上海交通大学工学硕士专业学位论文

符号化仿真器用于CMOS模拟集成电路设计自动化的新进展[1]

硕士研究生：马迪铭

学　　号：1082109016

导　　师：施国勇教授

副 导 师：施国勇教授

申请学位：工学硕士

学　　科：电路与系统

所在单位：微电子学院

答辩日期：2010 年 1 月

授予学位单位：上海交通大学

[1] 本研究由国家自然科学基金（项目号 60876089）资助。

Dissertation Submitted to Shanghai Jiao Tong University
for the Degree of Master

NEW PROGRESS OF SYMBOLIC SIMULATOR APPLICATIONS ON CMOS ANALOG CIRCUIT DESIGN AUTOMATION

Candidate：	Ma Diming
Student ID:	1082109016
Supervisor：	Prof. Shi, Guoyong
Assistant Supervisor:	Prof. Shi, Guoyong
Academic Degree Applied for：	Master of Engineering
Speciality：	Circuits and System
Affiliation：	School of Microelectronics
Date of Defence：	Jan, 2010
Degree-Conferring-Institution：	Shanghai Jiao Tong University

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文《符号化仿真器用于 CMOS 模拟集成电路设计自动化的新进展》，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：马迪铭

日期：　　2010 年　　10 月　　25 日

上海交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

保密□，在___年解密后适用本授权书。

不保密□。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

符号化仿真器用于 CMOS 模拟集成电路设计自动化的新进展

摘 要

本文论述了一个基于符号化模拟电路仿真器求解模拟电路频域响应传输函数关于电路半导体器件（CMOS 晶体管和电容电阻等分立元件）尺寸敏感度的计算方法，介绍并分析了敏感度计算的价值和意义，阐明了电路器件尺寸敏感度和电路频域响应之间特殊的内在联系。本文同时给出了一个包含该计算方法和仿真器引擎的模拟电路交互设计平台和相关的设计案例应用。该平台通过一个用户友好的交互界面，帮助模拟电路设计人员直观、快速的获得最佳的电路器件尺寸，从而可以方便的进行模拟电路设计和分析。

本文第一章和第二章介绍符号化模拟电路仿真器的历史和原理，第三章则展示半导体器件尺寸敏感度计算方法的算法和实现，第四章具体介绍了整个符号化设计平台的搭建和构成，第五章则给出了两个具体的电路例子做说明和分析，第六章做了 CMOS 晶体管不同工作区敏感度求解的一些讨论，最后对全文作了总结和展望。

本文第一次发现了半导体器件尺寸频域响应敏感度和模拟电路之间的特殊内在联系，对理解电路行为具有很好的指导价值，对电路设计的自动化和设计方法学的革新具有一定的价值和意义。

关键词：模拟集成电路设计、频域响应敏感度、二叉决定图（BDD）、器件尺寸设计、零极点分析、符号化仿真

SYMBOLIC SIMULATOR APPLICATION ON CMOS ANALOG CIRCUIT SIZING

ABSTRACT

A symbolic calculation method based on the symbolic analog simulator for the sensitivity of frequency response to semiconductor device sizes is addressed for application in analog integrated circuit design in this paper. The transistor size-based ac-sensitivity can be used for sizing devices and understanding the circuit behavior. Examples are provided to demonstrate that a design platform supported by symbolic ac-sensitivity and visualization can be a helpful tool for computer aided design if analog integrated circuit.

The thesis is organized as follows. A brief introduction to the history and algorithm of symbolic simulator is in Chapter 1 and Chapter 2. The algorithms and implementations of symbolic sensitivity to device sizes are presented in Chapter 3. In Chapter 4, the symbolic analog design simulator platform is introduced. Example cases and applications are demonstrated in Chapter 5. Discussions on symbolic ac-sensitivity under different working region of transistors are presented in Chapter 6. Conclusions and future work are reported in Chapter 7.

The symbolic ac-sensitivity calculation in this thesis is probably the first on to reveal the connections between the sensitivity and the circuit behavior. It will probably help to promote the analog design methodology and automation.

KEY WORDS: Analog Circuit design, AC-Sensitivity, Binary Decision Diagram (BDD), Device Sizing, Pole-Zero Analysis, Symbolic Simulation

目 录

图 录

表 录

第一章 绪论

随着集成电路设计特征尺寸的不断缩小，集成电路规模得到了很大的增长。数字集成电路设计由于电子设计自动化工具的同步发展和支持，在这样迅速的工艺增长模式下，仍然能够维持其高度的自动化和高效的设计方法学。但是模拟集成电路设计却没有办法跟上集成电路发展的摩尔定律。在当今 90 纳米、65 纳米、45 纳米，甚至 30 纳米以下半导体制造工艺的技术工艺条件下，大量的模拟集成电路设计所采用的工艺制程仍然是 130 纳米甚至更早的工艺。究其原因，一方面是因为模拟电路本身在整块集成电路芯片中所占的面积和性能开销并没有像数字集成电路那样的紧迫性；另一方面，则是因为模拟集成电路设计长期处于半自动化或者全人工的设计方法学下。自动化程度的限制制约了模拟集成电路设计向大规模、深亚微米等先进工艺发展的脚步；而一个具备设计实际模拟集成电路能力和经验的资深工程师培养所需要的时间也进一步制约模拟集成电路设计自动化的发展。由于对于模拟集成电路设计需要相当深厚的电路理解和物理层经验，模拟集成电路设计是一个长期积累和经久学习的过程。在当今摩尔定律统治半导体行业的状况下，模拟集成电路设计对设计工具和设计方法学的自动化程度的要求越来越高。基于这样的研究动机和应用背景，结合实际的模拟电路设计需求，本文展开了一些关于模拟电路设计自动化的讨论和研究。

电路综合和电路仿真是模拟集成电路设计领域两个重要的课题。传统的模拟集成电路设计方法学，采用设计人员手动设置传输管的长度、宽度等重要的参数值并调用相应的仿真工具（比如著名的数值仿真工具HSPICE[1, 2]）加以验证的方式重复进行，直到仿真结果的电路指标参数达到电路的设计要求。由于传统的数值仿真工具采用迭代求解矩阵的方式进行仿真，对不同的参数值同等对待，而且也无法获得关键的器件和尺寸参数从而加以区分，因而导致传统的设计方法学虽然精准但是显得盲目、冗长而且设计周期非常长。相对于数值仿真的是符号化仿真[3-5, 19-21]，也就是将整个电路或者一个器件视为一个具备一定特征的黑盒，使用一个具体的近似公式描述这个黑盒的输入输出特性。由于公式的存在，电路或者器件对于其中某个参数的依赖关系和重要性就可以很清晰的显现，而电路求解也变成了简单的公式求值。数值仿真可以大大缩短电路求解的周期，降低盲目性，并可以方便的给设计者提供关键器件或者重要参数的描述。

符号化仿真的另一个优点是可以方便的对电路或者器件的行为进行趋势化描

述。利用黑盒公式，符号化的方法可以方便的对公式进行敏感度求解。而求得的敏感度数据恰恰就是这个趋势化特性的直接反应，是电路或者器件内在行为特征的一个天然表现。类似的工作数值仿真的方法也可以做，但是由于数值仿真器数值运算的特性，存在精度误差和无穷大过冲的不稳定情况，使得利用敏感度分析电路的方法一直无法获得很好的使用。

本文主要叙述的内容正是利用符号化方法求解电路和器件关于其参数的敏感度的方法，以及该方法所揭示的数据同电路本质属性的内在联系。同时本文还给出了从敏感度数据中获得电路关键器件和关键参数的方法，并由此辅助完成模拟集成电路设计自动化的设计方法学和设计流程。

1.1 模拟集成电路设计流程和自动化

传统模拟集成电路设计主要依靠设计者丰富的经验和渊博的知识，在仿真工具的帮助下反复尝试和验证，最后达到并实现设计指标。虽然近些年来关于模拟集成电路设计新的方法学的尝试不断涌现，各种新的综合方法和基于权重的目标最优化算法层出不穷，但是各种方法的局限性都很明显，模拟集成电路设计的设计流程和设计方法学并没有发生什么本质的变化。

1.1.1 模拟集成电路设计流程

图 1-1 给出了传统模拟集成电路设计的一个设计流程。

从图 1-1 中可以看到，模拟集成电路设计的流程中人工介入成分非常多，仿真工具的主要作用仅仅是提供实际的数据信息，设计指标的符合判断要有设计人员根据自己的经验和知识进行；而相应的，对设计电路的调整也必须由设计人员手动的完成。如果设计人员对设计的电路或者模块不熟悉或者缺乏经验，整个设计流程就无法完成。工具在设计流程中的作用相当有限，既不能有效地指导设计人员更好的设计，又不能自动对设计结果进行判断。这将造成整个设计的周期变得非常长、难度和复杂度都变得非常大。

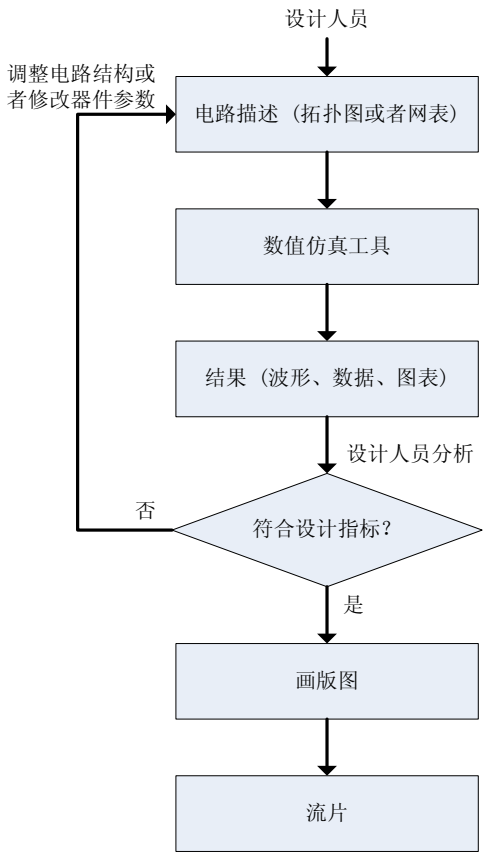


图 1-1 模拟集成电路设计流程图
Fig.1-1 Flow chart of analog integrated circuit design

1.1.2 模拟集成电路设计关键点

模拟集成电路设计实际上是一个反复尝试器件参数从而获得一个符合各个电路设计指标的过程。在这个过程中，参数和指标的关系是最为重要，也是最为关键的一个问题。模拟电路中指标和参数间有着复杂的关系。一个指标可能有很多参数共同影响，而一个参数也可以影响很多指标，而不同指标同参数的互相对立关系则使得设计难度进一步加大。

模拟电路设计人员之前经验的差别，就是体现在对电路的理解上，也就是指对电路指标和电路参数之间的依赖关系的掌控上。一个有经验的设计者可以很好的把握不同参数和指标之间的权衡。因而找到电路指标和参数之间的内在联系，或者说将这种内在联系以一种直观的方式展示出来，就成了模拟集成电路设计中的关键点。而这恰恰是自动化工具可以发挥能力的地方。

由于传统的数值设计方法采用的是迭代求解的方式，无法抓住实际电路指标

和参数的内在联系；而符号化的方式则天然的解决了这个问题。一旦电路的指标同电路参数的关系被直接的给出，设计就成了一件很简单的事情，年轻的设计人员可以迅速完成对电路的理解，而经验丰富的设计者也可以进一步明确设计方向。

1.1.3 模拟集成电路设计自动化

在模拟集成电路设计中，借助电子设计自动化工具的帮助，找到电路指标同参数的直接联系，并由此作出分析，指导设计人员完成设计，就是模拟集成电路设计的自动化。如果工具的能力足够强大，整个设计过程可以完全不需要设计人员介入，而是通过工具自动的智能分析，完成模拟集成电路的设计，这样就形成了模拟集成电路设计的完全自动化，也有的人称之为模拟集成电路的自动化综合。

相较于图 1-1，图 1-2 给出了模拟集成电路设计自动化的流程。

在流程中我们看到，自动化工具在设计人员提供了设计指标、预选电路拓扑结构和可选电路模块，以及制造工艺库的前提下，可以取代设计人员完成设计的整个过程。由于整个过程没有人为的介入和人脑的思考，设计周期大大缩短，设计复杂程度大大降低。

自动化工具完成自动化的指导依据就是工具自身发现的电路指标同器件参数之间的内在联系。显然这项工作天然的适合与符号化的方法。数值的方法虽然也可以完成类似的工作，但是所付出的时间代价是巨大的。

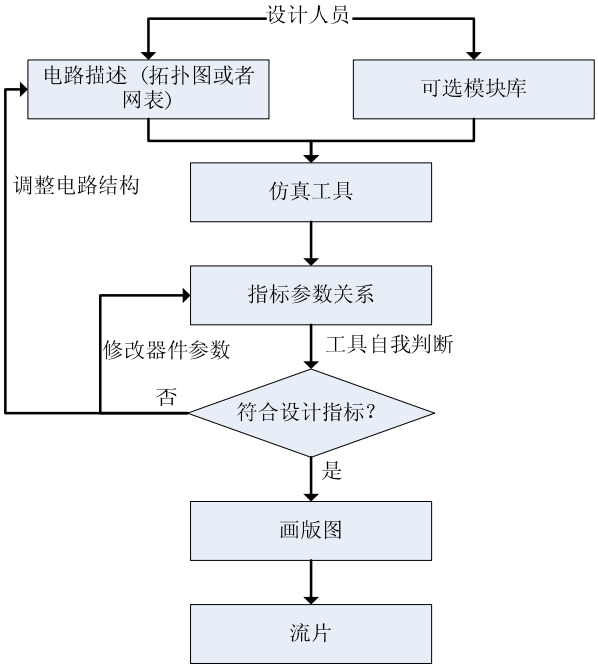


图 1-2 模拟集成电路自动化设计流程图
Fig.1-2 Flow chart of analog integrated circuit automation design

1.2 符号化分析的历史和方法

1.2.1 符号化电路分析的定义

电路的符号化分析本质上是一种形式化的方法。大量的符号化分析方法集中在对线性电路的频域特性的研究。对于一个集总的 (lumped) 线性时不变电路，我们总能使用一个频域的传输函数 H(s)来表征电路的输入输出特性。其中 H(s)可以具体定义为：

$$H(x)=\frac{Y(x)}{X(x)}=\frac{N(x)}{D(x)}=\frac{\sum_i x^i a_i(p_1,...,p_m)}{\sum_i x^i b_i(p_1,...,p_m)} \tag{1-1}$$

其中X(x)是输入的频域函数，Y(x)是输出的频域函数，当我们假设X(x)为 1 时，也就是对输出Y(x)做关于输入X(x)的归一时，传输函数就等价于输出。公式中的$a_i(\ldots)$和$b_i(\ldots)$是关于各个参数的函数，这些函数的值作为复数形式的频域变量x（离散时域电路的z变量或连续时域电路的s变量）的多项式系数，其中p_i就是各个参数的符号表征。完全符号化的电路的p_i对应的就是实际的电路元件；而部分符号化的电路p_i对应的则是实际电路元件同具体数值的组合。图 1-3 给出了一个实际的例子。

在这个例子中，我们得到符号化表示的传输函数为：

$$H(s)=\frac{1}{1+RCs} \tag{1-2}$$

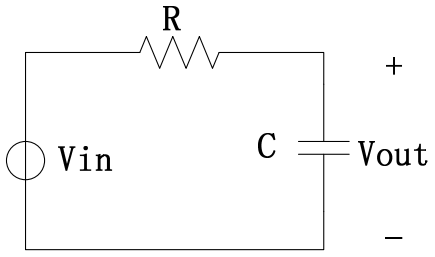


图 1-3 符号化电路分析举例
Fig.1-3 An example for symbolic circuit analysis

1.2.2 符号化电路分析的历史

从上世纪 60 年代末开始，符号化分析就已经被提出。当时符号化设计的应用和研究主要集中在模拟滤波器的分析。采用诸如符号流图[17]和生成树枚举[18]的方法进行的图形分析法在小规模整体均符号化电路中得到了一定的应用。由于数值分析工具SPICE[1, 2]的诞生和他处理大规模电路的强大能力，符号化的研究和应用被弃置，同数值方法的结合产生了符号数值混合分析方法。这一方法采用只处理频率一个符号，而其他均使用数值方法的方式扩大了符号化分析的应用能力。对符号化分析方法处理能力的扩大的需求，使人们提出了层次化分解的方法，这种方法将电路划分为不等的小电路块，逐块采用符号化的方法加以解决，一定程度的扩大了符号化的分析方法的应用范围。

在 2000 年以后，由于二叉决定图（BDD[23-26], Binary Decision Diagram）在符号化分析中的应用，使得符号化分析方法获得一个革新性的发展。行列式决定图法（DDD[28-30], Determinant Decision Diagram）就是一个很好的例子。

从上世纪 80 年代后期至今，在集成电路设计领域产生了许多符号化分析成功应用的例子，比如ISAAC[6, 7], ASAP[8, 9], SYNAP[10, 11], SAPEC[12], SSPICE[13], SCYMBAL[14], SCAPP[15], GASCAP[16], ZBDD[27], DDD[28-30], GPDD[31-33], ECD[36-38]等等。

1.2.3 符号化电路分析的方法

符号化分析方法通过将电路中的器件参数做一定的转换，然后用以表达电路的传输函数的方式实现电路的分析。由于符号化分析方法针对的是线性电路，非线性器件需要经过线性模型的提取才能转化为实际的电路参数。

符号化分析一般主要有以下几种方法[33]：

行列式决定图法 (DDD) [28-30]： 该方法主要采用符号化的方法表示电路的求解矩阵的解，从而符号化的表示电路的传输特性。早期大部分的符号化仿真器都是采用该方法。

符号流图法[17]：该方法根据梅森公式，依照一定的法则寻找传输特性中各阶项所对应的路径和回路，并计算结果。采用该方法的仿真器有ASAP等。

生成树枚举法[18]：该方法通过枚举由电路所转化而成的图的生成树的方式，实现电路传输特性的获取。电路图的每个分支都有各自对应的权重，而生成树项由这棵树每个分支的权重决定。本文中论述的符号化仿真器引擎GPDD所采用的就是这个方法。

参数提取法：该方法类似于行列式决定图法，但是不同之处在于，他所表达的符号完全对应于电路中的元件符号，因而需要复杂的算法对矩阵的元素进行符号的转化。该方法适用于部分电路符号化的分析。

数值插入法：该方法基于电路某些工作点的数值分析结果进行分析。该方法适用于仅以频率为符号的分析以及规模较大的电路。

以上五种方法又可以根据所生成的传输函数的符号是否与电路元件完全对应归为两类：代数方法和拓扑方法。

代数方法主要通过建立和求解电路方程来获取最终的结果。这类方法基本上不能保证电路元件与传输函数符号的完全对应。

而拓扑方法则不通过电路方程，而是将电路本身当作一个图来处理，通过一定规则下的图分解和图约化，实现传输函数的构建。这类方法因为没有额外引入其他符号，传输函数的符号一般是完全与电路元件对应的。

1.3 符号化分析同数值分析的异同

1.3.1 数值分析

数值分析方法的基本原理就是从电路和模型构建基尔霍夫电流定律以及基尔霍夫电压定律的方程组，然后采用各种数理代数的方法对方程组进行求解。将求得的数值解经过处理后以图表的形式呈现给用户。

数值分析方法是直接的，他像一个计算器简单快速的把用户的输入经由计算转化为输出，在很多场合下这是一个最佳的解决方法。但是他不是抽象成了数学公式的函数，每次电路的参数或者结构发生变化，整个计算流程就要重头重新来过一遍。如果电路的参数或者结构发生变化的频率很高，求解的代价就会变得很大。而在电路设计中，由于这种方法并没有给出参数或者电路结构变化的方向和建议，整个调整过程即便是对经验丰富的工程师来说都是很冗长的，有的时候甚至会变得完全盲目。

数值分析的本质是矩阵数值求解，是迭代或者拟合方法的应用。

1.3.2 符号化分析和数值分析的比较

符号化分析和数值分析的基本思路都是为了完成对电路的求解。但是采用的确实完全不同的两条思路。

数值分析类似于先秦时期的中国数学，以算术为主，在算术过程中各种专门

的、针对性的技巧被广泛应用，以追求更快的运算速度和更大的运算精度为目的；而符号化分析则类似于后来的西方数学，以归纳和推演为主，更多的采用变量和函数的方式去对各种算术进行归纳总结，并将函数的表达建立在以笛卡尔坐标系为基准的表示空间内。这样离散的数据关系被平滑的曲线简单明了的表示出来了，除了对原有问题的解的陈述，函数关系还给出了对未知的解的预测。在形式上是更高级的形式。

数值化的方法和符号化的方法由于本身各自的特点，因而也各具明显的优点和缺点。

数值化的方法的优点显然是快速、直接和精准。由于针对不同的情况有不同的优化和快速求解的特殊处理，所以特殊化的问题总能得到特殊化的求解。而现实中的各类问题总能找到它对应的一种分类。而且由于数值化方法本身的求解优化措施和求解空间的有限性，他的问题处理能力相对更大，可以处理大量大规模的电路，是某种意义上的按需求解。而符号化的方法的优点则是清晰的函数关系，换言之，就是对问题本身和问题解空间之间内在联系的清楚表述。同时符号化方法对结果的可预见性和解空间的涵盖能力都是数值化方法无法比拟的。符号化的方法的解空间其实涵盖了所有离散点的情况，因此是最大范围的求解方法。是某种意义上的按解求解。当然正是由于清晰的内在关系的提炼和解空间的完全性，数值化的方法在求解中的重复操作在符号化求解方法中全部可以省去，取而代之的是第一次求解成本的提升。但是对于求解频率更好的问题，符号化的方式优势就显得很明显了。

作为互为对比的两种方法，一种方法的优点就是另一种方法的缺点。数值化方法的最大缺点就是求解次数的增多的情况下，求解代价是线性增加的。而且数值化方法无法给出指导性的结果，只能简单的呈现数据对应的解。而符号化方法的最大瓶颈则在于内部表示的数据结构的节点数同问题的规模呈指数关系，同时受符号化方法所采用的符号顺序影响很大。由于符号化方法需要存储所有可能的解，因此内存的开销在很多情况下是冗余的。在问题电路规模较大或者符号顺序不好的情况下，内存的开销将会直接限制符号化方法的应用。另外，符号化方法在第一次求解的时候因为必须马上构建所有的解空间，因此求解速度会慢于数值求解方法。表 1-1 给出了这两种方法的比较。

表 1-1 数值方法与符号化方法的比较
Table 1-1 Numerical method VS Symbolic method

方法	求解速度	内存开销	解空间完备性	解启发性
数值化	1 次求解快	小	离散	无
符号化	1 次求解慢，多次求解快	大	完备	有

1.4 本章小结

本章主要介绍了模拟集成电路设计的传统流程和自动化流程。在介绍模拟集成电路设计的过程中引入了模拟集成电路设计仿真器常用的求解方法数值方法和符号化方法。本章给出了这两种方法的定义、原理和历史，给出了这两种方法的优缺点和比较。通过比较我们会发现，传统的仿真器求解方法，数值方法，能够很好的处理设计本身的问题，但是在考虑到自动化设计和加快设计流程和周期，尽可能降低设计经验对电路设计的制约的问题后，符号化的方法所具备的指导性和本质揭示性能力显然更具有竞争力。在当今集成电路制造和设计飞速发展的大背景下，电路的复杂度呈指数倍的提升，经验的积累速度已经渐渐跟不上半导体行业的发展速度了，因而自动化的设计方法学越来越显现出他的重要性和价值。

本文将重点描述一个完整的模拟集成电路设计自动化平台，并介绍这个平台采用符号化仿真器后给设计者提供的新的电路信息和具体的实现细节，帮助设计者更快更好的完成设计收敛，为基于符号化方法的模拟集成电路设计自动化提供一点新的想法和发展可能。

第二章 模拟电路符号化仿真器

本章将对模拟集成电路符号化仿真器做一个完整的、综述性的描述和介绍。这个仿真器是本文的基础，本文所做的所有研究和工作都是基于这个仿真器的拓展和进一步开发。

2.1 符号化仿真器历史和之前的工作

本文中提到的符号化仿真器，在发表的论文中被称之为GRASS (Graph Reduction Analog Symbolic Simulator)[31-33]，是我们实验室研究的核心方向之一。从研究的第一个人开始，符号化仿真器就一直不断给我们带来新的想法和实践的动力。本节会主要对符号化仿真器的研究历史做一个概要的介绍。本文的核心工作正是基于这些前期的研究工作的积累所展开的。

2.1.1 基本框架的构建：电路的符号化表示

符号化的方法的基本思想就是将电路转化为可以用符号化表示的公式集合。这些符号既可以是实际电路中的元件符号，也可以是矩阵中的阵列符号，或者是频域响应的频率符号。在获得转化好的公式集合后，电路的分析和相关的分析操作都会变得自然而然了。因此，从电路到电路的符号化表示是一个最基本也是最重要而且最关键的步骤。而这个步骤的完成是由陈薇薇[31-33]完成的。

电路的符号化表示涉及两个内容：一个是符号化表示的形式，也就是所采用的数据结构类型；另一个是电路的符号化转变的方法，也就是将电路转化为对应的符号化表示的算法和规则。

我们采用的符号化表示的形式是一个已经被证明非常高效、紧凑的数据结构二叉决定图 (BDD, Binary Decision Diagram)。利用二叉决定图的高压缩性和唯一性，我们可以将一个电路在按照一定符号顺序的情况下，紧凑的表示出来。关于二叉决定图的基本理论的研究已经很多也很充分了。他最初是由 Aker 在[32]中提出，目的是为了进行布尔函数的转化。后来 Bryant 将二叉决定图引入到了门级电路优化的研究中，提出了有序简化的二叉决定图，也就是 ROBDD (Reduced Ordered BDD)， 指出了二叉决定图在特定的符号顺序下，他的表示在经过冗余去除后必定是唯一的。现在所说的二叉决定图都是有序简化的二叉决定图。二叉决定图本质上是一个有向无环图，一个根节点，两个终点，分别为 0 和 1。每个节

点都有两个子节点，我们可以称之为 0 子节点和 1 子节点。二者分别表示一个决定的两个方面。比如假设我们取硬币的正面为 1，背面为 0，那么 0 子节点就表示抛了一次硬币结果为 0，而 1 子节点则刚好相反。习惯上用实线表示到 1 子节点的过程，而用虚线表示到 0 子节点的过程。图 2-1 给出了一个函数 f(a,b,c) = 2b+3ac 的例子，其中 a,b,c 都是符号，而 1 子节点表示乘法操作，0 子节点表示加法操作，边的权重是则系数。没有画出来的 b 和 c 的 0 子节点都是 0 节点。

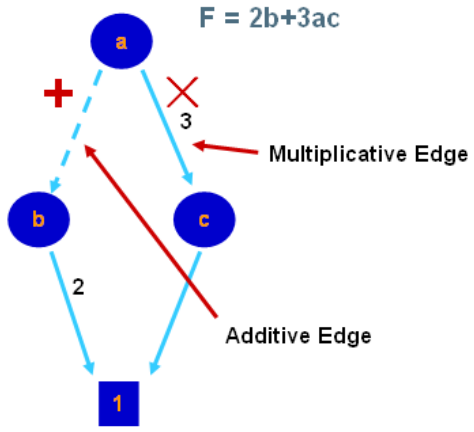


图 2-1 二叉决定图举例

Fig.2-1 An example for binary decision diagram

关于电路的符号化转变方法，则是由一系列规则构成的。本文的符号化仿真器在电路转化中采用的正是前面提到过的生成树枚举法。也就是将电路看成是一个由以各种线性元件（非线性元件则采用具体的模型将其替换为线性元件）为节点构成的符号化网络，对这个网络进行生成树枚举。生成树枚举的具体规则可以参照论文[33]。其基本思想就是对一个电路网络中的一条边做移除后短路还是断路的二叉决定，短路为 1，断路为 0。然后继续对做完决定的电路进一步枚举生成树，直到网络已经简化为一棵树为止。整个过程中，符号元件和以符号元件为中心的三元组都会做哈希，以保证节点和树的唯一性。经由这样过程枚举完并构建出的二叉决定图已经记录了所有的电路信息，从根节点开始，根节点的 1 子节点对应的是电路传输函数分子的信息，而 0 子节点对应的则是电路传输函数分母的信息；从这两个子节点往下，每条经由全部实边到达 1 节点的路径上的符号组合就是分子或者分母上符号多项式中的一个最小生成项。而且这些符号是与电路中线性元件一一对应的符号。图 2-2 给出了针对图 1-3 所获得的二叉决定图。X 表示传输函数入口，G 就是 R 的倒数，C 节点在传输函数中以 Cs 的形式呈现，整个二叉决定图表示的就是公式 (1-2)，即：

$$H(s)=\frac{N(s)}{D(s)}=\frac{-1\cdot -G}{G+Cs}=\frac{G}{G+Cs}=\frac{1/R}{1/R+Cs}=\frac{1}{1+RCs} \tag{2-1}$$

