



上海交通大学

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 TITLE: SOFTWARE IMPLEMENTATION FOR
INTEGRATED CIRCUIT SIMULATION

学生姓名：_____ 孙永帅 _____

学生学号：_____ 515030910096 _____

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：_____ 2018 年 (春季) _____

目录

摘要.....	1
ABSTRACT.....	1
1.绪论.....	2
1.1 研究背景.....	2
1.2 设计理论基础.....	2
1.3 Python 编程语言	2
2.电路网表解析.....	3
2.1 电路网表格式.....	3
2.2 Python 正则表达式与网表解析	3
2.3 参数提取.....	4
2.3.1 元器件参数.....	4
2.3.2 命令参数提取.....	5
2.4 数据存储.....	5
2.4.1 元器件类的定义存储.....	5
2.4.2 节点、参数存储.....	6
2.4.3 命令指令的存储.....	6
3.电路 Stamp	6
3.1 线性元器件 Stamps.....	6
3.1.1 电阻 R Stamps	6
3.1.2 电流源的 Stamps.....	7
3.1.3 电压源 Stamps.....	7
3.1.4 受控源 Stamps.....	7
3.1.5 C/L 的 AC 的 Stamps	8
3.1.6 C/L 的 Tran 仿真 Stamps	8
3.2 非线性元件 Stamps.....	10
3.2.1 Diode 建模与 Stamps	11
3.2.2 MOSFET 建模与 Stamps	11
4.结果举例.....	13
5.收敛性问题.....	17
6.总结.....	18
6.1 设计总结.....	18
6.2 感想与展望.....	18

集成电路仿真器软件设计

姓名: 孙永帅 学号: 515030910096

(电子邮件: sunyongshuai@sjtu.edu.cn 电话: 15821886825)

摘要

EDA (Electronic design automation) 工具在当今的集成电路设计的过程中, 发挥着巨大的作用。本文从基本原理出发, 介绍了 EDA 工具的实现方法, 并借助 Python 语言简要设计并实现了一个集成电路仿真器软件。本文从电路网标的解析出发, 使用 MNA 方法生成电路矩阵的数据结构, 通过求解矩阵来完成电路的状态解析。本文的设计涵盖了基本线性原件、受控源、非线性二极管、基本 MOS 管等的 DC 仿真、AC 仿真、Tran 仿真等。结果表明, 本文设计在线性原件、Diode 模型原件仿真上表现良好, 适用于 MOSFET 仿真设计。

关键词: 电路仿真器; MNA; SPICE 模型; Python

SOFTWARE IMPLEMENTATION FOR INTEGRATED CIRCUIT SIMULATION

Name: Yongshuai Sun Student ID: 515030910096

(Email: sunyongshuai@sjtu.edu.cn Tel: 15821886825)

ABSTRACT

EDA (Electronic Design Automation) tools play an enormous role in the design of today's integrated circuits. This article starts from the basic principle, introduces the implementation method of EDA tool, and uses Python language to design and implement an integrated circuit simulator software. This article starts from the analysis of the circuit network standard, uses the MNA method to generate the data structure of the circuit matrix, and solves the state analysis of the circuit by solving the matrix. The design of this article covers basic linear elements, controlled sources, nonlinear diodes, basic MOSFET and so on. DC simulation, AC simulation and Tran simulation are solved in this design. The results show that the design of this paper works well on the simulation of linear elements and diode models, which is also suitable for MOSFET simulation design.

KEYWORDS: Circuit Simulator; MNA; SPICE Model; Python

1. 绪论

1.1 研究背景

随着集成电路的发展, 集成电路的仿真成为了集成电路设计中的重要一环。而其中的仿真软件给我们了很大的便利, 为我们的设计提供了有效的验证方式, 给我们的电路设计的正确性和性能要求提供了参考。

电路仿真的设计以 Berkeley 大学开发的 spice 模拟算法为代表, 很多商业公司也基于 SPICE 模型建立了自己的仿真软件, 比如 Synopsys 的 Hspice、Cadence 的 Spectre 等, 这些商业软件, 凭借软件授权, 每年获取巨额的利润; 同样, 这给研究和学习性质的使用带来了巨大的不便! 因此, 掌握 EDA 集成电路仿真工具的设计原理和相关知识, 有利于推动相关领域的进步和发展。

作为微电子系的学生, 电路设计之外, EDA 工具也是相当重要的一环。面对国内没有自己的商业仿真工具面临的技术封锁, 面对使用 Synopsys、Cadence 等商业仿真工具所带来的巨额开销, 学习 EDA 工具设计相关的知识不仅有利于对电路知识的应用和掌握, 也是面对商用 EDA 工具现状的一种选择!

1.2 设计理论基础

EDA 工具的设计是多学科、多领域的交叉, 需要较强的专业电路、应用数学、计算机编程等相关的知识作为基础。

- 1) 电路知识: 集成电路仿真器的本身是解电路的工作, 基础的电路知识是解析电路的基础; 其中包括但不限于 KCL 方程、KVL 方程、原件模型等知识。
- 2) 数学知识: 解析电路的方式主要有基于图论、基于矩阵的方式来表征电路, 进而求解电路的状态, 因此数学的结构、运算在电路解析中有很重要的角色。另一方面, 在非线性电路的收敛性实现求解上, 牛顿法求解非线性交点问题的方法涉及到数学几何学理论、微积分知识等。
- 3) 计算机编程: 集成电路仿真器的设计是软件设计的一种, 因此基础的计算机的编程知识是仿真器设计实现的基础。在本文的设计中采用 Python 语言作为设计语言。针对电路仿真器的设计, Python 编程语言具有的正则表达式、数据结构、功能包等特点为设计带来了很大的便利。

1.3 Python 编程语言

在本文的设计中基于 Python 编程语言设计。Python 是一种面向对象的解释型计算机程序设计语言。虽然其相比于编译型语言(如 C/C++)等在速度、效率等上存在不足, 但其丰富强大的库为我们使用别人设计完成的基本功能提供了可能和便利, 使得其得到广泛的应用。

在本文的设计中, Python 也为我们提供了很大的便利。设计中我们是用来 Python 的 Tkinter (图形界面)、numpy (数据结构)、re (正则表达式)、matplotlib (绘图) 等库, 为我们可视化、数值运算等提供了极大的便利。

2. 电路网表解析

本文的设计是以 Synopsys 公司的 Hspice 作为参考,其电路的输入采用 Hspice 的电路网表格式, 因此电路的解析方式以 Hspice 的电路网表格式为参考。

2.1 电路网表格式

在 Hspice 中电路网表主要包含原件（电阻、电容等）、命令行（以 DOT 起始）、注释行（以 ASTERISK 起始），因此我们在电路解析中主要是对原件、命令和注释的识别和解释。电路网表格式实例如下图 2_1 所示：

1	.title BandPass	}	第一行为标题
2			
3	Vin port1 gnd 1	}	原件(R/C...)
4	R1 port1 port2 1k		
5	C1 port2 gnd 0.022u		
6	C2 port1 port3 0.1u		
7	R2 port3 gnd 1k		
8	C3 port3 out 0.1u		
9	R3 port2 out 1k		
10			
11	.ac fre 1 1G	}	命令(ac/plot...)
12			
13	.plot ac v(port1) v(out)	}	注释行
14	*Must more than two,The first is Reference		
15			
16	.END		

图 2_1 电路网表格式实例

2.2 Python 正则表达式与网表解析

在上述的电路网表中,我们想要得到电路的结构信息就必须对电路进行识别,将原件、命令、注释等分别开然后在进行相应的处理。这里我们用到了 Python 中的正则表达式。

