设计自动化引论课程项目

简化 SPICE 设计

姓名：胡翰彬

班级：F0921101

学号：5092119008

同组成员：朱晶阳

摘要：

SPICE 的全名为 Simulation Program with Integrated Circuit Emphasis，是一种用于电路描述与仿真的语言与仿真器软件，用于检测电路的连接和功能的完整性，以及用于预测电路的行为。主要用于模拟电路和混合信号电路的仿真。本项目完成了一个简单的 SPICE 仿真器，提供了友好的 GUI 界面，支持部分元件的输入并允许进行 OP、DC、AC、TRAN 仿真，并能将结果绘制成曲线输出。

本仿真器采用了 Continuation、limiting 技术，并对 MOS 管模型进行了优化，以得到更好的收敛性。对数据结构组织做了进一步规划，以减少仿真时间。

对部分简单电路测试，得到预期结果。但仍存在一定收敛问题，需进一步研究。

关键词：

SPICE、GUI、收敛、MOS 管模型

I. 项目介绍

SPICE 的全名为 Simulation Program with Integrated Circuit Emphasis，是一种用于电路描述与仿真的语言与仿真器软件，用于检测电路的连接和功能的完整性，以及用于预测电路的行为。主要用于模拟电路和混合信号电路的仿真。

SPICE 主要由 GUI 界面、Parser 词法语法分析、MNA 矩阵构造、矩阵求解、器件模型建立、收敛控制、结果输出等模块组成。本项目实现过程中涉及了其中几乎所有的模块，采用各种解决收敛问题的方法，最终通过在 CYGWIN 环境下使用 C++语言编程，实现了 SPICE 的一些基本仿真功能，并提供了友好的 GUI 界面。

II. PARSER

Parser 主要分为词法分析和语法分析。词法分析主要将进行编译的文件中的字符串匹配一定的模式，并返回相应的 token。语法分析主要将由词法分析返回的 token 对其顺序进行递归定义，定义文件语法规则，并做相应的操作。

本项目中词法分析所使用的软件为 Flex，语法分析器则为 Bison。

本项目中实现的器件分析有电阻、电容、电感、Diode、MOSFET、独立电压源、独立电流源和受控源（VCVS、VCCS、CCVS、CCCS），实现的仿真分析有 OP、DC、AC、TRAN 仿真分析，其语法结构类似于 SPICE 的结构，但仍有一定区别，区别如下：

1. 电容和电感可设置初始电压和电流，不输入则默认为 0，但是仅对 TRAN 仿真产生影响。
2. Diode 的模型为任意字符，但与仿真结果无关。
3. MOS 的模型仅可使用 NMOS 和 PMOS 两种模式，否则无法识别。MOS 管长宽设置的先后顺序可以任意调换。
4. 独立电流源仅支持直流电流的方式。独立电压源支持直流方式、交流方式和 sin 发生源，同时在 TRAN 仿真时将直流方式中的电压默认为 step 信号。
5. CCVS 和 CCCS 可以正常进行分析，但是具体仿真并未实现。

6. .op 仅支持时间为 0 的直流工作点分析，即仅可使用.op。
7. .dc 仅可选择电压源作为扫描对象。
8. .ac 仅支持 DEC 的十倍频扫描方式。
9. .tran 仅支持从 0 时刻开始的瞬态仿真。

由于在文法分析过程中经常发生二义性问题，如：读入一个数值，应将其作为节点处理还是将其作为元件参数值处理等，本项目实现过程中主要采取以下 3 种办法消除这种二义性。

方法 1：Flex 中定义更多模式

```
RESISTOR        [Rr]{ALPHANUM}+
…
{RESISTOR}      {yylval->s = (char*)malloc((strlen(yytext) + 1) * sizeof(char));
                 strcpy(yylval->s, yytext);
                 yylval->s[strlen(yytext)] = '\0';
                 return token::RESISTOR;
                }
```

如上述程序在 Flex 中定义一个 Resistor 的模式，并返回一个称之为 RESISTOR 的 token，以利于在 Bison 中对于以‘R’或‘r’开头字串的文法分析，以区别电阻与一般节点名。

方法 2：Flex 中某种模式多种返回

```
DIODE           [Dd]{ALPHANUM}+
…
{DIODE}         {yylval->s = (char*)malloc((strlen(yytext) + 1) * sizeof(char));
                 strcpy(yylval->s, yytext);
                 yylval->s[strlen(yytext)] = '\0';
                 if(!stricmp(yylval->s, "DEC")) return token::DEC;
                 return token::DIODE;
                }
```

如上述程序在 Flex 中 DIODE 这个模式可能返回两种不同的 token，以区分 Diode 名和 AC 分析中 DEC 十倍频扫描方式，利于 Bison 分析。

方法 3：Bison 中递归定义

上述两种方式对 Bison 来说仅增加了新的返回的 token，却并没有从根本上解决二义性问题，分析二义性仍主要由 Bison 完成。由于 Bison 可以递归定义，故可使上述返回的 token 在不同的规则下选择合适的意义。

```
node: variable    {……}
        | INTEGER    {……};
value: VALUE    {$$ = $1;}
         | FLOAT    {$$ = $1;}
         | INTEGER     {$$ = $1;};
……
resistor: RESISTOR node node value    {……};
```