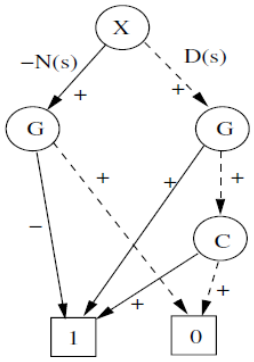


图 2-2 图 1-3 对应的二叉决定图[34]

Fig.2-2 Binary decision diagram for Fig. 1-3[34]

2.1.2 线性元件的敏感度求解

由于以线性元件为符号的图本身可以直接表示成相应符号顺序构建的二叉决定图，利用二叉决定图数据结构上的天然优势，可以很方便的对电路的传输函数进行关于某个特定符号，也就是某个特定线性元件进行敏感度求解。

敏感度是电路某一个特性关于某一个参数的敏感程度，是电路该特性关于该参数导数的归一化表示。敏感度的定义[22]可以表示为：

$$Sens(H(s),p):=\frac{p}{H(s)}\cdot\frac{\partial H(s)}{\partial p}=\frac{\partial \ln H(s)}{\partial \ln p} \tag{2-2}$$

传输函数关于某个参数的敏感度就是指该参数变化下对应传输函数变化的幅度和程度。

对于二叉决定图这样特殊的数据结构，如果我们采用布尔运算的表示法设定二叉决定图，也就是每个符号都是布尔符号（只有 0 和 1 两个值），实边表示乘法操作（也就是布尔与运算），虚边表示加法（也就是布尔或运算），那么对这样的二叉决定图进行求导操作，只需要简单的将被求导的符号节点的虚边所指的 0 子节点设为 0 节点，而将符号的值设为 1 即可。图 2-3 给出了图 1-3 对应的二叉决定图（图 2-2）关于 G 也就是 R 求导的结果。

由于我们有以下关系（具体证明请参考[34]）：

$$Sens(H(s),R)=-Sens(H(s),G) \tag{2-3a}$$

$$Sens(H(s),p)=-Sens(X(s),p) \tag{2-3b}$$

$$Sens(H(s),R)=Sens(X(s),G) \tag{2-3c}$$

$$Sens(H(s),p)=Sens(N(s),p)-Sens(D(s),p) \tag{2-3d}$$

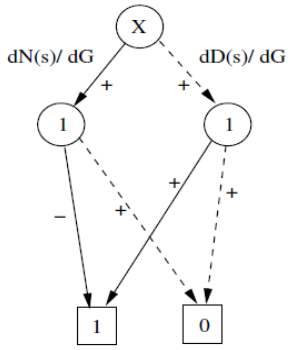


图 2-3 图 2-2 关于G (R) 求导后对应的二叉决定图[34]

Fig.2-3 Binary decision diagram for Sensitivity w.r.t. G(R) of Fig. 2-2[34]

因此我们可以从图中获得（图中我们可以得到分子和分母关于 G 的求导结果都是 1）：

$$Sens(H(s),R) = Sens(X(s),G) = -Sens(H(s),G)$$
$$-Sens(N(s),G) + Sens(D(s),G) = -\frac{G}{G}\cdot 1 + \frac{G}{G+Cs}\cdot 1 = \frac{-Cs}{G+Cs} \qquad (2\text{-}4)$$

而这和我们直接对公式 (2-1) 的结果是完全一致的。

2.1.3 接口和平台化构建

符号化仿真器要想发挥它的作用，就要将这个仿真器内嵌于一个实际的、接口完备的平台化工具中。本文所论述的自动化平台的前身正是本章所提到的符号化仿真器的接口和平台。

仿真器是一个用以对模拟电路进行分析和提取的引擎工具。它的输入是特定的电路网表和输入输出端口设置，它的输出是他构建的二叉决定图以及相关的有效数据，比如对某一特定的线性元件的敏感度值。要将这个仿真器很好的前后衔接起来，就需要提供专门的接口给这个仿真器，用以向仿真器传递输入并接收输出。GRASS的接口的雏形化设计是由李骥[35]完成的。

在这个原型设计中，输入电路采用标准的HSpice网表格式，并辅以GRASS可以识别的端口定义语句（以注释的方式给出），使用已有的开源词法句法解析工具PCCTS (Purdue Compiler Compiler Tool Set)[33]进行网表的解读，从而完成从电路拓扑结构到对应的以电路线性元件为符号的图的构建。然后进行电路图到二叉决定图的提取的转化。在完成二叉决定图的提取和转化后，根据获得的传输函数，直接构建电路的频域响应的幅频和相频曲线，以 2D和 3D的图形化接口的方式呈现给设计者，让设计者直截了当的获得电路的相关信息。图 2-4 给出了这个过程。

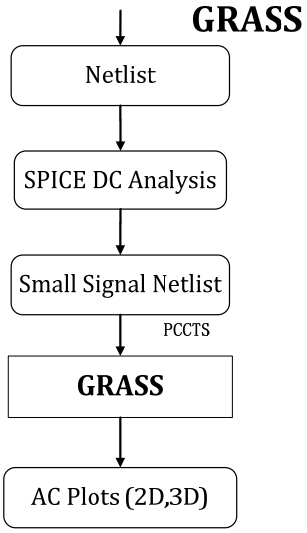


图 2-4 GRASS 流程图
Fig.2-4 Flow chart of GRASS

在整个流程中，由于负责人不同，每一个环节的完成都基本上是通过手工的调整接口格式和内容完成的。参数提取部分的工作几乎是针对每一个测定的电路有一个特定的提取脚本和设置。对网表端口的设置也必须通过手动的添加相关声明来完成。虽然接口的话工作已经有了雏形，但是接口的衔接和自动化还完全处在原始状态。

GRASS 的接口采用 C/C++以及 Tcl 脚本搭建，终端的图形化用户接口（GUI）采用 GTK/OpenGL 的方式构建，整个 GRASS 的编译通过 Makefile 联系组合到一起。

2.1.4 主极点的符号化求解

主极点是模拟集成电路设计中非常重要的一个参数指标。主极点的位置以及主极点同其他极点以及零点的距离对模拟集成电路设计有重要的影响。很多模拟集成电路设计中的重要指标都是和主极点直接联系在一起的，比如 3db 带宽，单位增益频率等。而符号化的仿真器因为采用了先进的二叉决定图对电路传输函数进行表示，因而对主极点的符号化表示也变得相对简单了。而主极点一旦得以被符号化表示，那么电路的主极点分析将会从数值方法的数据为主升迁到符号化方法的关键器件为主。从主极点的符号化表示中我们可以方便的、直截了当的看到影响主极点的主要电路线性元件和这些线性元件的影响途径和方式。这样就省去了模拟集成电路设计人员复杂而繁冗的电路分析和公式简化，直入主题。

关于主极点的符号化表示有很多相关的研究和算法。我们采用的是通过矩的方式进行求解。所谓的矩就是电路传输函数的多项式表达形式下的系数。电路的0阶矩就是电路传输函数频率项指数为0的系数，也就是直流增益值，而电路的1阶矩则是电路传输函数频率项指数为1的系数，也就是电路的Elmore延时，余下的依此类推。我们采用符号m_k表示电路的第k阶矩，于是我们有：

$$H(s)=\frac{N(s)}{D(s)}=\frac{b_0+b_1s+\cdots+b_qs^q}{a_0+a_1s+\cdots+a_ps^p}=m_0+m_1s+m_2s^2+\cdots \tag{2-5}$$

其中$m_0=b_0/a_0$。

对于主极点，我们可以有如下的关系：

$$\begin{cases} a_0m_0=b_0 \\ a_0m_1=b_1-a_1m_0 \end{cases} \Rightarrow p_d=\frac{m_0}{m_1}=\frac{a_0b_0}{a_0b_1-a_1b_0} \tag{2-6}$$

因此我们只需要从我们构建的二叉决定图中获得a0,b0,a1,b1这四个参数的值就能近似获得主极点的值。而获得这四个参数的值只需要对二叉决定图进行一次遍历即可。因此获得主极点的代价非常的小，而精度却很高。

主极点的符号化表示为我们利用二叉决定图进一步获得电路特性打开了一个很好的口子。从主极点的符号化表示中我们可以获得很多有用的信息。而从主极点扩展开去以及衍生出来的诸多电路特性的符号表示，比如主极点的敏感度，则进一步揭示了电路的内在特性同电路参数之间的联系，为模拟集成电路设计的自动化提供了有力的支持，同时也为模拟集成电路设计者提供了有用的导向性信息。

2.2 符号化仿真器的原理

符号化仿真器的原理相对来说还是比较复杂的。作为整个算法和数据结构的核心，整个电路到二叉决定图的转换过程有着严格的数学证明和规则引导。具体的理论内容可以参见论文[33]。本节只对其中的关键算法和主要思想做概述。

2.2.1 二叉决定图的构建

二叉决定图的构建过程是一个在一系列规则指导下的电路图的约化过程。算法首先将根据电路图构建一个对应的电路有向图（有向图中受控源或者独立源转化成有向边，电压源的方向沿着正极到负极，电流源的方向沿着电流方向；对应电压控制支路添加一条有向边，对应电流控制支路添加一个结点和一条单独的有向边），然后将这个有向图拆分成左右子图（分别用以生成传输函数的分子和分母），再然后对左右子图分别进行约化操作，直至处理完有向图中的所有的边为止。

整个过程可以理解为对于传输函数中的多项式表示的每一个最小生成项，也就是这里进行枚举的每一棵生成树，进行一个电路元件符号在（1 子节点）与不在（0 子节点）的判断操作。当把左右的判断操作都完成，则对应的二叉决定图就是所有在与不在组合的集合，因此也就包含了整个多项式表示的所有最小生成项。整个约化的过程就是不断重复对一个电路元件符号在与不在枚举的判断操作的过程。图 2-5、图 2-6 和图 2-7 分别展示了电路有向化、图拆分和图约化的过程。

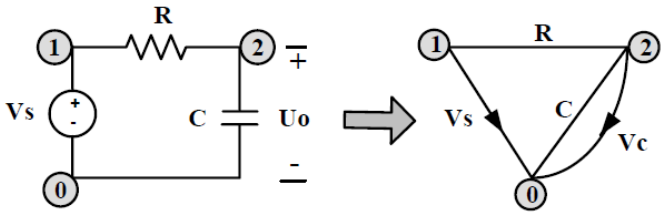


图 2-5 电路图的有向化[33]
Fig.2-5 Directed graph of the circuit[33]

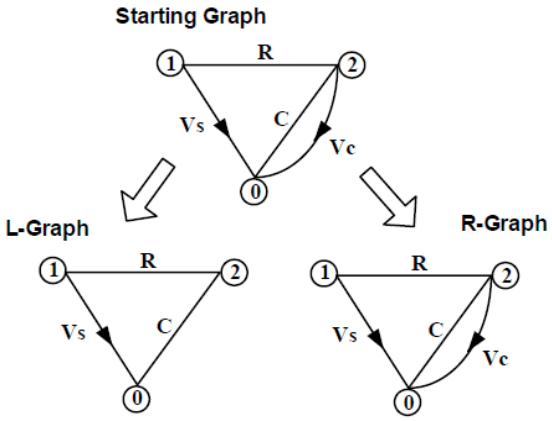


图 2-6 有向图的拆分[33]
Fig.2-6 Split of directed graph[33]

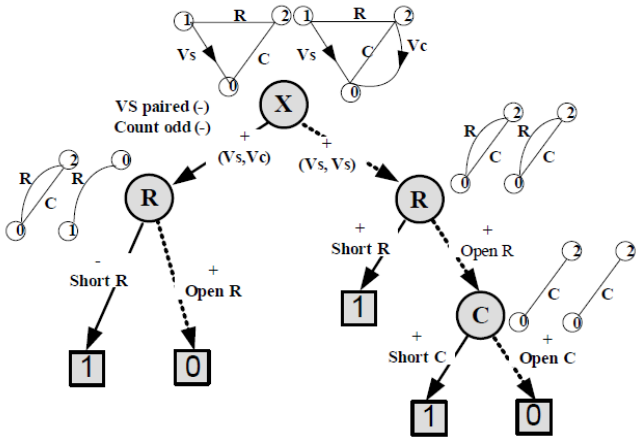


图 2-7 有向图的约化[33]
Fig.2-7 Reduction of directed graph[33]

在对应的二叉决定图构建完毕后，为了降低二叉决定图的存储空间，同时也为了保证在特定符号顺序下二叉决定图的唯一性，我们会对二叉决定图做一次压缩，对重复的三元组对进行哈希。压缩后的二叉决定图是唯一的。图 2-8 展示了可以共享的三元组的压缩例子。

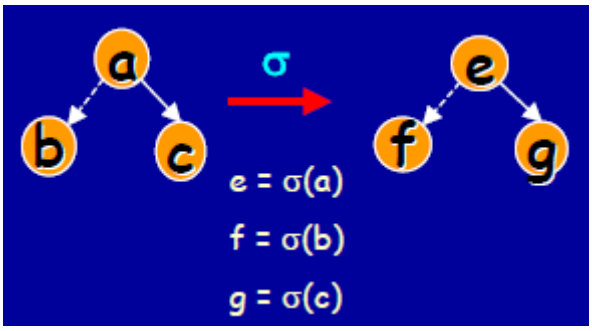


图 2-8 同构三元组
Fig.2-8 An example of isomorphism

在二叉决定图的构建中，还需要进行节点符号的设置。节点符号是用以在求值过程中保证求得的最小生成项的符号正确性的重要因素。节点符号的设置算法可以在[33]中找到。

2.2.2 二叉决定图的求值

在获得二叉决定图之后，我们就需要对二叉决定图进行求值。求值操作是通过对图的遍历完成的。基于动态规划的思想，每个节点都是以该节点为根节点的二叉决定图的值，因此对整个二叉决定图的遍历只需要对每个节点进行一次访问即可。所以对二叉决定图进行一次遍历的代价和图中的节点数量成正比。

由于我们的二叉决定图表示的是传输函数，而符号化表示的传输函数的每一个最小生成项对应的就是二叉决定图中每条通往 1 节点的路径中含有 1 子节点的符号的与（乘法）。因此我们对每个二叉决定图的节点的求值就是对一个他的 1 子节点进行与（乘法）操作并同它的 0 子节点进行或（加法）操作。这个操作的结果的值就是该节点的值。依此类推，在一次遍历结束的时候，根节点的值就是传输函数在当前电路拓扑结构和参数值下的值。如果电路的参数值发生变化，由于二叉决定图中的符号和电路中的线性元件的符号都是一一对应的，所以只需要更新相应的符号的值并对二叉决定图再进行一次遍历即可。只有当电路的拓扑结构发生变化的时候，二叉决定图才需要重新构建。因此整个电路的二叉决定图的求

值是非常快速的，代价是非常小的。

2.2.3 二叉决定图的存储

由于二叉决定图存储了整个传输函数所有的最小生成项，因此在内存开销上很大的。同时由于一棵二叉决定图的节点个数同电路的线性元件（符号）的个数成指数关系，为了降低存储二叉决定图的空间大小，内存管理显得尤为重要。

二叉哈希表被用来存储每一个二叉决定图中的节点和每一个二叉决定图中的三元组对。同时软件结构的高速缓存被用来降低局部效应带来的额外开销。

为了进一步降低二叉决定图的大小，我们将二叉决定图中并联的电阻符号加以合并，以“^”开头的新符号代替原来的一组符号，有效地降低了符号的数量。

2.3 符号化仿真器的结构和实现

2.3.1 符号化仿真器的结构

符号化仿真器的结构如图 2-9 所示。它由词法语法解析器（也就是本文上面提到的 PCCTS 工具）、仿真器内核引擎（完成电路的有向图表示和二叉决定图的构建）和求值运算器（完成各种二叉决定图的求值和数据处理）构成。同时整个符号化仿真器因为涉及非线性元件的线性化问题，需要一个标准的电路仿真工具（HSPICE）做直流工作点分析，然后完成模型参数的提取（这部分的内容会在第三章中提及）。在电路结构不变的情况下，直流工作点的分析工作只需要做一次。

2.3.2 符号化仿真器的实现

符号化仿真器 GRASS 的实现在原理部分已经做了主要的表述。很多具体的实现都非常的常规，比如二叉决定图的数据结构表示和构建，二叉哈希表的建立和相关操作，三元组的哈希以及相关的 ITE（If-Then-Else）[26]操作，二叉决定图的遍历和求值等，大部分的实现都是有论文或者有研究可查的。本节主要简单介绍在生成树枚举过程中二叉决定图的构建的非递归算法。

如果采用递归算法，那么对于一棵二叉的树的构建是非常简单的，无论是采用前序、中序、还是后序遍历，都能够很好的处理。但是因为递归算法本身要保护上下文，建立函数堆栈等各种操作，由此产生了一些因存储无用上下文信息而形成的不必要的开销。因此我们采用了一个标准模板库的堆栈来代替递归算法，只保留节点指针作为有效地上下文信息。这样的算法可以被做如下描述：

Step1. 将根子树节点压栈。

Step2. 查看堆栈是否为空，为空则跳至 Step7，否则继续。

Step3. 取出栈顶节点，若该节点的 1 子节点和 0 子节点都已构建完毕，则弹栈该节点，回到 Step2，否则继续。

Step4. 若取出的节点的 1 子节点构建完毕而 0 子节点尚未构建完毕，则压栈该节点的 0 子树，回到 Step2，否则继续。

Step5. 若取出的节点的 1 子节点尚未构建完毕，则压栈该节点的 1 子树，回头 Step2。否则继续。

Step6. 进行子树约化，构建节点。回到 Step2。

Step7. 二叉决定图构建完成。

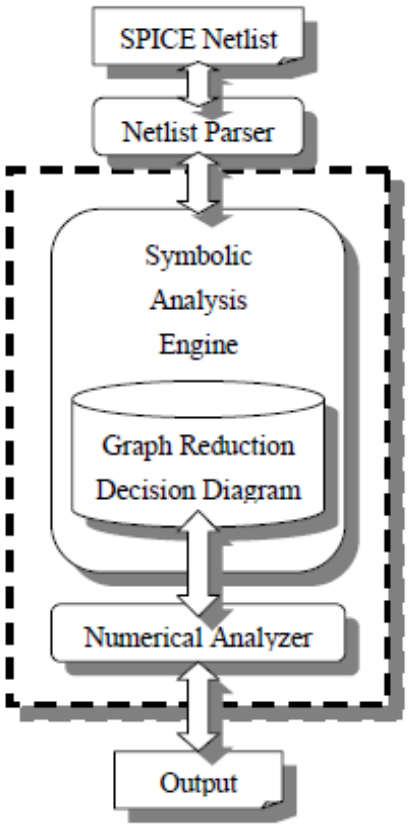


图 2-9 符号化仿真器结构[33]

Fig.2-9 The structure of symbolic simulator[33]

2.4 符号化仿真器的能力

因为采用了二叉决定图的数据结构和各种内存管理以及算法上的优化和改

进，符号化仿真器的能力得到了很大的提升，能够处理的电路规模达到了 10-20 个CMOS晶体管和 20-30 个BJT晶体管。对于像ua741 和ua725 这样的复杂模拟电路，符号化仿真器的建立时间和求解时间总共都只花了 1.9s和 22.6s；相应的内存开销为 34.78MB和 358.7MB[31-33]。

正是符号化仿真器强大的能力，才使得本文下面叙述的相关算法和相关的电路规模的符号化求解和应用得以展开。

2.5 本章小结

本章主要介绍了符号化仿真器的原理、结构和一些实现细节。通过对符号化仿真器的发展历史和开发研究过程的描述，展示了符号化仿真器的强大能力和诸多优化。符号化仿真器是整个研究和模拟集成电路设计自动化的基础和核心，是本文研究得以展开的保障和基石。从本章中可以看出，本文的研究是基于先前若干研究者的研究和努力的基础上，进一步发展进行的。

第三章 CMOS晶体管尺寸符号化敏感度

3.1 CMOS晶体管尺寸符号化敏感度的意义和作用

CMOS 晶体管尺寸的敏感度就是指改变 CMOS 晶体管的宽度、长度或者宽长比的值对 CMOS 电路的传输函数的影响程度。一旦我们知道了 CMOS 晶体管尺寸的敏感度，我们就可以有方向的、有目的的调整关键 CMOS 晶体管的相应宽度、长度或者宽长比的值以获得我们需要的电路响应特征和关键参数的值。快速的完成设计的收敛。

相比于数值的敏感度，符号化的敏感度除了反映电路传输函数关于每个 CMOS 晶体管的尺寸变化的影响程度，还展示了在频域上这个影响程度的变化趋势和变化范围，进一步揭示了电路频域特征和晶体管尺寸的关系。完成了电路的零极点响应、频域带宽等特征同晶体管尺寸的直接联系。

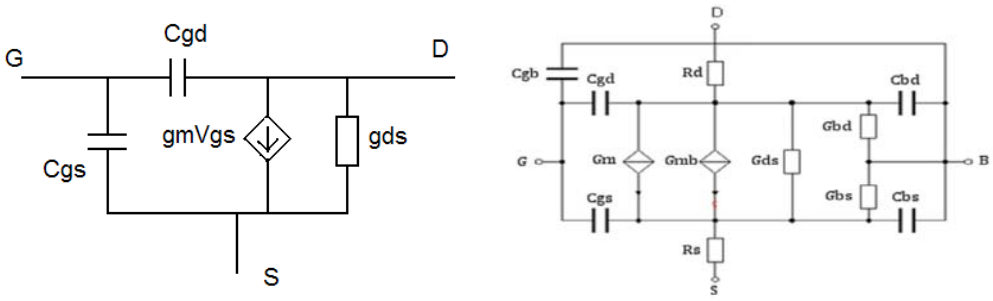
本文提供的 CMOS 晶体管尺寸符号化敏感度在频域的表示，第一次给模拟集成电路设计人员提供了一种了解电路行为和电路特征的新角度和新方式，是全新的电路数据的呈现，对模拟集成电路设计人员来说是一个有趣的尝试。

3.2 非线性元件的线性化

由于符号化仿真器本身的限制和制约，符号所对应的电路元件必须是线性元件，这就迫使我们将电路中存在的大量的非线性元件转变为线性元件加以表示。而转变的基础就是非线性元件的线性化模型。

对于CMOS晶体管的线性化模型包含很多层次，有简单的大信号模型（主要用以求解直流工作点）和各种复杂程度的小信号模型（主要用以进行频域响应的表征，包括BSIM1、BSIM2、BSIM3、BSIM4、BSIM6、RF[1, 2]相关的模型等）。各个模型之间的复杂程度差别很大，而且所采用的线性元件的数量和丰富程度也千差万别。图 3-1 给出了两种常见的CMOS晶体管的信号模型。

复杂程度越高的 CMOS 晶体管模型在仿真中所获得的精度越高，当然所需要的求解代价无论是对数值化的仿真器还是符号化的仿真器来说也都更大。因此选择合适的晶体管模型是确立仿真器仿真效率的基础，我们必须在精度和效率之间做出一个权衡。



a) 简单的 CMOS 信号模型　　b)相对复杂的 CMOS 信号模型

a) Simple CMOS model　　b) Complex CMOS model

图 3-1 常见 CMOS 晶体管信号模型

Fig.3-1 Common CMOS transistor models

为了获得更好的求解效率，同时降低符号化仿真器在构建二叉决定图时的内存开销，本文选择图 3-1b 的晶体管模型作为我们符号化仿真器的 CMOS 晶体管模型。

为了获得每个电路中 CMOS 晶体管模型中的线性元件的参数值，本文采用图 3-1a 的模型作为电路求解直流工作点的大信号模型，并采用调用权威数值仿真器 HSpice 求解的方式获得。因为 HSpice 采用数值迭代的方法在求解大信号模型上具有很快的收敛速度和计算性能，因此采用 HSpice 完成这项工作的代价非常小。

在调用 HSpice 对电路做过直流工作点的仿真后，我们就要对仿真获得的小信号模型中的线性元件的参数值进行提取，以完成非线性元件的线性化。

关于非线性元件的线性化的模型的选取，其实可以根据 CMOS 晶体管在电路中的关键性采用不同的复杂程度的模型。对关键的晶体管采用更复杂、更高精度的模型，而对不重要的晶体管则可以适当降低模型的复杂程度，以换取求解精度和效率的更好的权衡。这个部分相关的研究工作仍在进行中。

3.3 CMOS晶体管小信号模型参数的提取

对于 CMOS 晶体管小信号模型参数的提取本文采用 Tcl 脚本的方式进行。

在调用 HSpice 进行直流工作点的求取后，HSpice 将会生成一个以 lis 为后缀的文件。文件中记录了所有的中间过程和求值结果。采用 Tcl 脚本逐行处理这个文件，当发现文件数据属于某个 CMOS 晶体管或者文件数据属于 CMOS 晶体管模型中相关的基本参数的时候，就记录这些数据到相应的存储信息域里面。在数据处理完毕后再将存储信息域里面的信息根据我们的需要和相应的公式计算获得小信号模型中实际的参数值。并用这些参数值替换原先网表中的 CMOS 晶体管的网

表，从而完成非线性元件的线性化。

在实际操作中，为了进一步降低无用符号或者低权重符号对最后生成的二叉决定图的影响，我们会去掉一些关系不大的线性元件。

为了保证最后生成的二叉决定图依然含有每个 CMOS 晶体管的尺寸信息，我们在网表改写的过程中会将原来的 CMOS 晶体管相关的网表语句以注释的形式保留下来。在网表解析的过程中这些 CMOS 晶体管相关的尺寸信息会读入并保存在链表化的 CMOS 晶体管节点中。

同时脚本为了能够自动化的设定传输函数的输入输出接口，会自动在网表改写的过程中以注释的形式插入 GRASS 能够识别的接口定义的语句。输入输出端口的设定根据用户网表中的节点命名自动选取。

小信号模型参数的提取所采用的仿真器是 HSpice，这是为了标准化和正确性的考虑。在不远的将来，HSpice 会被替换为我们自己开发的 XSpice 以完成无缝的结合。

另外，小信号模型参数提取出来的值都是晶体管相关的，也就是说每个参数对不同的晶体管会有不同的值。但是，根据小信号模型，有的参数在相同的工艺条件下他的值应该是一样的，比如单位面积氧化层电容C_{ox}、工艺传导参数k’等。为了更好的符合模型，也为了使得仿真结果更好的接近真实的情况，在网表解析的过程中，程序会将读入这些参数做平均化求值。在实际应用这些参数的时候，采用的是他们的平均值，而不是小信号参数提取的时候的实际值。

3.4 CMOS晶体管尺寸敏感度的符号化求解

CMOS 晶体管尺寸敏感度的符号化求解是本文的核心和关键。本文在算法上相关的研究是基于论文[34]的研究基础上的拓展。对于非线性元件的敏感度求解问题，需要将其进行转化，然后再采用线性化的敏感度符号化求解方法求解。

3.4.1 CMOS晶体管尺寸敏感度求解的原理

CMOS 晶体管尺寸敏感度的符号化求解采用的正是非线性元件线性化得方法进行的。根据本文前述章节提出的小信号模型和非线性元件的线性化问题，我们可以将问题进行转化。

我们采用图3-1b的小信号模型作为我们的CMOS晶体管尺寸敏感度问题的线性化模型。因为敏感度问题的乘积累加性，也就是：

$$Sens(f(x,y),z) = Sens(f,x)Sens(x,z)+Sens(f,y)Sens(y,z) \quad (3\text{-}1)$$

我们可以将传输函数关于非线性元件CMOS晶体管尺寸的敏感度问题看成是传输函数关于晶体管模型中一组线性元件的敏感度问题。只要能够找到模型中的这组线性元件各自关于CMOS晶体管尺寸的敏感度大小，我们就能够像求连续偏微分一样利用乘积累加性求解连续敏感度。也就是说我们可以利用公式(3-2)进行CMOS晶体管尺寸敏感度的符号化求解。在这里我们以晶体管的宽度为例，用Wk表示第k个晶体管的宽度，用$p_k^{(i)}$表示第k个晶体管的模型中的第i个线性元件（假设该模型中共有m个元件）。

$$Sens(H(s),W_K)=\sum_{i=1}^{m}Sens(H(s),p_k^{(i)})\cdot Sens(p_k^{(i)},W_k) \tag{3-2}$$