正则表达式 (Regular Expression, RE) 是对字符串操作的一种逻辑公式,就是用事先定义好的一些特定字符、及这些特定字符的组合,组成一个“规则字符串”,这个“规则字符串”用来表达对字符串的一种过滤逻辑。通常被用来检索、替换那些符合某个模式(规则)的文本。其具有如下特点:

- 1) 灵活性、逻辑性和功能性非常强;
- 2) 可以迅速地用极简单的方式达到字符串的复杂控制。

基于正则表达式的以上特点,对于我们电路网表的初步解析很是方便和快捷。比如在判断一行是否是命令行,我们可以通过简单的正则表达式来匹配,其正则表达式定义如下图 2_2 所示:

```
regExpCommand = r'^\s*' #Command line
```

图 2_2 命令行正则表达式定义

通过上面的正则表达式的定义, 另外通过 Python 中的 re 正则表达式库中的 rematch 函数等, 我们可以快速判断出表达式的属性。

在语句的解析中, 我们可以按照其首字母将各个元器件分离出来。以供后续的处理和原件参数提取。在这个电路的解析的过程中, 其流程如下图 2_3 所示:

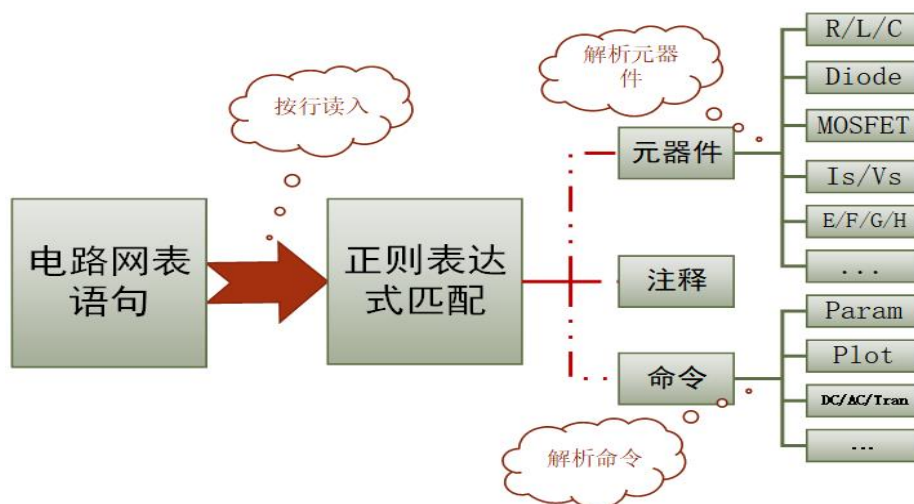


图 2_3 电路网表解析流程

2.3 参数提取

通过上面电路网表的解析, 我们确定了每一行代表什么元器件、什么命令, 但是其中的细节我们还没有提取出来, 也就是原件的连接信息和相关值得信息我们还没有得到, 因此我们需要对前面得到的结果进一步解析来获取其中的全部信息。其主要包括元器件中的参数提取和命令行参数提取 (命令行的要求等)。

2.3.1 元器件参数

在解析其中的信息时, 我们主要是提取相应的特征信息, 比如对于电阻 R 来讲, 我们定义的结构如下图 2_4 所示:

▪ **Rname <node> <node> <val>**

图 2_4 电阻 R 的定义格式

因此, 我们可以通过将 R 语句通过空格分离, 分别得到其中的 NODE1、NODE2 和 Value 的信息。

值得注意的是, 在很多的元器件中有些可省略参数, 就是这些参数可以由用户指定, 或者使用程序内部 Default 值, 这样的模型定义, 我们就需要对相应的参数、相应的位置进行判断, 以获取对应的参量。比如, Hspice 中 MOSFET 的模型定义如下图 2_5 所示:

```

MXXXXXXX ND NG NS NB MNAME < L=VAL> < W=VAL> <
AD=VAL> < AS=VAL> < PD=VAL> < PS=VAL> < NRD=VAL> <
NRS=VAL> < OFF> < IC=VDS, VGS, VBS> <TEMP=T>
  
```

图 2_5 MOSFET 的定义

如其中的 AD/AS/PD 等参数，我们可以不进行特别指定，使用 Default 值；并且一般这样的参数都是通过赋值语句来完成，比如 AD=VAL，这样我们就可以通过判断某个赋值块是对哪个参数赋值，进而得到对应参数的值。

2.3.2 命令参数提取

需要提取的命令指令主要包含如下表 2_1 所示：

表 2_1 命令语句格式表

命令	格式	说明
.param	.PARAM pName=VAL	参数定义
.ac	.AC StartFre StopFre	AC 仿真
.dc	.DC ParamName Start Stop Step	DC 仿真
.tran	.TRAN TimeStep TimeStop	Tran 仿真
.plot	.PLOT v(NODE1) i(R1)	作图

从上面的表中我们总结出相关的命令语句的特征，从而提取其中的参数。以较为复杂的 plot 为例，我们可以使用 python 对其分割，然后对分割后的字符串进行判断处理，得到其中的参数。因此，对于参数的提取来讲，主要是通过特定字符分割字符串，然后进行判断识别。

2.4 数据存储

进行过参数提取后的电路网表包含了整个电路的信息，等价于一个电路图（元器件部分），如何对这些信息进行有效的存储是我们解决的问题。在本文的设计中，本文的设计中充分利用 Python 面向对象的特点，将元器件、命令指令等进行有效存储。

2.4.1 元器件类的定义存储

本文中将每一种元器件定义一个类，其类中的元素包含所需要的参数，比如我们定义电阻的类结构如下图 2_6 所示：

```
class Resistor:
    name = ''
    port1 = ''
    port2 = ''
    value = '1k' #1000 ohm for default
    StampMatDict = {}
```

图 2_6 电阻类定义

这样我们就可使用面向对象的方式来调用元器件，进而使用其中参数。

另一方面，元器件类的存储上，在解析到每个元器件的时候，我们使用 List 的数据结构，来存储同类型的元器件。这样，我们每个元器件的类型定义一个 List 来保存电路网表中的所有元器件，在后面的 Stamps 过程中依次遍历即可。

2.4.2 节点、参数存储

节点的信息涉及到后面的电路矩阵的生成、元器件的电压的计算等, 因此有效的节点的存储有利于后面的计算。在后面的电路矩阵的生成时需要对节点进行编号, 因此在节点存储上, 我们可以使用 Python 中的字典的数据结构, 其中 value 为节点的 Name, Key 为节点的编号。同理, 对于 param 定义的变量, 同样可以使用 Python 中的字典, 以方便其读取使用。

2.4.3 命令指令的存储

在遍历电路网表时, 我们同样会遍历到电路网表中的一些命令指令, 我们需要一定的数据结构去存储它们。这里我们采用 List 的数据结构来分别存储 AC\DC\Tran 的变量, 这样在进行 AC\DC\Tran 的仿真时, 我们只需要读取相应的数组即可。对于 Plot 命令, 我们同样可以通过数字来记录其中的数据, 只不过 plot 的数组中, 存放的将是 plot 的类, 包含的信息包括 plot 的类型 (V, I), plot 的节点、元器件信息等。

3. 电路 Stamp

在本文中使用矩阵数据结构来表征电路的, 这样能够很方便地求解出电路的解。另一方面, 使用矩阵来表征电路取决于电路求解的 KCL、KVL 方程。而在这个过程中, 各个原件的 Stamps 是 SPICE 仿真的重点。