上述程序在 Bison 中对于 node 这个意义，定义为 variable（一些字符串）或者整数。value 意义中同样包含 INTEGER 这个 token。在定义 resistor 时，调用 node 和 value 这两个定义，这样就可以将 INTEGER 存在的节点和参数值的二义区分开来，并进行正确的分析。

III. 数据结构组织

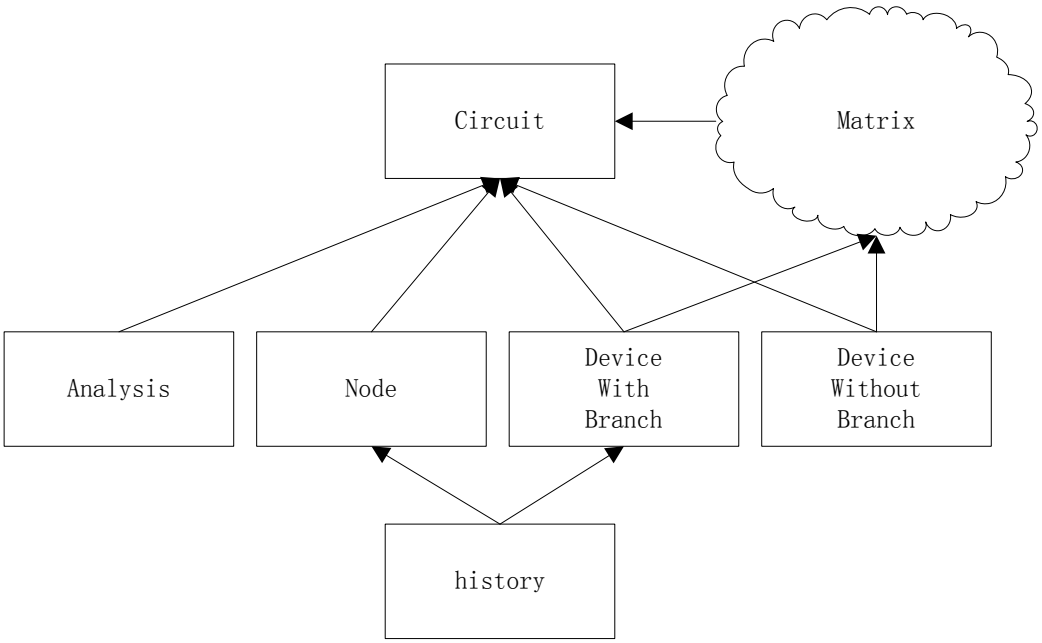

图 1 整体数据结构架构

整体仿真由 Circuit 类进行全局管理（如图 1 所示），其主要功能有如下：

1. 仿真计算过程管理
2. 对不同的元件和节点通过链表进行管理。
3. 矩阵的 Stamp，求解
4. 仿真结果存储

History_data 类：

首先对一个最底层数据结构之一 history 进行说明由于在仿真过程中需要存储两个维度上的迭代值（如图 2 所示），NR 迭代过程仅需前一次的结果，故分配两个 double 或 complex 类型存储结果进行收敛的判断。在时间维度上，在结果输出时需要将所有的结果都输出，故所有值都需记录，而且在 TRAN 仿真时根据选择算法不同，可能会使用数量不同的前 n 次仿真的结果，所以采取链表的方式对时间维度上的值进行存储。

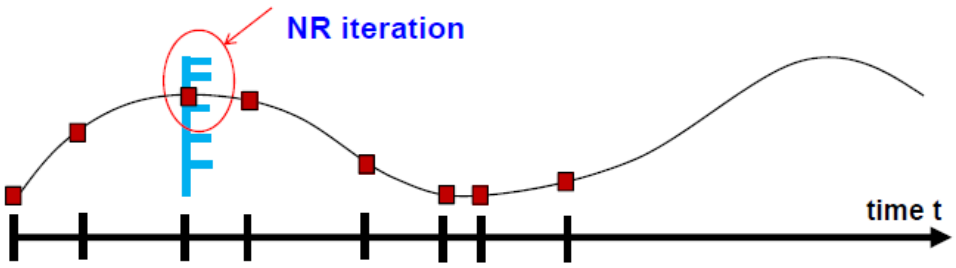

图 2 两位迭代存放

Node 类：

考虑到为方便用户查找等用途，在节点数量规模较大时，搜索效率可能较低。故采用 Hash 表对所有节点进行存储，选择的策略为将节点名中各个字符的 ASCII 码值相加后模 127 的方法，由于 127 为质数，根据在有限域 GF（p），当 p 为质数时，加法、乘法分布比较均匀。如 p=7 时，可得到下加法表和乘法表（表 1）

从表中看出数值分布均匀，hash 性质较好。乘法表同样有较好的均匀性，同时有更好的随机性，但是由于对数据安全性没有要求，故采用加法表。

表 1 GF(7)上的运算表

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

但是由于在仿真过程有大量的遍历所有节点操作，而 hash 表并不利于遍历的效率，故在 hash 表之外仍采用一链表对其中节点做指向（如图 3 所示），并用 index 标号，顺序与 parser 时的先后顺序一致，且地节点总为 index= 0 的节点，判别方式为节点名为 0，故若网表中无此节点，仿真可能出现异常。