对于传输函数关于晶体管模型中的每个线性元件的符号化敏感度的求解可以采用论文[34]中提到的方法实现。

为了获得晶体管模型中的线性元件关于晶体管宽度的敏感度，我们需要通过晶体管的行为模型建立二者之间的联系。我们知道，CMOS 晶体管的常见工作区主要有三个：截止区、线性区（也叫电阻区）和饱和区。由于大部分模拟集成电路设计中的 CMOS 晶体管都工作在饱和区，同时也为了简化问题分析，我们做了这样的假设，即所有我们讨论的 CMOS 晶体管的工作区是饱和区。也就是说，网表文件在读入的时候管子尺寸的初始值已经大致让管子进入到了饱和区。关于其他工作区的讨论请见本文后面的章节。有了饱和区的假设，我们就可以方便的利用管子模型中的线性元件在饱和区的相关公式进行求解了。一般的，我们知道对于图 3-1b 的模型，我们有饱和区的公式：

$$g_m=\sqrt{2k'\frac{W}{L}I_D} \tag{3-3a}$$

$$r_{ds}=\frac{1}{\lambda I_D} \tag{3-3b}$$

$$g_{mb}=\frac{\gamma g_m}{2\sqrt{2\phi+V_{SB}}} \tag{3-3c}$$

$$C_{SB}=C_{DB}=C_jL_SW+C_{jsw}(2L_S+W) \tag{3-3d}$$

$$C_{GS}=\frac{2}{3}C_{ox}WL+C_oW \tag{3-3e}$$

$$C_{GD}=C_oW \tag{3-3f}$$

其中k’是工艺传导参数，λ是沟道长度调制参数，γ是体效应参数，Φ强反型层电压，即费米能级的两倍，C_j是底部单位面积结电容，C_{jsw}是边墙单位面积结电容，L_S是边墙源漏长度，C_{ox}是单位面积栅氧化层电容，C_o是单位宽度重叠电容

[40]。

通过参数提取我们就可以方便的获得以上公式中我们需要的全部参数的值。

但是根据以上公式计算敏感度的值时，有一个问题，那就是对于CMOS晶体管模型中的某些线性元件，比如g_m，在饱和区仍有很多的公式组合，每个公式都是模拟电路设计中常用的，我们应该如何选择呢。由于在以CMOS晶体管为主的模拟集成电路设计中，晶体管经常被做电流偏置和电压偏置两种类型的设置。比如在各种电流镜中，用于镜像的晶体管一般都是被设计为电流偏置的方式；又比如，很多自偏置的晶体管就是为了固定电路分支的偏置电流，起到理想电流源的作用。考虑到实际的设计中电流偏置的电路远远大于电压偏置的电路，而电压偏置电路中也广泛使用了电流偏置的方式。因此为了求解敏感度的方法和公式的一致性，我们在这里做了一个假设，我们对公式(3-3)的敏感度求解基于电流偏置的方式，也就是说公式中的直流电流被认为是常量，而直流电压则被认为是变量。关于电流偏置和电压偏置的讨论，会在下面的章节中进一步给出。

于是根据公式(3-3)，我们就能直接获得模型中线性元件关于晶体管宽度的敏感度值，即$Sens(p_i,W_k)$。比如$Sens(g_m, W)=0.5$ 等等。

有了以上的公式和方法，我们就能够计算获得传输函数关于电路中任意一个CMOS 晶体管尺寸的敏感度符号化表示了。

3.4.2 CMOS晶体管尺寸敏感度求解的算法

根据上一节的原理，我们可以直接实现 CMOS 晶体管尺寸敏感度的求解。但是为了提高求解效率和求解的内存开销，我们采用了以下的优化方案。

考虑到采用二叉决定图表示的电路传输函数无论是分子还是分母都是所有最小生成项的或组合（SOP，Sum of Product），而每个最小生成项内的符号都与电路中的线性元件一一对应；另外，当最小生成项被用以求导的时候，每个符号都被当做布尔符号对待。因此，如本文前面章节所述，对参数 p 的求导操作就是移除不含 p 的最小生成项，和将含 p 的最小生成项中的 p 设为 1。根据敏感度的定义，我们要给做过求导的传输函数乘上 p，也就说，原先含有 p 的最小生成项在被将 p 设为 1 以后，又乘回了 p，于是对传输函数求敏感度的操作就相当于对分子和分母的多项式分别进行移除不含 p 的最小生成项操作，然后再除以原来该分子和分母的多项式的值。比如对于 N(s)=abp+bcp+def，我们有：

$$Sens(N(s), p) = (ab + bc) \cdot \frac{p}{N(s)} = \frac{abp + bcp}{N(s)} \tag{3-4}$$

因此，我们采用$N_p(s)$和$D_p(s)$分别表示移除了不含有p的最小生成项的分子多项式和分母多项式，我们有：

$$Sens(N(s),p)=\frac{p}{N(s)}\cdot\frac{\partial N(s)}{\partial p}=\frac{N_p(s)}{N(s)} \tag{3-5a}$$

$$Sens(D(s),p)=\frac{p}{D(s)}\cdot\frac{\partial D(s)}{\partial p}=\frac{D_p(s)}{D(s)} \tag{3-5b}$$

考虑到公式(2-3d)我们有：

$$Sens(H(s),p)=Sens(N(s),p)-Sens(D(s),p)=\frac{N_p(s)}{N(s)}-\frac{D_p(s)}{D(s)} \tag{3-6}$$

因此我们在求解符号化敏感度的过程中，只需要将二叉决定图中的 p 符号对应的节点的 0 子节点直接指向 0 节点即可。而为了不破坏原有的二叉决定图的数据结构，我们只需要在遍历二叉决定图进行求值的过程中，将 p 符号对应的节点的 0 子节点用 0 节点指针替代进行递归迭代或者堆栈处理即可。这样既没有损失原有的数据结构和图，又没有损失遍历求值的效率。事实上，因为省去了对原有二叉决定图的结构调整，整个操作的代价将会得到减小，而使得求解的性能进一步提升。这样做的另一个好处是，相对与论文[34]中的算法，不需要再讨论 p 本身是 R 还是 L 还是 C 而进行额外的判断和特殊处理。

于是我们得到了如下的算法：

Step1. 构建电路对应的二叉决定图。

Step2. 对每个用户选择关心的CMOS晶体管进行传输函数敏感度的符号化求解。

Step3. 对每个选择的CMOS晶体管k，根据其所在的工作区，根据公式(3-3)依次求得晶体管模型中第i个线性元件关于晶体管宽度的敏感度值，也就是$Sens(p_k^{(i)},W_k)$。

Step4. 对每个离散的频率值s，采用公式(3-6)计算$Sens(H(s),p_k^{(i)})$，回到Step3继续计算第i+1 个线性元件关于晶体管宽度的敏感度值。

Step5. 对每个离散的频率值s_n，采用公式(3-2)计算$Sens(H(s),W_k)$，回到Step2继续计算第k+1 个CMOS晶体管的敏感度。

Step6. 回到Step2，继续计算s_{n+1}下的所有用户关心的CMOS晶体管的敏感度值。

Step7. 结束所有频率所有用户关心的 CMOS 晶体管的敏感度的求解。

通过该算法获得的频域不同频率下的电路传输函数关于每个CMOS晶体管宽度的敏感度都可以直接通过波形和图像的形式呈现给设计者。

3.4.3 CMOS晶体管尺寸敏感度求解的实现

对于 CMOS 晶体管尺寸敏感度求解的实现，基本采用上一节描述的算法。

数据结构方面，为了保存 CMOS 晶体管信息，采用专门的 CMOS 晶体管类存储每个晶体管节点，节点之间采用链表的数据结构进行链接。每个节点除了存储晶体管的名称、连接的电路节点名称、晶体管类型、晶体管的宽度、长度、宽长比外，还存储了该晶体管的直流工作点数据，包括工作区、源漏电压、栅源电压、速度饱和电压、阈值开启电压、背栅电压、偏置电流等；同时存储了一个指向晶体管小信号模型的指针。考虑到实际电路中设计者可以改变晶体管的具体小信号模型，这个指针类型是一个模型类集合的基类。同时，考虑到实际模拟集成电路中大量的晶体管存在成对或者成组设计的晶体管（在镜像电流源中大量存在），因此，节点中还保存着指向下一个同对或者同组的晶体管的节点指针。这个指针引出的是一个环状链表，链表的起点和终点均是该节点。图 3-2 给出了该数据结构的一个示意。

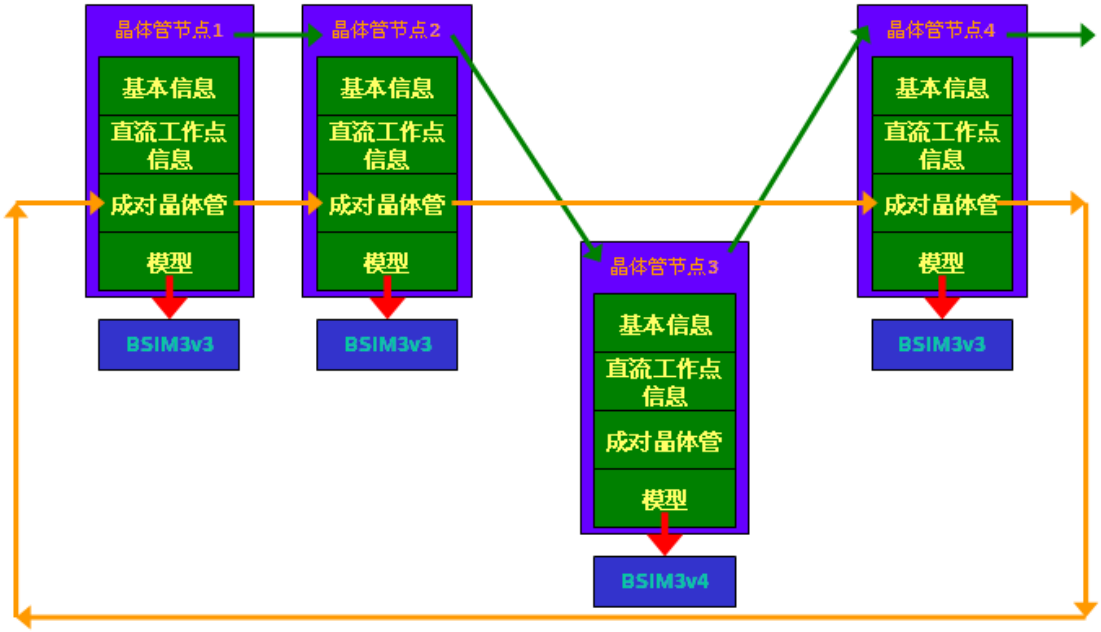


图 3-2 CMOS 晶体管节点示意
Fig.3-2 Node of CMOS transistor

而小信号模型类的每个节点则以静态变量的形式存储了这个模型中所有线性元件的基本工艺参数，比如工艺传导参数和沟道长度调制参数等。这些基本工艺参数之所以以静态变量的形式存储，是为了能够被所有同模型的晶体管共享。正如上文中提到的方法，他们会在进行晶体管解析（也就是晶体管链表构建）的过

程中被以求平均的方式获得，并在做模型参数更新时同步更新所有的晶体管的相关线性元件的值。除此之外，模型还存有一些由非公有工艺参数控制的模型中的线性元件的值，比如r_d、r_s之类的。模型类对象的主要方法就是求解共享的工艺参数的平均值和根据变动的晶体管尺寸值（比如宽度、长度和宽长比）进行模型中线性元件参数值的更新。不同的模型类有不同的模型线性元件组合。

在通过以 PCCTS 解析器完成的词法和句法分析之后，也就是 CMOS 晶体管的链表构建完成的时候，我们要将晶体管小信号模型中线性元件同其他线性元件一起拿来按照本文前述的算法构建一棵对应的二叉决定图。在构建好的二叉决定图的类对象里面，我们保存了相关的 CMOS 晶体管链表信息的头指针，用以完成对 CMOS 晶体管的选择和敏感度求解。

利用这个头指针，我们可以构建一个图形化的选择界面（详细描述见下一章节），然后交由设计者选取关心的 CMOS 晶体管进行相关的尺寸变化操作和敏感度求解操作。在完成选择后（默认是遍历整个晶体管链表的每个晶体管），对选择的晶体管的相关操作就只需要遍历一次晶体管链表，并对含有被选择标记的晶体管进行操作即可。

对于CMOS晶体管的尺寸变化引起的电路传输函数的变化并最终导致电路频域响应的变化，可以采用将晶体管的尺寸变化通过公式(3-3)更新到对应小信号模型中的线性元件中去，然后再根据这些线性元件的值对对应二叉决定图进行一次求值操作。最后将新求得的值更新到图形化的用户界面中去。图 3-3 给出了这个过程。

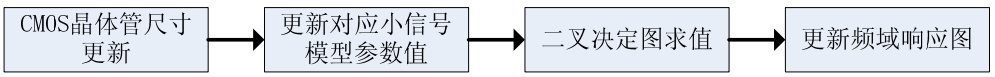


图 3-3 CMOS 晶体管尺寸更新导致频域响应更新流程图
Fig.3-3 Flow chart of updating frequency response caused by transistor sizing

而对于 CMOS 晶体管的尺寸变化的敏感度的求解，则采用前述的算法，依次求解每个被选中的晶体管（当然，实际中也可以选中感兴趣的电容或者电感等线性元件）的敏感度。求解所使用的频率点和传输函数的频率点相同。

3.5 模拟电路传输函数关于CMOS晶体管器件尺寸敏感度的叠加表示

本文下述章节将会给出通过本文上述章节的算法和数据结构获得的模拟电路传输函数关于CMOS晶体管器件尺寸敏感度在频域的表示的实际波形曲线和分析结果。本节将先从理论上对传输函数的的这个敏感度进行分析，从而对后面的章

节的内容作出数学上的证明和理论上的验证。

我们知道一个电路的传输函数除了可以采用诸如公式(1-1)的分子分母多项式的形式进行表示外，也可以表示成电路各个极点和零点的连乘积形式。即：

$$H(x)=\frac{K_2\prod_j(1+\frac{x}{z_j})}{K_1\prod_i(1+\frac{x}{p_i})} \tag{3-7}$$

其中，z_j表示各零点，而p_i则表示各极点。

这里为了方便起见，我们将公式(3-7)中的 x 替换为 s，以连续域传输函数为例子进行讨论。同时也为了形式化的统一和便于书写，我们将公式(3-7)转化为：

$$H(s)=\frac{N(s)}{D(s)}=\frac{K_2\prod_j(s+z_j)}{K_1\prod_i(s+p_i)} \tag{3-8}$$

于是按照公式(2-2)给出的敏感度定义，我们有如下的推导过程：

$$\begin{aligned}&Sens(N(s),W_k)=\sum_j\frac{W_k}{N(s)}\frac{\partial N}{\partial z_j}\frac{\partial z_j}{\partial W_k}\\&=\sum_j\frac{W_k}{K_2(s+z_j)\prod_{l\neq j}(s+z_l)}K_2\prod_{l\neq j}(s+z_l)\frac{\partial z_j}{\partial W_k}=\sum_j\frac{W_k}{(s+z_j)}\frac{\partial z_j}{\partial W_k}\end{aligned} \tag{3-9a}$$

$$\begin{aligned}&Sens(D(s),W_k)=\sum_i\frac{W_k}{D(s)}\frac{\partial D}{\partial p_i}\frac{\partial p_i}{\partial W_k}\\&=\sum_i\frac{W_k}{K_1(s+p_i)\prod_{l\neq i}(s+p_l)}K_1\prod_{l\neq i}(s+p_l)\frac{\partial p_i}{\partial W_k}=\sum_i\frac{W_k}{(s+p_i)}\frac{\partial p_i}{\partial W_k}\end{aligned} \tag{3-9b}$$

结合公式(2-3d)，我们可以立即得到电路传输函数关于 CMOS 晶体管的尺寸的敏感度的零极点表示，即：

$$\begin{aligned}&Sens(H(s),W_k)=Sens(N(s),W_k)-Sens(D(s),W_k)\\&=\sum_j\frac{W_k}{(s+z_j)}\frac{\partial z_j}{\partial W_k}-\sum_i\frac{W_k}{(s+p_i)}\frac{\partial p_i}{\partial W_k}\end{aligned} \tag{3-10}$$

为了能够清晰地展示这个结果的特征和实际的物理意义，我们将公式(3-10)分别进行取实部和取虚部的操作，并用 jω 取代 s，将实际的频率值代入公式获得实际的表示，于是我们得到了如下的结果：

$$
\begin{aligned}
&\mathrm{Re}\{Sens(H(s),W_k)\} \\
&=\sum_j \frac{W_k}{z_j(1+(\frac{\omega}{z_j})^2)}\frac{\partial z_j}{\partial W_k}-\sum_i \frac{W_k}{p_i(1+(\frac{\omega}{p_i})^2)}\frac{\partial p_i}{\partial W_k} \\
&=\sum_j \frac{W_k}{z_j}\frac{\partial z_j}{\partial W_k}\frac{1}{(1+(\frac{\omega}{z_j})^2)}-\sum_i \frac{W_k}{p_i}\frac{\partial p_i}{\partial W_k}\frac{1}{(1+(\frac{\omega}{p_i})^2)} \qquad \text{(3-11a)} \\
&=\sum_j Sens(z_j,W_k)\frac{1}{(1+(\frac{\omega}{z_j})^2)}-\sum_i Sens(p_i,W_k)\frac{1}{(1+(\frac{\omega}{p_i})^2)} \\
&=\sum_j Sens(z_j,W_k)\varphi(\frac{\omega}{z_j})-\sum_i Sens(p_i,W_k)\varphi(\frac{\omega}{p_i}) \\
&\mathrm{Im}\{Sens(H(s),W_k)\} \\
&=-\sum_j \frac{W_k}{\omega(1+(\frac{z_j}{\omega})^2)}\frac{\partial z_j}{\partial W_k}+\sum_i \frac{W_k}{\omega(1+(\frac{p_i}{\omega})^2)}\frac{\partial p_i}{\partial W_k} \\
&=-\sum_j \frac{W_k}{z_j}\frac{\partial z_j}{\partial W_k}\frac{\frac{z_j}{\omega}}{(1+(\frac{z_j}{\omega})^2)}+\sum_i \frac{W_k}{p_i}\frac{\partial p_i}{\partial W_k}\frac{\frac{p_i}{\omega}}{(1+(\frac{p_i}{\omega})^2)} \\
&=-\sum_j Sens(z_j,W_k)\frac{\frac{\omega}{z_j}}{(1+(\frac{\omega}{z_j})^2)}+\sum_i Sens(p_i,W_k)\frac{\frac{\omega}{p_i}}{(1+(\frac{\omega}{p_i})^2)} \\
&=-\sum_j Sens(z_j,W_k)\psi(\frac{\omega}{z_j})+\sum_i Sens(p_i,W_k)\psi(\frac{\omega}{p_i}) \qquad \text{(3-11b)}
\end{aligned}
$$

其中 $\varphi(x)=\frac{1}{1+x^2}$， $\psi(x)=\frac{x}{1+x^2}$。

从公式(3-11)的结果我们发现，其实传输函数关于CMOS晶体管尺寸符号化表示的敏感度都是一组以频域频率变量和各个零极点值比值为参数的函数，在各个零极点关于CMOS晶体管尺寸敏感度为权重的表示下的累加结果。关于电路零极点关于CMOS晶体管尺寸敏感度的求解将在下一节中具体表述，现在我们先将这个敏感度当成一个具体的常数来理解。现在我们先来讨论公式 3-11 中的 $\Phi(z_j/\omega)$ 和 $\Phi(p_i/\omega)$以及 $\Psi(z_j/\omega)$和 $\Psi(p_i/\omega)$函数的具体形式。图 3-4 给出了这两类函数(Φ ()和 Ψ())的示意图。

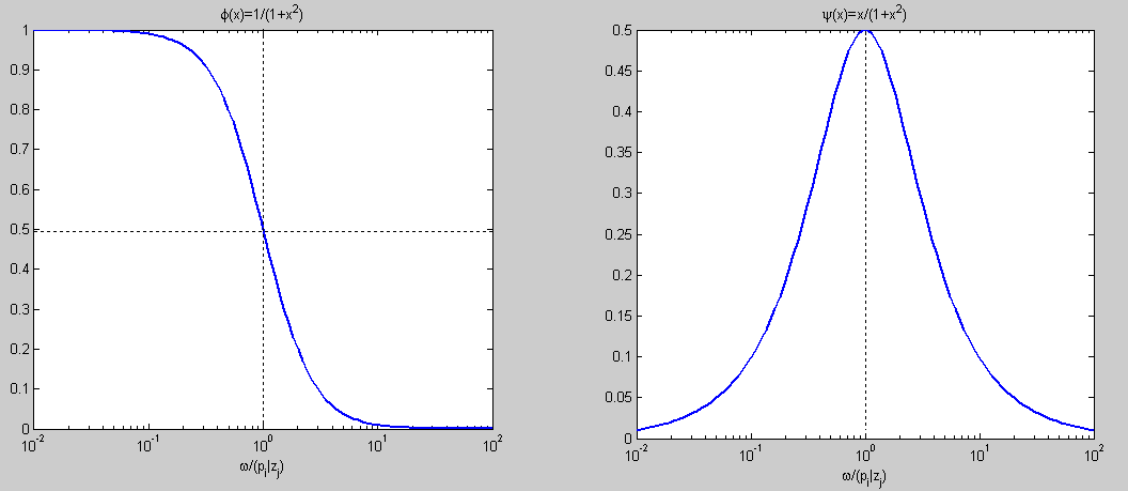


图 3-4 两类函数(ϕ()和ψ())示意图（左-ϕ(); 右-ψ()）

Fig.3-4 Function graphs of φ() (Left) and ψ() (Right)

从图中我们可以清楚的发现，当两个函数的变量 x 等于 1 时，也就是频率ω和零极点的值大小一样时，函数ϕ(x)的值是其最大值的中间值，而函数ψ(x)的值则是其的最大值。这个现象其实可以直接从函数的表达式中推导获得，即：

$$\phi(1) = (\frac{1}{1+x^2})|_{x=1} = \frac{1}{2} = \frac{1}{2}\cdot 1 = \frac{1}{2}(\frac{1}{1+x^2})|_{x\to 0} \tag{3-12a}$$

$$\psi(1) = (\frac{x}{1+x^2})|_{x=1} = \frac{1}{2} = (\frac{x}{1+x^2})_{\max} \tag{3-12b}$$

图中用虚线加以标示和说明。

由于电路的零极点都是独立的离散点，也就是说，最后我们得到的传输函数关于 CMOS 晶体管的尺寸的敏感度其实就是这些离散点各自以图 3-4 的波形为基础乘上各自关于 CMOS 晶体管尺寸的敏感度的值后叠加的结果。而且假设各个零极点互相之间的距离离得比较远的话，我们将会看到传输函数实部和虚部敏感度的结果会在各个零极点的位置处发生类似ϕ(x)和ψ(x)的变化趋势，也即，对于实部，我们会看到波形呈现由上至下（或者由下至上，考虑到零极点关于晶体管尺寸的敏感度的值可正可负）的变化趋势，而变化趋势的中点恰好是电路的零点或者极点；而对于虚部，我们将会看到波形呈现一个波峰（或者一个波谷，考虑到零极点关于晶体管尺寸的敏感度的值可正可负）的变化趋势，而变化趋势的顶点恰好是电路的零点或者极点。由于一般的模拟集成电路设计中，主极点是相对于其他极点和零点最接近与 0 的第一个零极点，而且他同其他的零极点的距离一般隔得相对较远，因此上面发现的这个规律可以很好的拿来用以直接找主极点的位置。而这个位置就是传输函数关于 CMOS 晶体管尺寸敏感度在频域的实部的第一

个 1/2 点或者虚部的第一个波峰或者波谷。

由于存在公式：

$$Sens(|H(s)|, p) = \mathrm{Re}\{Sens(H(s), p\} \tag{3-13}$$

$$Sens(\angle H(s), p) = \frac{1}{\angle H(s)} \mathrm{Im}\{Sens(H(s), p\} \tag{3-14}$$

公式的具体证明请参考[34]。

传输函数关于CMOS晶体管尺寸的频域敏感度的幅频和相频曲线同他的实部和虚部之间建立了一种直接的联系，因此我们只要将采用符号化仿真器计算获得的频域敏感度的值分别表示成幅频和相频的结果，我们就可以运用上面的分析和理论推导的结果直截了当的获得我们需要的零极点的可能分布情况，同时这也符合一般模拟电路设计指标和数据显示的习惯，更够给设计者提供更多直观有效的信息和帮助。

3.6 电路主极点关于CMOS晶体管器件尺寸的敏感度

电路主极点是模拟集成电路设计中最重要的一个参数指标，他表征了电路的诸多特性。一旦能够建立电路主极点同电路晶体管尺寸参数相关联的内在联系，并且这种联系能够被简单明了的呈现，设计者在设计中的选择将会变得非常明晰，电路的行为同管子的参数的关系也会变得很直接。关键的晶体管将会被迅速的找到，这将给设计者对电路一个非常直观的帮助和理解。本文采用对电路主极点关于CMOS晶体管器件尺寸敏感度的说明来展示电路特征指标参数关于CMOS晶体管器件尺寸敏感度的一般处理方法和所提供的巨大价值，并由此推而广之到各种其他电路特征指标。一旦电路的各个设计指标都能够被以这样的方法加以解决，那么电路的自动化设计方法学将会天然的完成。

本文前述章节已经介绍了符号化的仿真器可以通过电路的传输函数构建公式近似求解电路的主极点的值（公式(2-5)和公式(2-6)）。而电路的传输函数本身可以符号化的从电路中通过构建二叉决定图得到。因此，基于二叉决定图天然的求解敏感度的优势，我们可以方便的从二叉决定图中直接获取电路主极点的敏感度公式。我们从主极点求解公式(2-6)和敏感度定义公式(2-2)入手，我们有：

$$
\begin{aligned}
&Sens(P_d, W_k) = \sum_i Sens(P_d, p_i) \cdot Sens(p_i, W_k) \\
&= \sum_i (Sens(P_d, a_0) \cdot Sens(a_0, p_i) + Sens(P_d, a_1) \cdot Sens(a_1, p_i) \\
&+ Sens(P_d, b_0) \cdot Sens(b_0, p_i) + Sens(P_d, b_1) \cdot Sens(b_1, p_i)) \cdot Sens(p_i, W_k) \\
&= \sum_i (\frac{-a_1 b_0^2 a_0}{[a_0 b_1 - a_1 b_0]^2 P_d}) \cdot Sens(a_0, p_i) + (\frac{a_1 b_0^2 a_0}{[a_0 b_1 - a_1 b_0]^2 P_d}) \cdot Sens(a_1, p_i) \\
&+ (\frac{b_1 a_0^2 b_0}{[a_0 b_1 - a_1 b_0]^2 P_d}) \cdot Sens(b_0, p_i) + (\frac{-b_1 a_0^2 b_0}{[a_0 b_1 - a_1 b_0]^2 P_d}) \cdot Sens(b_1, p_i)) \cdot Sens(p_i, W_k)
\end{aligned}
\tag{3-15}
$$

其中P_d表示主极点，公式中的a_0，b_0，a_1，b_1关于p_i的敏感度和p_i关于W_k的敏感度的值可以分别根据公式(2-6)和公式(3-3)推导求得。

有了以上公式，我们就可以方便的从电路传递函数对应的二叉决定图中获得相应的数据，然后代入上面的公式进行计算即可获得主极点关于晶体管尺寸参数的敏感度的值，从而找到影响电路主极点的关键晶体管和各个晶体管对电路主极点这一指标参数的影响程度和影响方向。而符号化仿真器对于获得传输函数中分子分母的多项式的各阶矩具有很方便的能力和处理。在符号化的二叉决定图中，为了存储的符号能够做到和电路的传输函数中的最小生成项完全的一致对应，我们针对不同的线性元件采用了不同的符号策略。比如电阻用 G（也就是 1/R），电容用 Cs，电感用 1/(Ls)等等。因此我们只需要在对二叉决定图进行求值的时候，记录二叉决定图每条路径中电容元件和电感元件的个数，那么我们就可以得到如下的关系式：

$$
s' \quad power = (\#Caps - \#Inds) / Path \tag{3-16}
$$

这样，我们分子和分母的多项式的各阶矩就可以通过以上公式选取对应的二叉决定图的路径值并作累加直接获得。于是上面公式中的系数也就得以求解了。

3.7 本章小结

本章主要介绍了使用符号化的方法进行电路传输函数和相关电路指标参数关于 CMOS 晶体管敏感度的求解方法和理论数据的分析，并对相关的算法、模型和求解过程都做了详细的描述，给出了具体的公式推导过程和理论验证，并对部分实现细节做出了详细描述。

