上海交通大学硕士学位论文

MOS 模拟集成电路交互式自动化 Sizing 符号化方法探索研究和软件实现[1]

硕 士 研 究 生：陈家俊

学　　号：1102109016

导　　师：施国勇

专　　业：电子科学与技术

所 在 单 位：微电子学院

答 辩 日 期：2012 年 12 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 61176129）资助

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

Research and Implementation on the Symbolic Methods of Interactive MOS Analog Circuit Sizing Automation

Candidate：	Chen, Jiajun
Student ID:	1102109016
Supervisor：	Prof. Shi, Guoyong
Speciality：	Electronic Science and Technology
Affiliation：	School of Microelectronics
Date of Defence：	Dec, 2012
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《基于符号化电路仿真器的模拟电路设计工具的算法研究和软件实现》，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：陈家俊

日期：2013 年 3 月 14 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：陈嘉俊　　指导教师签名：施国勇

日期：2013年3月14日　　日期：2013年3月14日

上海交通大学学位论文答辩决议书

姓　　名	陈家俊	学号	1102109016	所在学科	电子科学与技术
指导教师	施国勇	答辩日期	2012.12.19	答辩地点	微电子楼会议室
论文题目	MOS 模拟集成电路交互式自动化 Sizing 符号化方法探索研究和软件实现				

投票表决结果：3/3/3 （同意票数/实到委员数/应到委员数）　答辩结论：☑通过　□未通过

本文主要介绍了符号化电路仿真器的原理和特点，并在符号化仿真算法的基础上，实现了对电路精确的灵敏度分析的计算求解算法．并利用了符号化仿真器计算得到的灵敏度信息，尝试了对模拟集成电路（主要是运算放大器电路）进行 Size 并优化。同时，还利用了灵敏度分析的结果改进了经典的模拟退火的模拟集成电路的自动优化算法，并取得了不错的结果。

本文的主要研究内容及创新点如下：

1. 实现并改进了符号化电路仿真器 GPDD 的算法
2. 在 GPDD 算法的基础上实现了对电路的灵敏度分析，并修正了前人类似研究上的一些错误和不完整。
3. 在灵敏度分析的基础上，尝试通过电路的灵敏度信息来对电路做互动 Size，改进现有的设计方法
4. 利用灵敏度分析的结果改进了传统的模拟退火算法的效率。
5. 完成了一个简单，包含了互动设计和自动设计方法的模拟电路设计平台

该论文结构完整 ，条理清晰。答辩时，思路严密，回答问题简洁而准确。

经答辩委员会认真讨论，投票表决，通过该同学的论文答辩。建议校学位委员会授予该同学硕士学位。

答辩委员会成员签名	职务	姓名	职称	单 位	签名
	主席	付宇卓	教授	上海交通大学微电子学院	付宇卓
	委员	王国兴	副教授	上海交通大学微电子学院	王国兴
	委员	莫亭亭		上海交通大学微电子学院	莫亭亭
	委员				
	委员				
	委员				
	秘书	陆瑾		上海交通大学微电子学院	陆瑾

MOS模拟集成电路交互式自动化Sizing符号化方法探索研究和软件实现

摘 要

在如今的集成电路设计流程中，所占面积很小的模拟集成电路往往需要很长的设计时间，自动化程度较低。本文尝试在新型的符号化电路仿真器的基础上，研究实现了通过符号化灵敏度计算来优化运算放大器电路 Sizing 的方法，并完成了一个简单的运算放大器设计软件。

本文将主要介绍利用符号化仿真引擎和 EKV 器件模型，进行灵敏度分析的算法，并在得到的电路灵敏度分析的结果的基础上，尝试了使用灵敏度信息来对电路进行互动设计方法以及自动优化电路的算法（自适应模拟退火算法），比较了这些方法优化后的结果，并尝试结合它们的优点来提高模拟集成电路设计的效率。

关键词： 模拟集成电路设计自动化、符号化仿真器、灵敏度分析、互动电路设计、自适应模拟退火

RESEARCH AND IMPLEMENTATION ON THE SYMBOLIC METHODS OF INTERACTIVE MOS ANALOG CIRCUIT SIZING AUTOMATION

ABSTRACT

In nowadays' integrated circuit design, the design of analog part is rather time-consuming compared with its digital companions, although it occupies only a very small part of the area of the chip, which is far from automation. And In this thesis, we are trying to make use of the symbolic sensitivity information of the circuit and developed an OPAMP sizing tool on the basis of the emerging symbolic circuit simulator GPDD.

In this thesis, several algorithms will be introduced and implemented, including the algorithms of computing sensitivity via symbolic simulator GPDD, the methods of interactive circuit sizing and the algorithms of automatic circuit sizing. And we will compare the sizing results of different methods to reveal their pros and cons, attempting to combine their advantages to improve the efficiency of analog circuit sizing.

KEY WORDS: Analog Circuit Design Automation, Symbolic Circuit Simulator, Sensitivity Analysis, Interactive Circuit Sizing, Adaptive Simulated Annealing

目 录

图 录

表 录

第一章 绪论

本章绪论主要介绍关于模拟电路设计工具、设计方法的现状以及符号化电路仿真器的基本情况，并主要说明本文的主要的内容和结构。

1.1 模拟电路设计工具的现状及研究意义

随着集成电路产业的发展，对电路设计自动化工具的需求也日益增长。相比于已经相当成熟的数字电路的设计综合工具，模拟电路设计的自动化工具还显得十分的落后。因而在现今的芯片设计过程中，即使模拟电路仅仅占据电路相当小的一个部分，模拟电路设计所花费的时间却远远超过数字电路的设计。模拟电路设计自动化工具的欠缺，使得模拟电路设计自动化成为学术研究相当活跃的一个领域[1]。

传统上来说，模拟集成电路的设计主要是使用数值电路仿真器，诸如 SPICE [2]，不断地仿真迭代，手工寻找最优解。工程师输入电路网表，给定电路参数，通过不断地观察仿真结果，调整电路参数，直至达成电路的指标要求。这一过程需要反复的运行仿真程序，相当耗费时间。并且没有一个直观的表达电路参数和电路指标的方式，使得这一过程十分的复杂。因此长久以来，模拟电路的设计更多的需要工程师经验的积累以及反复的流片测试，被数字电路的设计周期拉开了很大的差距，成为了混合信号电路设计中的瓶颈。

因此，模拟电路设计自动化成为一个重要的研究领域，并提出了不少用于模拟电路设计自动化的算法。一种主要的方法就是将模拟电路的设计转换为带有约束条件优化问题。对于求解和电路优化有关的这类带约束条件的问题，已经提出了诸如模拟退火（Simulated Annealing）[3]、遗传算法（Genetic Algorithm）[4]、凸集规划（Geometry Programming）[5] 、基于导数的优化方法[6]等人工智能算法。使用这些算法的确能够取得不错的自动优化的效果，但其自动优化的过程掩盖了电路本身的特性，因而这类研究还停留在学术研究的阶段，并没有得到广泛的接受和应用。

因此，作为一个好的模拟集成电路设计工具，在提供了一系列可以自动优化调整电路的过程的同时，还要兼顾模拟集成电路设计者的习惯和固有的方法，以及模拟电路本身的特性，才能更好地提高模拟集成电路设计的效率。本文旨在传统的设计方法和自动优化的算法之间，探索寻找出一种高效地，结合两种方法优

点的模拟电路设计工具。

1.2 基于灵敏度分析的设计方法简介

传统的模拟设计电路方法，主要通过近似的小信号分析和近似的模型公式，来反映电路性能指标和晶体管尺寸之间的关系[7]。这种方由于忽略了许多晶体管的寄生参数，因此会产生比较大的误差。随着工艺尺寸的缩小，寄生参数的影响越发重要，使得这种方法在复杂的纳米级电路设计中显得日益捉襟见肘。

一种解决模拟电路 Sizing 的方法，就是利用电路的灵敏度信息，即电路的传输函数关于晶体管尺寸的导数（Size-Referred Sensitivity）。灵敏度大小和方向的信息，很好地反映了电路的指标和晶体管尺寸之间的关系，并且可以通过某种方式绘制相应的曲线，给模拟电路的设计者一个直观的印象，告诉他们某个变量是如何影响某个电路指标的。

比如电路的传输函数 $H(s)$ 的模 $|H(s)|$ 在 $\omega=0$ 对变量 W_1 的灵敏度为+0.5，对变量 W_2 的模为-0.1，那么就表示了增加变量 W_1 或者减小变量 W_2 的值可以提高电路的直流增益。同时由于对 W_1 的灵敏度的绝对值更大，因此变量 W_1 的变化将比变量 W_2 的变化对电路的直流增益有着更大的影响。模拟集成电路的设计者就可以更具这些数据，找出影响电路性能指标的关键变量，从而能够更高效的优化电路。

同时，一些自动优化算法也需要使用电路的灵敏度信息[6]。无论是用灵敏度分析的结果指导电路设计，还是将其用于电路的自动优化，电路的灵敏度计算的效率和准确性都是需要考虑的因素。用数值算法计算电路的灵敏度，一般会使用有限差分的方法（Finite Difference），这种方法要跑两次仿真，求取两点之间的差分，会产生一定的数值误差，并且效率不高。

而另一种求取电路灵敏度信息的方法，就是通过符号化仿真求解[8][9]，这类方法通过一些特定的方式，用某种数据结构来表达电路的传输函数（Transfer Function）$H(s)$，并通过对数据结构的一些操作，实现对 $H(s)$ 的求导。这种方法能够有很高的效率，并且避免了数值方法求解矩阵过程中引入的数值误差，实现对电路灵敏度信息的高效准确的求解。并且在文献[10, 11, 12] 中，已经有了使用符号化灵敏度方法来 Size 模拟电路的尝试。

然而，在文献[10, 11, 12] 的方法中，使用符号化仿真器求取灵敏度信息的过程中，仍然存在着一些近似和不精确之处，忽略了一些重要的因素。在某些情况下，甚至会求出完全错误的灵敏度信息，会误导工程师的设计以及自动优化设计

的方向。

因此，本文在此基础上，计划改进符号化仿真器计算电路灵敏度仿真的算法，使得灵敏度的求解准确而高效，并且利用这种算法，尝试探索改进模拟电路互动和自动设计的方法，来实现一个简单的模拟集成电路（主要是运算放大器电路）的 Size 工具。

1.3 研究和实现的基本方法

在 EDA 实验室，已经有了比较成熟的、自行研发的各种仿真工具：包括类似 SPICE 的简单数值仿真器[13]和新型的符号化电路仿真器 GPDD[8]。本文尝试在这些仿真工具的基础上，整合改进现有的算法和代码，实现精确的符号化电路灵敏度的计算方法，并且从模拟集成电路的最基本单元运算放大器入手，来研究如何通过结合手动和自动的设计方法，来尝试设计一个简单的模拟集成电路设计工具。考虑模拟电路设计者的习惯，比较两种方法的优缺点，并试图结合它们的优势，来完成一个能够比较高效地帮助模拟集成电路设计者完成设计的工具。

1.4 论文的主要内容与章节安排

本文主要通过对符号化仿真器的改进，实现了基于符号化仿真的电路的灵敏度分析。并在基础上，针对一些常见的模拟电路（主要是运算放大器）的拓扑结构进行测试，并将基于灵敏度的设计方法和传统的手工设计方法以及自动优化的设计方法的结果进行比较。并期望以此完成一个功能比较完整的模拟集成电路设计工具。

本文的章节安排如下：。

第二章将主要介绍符号化电路仿真器 GPDD 的基本算法、实现、特点以及应用。

第三章在第二章的基础上，阐述了运用符号化电路仿真器进行灵敏度分析的算法和实现，简要地介绍了其中的数据结构和接口实现，并提出了基于灵敏度的互动电路设计方法。

第四章介绍了几种常用的自动优化电路的方法，并结合电路的灵敏度分析说明灵敏度分析在自动电路优化问题中的应用。

第五章介绍了一些软件实现上的细节，说明了软件中的部分数据结构以及对 GPDD 算法的一些改进。

第六章通过对几个常用的运算放大器电路的分析，以及软件运行的结果，比

较互动和自动方法在调整这些模拟电路尺寸中的应用。

第七章总结全文，并说明其中的不足和今后的研究方向。

第二章 符号化仿真器的算法介绍

本章主要介绍基于 GPDD 的符号化电路仿真器的基本原理和实现方式，简要说明了符号化电路仿真器 GPDD 数据结构的构造以及求值的全过程，以及在实现上的一些优化。

2.1 符号化电路仿真器概述

在模拟电路的设计过程中，电路仿真器是必不可少的工具。常用的电路仿真器使用数值计算的方法，通过一定的方式，将电路方程（KCL、KVL 方程）转化为矩阵的表示形式，通过对矩阵的求解和操作，来求得电路的节点电压、支路电流等变量的值，从而得到电路方程的数值解。这一过程如图 1 所示：