3.1 线性元器件 Stamps

3.1.1 电阻 R Stamps

对于右图 3_1 所示的电阻 R, 由 KCL 方程:
对于 N+ 节点:

$$\frac{V_{N+} - V_{N-}}{R_k} = i_{N+}$$

对于 N- 节点:

$$\frac{V_{N+} - V_{N-}}{R_k} = -i_{N-}$$

在 Stamps 的矩阵中, 其基本的思路是:

$$Y * V = RHS$$

对于电阻 R 来讲即是:

$$G * V = I$$

因此, 根据矩阵中求解的方程对应, 我们应该在矩阵的 N+/N- 节点对应的位置上添加 V_{N+}/V_{N-} 的系数, 以满足上述方程的体现。

所以对于 R 的 Stamps, 如下图 3_2 所示:



图 3_1 电阻 R 实例

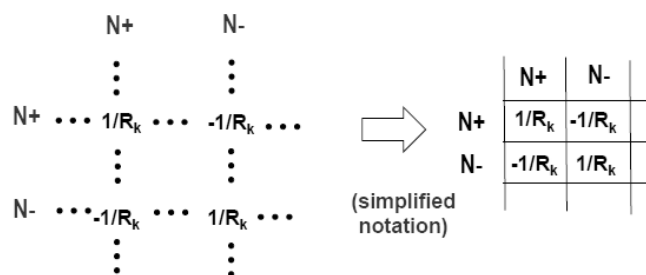


图 3_2 电阻 R 的 Stamps

在最后的矩阵的整合的过程中, 我们只需要将上述的 Stamps 添加到相应的整体矩阵中即可。

3.1.2 电流源的 Stamps

如右图 3_3(a)所示的电流源, $N+$ 节点电流为 i_k , $N-$ 节点的电流为 $-i_k$, 按照 KCL 方程,

$$\mathbf{i}_{N+} = \mathbf{i}_k, \quad \mathbf{i}_{N-} = -\mathbf{i}_k$$

我们可得, 其 Stamps 如右图 3_3(b)所示, 其结果仅体现在 RHS 矩阵。

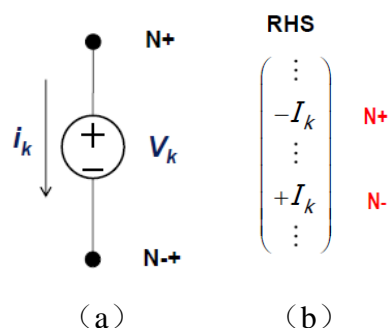


图 3_3 电流源及其 Stamps

3.1.3 电压源 Stamps

如右图 3_4 所示的电压源, 由于其两端的电压值为固定的, 而其电流值反倒取决于外部电路的状况, 因此单纯地从 KCL 出发不能够表征电压源的特征, 这里我们要引用 KVL 方程:

$$V_{N+} - V_{N-} = V_k$$

在 $N+$ 点, V_k 贡献的电流值是 I_{V_k}

在 $N-$ 点, V_k 贡献的电流值为 $-I_{V_k}$

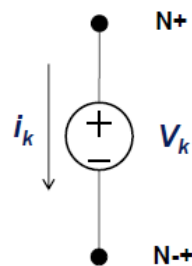


图 3_4 电压源模型

因此, 其 Stamps 如下图 3_5 所示, 相对于节点的编号多了一行一列。并且, 在求解矩阵中将可以直接得到电压源的电流值。

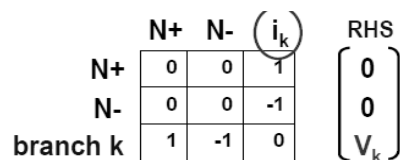


图 3_5 电压源 Stamps

3.1.4 受控源 Stamps

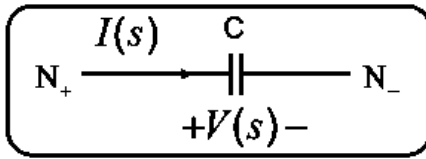
上述对于 $R/I_s/V_s$ 等的 Stamps 讨论, 讲述了 Stamps 过程中的集中不同的情况, 而对于以下的各种情况均是基于此的组合, 只要列出其 KCL 和 KVL 方程,

即可据此得到其 Stamps 规则，因此详细过程将不做讨论。

3.1.5 C/L 的 AC 的 Stamps

对于电容 C 和电感 L 来讲，其在 DC 仿真可以将电容 C 考虑为开路，电感 L 考虑成导线（可以使用电压源形势来表示，其值为 0），而至于其 AC 仿真，则需要频域进行 Stamping，对于其 Stamps 如下图 3_6 所示：

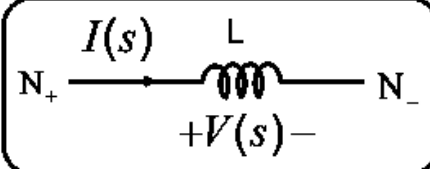
	N_+	N_-	RHS
N_+	sC	$-sC$	
N_-	$-sC$	sC	



$$I(s) = sC \cdot V(s)$$

(a) 电容的 NA 方法 Stamps

	N_+	N_-	RHS
N_+	$\frac{1}{sL}$	$-\frac{1}{sL}$	
N_-	$-\frac{1}{sL}$	$\frac{1}{sL}$	



$$V(s) = sL \cdot I(s)$$

(b) 电感的 MNA 方法 Stamps

图 3_6 C/L 的 AC Stamps

对于电容电感的 Stamps，在编程时只需要将 s 转化为 jw，在复数域来完成矩阵的运算即可。

3.1.6 C/L 的 Tran 仿真 Stamps

电容电感作为储能元件，其在时间上的状态和其上一时刻的状态有关，因此在 Tran 仿真的 Stamps 中，电容和电感的 Stamps 应和上一时刻的状态相关联。在 C 和 L 的特征方程中，其存在微分项，因此我们应用到其相关含义。

在离散化的微分和积分求解中，我们通常使用的方法包括 Forward Euler，Backward Euler 和 Trapezoidal Rule，其含义如下图 3_7 所示：

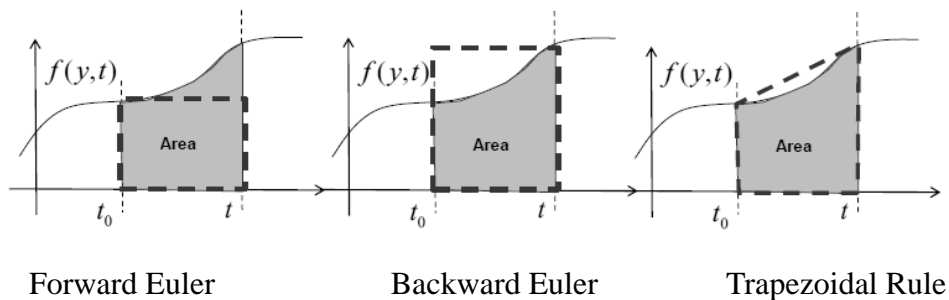


图 3_7 离散微积分求解方式图示

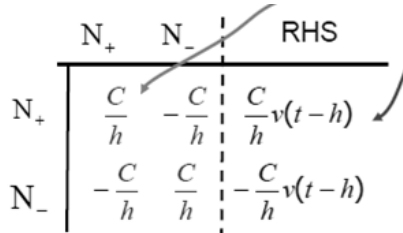
其计算表达式如下图 3_8 所示:

$$\begin{aligned} \text{Forward Euler: } & y(t_n) = y(t_{n-1}) + h \cdot f(y(t_{n-1})) \\ \text{Backward Euler: } & y(t_n) = y(t_{n-1}) + h \cdot f(y(t_n)) \\ \text{Trapezoidal Rule: } & y(t_n) = y(t_{n-1}) + \frac{h}{2} [f(y(t_n)) + f(y(t_{n-1}))] \end{aligned}$$