本章给出的理论证明和推导过程以及算法推演都将在本文下述的内容中通过案例实际的得到验证和说明。本章和下一章是本文的核心，描述了本文工作的主要理论依据。

第四章 符号化模拟电路设计平台

符号化模拟电路设计平台是符号化模拟集成电路设计方法学的完整实现和封装。电路设计人员通过平台完成模拟集成电路设计的所有相关操作。平台按照设计方法学的流程，提供给用户友好的人性化接口，引导用户一步一步的完成整个设计的全过程。目前设计平台并没有完成全部的自动化，但是设计平台的半自动化已经可以给用户提供相当强有力的设计支持和设计指导了。

4.1 符号化模拟电路设计平台框架

符号化模拟集成电路设计平台是参照图 1-2 的模拟集成电路设计自动化流程，结合本文描述的算法和符号化仿真器引擎的优缺点，有的放矢的设计和进行架构的。图 4-1 给出了我们称之为 SPADE（Symbolic Platform for Analog Design Exploration）的符号化模拟电路设计解探索平台的架构框图。

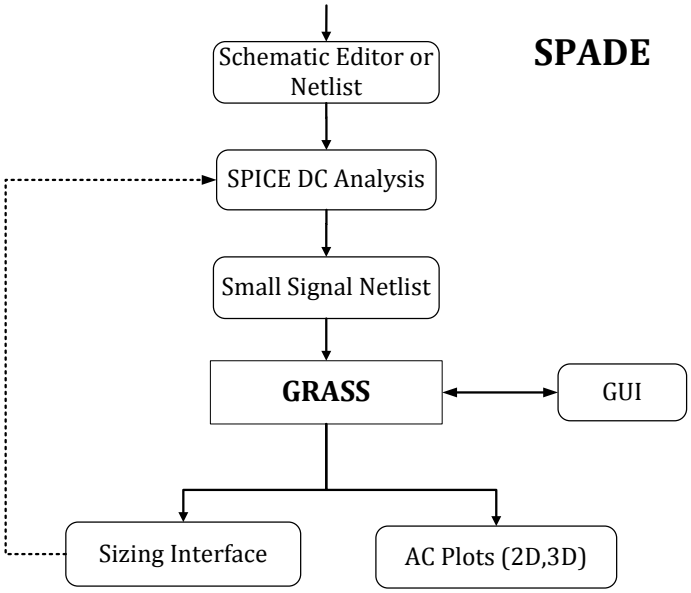


图 4-1 符号化模拟设计解探索平台 SPADE 架构框图
Fig.4-1 Framework of SPADE

我们可以从图 4-1 中看到，整个平台提供了丰富的图形化输入输出接口用以和设计者进行交互。平台由一个拓扑结构编辑器、一个网表编辑器、一个仿真参数配置接口、一个直流工作点求解器（目前采用 HSpice，接下来的工作会将其替换为我们自己编写的 XSpice）、一个小信号模型参数提取器、一个符号化仿真求

解引擎 GRASS (Graph Reduction Analog Symbolic Simulator)、一个提供给用户进行二维和三维仿真结果（二维展示的主要是电路的幅频响应和相频响应，三维显示的主要是多参数情况下，多设计指标的交叉域求解和参数波动情况下的设计指标变化范围展示）显示和动态 CMOS 晶体管尺寸参数调整设计的图形化接口、一个电路传输函数关于 CMOS 晶体管尺寸敏感度频域值的结果显示接口和主极点的符号化求解以及主极点的符号化敏感度求解的数据。图中的虚线用以描述尚未完成的自动化过程，也就是通过和设计指标的比对，自动化的根据敏感度的值调整相应晶体管尺寸以达到设计指标的过程。整个平台还有一个独立的图形化界面用以进行符号化的模拟集成电路设计常用公式表示、编辑、求值和波形展示。

整个平台是被设计用来一方面提供给有经验的模拟集成电路设计者做设计参数的参考和电路特性同晶体管参数内在关系的辅助理解；另一方面则被用来提供给模拟电路设计初学者以快速理解电路行为同电路晶体管尺寸参数关系和电路特性的有效工具。平台的设计是面向使用的，因此包含了丰富的手段用以控制意外和误操作以及计划外输入所产生的诸多问题。整个平台已经集成到我们学院本科实验室的工具环境中去，并试图在接下来的模拟集成电路设计的教学中发挥主导性的作用。

平台的工作环境是以 Linux 操作系统为主的。全部的程序代码都是由 C 和 C++ 完成，平台各个组建的粘连用 Tcl 和 Perl 脚本组装在一起。平台的图形化界面采用的是 Gtk 和 Glut 做渲染。平台将用户所需要的所有操作以按钮的形式集成在主界面，而采用命令和脚本完成大量衔接和接口工作，使得整个平台对用来说是透明的，清晰地，所有操作都会在背后自动化的完成。用户只需要按照流程依次完成按钮的点击操作就可以完成所有的设计工作了。图 4-2 给出了符号化模拟集成电路设计解探索平台的主界面。

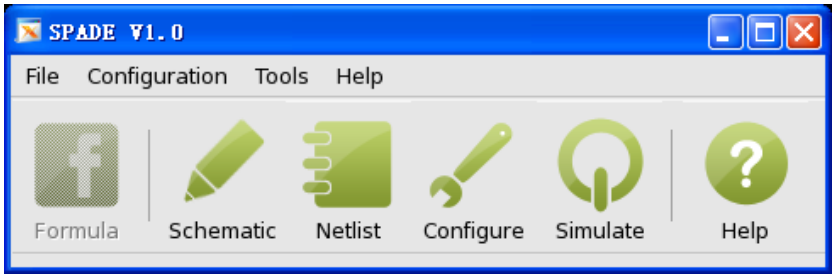


图 4-2 符号化模拟设计解探索平台 SPADE 主界面
Fig.4-2 Top interface of SPADE

按照图 4-2 中的图标顺序从左至右依次进行点击和编辑操作，整个设计就可以完成了。整个流程基本上是：先进行电路拓扑结构的设计和编辑，符号化模拟

设计解探索平台会从设计者的设计中自动的生成相应的网表，然后这个用户可以经由用户二次编译（如果需要的话）。接着，设计者点击配置按钮，进行仿真环境和一些基本参数的设定，在设定完毕并保存确认后，单击仿真按钮，程序将会完成剩下的全部操作，设计者可以根据仿真结果的二维和三维显示和辅助，半自动化的选择合适的晶体管尺寸的值。当然用户也可以回到流程的开始，重新修改电路的拓扑结构或者网表，再次进行这样的仿真操作，直至达到满意的电路设计指标值。

图 4-2 显示的公式图表是灰色的，这是因为现在暂时将符号化的公式表达的图形化接口做成独立于整个平台之外，等到整个公式表达的功能达到设计最初的要求，会重新集成到整个平台中来的。

平台设计的一些尚未实现的功能还包括：对电路的层次化分析（目前已完成理论证明和原型设计及验证，但是尚未整合到平台中）、所有模拟集成电路设计常见指标的符号化支持（目前并没有全部做到）和关键器件的特殊模型仿真等等。

4.2 拓扑结构设计和网表接口

拓扑接口设计图形化接口是本文介绍的符号化模拟设计解探索平台的第一个输入接口，用以提供给设计者进行电路拓扑结构的设计和晶体管参数的修改。接口的界面部分采用 Gtk 开发，核心程序采用 C 和 C++书写。

整个界面将主要的电路设计元件（包括电阻、电容、电感、理想电源、受控电源、CMOS 晶体管、二极管等）以按钮的形式置于界面的菜单下面，用设计者方便的进行拖拉。同时界面提供专门的连线按钮，方便设计者进行连线操作。在连接好的电路中，用户可以通过直接的拖拉元件或者连线，达到调整位置的目的。设计者可以通过右键菜单完成图形界面网格的打开和关闭，元件的旋转和翻转，元件属性的设置等功能。在设计者确认电路的拓扑结构搭建完成，电路元件的参数也设置完成后，设计者可以设定一些常用的仿真参数，比如直流分析、交流分析和瞬态分析等，在这些都设定完成，用户可以采用“Netlist Generation”菜单一键完成电路对应的 HSpice 格式的网表文件生成。另外，用户还可以通过按住 ctrl 键选取若干个电路元件，将他们归为一组，这样在符号化仿真器引擎处理的时候就会将他们视为一组做层次化分析，同时也使得这一组元件对应的符号在构建电路对应的二叉决定图的时候在一起按照先后顺序构建，这样有助于构建的二叉决定图获得更好的符号顺序，也即获得更小的内存开销和节点数量。分组的操作还可以用以拓展，比如设定匹配的晶体管，设定采用同一小信号模型的 CMOS 晶体

管或者电路元件等等。另外用户可以将当前设计的电路存储下来，以备下次打开的时候直接加载就好了。图 4-3 展示了这个拓扑接口设计接口图形化的主界面。（界面中的电路正好是图 5-1 中所示的电路，具体的参数值也是与图 5-1 中一一对应的）图 4-4 展示了电路元件属性设置的窗口（可以看到元件属性的设置包括基本参数属性值、分析类型、模型和显示控制等）。

众所周知，采用 Gtk 的图形化开发很容易因为开发的图形界面涉及的元件内容比较多，元件对应的事件响应函数比较多而且比较负责，元件属性设置比较多，而且比较混乱等各种原因导致代码无限庞大而且混乱，既不利于管理，又不利于理解和扩展，很多时候连代码的作者自己也会忘掉一段代码是做什么用的。虽然 Gtk 本身提供了 glade 工具进行 xml 化的图形元件管理，但是由于还是需要通过元件的信息先找到元件，然后进行元件对应事件的回调函数绑定等手工代码操作，才能完成整个图形化的构建，对于开发比较大的图形化界面来说代码还是显得相对比较冗长和混乱。为了解决这个问题，我采用自己开发的一套 xml 格式的编码

解码机制，对图形化界面进行 xml 文件动态构建的方法，使得代码的结构非常清

晰简短，代码的内容非常的直接，xml 的改变将直接影响图形化界面的外观而不需要再次重新进行程序的编译，提高了界面设计的灵活性，将界面设计从代码中剥离出来，既提高了并行化程度，又降低了代码的耦合性，使得程序的维护变得非常容易和简单。而且 xml 的标记语言格式简单直接，易于理解，对于新元件的开发或者原有元件参数或者属性和回调函数的添加都有清晰的模板可以参考，大大降低了代码的复杂程度和关联性，这也保证了我们项目的多线进行和质量。另外，动态化的图形界面的构建方式只在程序启动的时候进行，因此不会影响程序性能（图形界面本来就会在加载的时候比较慢）。本文论述的图形化界面除了三维渲染采用了 glut 的库以外，其他的二维图形界面都是采用这种方式进行构建的。

接下来将重点论述本文所采用的基于 xml 的动态图形化构建算法细节和框架。

本算法所采用的 xml 的格式和 glade 类似，也是以元件为核心，元件的参数就是 xml 的元件属性，嵌套的标记表明图形元件层次上的父子关系，容器（比如 VBox、HBox 之类的）也显式的被和其他元件同等对待。同 glade 不同的是，事件和对应的回调函数的绑定也被放在了 xml 里面，绑定的具体工作会在引擎解析 xml 的时候实际自动化的完成。xml 中的元件是对 Gtk 的图形元件经过 C++封装的

自定义类实例。下面的一段 xml 代码是展示了菜单这样一个比较典型的例子：

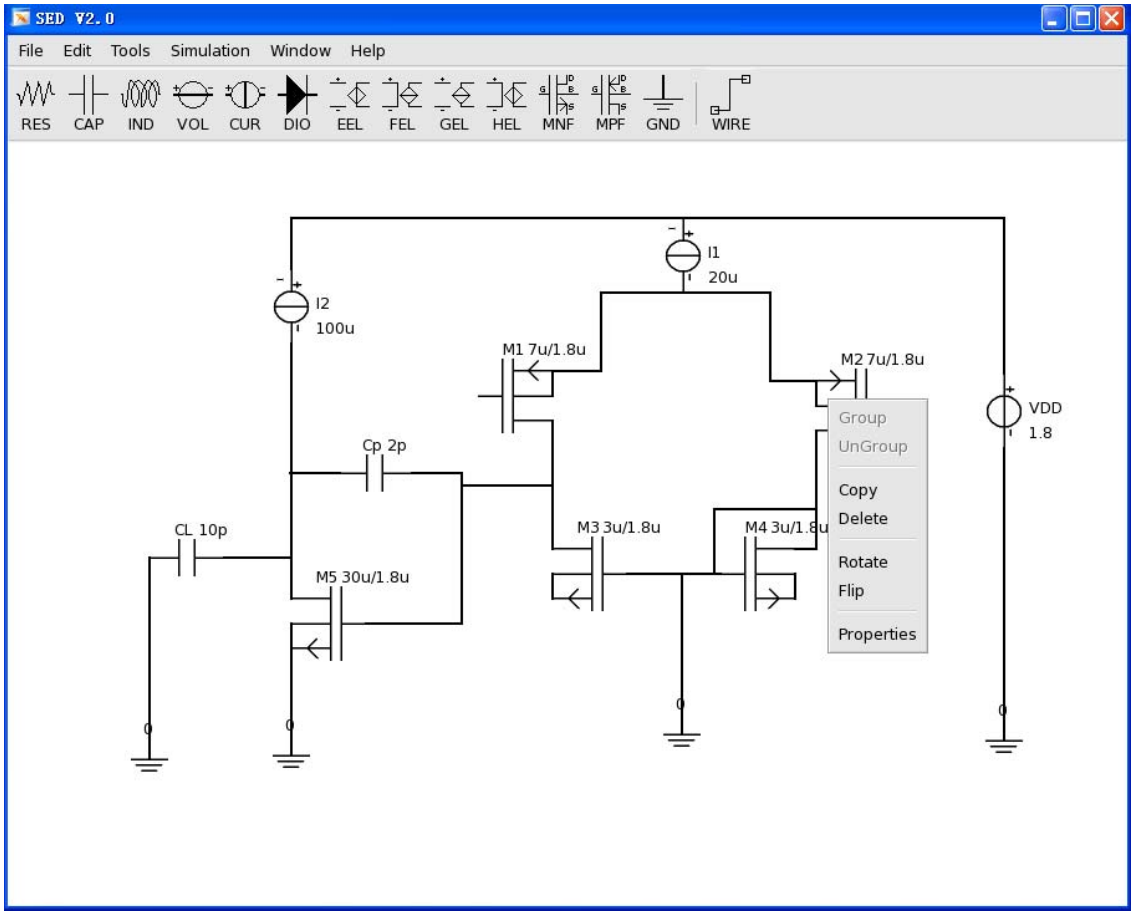


图 4-3 拓扑接口设计接口图形化的主界面

Fig.4-3 Top interface of schematic editor GUI

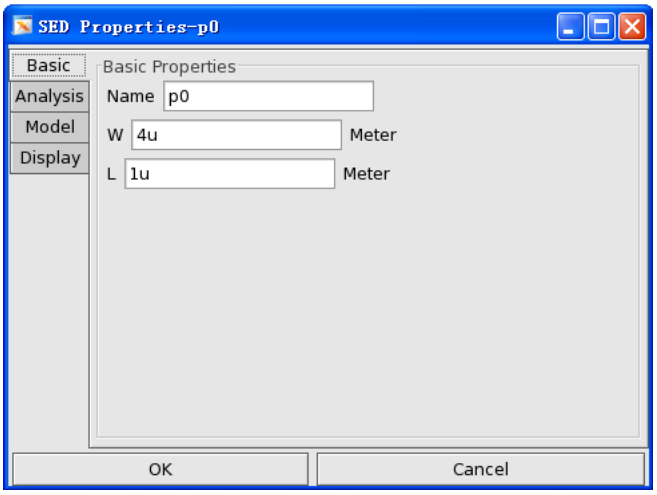


图 4-4 电路元件属性设置窗口

Fig.4-4 Properties of circuit element

```
<MenuShell>
    <Menu label = "Open...">
        <Event name = "activate" handler = "MGtkMenuOpenFile" data =
    "rpt/"/>                </Menu>
    <Menu label = "Save">
        <Event name = "activate" handler = "MGtkMenuSaveFile" data = "1.se"/>
    </Menu>
</MenuShell>
```

算法采用 libxml2 库进行 xml 的语法解析，分两个步骤，第一个步骤先对 xml 进行解析，解析的时候会根据解析的结果动态的分配空间生成每个元件对应的类实例，并根据属性值设定这个实例的对应域（所采用的函数是 processNode，这个函数又会调用每个新建的自定义类对象的解析函数 getXmlNodeProperties 进行参数值的解析和保存），同时构建对应的父子关系，将子对象都加入到父亲的容器中去，这些元件的顶层元件，也就是窗口元件对象会被存储到一个窗口对象的 map 里，用以后续步骤。第二个步骤就是从存储窗口元件对象的 map 读取窗口元件，自顶向下的依次对每个元件完成其存储的参数值到对应的 Gtk 元件的设置和事件/回调函数绑定（绑定中会采用函数 isValidEvent 判别事件以及回调函数的合法性，未定义的事件或回调函数的信息会被丢弃）的操作（操作通过调用每个自定义类对象的函数 genGtkGUI 完成）。经过这样的两遍遍历，所有窗口的所有元件都得到设置和创立，整个代码显得非常紧凑。对于一个新添加的元件来说，代码编写者只需要编写对应事件的回调函数、getXmlNodeProperties、isValidEvent 和 genGtkGUI 这几个函数就可以了。整个过程如图 4-5 所示。

关于电路拓扑结构设计接口程序的核心算法，主要采用的是将每个元件进行链表串接然后操作的基本方法。每个电路元件类的基本方法包括生成 HSpice 格式的网表文件语句（genNetlist）、生成电路拓扑结构的保存文件语句（genSEDFile）、解析电路拓扑结构文件语句（parseSEDFile）、复制（copy）和生成属性设置窗口（genPropertyWindow）等。程序的主要操作集中在菜单、按钮、画图区和连线操作的回调函数中。接下来本节将对这些回调函数中重要的几个做简单的说明。

自动生成 HSpice 格式的网表文件的功能的实现就是对当前电路的元件链表的元件逐个访问，然后调用各个对象的 genNetlist 函数生成语句输出到网表文件中。属于同一组的元件会被按照元件在链表中的顺序组合输出到一起。而如果用户设置了诸如直流分析、交流分析这样的分析的话（这些信息作为静态共享

信息存储在每个元件对象里面），那么还要在网表的最后输出符合格式的分析语句。另外根据理想电源的名字，会自动添加端口定义语句（以注释的方式，但是符号化仿真器的词法句法解析器是可以识别的）。

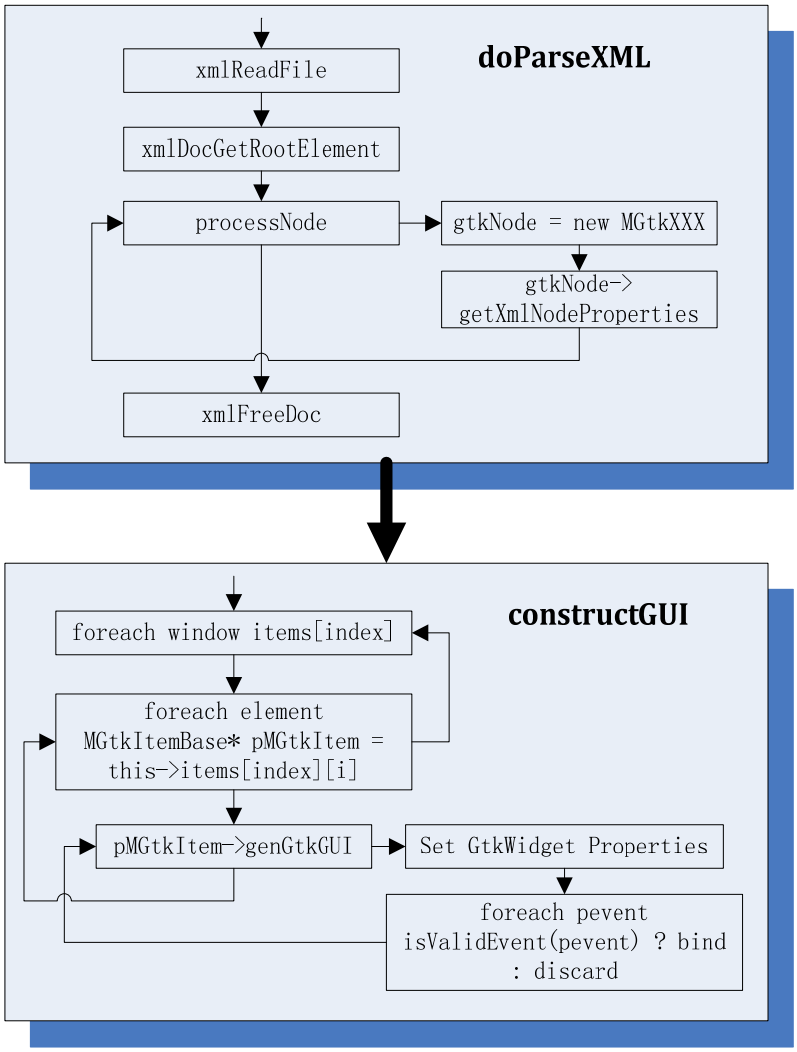


图 4-5 基于 xml 动态解析的引擎算法流程
Fig.4-5 Algorithm flow of xml based dynamic parsing engine

元件按钮被点击时会创建对应元件类的对象，并将该对象加入到全局的对象链表中去，同时设置当前按钮状态为 insensitive（用户点击无效），直到用户在画图区摆放下该元件才会重置该按钮的状态。

画图区的操作相对比较多，也是最为复杂的。对于摆放元件的左键操作，直接更新当前元件的位置和状态信息，同时更新周边关联的元件的位置信息，做到自动调整；而相应的右键操作，则需要将当前元件从元件链表中移除。否则则做

选中（更新元件当前状态）和拖动（跟随鼠标做实时的位置和状态更新，拖动后需要更新全部相关联的元件的相关信息）操作，同时更新画图区的显示内容。

连线在引擎中被视为一个特殊的元件类，他有很多特有的操作，却不属于真正的元件链表。一条线有 4 个可动节点，如图 4-6 所示，其中左右两个节点是线的端点，用以连接具体的元件；中间两个节点是线的调整端点，用以调整线的位置和形状。由于这 4 个端点的存在，使得线的区域和位置的计算以及画图区的更新都显得相对复杂。所有的操作都是基于点到点的段而非线这样一个整体单一的元件。线的自动走线算法基于的就是对初始化段所在的区域是否和现有的元件链表上的元件位置重叠。如果重叠，则自动调整中间两个点的位置直到不再重叠，调整的依据是重叠位置同被重叠元件中心点位置的相对距离和方位。当一条线被连接到另一条线或者另外若干线上的时候，删除和移动的操作就会使得线的算法复杂性直接上升。为了确保操作结果的合理性和正确性，需要将所有相关联的元件做遍历和更新，同时调整线的位置至最佳。线的对象不会加入到元件链表中，而是会被加入到与他连接的所有元件的连线域中。

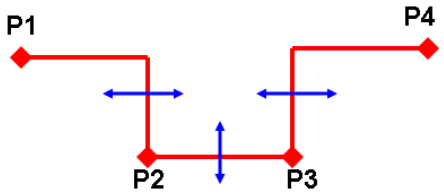


图 4-6 线的基本形状和结构
Fig.4-6 Basic shape and structure of wires

电路拓扑结构设计接口程序的算法细节太多，囿于本文篇幅的限制和本文主要工作的重心并不在于实现一个图形化的用户界面，因此本文不再细述图形操作的相关算法细节，有兴趣的读者可以直接查看代码。

考虑到电路拓扑结构设计接口程序的自动化程度的可能限制和用户需要的广泛性，符号化模拟电路设计解探索平台还提供了文本化的网表编辑接口。用户可以通过主界面的按钮调用当前平台的默认编辑器打开网表文件进行微小内容的编辑和修改。这样的途径增加了用户的选择性和处理问题的灵活程度，使得接口更加丰富。

4.3 电路仿真基本参数配置

符号化模拟设计解探索平台给用户提供了一个图形化的仿真参数配置接口，

如图 4-7 所示。用户可以通过这个接口完成所有与仿真参数相关的配置任务，在保存确认后还可以通过文本编辑器对参数内容进行文本化的灵活调整和修改。通过这个配置文件，平台会自动的设定和配置相关的参数用于仿真和最后的结果呈现（比如仿真库、网表文件、CMOS 晶体管模型、仿真界面二维和三维的选取，三维界面的多参数配置等）。整个配置接口采用的就是上一节介绍的基于 xml 的动态图形化接口算法生成的。

有这个图形化配置接口配置的信息将会经由下一节介绍的全平台操作脚本自动的转化成符号化的仿真引擎 GRASS 的配置文件，并将部分参数提取出来提供全局仿真和全流程使用。

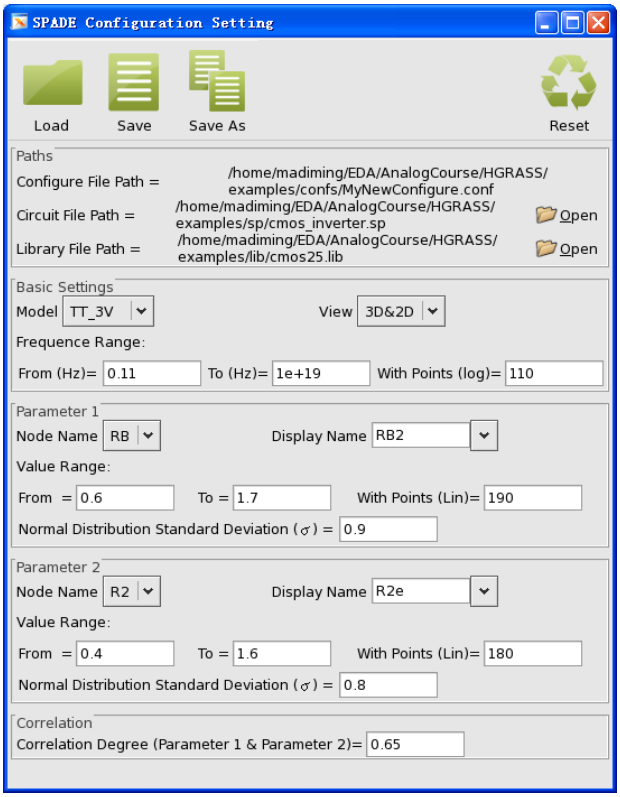


图 4-7 电路仿真参数配置接口

Fig.4-7 Interface of circuit simulation configuration

4.4 全平台操作流脚本控制

整个符号化模拟设计解探索平台包含了非常多的组件，包括电路拓扑结构编译器、图形化配置接口、直流工作点求解器、小信号模型参数提供工具、仿真引擎、图形化结构呈现和符号化公式编辑器等各种接口和工具。这些接口和工具有

的采用 C 代码编写，有的则采用 C++代码编写，有的是 Tcl 脚本，有的则是 Perl 脚本，还有的是动态库、有的是独立的二进制文件，还有的甚至需要大量的手工命令的操作完成工具使用，这么多且复杂的工具和接口如何整合到一起，成为一个对用户来说单一的、透明的平台工具，靠的就是全平台脚本的粘合和控制。所有的程序和代码都被加入了同脚本的交互部分（C 和 C++的代码采用 system 函数完成同脚本的交互，脚本之间通过 exe 命令进行互相调用，通过返回的错误值进行结果的甄选和流程的控制；而所有程序和脚本的输入和输出都同主脚本相连，输入和输出都经过主脚本的封装和过滤进而同其他脚本和程序完成衔接）。整个全平台脚本在考虑兼容性的前提下采用 Tcl 的脚本书写，图 4-8 展示了这个主脚本的工作流程。