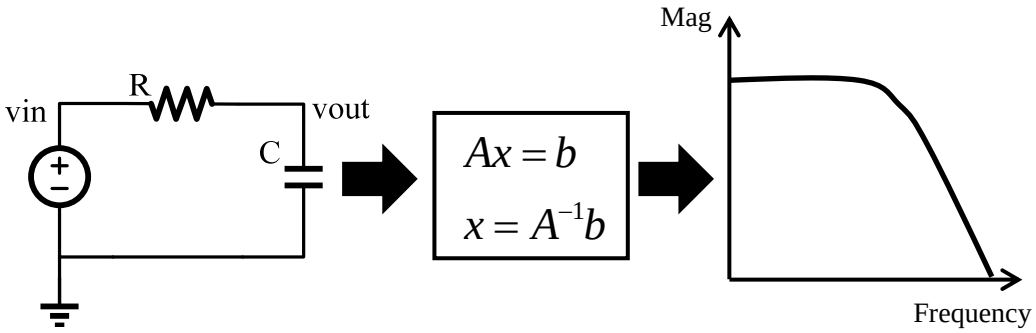


图 1：数值化电路仿真的流程
Fig 1: The Basic Flow of Numerical Simulation

加州大学伯克利分校（University of California, Berkeley）在上个世纪末基于这一系列矩阵理论和数值分析的算法，开发出了数值化的电路仿真软件 SPICE。此后工业界基于 Berkeley SPICE 开发了许多高性能的数值化电路仿真器（如 Synopsys 的 HSPICE，Cadence 的 SPECTRE 等），这些仿真器成为了工业界的主流。而基于数值化仿真器，通过反复修改电路参数并运行仿真程序，观察仿真结果的设计步骤，也成为了模拟集成电路设计的常规步骤。

而符号化电路仿真器另辟蹊径，通过对电路自身的图的结构进行一些特定的操作，将其转化为某种特定的数据结构。再从该数据结构中导出电路的传输函数。

数据结构构造完成后，如果电路的拓扑结构不发生变化，只需要改变数据结构中表示电路元件参数的数值，就可以求得新的电路传输函数。

符号化仿真器通过对数据结构的操作，可以直接获得电路传输函数的解析表达式。这个表达式是和电路的拓扑结构一一对应的。相比于数值仿真器，当某个元件的参数发生变化时，并不需要重新求解矩阵，只需要更新数据结构中某个变量的值，代入解析表达式就可以求得。这不但可以加快求解的速度，还减少了数值分析过程中计算误差的积累。

此外，符号化仿真器完整地存储了电路中的信息，因此可以很方便地观察那些变量对电路的传输函数有比较大的影响，哪些变量和性能指标关系比较密切等。更重要的是，通过导出的传输函数的解析表达式，可以在此基础上进行电路的自动优化以及设计，提高模拟集成电路设计的效率。

符号化仿真器早在上个世纪六、七十年代就引起了广泛的关注，并提出了一系列算法来实现电路的符号化仿真[14]。然而，由于符号化仿真，尤其是对于较大规模的模拟电路进行仿真时，存储电路表达式所需要的空间以及仿真所需要的时间会呈现指数级的增长，这使得符号化仿真器的应用变得相当有限，并且几乎在在诸如 SPICE 的电路仿真器的崛起后销声匿迹。

直到 2000 年，X.D. Tan 等[15]提出了 DDD（Determined Decision Diagram）的数据结构，将原本用于数字集成电路综合技术的数据结构 BDD（Binary Decision Diagram）[16]用来表示模拟电路传输函数的乘积项，通过 BDD 构造过程中的共享机制，很好地控制了符号化仿真中内存和时间随着电路规模的增长速度。

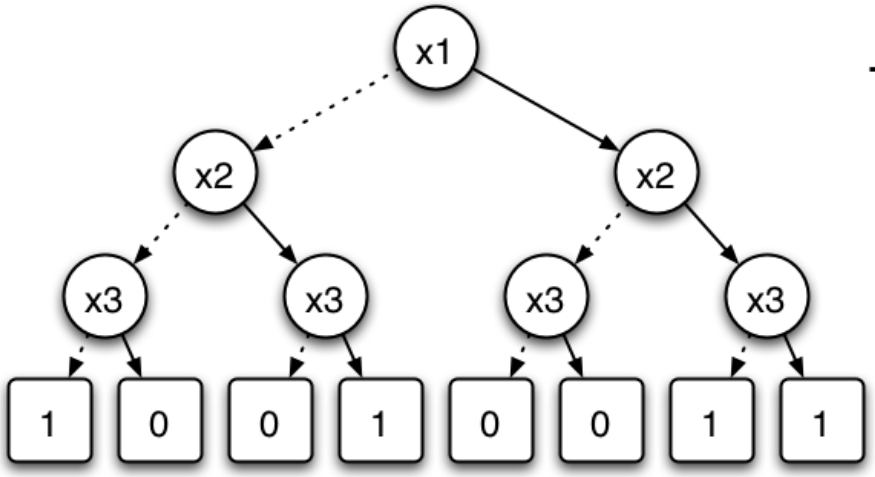
之后，G. Shi[17, 18]等通过图约化的方法，同样利用 BDD 这一个数据结构，枚举电路中的生成树，实现了基于图约化的符号化仿真器 GPDD（Graph Pair Decision Diagram），相比 DDD，消除了对消项在表达式中的影响，并且已经能够处理二十多个晶体管的模拟电路了。

但在处理更大规模的电路时，这些方法还是显得有些无能为力。H. Xu[19]等在 DDD 和 GPDD 的基础上，尝试了使用层次化的分析方法，将电路进行模块划分，自底向上地进行求解，并获得了不错的效果。现在的符号化仿真器，已经可以很高效率得精确处理诸如 ua741、ua725 等有一定规模的模拟集成电路了。下面就来简单介绍一下 GPDD 算法的原理。

2.2 数据结构：BDD

BDD（Binary Decision Diagram）[16]是一种用于表达布尔逻辑函数的数据结

构，它是一个有向无环图（Directed Acyclic Diagram）。它从一个表示逻辑函数的根节点出发，有 0 和 1 两条边指向两个子节点，而每个子节点又是一个 BDD。使用递归的方式定义，直至两个终止节点 0 和 1。图 3 显示了如何使用 BDD 来表达一个逻辑函数



x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

图 2：用 BDD 的数据结构表示逻辑函数

Fig 2: Representing Logic Functions with BDD

图 3 中的真值表表示了逻辑函数，而它所对应的 BDD 就是图 3 中形如二叉树的数据结构。从根节点 x_1 开始，可以找到四条以节点 1 为终止节点，这些路径以及路径上 0 和 1 的排列，分别对应于真值表中的等于 1 的项。这样的数据结构唯一的表示了逻辑函数，可以通过比较两个逻辑函数的 BDD 是否相同来判断它们是否等价。

然而，这样的数据结构会随着输入变量的增加呈现指数的增长。就这个问题，R. E. Bryant 提出了 ROBDD（Reduced Order BDD）[20]，通过在 BDD 构造过程中重复的结构之间的数据结构的共享，以及无关项的消除，大大的减少了 BDD 的内存消耗以及构造时间，使得用 BDD 表示复杂的逻辑函数成为了可能。图 4 说明了图 3 经过约化以后生成的 ROBDD，可以发现重复出现的节点实现了共享，二叉树的结构被简化成了二叉的有向无环图（DAG），所占用存储空间也被显著压缩。

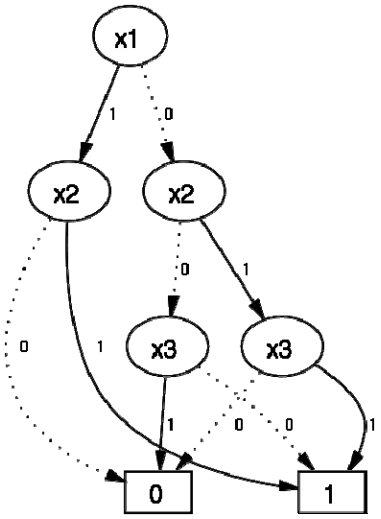


图 3：经过约化以后的 ROBDD

Fig 3: Reduced Order BDD(ROBDD)

事实上，BDD 的应用不止在于存储比较逻辑表达式。因为逻辑表达式可以表示成集合的形式，所以 BDD 可以用来表示集合中的组合。Minato[21]提出了 BDD 的一个变种 ZBDD（Zero-Suppressed BDD），用于经济地存储表达组合集（Combination Set）。

传统上来说，BDD 是一种用于存储逻辑表达式的数据结构，BDD 中每一个节点所表示的变量都对应于逻辑表达式的一个输入变量。然而，只要将这一数据结构进行推广，如果将 BDD 中的节点指向线性电路中的小信号参数，那么 BDD 将成为用于表达电路传输函数（Transfer Function）的一种紧凑的数据结构，下面就将介绍如何通过 BDD 来表示电路的传输函数。

2.3 符号化引擎：GPDD

如果把 BDD 中的逻辑变量换成连续的变量，BDD 就可以用乘积项之和的方式来表示连续的函数，因此用 BDD 来表示模拟电路的传输函数 H(s)就成为了可能。同时利用 BDD 构造过程中的共享机制[16]，就可以用一个相当经济、高效的数据结构来表征模拟电路连续的传输函数。

从输入的电路网表构造用于表示传输函数的 BDD，有两种构造方法。论文[15]中提出了 DDD（Determined Decision Diagram）的数据结构，使用了和数值仿真器类似的机制，将电路方程转换为矩阵，在通过矩阵的行列式展开来求取符号化的传输函数。而论文[18]中，提出了 GPDD（Graph Pair Decision Diagram），通过将

电路网表转换为有向图，再通过图约化的算法，来实现数据结构的构造。相比DDD，GPDD 的构造过程更直观，并且避免了表达式中对消项的出现（Cancellation Free），因而相比 BDD 有着更好的数值稳定性和空间效率。

GPDD 采用了图约化的方法。首先将电路网表转换为有向图，再对生成的有向图进行操作，在枚举有向图的生成树的过程中，就可以构造出用于表示电路传输函数的数据结构。使用这样的构造方法，BDD 中的每一个节点与一对生成子图相关联，因而得名 Graph Pair Decision Diagram。整个构造过程可以用图 5 表示。

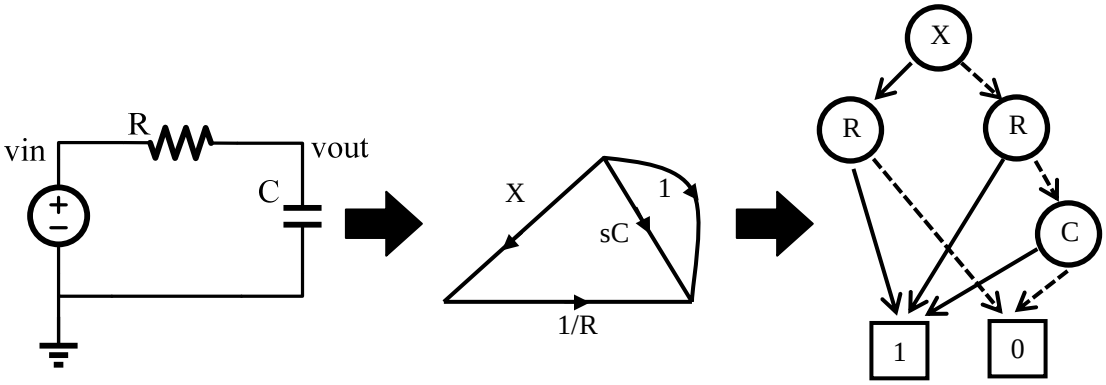


图 4：GPDD 的构造过程
Fig 4: Building Process of GPDD

GPDD 的构造过程中，需要通过 BDD 的共享机制，来枚举有向图图中所有的生成树[22]，此时就需要根据给定的小信号参数的顺序，按照一定的规则减少电路网表所对应的有向图中的边和节点，直至生成的子图不连通或者只剩单一节点的时候为止。这整个过程称为图的归约。当图的归约完成后，GPDD 就构造完成了。

与 DDD 相比，GPDD 的最大优势在于，生成的 BDD 中，每一个节点的名称与电路中的元件相对应，而不是抽象的矩阵中的元素。这样就可以很直观的表示一个电路的传输函数和每一个电路元件参数之间的关系，不需要像 DDD 一样将几个电路元件的值组合起来共同构造一个节点。这样的表述方式，可以很方便的求取电路传输函数对于某个电路参数的导数，以及进行一些其他的操作。

因此我们选用 GPDD 算法作为仿真引擎对电路进行分析。软件实现的过程中，在[8]的基础上重新实现了 GPDD，并对 GPDD 的构造算法进行了优化，这一部分内容将在 2.3 节中加以说明。

2.4 GPDD 的构造算法

GPDD 的构造的主要原理，是在于不断地对表示电路连接关系的有向图做约

化操作（Graph Reduction），以此来枚举该有向图中所有的生成树（Spanning Tree），并通过 BDD 的数据结构在归约的过程中共享其中的数据。可以证明通过这样的枚举算法，是可以求得电路的传输函数（Transfer Function）的。详细证明可参考[8, 23]的正文及其附录。

由于符号化电路仿真器主要处理的是线性电路，因此电路中出现的符号就只包括了导纳 Y（电阻、电容、电感）和四类受控源（VCCS、VCVS、CCCS、CCVS）以及用来表示理想运算放大器的 Nullor。每个电路元件，根据其在电路网表中的连接关系，都对应于有向图中的一条边（如电阻 R 对应一条导纳 Y 类型的边）或者一对边（如 VCCS 对应一条 VC 类型的边和一条 CS 类型的边），在构造的过程中，由于枚举生成树的需要，必须在有向图中“选取”（Short）或者“移除”（Open）相应的边（或边对），不断地缩小图的规模，直至图不连通或者为空。这一过程就被称为图的约化。