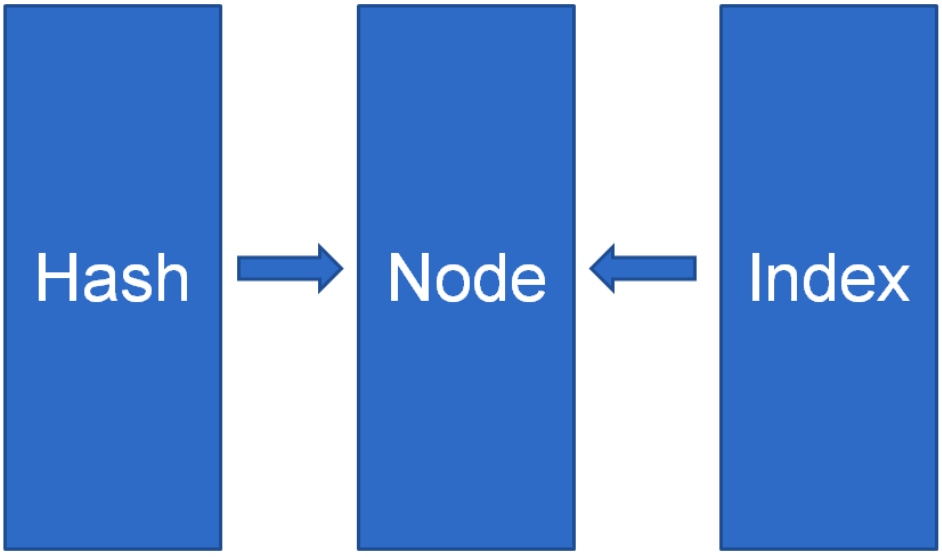

图 3 Node 类基本结构

Device 结构：

考虑到器件参数可能与周围节点电压有关（如非线性元件），故所有元件类中加入各个节点指针，以快速获取电压值，这样也有利于今后对尚未有电流结果的元件（即无需在 Stamp 时外加电流维度的元件类型）进行计算。

当矩阵变大时，或者采用一些十字链表的方式构造矩阵时，Stamp 过程中会耗费大量时间在搜索矩阵元素上，故通过 Setup 过程预先记录 Stamp 过程中矩阵

元素的指针，以减少查找时间。

由于部分器件在 MNA Stamp 时需要通过添加电流的维度以得到正确的表达式，故将这些元件区分开来，为其加入支路 Index，以标示其在矩阵构建时的位置。这些元件有电容、电感、独立电压源、VCVS。

IV. 仿真流程及采用技术

本项目共完成 OP、DC、TRAN、AC 四种仿真模式。DC 和 TRAN 仿真如图 4 所示，由于 OP 仿真仅对一点进行仿真，故仅进行内层循环，而 AC 仿真并未实现对于非线性元件的部分，故仅外层循环。

TRAN 仿真再一开始时会进行一次 OP 仿真，由于 OP 仿真即为时间时刻为 0 的直流工作点仿真，故可为 TRAN 仿真提供较好的初始工作点。

DC 和 TRAN 仿真过程由于有两层循环，故加入 Continuation 算法，即在发现内层函数收敛困难时，停止内层循环，调整外层步长，并重新计算，具体实现由朱晶阳同学完成。

AC 仿真由于一开始规划出现失误，对 complex 类估计不足，导致后期所有函数有两套模式，一套适用于 complex，一套适用于 double，程序代码略显臃肿，可以有进一步改进。

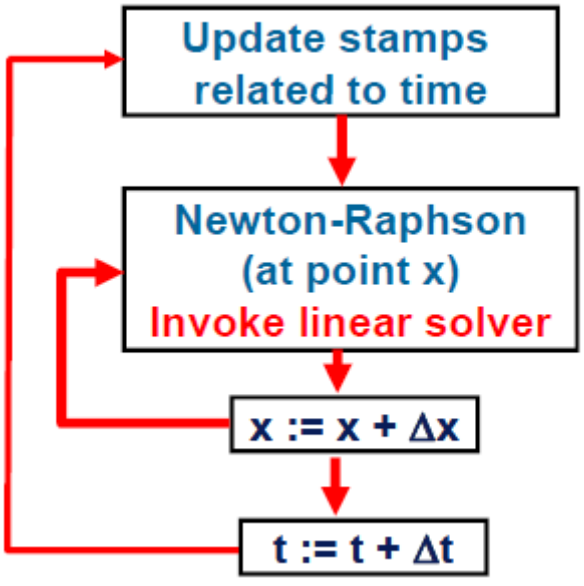

图 4 仿真流程图

V. 数据输出

仿真结束后，需要将仿真结果展示给用户。针对不同的仿真类型，数据的输出形式是不同。OP 仿真由于进行单个点的分析，数据量较少，故直接打印到 CYGWIN 环境中，这里不再赘述。下面对 DC、AC、TRAN 仿真结果输出方式进行描述。

由于在 GUI 环境中提供了独立的 Spice 仿真功能和 Plot 绘制曲线功能。由于 Spice 功能运行结束时，Circuit 类已经析构，用数据结构向 Plot 进行数据传递的方式有一定困难。所以最终采用文件读写的方式对仿真结果进行操作。