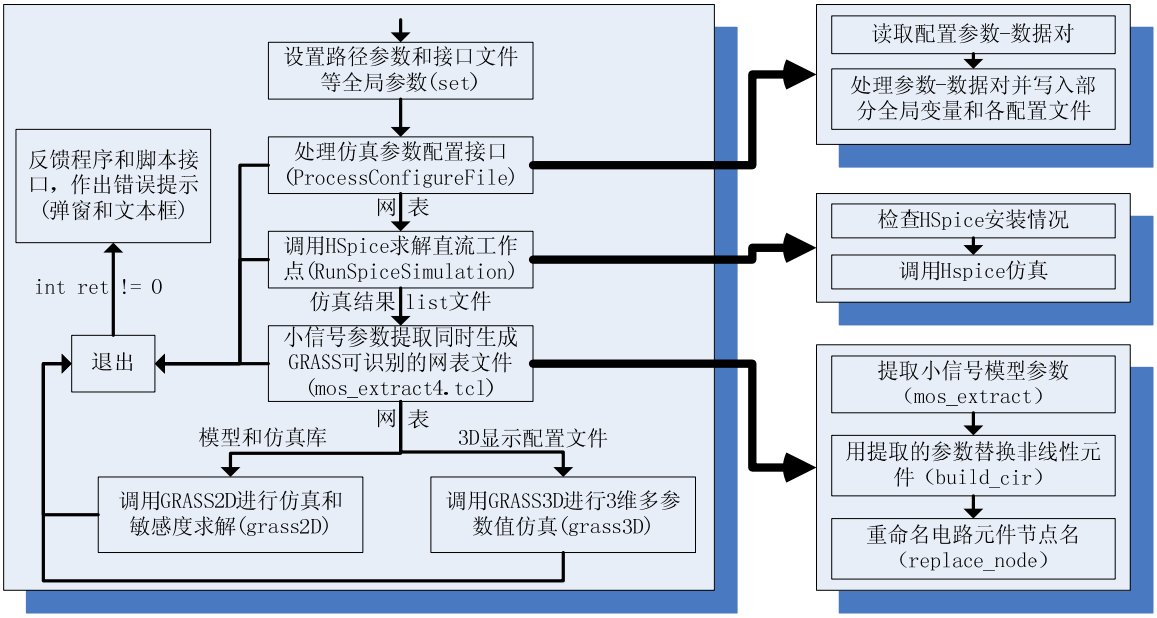


图 4-8 主脚本工作流程

Fig.4-8 Flow of main control script

从图 4-8 中可以看到流程先对仿真参数配置图形化接口的数据进行处理以获得全局的变量值和相应程序配置文件的生成。然后调用 HSpice 进行直流工作点分析，在完成直流工作点分析后调用脚本进行小信号模型的参数提取和原网表非线性元件的替换，最后将符号化仿真器可以识别的网表格式输出到结果的图形显示程序中去。如果用户在一开始仿真参数配置的时候配置的是二维和三维的结合显示，grass2D 和 grass3D 的程序都将被调用，否则，则根据用户的选择分别调用相关的程序进行结果呈现。可以注意到整个流程中，几乎所有的步骤都会检测到错误或者意外而进入到退出环节，脚本退出时将返回对应的退出值（整型）给调用

的程序（比如 system 函数）或者脚本，然后脚本或者程序根据返回的退出值进行不同的分支处理，或者以图形化的弹窗信息向用户描述问题或者以文本编译器的形式告知用户发生错误的具体内容以供用户参考和选择处理。表 4-1 列出了主要错误类型和对应的返回值。

经过整个脚本的调用和执行，仿真工作全部完成。

表 4-1 主要错误类型和对应返回值
Table 4-1 Main error types and return values

返回值	0	1	2	3	4
错误类型	正常	配置文件不存在	GRASS 不存在	HSpice 不存在	HSpice 仿真发生错误
处理方法	不处理	弹窗提示	弹窗提示	弹窗提示	文本显示错误信息
返回值	5	6	7	8	9
错误类型	未定义结果呈现类型	参数提取脚本不存在	参数提取脚本运行错误	Grass3D 仿真错误	Grass3D 错误报告存在
处理方法	弹窗提示	弹窗提示	文本显示错误信息	弹窗提示	文本显示错误信息
返回值	10	11	12	13	14
错误类型	Grass2D 仿真错误	Grass2D 错误报告存在	配置信息不充分	网表文件不存在	保留
处理方法	弹窗提示	文本显示错误信息	弹窗提示	弹窗提示	不处理

4.5 结果呈现和设计互动

在结果呈现上，符号化的模拟电路设计解探索平台提供了二维和三维的两种选择。

二维的图形化界面主要是展示符号化求解的电路传输函数的幅频响应、相频相应和主极点，同时给用户提供符号化求解的幅频、相频和主极点关于 CMOS 晶体管和线性元件的敏感度的频域分布和值的展示。同时二维的界面给设计者提供了非常直观的交互式操作流。设计者只需要拖动线性元件和非线性元件（比如 CMOS 晶体管）的参数值（比如尺寸大小），就可以即时的看到电路的传输函数的幅频响应和相频响应曲线的随动。设计者只需要根据曲线的形状和特性指标，记录当前的元件参数的值即可。这个二维界面给设计者提供了一个交互的、由设计者主导的半自动设计方法，能够大大加快设计的收敛速度。而同时提供的敏感度的频域分析有很清晰的提供了设计者设计的方向和对关键元件的理解和判断，使得设计变得更加明确和直观。

三维的图形化界面则提供了用户另一种设计指标和设计方法学的展示。界面展示了单位增益带宽和相位裕度以及他们的敏感度关于两个电路元件参数的三维立体曲线图。设计者可以通过截面同三维曲面相切的方式，找到一组符合单位增益带宽和相位裕度设计要求的两个电路元件参数的值的集合（以曲面上的曲线表示），当设计者对这某两个设计要求的元件参数值曲线做交集操作时，设计者可以直接从相应的二维区域中看到同时满足两个设计指标要求的合适的两个元件参数值的集合。同时三维的图形化界面还提供了参数波动的情况下的分析。当一个电路元件的参数值由于生产工艺的原因存在制造值波动的情况下（一般为正态分布），界面可以展示这种波动，并将波动以典型值为圆心，波动概率对应的值为半径，画出一组不同半径的同心圆，这些同心圆的不同边界代表了不同的波动概率值。因此在前述的交集的有效区域内加上这些同心圆，就可以清楚的看出来波动概率下的最优解，和元件参数波动本身对设计指标的影响程度。

图 4-9 和图 4-10、图 4-11 分别给出了二维和三维图形化接口界面的展示。

二维图形化接口的敏感度相关的结果展示将会在本文下一章节中详细介绍。从图 4-9 中我们可以直接的获知相关按钮和输入域的功能和操作。而作为跟随鼠标移动的绘图区光标可以实时的读取当前频率和幅度及相位的信息并显示在相关的文本框内。图形接口中的波形还能输出保存为 png、jpg 等图片格式，以供科学研究用。

三维图形化接口中的三维图像可以通过键盘的操作进行各个角度的移动和翻转，可以放大缩小，可以移动切面并保存切线，从而用以构建符合多设计指标的有效设计区域。工艺参数的波动的正态分布指数在仿真参数配置接口中进行设置，同时整个同心圆可以通过具体的按键关闭或者打开。

虽然在三维图形化接口中也提供了单位增益频率和相位裕度关于两个坐标轴元件的敏感度的图像，但是因为这个敏感度的图像是用数值的方法计算获得的，主要的目的是为了进行同符号化的方法获得的敏感度的值作比较，因此会因为微

小误差而导致巨大的数值（也就是过冲现象），反应在图 4-11 中就是明显的直角锯齿切线的存在。一般来说这个敏感度功能仅仅具有比较意义，不具有实际价值。敏感度的获取还是要通过二维的图形化接口获得。

二维和三维的图形化接口的背后的算法除了绘图和渲染相关的算法之外，主要是和平台的符号化仿真器引擎交互的算法。

二维的接口所提供的幅频和相频曲线在元件参数变化情况下的更新就是通过将元件的参数的变化最终转移到电路对应的已构建的二叉决定图的符号的值的变

化（对于线性元件，可以直接进行对应符号的值更新；而对于非线性元件的参数，比如 CMOS 晶体管尺寸的更新，则需要通过小信号模型做中转，并利用公式(3-3)将尺寸的值转移到各个对应的小信号模型中的线性元件中去），并完成一次快速的求值操作的方式进行的。由于二叉决定图的求值非常的迅速，因此无论多大的电路，在曲线更新的实时性上都不会受到任何影响。事实上每个二维图形接口的一个值变化操作对应到符号化仿真器引擎上都是一次相应符号值更新和整个二叉决定图重新求值的操作。

三维的图形化接口在三维图生成之后就基本固定，后面的所有关于三维图像的各种操作都是基于 glut 库的 API 调用的。而和符号化仿真器引擎交互的事情都是发生在三维图初始化的时候。在初始化的时候，通过不断地提供各个电路元件和频率符号以新值，然后进行对应的二叉决定图求值操作，在对求得的值做处理之后，就得到了一组点集，将这些点集以画三角形的方式呈现在图上经过渲染就是我们看到的三维图形了。

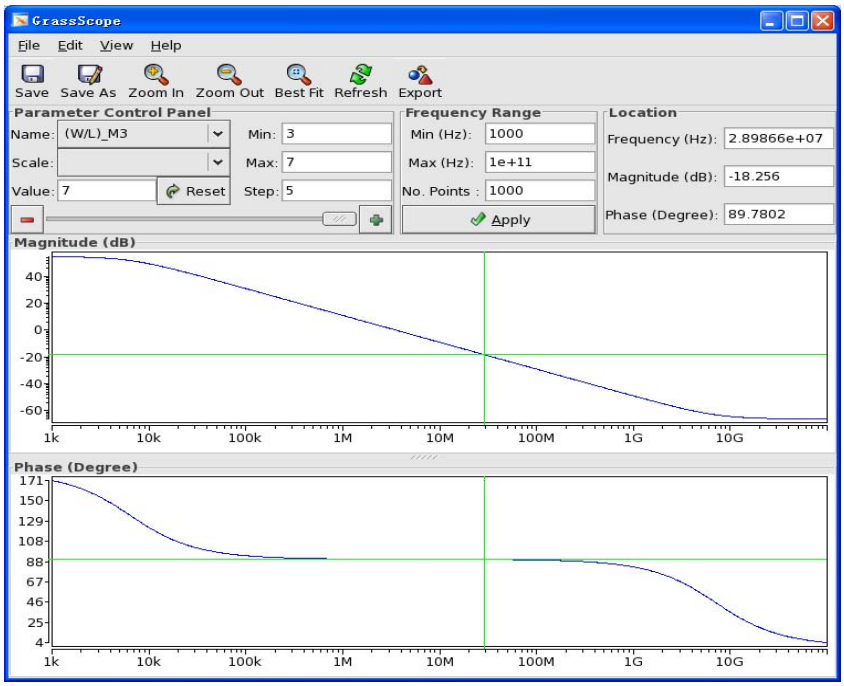
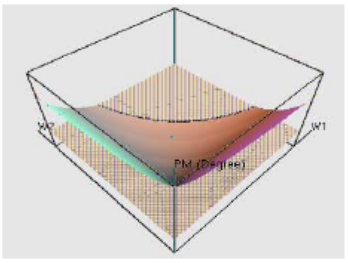
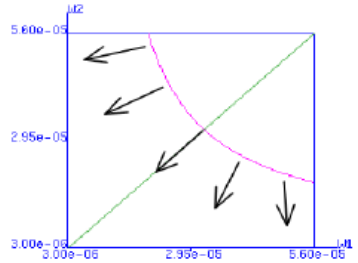


图 4-9 二维图形化接口界面

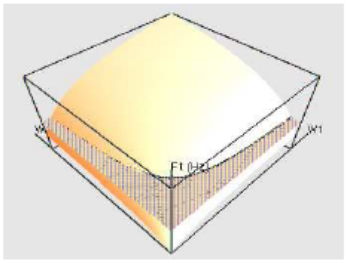
Fig.4-9 2D GUI



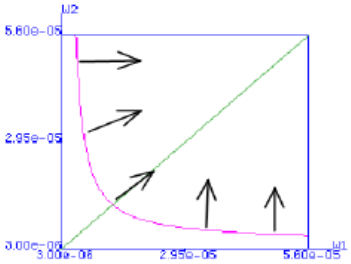
(a) 3D phase margin plot.



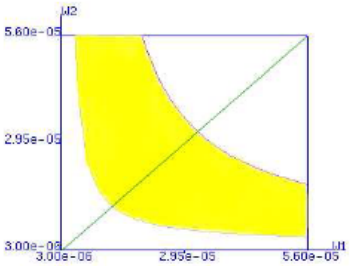
(b) Equi-PM contour.



(c) 3D unity-gain frequency plot.



(d) Equi-f_t contour.



(e) Design feasibility region.

图 4-10 三维图形化接口界面

Fig.4-10 3D GUI

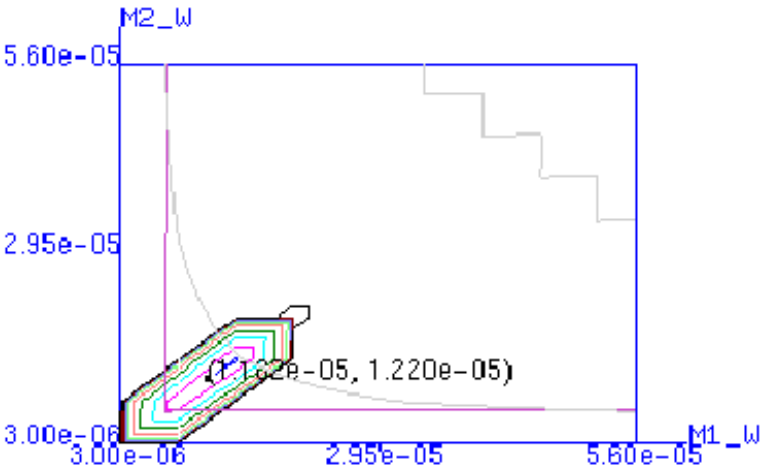


图 4-11 三维考虑工艺参数波动的可能的有效区域图

Fig.4-11 3D GUI considering fab violation

4.6 符号化公式编译器

符号化公式编辑器是符号化的模拟电路设计解探索平台提供的一个类似插件工具的功能，该功能基本独立于整个平台的操作流之外，提供用户和设计者以方便的符号化公式编辑、求解和波形显示等功能，类似于 Matlab 的画图功能。目前符号化的公式编辑器仍在开发之中，图 4-12 显示了他的一个设计架构图。在未来进一步的研究中，符号化的公式编辑器将作为电路拓扑结构设计接口的一个插件工具内嵌进接口中，提供用户和设计者对元件所采用模型的传输函数的直接描述，这样对于一些确定传输函数行为的元件或者不重要或者很重要的元件，仿真器可以区别对待，以使得仿真更加有针对性，也更加的准确和快速。

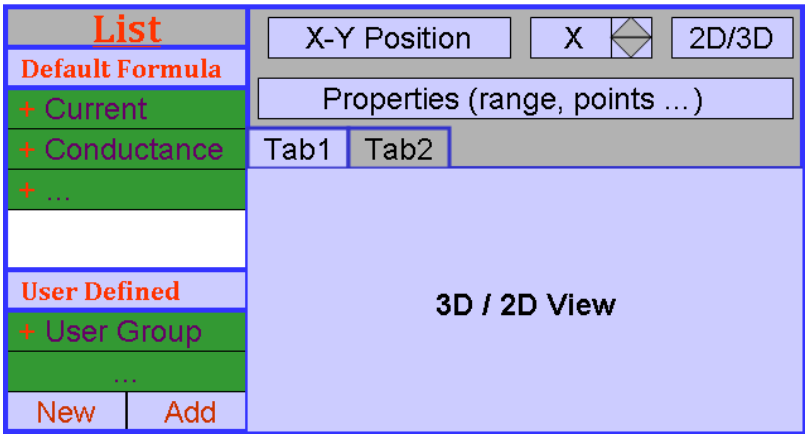


图 4-12 符号化公式编辑器设计架构图
Fig.4-12 Design architecture of symbolic formula editor

4.7 本章小结

本章主要介绍了模拟电路设计解探索平台的框架和构成这个平台除了符号化仿真器引擎以外的各个组件的设计原理、设计功能、行为模式和实现技巧。是本文的核心章节之一。整个模拟集成电路设计解探索平台本着自动化的设计动机，尽可能的将用户的操作行为变得直截了当、易于理解，而将复杂的流程化的手工命令和设置隐藏起来，让用户感觉不到他们的存在。整个平台为了保证程序的健壮性做了大量的工作。平台的图形化界面的用户友好化设计和 xml 动态构建算法使得平台在用户体验上有了很大的提升，而且在亲近用户的设计上变得非常灵活，游刃有余。同时二维和三维的图形化界面所提供的关于电路的信息和内容足够丰富，让用户能够在平台的帮助下寻找明确设计的方向，快速完成设计的收敛，同时又能进一步加深对电路的理解和把握。

第五章 电路设计例子应用和分析

本章主要通过对两个设计实例的分析和演示，说明本文介绍的模拟电路设计解探索平台的使用方法和符号化仿真器提供的电路传输函数关于CMOS晶体管尺寸敏感度在频域的分布所包含的特殊信息和可能的解读。

5.1 电路设计例子I

本节所采用的电路设计实例是一个由5个CMOS晶体管构成的二级差分输入单输出的运算放大器，如图5-1所示。

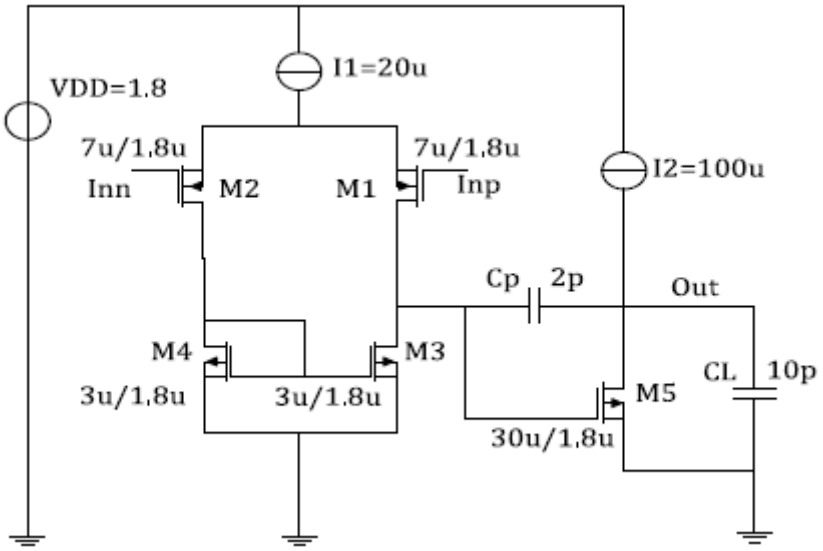


图 5-1 二级差分输入单输出运算放大器电路
Fig.5-1 A single-output two-stage differential amplifier

从图 5-1 中我们可以看到，这个电路的传输管的尺寸的值能够保证电路的传输管都工作在饱和区，同时我们采用输入正负极均为 0.6v 直流偏置，总共 1v 交流偏置（也就是输入正端接 1v 交流电源，输入负端接地）的电路偏置方式。我们采用的仿真库是 0.18 微米的仿真工艺库。为了降低电路的复杂程度，方便我们讨论和分析，将原始电路中的电流偏置管全部直接用理想电流源替代，用以提供稳定的电流输出和偏置。

5.1.1 电路特性分析

我们这里用到的这个运算放大器电路是教科书上非常常见的经典电路，对他

的分析和讨论随处可见，有大量的资料可以参考。因为他的偏置电路已经全部被理想电流源取代，因此分析上将会更加的简单。针对我们行文的需要，我们将会将这个电路的几个主要的属性做一个简单的描述。

M1、M2、M3 和M4 构成了二级运算放大器的第一级，也就是输入级；而M5 则是这个运算放大器的第二级，也就是放大级。两级之间通过米勒补偿电容C_p进行联系，C_L是负载电容。在第一级电路中，M1 和M2 构成了电路的差分输入，一方面主要负责对噪声信号的抑制和滤除，另一方面也兼有部分对信号的放大作用。M3 和M4 是维持偏置电流的反馈镜像对。他们的主要作用就是将两个差分输入端的分支的偏置电流保持在一致。他们基本没有放大作用，对电路主要性能指标的影响很小，起辅助作用。

在使用小信号模型进行电路分析后，这个电路的放大增益可以近似由公式(5-1)表示：

$$A_v = A_1 A_2 = g_{m1}(r_{o2} \| r_{o4}) g_{m5} r_{o5} \tag{5-1}$$

其中，A_1和A_2分别表示运算放大器的第一级和第二级的增益，r_o表示输出电阻，也即图 3-1b小信号模型中的G_{ds}，g_m就是图 3-1b小信号模型中的G_m，数字表示对应的晶体管，因为M1 和M2 是一对，M3 和M4 是一对，因此，g_m和r_o的值两个成对的管子都是一样的。符号“||”表示元件之间是并联关系。

由于实际电路中，g_m和晶体管的尺寸关系最为紧密，一般情况下，晶体管宽度的增加将会导致g_m在一定范围内增大，因此从公式(5-1)中我们可以发现M1 和M2 的管子尺寸对增益的影响是显著的，而M3 和M4 的管子尺寸相比于M1 和M2 来说就没有那么重要了。虽然M5 在公式中的地位同M1 和M2 类似，但是由于M5 作为运算放大器的放大级，是实际放大的主要电路，因此M5 的尺寸对增益的影响显然是最大的。通常情况下g_{m5}的值会较g_{m1}大出很多，对应的M5 的宽长比也要比M1 和M2 大出很多，M5 所在的分支的偏置电流也成比例的比M1 和M2 所在分支的偏置电流大出很多。

另外，在电路零极点方面，有如下的近似公式[40]：

$$f_{p1} = \frac{1}{2\pi g_{m5}(r_{o2} \| r_{o4}) r_{o5} C_p} \tag{5-2}$$

$$f_{p2} = \frac{g_{m5}}{2\pi C_L} \tag{5-3}$$

$$f_z = \frac{g_{m5}}{2\pi C_p} \tag{5-4}$$

其中，f_p表示极点，f_z表示零点。

从公式(5-2)到公式(5-4)中可以看到，M1 和 M2 的尺寸对主极点、次极点和主零点都没有太大的影响，M3 和 M4 的尺寸也是一样的情况。M5 的尺寸影响对这三个零极点都很大。其中，对主极点是反相关的，而对次极点和主零点则是正相关的，也就是说，如果增大 M5 的管子尺寸（当然，为了保证电路的电流能够不失配，M3 和 M4 这两个控制支路电流的管子尺寸也要做同比例的增加）的话，那么电路的主极点将会变小而左移，电路的次极点和主零点则会变大而右移，这将造成标准的电路幅频响应的曲线进一步被拉升，3db 位置进一步靠近 y 轴，而曲线与 x 轴的交点则进一步远离。

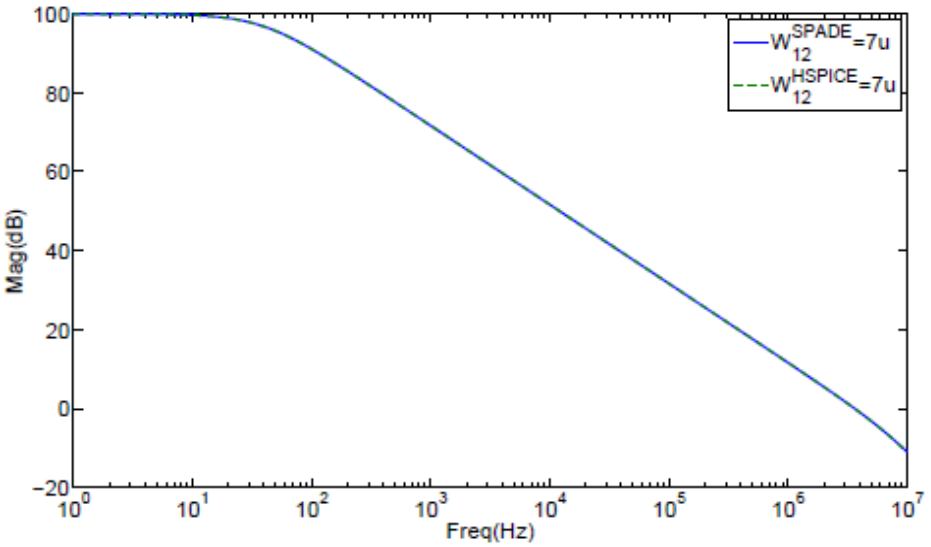
除此之外，在电路单位增益频率，也就是增益带宽积方面，有如下的近似公式：

$$f_t = |A_v| f_{p1} = \frac{g_{m1}}{2\pi C_p} \tag{5-5}$$

从公式(5-5)可以看出 M1 和 M2 的尺寸对增益带宽积的影响是正相关的，而 M3、M4 和 M5 则都没有什么影响。换句话说，当 M1 和 M2 的尺寸增加时，增益带宽积变大，幅频响应的曲线和 x 轴的交点会被右移，从而使得幅频响应曲线的下降的斜率变小。

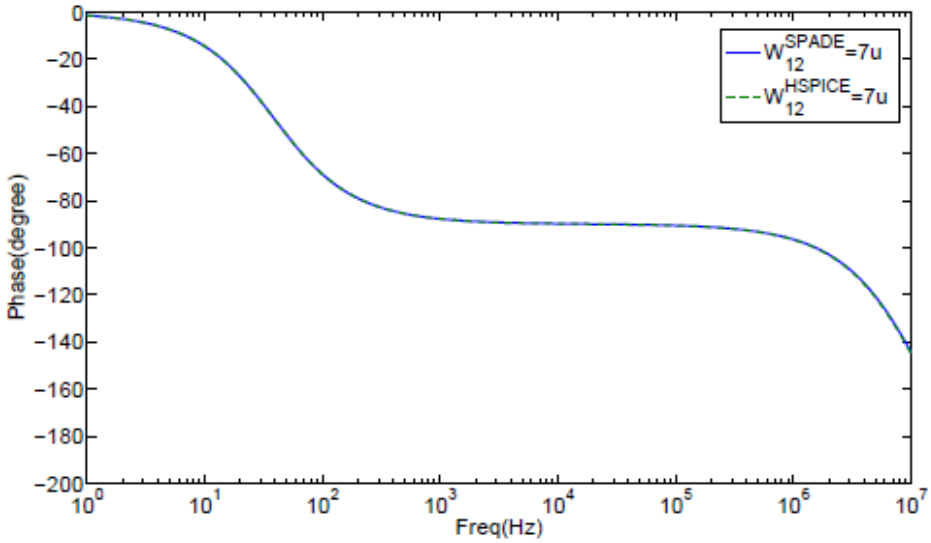
5.1.2 符号化仿真器结果与HSpice结果比较

根据本文前述章节描述的符号化模拟电路设计解探索平台的使用流程，在按照流程执行外操作后，我们将获得当前电路各元件参数下的符号化仿真的传输函数的幅频响应曲线和相频响应曲线。我们首先将这个曲线同 HSpice 的直接仿真结果进行比较，以确认我们所采用的小信号模型参数提取方法的准确性和可操作性。图 5-2 给出了这两个的方法的比较结果。



a) 幅频曲线 ($W_{1,2}$=7μ)

a) Frequency Response ($W_{1,2}$=7μ)



b) 相频曲线 ($W_{1,2}$=7μ)

b) Phase Response ($W_{1,2}$=7μ)

图 5-2 符号化仿真器GRASS同HSpice的仿真结果比较 ($W_{1,2}$=7μ)

Fig.5-2 Simulation results by GRASS and HSpice ($W_{1,2}$=7μ)

从图 5-2 中我们可以看到，符号化仿真器和商业化的标准数值仿真器的结果几乎完全一致。因此也证明了我们所采用的方法和模型的可靠性和可操作性。

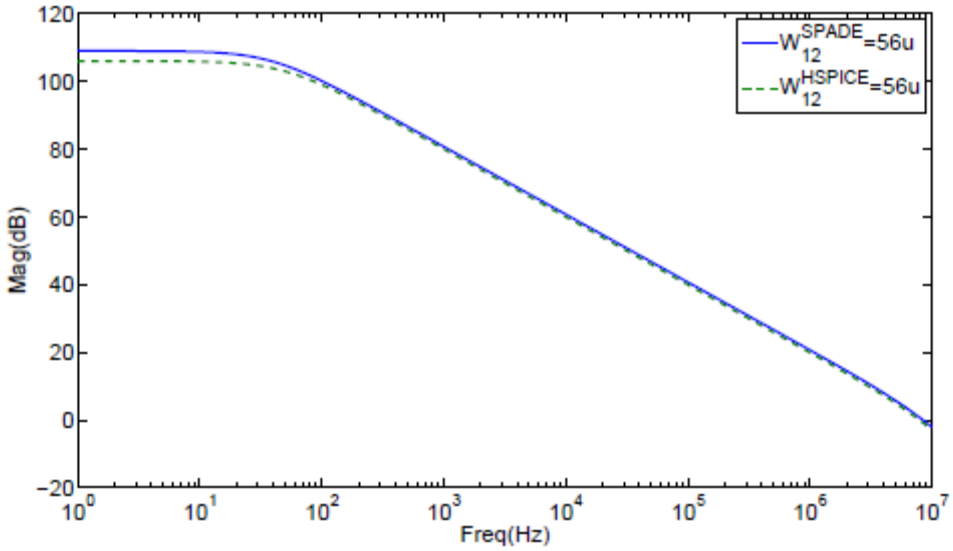
下面我们利用符号化模拟电路设计解探索平台的能力进行电路设计。假设我们的电路设计指标中要求：