这几种元件约化的基本规则如表 1 所示：

	选取元件		移除元件	
	左图	右图	左图	右图
VCVS	选取 VS	移除 VS 选取 VC	选取 VS	选取 VS 移除 VC
CCVS	选取 VS 移除 CC	取择 CC 移除 VS	选取 VS 选取 CC	选取 VS 选取 CC
VCCS	选取 CS	选取 VC	移除 CS	移除 VC
CCCS	选取 CS	选取 CC	选取 CC 移除 CS	选取 CC
Nullor	选取 NO	选取 NU	移除 NO	移除 NU
Y	选取 Y	选取 Y	移除 Y	移除 Y

表 1：图约化的基本规则

Table 1: The Basic Rule of Graph Reduction

图约化（Graph Reduction）算法是 GPDD 构造过程中的核心算法，可以通过自上而下的对有向图进行约化并构造 GPDD 中的节点。每个 GPDD 节点需要存储两张约化后的子图 GLeft 和 GRight，并且有两个指向后继节点的指针 pInclude 和 pExclude，直到表示约化终止的叶子节点“1”和“0”为止。

由于 GPDD 中同一层的节点所指向的符号是相同的，因此可以通过层次遍历，按广度优先的方法，对图进行约化操作，而无须使用[8]中所使用的深度优先的递

归算法，其过程如下：

（1） 根据电路的拓扑结构构造初始的有向图，并将其按一定的规则拆分成左右子图，生成根节点 *root*，根节点的符号为表示输入输出关系的符号 *X*；

（2） 将根节点压入队列 *q*；

（3） *q* 非空时，执行（4）到（7），否则返回，构造完成；

（4） 弹出 *q* 队首的节点 *pNode*，得到它的符号 *symbol* 及左右子图 *GLeft* 和 *GRight*，对其左右子图根据表 1 的规则进行约化，生成两对子图 *GLeft_in* 和 *GRight_in* 以及 *GLeft_out* 和 *GRight_out*。

（5） 判断约化后的子图对是否满足终止条件，若枚举终止返回（3），否则转至（6），终止条件为：若 GLeft 和 GRight 均为只有一个点的子图，则表示枚举到了生成数，返回节点“1”，若 GLeft 和 GRight 均为不连通的子图，则表示不可能有生成树产生，返回节点“0”；

（6） 在哈希表中查找是否存在两队子图中的四个子图分别同构的子图，若存在返回同构子图的地址，否则将子图存入哈希表；

（7） 根据 *symbol* 的名称哈希查找后的 *GLeft_in* 和 *GRight_in* 以及 *GLeft_out* 和 *GRight_out*，构造 *pNode* 的后继节点 *pIn* 和 *pOut*，并确定其符号。在哈希表中查找表示 *pIn* 和 *pOut* 的三元组（symbol，GLeft，GRight）是否存在，若存在返回三元组的地址，否则将该三元组存入哈希表，并将该节点压入队列 *q*，并返回（3）；

相比于深度优先的构造方法，由于广度优先的构造方法在层与层之间的符号不同，不存在节点的共享问题，因此可以节省哈希表中的空间，减少构造过程中内存的占用，提高 GPDD 的构造效率。

在图约化的操作完成后，再对生成的数据结构进行零压缩（Zero-Suppress）以及 BDD 三元组的共享后，就完成了对 GPDD 的构造，详细过程可参考相应的参考文献。

GPDD 算法通过利用 BDD 的共享机制，可以高效的表达出电路传输函数的符号化的表达形式，一旦传输函数的符号化表达形式导出，用户就可以反复的利用 BDD 的求值的规则，对电路的传输函数进行求值或者求导操作。虽然导出 GPDD 的算法代价比较大，但只需要做一次就行，因此 GPDD 的算法还是很有竞争力的，故在设计工具的实现过程中，使用了 GPDD 算法作为符号化电路仿真的引擎。

不过 GPDD 的规模会随着电路规模的增加而显著增大，从而降低了构造

GPDD 结构和对 GPDD 求值的效率，故发展起了层次化的 GPDD[19]来应对电路规模增大带来的影响。不过由于一般运算放大器电路的规模不是很大，GPDD 足以应对，因而并为采用实现复杂的层次化 GPDD。

2.5 GPDD 的求值与求导

在 GPDD 的结构构造完成之后，可以对其进行反复的求值运算，以获取电路的传输函数的数值形式，绘制电路的传输特性曲线。GPDD 使用了 BDD 的数据结构，因此和 BDD 有着相似的求值方式，只是将逻辑“与”运算推广成了实数的相乘运算，将逻辑“或”运算推广成了实数的相加运算。

BDD 可以看作是由共享节点的二叉树，其基本单元和二叉树类似，由一个根节点和左右两个子节点递归定义。在 GPDD 中，基本单元及其求值的方式如图 5 所示：

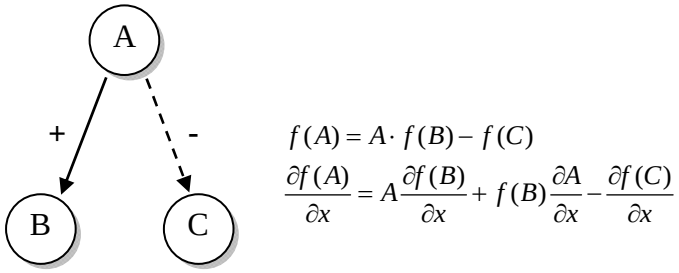


图 5：GPDD 的求值规则

Fig 5: The Evaluation Rule of GPDD

GPDD 由许多如图的基本单元递归构造而成，按照图中所示的递归求值的规则，先对根为 root->pInclude 为根的 BDD 求值，在对根为 root->pExclude 为根的 BDD 求值，最后根据符号和生成项确定法则，就可以自底向上地对电路的传输函数进行求值。注意到 GPDD 中，“二叉树”的节点中形成共享，因此在递归求值的过程中，如果碰到已经求过值的节点，就无需继续递归向下，可以直接获取已经计算的函数值。因此 GPDD 的结构也是很适合对传输函数进行求值的。

通用同样的递归求值的方式，也可以求得传输函数对某个变量的导数。假设参数 A、B 和 C 都是变量 x 的函数，那么对图 5 的求值递归方程求导，即 $\frac{\partial f(A)}{\partial x} = A\frac{\partial f(B)}{\partial x} + f(B)\frac{\partial A}{\partial x} - \frac{\partial f(C)}{\partial x}$，这样就可以在同一次的递归求值的过程中，同时求出传输函数和传输函数对于某个变量的导数的值。

能够反复进行求值与求导操作，便是 GPDD 数据结构的一大优势。GPDD 的

构造算法虽然复杂度很高，但对同一种拓扑结构，只需要进行一次构造，之后无论电路中的参数如何改变，都不会影响这个数据结构的组成，此后对该种拓扑结构的电路传输函数进行求值或者求导操作时，只需要遍历该数据结构就可完成，而无需像传统的数值仿真器那样重新构造矩阵的数据结构。

此外，由于不同情况下的求值和求导操作使用的是相同的遍历 GPDD 的操作，因此如果使用类似多核和 GPU 加速的并行计算的机制，可以在一次遍历过程中同时求得电路传输函数在多个频率点的传输函数或者对多个变量的导数，这种有很高并行度的算法，是 GPDD 用于模拟电路设计的一个重要的优势。

2.6 本章小结

本章主要介绍了 GPDD 的基本原理和构造算法，说明了 GPDD 的特点和它主要的用途，为后文说明如何利用 GPDD 实现运算放大器设计工具奠定基础。

第三章 基于符号化仿真器的灵敏度分析及其应用

本章主要介绍了在 GPDD 的数据结构的基础上对电路进行灵敏度分析的基本原理和实现方法，阐述了灵敏度仿真在电路分析中的一些应用，并提出了基于灵敏度分析对电路进行互动优化的基本方法。

3.1 灵敏度（Sensitivity）分析

电路的灵敏度分析，就是指某个电路参数对某个电路指标的影响程度。一般来说，这一数值使用归一化的导数来表示的，如公式（3.1）所示：

$$Sens(f, p) = \frac{d \log f}{d \log p} = \frac{\partial f}{\partial p} \frac{p}{f} \qquad (3.1)$$

灵敏度作为归一化以后的导数，反应了目标函数随着某些变量变化而改变的程度和方向。灵敏度的绝对值越大，则表示该变量对目标函数的影响也越大，反之亦然；同样，灵敏度的符号也可以反应出这种变化的方向，正值表示目标函数与该变量的值正相关，负值则表示负相关。

所以对电路图的传输函数（Transfer Function）求取灵敏度信息，就可以知道电路的指标（如增益、带宽、相位等）对电路的设计变量（如晶体管的宽度、长度、偏置电压、偏置电流等）之间的变化关系，这些信息为电路的优化提供了一个很好的参考方向，将有助于对电路进行进一步的分析了解和优化。

在现有的诸如 SPICE 的电路仿真器中，已经提供了这类型的仿真，这些仿真器采用数值计算的方法，计算某个节点电压或者节点电流关于某个节电压和支路电流的灵敏度（DC Sensitivity），以及小信号等效电路的灵敏度响应（AC Sensitivity）。通常，这些灵敏度分析会针对每一个工艺参数的变化（Variation），通过有限差分（Finite Difference）或者模型方程求导（Derivative）的方法，反复求解矩阵来获得。这种方法计算代价高。应用也非常有限，因为集成电路的设计者并不是对每一个电路参数都感兴趣，往往只会挑选一些重要的电路参数来对电路进行优化。

3.2 GPDD 的灵敏度分析

符号化仿真器一个重要的应用，就是用于求取精确的、解析的灵敏度的表达

式。基于 BDD 的符号化仿真器，用于求取电路的灵敏度方面，是有独特的优势的。因为在 BDD 的数据结构下，函数的求导是十分容易实现的。在[24]中，已经提及了 DDD 在灵敏度分析中的应用，但未做深入的展开。在文献[9]的工作中，研究并实现了符号化仿真求取 ac 响应的灵敏度（Symbolic AC Sensitivity）的算法。而在论文[12]中提到的如何在 DDD 以及 GPDD 的数据结构下，进行层次化的符号化灵敏度分析。

GPDD 这个数据结构能够直观的表达电路的传输函数，并且数据结构中的每个节点都能对应到电路中的某个电路元件。这一特点使得在 GPDD 的数据结构下，对传输函数求导数的操作会很容易的实现。文献[9 **错误!未定义书签。**]中已经在 GPDD 的基础上实现了求导的过程。之后的论文[23]中又提出了基于 GPDD 的灵敏度计算的简化算法。

考虑如式 3.2 所表示的一个 SOP 形式的表达式：

$$E(a,x,b,c) = axb + xbc + abc \tag{3.2}$$

其归一化的导数（即灵敏度）

$$\begin{aligned} Sens(E,x) &= \frac{x}{E}\frac{\partial E}{\partial x} = \frac{x}{E}(ab+bc) \\ &= \frac{axb+xbc}{E} = \frac{E - E|_{x=0}}{E} \end{aligned} \tag{3.3}$$

其中 $E|_{x=0}$ 表示不包含 x 的乘积项，那么灵敏度的计算就相当简单了，只需要将 SOP 中不包含 x 的项舍去，在与函数的值作商，就能求得电路的灵敏度。

对应到数据结构上，求取灵敏的操作就相当于将 BDD 中所有与变量 x 相对应的节点的 0 指针指向 0 节点，再对 BDD 进行遍历求值，就能获得 $E - E|_{x=0}$ 的值。再与 E 相除，就得到了电路的灵敏度。这一过程如图 6 所示：

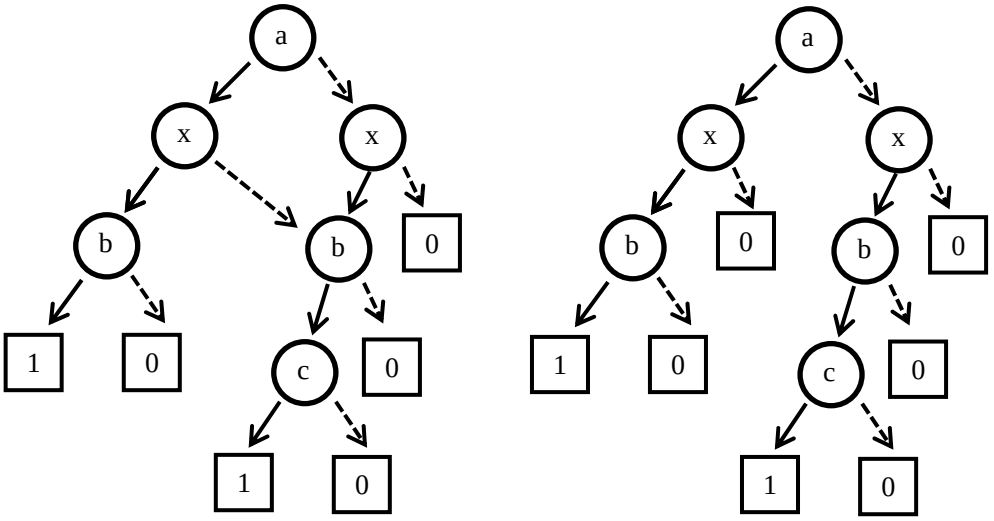


图 6：GPDD 的求导操作 (a) E 的数据结构 (b)求导时的等效数据结构

Fig 6: Obtain the Derivative of GPDD (a) Data Structure of E (b) Equivalent Data Structure when Derivating