A．Spice 过程：

1. 在发现 DC、AC、TRAN 仿真后，生成相应的.lis 文件，并在文件头写入节点数量，MNA 支路元件数量，及相应的节点名和元件名。
2. 在执行完一个仿真后，输出该仿真类型及数据点个数，然后第一行写入 X 坐标（DC 为电压值，AC 为频率值，TRAN 为时间），接下来各行按节点名、元件名顺序写入各个电压值和电流值。由于 AC 仿真不仅需要输出幅度值，还需输出相位值，故 AC 仿真需将本过程执行两次。
3. 写入文件结束后，本次仿真结束，清空各 history 类，完成写文件操作。

B．Plot 过程：

1. Plot 执行后按上述写文件得到的文件格式将对应的.lis 文件数据读入，并将所有数据记录在新生成的数据结构中，并生成选择曲线界面。在分析类型添加各次不同仿真模式，AC 则分为幅度和相位的两种模式，变量中则添加节点、MNA 支路元件，以开头的‘V’和‘I’区分，如图 5 所示。

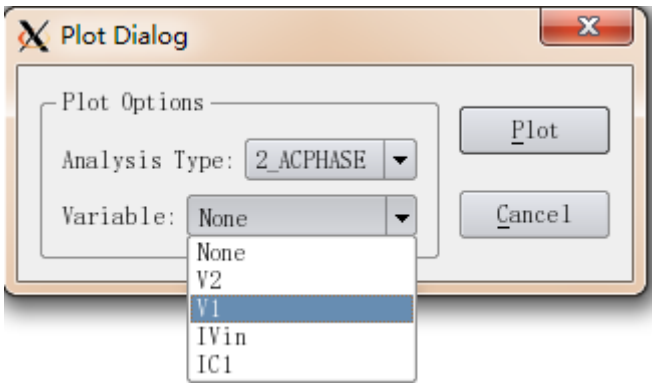

图 5 选择曲线窗口

2. 根据用户选择的仿真类型和选择的曲线，绘制曲线如图 6 所示。

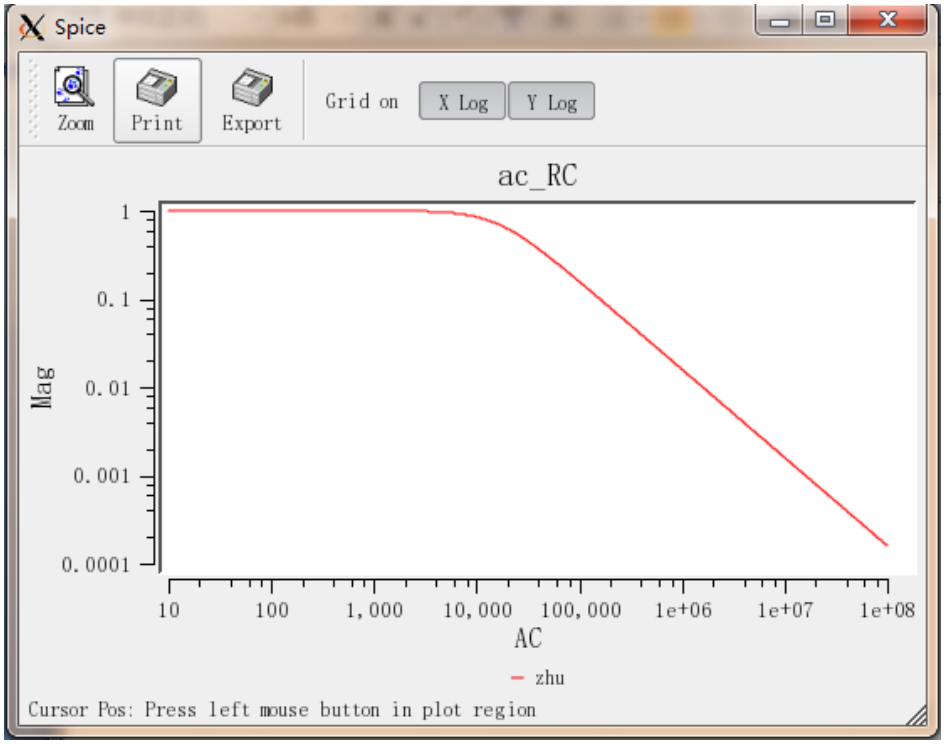

图 6 曲线显示界面

VI. MOS 管收敛问题

由于仿真过程中经常出现反相器仿真不收敛问题，故对 MOSFET 模型进行多个版本的调整。

版本一：平方率并且没有沟道长度调制

发生问题：大量点无法收敛，发生震荡、溢出等错误

原因分析：由于没有沟道长度调制情况下，两个 MOS 管饱和区曲线平行，故在 V_{out} 等于 $V_{dd}/2$ 时，两条曲线交点有很多种可能性，而偏离 $V_{dd}/2$ 时，交点在两侧，收敛难度较大。