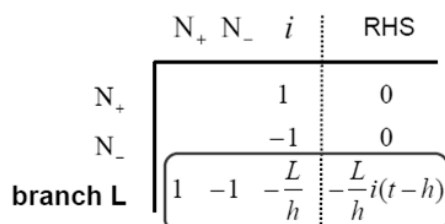
图 3_8 微积分求解方式表达式

其中 $f(y(t))$ 表示 $y(t)$ 的导函数。同时, 一般来讲 Forward Euler 的精度远不及后两者, 同时 Backward Euler 的精度稍不及 Trapezoidal Rule。本文的设计中我们采用折中的方式, 使用 Backward Euler 作为计算。

将其应用到 C/L 的 Stamping 中去结果如下图 3_9 所示: (Backward Euler)



(a) C NA Stamping



(b) L MNA Stamping

图 3_9 C/L 的 Tran Stamps (B.E)

作为对比, 我这里给出 Trapezoidal Rule 方式的 Stamping 过程。

根据上面 Trapezoidal Rule 的表达式, 对于电容来讲, 其方程为:

$$i(t) = C \frac{dv(t)}{dt}$$

因此, 根据 Trapezoidal Rule, 我们可以得到:

$$\frac{1}{2}(i(t) + i(t-h)) = \frac{C}{h}[v(t) - v(t-h)]$$

移项即为:

$$\frac{C}{h}v(t) - \frac{1}{2}i(t) = \frac{C}{h}v(t-h) + \frac{1}{2}i(t-h)$$

因此 C 的 Stamp 如下表 3_1:

表 3_1: 电容的 Stamp (TR)

	N+	N-	Branch C	RHS
N+	0	0	1	0
N-	0	0	-1	0
Branch C	$\frac{C}{h}$	$-\frac{C}{h}$	$-\frac{1}{2}$	$\frac{C}{h}v(t-h) + \frac{1}{2}i(t-h)$

同理, 对于电感来讲, 其方程为:

$$v(t) = L \frac{di(t)}{dt}$$

因此, 根据 Trapezoidal Rule, 我们可以得到:

$$\frac{1}{2}(v(t) + v(t-h)) = L \frac{i(t) - i(t-h)}{h}$$

移项即得:

$$\frac{1}{2}v(t) - \frac{L}{h}i(t) = -\frac{1}{2}v(t-h) - \frac{L}{h}i(t-h)$$

即有 L 的 Stamp 为表 3_2:

表 3_2 L 的 Stamp (TR)

	N+	N-	i _L	RHS
N+	0	0	1	0
N-	0	0	-1	0
Branch L	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{L}{h}$	$-\frac{1}{2}v(t-h) - \frac{L}{h}i(t-h)$

这里对于其中用到的前面时刻状态的值, 我们可以通过得到的 Result 矩阵进行传递, 达到上述的效果。

3.2 非线性元件 Stamps

在本文的设计中主要涉及的非线性元件包括 Diode 和 MOSFET, 这里主要讨论 Diode 和 MOSFET 的建模和 Stamps。

在非线性问题的求解中, 我们通常使用到 Newton-Raphson 方法。牛顿法 (Newton-Raphson Algorithm) 是通过不断的线性化和迭代求解来实现求解的, 对于曲线 $f(x)$, 我们有其泰勒展开为 (在 $x=a$ 点):

$$f(x) = f(a) + f'(a)(x-a) + \dots$$

因此其切线可以表示为泰勒展开的低阶部分, 即 $x=a$ 点的切线为:

$$y = f(a) + f'(a)(x-a)$$

曲线和它的切线如下图 3_10 所示。这样, 我们通过使用切线来不断靠近其焦点的方法, 我们可以得到交点, 进而解的方程组的解。迭代过程如图 3_11。

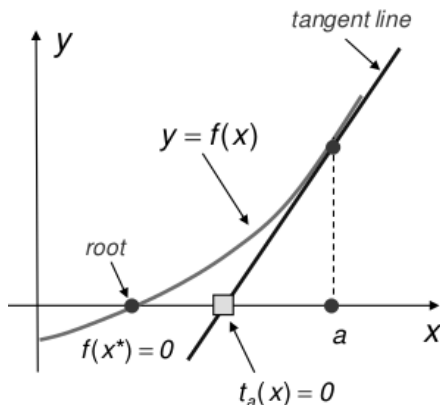


图 3_10 曲线和它的切线

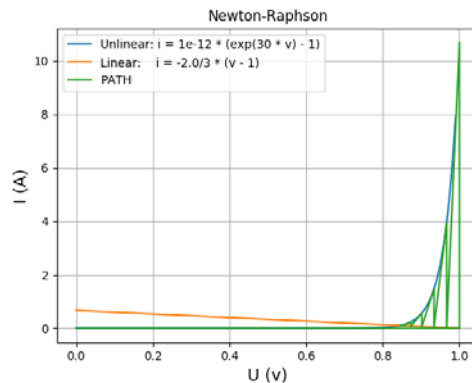


图 3_11 Newton-Raphson 方法

3.2.1 Diode 建模与 Stamps

Diode 的结构图示右图 3_12 所示：

其相当于非线性电阻，其模型的方程表示如下：

$$i_s = I_{sat}(e^{\alpha V_d} - 1)$$

其中：

$$\alpha = \frac{1}{nV_T}$$

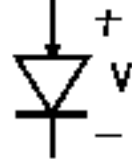


图 3_12 Diode

对于硅二极管来讲，其中 V_T 是热电压 (kT/q ，在常温下大约是 26 mV)，而 (n 被称为二极管理想因子)，对于硅二极管 n 约是 1~2)。因此而言， α 的值可以估算为：

$$\frac{1}{nV_T} = \frac{1}{2 * 26m} \rightarrow \frac{1}{1 * 26m} = 20 \rightarrow 40$$

其 Stamping 结果如下表 3_3 所示：

表 3_3: Diode 的 Stamp 表格

	Port Pos	Port Neg	RHS
Port Pos	$\alpha e^{\alpha V_D^{(n)}}$	$-\alpha e^{\alpha V_D^{(n)}}$	$-(e^{\alpha V_D^{(n)}} - 1) + \alpha e^{\alpha V_D^{(n)}} V_D^{(n)}$
Port Neg	$-\alpha e^{\alpha V_D^{(n)}}$	$\alpha e^{\alpha V_D^{(n)}}$	$(e^{\alpha V_D^{(n)}} - 1) - \alpha e^{\alpha V_D^{(n)}} V_D^{(n)}$

3.2.2 MOSFET 建模与 Stamps

MOSFT 作为四端口的元器件，如右图 3_13 所示。这里其 Body 端默认接地，所以不做讨论。

MOSFET 作为典型的非线性元件，其模型的方程为：

$$I_D = 0, \quad V_{gs} < V_{T0}$$

当 $V_{gs} > V_{T0}$ 时，其方程如下图 3_14 所示：

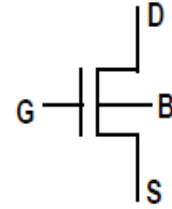


图 3_13 MOSFET

$$\begin{cases} I_{ds} = \frac{\partial I_{ds}}{\partial V_{gs}} = \frac{W}{L} * K' (2(V_{GS} - V_T) * V_{DS} - V_{DS}^2) * (1 + \lambda V_{DS}) & \text{if } V_{DS} < V_{GS} - V_T \\ I_{ds} = \frac{\partial I_{ds}}{\partial V_{gs}} = \frac{W}{L} * K' * 2(V_{GS} - V_T)^2 * (1 + \lambda V_{DS}) & \text{if } V_{DS} \geq V_{GS} - V_T \end{cases}$$