而且在实际操作中，并不需要修改数据结构，只需要在遍历求值的过程中，如果找到含有被求导的变量的节点，不计算 0 指针所指向的子节点的值即可，并且可以重复利用之前求取 E 时的计算的结果，有很高的效率。

3.3 关于晶体管尺寸的灵敏度分析

3.3.1 求解灵敏度的链式法则

GPDD 中的每个节点，都对应于电路的小信号网表中的一个符号，因此我们可以很方便的根据上述的规则求取电路的传输函数关于某个小信号参数的灵敏度信息。

然而，要求取电路传输函数关于电路的设计参数（比如晶体的宽度W_i）的灵敏度信息，就不是一件容易的事了。晶体管有很强的非线性，所有的小信号参数都是在某个直流偏置下线性化以后所得到的。因此晶体管宽度的变化不仅会影响该晶体管本身所有的小信号参数，还会影响到该晶体管周围的端口电压，进而影响到其周围的晶体管的偏置，从而影响周围的晶体管的小信号参数。因此，求取电路的传输函数关于晶体管尺寸的灵敏度信息，必须考虑到该变量变化引起的所有的小信号参数的变化。所以根据函数的复式求导法则，就有了如下的计算公式：

$$sens(H(s),W_i)=\sum_{j=1}^{P} sens(H(s),g_j)\cdot sens(g_j,W_i) \qquad (3.4)$$

其中 P 表示变量W_i所影响到的所有的小信号参数的数量。这一规则就称为求解灵敏度的链式法则。其中第一项$sens(H(s),g_j)$可以用 GPDD 快速求得；但是第二项$sens(g_j,W_i)$是小信号参数g_j关于W_i的灵敏度，它和电路仿真中使用的晶体管模型，以及该小信号参数所在的晶体管周围的偏置有关，有着很强的非线性，因此无法使用 GPDD 求得，只能依靠数值分析的方法获得。

此前，H. Yang[10]和 Ma Et. al[11]已经尝试利用符号化仿真引擎以及链式法则求取电路的指标和设计参数（如晶体管宽度）的灵敏度关系了。然而，由于他们在计算灵敏度关系时，没有把式（3.4）中的链式法则考虑完整，忽略了晶体管之间的影响，因而在某些情况下，计算结果会产生较大的偏差，这样的结果将会误导电路的设计者。下面就来说明如何利用链式法则精确的计算电路的灵敏度关系。

3.3.2 小信号参数的灵敏度

求和项中的$sens(g_j,W_i)$是小信号参数g_j关于W_i的灵敏度，根据小信号参数的定义，对于第 k 个晶体管 M_k 的第 j 个小信号参数 gk,j 定义为公式（3.5）的形式：

$$\begin{aligned} g_{k,j} &= \frac{\partial I_k}{\partial V_j} = \frac{\partial f(p_1,p_2,...;V_1,...,V_N)}{\partial V_j} \\ &= g_{k,j}(p_1,p_2,...;V_1,...,V_N) \end{aligned} \tag{3.5}$$

与（2）式中的形式相同，这里的端口电压V_j也应该视为电路参数p_i的函数。因此对公式（3）求关于参数p_i的全导数，得到：

$$\begin{aligned} \frac{dg_{k,j}}{dp_i} &= \partial_{p_i}^{(1)} g_{k,j} + \partial_{p_i}^{(2)} g_{k,j} \\ &= \frac{\partial g_{k,j}(p_1,p_2,...;V_1,...V_N)}{\partial p_i} + \sum_{m=1}^{N} \frac{\partial g_{k,j}}{\partial V_m} \frac{\partial V_m}{\partial p_i} \end{aligned} \tag{3.6}$$

关于小信号导纳参数的灵敏度方程（4）有两部分构成：第一部分是小信号参数方程关于某个电路参数的偏导数，而第二部分的和式则表示小信号受到偏置电压变化所引起的小信号的参数的改变。它由两部分构成，第一部分是小信号参数关于端口电压的导数，其实质就是晶体管模型的电流方程关于端口电压的二阶偏导数；另一部分则为晶体管的端口电压关于参数的导数，即直流灵敏度（DC Sensitivity）。

正如前一节所说，晶体管的偏置可能会受到周围晶体管尺寸的变化而改变。故公式（3.6）中的整个求和项表示了电路之中器件之间的灵敏度的影响。然而在之前的类似的工作中[10, 11, 12]，都没有考虑第二部分的影响，因此计算的结果将

会产生一定的偏差。所以在使用符号化引擎计算电路 ac sensitivity 的时候，必须把电路的 dc sensitivity 考虑进去。

3.3.3 DC 灵敏度分析

电路中的某个 MOS 管尺寸发生变化，电路中各个的节点的电压也会随之发生变化，这就是电路的 DC Sensitivity。

DC Sensitivity 可以通过数值计算的方法，结合电路的 DC 解以及 MOS 管模型方程的导函数来求得。这种方法利用了求解 DC 解释矩阵分解的结果，因此效率很高。具体方法如下[2]：

假定电路中的某个 MOS 管 M_k 所用模型的方程为 $I_k = f_k(p_1, p_2...; V_1,...,V_N)$，其中 p_i 表示电路的参数，V_j 表示 MOS 管的端口电压。注意到因为电路参数 p_i 的变化会引起端口电压 V_j 的变化，因此端口电压 V_j 应该被视为参数向量 p_i 的函数。因此，该电路方程关于某个参数 p_i 的全导数应该写为式（3.7）的形式：

$$\frac{dI_k}{dp_i} = \frac{\partial f_k}{\partial p_i} + \sum_{j=1}^{N} \frac{\partial f_k}{\partial V_j} \frac{\partial V_j}{\partial p_i} \tag{3.7}$$

注意到在第二项的和式中，$\frac{\partial f_k}{\partial V_j}$ 恰好是晶体管的小信号导纳参数，而 $\frac{\partial V_j}{\partial p_i}$ 恰好是要求的 DC Sensitivity。因此根据 KCL 方程的要求，将式（3.7）改写成如公式（3.8）所示的矩阵的形式：

$$G\frac{\partial V}{\partial p_i} = \frac{\partial I}{\partial p_i} - \frac{\partial f}{\partial p_i} \tag{3.8}$$

其中 G 与求解电路 DC 解时所用的小信号导纳矩阵完全相同，而等号右边的两项都可以通过电源的方程以及器件模型方程的偏导数求得，可以作为一个虚拟的电源。利用求解 DC 解时对 G 进行分解的结果，就可以快速得到电路的 DC Sensitivity，没有附加的用于高斯消去的代价。

在实际实现中，利用本科毕业设计[13]所完成的电路仿真器的基础上，增加对模型方程求偏导数的实现，并根据求解 DC 时电路 Stamp 到矩阵的规则实现了 DC Sensitivity 的 Stamp，再重新调用求解引擎，就完成了对 DC Sensitivity 的求解。

3.3.4 与数值仿真器的接口

由于求取精确的关于晶体管尺寸的灵敏度信息需要数值仿真器的帮助，因此需要完善符号化仿真器和数值仿真器之间的接口以及数据结构。

在 GPDD 中，每个 BDD 节点都对应于一个电路中的小信号参数，如图 7 所示：

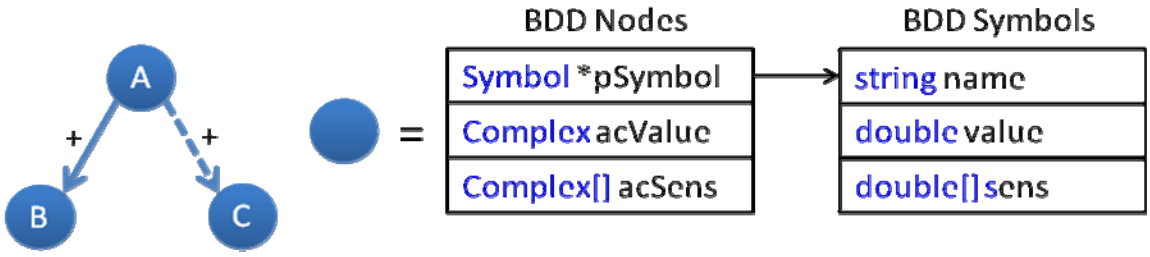


图 7：GPDD 节点的数据结构

Fig 7: Data Structure of GPDD Node

根据 GPDD 递归求值的规则，可以根据公式（3.9）求得：

$$f(A) = A \cdot f(B) + f(C) \tag{3.9}$$

对公式（3.9）求导，得到公式（3.10）：

$$\frac{\partial f(A)}{\partial W_k} = \frac{\partial A}{\partial W_k} \cdot f(B) + A \cdot \frac{\partial f(B)}{\partial W_k} + \frac{\partial f(C)}{\partial W_k} \tag{3.10}$$

由此可见，导数也可以通过递归求值的方法。考虑到递归求值需要大量的堆栈操作，因此，在这次实现中，修改了 BDD 的数据结构，在每个 BDD 节点中，增加一个数组用于存储该节点的导数，以及根据（3.5）中计算的小信号参数，就可以通过一次递归求值，获得电路的传输函数，以及针对所有变量的导数（灵敏度）。

因此，只需要在完成 DC Sensitivity 的仿真后，将所有小信号参数针对所有变量的导数信息存储在表示晶体管的对象中，通过如图中的指针关系，就可以在 GPDD 的遍历求值过程中，找到所有小信号参数的导数，进而在一次递归求值后计算出所有的灵敏度信息，这就通过器件模型，完成了符号化仿真器和数值仿真器之间的接口和数据交互。

3.3.5 结果比较

之前提到，使用 GPDD 引擎对电路进行灵敏度分析时，必须考虑晶体管之间的影响，因而必须把小信号参数的灵敏度以及电路中节点电压和支路电流的敏感度考虑进去，否则计算出来的灵敏度的信息会产生误差甚至错误，从而误导使用设计工具的集成电路设计者。

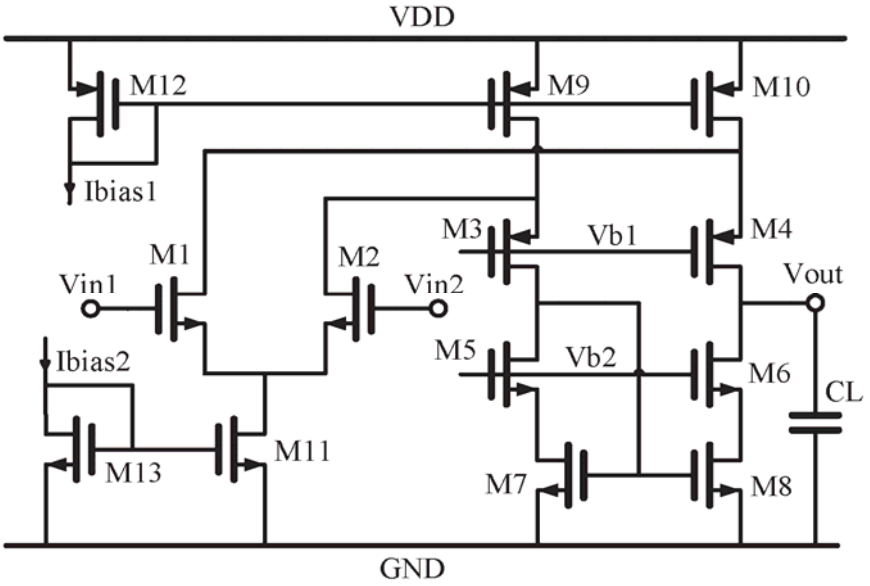


图 8：折叠式共源共栅运算放大器

Fig 8: Folded Cascode OPMAP

图 9 中比较了使用四种不同的方法计算电路的 ac 响应关于晶体管宽度 W9 灵敏度分析的结果。测试的电路是一个如图所示的 Folded Cascode 运算放大器电路，选取的变量 W9 将决定 Cascode 级的偏置电流，因此对所有 cascode 级的晶体管都有影响。从图中可以发现，用本文提出的计算符号化灵敏度分析的方法所得到的结果，和用有限差分法计算所得的结果基本上是吻合的，而使用之 Ma et al.[11] 和 H.Yang[10]的论文中的方法所得的结果都与有限差分法的结果有着较大的误差。