电路的增益大小达到 100dB 以上；

单位增益带宽达到 7.5MHz 以上；

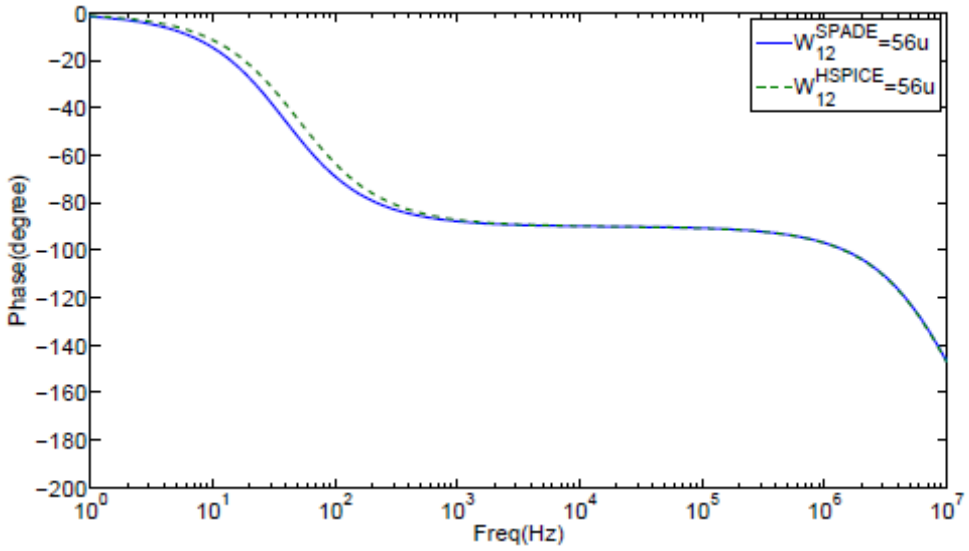
而我们当前仿真结果所显示的电路的这两个指标分别为：接近 100dB 和 3.6MHz。现在先假设我们当前的设计并没有得到符号化 CMOS 晶体管尺寸敏感度的支持，我们对电路的指标特性同晶体管尺寸之间的关系也不清楚，而是一种完全盲从的尝试。待会儿我们会看到敏感度的数据和波形会给我们带来多么巨大的帮助和好处。

在没有敏感度的支持下，我们先选定图 4-9 中的元件对象选择框，选择 M1 和 M2 并进行向左和向右的拖动。我们先选择向左的拖动，也就是将 M1 和 M2 的宽度减小，我们发现这个时候，电路的属性，也就是电路的增益和单位增益带宽的值，已经进一步远离了我们的指标，都在变小，因此这个时候我们改变方向，改为增大 M1 和 M2 的宽度大小。当我们将 M1 和 M2 的宽度增加到 56μ 米的时候，我们发现这个时候的电路增益大小和单位增益带宽都已经符合我们一开始设定的电路指标了，也就是说这时候对所描述的电路指标的 CMOS 晶体管的参数设计已经完成。现在让我们来看一下在新的这个 CMOS 晶体管的尺寸下（这时候的尺寸已经是原来初始尺寸的 8 倍了），符号化仿真器的精度和可靠性是否仍然能够得到保证呢。图 5-3 显示了这个结果。



a) 幅频曲线 ($W_{1,2}$=56μ)

a) Frequency Response ($W_{1,2}$=56μ)



b) 相频曲线 ($W_{1,2}$=56μ)

b) Phase Response ($W_{1,2}$=56μ)

图 5-3 符号化仿真器GRASS同HSpice的仿真结果比较 ($W_{1,2}$=56μ)

Fig.5-3 Simulation results by GRASS and HSpice ($W_{1,2}$=56μ)

图 5-3 很好的说明了符号化仿真器对于 CMOS 晶体管尺寸参数在一定范围内变化情况下的仿真精度是高度可靠的。由于这个过程中并没有重新进行电流直流工作点的分析和二叉决定图的重新构建，而仅仅是对二叉决定图进行求值操作，因此这个过程大大发挥了符号化仿真器的优势，使得仿真速度上得到了大大的提升。

在这个设计过程中我们发现，虽然我们盲目地在尝试可能合适的值，但是因为我们能够快速的得到仿真结果，所以我们的设计还是很快的达到了收敛。符号化的威力可见一斑。

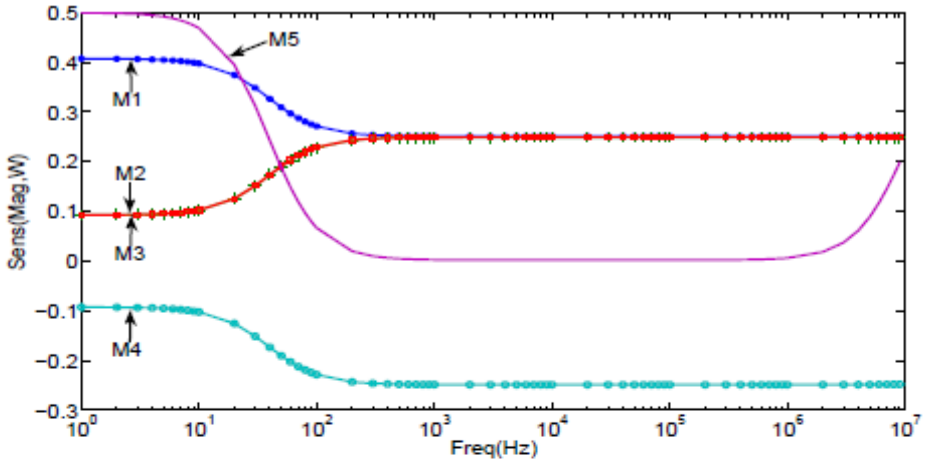
5.1.3 符号化CMOS晶体管尺寸敏感度

现在我们考虑加入符号化 CMOS 晶体管尺寸敏感度的支持。二维图形化界面对于符号化敏感度的支持采用的是借助 Matlab 的方法获得的。通过内嵌对 Matlab 相关程序的调用命令，可以在二进制程序中直接完成和 Matlab 的对接。

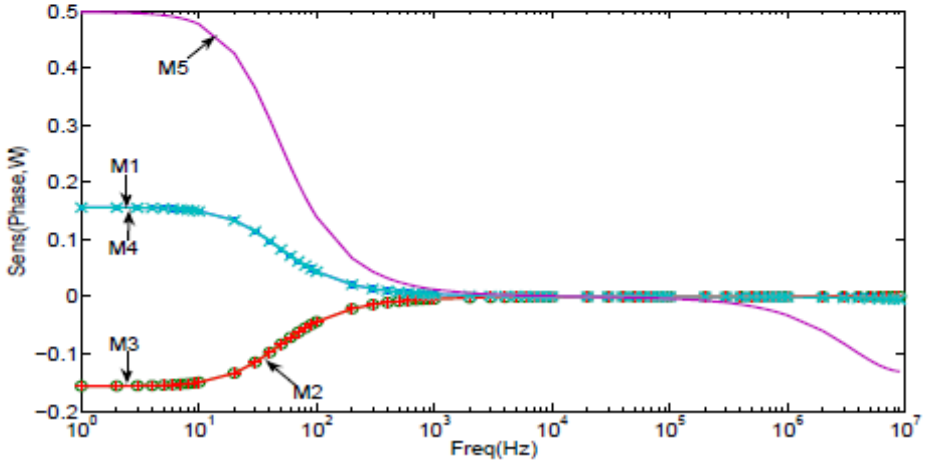
对于传输函数关于晶体管尺寸的敏感度问题，由于存在晶体管配对的情况（也就是为了符合电路设计规则和保证电路设计的正确性，若干个晶体管的尺寸永远同时按照一定的比例关系进行变化），一般的敏感度问题会分为不考虑晶体管配对和考虑晶体管配对两种情况。根据敏感度的定义和偏微分的定义，对于考虑晶体

管配对的情况的处理，只要将配对的晶体管按照比例系数作为权重，将各自的敏感度进行累加即可。

图 5-4 给出了在电路初始化的元件参数值情况下的传输函数关于 5 个 CMOS 晶体管尺寸敏感度的幅频和相频曲线。而图 5-5 则给出了在考虑了晶体管配对的情况下（也就是 M1 和 M2 配对，M3 和 M4 配对）的相应曲线。



(a) Magnitude sensitivity plot.



(b) Phase sensitivity plot.

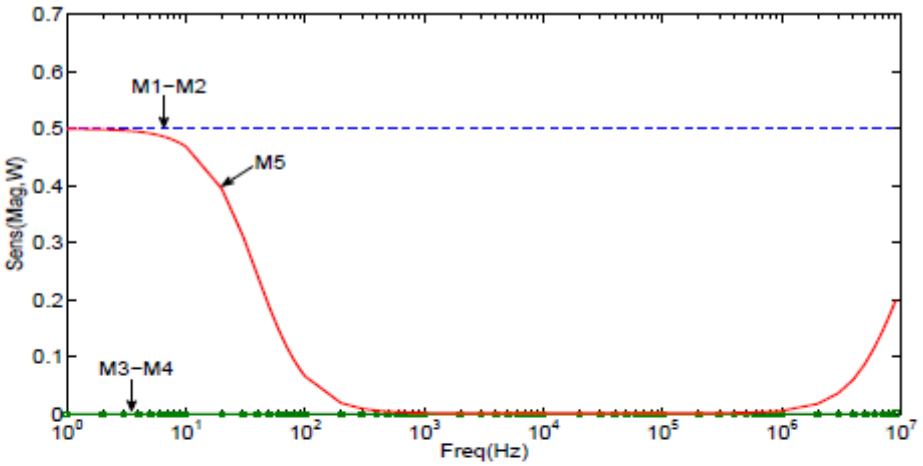
图 5-4 各个晶体管独立的传输函数关于尺寸的符号化敏感度（图 5-1）
Fig.5-4 Independent symbolic sensitivity for each transistor (Fig. 5-1)

从图 5-4 中我们可以清楚地看到每个晶体管敏感度的幅频和相频曲线都在 10Hz到 100Hz之间做了一个波动，虽然波动的起点和终点不尽相同，但是波动大概都在 38.7Hz的位置达到中间值。由于传输函数关于晶体管尺寸的敏感度的幅频和相频响应是其实部和虚部响应的一种关系变换，因此基本上这和本文前述的理

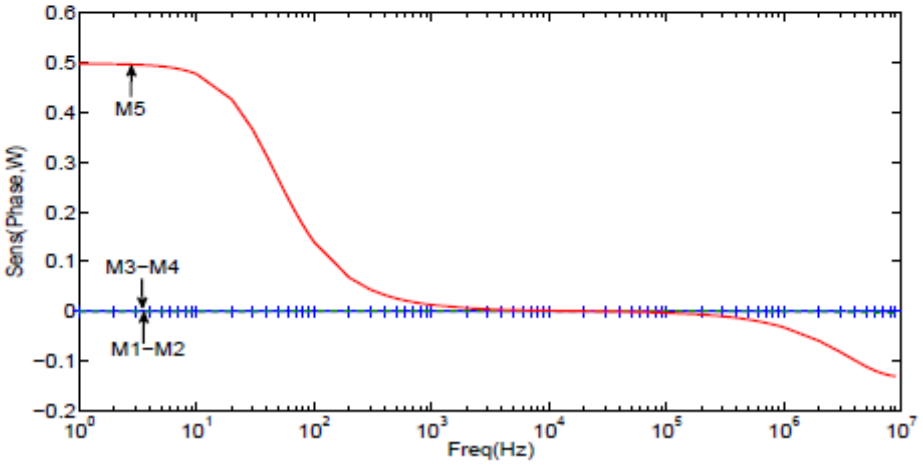
论推导完全一致。考虑到公式(3-13)，我们更加可以确认，幅频响应的敏感度曲线同实部的敏感度曲线是完全一致的，而这个曲线的行为完全同第三章中所做的分析一致，也就是对初始值为$Sens(p_i|z_i, W_k)$且经过频率和零点或者极点相同的点时达到初始值的一半的并继续趋近于 0 的一系列曲线的叠加。考虑到主极点的位置远离次极点和零点的情况，基本上我们从图 5-4 中看到的第一个曲线变化的情况就是主极点主导的情况，这个拐弯的中点位置恰好就是主极点位置，因而曲线起点的正负和数值恰好就是主极点关于晶体管尺寸的敏感度的值，这点我们将在下

一节进一步予以证明。而曲线变化的第二个地方（在这个例子中只有 M5 的曲线

发生了变化，说明只有 M5 的管子的尺寸参数会对这个变化点有影响）则是次极



(a) Magnitude sensitivity plot.



(b) Phase sensitivity plot.

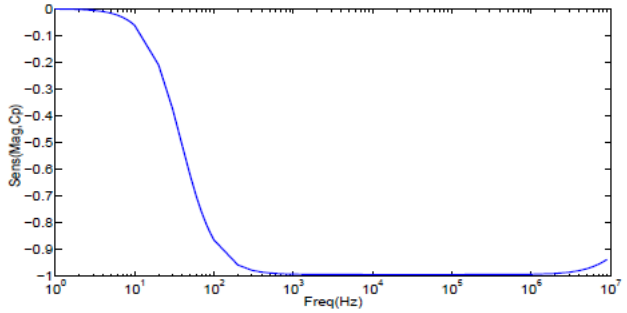
图 5-5 考虑了配对的晶体管的传输函数关于尺寸的符号化敏感度（图 5-1）
Fig.5-5 Symbolic sensitivity for transistors considering match: M1-M2, M3-M4 (Fig. 5-1)

点或者主零点的位置，以小者为准。这种情况恰好说明了公式(5-3)和(5-4)，同时我们可以从图中发现，此时M5 曲线上升到初始值一般的位置大概是 10^7，这和我们通过HSpice得到的次极点的值 10498100 极为接近，再次验证了我们的公式和分析。

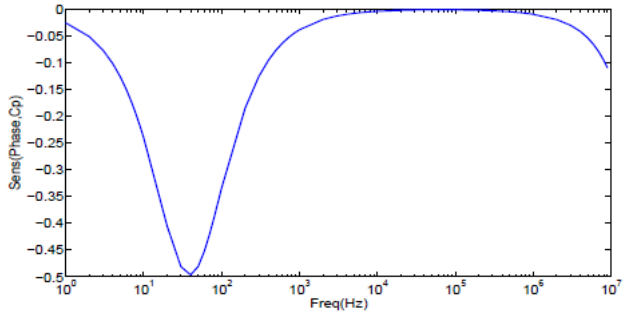
进一步的我们可以发现，图 5-4 中的M1 和M2、M3 和M4 这两对配对的晶体管的曲线，无论是幅频还是相频，都惊人的对称，图 5-5 的结果验证了这个对称的观察。图 5-5 中M1-M2 和M3-M4 晶体管对的幅频和相频曲线都近乎一条水平的线，这恰恰是配对的两个晶体管对传输函数的幅频和相频的影响互相抵消的结果。考虑到实际电路设计中配对晶体管的尺寸一般永远同步变化，我们参照图 5-5 做下面进一步的分析。从图 5-5 中我们发现，对于放大增益，M1-M2 和M5 有基本一样的初始值，这也恰恰证明了这两者在公式(5-2)中g_{m1}和g_{m5}的平等地位，他们对放大增益的影响在不考虑实际值的情况下是一样的。考虑到两者的值均为正，所以他们对增益的影响是正相关的。M3-M4 对放大增益的贡献几乎为 0，这也从侧面说明了在公式(5-2)中的r_o的作用不是很大，而M3-M4 除了r_o就没有其他影响因子了。事实上，通过放大图 5-5 我们可以发现，在图 5-5a中，M3-M4 的曲线并不是和x轴重合，而是在x轴上方一点位置，这恰好就是r_o所带来的贡献。关于零极点的分析已经在前面做过讨论了，图 5-5 进一步说明了M1-M2 和M3-M4 对主极点、次极点和主零点的贡献几乎没有。和M5 的情况是其主极点、次极点和主零点 3 个主要零极点叠加的结果，因此会出现曲线变化先是先减少后增加的情况，而这个情况对应的恰好是M5 的主极点和次极点以及主零点影响效果的相反的分析。对单位增益频率来说，M1-M2 对幅频的持续正作用相当于将幅频曲线做了一个全频域的向上平移操作，这个操作的结果是间接增加了单位增益频率的大小，而这又验证了公式(5-5)的正确性。

因此依靠以上的分析，事实上我们在手动调整晶体管的尺寸参数的时候，我们很容易就能从图 5-4 和图 5-5 中发现，对应我们指标的关键晶体管正是 M1 和M2，而由于正作用的敏感度，我们只需要增大 M1-M2 的宽度值到符合指标参数的打小就可以了，这样设计收敛性得到了大大的提升，设计的方向非常明确而且直接，省去了盲目的调试和方向性尝试，节省了大量的时间。

为了进一步说明本文上述章节关于传输函数敏感度零极点理论分析的正确性，图 5-6 展示了传输函数关于C_p的幅频和相频敏感度结果。



(a) Gain sensitivities.



(b) Phase sensitivities.

图 5-6 传输函数关于米勒补偿电容C_p的符号化敏感度

Fig.5-6 Symbolic sensitivity for C_p

因为C_p本身的单一性，它的敏感度曲线远没有CMOS晶体管那么复杂（晶体管小信号模型包含了很多个线性元件）。我们从图 5-6 中很好的看到了类似图 3-4 的信息，同时我们也可以用图 5-6 验证公式(5-2)、(5-4)和(5-5)。零极点方面，第一个拐点很好的反映了主极点的位置，同时幅频曲线的负值和相频曲线的尖峰朝下都说明了C_p对主极点的负相关作用。而由于本例中次极点的位置相对主零点靠前，因此，在图 5-6 右边界的地方，C_p的波形刚刚开始变化，他会在更远的地方达到他的拐点。在单位增益频率方面，持续的幅频负敏感度值使得C_p对他的影响是负相关的，而在-1 值上长长的水平线说明C_p的作用刚好是反比例的。因此图 5-6 非常完美的诠释了公式的意义，解释了电路特性和电路元件参数之间的内在关系。

最后，为了证明符号化敏感度求解的正确性，我们采用了将求得的符号化敏感度同用HSpice的数值结果做验证的方法，这里我们选取M5 这个变化范围最大的晶体管（在用HSpice求解敏感度的值的时候，为了保证电流不失配，我们采用同比例增加M3-M4 的尺寸宽度，也就是说W_5=31μ，$W_{3,4}$=3.1μ）。图 5-7 显示了这个结果。

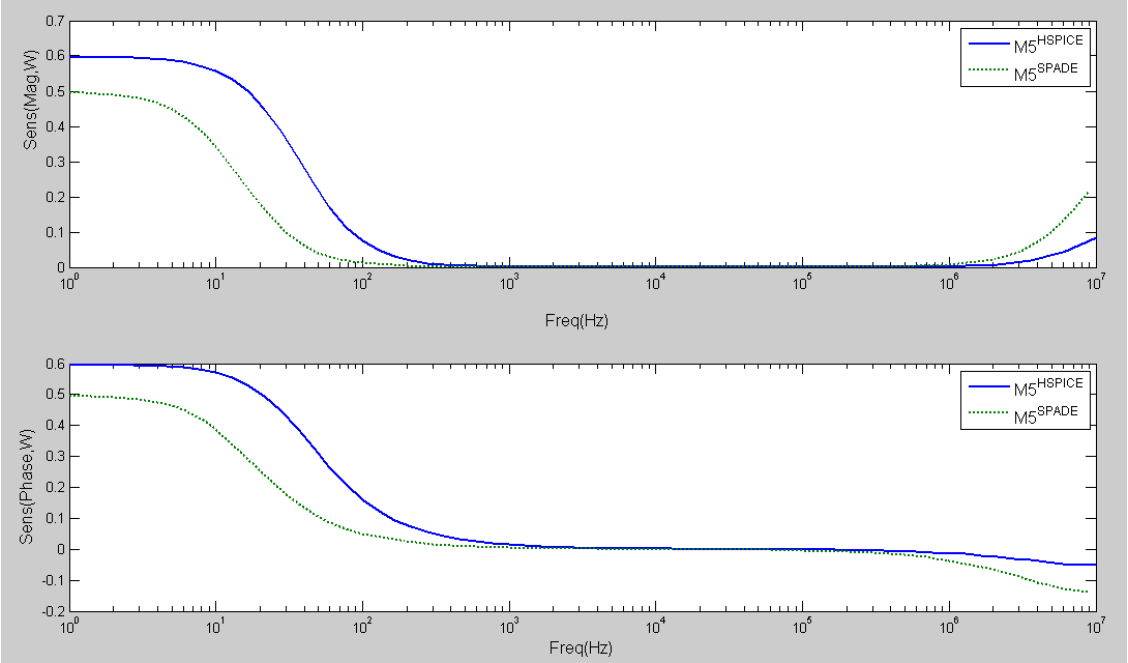


图 5-7 传输函数关于晶体管 M5 尺寸敏感度数值和符号化对比

Fig.5-7 Symbolic sensitivity VS Numerical sensitivity of W_5

从图 5-7 中可以发现，符号化敏感度和数值敏感度之间计算的值是有差别的，但是基本的形状是大致一致的。具体到 M5 来说，实际上由于数值仿真是一个近似求解敏感度的方法，而 M5 的变动会改变整个电路晶体管所在的工作区，因此数值仿真的敏感度距离敏感度的真实值较符号化敏感度更远一些。而且这个方法计算得到的敏感度由于为了保证电路支路电流匹配是加入了 M3-M4 的效果的（虽然 M3-M4 对传输管幅频和相频的敏感度的影响都非常小）。而符号化敏感度的精度问题主要取决于小型号模型的精确度和复杂程度。另外数值仿真可能存在因为数值不稳定性而导致的敏感度求解无穷大的情况，进一步造成敏感度求解的误差。总的来说，二者的大致形状的一致性足以提供我们足够的信息，进行模拟电路的设计和关键器件的判断。因此基于模拟电路传输函数晶体管尺寸敏感度的方法学和理论分析是可靠的。

5.1.4 符号化主极点和其敏感度

符号化的主极点和主极点关于各个晶体尺寸管尺寸敏感度的求解采用的方法的理论证明在前述的章节中已经给出，具体的算法也已经做过说明。表 5-1 给出了本案例电路的符号化求解的电路主极点和该主极点关于各晶体管尺寸敏感度的值。并列出了数值的结果用以比较。

表 5-1 符号化主极点及主极点关于各晶体管尺寸敏感度（图 5-1）
Table 5-1 Symbolic dominant pole & related sensitivity (Fig. 5-1)

项目	符号化方法		数值化方法
主极点P_d (Hz)	38.67		38.6668
Sens(P_d,W_1)	-0.156424	0.000000	0.018496
Sens(P_d,W_2)	0.156424		
Sens(P_d,W_3)	0.156694	0.000603	(因电流匹配随 M5 同比例变化，故影响已计入 M5)
Sens(P_d,W_4)	-0.156091		
Sens(P_d,W_5)	-0.497918		-0.573709

某种意义上说表 5-1 进一步说明了符号化同数值化方法的差别。符号化的方法的精确程度取决于所采用的模型的精度和复杂程度，由于所采用的小信号模型本身的复杂程度限制，符号化方法无法像数值化方法那样精确的符合实际，但是也是因为符号化方法符号的缘故，各种电路特性都能够很理想的得到体现。而数值化方法则显得整体性更强，管子之间的关联性非常紧密。这里之所以采用配对的方式进行测量，就是因为不这么做的话，电路的部分晶体管就会离开饱和区而进入线性区，甚至截止区，因而也只能采用若干个管子一起测量的方式进行评测。因此可以这么说，符号化反映的是电路理想情况的内在联系，是电路的本质属性的揭示；而数值化反映的是电路实际情况的表征，是电路内在形式的具体表现。符号化方法是内涵，数值化方法是外延。

现在我们仅仅讨论符号化方法的结果。从表 5-1 中我们可以发现，这里的数据和我们求得的符号化的幅频响应和相频响应的敏感度曲线几乎完全一致。大部分的分析本文前述章节已经做过了，此处仅对 M5 的敏感度为负值但是我们得到的幅频曲线波形的初始值（见图 5-4 和图 5-5）却为正值作出解释。

本节讨论的设计电路的主要零极点就是主极点、次极点和主零点，于是根据公式(3-11a)的证明，我们有：

$$Sens(|H(s)|,W_k)=\sum_j Sens(z_j,W_k)\frac{1}{(1+(\frac{\omega}{z_j})^2)}-\sum_i Sens(p_i,W_k)\frac{1}{(1+(\frac{\omega}{p_i})^2)}$$

$$=Sens(z_1,W_k)\frac{1}{(1+(\frac{\omega}{z_1})^2)}-(Sens(p_1,W_k)\frac{1}{(1+(\frac{\omega}{p_1})^2)}+Sens(p_2,W_k)\frac{1}{(1+(\frac{\omega}{p_2})^2)})$$

$$=Sens(g_{m5},W_k)\frac{1}{(1+(\frac{\omega}{z_1})^2)}-(Sens(p_1,W_k)\frac{1}{(1+(\frac{\omega}{p_1})^2)}+Sens(g_{m5},W_k)\frac{1}{(1+(\frac{\omega}{p_2})^2)}) \quad (5\text{-}6)$$

而对于敏感度的初始值，我们可以做一个简单的加法，也就是：

$$\begin{aligned} &Sens(|H(0)|,W_5) = Sens(g_{m5},W_5) - Sens(p_1,W_5) - Sens(g_{m5},W_5) \\ &= -Sens(p_1,W_5) = -(-0.497918) = 0.497918 \end{aligned} \tag{5-7}$$

其他的敏感度曲线的初始值都可以通过这个方法计算获得。

5.1.5 符号化多设计目标的有效域

符号化多设计目标的有效域问题是符号化模拟电路解探索平台的三维图形化接口提供的一个开创性的功能。本节中的案例电路利用该平台工具生成的三维界面如图 4-10 和 4-11 所示。我们在这个设计中选择差分输入对 M1 和 M2 的宽度作为我们的两个电路参数，选择单位增益频率和相位裕度作为我们的两个设计指标。我们的目的是为了找到同时符合两个设计指标的一组有效地 M1 和 M2 的值。考虑到 M1 和 M2 在设计中是配对的，他们的尺寸将会同时变化，因此图 4-10 给出了一条绿色的对角线，表征我们需要的问题解都应该在这条线上。设计中，我们将 M1 和 M2 的设计参考值设在 3μ 到 56μ 之间。

为了实现设计，我们先切换到图 4-10a 所示的三维界面，也就是相位裕度界面，然后将水平切面移至我们的设计指标值，通过快捷键将曲面的切线保留下来，如图 4-10b 所示，图中的箭头方向表示更大的相位裕度对应的晶体管尺寸有效区；然后我们再将图形界面切换至单位增益频率界面，如图 4-10c 所示，同样将水平切面移至我们的设计指标值并保存切线，如图 4-10d 所示，箭头方向表示更大的单位增益频率对应的晶体管尺寸有效区。最后我们将两个有效区相交，我们就得到了如图 4-10e 中黄色区域所示的同时符合两个设计指标的晶体管尺寸有效区。黄色区域中的绿线就是符合要求的值。

其实我们可以发现，由于通过三维图形化界面我们可以清楚的观察到 M1 和 M2 两个晶体管尺寸不一致的情况下的电路设计指标的变化情况。我们还可以利用平台提供的三维图形化接口作为我们分析晶体管尺寸不匹配情况下电路行为的一种参考。由于符号化的方法在求值时不再进行二叉决定图的构建，因此对于小信号模型的线性元件值的更新仍然是基于晶体管工作在饱和区的假设，因此这种参考仅限晶体管尺寸的波动在有限的范围内。

5.2 电路设计例子II

本节所采用的电路设计实例是一个由 15 个CMOS晶体管（有三个晶体管以自饱和的方式作为V_{b1}、V_{b2}和V_{b3}的偏置管没有画出来）构成的三级差分输入单输出