图 3_14 MOSFET 方程

为了保证，其在 $V_{DS} = V_{GS} - V_T$ 处连续，这里我们在 $V_{DS} \geq V_{GS} - V_T$ 时仍使用沟道调制系数 $1 + \lambda V_{DS}$ ，同时将其方程扩展到另半平面，其最后的结果如下图 3_15 所示：

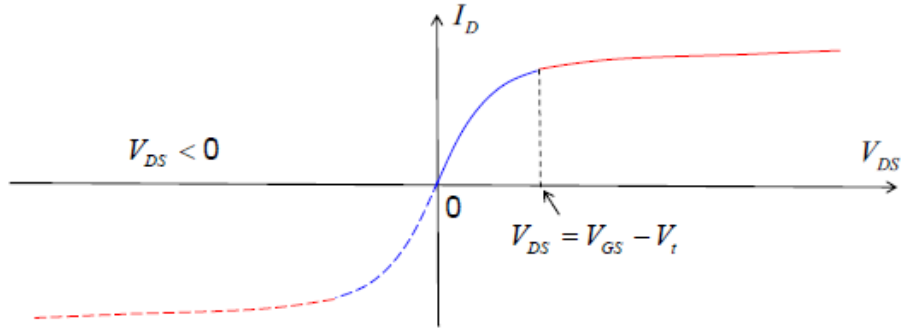


图 3_15 MOSFET 模型方程

由于在 Stamp 是需要用到 G_m 和 G_{ds} 方程, 这里给出部分, G_m 方程如下图 3_16 所示, G_{ds} 方程如下图 3_17 所示:

$$\begin{cases} g_m = \frac{\partial I_{ds}}{\partial V_{gs}} = \frac{W}{L} * K'(V_{DS}) * (1 + \lambda V_{DS}) & \text{if } V_{DS} < V_{GS} - V_T \\ g_m = \frac{\partial I_{ds}}{\partial V_{gs}} = \frac{W}{L} * K'(V_{GS} - V_T) * (1 + \lambda V_{DS}) & \text{if } V_{DS} \geq V_{GS} - V_T \end{cases}$$

图 3_16 G_m 方程

$$\begin{cases} g_{ds} = \frac{\partial I_{ds}}{\partial V_{ds}} = \frac{W}{L} * K'(2(V_{GS} - V_T - V_{DS})) & \text{if } V_{DS} < V_{GS} - V_T \\ g_{ds} = \frac{\partial I_{ds}}{\partial V_{ds}} = \frac{W}{L} * K' * 2(V_{GS} - V_T)^2 * \lambda & \text{if } V_{DS} \geq V_{GS} - V_T \end{cases}$$

图 3_17 G_{ds} 方程

因此, MOSFET 的 Stamps 如下图 3_18 所示:

	n_D	n_S	n_G	RHS
n_D	$g_{ds}^{(n)}$	$-g_{ds}^{(n)} - g_m^{(n)}$	$g_m^{(n)}$	$-I_{ds}^{(n)}$
n_S	$-g_{ds}^{(n)}$	$g_{ds}^{(n)} + g_m^{(n)}$	$-g_m^{(n)}$	$I_{ds}^{(n)}$

$g_m^{(n)} = \frac{\partial I_{ds}(V_{gs}^{(n)}, V_{ds}^{(n)})}{\partial V_{gs}}$

$g_{ds}^{(n)} = \frac{\partial I_{ds}(V_{gs}^{(n)}, V_{ds}^{(n)})}{\partial V_{ds}}$

图 3_18 MOSFET Stamps

其中:

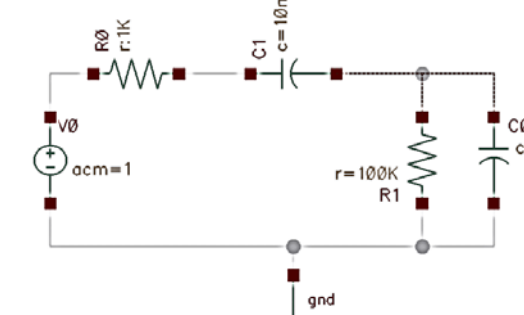
$$\boxed{I_{ds}^{(n)}} \triangleq I_{ds}(V_{gs}^{(n)}, V_{ds}^{(n)}) - g_m^{(n)} V_{gs}^{(n)} - g_{ds}^{(n)} V_{ds}^{(n)}$$

至此, 我们完成了对 MOSFET 的 Stamps, 值得注意的是, MOSFET 与 MOSFET 的连接中存在 Floating Point 的问题, 比如: NMOS 和 PMOS 组成的 CMOS 反相器中 Drain 端在没有负载的情况下即为 Floating Point, 因此我们需要通过给其赋初始值, 以保证其能在合适的位置开始迭代。程序体现在对 V_{ds} , V_{gs} 的初值和和在 Floating point 处添加上拉电阻或者下拉电阻等。