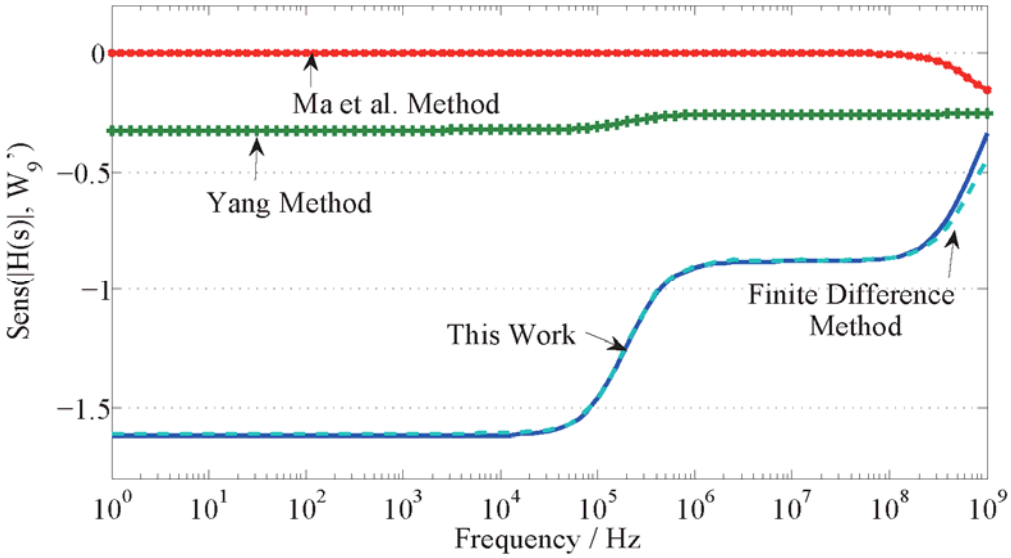


图 9：四种算法进行灵敏度分析的结果比较

Fig 9: Computational Results of Sensitivity Analyses via Four Different Algorithms

在 Ma et al.[11]和 H.Yang[10]的论文中，也尝试使用了符号化仿真引擎对模拟电路进行 ac 的灵敏度分析，但在计算关于晶体管宽度 W 的灵敏度时，都没有考虑到式中求和项对小信号参数的影响，因而在对某些全局影响比较大的晶体管的宽度求取灵敏度时，会产生比较大的误差。在这个例子中，W9 将影响所有 Cascode 级电路中的小信号参数，而 Ma et al.和 H.Yang 的论文中都没有考虑这方面的影响，因而产生了很大的误差。由此可见，在使用符号化仿真引擎时，必须将晶体管宽度对周围晶体管的小信号参数的影响考虑进去。

3.4 灵敏度分析的应用

电路的灵敏度信息主要有两个用途，一是用于提供调整电路尺寸的信息，另一个用途是可以用来预测电路的制造偏差可能对电路性能所产生的影响。

3.4.1 调整晶体管尺寸

电路的灵敏度信息，表征了电路的传输函数关于随着给定的变量的变化关系和变化的程度，可以帮助模拟电路设计者找出影响电路指标的主要的影响的因素，从而进行调整，并优化电路。

使用 3.3 节中的符号化仿真算法在频域上进行扫描，对 GPDD 不断的求取在各个频率点上的传输函数的值和灵敏度，就可以获得电路的 ac 响应和 ac 灵敏度

曲线。一般来说，求得的灵敏度是一个复数，通过一系列的运算就可以求得电路传输函数的 ac 响应的幅度和相位的灵敏度：

$$\begin{cases} sens(H(s), p) = a + jb \\ sens(|H(s)|, p) = a \\ sens(\angle H(s), p) = b / \angle H(s) \end{cases} \quad (3.11)$$

在模拟电路的设计过程中，需要修改电路的设计参数，来调整电路的 ac 小信号响应，使得电路能够满足带宽（GBW）和相位裕度（Phase Margin）的指标要求。而通过符号化仿真获得的小信号响应的灵敏度曲线，就可以将电路参数在各个频段对 ac 响应的影响可视化，帮助模拟集成电路的设计者优化电路。

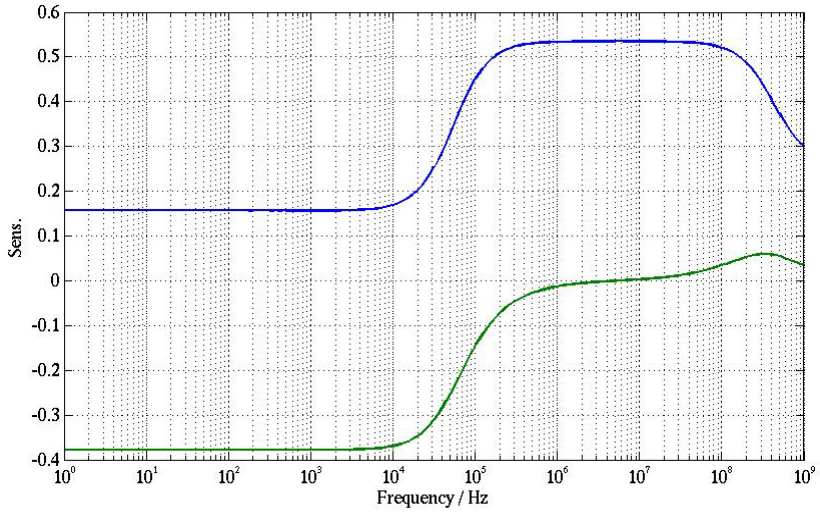


图 10：典型的 ac 响应的灵敏度曲线

Fig 10: Typical Curves of AC Sensitivity

如图 10 所示的两条曲线就是典型的 ac 响应灵敏度的曲线。该灵敏度曲线的幅度，就表示了改变量对电路性能指标影响的强烈程度，而曲线的正负责反映了该影响的方向。

例如在低频段，ac 响应的灵敏度在 0.15 左右，可以看出增加改变量 p 的值可以略微增加电路的直流增益（DC Gain），而在高频段（10kHz--100MHz）段，ac 相应的灵敏度达到了 0.5 以上，表明电路的 ac 响应在高频段受到变量 p 的影响比较大。如果需要增加或减少电路的单位增益带宽，那么就需要相应的增加或者减少改变量 p 的值。

同理，从 ac 响应的相位灵敏度曲线中可以看出，在 1MHz--10MHz 的频段中，相位的灵敏度曲线接近于零，表示该变量 p 在这个频段内对 ac 响应的相位基本没有影响，而在更高的频段上，ac 响应的相位灵敏度曲线为正，且绝对值小于 0.1，

则表示了 ac 响应的相位和变量 p 存在微弱的正相关性，增加变量 p 的值可以略微增加电路的相位裕度。

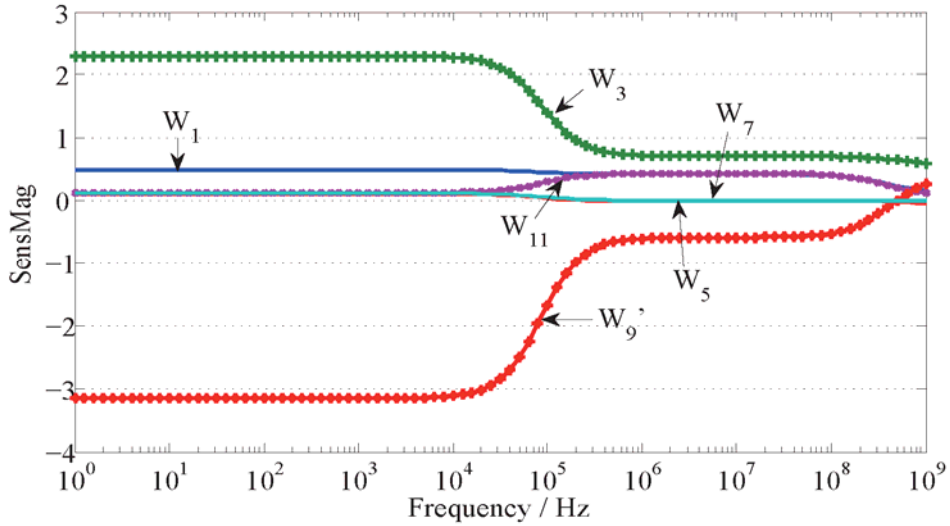


图 11：多变量的灵敏度曲线

Fig 11: Sensitivity Curves of Multiple Variables

如果将电路 ac 响应对多个变量的灵敏度曲线绘制在一起，如图 12 所示，就可以为模拟集成电路的设计者提供丰富的信息，告诉设计者哪些变量在设计中占据主导地位，从而帮助设计者找出影响电路性能指标的最主要因素，能够更高效率的优化电路。

比如从图 11 中可以看出，在低频段，传输函数的幅度响应对变量 W3 和 W9 的灵敏度比较大，因此如果需要增加电路直流增益（DC Gain）的时候，就可以增加 W3 的宽度，同时减小 W9 的宽度；而在高频段，传输函数对 W1、W3、W11 正相关，与 W9 负相关，所以要增加电路的单位增益带宽（GBW）的时候，就可以选择增加 W1、W3、W11 并且减小 W9 的值，从而达到指标要求。

因此灵敏度曲线有助于模拟电路设计者更快地找出影响电路指标的关键变量，实现对电路的优化。

3.4.2 零极点的灵敏度信息

从上一节中的传输函数的幅度灵敏度响应的仿真结果中可以看到，响应曲线会在某一个频段内保持稳定，而会在两个稳定的频段内发生“阶跃”的变化，并且在这个“阶跃”的中间存在一个明显的拐点。如果将这条灵敏度的响应曲线，和电路的零极点信息绘制在一张图上（如图所示）。从图中可以发现，灵敏度曲线

的位置恰好与电路零极点的位置相对应。

这并不是巧合，而是可以根据传输函数的灵敏度曲线的定义推导而来的。假设某传输函数 H(s)用 n 个极点、m 个零点，那么传输函数 H(s)可以表示成：

$$H(s)=A\frac{\prod_{i=1}^{m}(s+z_i)}{\prod_{j=1}^{m}(s+p_j)} \tag{3.11}$$

对 H(s)求取灵敏度，得到：

$$sens(H(s),W)=\frac{\partial\log(H(s))}{\partial\log(W)}=\sum_{i=1}^{n}\frac{\partial\log(s+z_i)}{\partial(W)}-\sum_{j=1}^{m}\frac{\partial\log(s+p_i)}{\partial(W)} \tag{3.12}$$

选取求和项中的任意一项，经化简得到：

$$\frac{\partial\log(s+p_i)}{\partial\log(W)}=\frac{W}{s+p_i}\frac{\partial(s+p_i)}{\partial W}=W\frac{\partial p_i}{\partial W}\frac{1}{s+p_i} \tag{3.13}$$

在一般的运算放大器电路中，零极点一般为实数，故令 $s=j\omega$，可以求得：

$$\frac{\partial\log(s+p_i)}{\partial\log(W)}=k_i\frac{1}{j\omega+p_i}=\frac{k_i}{\omega^2+p_i^2}(p_i-j\omega),k_i=W\frac{\partial p_i}{\partial W} \tag{3.14}$$

可以发现，对于（3.12）中求和项中的任意一项，其实部和虚部都可以写成如式（3.15）的形式：

$$\frac{\partial\log(s+p_i)}{\partial\log(W)}=\frac{k_ip_i}{\omega^2+p_i^2}-j\frac{k_i\omega}{\omega^2+p_i^2}=R_i(\omega)+jI_i(\omega) \tag{3.16}$$

分别绘制实部和虚部对应函数的图像，如图中所示，可以观察到，零极点的位置恰好正是实部函数 $R_i(\omega)$ 的拐点，虚部函数 $I_i(\omega)$ 的极值点所在的位置。

将式（3.16）代回（3.12）式中得到：

$$sens(H(j\omega),W)=\sum_{i=1}^{m+n}k_i\frac{p_i}{\omega^2+p_i^2}-\sum_{i=1}^{m+n}k_i\frac{j\omega}{\omega^2+p_i^2} \tag{3.17}$$

可以发现灵敏度响应曲线的实部和虚部分别是基函数 $R_i(\omega)$ 和 $I_i(\omega)$ 的线性组合，而该线性组合的系数则反映了所对应的零极点的灵敏度信息。对照图中的基函数的图像和图中的幅度响应的灵敏度曲线，可以得到以下结论：

1. 当电路的相邻零极点之间分离的比较好（ p_{i+1} □ p_i ）时，AC 灵敏度曲线实部的拐点、虚部的极值点的位置恰好是电路的零极点所在的位置；
2. 拐点前后函数稳定值之间的差值、极值点的函数值的大小，反映了对应的零极点关于该变量的灵敏度的大小和方向；
3. 如果相邻的零极点比较靠近的时候，可能无法从曲线中看出明显的零极点的位置和灵敏度。

由于在模拟集成电路的设计过程中，往往需要调整零极点的位置来达到电路的指标要求，因此 ac 相应的灵敏度信息就可以很快地告诉集成电路设计者当前电路的零极点所在的位置、零极点位置和设计变量之间的关系以及零极点对消的情况等，从而帮助集成电路设计这更好地优化电路。甚至可以在已知部分零极点位置的情况下，通过数值拟合的方法求得对应的零极点的灵敏度信息。

3.5 本章小结

本章主要讲述了如何结合数值仿真器和符号化仿真器的特点，来对电路进行灵敏度分析的算法和实现，并简要说明了符号化灵敏度算法在运算放大器电路设计中的几种主要的应用。

第四章 模拟电路设计的自动优化算法

本章回顾了几种模拟电路设计自动优化算法，并将着重介绍基于梯度的自动优化算法以及模拟退火的自动优化算法，旨在运用符号化电路仿真器的特点来对电路中晶体管的尺寸进行自动优化。

4.1 自动优化算法综述

一直以来，模拟电路设计的自动化是一个相当复杂的问题。相比于数字集成电路，模拟集成电路的设计的自动化程度一直不高。模拟集成电路的设计者需要对电路中的非线性元件线性化，然后利用电路理论分析的知识了解各个设计变量对电路指标的影响来调整和优化电路，并通过不断的仿真验证来确定电路是否能够正常地工作，符合电路指标的要求。这一过程和数字电路的逻辑综合、布局布线等流程相比，自动化程度相当的低。