版本二：平方率并且仅在饱和区有沟道长度调制

发生问题：V_{in}=2.9V(V_{dd}=3.3V)不收敛

原因分析：由于在 MOS 管在 $V_{gs}=V_{TH}$ 时有一个点不连续。故 2.9V 难以收敛。

版本三：平方率并且仅在饱和区有沟道长度调制且添加 Continuation 算法

发生问题：曲线中间部分震荡（如图 7 所示）

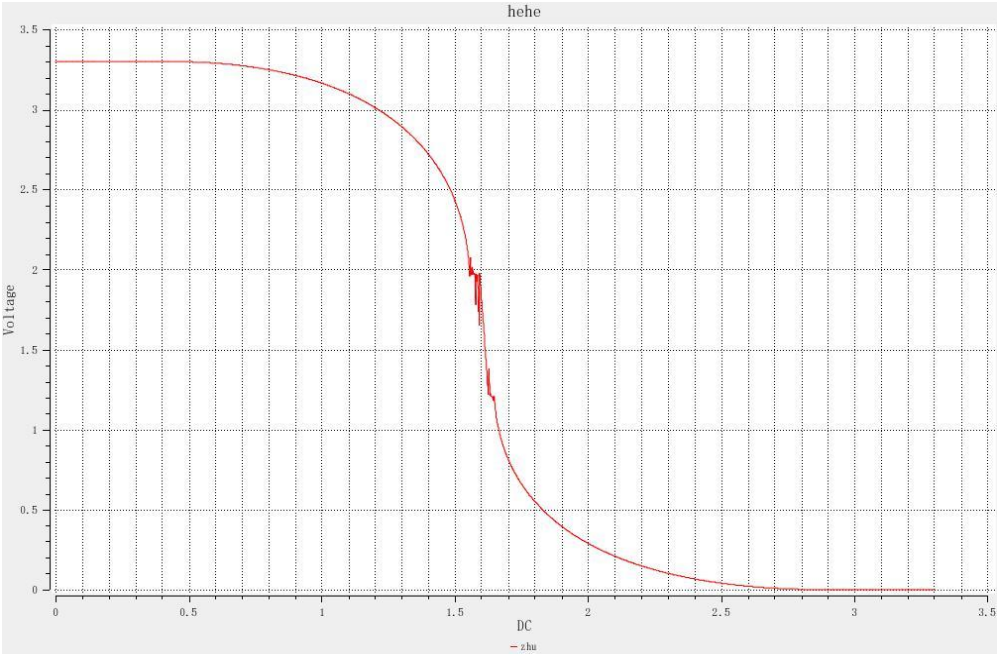

图 7 MOS 管版本 3 结果

原因分析：由于 MOS 管曲线在中间部分斜率均较小，可能因此存在收敛问题。

版本四：平方率并且考虑沟道长度调制

发生问题：当对 V_{in} 赋初值 0.7V，得到图 8。当对 V_{in} 赋初值 0.7V 且对 V_{out} 赋初值 3.3V，得到图 9。

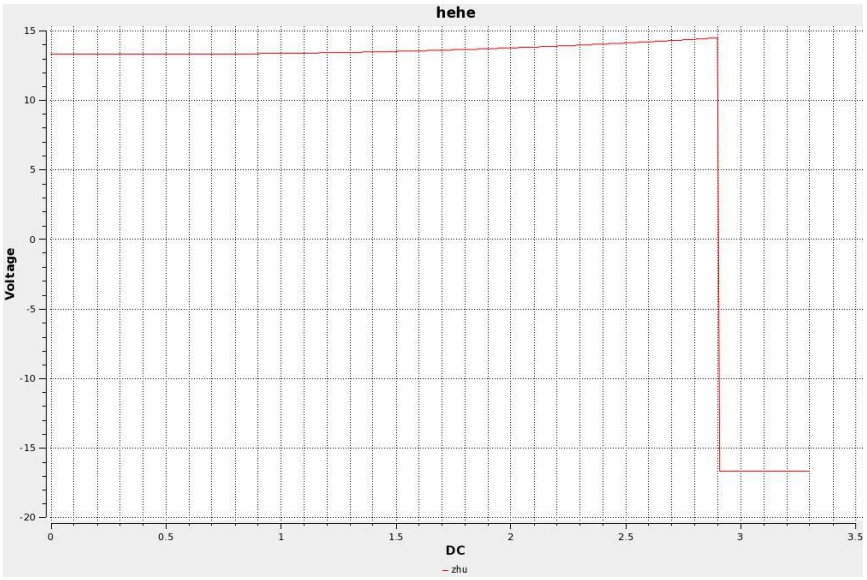

图 8 MOS 管版本 4 结果 a

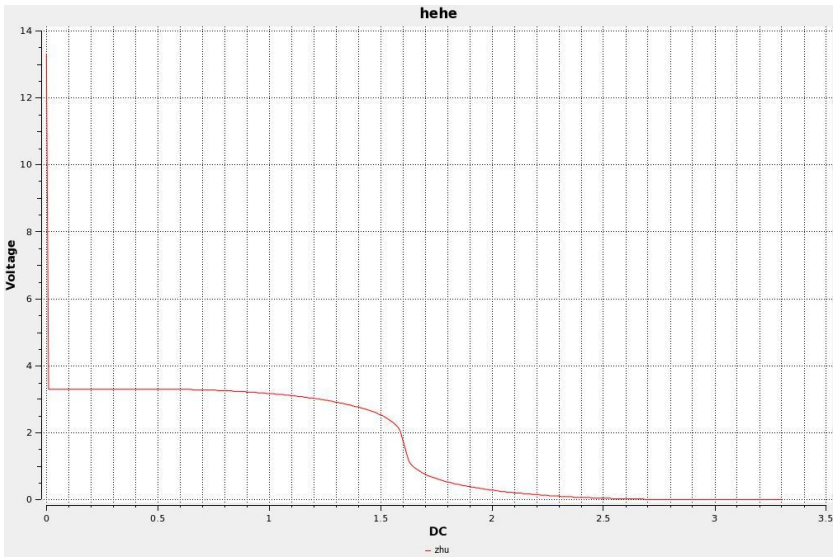

图 9 MOS 管版本 4 结果 b

原因分析：由于考虑沟道长度调制，故 MOS 管公式可能出现两个解，如下式为 MOS 管线性区公式：

$$I_{ds} = k' \frac{W}{L}\left(V_{ov}V_{ds} - {V_{ds}}^2\right)(1 + \lambda V_{ds})$$

