamp，实现线性器件的 dc、ac、tran 仿真，以及非线性器件的 dc、tran 仿真，当然目前的非线性器件仅支持二极管，同时也负责结果的 plot。

Analyzer 的调用很简单，只需要把上面得到的 Parser 给到构造函数，构造函数就能读取 Parser 中的 netlist 信息以及对应的分析参数，然后根据分析参数进入不同的函数，对 netlist 进行 stamp、求解，若有要求，plot。

对于 **dc 仿真**，首先将电路 stamp，得到分析 MNA 矩阵以及 RHS，然后根据要求，改变相应电流/电压源对应节点的 RHS 值，然后使用 armadillo 自带的 arma::solve 进行求解。

对于 **ac 仿真**，首先通过仿真参数（DEC, OCT, LIN）计算出需要扫描的频率点，然后将频率作为参数传到 stamp 函数，得到电路在此频率下的 MNA 矩阵以及 RHS，然后同样使用 arma::solve 求解。

dc 和 ac 共用同一个 stamp 函数，因为 dc 其实就是在 frequency=0 时的 ac 仿真，所以只需要将频率作为参数传入即可。

对于 **tran 仿真**，因为 stamp 有所不同，所以不能共用 acdc 的 stamp 函数。同时，因为 tran 的 RHS 实际需要通过前一个时刻的结果进行计算得到，我在此处使用添加了一个 RHS_gen 矩阵，我命名为 RHS 生成矩阵，事实上就是通过该矩阵记录 RHS 的生成逻辑，此后只需要将此 RHS_gen 矩阵乘上个时间点的结果 result_t-1 就能得到 t 时刻的 RHS。

目前 stamp 的逻辑使用 BackEuler。

在 stamp 之后，便开始在时序上迭代求解，每次求解也是直接使用 arma::solve。我在这里使用了一个矩阵来保存所有的结果，以时间 t 作为 column 的 index，当然这里可以使用一个 std::vector<arma::vec>替代。

下面介绍关于二极管，即**非线性电路的求解**。求解使用 Newton-Raphson 迭代。首先遇到的第一个问题是如何正确 stamp。对于二极管来说，左侧分析矩阵包含形如 a*e^{bx}的项，而对于 RHS 来说更为复杂，包含 a*e^{bx}*x + c*e^{bx} + d 的项。最初我想采用一些矩阵来记录这些参数，但是发现需要的矩阵参数量过大。然后根据上课的内容，决定使用稀疏矩阵的一些思想，通过位置来索引参数。最终使用了一个如下的 struct 来保存所需 stamp 参数：

```
1  struct ExpTerm {
2      // The position of the ExpTerm
3      int row_index;
4      int col_index;
5      // x = V(node_1) - V(node_2)
6      int node_1_index;
7      int node_2_index;
8      ExpCoeff zero_order;   // a*e^{bx}+c → (a, b, c)
9      ExpCoeff first_order;  // (a*e^{bx}+c)*x → (a, b, c)
10 }
```

row_index, col_index 是 stamp 的位置 index。

node_1_index, node_2_index 是表示二极管所接 netlist 中节点的位置，用于后续计算 Vd[n]，即注释中的 x。

ExpCoeff 是一个包含 a, b, c 三个参数的 struct。分别意义可以参考代码的注释。事实上，这里可以用一个 std::vector<ExpCoeff>来保存所有形如 (a*e^{bx}+c) * x^n 的参数。当 a=0 时，其实也包含了所有 x^n 的线性项。所以这种表示方法在对 MOS 管的 stamp 时也能非常有效。

关于求解，主要分为 3 步。第一步是通过上述响应 stamp 参数求得相关矩阵和 rhs，第二步是通过 arma::solve 对矩阵求解，第三步比较两次求解的结果，若符合收敛条件，则 break，反之继续循环迭代。

这里遇到了一个问题，armadillo 的 solve 在应对一个行列式很小的矩阵（接近线性）时，得到的结果并不会是精确解，这将会导致有些电路在求解时无法收敛（比如 hw6 的 tb3），在这时需要在 arma::solve 中添加 arma::solve_opts::allow_ugly 参数以强制求得精确解。

对于 **plot**，调用 qcustomplot 库，并实现了一个函数，参数为 std::vector<QVector>的一些 QVector，以实现在同一图中绘制多根曲线。同时加入了相关参数设置，使得窗口中图像可以互动，即使用鼠标或键盘实现拖动以及缩放。X 和 y 轴均能独立设置是否使用 log 坐标。

4 总结

在本次的 EDA 课中，学习了使用 Qt5 创建简单 GUI，以及如何 parser 如何实现，当然本次我只使用了 QString 和 QRegularExpression 来实现我的 Parser，因为我的毕设也是 EDA 相关，准备在后续使用一些其他较强的语法分析器来实现 Parser。然后我认为本次课最重要

的内容为器件的解析，即 stamp 的过程。将电路的 kcl 方程组化为矩阵求解，更加符合计算机的思考方式，同时如何对 tran 以及非线性器件的迭代解析也是我在此之前未接触的知识。

同时我也尽全力去满足每一次的作业 testbench，除了某次作业理解错误导致某个 testbench 结果错误外，其他都做到了正确。这也让 8 周的课程有了一个非常可用的 EDA 程序产出。

然而依然有一些内容由于时间问题无法在短期内实现，比如对 MOS 管的仿真，可变步长 LTE，非线性电路收敛性等。这些内容我也希望在后续继续完善。

总而言之，对于一个 8 周的课程，我学到了很多非常有用的 EDA 相关知识，提高了我 c++的编程能力，并实现了一个自认为结果不错的课程项目。

参考文献

【1】 施国勇，“设计自动化引论课件，Lecture X”，上海交通大学微纳电子学系，**2020**。

【2】 https://doc.qt.io/qt-5/classes.html, QT5 - All Classes

【3】 Armadillo: C++ library for linear algebra & scientific computing - API Documentation (sourcefore.net)