学术界一直以来都在对模拟集成电路设计的自动化的研究，并提出了一系列自动优化电路的算法来对电路进行优化。主要有遗传算法（Genetic Algorithm）[4,[25]]、凸集规划（Geometry Programming）[5]、基于梯度的优化算法[6]、模拟退火法（Simulated Annealing）[3]等方法。这类算法使用函数优化或人工智能的方法对电路进行优化，都能够获得不错的效果。

然而，这些算法并没有在模拟集成电路的设计过程中得到广泛的应用。凸集规划法（Geometry Programming）由于需要对晶体管重新建模拟合，并且拟合的公式和传统的器件模型相去甚远，因而不被设计人员所接受；同样，利用了生物遗传的规律的人工智能算法遗传算法（Genetic Algorithm）也因为不符合常规的模拟集成电路的设计方法而不被接受。

而剩下的两种算法，则将模拟集成电路的设计优化问题，转换为代约束条件的最优化问题。在此类算法中，模拟集成电路的自动优化问题，被转换为了一个带约束条件的最优化问题。 需要在给定的设计变量的范围内，找出目标函数的最小值。一般来说，需要最优化的函数 $f(x_1,x_2,...x_n)$ 可以表示为：

$$\min f(x_1,x_2,...x_n)=\sum_{i=1}^{s}k_i g_i(x_1,x_2,...,x_n)+\sum_{i=1}^{p}\sigma_i p_i(x_1,x_2,...x_n) \qquad (4.1)$$

其中函数 $g_i(x_1,x_2,...x_n)$ 表示的是需要最优化的对象，在模拟集成电路的自动优

化问题上，一般会是该模拟电路所用的面积，或者是所需要消耗的功耗，系数k_i则衡量了；而函数$p(x_1,x_2,...x_n)$用作惩罚函数，与电路的性能指标有关（如直流增益、单位增益带宽、相位裕度等），当性能指标达到时，函数值很小，反之则很大。系数σ_i是一个很大的数，以保证在最优化的过程中，这些电路的性能指标始终能够达到。

这类算法和当前模拟电路设计的分析以及迭代仿真的方法类似，并且能够与符号化仿真器的特点相结合，也比较容易进行软件实现，因而就在符号化仿真器的基础上尝试实现了这两种自动优化的算法，来验证并扩展符号化仿真器在模拟集成电路设计自动化中的应用。

4.2 基于梯度的优化算法

对于式（4.1）中的最优化问题，一个比较自然的方法就是希望能够从某一点出发，找到一个能使得目标函数的数值最快下降的方向，并在此方向上寻找到相应的最小值的点[26]。可以证明，对于一个多元函数$f(x_1,x_2,...x_n)$在某一点$(x_1,x_2,...,x_n)$数值下降最快的方向，就是它在该点的负梯度方向，即：

$$\mathbf{d}=-\frac{\nabla f(x)}{\|\Delta f(x)\|} \tag{4.2}$$

向量$\mathbf{d}$所表示的方向就被称为最速下降方向，这种方法就称为最速下降法[26]。利用符号化仿真引擎求得的传输函数的符号化表达形式，就可以很高效地计算出类似式（4.2）形式的目标函数的值及其梯度信息，使用符号化仿真引擎的最速下降法的基本流程如下：

（1） 使用符号化引擎构造 GPDD，获得电路的传输函数$H(s)$；

（2） 使用小信号分析法计算计算初试点$x^{(0)}$

（3） 给定允许的误差$\varepsilon>0$，迭代次数$k=0$；

（4） 对 GPDD 进行求值，求得$H(s)$及其关于各个变量的导数，计算目标函数值$f(x^{(k)})$，并求得最速下降方向

$$d^{(k)}=-\nabla f(x^{(k)})=-(\frac{\partial f}{\partial x_1^{(k)}},\frac{\partial f}{\partial x_2^{(k)}},...,\frac{\partial f}{\partial x_n^{(k)}})$$

（5） 如果$\|\mathbf{d}^{(k)}\|\le\varepsilon$，则完成计算，否则用 0.618 法[26]沿$\mathbf{d}^{(k)}$进行一位搜索，找到系数$\lambda_k>0$使得：

$$f(x^{(k)}+\lambda_k d^{(k)})=\min f(x^{(k)}+\lambda d^{(k)})$$

（6） 令 $x^{(k+1)} = x^{(k)} + \lambda_k d^{(k)}$， $k = k+1$，跳转至步骤（4）

一般来说，目标函数是一个可微的连续实函数，可证明此种方法是收敛的。但自靠近极值点的地方，会由于梯度向量的模值太小而放慢收敛，此时会用诸如共轭梯度法来对下降方向进行一些“修正”，加快其收敛性[26]。

遗憾的是，基于导数的最优化方法的收敛性是局部的，只对诸如图所示中的单峰函数适用，而不适用于图所示的多峰函数。对于此类函数，最优化方法的函数值将会陷在某个极值点内，如果该极值点不是全局的最优点，那么基于梯度的优化算法将会返回局部的最优解而无法返回全局的最优解。

在诸如[6]的一些文献中，尝试使用了基于梯度方向的最优化算法来设计电路，获得了不错的效果。但该算法对初始值的要求较高，如果初始值离最优解相去甚远，将很难获得较好的结果。

4.3 模拟退火算法（Simulated Annealing）

不同于基于梯度方向的搜索算法，模拟退火算法（Simulated Annealing）是一种通用概率算法[27]，能够在固定的时间内找到某个搜索空间内的最优解。模拟退火这一名词来自于冶金学中的退火过程。而模拟退火算法则借鉴了金属退火的过程，将搜索空间内的点想象成具有“能量”的“分子”，每个“分子”都带有“能量”，如果能找到能量越低的分子，就表示该空间内点是更优化的解。

模拟退火算法会从搜索空间内的某一点作为起始点，然后随机地选择该点的一个“邻居”，计算该“邻居”的能量，并根据“邻居”的能量计算接受该“邻居”点的为更优解的概率，直至找到最优解。可以证明，模拟退火算法可以依概率收敛于全局的最优解。一般来说，使用模拟退火算法来优化电路的过程如下：

（1） 产生初始解 $x = x^{(0)}$，给定初始“温度”参数 T；

（2） 根据电路优化的指标要求，给定需要最小化的非负目标能量函数 $E(x)$；

（3） 通过电路仿真，计算初始能量 $E = E(x^{(0)})$；

（4） 在当前解 x 的邻域内产生一个新的解 x'，通过电路仿真计算能量差 $\Delta E = E(x') - E(x)$；

（5） 如果 $\Delta E \le 0$，则接受新解： $x = x', E(x) = E(x')$；

（6） 如果 $\Delta E > 0$，则以概率 $p(\Delta E) = \exp(-\frac{\Delta E}{kT})$ 接受新解 x' 以及 $E(x')$；

（7） 判断是否满足收敛条件，若收敛返回最优解，否则降低系统温度，令 $T := rT, r < 1$，然后返回（4）。

在模拟退火算法优化电路的过程中，算法会始终接受能够使得能量降低的新解，而以一个不断降低的概率接受使得能量增高的解。这样就有助于算法能够离开局部的最优解，而渐进地找到全局的最优解。

在之前的文献中，已经尝试了使用模拟退火算法对电路进行优化，取得了不错的成果。但限于当时计算机的运行速度，往往使用一些近似公式来代替数值仿真来加快优化的速度，因此可能在优化过程中引入一定的误差。

与基于梯度的自动优化算法相同，模拟退火算法的主要代价在于需要不断的进行电路仿真，而电路仿真的时间代价远远高于优化算法本身的代价。尤其是在模拟退火算法搜索的初始阶段，由于随机产生的新解具有一定的“盲目性”，有一定的概率会使得算法需要很长的时间才找到正确的方向，大大增加所需要的仿真的次数，降低优化过程的效率。

4.4 基于梯度的自适应模拟退火算法

在模拟退火算法执行的过程中，每一次的迭代过程都需要在原先的解向量 $x^{(k)}$ 的邻域内随机产生一个新的解向量 $x^{(k+1)}$，并计算 $x = x^{(k+1)}$ 时的目标函数值，并判断是否接受这个新产生的解向量 $x^{(k+1)}$。因此每次迭代的过程中，都需要对电路进行仿真分析，这个代价会远远高于仿真算法本身的代价。所以如果有方法能够减少模拟退火算法的迭代次数，减少优化过程中反复队进行电路进行仿真的次数，就可以极大地提高方针过程的效率。

减少迭代次数的一种方法，就是改变每次迭代方法中产生新的随机解 $x^{(k+1)}$ 的方式。传统的模拟退火算法产生的新解，一般是在原先的解向量 $x^{(k)}$ 的某个邻域内均匀分布的，因此新解向量 $x^{(k+1)}$ 对目标函数 $E\left(x^{(k+1)}\right)$ 的改进也基本上是均匀分布的，所以每一次迭代过程中都有一定的概率走上了“错误”的优化方向，此后模拟退火算法可能会花费很长的时间才能重新找回“正确”的优化方向，从而影响了模拟退火算法的效率。

基于这样的想法，Ingber 提出并实现了一种自适应的模拟退火算法[28]，可以在模拟退火的迭代过程中，自适应地改变新的随机解向量 $x^{(k+1)}$ 的分布，使得目标函数 $E\left(x^{(k+1)}\right)$ 能够以更高的概率走上“正确”的优化方向，从而减少模拟退化算法的迭代次数，提高自动优化算法的效率。

传统的模拟退火算法中，有一个全局的控制变量 T_{Global} 来控制全局的收敛性。而在自适应模拟退火算法中，对于每一次迭代过程中产生的解向量

$x^{(k)}=(x_1^{(k)},x_2^{(k)},...,x_n^{(k)})$，都会引进一个类似 T_{Global} 的“温度”控制向量 $T^{(k)}=(T_1^{(k)},T_2^{(k)},...,T_n^{(k)})$，并用如式（）的表达式来产生新的解向量 $x^{(k+1)}=(x_1^{(k+1)},x_2^{(k+1)},...,x_n^{(k+1)})$。

$$\begin{cases} u_i^{(k)} \sim U[0,1] \\ y_i^{(k)}=\mathrm{sgn}(u_i^{(k)}-0.5)T_i^{(k)}\left[\left(1+1/T_i^{(k)}\right)^{\left|2u_i^{(k)}-1\right|}-1\right] \\ x_i^{(k+1)}=x_i^{(k)}+y_i^{(k)}\Delta x_{i,\max}^{(k)} \end{cases} \qquad (4.3)$$

其中 $u_i^{(k)}$ 是一个在[0, 1]内平均分布的随机数，式（4.3）中的公式计算 $y_i^{(k)}$ 的数值，可以发现 $y_i^{(k)}$ 是一个在[-1, 1]范围内的实数。从而计算得到新的解向量的分量 $x_i^{(k+1)}$ 的值。值得注意的是，$y_i^{(k)}$ 在区间[-1, 1]内的分布，是受变量 $T_i^{(k)}$ 控制的，如图所示：

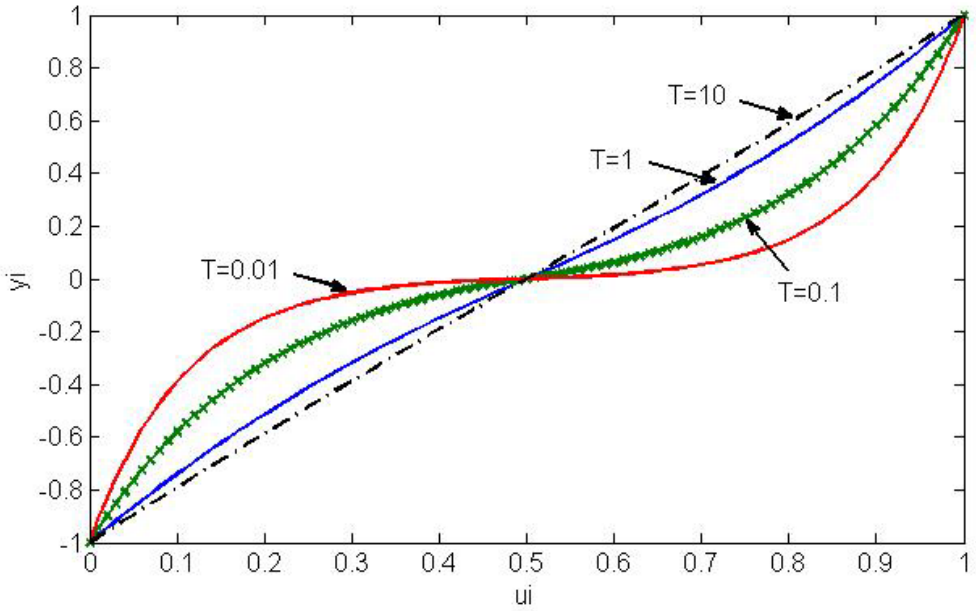


图 12：不同的 T 下变量 y_i 关于变量 u_i 的函数关系

Fig 12: The Relationship between y_i and u_i under Different T

从图 12 中可以发现，$T^{(k)}$ 的值越大，$y_i^{(k)}$ 越接近一条直线，则表示其在[-1, 1]内更接近均匀分布；而当 $T_i^{(k)}$ 的值越小，$y_i^{(k)}$ 的值靠近 0 的概率就越大，$y_i^{(k)}$ 的绝对值得期望 $E\left(\left|y_i^{(k)}\right|\right)$ 的值也越小。因此如果在自适应模拟退火的过程中，控制温度控制向量的分量 $T_i^{(k)}$ 越小，则产生的新随机解的分量 $x_i^{(k+1)}$ 和原解的分量 $x_i^{(k)}$ 的差值 $\left|x_i^{(k+1)}-x_i^{(k)}\right|$ 也会越小。