当 V_{in}=0 时，即 I_{ds}=0，故可有

$$V_{ds} = -\frac{1}{\lambda}$$

故有几率收敛至另外一个解中。Matlab 作图（图 10）可以看到 MOS 管曲线存在两个零点：

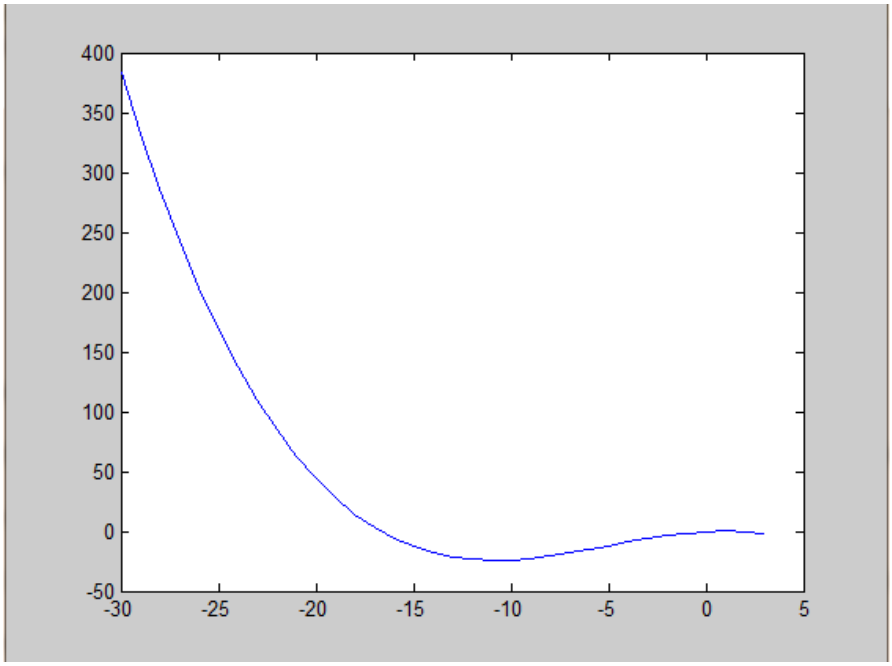

图 10 Matlab MOS 模型作图

故在调整初值过程中，可使初始情况接近解，以期获得正确结果，故第二种赋初值的策略得到了更好的结果，但仍然不解决根本问题。

版本五：平方率并且考虑沟道长度调制，并对 Vds<0 为单纯平方率，并考虑 Continuation 算法

发生问题：反相器曲线（图 11）得到了很好的结果，但是 buffer（图 12）仍存在一个错误的点。

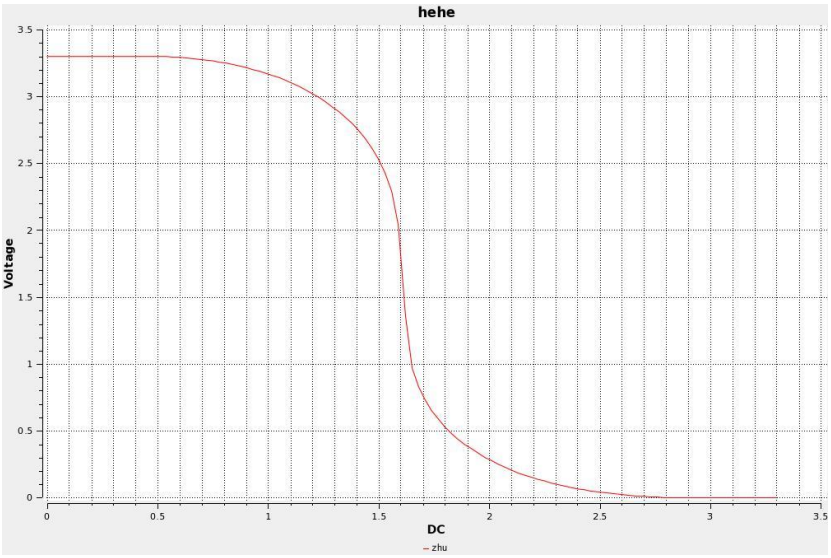

图 11 MOS 管版本 5 反相器曲线

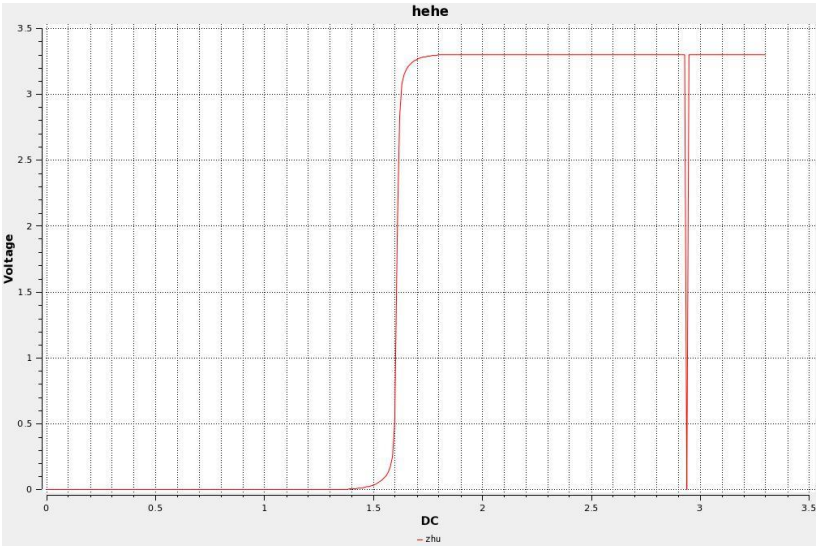

图 12 MOS 管版本 5 buffer 曲线

原因分析：具体原因不明，可能为浮点数运算精度导致

版本六：版本五基础上修改 Stamp 方式

发生问题：未发生问题，得到了正确的 buffer 曲线和反相器曲线，同时也可以对与门和或门一些简单的逻辑门进行仿真。