的运算放大器，如图 5-8 所示。

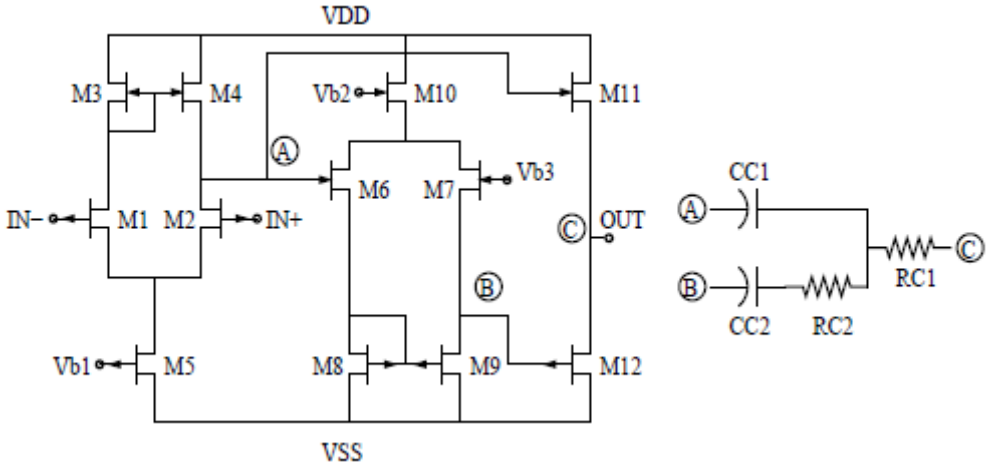


图 5-8 三级差分输入单输出运算放大器电路[41]

Fig.5-8 A single-output three-stage differential amplifier[41]

图 5-8 中的反馈回路采用C_{C1}=5pF，C_{C2}=20pF，R_{C1}=600Ω，R_{C2}=5Ω的值配置。其他电路管子的参数和工作点环境配置请参考[41]。同样的，这个电路的所有管子的初始尺寸参数保证这个电路的所有管子都已经进入饱和区。我们的仿真采用的仍然是 0.18 微米的仿真工艺库。

5.2.1 电路特性分析

这个电路的具体分析请参照论文[41]中的描述，本文仅选择本文需要的电路特性做简单的总结。

这个电路一共分三级。其中 M1、M2、M3、M4 和 M5 组成了电路的第一级，也就是差分输入级。M1 和 M2 起到主要的差分和放大作用，M3、M4 和 M5 构成电路的偏置电流设定电路；M6、M7、M8、M9 和 M10 构成了电路的第二级，也就是电路的放大级。M6 和 M7 一方面起到了放大的作用，另一方面也起到了对差分信号的处理的功能。M8、M9 和 M10 是电路的偏置电流设定电路；M11 和 M12 构成了电路的第三级，也就是放大输出级。M12 是主要的放大管，因为 M11 的输入来自差分输入端的一支，因此 M11 也具有一定的放大能力，同时 M11 也作为 M12 的偏置电流设定电路。

根据论文中的分析，我们有放大增益和主极点这两个主要的电路指标的公式如下[41]：

$$A(s)=A_o\frac{1+[R_{C1}C_{C1}+(R_{C2}+R_{C1}-\frac{1}{g_{m3}})C_{C2}]s+\frac{(1+g_{m2}R_{C2})g_{m3}R_{C1}-1}{g_{m2}g_{m3}}C_{C1}C_{C2}s^2}{(1+\frac{s}{p_1})[1+(R_{C2}+\frac{1}{g_{m2}}-\frac{1}{g_{m3}})C_{C2}s+\frac{1-g_{m2}R_{C1}}{g_{m2}g_{m3}}C_LC_{C2}s^2]} \tag{5-8}$$

$$p_1=\frac{1}{r_{o1}g_{m2}r_{o2}g_{m3}r_{o3}C_{C1}} \tag{5-9}$$

其中g_{m1}、g_{m2}和g_{m3}分别为对应电路级的跨导，r_{o1}、r_{o2}和r_{o3}分别为对应电路级的输出电阻。也就是说g_{m1}主要受M1 和M2 的尺寸决定，g_{m2}主要受M6 和M7 的尺寸决定，而g_{m3}主要受M11 和M12 的尺寸决定。

也就是说在直流增益方面，和我们之前的分析是一致的，M1 和 M2、M6 和 M7、M11 和 M12 均有放大作用，其中 M11 的放大作用相对较小。M1 和 M2 是成对的晶体管，M6 和 M7 也是成对的晶体管，电路的主要放大作用集中在 M6 和 M7 以及 M12 上。

在主极点方面，M6 和 M7、M11 和 M12 的尺寸对电路的主极点有比较强烈的负相关作用，而 M1 和 M2 对电路的主极点几乎没有影响。

考虑到篇幅的限制，本文仅选取 M1-M2 对、M6-M7 对、M11 和 M12 作讨论和分析，其他晶体管因为主要的作用是辅助和设定偏置电流，故不作深入讨论。

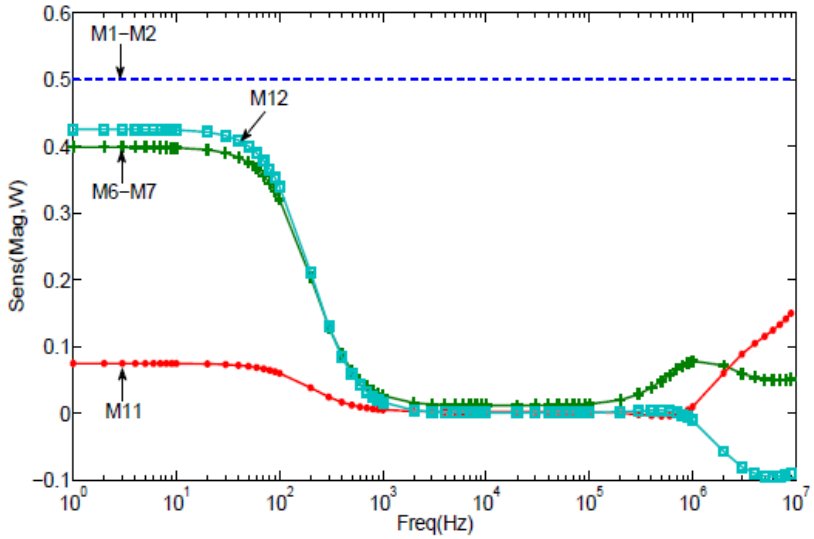
5.2.2 符号化CMOS晶体管尺寸敏感度

对于本案例电路的传输函数关于 CMOS 晶体管尺寸的幅频响应和相频响应的敏感度，我们仍采用本文上述的办法进行求解。图 5-9 给出了求解获得的考虑匹配的敏感度曲线。

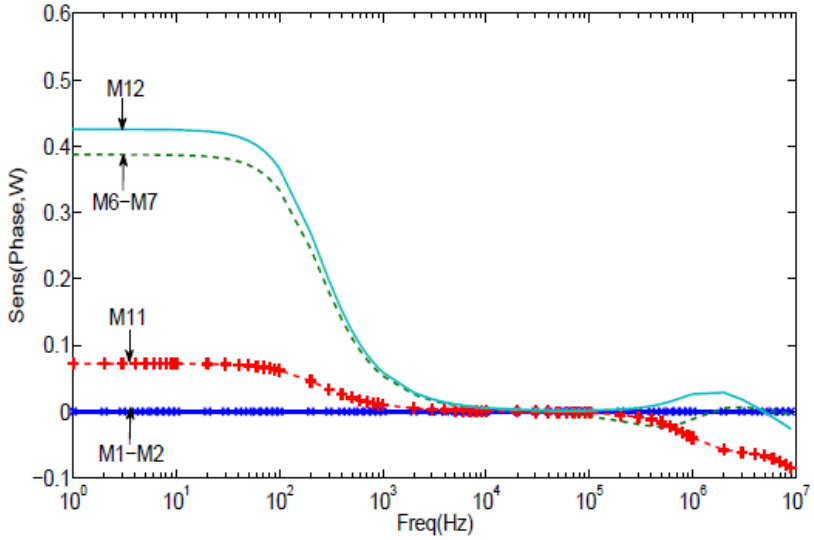
从图 5-9 中，我们可以发现曲线给出的信息和上一节中的电路理论分析是基本一致的。首先对于直流增益来说，M1-M2、M6-M7、M11 和 M12 都有贡献，M11 的贡献相对较小。地位上 M1-M2、M6-M7 和 M12 差不多是同等地位的。由于从 M6 开始，电路连接的是差分输入级的单一支路，因此 M6-M7 和 M12 的作用相对 M1-M2 来说要小一些。

其次在主极点问题上，正如上面的分析中描述的那样，M1-M2 并没有在其他晶体管出现第一个拐点的地方也出现拐点，说明 M1-M2 对于主极点没有什么影响力，实际上从图中可以得到本文上面分析以外的额外信息，那就是 M1-M2 对于分布在图示的频率区域的零极点都没有什么影响力。这里的 M1 和 M2 的行为和图 5-1 中的 M1 和 M2 的行为完全一致，这说明对模拟电路来说，标准运算放大器结

构的差分输入级对电路指标，比如直流增益和主极点的影响是一致的。他们都对直流增益有正作用，但是对主极点不起影响力。除了 M1-M2 外，其他晶体管或者晶体管对的曲线都在 100Hz 到 1000Hz 之间出现拐点，放大图片后可以发现，这个拐点，也就是值为初始值一半的点，所对应的频率在 200Hz 左右，200Hz 不到一点，而用 HSpice 数值仿真工具求出来的主极点大小是 198.8032Hz，基本上通过对符号化曲线肉眼的直接观察和实际的数值计算结果很是接近。另外从敏感度的



(a) Gain sensitivities.



(b) Phase sensitivities.

图 5-9 考虑了配对的晶体管的传输函数关于尺寸的符号化敏感度（图 5-8）

Fig.5-9 Symbolic sensitivity for transistors considering match: M1-M2, M6-M7 (Fig. 5-8)

值来看，因为初始值为正，而第一个零极点为主极点，根据公式(3-11a)，因此主极点关于所有晶体管和晶体管对的尺寸的敏感度的值为负值，也就是说所有晶体管和晶体管对的尺寸对主极点的影响为负相关。这和公式(5-9)完全一致。对于主极点之外的次极点、主零点、次零点等零极点，由于我们实现知悉零极点孰先孰后的相对顺序，因此我们无法做具体的判断，但是从每个晶体管敏感度曲线在图示频域末端的不同行为看，这个地方至少有两个零极点。而从公式(5-8)来看，这个案例电路除了一个主极点之外，还有两个次极点，一个主零点，和一个次零点。这些零极点均和M1-M2 的尺寸没有关系，但是和M6-M7 还有M11 和M12 都有关系的。基本上，如果我们对公式(5-8)中包含这四个零极点的分子和分母多项式做一元二次方程求根运算，我们就能获得具体的零极点关于具体晶体管和晶体管对的直接相互关系。但是这样的求解过程过于复杂，而且在根号存在的情况下，很难明确关系。但是依据图 5-9a，我们可以发现，M6-M7 的变化开始的比较早，因此他对最早出现的那个零极点的影响比较大，而M11 和M12 的变化比较晚，在M11 和M12 变化的时候，M6 和M7 的变化趋势也发生了变化，因此，这里又出现了一个零极点。实际上，这个时候我们观察图 5-9b的效果会更加的明显，我们会发现M6-M7 出现了两个峰峰值，第一个是峰谷，大概在 10^6Hz不到一点的地方，是一个极点或者零点，而第二个是峰顶，大概在 10^6Hz过去一点的地方，是另一个零点或者极点。而在第二个峰顶的地方，M11 出现了一个停顿的拐点，而M12 也出现了一个峰顶，这说明对这个零极点来说，M11 的影响较小，但是M12 和M6-M7 的影响相对较大。从M11 的幅频和相频相应来看，M11 对这两个零极点之后的三个零极点的影响比较大。从HSpice的数值分析结果看，在图 5-9 所示的频率范围内，这个电路此时有一个极点位于 945003.8Hz，有一个零点位于1208900Hz，而另两个零极点则在很远的地方，同我们上面的分析完全一致。不过至于如何根据图示的波形分辨是零点还是极点，我们就得借助正常的频率响应的幅频和相频曲线了。结合正常的幅频和相频曲线后，如图 5-10 所示，我们就很容易区分零点和极点的位置了。我们可以看到图 5-10 中的相频响应在 10^6Hz附近遇到了一个平坡，然后继续减小。这里表征先是遇到了一个极点，而后有一个零点出现抵消了这个极点的效果。由于这个零极点对挨的太近，出现了相消的情况，因此反映在幅频曲线上我们就几乎看不到变化。所以，从图 5-9a中的波形我们也可以预见该波形的波峰或者波谷处存在零极点相消的情况。

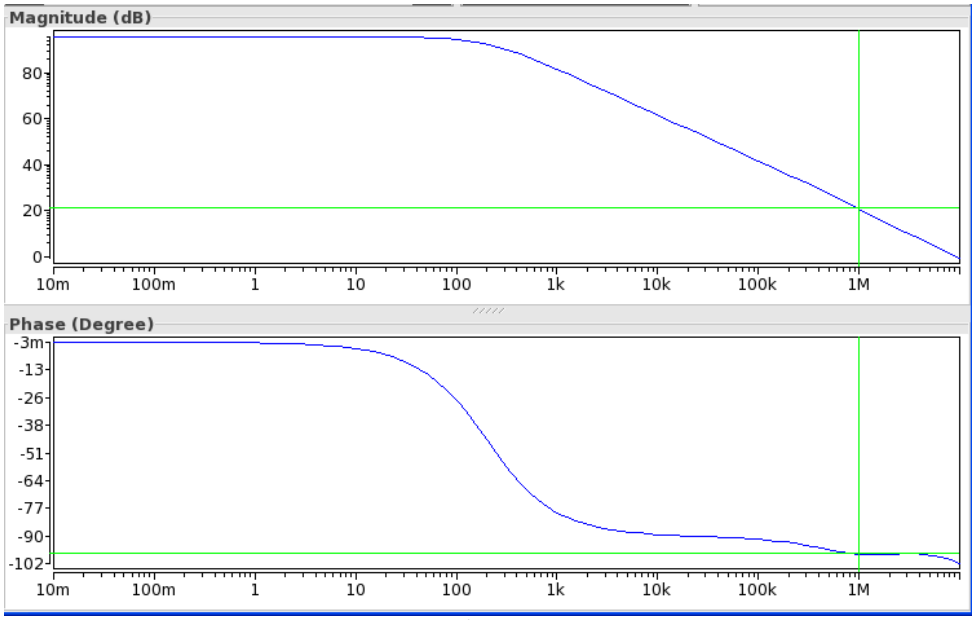


图 5-10 电路的频率响应曲线（图 5-8）
Fig.5-10 Frequency response of circuit (Fig. 5-8)

5.2.3 符号化主极点和其敏感度

类似于我们在案例电路 1 中所作的，针对这个电路，我们也可以采用符号化的方法求解他的主极点同主极点关于各个晶体管或者晶体管对的敏感度值。表 5-2 给出了相关的求解数据。

表 5-2 符号化主极点及主极点关于各晶体管尺寸敏感度（图 5-8）
Table 5-2 Symbolic dominant pole & related sensitivity (Fig. 5-8)

项目	符号化方法
主极点P_d (Hz)	198.189
Sens(P_d,$W_{1,2}$)	0.000000
Sens(P_d,$W_{6,7}$)	-0.387187
Sens(P_d,W_{11})	-0.072286
Sens(P_d,W_{12})	-0.424609

从表 5-2 中，我们发现符号化方法求得的数据同我们之前的分析仍然完全一致，并且很好的同敏感度曲线进行了呼应。因此，具体的分析我们就不在做具体介绍了。

5.2.4 符号化多设计目标的有效域

根据公式(5-8)，我们可以发现这个电路设计案例除了主极点外，两个次极点和两个零点主要决定于反馈回路中的两个电容和两个电阻的取值。因此，这个电

路其实是为了获得更好的次极点和零点和设计的。那么如何通过调整电阻电容的值至合理的值来获得合适的设计指标呢？本文介绍的符号化模拟电路设计解探索平台的三维图形化界面是这个设计的最佳选择。

设计者可以选定两个电阻和两个电容中的任意两个线性元件作为两个坐标轴，然后采用本文前述的方法，获得符合设计要求的有效数据域；再然后再次选择另外的两个线性元件作为新的两个坐标轴，同样重复原来的方法，获得一个符合相同设计要求的有效数据域；重复以上方法，直到完成所有需要的线性元件组合，最后对所有求得的数据域进行交集操作，就得到了设计指标下所有符合设计要求的有效解域了。

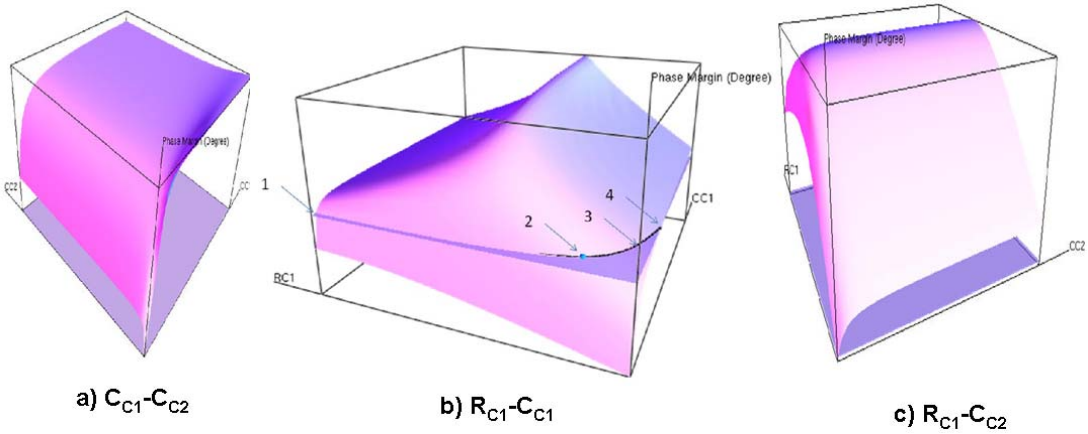


图 5-11 多参数多设计目标有效域求解示例（图 5-8）

Fig.5-11 Valid region example of multiple parameters & multiple design targets (Fig. 5-8)

图 5-11 给出了这个方法的一个例子。例子分别选择了C_{C1}-C_{C2}、R_{C1}-C_{C1}和R_{C1}-C_{C2}作为三组双坐标。所采用的都是相位裕度的设计指标。

5.3 本章小结

本章主要介绍了两个实际的设计案例使用本文描述的符号化模拟电路解探索平台 SPADE 工具的设计方法和分析。通过对案例的充分说明和分析，本章一方面展示了 SPADE 工具在提供模拟电路设计者丰富的设计辅助支持和电路信息以及关键器件参数方面的强大能力，另一方面也阐释了采用 SPADE 工具对于快速、高效的完成电路设计，收敛设计方向的巨大作用。SPADE 工具提供给模拟电路设计人员的是一个全新的设计概念和设计内容，他可以借助对传输函数关于 CMOS 晶体管尺寸参数和电路中其他线性元件值的敏感度在频域的幅频和相频分布（或者实部和虚部分布）的各种解读，提供设计人员传统设计方法学无法提供的设计信

息和设计理念。借助符号化快速求值的能力，SPADE 可以提供用户引导下的实时的设计半自动化的设计流程和设计方法学，使得电路的设计变得更加的轻松和直接。SPADE 工具无论是对模拟电路设计初学者还是对模拟电路设计经验丰富的设计人员都有很大的帮助，尤其在理解电路特性和对关键器件参数的挖掘中。一旦这个方法的涵盖面逐渐丰富和全面，这将是一个很有未来发展的工具和设计平台。

第六章 不同工作区符号化敏感度讨论

本文在一开始就做出了本文所提供的敏感度符号化方法在求解电路传输函数关于 CMOS 晶体管尺寸敏感度的时候的两个假设，也即：

a) 所有的晶体管都工作在饱和区。

b) 所有的晶体管的偏置都采用的是电流偏置。

当然，其实还有一个潜在的假设，那就是所有的晶体管的尺寸敏感度都仅和晶体管本身相关。

本章的内容旨在就上面提到的两个假设做相关介绍和讨论，一方面论证这两个假设的必要性，另一方面将本文作者在这个问题上的理解和搜集到的资料做整理，希望能给需要帮助的人提供有用的信息和参考。

6.1 CMOS晶体管工作区和偏置设置

基本上，传统的 CMOS 模拟集成电路设计课程在教授的时候，总是会选择先从晶体管的工艺结构和特性开始讲起。我们从任何一本教科书上都可以了解到以下两个关于 CMOS 晶体管工作区和偏置设置的基本信息：

CMOS 晶体管在不同栅、源、漏和衬底电压下，会分别处在不同的工作区：截止区、线性区（电阻区）和饱和区（其实还有速度饱和的情况）。

大部分的晶体管除了自偏置之外，要么被按照电流偏置进行设定（也就是I_D值不变），要么被按照电压偏置进行设定（也就是V_G或者V_{ov}的值不变）。

对于晶体管所处的三个主要工作区，我们可以参照图 6-1 所示，给出具体的小信号模型中的线性元件同晶体管尺寸的对应公式，这里我们以NMOS管对应图3-1b的小信号模型中的跨导g_m为例，在不考虑晶体管的二级效应的情况下有：

截止区：$g_m = 0$ (6-1a)

线性区：$g_m = \mu_n C_{ox} \frac{W}{L} V_{DS}$ (6-1b)

饱和区：$g_m = \mu_n C_{ox} \frac{W}{L}(V_{GS} - V_{TH}) = \sqrt{2\mu_n C_{ox} \frac{W}{L} I_D} = \frac{2I_D}{V_{GS} - V_{TH}}$ (6-1c)

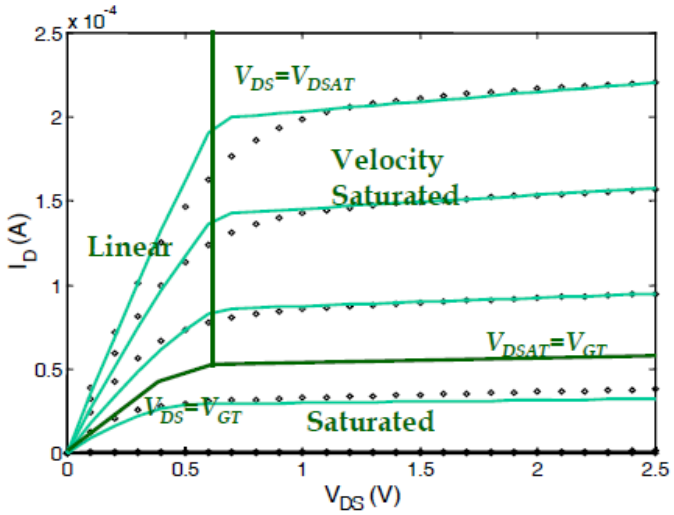


图 6-1 NMOS晶体管工作区划分[40]
Fig.6-1 Working regions of NMOS transistors[40]

因而，显然对于晶体管的不同工作区域，我们就应该采用不一样的公式来符号化的更新 CMOS 晶体管小信号模型中的线性元件的值，而由于更新线性元件的值的操作只是改变电路二叉决定图中的符号结点所对应的参数值，并不需要改变二叉决定图本身的结构，目前来说，是一件很容易的事情，只需要针对不同的工作区进行不同公式的分类讨论即可。

然而问题是，如何判别一个晶体管所在的工作区。传统的模拟电路理论采用的都是利用晶体管的直流偏置信息，也就是V_{DS}、V_{GS}、V_{TH}、V_{BS}、I_D等的相互关系，也即对NMOS晶体管来说，有：

截止区：$V_{GS} < V_{TH}$ (6-2a)

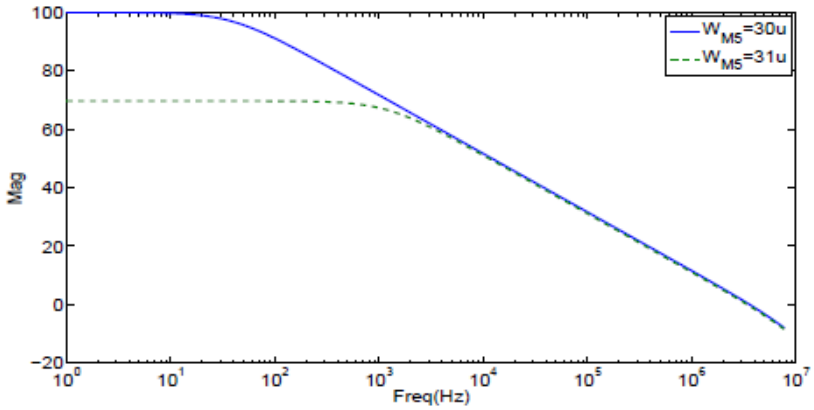
线性区：$0 \le V_{DS} \le V_{GS} - V_{TH}$ (6-2b)

饱和区：$0 \le V_{GS} - V_{TH} < V_{DS}$ (6-2c)

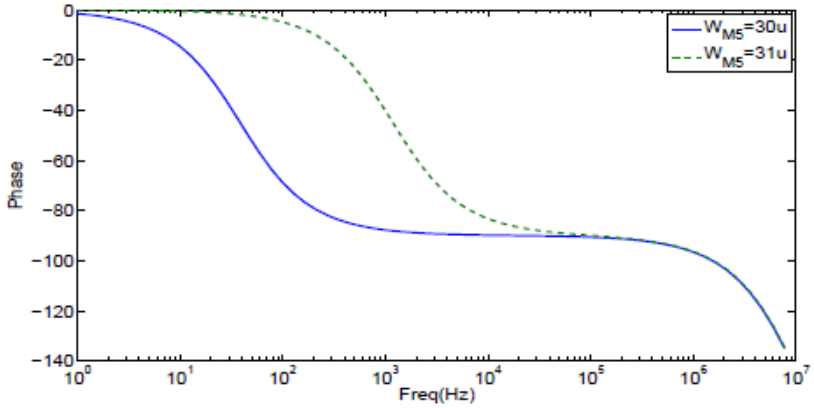
而这本身也就是各个工作区的定义。

然而对于这些直流偏置信息的获得，传统的数值仿真器采用的是迭代求解的方式，因此不存在相关的问题。但是对于符号化仿真器，在不依靠数值仿真器求解的情况下，是无法准确这些直流量的获得相关的值的，究其原因，恰恰是因为我们无法确知一个晶体管采用的是电流偏置还是电压偏置，而这本质上恰好是模拟电路设计复杂性的所在。在对一个晶体管进行尺寸设计的时候会因为新的尺寸

破坏了原有电路结构中存在内在联系的晶体管而导致诸如电流失配等问题，进而导致一个晶体管的偏置方式从电流偏置进入到电压偏置或者相反。这也正是一开始的假设中提出来的第三条假设，晶体管间的独立性的原因。诚然这个问题可以通过由电路的设计人员根据经验给出晶体管之间的内在联系和耦合关系进行解决，但是对于复杂的电路设计，显然，再有经验的人也会显得无能无力，而这正好是 EDA 工具所应该接管的事情。对于电路结构的自动识别是不可避免的，但这需要很多的经验算法和可能的自学习的过程，这个问题是本文遗留下来的一大问题。图 6-2 给出了针对本文中的案例 1（也就是图 5-1 的电路）的一个独立性导致问题的例子。在图 6-2 中，虽然晶体管 M5 的尺寸仅仅是增加了 1μ，而且新的值下的所有晶体管仍然在饱和区，但是电路的特性已经发生了变化，由于电路电流失配的原因，直流增益大大降低，整个电路的主极点严重右移。如果对 M5 的尺寸继续增加 0.5μ，那么 M5 就将会进入线性区。



(a) Gain sensitivities.



(b) Phase sensitivities.