值得注意的是，通过 GPDD 引擎对电路进行灵敏度分析，可以很方便的求出目标函数 $E(x^{(k)})$ 的最速下降方向 $\vec{d}^{(k)}=\left(d_1^{(k)},d_2^{(k)},...,d_n^{(k)}\right)=-\left(\frac{\partial E}{\partial x_1^{(k)}},\frac{\partial E}{\partial x_2^{(k)}},...,\frac{\partial E}{\partial x_n^{(k)}}\right)$。如果将最速下降方向向量 $\vec{d}^{(k)}$ 与温度控制向量 $T^{(k)}$ 相关联，就可以实现通过梯度值控制自适应模拟退火算法中产生新的解向量的分布，从而使得新的解向量和原来的解向量之间的差 $x^{(k+1)}-x^{(k)}$ 的分布更接近于目标函数 $E(x)$ 的最速下降方向，减少了模拟退火算法迭代过程中随机搜索的盲目性，预计能减少迭代的次数。

在实现过程中，温度控制向量 $T^{(k)}$ 会按照公式（4.4）与最速下降向量 $\vec{d}^{(k)}$ 相关联：

$$T_i^{(k)}=c^{(k)}\frac{d_i^{(k)}}{d_{\max}^{(k)}},d_{\max}^{(k)}=\max\left|d_i^{(k)}\right| \qquad (4.4)$$

这样的好处在于，如果目标函数 $E(x^{(k)})$ 对某个电路参数 $x_i^{(k)}$ 的方向导数（灵敏度）的绝对值越大，那么在下一次模拟退火迭代的过程中，温度控制变量的分量 $T_i^{(k)}$ 的值也会越大，那么产生的新的解向量的分量差 $\left|x_i^{(k+1)}-x_i^{(k)}\right|$ 的期望值也会越大。那么这样产生的搜索方向 $x^{(k+1)}-x^{(k)}$ 更接近最速下降方向 $\vec{d}^{(k)}$，目标函数 $E(x^{(k)})$ 下降的期望值也就越大。

这种方法结合了梯度搜索法明确的方向性和模拟退火算法的全局收敛性，便于实现以及和 GPDD 仿真因引擎的集成，因而在我们的运算放大器设计平台上实现了这种基于梯度的自适应模拟退火算法，并期望能够很好的优化运算放大器的设计，提高模拟电路设计的效率。

4.5 与互动设计方法的比较

相比于第三章提到基于灵敏度分析的互动设计方法，上述自动优化电路的方法同样使用了电路灵敏度分析所得到的结果对电路进行优化。不同的是，自动优化方法将灵敏度的信息隐藏在了优化流程的背后，电路设计成为了“黑匣子”，直接返回了算法认为的最优解。

而互动设计方法可以将这些隐藏的信息可视化，帮助集成电路设计者更好的理解电路。不过在互动设计方法中，由于需要绘制多目标对多变量的灵敏度曲线，因此灵敏度曲线过多，信息量过于庞大，如何很好的使用这些信息也会是一个很大的挑战。因此如果能找到一种好的方法将两种方法结合起来，会是一条解决模拟电路设计自动化问题的很好的出路。

4.6 本章小结

本章回顾了几种主要的模拟电路自动优化的算法，并详细介绍了基于梯度搜索的自动优化算法以及模拟退火的优化算法。最后基于参考文献提出了一种基于梯度的自适应的模拟退火算法，以期望能够提高模拟电路自动优化算法的效率。并和上一章提出的互动设计方法进行了比较，希望能够结合它们的优点，来完成一个简单的运算放大器设计自动化工具。

第五章 软件实现

本章将在前三章介绍的一系列算法的基础上，详细介绍本次毕业设计实现的运算放大器设计工具的软件架构、主要的数据结构，以及电路仿真中所使用的紧凑器件模型。

5.1 整体架构

我们的运算放大器设计工具在 Linux 环境下，使用 C++语言编译而成，主要由五部分组成，包括网表解释器、类似 SPICE 的数值仿真引擎、GPDD 符号化仿真引擎、自动优化引擎以及用于与用户交互的用户界面组成，如图 13 所示：

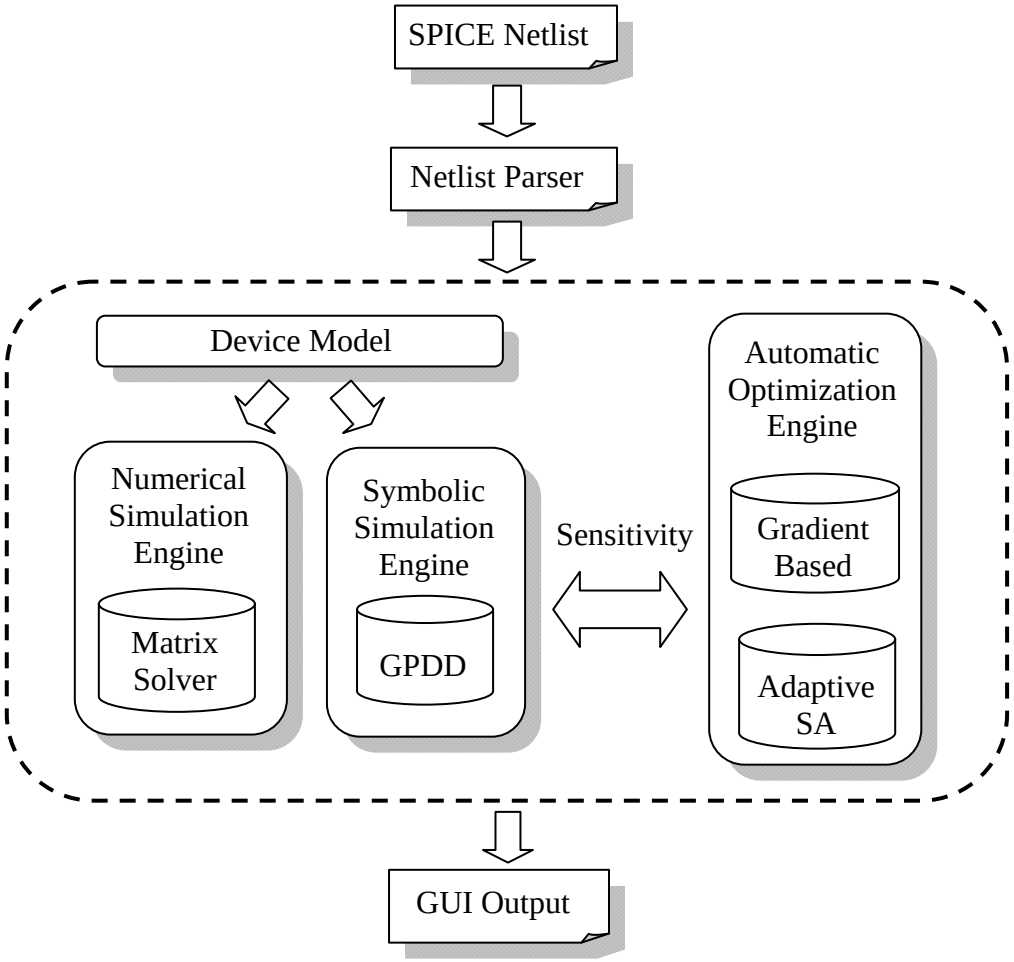


图 13：运算放大器设计工具的软件架构

Fig 13: The Software Structure of the CMOS Design Tool

该设计工具的输入是一个类似SPICE语法规则的电路网表，由Linux下的Flex & Bison工具联合根据网表的语法要求自动编译而成，并且支持变量的定义和简单的四则运算，以方便对晶体管尺寸的定义。经过网表解释器的解析以后，设计工具就获得了电路的拓扑结构、设计过程中需要优化的变量，以及电路的指标要求等。

图5.1中虚线框内的部分是本设计工具的核心部分，主要由一个类似于SPICE的数值仿真引擎、GPDD 符号化仿真引擎以及一个实现上述优化算法的自动优化引擎。数值仿真器和符号化仿真器共享器件模型，用于求解电路的灵敏度信息；自动优化引擎不断的调用数值和符号化仿真引擎，对电路求取灵敏度信息，以完成自动优化算法中所需要的仿真计算过程。

最后整个软件有一个简单的用户界面 GUI，可帮助用户调整电路参数，并用于输出各种响应曲线、灵敏度曲线、以及优化的输出结果，已帮助用户完成整个互动及自动优化的过程。

5.2 主要算法及数据结构

5.2.1 网表解释器

与SPICE的电路网表解释器相似，本工具的电路网表解释器通过读入电路网表，得到电路中的元器件及其连接关系，以获得输入电路的拓扑结构。本工具的电路网表解释器直接沿用了[13]中实现的基本功能，并作了改进。具体实现（如基本语法、哈希表、存储结构等）可参考[13]。

而该设计工具的电路网表解释器，则在原来的基础上，增加了对数学表达式的支持，以帮助用户在电路优化过程中实时更新变量的值，或者描述电路设计中的约束条件（比如晶体管之间的匹配、晶体管尺寸间的比例关系、电路面积、功耗的表达式等）。

假设某个待优化的电路中有5个晶体管$\{M_1, M_2, ..., M_5\}$，它们的长度均为L，宽度为$\{W_1, W_2, ..., W_5\}$，电路匹配的情况下要求$W_1 = W_2$，$W_3 = W_4$，那么彼此独立的变量只有W_1, W_3, W_5, L这四个变量，这些晶体管的总面积$S = \left(2(W_1 + W_3) + W_5\right)L$。由于第四章介绍的优化算法在优化过程中要求晶体管的面积尽可能的小，故在优化过程中需要不断地对电路的面积进行求值以及求导运算，这是就需要将这些变量W_1, W_3, W_5, L

以及面积的表达式 $S=\left(2\left(W_1+W_3\right)+W_5\right)L$ 存储在内存中，以便在优化过程中能迅速找到并更新这些变量的值。

图 14 中展示了通过语法树的方法[29]存储电路中优化变量以及表达式的方法：

$$S=\left(2\left(W_1+W_3\right)+W_5\right)L$$

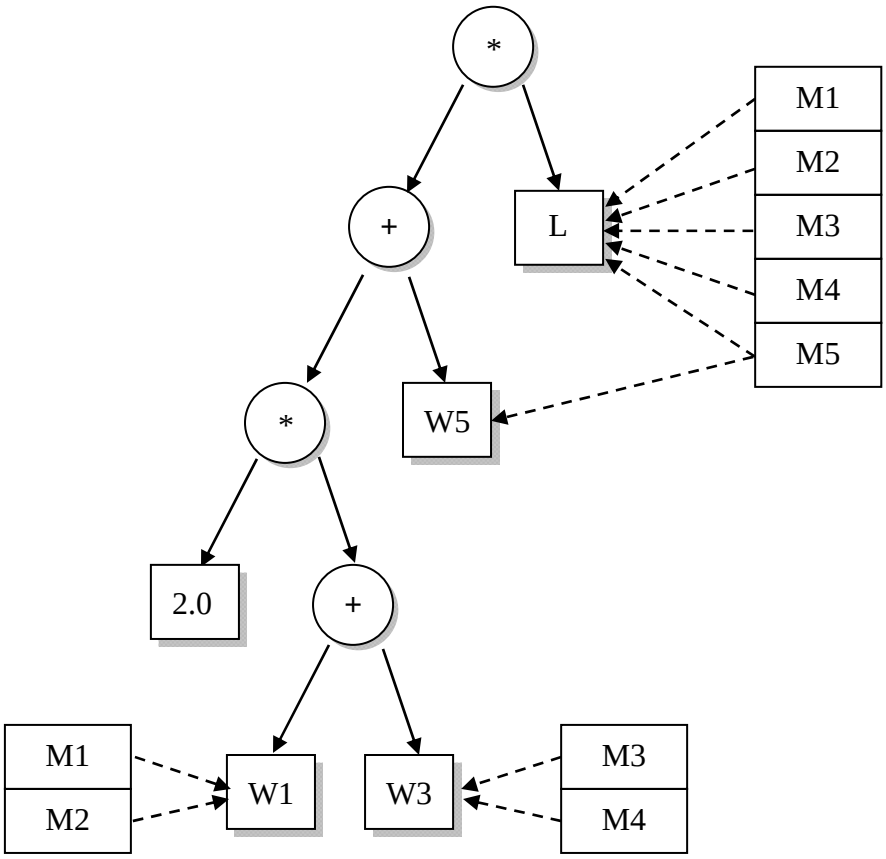


图 14：使用语法树存储电路优化参数和表达式

Fig 14: Storing the Parameters and Expressions using Grammar Tree

其中带阴影的矩形框中表示的是被优化的电路中的变量（W_1,W_3,W_5,L），不带阴影的矩形框中存储的是电路中的晶体管 $\{M_1,M_2,...,M_5\}$。考虑到电路中元器件匹配的要求：$W_1=W_2$，$W_3=W_4$，M_1 和 M_2 中都有指向变量 W_1 的指针，M_3 和 M_4 中都有指向变量 W_3 的指针，M_5 有指向变量 W_5 的指针。这样在每次优化过程中产生的新变量值之后，每个电路元件都可以迅速更新它的值，以便之后通过电路仿真得到电路的性能指标或者目标函数的值。