原因分析：原本的 Stamp 方式为迭代开始前，对线性元件进行 Stamp，然后每一轮迭代中，将前一次的非线性元件迭代值减去，并加上此次的迭代值。这样的操作容易造成矩阵中各个元素由于每次迭代都会有新的浮点运算误差累加，导致在运算至后期，误差较大，容易不收敛，故改变了 Stamp 方式为每次迭代时将矩阵清零，并重新 Stamp。

经过各个版本的修正，总结主要的一些发生收敛问题时的修改方法有：

1. 修改非线性元件模型（连续、导数连续）
2. 选择合理初始值
3. 算法辅助（Continuation、Limiting）
4. 减少单个数据的浮点运算次数，增加精度

VII. 整流电路问题

目前仿真器进行半波整流电路仿真，如果仅为一个 Diode 和一个电阻进行串联形成的半波整流电路，可以得到正确的结果，结果如图 13 所示。

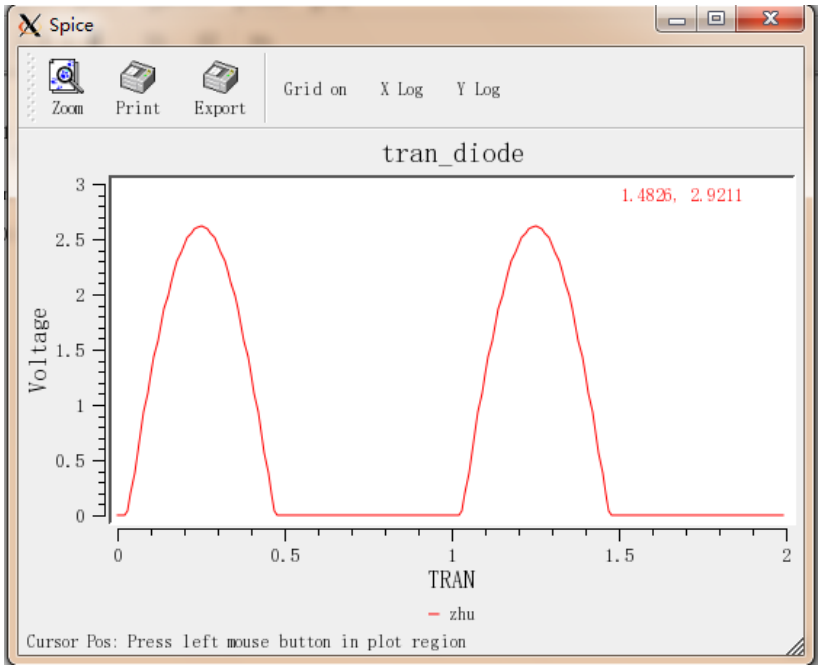

图 13 输出级无电容半波整流电路结果

但是当输出级负载并联一个电容时，仿真可能出现不收敛，电路结构如图 14 所示：

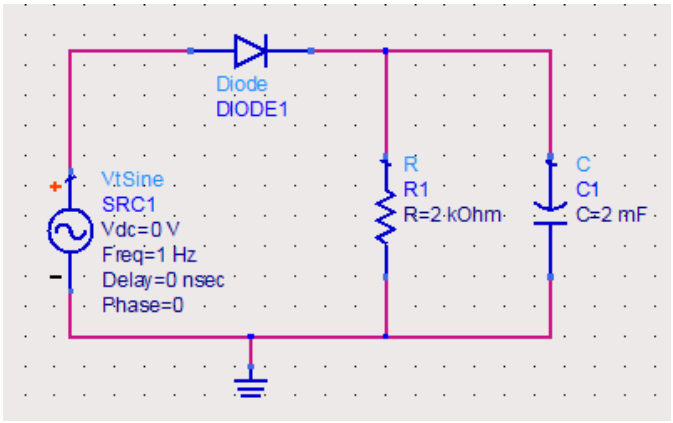

图 14 半波整流电路

当参数如上图所示时，一开始会出现矩阵中元素计算得出 inf 或 nan 的结果，故怀疑存在 Diode 的 overflow 情况，故加入 limiting 技术，但是对于 limiting 技术加入的方式存在一定问题。