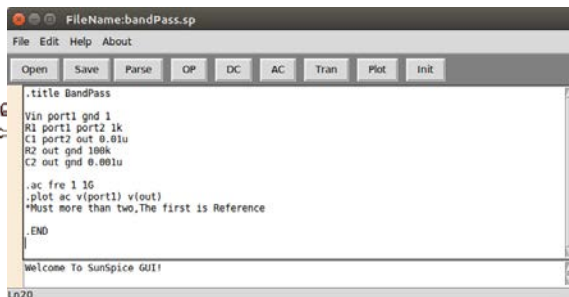
4. 结果举例

通过上面的实现, 我对一些简单电路进行了测试, 下面简要列出一些结果。

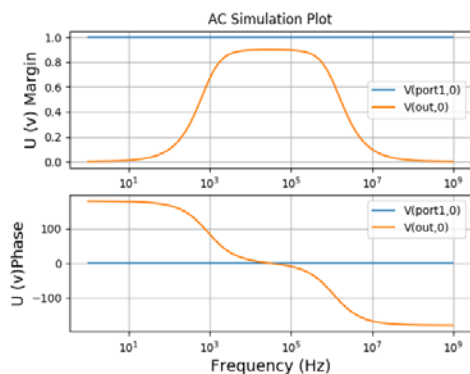
1. 带通滤波器实现



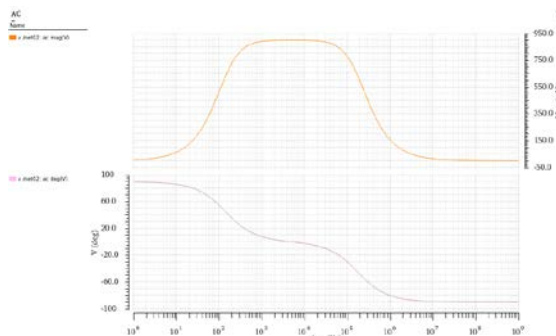
(a) 带通滤波器电路图



(b) 程序 GUI 与代码



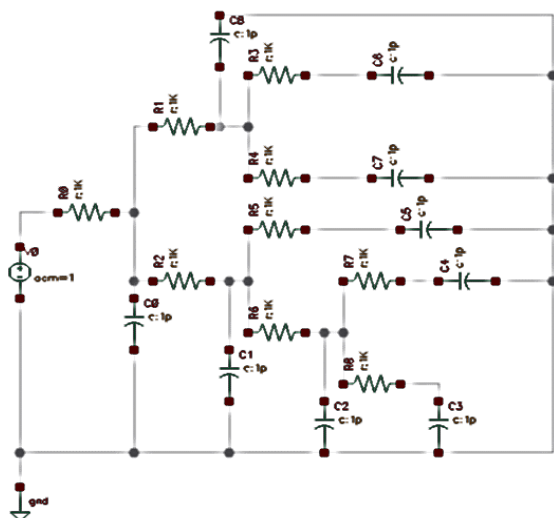
(c) Sun Spice AC 仿真结果



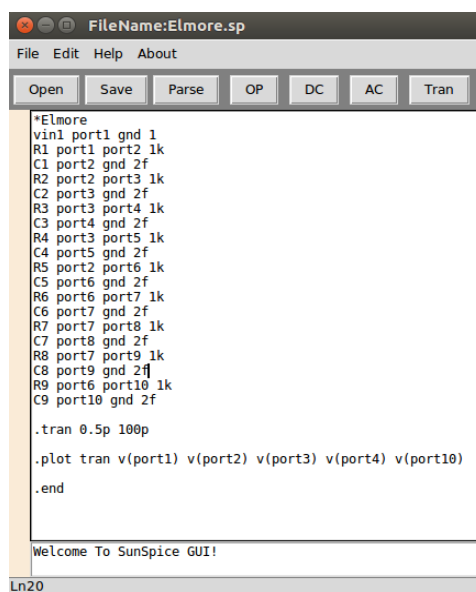
(d) Cadence 结果对比

图 5_1 带通滤波器实现和结果

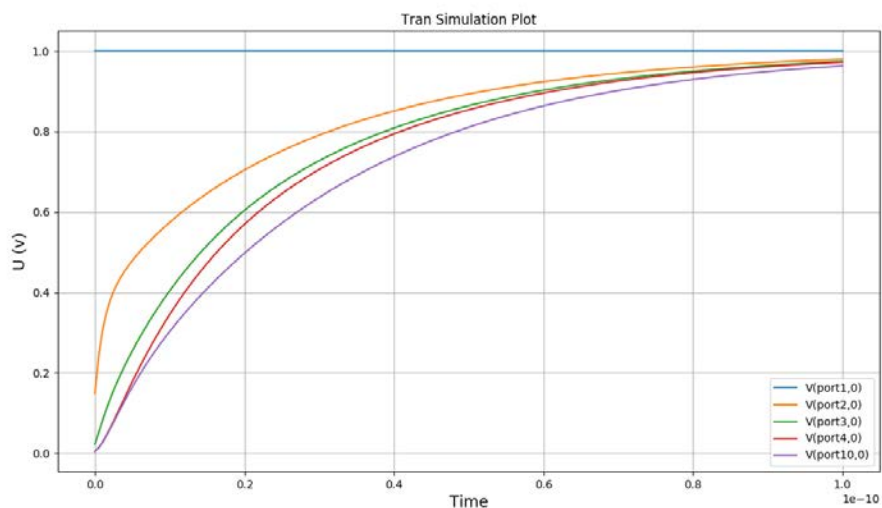
2. Almore 延迟电路



(a) Almore 延迟模型定义



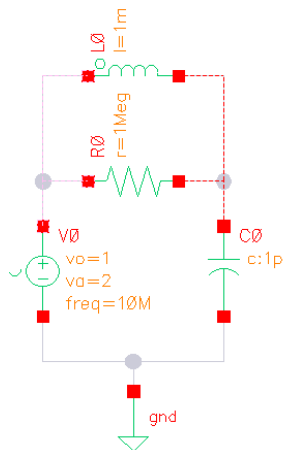
(b) Almore 延迟文本定义



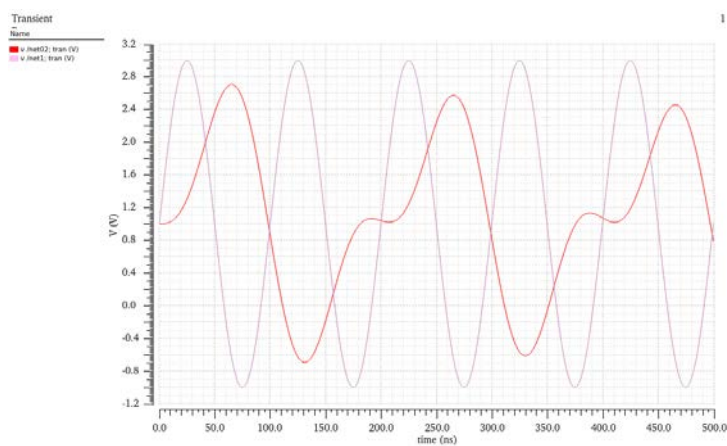
(c) Almore 延迟结果图（输入为阶跃函数）

图 5_2 Almore 延时输出结果展示

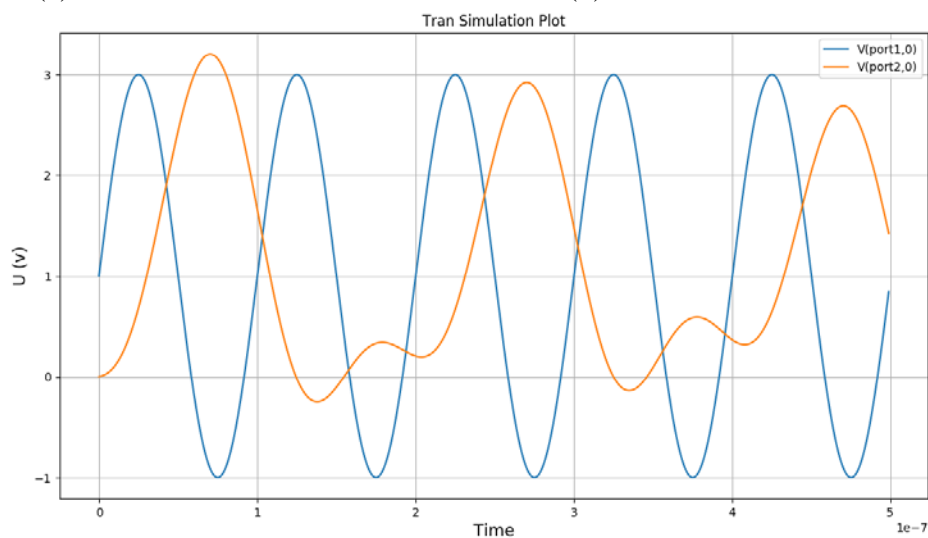
3. RLC 振荡电路



(a) 电路图



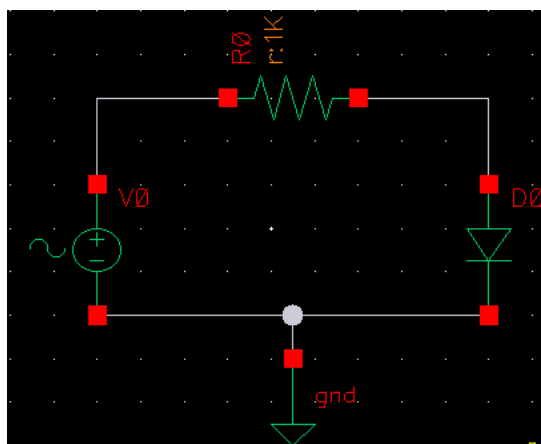
(b) Cadence 仿真结果对比



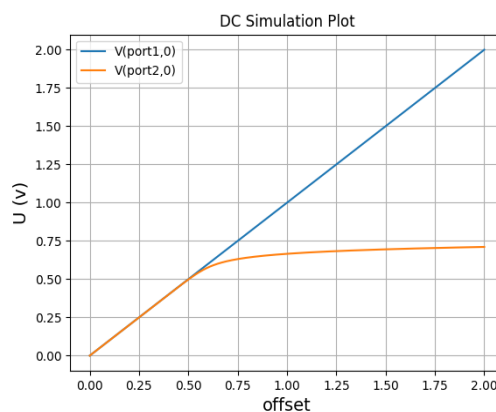
(c) SunSpice 结果展示

图 5_3 RLC 振荡电路仿真结果

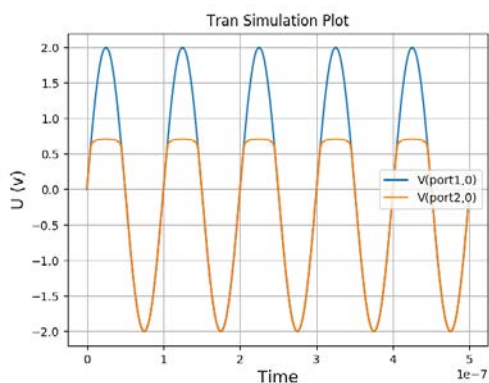
4. Diode 电路仿真



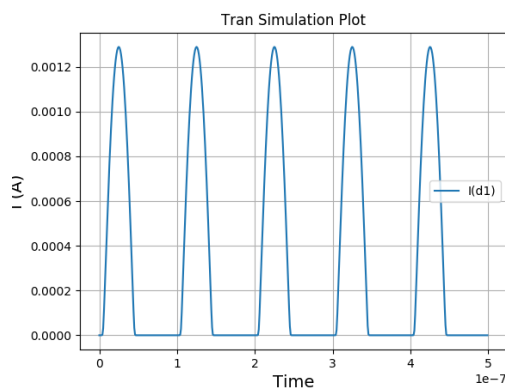
(a) Diode 电路图



(b) DC 仿真电压结果



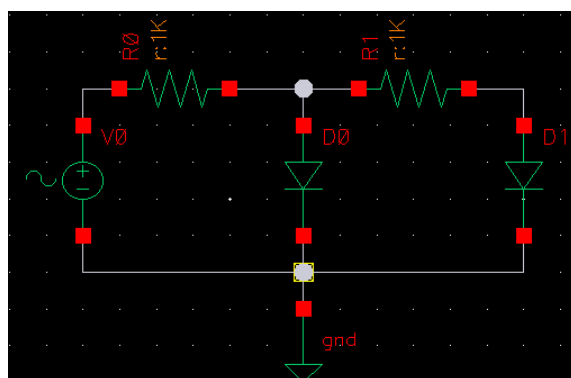
(c) Tran 仿真电压结果