图 6-2 晶体管独立性的例子（图 5-1，M5）

Fig.6-2 An example of dependency of transistors (Fig. 5-1, M5)

现在抛开晶体管的独立性问题，让我们看看在不同的偏置假定下，晶体管小信号模型线性元件参数值会受晶体管直流参数的影响有多大。

我们仅就饱和区的公式(6-1c)进行讨论。当我们假定电流偏置的时候，我们可以看到，跨导g_m是和晶体管宽长比成带根号的正比例关系，而当我们假定电压偏置的时候，跨导g_m则是和晶体管宽长比成正比例关系。同样的公式再反过来进行直流参数的计算的时候，就会引发问题，究竟什么样的公式才能真实的表示实际的电路情况。这个问题将会在下一节中具体给出。

这里还有一个问题，那就是工作区切换的时候从一个工作区到另一个工作区，即便不存在偏置假定的问题，也存在信息量不对称的问题。符号化的工具无法从电路的截止状态$g_m=0$ 的信息中通过单纯的公式计算就能够获得相关的直流量的更新信息，并判定电路进入了线性区或者饱和区。虽然反向的切换（也就是从饱和区到线性区、截止区；从线性区到截止区）信息是足够充分的。这正是在本文介绍的符号化模拟电路解探索平台中必须要有一个直流工作点求解器的原因。当然，现在的直流工作点求解器采用的HSpice工具，因此每次求解完需要重新构建二叉决定图；一旦内嵌的直流工作点求解器完成，那么无论上述的问题如何解决，二叉决定图的构建仍然只需要完成一次。这是符号化求解方法真正的优势。

6.2 不同工作区和偏置设置的敏感度求解的讨论

在符号化敏感度求解的问题上，有过很多不同的尝试和理解。很多求解手段和方法恰恰是对工作区和偏置设置问题的本质没有很好的理解，而发生了错误。本节希望能够通过对一些错误方法的论述，进一步向读者明确工作区和偏置设置假设的重要性和不可获取性。

论文[36-38]中提出了采用参数系数图(ECD, Element Coefficient Diagram)的方法进行电路敏感度求解的思路。参数系数图本质上也是符号化方法的另一种形式，采用对传输函数的符号化表示，求解传输函数关于CMOS晶体管尺寸的敏感度，并用敏感度引导自动化的设计。论文中所使用的CMOS晶体管小信号模型和本文图 3-1b的小信号模型基本上是一致的。作者在论文[37]中列出了他用以更新小信号模型参数在不同工作区的公式，对于像g_m这种重要的直流参数相关的线性元件，作者似乎直接引用了经典公式中的某些公式进行求解。实际上，在作者给出的公式中，直流参数的典型值被当做一个常数直接用以计算，这种情况就相当于我们在求解的时候做了一个电压偏置的假设，而且作者一视同仁的对饱和区和线性区

的情况同等对待，根据本文上面的分析，这已经隐含了一个错误。而论文中引用的电路设计案例，恰好是本文的第一个设计案例，也就是图 5-1 的电路，因而这个假设完全脱离了这个电路本身的特性。图 6-3 给出了使用论文中的公式推导求解获得的进入线性区的M5 的敏感度的值同HSpice的比较，而图 6-4 给出了使用本文假设下的公式求解的进入线性区的M5 的敏感度的值同HSpice的比较。

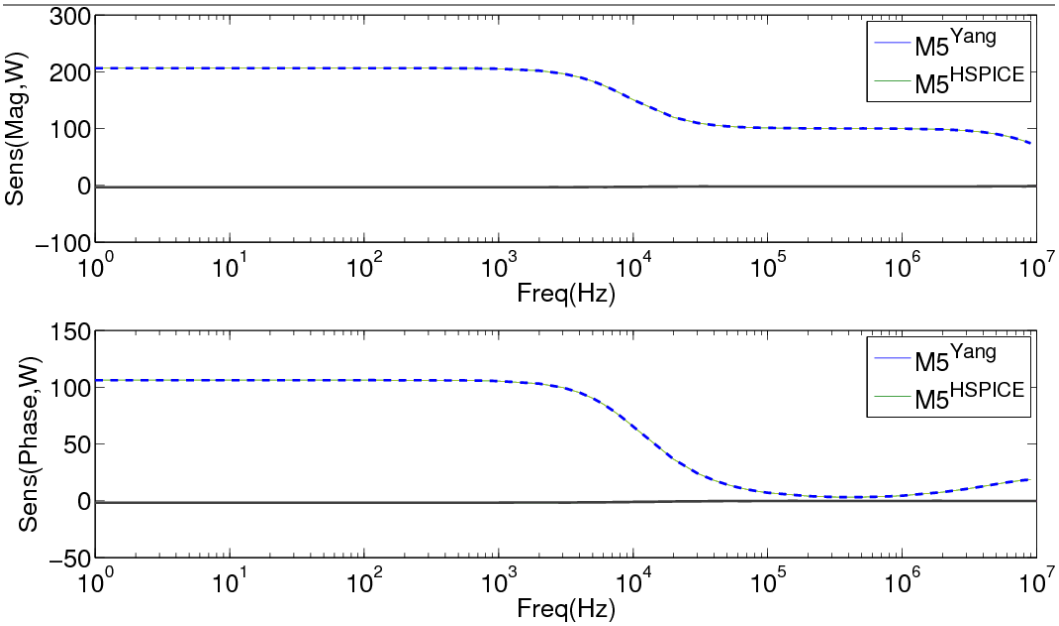


图 6-3 [37]中的公式求解的线性区敏感度同 HSpice 的比较（图 5-1，M5）

Fig.6-3 Linear sensitivity by [37] VS HSpice (Fig. 5-1, M5)

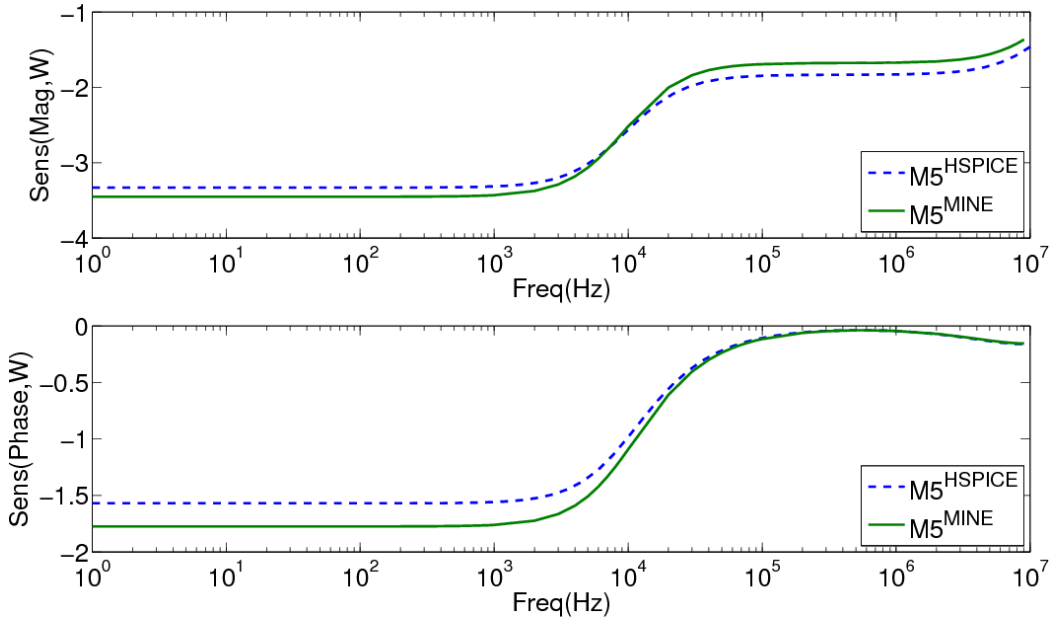


图 6-4 本中的公式求解的线性区敏感度同 HSpice 的比较（图 5-1，M5）

Fig.6-4 Linear sensitivity by SPADE VS HSpice (Fig. 5-1, M5)

从图中我们可以清楚地看到，假设错误所带来的完全错误，因为论文中并没有论及同 HSpice 或者类似标准求解器的比较结果，因而可能作者也不知道自己的

假设已经发生了错误。另外从论文中可以发现，他对于图 5-1 的案例电路的自动

化设计的增益指标仅仅是 40dB，作为两级运算放大器，40dB 的增益说明，必然有晶体管处在线性区，也就是说这样的自动化设计完全没有实际的意义和价值。

另外，现在也有人提出使用g_m的进行自动化设计的方案，也就是说，不提供设计者晶体管的直接尺寸的敏感度和自动化设计流程，而是提供基于晶体管跨导这一重要参数的敏感度和自动化设计流程。这个方案的好处是避免了关于电流偏置还是电压偏置的讨论。但是问题仍然没有解决。一旦晶体管在不同的工作区之间切换，那么仍然无法通过纯符号化的方法获悉切换的瞬间。另外没有g_m和晶体管尺寸的关联，也使得对设计人员的参考意义大大下降。

6.3 本章小结

本章主要做了关于在符号化求解电路传输函数关于CMOS晶体管尺寸的敏感度问题中的假设问题的讨论和分析，强调了关于工作区、直流参数偏置和晶体管耦合独立性假设的重要性和必要性。并通过一个实际的例子对现在的研究中存在的对这一系列问题认识不清、理解不深而导致的问题加以说明，意在强调本章所讨论的问题将会成为今后符号化模拟集成电路设计自动化最为重要的一个问题。就目前来看，以上几个问题的解决并没有非常成熟的方案，尤其是在对工作区的判断问题上，都仍处在探索阶段。因此结合内嵌的直流工作点求解器的混合方案是一个理论上可行并具有一定可操作性的方案，可以值得一试。

第七章 结论与研究展望

7.1 总结

本文主要介绍了基于符号化方法的一系列理论研究和相关应用实例。本文通过对基于符号化电路图约化算法的符号化仿真器 GRASS 的拓展和优化，完成了符号化传输函数关于CMOS晶体管尺寸的敏感度的幅频和相频响应的研究和算法

开发，在开发的过程中，通过观察产生波形结果，发现了敏感度曲线对于揭示电路零极点特性的潜在作用，并从理论上做出了证明。同时本文结合符号化仿真器原有的接口和模拟集成电路设计自动化流程，开发了一个适合模拟电路设计人员快速完成电路设计和电路收敛的符号化模拟集成电路设计解探索平台工具SPADE。工具通过脚本和程序的整合，给设计者提供了丰富的电路信息和辅助支持，引导设计人员更好更快的找到电路设计指标对应的关键元件和设计方向，帮助设计人员进行半自动化的设计。本文通过给出两个电路设计的实例，用以说明这个平台工具的功能和价值，也进一步说明从符号化敏感度数据中揭示出来的电路潜在的零极点信息，以及设计人员如何利用这些信息完成设计的方法。

本文从符号化的历史开始，通过对符号化仿真器的原理和实现的描述和介绍展开文字。最后又将讨论拉回到问题最开始的假设上去。通过对假设的进一步说明和分析，并通过一个实际的例子，展示了假设的重要性和必要性，从而清晰的指出了符号化仿真工具和设计平台的研究方向和难点。使得整篇论文的结构完整、内容清晰。符号化的方法之于模拟集成电路设计的应用和模拟集成电路设计自动化的最终实现必然将不久的将来得以实现。

7.2 研究展望

本文所论述的符号化敏感度求解算法和符号化的模拟电路设计解探索平台是符号化模拟电路设计自动化的一个开始，一个原型性实验。

在这个研究展开的过程中碰到了很多问题，有的已经被解决了，而有的则被以假设的形式暂时搁置在一旁。然而在实现完全自动化的道路上，这些问题无论如何都是要被解决的。而目前的假设所带来的问题在本文上一章节中做过详细的描述，这里就不在做详细论述。但是关于CMOS晶体管工作区、电路直流偏置信息的识别和判断以及CMOS晶体管之间的耦合性问题将会作为一个长期而实际的问题遗留下来，需要新的研究来解决。

作为符号化的模拟电路设计自动化的解决方案的研究，有下述几个方面可以展开：

符号化方法对电路的求解能力。符号化方法的优点很多，但是其缺点也很明确，那就是本身对内存的开销。为了增加符号化方法对电路的处理能力，层次化的方法是一种解决方案，关于元件的有效分组以获得更好的符号顺序是另一种解决方案。几种解决方案都要结合起来，才能最大限度的降低符号化的方法的限制。另外，采用不同精度的宏模型和模型优化也是一个办法。

数值-符号化方法的混合解决方案。这是用以解决本文上一章节描述的关于CMOS 晶体管工作区、直流偏置参数和晶体管依赖性的办法。关键是处理好两个方法之间的接口和衔接，以及在何时使用数值的方法进行直流分析。

模拟电路设计自动化的方法学。模拟电路的自动化设计方法学至今还是各自为政的状态，之所以没有被统一到一个标准化的流程上，主要的问题就是大家的方法学都有各自的优点，也有各自明显的缺点和不足，这些问题直接制约了某个方法成为方法学的可能。因此开发一套合理有效、经得起具体细节考验的方法学是非常重要的工作。

多零极点的分析方法。本文关于零极点的描述大多集中在主极点的讨论上。主要的原因还是除主极点之外的其他零极点，我们无法知悉他们的先后顺序和排列组合。尤其是当存在零极点离得特别近或者存在共轭零级点的情况。因此关于除主极点外的零极点分析方法的开发和研究就变得非常有价值了。一旦这方面的工作得以完成，某种意义上说，敏感度的方法就可以取代根轨迹图等传统的零极点分析方法了。

电路模块化识别和模型优化。对于电路的模块化识别和模型优化既是降低符号化的内容限制的办法，又是提升仿真精准度的方案，同时还可以解决电路中直流偏置参数的问题。这类问题的解决基本上可以同时清除很多相关联的问题，是非常值的进一步探索的问题。

实际上，本文在撰写的同时，我们实验室已经展开了上述几个问题中的部分问题的研究和探索工作[39]，相信他们经过自己的努力，一定会攻克其中的几个问题，将符号化的模拟电路设计自动化的研究进一步往前推进一个台阶。

参 考 文 献

[1] P. Wambacq, G. Gielen, and W. Sansen, "A cancellation-free algorithm for the symbolic simulation of large analog circuits," in Proc. Int'l Symposium on Circuits and Systems, pp. 1157-1160, 1992.

[2] L.W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Ph.D. dissertation, Univ. California, Berkeley, CA, May. 1975.

[3] G. Gielen, P. Wambacq, and W. Sansen, "Symbolic analysis methods and applications for analog circuits: A tutorial overview," Proceedings of the IEEE, vol. 82, pp. 287-303, Feburary, 1994.

[4] P. Sannuti and N. N. Puri, "Symbolic network analysis - an algebraic formulation," IEEE Trans. Circuits Syst., vol. CAS-27, pp. 679-687, August. 1980.

[5] P. Wambacq, G. Gielen, and W. Sansen, "Symbolic network analysis methods for practical analog integrated circuits: a survey," IEEE Trans. on Circuits and Systems – II: Analog and Digital Signal Processing, vol. 45, no. 10, pp. 1331–1341, 1998.

[6] W. Sansen, G. Gielen, and H. Walscharts, "A symbolic simulator for analog circuits," in Proc. ISSCC, 1989, pp. 204-205.

[7] G. Gielen, H. Walscharts, and W. Sansen, "ISAAC: A symbolic simulator for analog integrated circuits," IEEE J. Solid-State Circuits, vol. 24, no. 6, pp. 1587-1597, December. 1989.

[8] F. Femandez, A. Rodriguez-Vazquez, and J. Huertas, "A tool for symbolic analysis of analog integrated circuits including pole/zero extraction," in Proc. ECCTD, 1991, pp.752-761.

[9] J. Kluwer "Interactive ac modeling and characterization of analog circuits via symbolic analysis," Analog Integrated Circuits and Signal Process., vol. l, pp. 183-208, November, 1991.

[10] S. Seda, M. Degrauwe, and W. Fichtner, "A symbolic analysis tool for analog circuit design automation," in Proc. ICCAD, 1988, pp. 488-491.

[11] "Lazy-expansion symbolic expression approximation in SYNAP," in Proc. ICCAD, 1992, pp. 31C317.

[12] S. Manetti, "New approaches to automatic symbolic analysis of electric circuits," Proc. Inst. Elec. Eng. pt. G, pp. 22-28, February, 1991.

[13] G. Wierzba et al., "SSPICE-A symbolic SPICE program for linear active circuits," in Proc. Midwest Symp. on Circuits and Systems, 1989.

[14] A. Konczykowska and M. Bon, "Automated design software for switched-capacitor IC's with symbolic simulator SCYMBAL," in Proc. DAC, 1988, pp. 363-368.

[15] M. Hassoun and P. Lin, "A new network approach to symbolic simulation of large-scale networks," in Proc. ISCAS, 1989, pp. 806-809.

[16] L. Huelsman, "Personal computer symbolic analysis programs for undergraduate engineering courses," in Proc. ISCAS, 1989, pp. 798-801.

[17] J. Starzyk and A. Konczykowska, "Flowgraph analysis of large electronic networks," IEEE Trans. on Circuits and Systems, vol. CAS-33, no. 3, pp. 302-315, 1986.

[18] G. Minty, "A simple algorithm for listing all the trees of a graph," IEEE Trans. on Circuit Theory, vol. CT-12, p. 120, 1965.

[19] F. Fern´andez, A. Rodr´ıguez-V´azquez, J. Huertas, and G. Gielen, Symbolic Analysis Techniques – Applications to Analog Design Automation. New York: IEEE Press, 1998.

[20] L. R. Carley, D. Garrod, R. Harjani, J. Kelly, T. Lim, E. Ochotta, and R. A. Rutenbar, “ACACIA: The CMU analog design system,” in Proc. IEEE Custom Integrated Circuits Conference, 1989.

[21] K. Swings and W. Sansen, “ARIADNE: A constraint-based approach to computer-aided synthesis and modeling of analog integrated circuits,” Analog Integrated Circuits and Signal Processing, vol. 3, pp. 197–215, 1993.

[22] J. Vlach and K. Singhal, Computer Methods for Circuit Analysis and Design. New York, NY: Van Nostrand Reinhold Company, 1983.

[23] S. B. Akers, "Binary decision diagrams," IEEE Trans. Comput., vol. C-27, pp. 509-516, June, 1976.

[24] C. Lee. Representation of switching circuits by binary-decision programs. Bell System Technical Journal, 38:985-999, July, 1959.

[25] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. Comput., vol. C-37, pp. 677-691, August, 1986.

[26] K. Brace, R. Rudell, and R. Bryant, "Efficient implementation of a BDD package," in Proc. 27th IEEE/ACM Design Automation Conference, pp. 40-45, June, 1990.

[27] S. Minato, "Zero-suppressed BDD's for set manipulation in combinatorial problems," in Proc. 30th IEEE/ACM Design Automation Conf., (Dallas, TX), pp.272-277, 1993.

[28] C.-J. Shi and X.-D. Tan, "Canonical symbolic analysis of large analogcircuits with determinant decision diagrams," IEEE Trans. on Computer-Aided Design, vol. 19,

no. 1, pp. 1-18, January, 2000.

[29] X. Tan and C.-J. Shi, "Hierarchical symbolic analysis of large analog circuits with determinant decision diagrams," in Proc. IEEE Int. Symp. Circuits and Systems, vol. VI, pp. 318-321, May, 1998.

[30] C.-J. Shi and X. Tan, "Symbolic analysis of large analog circuits with determinant decision diagrams," in Proc. IEEE/ACM Int. Conf. Computer-Aided Design, San Jose, CA, pp. 366-373, November, 1997.

[31] W. Chen, G. Shi, "Implementation of a symbolic circuit simulator for topological network analysis," in proceedings of IEEE Asia Pacific Conference on Circuit and System 2006, Singapore, pp.1327-1331, Dec. 2006.

[32] G. Shi, W. Chen and C.-J. Shi, "A graph reduction approach to symbolic circuit analysis," Asia South-Pacific Design Automation Conference, Yokohama, Japan, pp. 197-202, Jan. 2007.

[33] 陈微微. 符号化模拟电路仿真器的实现与应用[硕士论文]. 上海: 上海交通大学. 2007.

[34] G. Shi and X. Meng, "Variational analog integrated circuit design via symbolic sensitivity analysis," International Symposium on Circuits and Systems, Taiwan, pp. 3002-3005, May. 2009.

[35] 李骥. 模拟电路符号化仿真器（GRASS）图形化用户界面开发与电路设计案例[硕士论文]. 上海: 上海交通大学. 2009.

[36] H. Yang, M. Ranjan, W. Verhaegen, M. Ding, R. Vemuri, and G. Gielen, “Efficient symbolic sensitivity analysis of analog circuits using elementcoefficient diagrams,” in Proc. Asia South-Pacific Design Automation Conference (ASPDAC), Yokohama, Japan, Jan. 2005, pp. 230–235.

[37] H. Yang, A. Agarwal, and R. Vemuri, “Fast analog circuit synthesis using multi-parameter sensitivity analysis based on element-coefficient diagrams,” in Proc. IEEE Computer Society Annual Symposium on VLSI, Tampa, Florida, USA, 2005, pp. 71–76.

[38] H. Yang，” Symbolic Sensitivity Analysis Techniques and Applications in Analog Circuit Synthesis,” Ph.D. dissertation, Univ. Cincinnati, Dec. 2006.

[39] H. Xu, G. Shi, and X. Li, “Hierarchical exact symbolic analysis of large analog integrated circuits by symbolic stamps,” in Prof. Asia South-Pacific Design Automation Conference (ASPDAC), Yokohama, Japan, 2011, to appear.

[40] A. S. Sedra and K. C. Smith, Microelectronic Circuits (4th edition). New York: Oxford University Press, 1998.

[41] G. Palumbo and S. Pennisi, “Design methodology and advances in nested-Miller

compensation," IEEE Trans. Circuits and Systems - I: Fundamental Theory and Applications, vol. 49, no. 7, pp. 893–903, 2002.

致　谢

时光如梭，岁月荏苒，转眼间，居然已经到了研究生快毕业的时候了。细细算来，自己已经在交大整整呆了 6 个年头，很快第 7 个年头就要到来了。在这充实的 6 年的时光里，我从一个对微电子什么都不懂的学生，逐渐变得对微电子这个行业懵懵懂懂，碰到了很多的挫折，也得到了很多人的帮助。正是在这一路走来的好心人帮助下，我才走到了现在。在这个即将毕业的时刻，也正好是西方感恩节快要到来的时刻，我想我真的要好好向对我做出过帮助的人表达谢意。

首先，我要感谢国家和微电子学院能够给我这样一个机会接触微电子，进入微电子学院学习，就读本科和研究生。我们是学院成立的第一届本科生，那个时候交大的微电子学院刚刚成为中国的 9 大微电子基地，我们怀揣着对未来的美好梦想，和对微电子这个行业的向往和好奇，踏进了交大和微电子学院。

然后，我一定要好好谢谢我的父母亲，没有他们对我的“放任自流”就不会有我这一路走来的无悔无憾。父母亲在我个人的发展和学业问题上从来不给我施压，他们一直坚持我的未来要我自己把握，我的人生也要我自己去选择的理念，支持并鼓励我坚定自己的理想和选择。一步步走来，父母亲的年岁增长了许多，白发也一根根、一丛丛的爬了上去，但是他们对我的宽容和默默的支持从来没有减少。我想，或许现在是目前我能对他们的这种教育方式的一种这好的感谢吧。

当然，我一定一定要感谢我的导师施国勇教授和指导我模拟电路设计的李章全老师的。没有施老师一步步的耐心引导和共同讨论，我是不可能做到现在这个课题上来的，也是不可能完成到现在的这个程度的。我相信没有施老师的指导的现在的我，一定还在研究课题的苦海中苦苦挣扎。施老师总是能够在我的研究方向和研究内容陷入泥沼的时候伸出手拉我一把，帮助我摆脱无路可走的窘境，无论是之前的时钟树综合、三维芯片制造、高层次综合还是现在的模拟集成电路设计的符号化平台，施老师不仅帮我指引方向，还推着我走向那个方向。施老师不知道帮我在研究问题上搜过多少篇论文，现在我的电脑里面还存放着很多很多的论文，而这些工作原本就应该是我自己干的事情。施老师在我研究陷入困难的时候纠正我的错误，让我一步一个脚印的做尝试、做实验，正是施老师的这种耐心才使我发现了本来毫无概念的敏感度曲线所隐含的零极点关系。施老师在研究不

仅仅是简单的指导，而是亲身投入的帮助我展开，包括公式的证明，研究思路的生发，无一不致的帮助，才使得我现在勉强将这个研究课题做到了现在这个程度。真的特别感谢施老师在研究上对我的无法估价的帮助。虽然李老师对我的帮助并不是贯穿于研究生生活的始终，但是他对我在电路问题上的理解有很大的启发。李老师会为了让我深刻理解一些电路的行为和特性，设计特别的问题让我自己动手去解决，并手把手辅导我电路设计的基本知识，让我从一个模拟电路的门外汉将将踏在了模拟电路的门槛上。李老师治学的严谨让我十分的敬佩，我依然记得深夜 12 点李老师在办公室修改我论文初稿时红通通的眼睛，我想这个场景我一辈子都不会忘记。另外，李老师的为人准则和处事技巧都很值得我学习，李老师为了把问题解决，把问题搞清楚的探究精神将会是鼓舞一生的信念。

另外，在我整整 6 年的学业中，我要感谢谢憬老师、赵峰老师和祝永新教授。你们给我的帮助不仅仅是学业上的、项目上的还包括人生观、价值观上的，我真的非常感激我的人生能够遇到你们，能够得到的你们帮助、分享你们的人生，我想我对你们是感恩的心，真的。另外，我还要感谢汪辉老师，是你让我认识了金晶，也让我写了大学唯一的一份检讨；感谢何波涌教授，是你让我见识到了工业界的奇迹和努力以及一颗深藏在海龟内心的拼搏的心；感谢程秀兰老师，您的课上我真正明白了 IC 的含义；感谢刘婷老师，是你彻底让我改变了对科技文档的撰写的观念；感谢马骏教授，是您化解开了我对高层次综合问题的理解；感谢郭炜老师，虽然您不曾认得我，但是您在我大一的时候那神情的一声哭喊，让我有了激情和动力；感谢王琴教授、黄其煜教授老师、付宇卓教授、何卫峰教授、周建军教授、毛志刚教授、杨莉老师、莫婷婷老师，你们的课让我记住了你们，你们给我带来了不同的微电子的理解，是你们把我一步步领入了微电子的殿堂。我真的要谢谢你们！

还有，我要感谢学院的思政老师杨薛雯老师，是您一步步的带我展开学生工作，发掘并信任我，逐步锻炼我的能力，并在人生的发展中给了我很多的帮助和辅导，我真的特别特别感激我能遇到您这样的老师，这样的朋友。我要感谢胡薇薇老师，请原谅我们本科时候的任性和无知，您真的是很好很好的老师，我现在特别怀念您在学院的日子。感谢陆谨老师这么多年的工作，真的麻烦您好多次了。感谢任涛老师的那顿橘子聊天。感谢高延坤老师，感谢郁美娟老师、感谢李良军、石莎莎、感谢王展、还要感谢徐师傅。

在这六年的学习生涯中，我碰到了很多很好的学长和学姐，是他们帮我一点点理清问题的细节，搞清楚概念的本意，我真的要好好感谢你们！首先是实验室

的师兄师姐：郝志刚、于雪红、孟晓璇、黄世杰、刘安、李骥、陈安、谭焜元、王婷、陈硕、曾媚和谢边村，是你们引导我一点点开始研究，是你们和我不断地讨论问题，帮我理清思路，是你们帮我分析问题，找到方向，没有你们，我可能还无法理解什么事符号化，正是因为你们的帮助，我才能够做好研究。最早的数电集成电路设计助教韩晓鑫、大三生产实习的时候的向奔和王江、嵌入式设计的徐淑峰、模拟电路的胡悦和戚琦、器件的金晶、DSP 应用的祝翔羽、Verilog 设计的秦骅姜华和张智澄、工艺的黄锐、SoC 实验室的胡泊、朱礼波、李彦堃、董巍，还有赵丽丽、张衍、施跃华、王佶梁、彭光宇、郑振军、支铭杰、张晓晨、彭修宇、施展、戴煊、穆静，没有他们的全心帮助我能达到现在对微电子的程度，你们都是在我成长道路上写下重重一笔的人啊！

我还要感谢和我一起奋斗而今仍然在奋斗的 F04 和 B08 们。借此机会特别感谢郭嘉老师、崔晓、李涌伟、万江、郑诚、王汉卿、徐海宁、朱原、彭秋妍、单川、周鹤群、胡哲坤、孙海、翁朝明、孟骁、罗华庚、潘耀亮、梁天翼、曹超、李笑笑、吴俊和汪涵；我要感谢谢厉，你让我进一步懂得了人生；感谢黄伟坚，你让我不断地奋起和努力；感谢阙志强，你让我懂得了收放；感谢刘晓明，你让我懂得了沟通的重要性，也不断的挖掘我自己的内心。除此之外，还要感谢可爱而努力的学弟学妹们，林志平和张力，和你们合作是一件很开心的事情。

当坐下来细细列举要感谢的人时，才发现时间真的过的飞快，这些人曾经似乎都还近在眼前，而现在却已不知取向。真的要想电话问候一声感谢的时候，才发现这种发自内心的感恩之情无处投放，只能，通过这样的方式，在这样的只言片语中，表达我对他们的真心谢意和虔诚祝福，希望，我没有写错你们任何一个人的名字，谢谢，每一个人！

攻读硕士学位期间已发表或录用的论文

[1] Diming Ma, Guoyong Shi and Alex Lee, “A Design Platform for Analog Device Size Sensitivity Analysis and Visualization,” in Proc. Asia Pacific Conference on Circuit and System (APCCAS), Malaysia, pp. 48-51, Dec. 2010.