由于考虑当 Diode 数量较多时（如图 15 所示），在矩阵求解结束后，得到四个节点的电压，但是对求解顺序存在一定问题。如从 1 号节点开始，根据 Diode3 和 Diode5 的状态计算出 2 号和 3 号节点的电压，但在计算 4 号节点可由 3 号节点和 Diode2 得出，或者由 Diode4 和 2 号节点得出，如何选择成为问题。同时，一味通过 limiting 降低部分节点电压，可能造成别处压降变大，造成新的 overflow 问题，故现在 limiting 的方式采取如下所述。

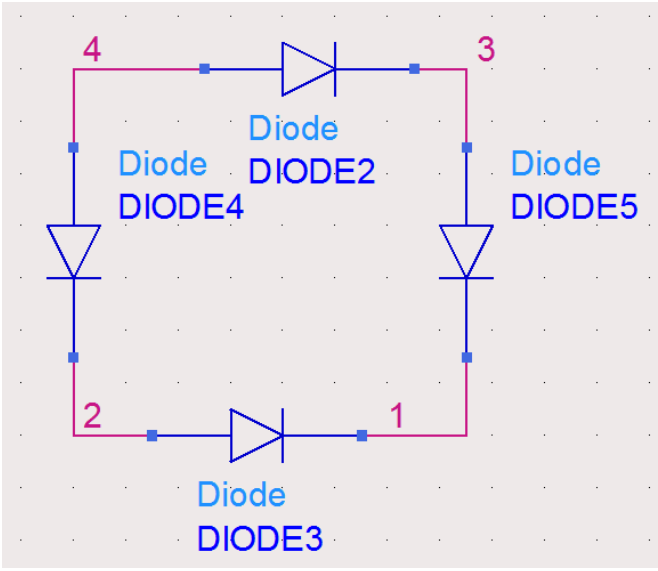

图 15 limiting 技术示例

目前采用的方式，直接将矩阵求解时所得的结果存入 history 类，然后当下一次 Stamp 时，Diode 根据上一次的求解结果 V(K)计算出 $V_{lim}(K)$，从而计算此次的 G_0 和 I_0 进行 Stamp，即一次矩阵 Stamp 的值可能是由不同的电路状态得出的结果。但是在仿真一些可正常仿真的电路时并未发生计算结果不正确的结果的情况，故暂时认可这种方式。MOS 的初值赋予方式也是通过这种不同电路状态方式得到的。

加入 limiting 后，发现仍然不可以收敛，且发现当调整电容大小为 2μ F 时，可以得到仿真结果，但是结果如图 13 所示，并没有看出电容滤去高频分量的效应，故怀疑可能是矩阵求解存在一定错误，当电容值较小时，在多次计算后会忽略电容的效应，故更换新的 Solver 可能可以解决该问题。Sparse1.3 由于其 Stamp 方式为同时 4 个值赋值，造成程序有较大变动，时间有限故未能使用。在互联网找到 Sparselib++，初步尝试下发现其矩阵求解能力有一定问题，也未能及时合进程序中，故该问题仍未解决。全波整流电路同样不可仿真。

VIII. 结论

最终，本项目完成一个简单的 SPICE 仿真器，运行环境为 CYGWIN，提供较友好的 GUI 界面。可支持电阻、电容、电感、Diode、MOSFET、独立电压源、独立电流源和受控源（VCVS、VCCS、CCVS、CCCS）的 Parser，实现的仿真分析有 OP、DC、AC、TRAN 仿真分析。利用 Qwt 制作了曲线绘制 GUI 界面，以输出仿真结果。对一些简单的电路进行了仿真，仿真结果与预期一致。但是对于部分电路仍存在不收敛等问题。需要进一步的解决和研究。

IX. 致谢

感谢朱晶阳同学与我共同完成这个项目，并在很多问题提出了很有建设性的意见。

感谢施国勇老师半个学期的辛勤授课，并能及时地在我们项目遇到问题时，通过各种方式给予我们意见和建议。

感谢宋阳助教随时随地回答我们一些问题，并在收敛问题上一些技术细节的给我们一些指导。

感谢一起上课的同学，你们的经验与实现过程给了我们很大的帮助和鼓励。

X. 参考文献

[1] Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic, Digital Integrated Circuits: A Design Perspective, Second Edition
[2] John Levine, flex & bison
[3] William Stallings, Cryptography and Network Security Principles and Practice, Fifth Edition
[4] Guoyong Shi, lecture pdf, School of Microelectronics, Shanghai Jiao Tong University
[5] http://en.wikipedia.org/wiki/SPICE