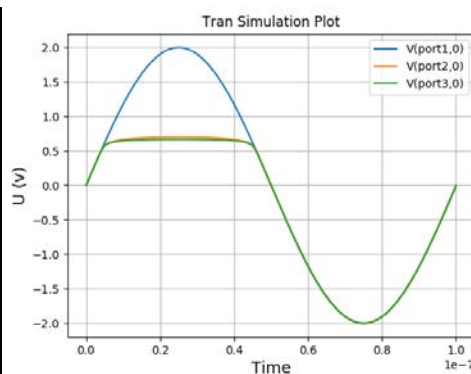
(d) Tran 仿真电流结果

图 5_4 Diode 仿真结果

5. 两个 Diode 收敛仿真



(a) 仿真电路图



(b) 仿真结果（电压）

图 5_5 两个 Diode 连接电路仿真

从上面的电路以及仿真结果中我们可以看出，其能够很好地表征 Diode 模型的特征，仿真结果与预期相符合。

6 MOS 管的仿真(CMOS 反相器)

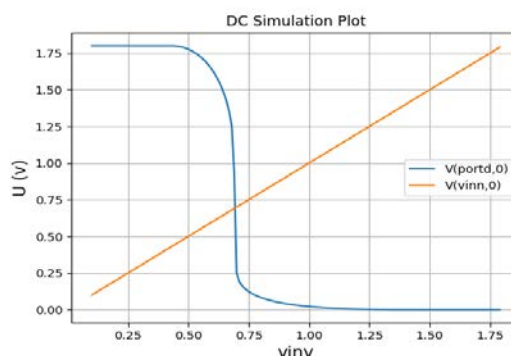
```

File Edit Help About
Open Save Parse OP DC AC
*Inverter
.param VinV=0.9
VDD1 portVDD gnd 1.8
Vin VinN gnd VinV
M1 portD VinN gnd 0 nmos w=1000n l=500n
*R2 portVDD portD 400k

M2 portD VinN portVDD gnd pmos w=1000n l=250n
*R1 portD gnd 100k

.dc VinV 0.1 1.8 0.01
.plot dc V(portD) V(VinN)
.end

```



(a) CMOS 反相器电路描述

(b) CMOS VTC 曲线

图 5_6 CMOS 反相器电路

7. CMOS 反相器的 Tran 仿真

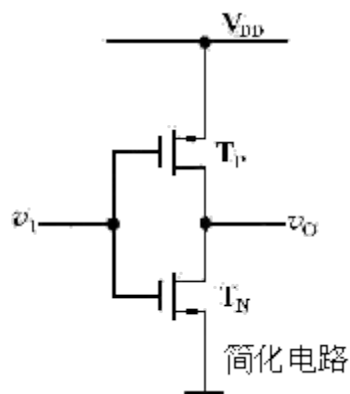
```

Open Save Parse OP DC AC
*Inverter
VDD1 portVDD gnd 1.8
Vpluse1 VinN gnd pluse 0 1.8 0 100p 100p 5n 10n
M1 portD VinN gnd 0 nmos w=1000n l=500n
R2 portVDD portD 600k

M2 portD VinN portVDD gnd pmos w=1000n l=500n
*R1 portD gnd 100k

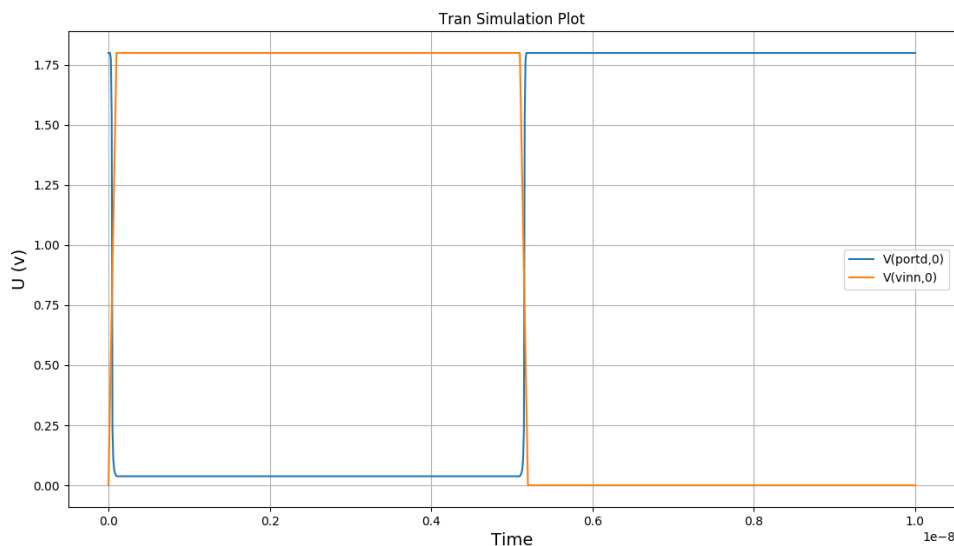
.tran 0.01n 10n
.plot tran V(portD) V(VinN)
.end

```



(a) CMOS 反相器的电路描述

(b) CMOS 反相器示意图



(c) CMOS 反相器的 Tran 仿真结果

图 5_7 CMOS 反相器 Tran 仿真结果

从上面的结果中我们可以看到, MOS 管的模型能够基本实现其相应的功能, 切能够正确的表征 NMOS\PMOS 的特性。

5. 收敛性问题

在 Diode、MOSFET 等非线性元件的实现过程中, 遇到的最大的问题就是收敛的问题, 如何使非线性元件能够很好地得到解析和收敛是一个难点, 在这方面也做了很多的工作, 但是由于理论的不足和相关学科知识的欠佳使得最后没能够在一些方面取得突破。下面简要介绍下遇到的相关问题。

首先是 MOSFET 模型的建立。我们知道, 普通的 NMOS 电压电流曲线仅会在 $V_{DS} \geq 0$ 的情况下使用, 但是, 由于在解决其收敛问题时采用的是 Newton-Raphson 方法, 需要通过迭代来完成收敛的值的求取, 这样就需要对 $V_{DS} < 0$ 的情况做一扩展。本文中作者尝试了老师提供的几种方案, 如下图 5_1 所示:

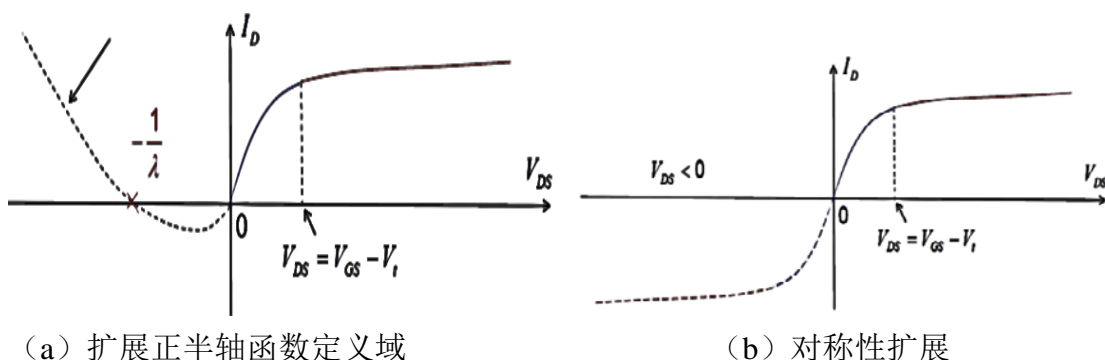


图 5_1 MOSFET 在 $V_{DS} < 0$ 的扩展方式

鉴于在方法 (a) 会在负半轴产生零点, 其收敛的位置可能会出现偏差, 经过对比和测试后, 采用了第二种方式。同时考虑到 MOSFET 的对称性, 其 Drain 和 Source 并不是一成不变的, 所以第二种方式有其合理性。因此在 $V_{DS} < 0$ 的区间相当于变换了其 Drain 和 Source 的位置。

但是另一种问题在于初始值的选择。第二种模式中会出现如下图 5_2 情况:

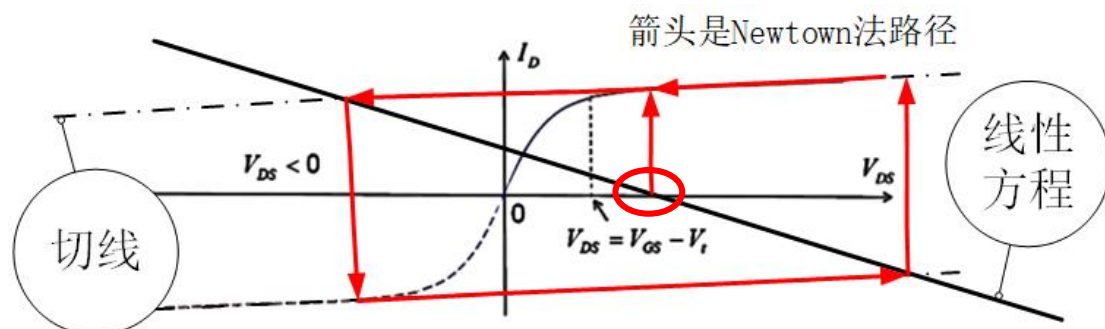


图 5_2 初始值与不收敛

在上图中我们发现, 当初始化的 V_{DS} 值为一些值时, 可能会造成上图所示的一种循环状态, 致使在结果上一直没有找到收敛的值, 在我的设计中, 设定不收敛时就一直运算, 这就导致程序界面“卡住了”, 而得不到结果。

上述问题的解决是基于 V_{DS} 的初始值得改进, 发现当我们的初始值较小时 (比如在线性区) 则不会出现上述情况, 因此可以通过找到合适的初始化值来完成相关的收敛性。

6. 总结

6.1 设计总结

在本文的设计中完成了一个简单的基于 SPICE 模型的仿真器的建立，完成了从电路网表解析，到器件模型建立和 Stamping，再到 AC/DC/Tran 的仿真方法的建立工作，最后将所得仿真数据通过作图曲线直观的表达，实现了电路元器件电阻 R、电容 C、电感 L、电压源 V、电流源 I、受控源 E(VCVS)、受控源 F(CCCS)、受控源 G(VCCS)、受控源 H(CCVS)、Diode D、MOSFET M 等模型的建立与仿真实现。能够完成基本的 AC、DC、Tran 等仿真模式，实现对包含上述元器件的电路进行直流工作点、时域分析、频域分析。

在设计中也存在一些不足和缺点，比如对 MOS 的仿真收敛性问题的解决不够充分，使得 MOSFET 仿真的收敛问题较大，在一些情况下不能完成收敛的解决方案。另一方面，在 Tran 仿真中，尝试的 Step Control 结构不太完善，致使在 Tran 仿真中存在着性能和速度效率的不足。

另一方面，在设计中，为了实现某些元器件的简单表达，省略了 Hspice 模型中的某些参数，比如对于电阻和电感而已，没有实现其初始值得赋值问题，其默认值为零状态响应；在 MOSFET 的设计中，省略了较多的参数的设计，比如面积 AS、DS 等，只保留了必要的参数 W 和 L；在 DC、AC、Tran 仿真命令语句中省略了 SWEEP 语句，不能实现对不同参数的扫描等。

因此，本文的设计在以下方面有待改进：

- 1) MOSFET 的收敛问题；
- 2) Tran 仿真的 Step Control 设计；
- 3) 器件模型的完善；
- 4) 命令语句的改进和完善。

6.2 感想与展望

在该项目的完成中，我了解了集成电路仿真器的基本原理和设计方法，掌握了一些基本元器件的模型建立，熟悉了通过计算机程序解决电路问题的基本方法。可以说收获很多。

在这门课程之前，仿真器对于我来讲是既熟悉又陌生的，熟悉的是我们很多的课程中都在使用，陌生的是对其中的原理和工作机制完全不了解。因此，在平时的使用过程中，也仅仅是照搬教程，按照步骤来操作。通过这门课程，熟悉其中原理的同时，也使用 Python 程序实现了自己的小的仿真器，巩固了其他课程所学习的数学、电路、计算机编程等知识，对以后再仿真器的使用上有很大的帮助。

上面也提到了在本次设计中的很多的不足之处，也是今后改善和努力的地方，争取在后续的完善中能够达到仿真其他课程作业的目标，实现自主化！

在本次课程设计的实现过程中，施老师在课程上的理论的讲解在设计的过程中有很大的帮助，同时施老师的指导和帮助学习到了解决问题的办法和思路，这里对施老师给予的指导和帮助表示衷心的感谢！