而图 5.2 中的二叉树的数据结构，就是一颗典型的语法树，表示了该电路中晶体管的总面积的表达式 $S=\left(2\left(W_1+W_3\right)+W_5\right)L$。每个圆形框中的符号表示一个二元的运算符，它的左右子女表示该二元运算符两边的参数。左子女和右子女的指针可以指向一个变量（矩形框），也可以指向一个子表达式（圆形框）。

通过对该语法树进行语义分析[29]，就可以很容易地获得该表达式的值，及其关于某个变量的导数。求值的方法就是对该语法树进行递归求值，方法如下：

*Function：Evaluate(Node *p)*

（1） 如果 *p* 是叶子节点，直接返回 *p* 的值 *p->value*，否则转（2）；

（2） 计算 *p* 的左子树的值 *val_left = Evaluate(p->left)*；

（3） 计算 *p* 的右子树的值 *val_right = Evaluate(p->right)*；

（4） 根据 *p* 中存储的运算符号的语义计算 *val = op(val_left, val_right)*，并返回 *val*。

同理可根据 *op* 的求导的规则很容易的求得表达式关于某个变量的表达式。限于篇幅就不再阐述。在实际实现的过程中，可以用二叉树的中序遍历来代替递归求值的过程，以加快求值或者求导的速度。、

事实上，GPDD 的结构也和语法树有相似之处，都是用类似二叉树的结构来表达一个符号化的表达式，只是两者构造以及求值的方法不同。此外，GPDD 引进了节点共享的机制，将二叉树变成了二叉的有向无环图（DAG），以压缩用于表示表达式的节点的个数。

此外，为了方便用户调用符号化引擎进行灵敏度分析以及自动优化的功能，在原先的网表解释器的基础上增加了如下三条语法：

sac: .sac input_src output_1 output_2

acsens: .acsens variable1 variable2 …

sopt: .sopt expr {double double double}

使得用户能够指定 GPDD 所表示的输入输出之间的关系、需要求取灵敏度的变量、以及自动优化功能的目标函数 *expr* 和性能指标的要求（“{}”中三个浮点数分别为直流增益 DC_Gain，单位增益带宽 GBW 和相位裕度 Phase Margin）。在解析到这些表达式后，网表解释器只需要将这些变量的值或者是表达式的语法树结构存储到相应的位置中就可以了，鉴于过程简单，以及篇幅限制就不展开分析了。

5.2.2 矩阵求解

本设计工具沿用了[13]中的稀疏矩阵的数据结构和 GMRES 迭代矩阵求解器[30]，并在此基础上增加了计算 DC 灵敏度的方法和过程，并改善了牛顿迭代法的收敛性。因此当前的矩阵求解引擎可以求解含有非线性元件的电路，可对其进行 dc、ac、tran、dcsens、acsens 等常见的电路分析方式。虽然 GPDD 可以很高效的求得电路传输函数的符号化表达形式，但实际电路存在着极强的非线性，因此电路中的小信号参数及其灵敏度还必须通过基于矩阵求解的数值仿真引擎来完成。

5.2.3 电路元件

电路元件的数据结构沿用了多重链表结构，将使用同一模型的电路元器件链接在同一个链表下。每一个元件的类都有 Stamp 和 Load 函数，用于将电路元件的参数映射到矩阵中，用于仿真求解。

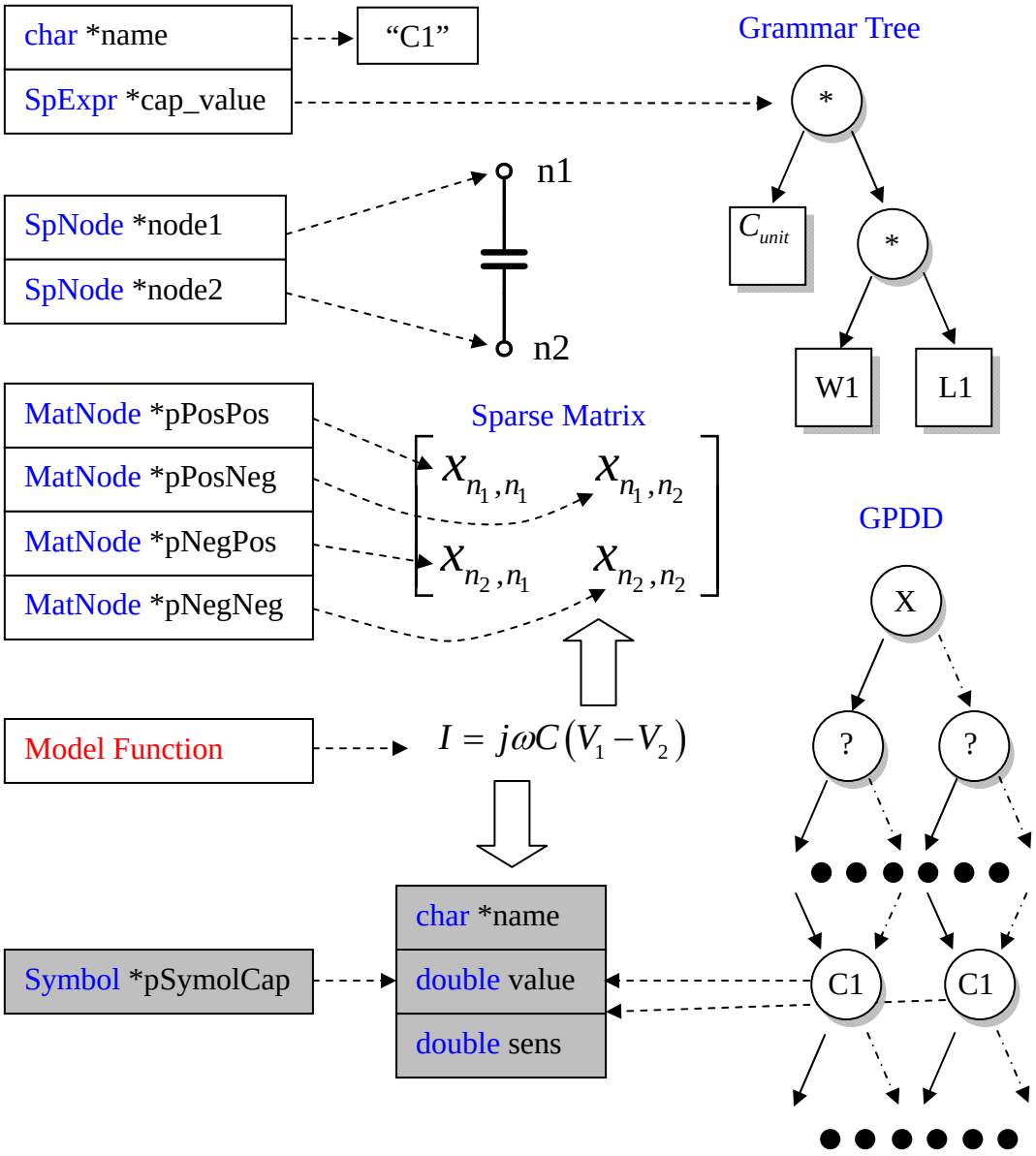


图 15：SpCap 类的基本数据结构

Fig 15: Data Structure of Class SpCap

同样在 GPDD 的仿真过程中，也需要提取电路元件的参数，来对电路的传输函数进行求值或者求导。由于 GPDD 进行的是线性电路的小信号分析，而有些电路元件可以包含多个小信号参数，因此电路元件作为数值仿真器和符号化仿真器公用的数据结构，需要包含和这两种仿真引擎的接口。

图 15 以最简单的电容元件 SpCap 类为例，来说明表示电子元件的类的数据结构以及它们和外界的接口。

首先有一个字符串“C1”用于存储该元件的名称。该元件的取值则由一个表达式确定，因此需要一个 SpExpr 类指针指向一个表示该数值的语法树。为了表示该电容的连接关系，就需要有两个 SpNode 类的指针指向表示该电容连接的节点的数据结构。

之后四个 SpMatNode 类的指针则指向表示电路的稀疏矩阵的结构。根据电容的 Stamp 规则[31]，电容的连接关系将映射到矩阵的四个位置。由于稀疏矩阵的结构无法对矩阵中的节点进行随机查找，因此用相应的指针存储下这些稀疏矩阵中的矩阵节点的地址，就可以快速地以 O(1) 的效率查找到 Stamp 过程中需要更新的矩阵节点的位置，提高了 Stamp 的效率。

除了存储了器件的 I-V 方程以外，还需要一个 Symbol 类的指针指向用来表示 GPDD 中表示线性化后的电路中出现的符号。同时对应的 GPDD 中每个节点都会指向这个结构。这样通过模型方程，就可以不断的计算电路的直流工作点，并且在每次电路的工作点改变时，通过器件的模型方程，可以实时地改变电路中对应的小信号参数的值及其灵敏度。此后通过 GPDD 的递归求值，就能获取电路的响应和灵敏度曲线。通过这样的数据结构，就实现了数值仿真器和符号化仿真器之间数据的交互。

5.3 器件模型

运算放大器属于非线性电路，电路中的晶体管的 I-V 特性有很强的非线性，因此作为一个运算放大器的设计工具，必须要和类似 SPICE 的电路仿真器一样实现晶体管的 I-V 方程。

最简单的 MOS 管的模型就是 Level-1 的平方律模型，易于实现，但精度比较低，会影响设计工具预测出来的结果；而工业界标准的 BSIM3 模型过于复杂，实现起来难度较大，而且模型变量过多，不便于进行第三章中所述的对小信号参数进行求导的运算。因此，在该设计工具中，使用了另一种紧凑的 MOS 晶体管模型：EKV 模型[32]。

EKV 模型是 MOS 管的 Charge Sheet Model[33]的简化版本，其模型公式的形式简单，变量数量少，便于实现，并且能很好地描述 MOS 管的在各个工作区域的 I-V 特性，并且及其适合在低电压设计中用来分析优化电路[34]，因此近年来使用率逐渐增加。而其比较简单的形式也很适合第三章中所述的灵敏度求解的算法，故在本次软件实现的过程中使用了 EKV 的 MOS 管模型。下面将简要介绍 EKV 模型的方程。

与一般的 MOS 管模型不同，EKV 模型的方程以晶体管的衬底（Base）为电压参考点，而不是以源极（Source）为电压参考点。当 MOS 管的栅极（Gate）、源极（Source）和漏极（Drain）上分别加上偏置电压以后，MOS 管的沟道中就会形成反型层，形成导电的通道，如图 16 所示：

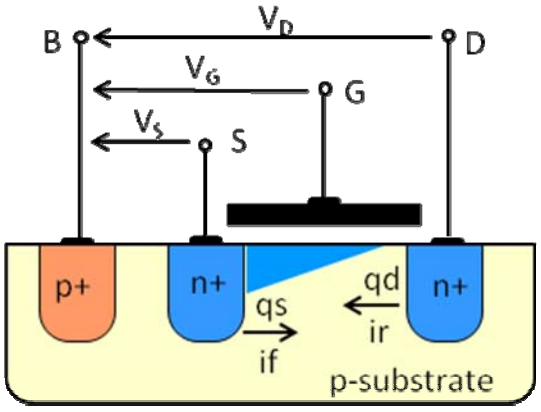


图 16：EKV 模型的示意图

Fig 16: EKV Model

反型层形成后，根据反型层中的电荷密度，会形成两股方向相反的反型电流，分别从 S 流向 D 的电流 I_F 以及 D 流向 S 的电流 I_R。而 S 极和 D 极附近的反型电荷密度 q_S 和 q_D 反映了沟道的反型程度，也决定了沟道电流的大小。计算沟道电流 I_{DS} 的方程如公式（5.1）所示：

$$\begin{cases} V_p = \dfrac{V_G - V_{T0}}{n} \\ \dfrac{V_P - V_{S,D}}{U_T} = 2(q_{S,D} - 1) + \log(q_{S,D}) \\ I_{sat} = 2n\beta U_T^2 \\ I_{ds} = (i_F - i_R)I_s = \left[(q_S^2 + q_S) - (q_D^2 + q_D)\right] I_{sat} \end{cases} \tag{5.1}$$

其中 n 为一个与晶体管制造工艺有关的常数，U_T 为热电压，由温度决定，表达式为 $U_T = kT/q$，其中 k 为波尔兹曼常数，T 为器件所处的温度，q 为单位电荷。通过加在晶体管上的偏置电压 V_G、V_D 和 V_S，就可以从式（5.1.2）中计算得到源极附近的反型电荷密度 q_S 和漏极附近的反型电荷浓度 q_D。得到了 q_S 和 q_D 之后，就可以用公式（5.1.3）和（5.1.4）计算得到沟道电流 I_{ds} 的大小了。

不过公式（5.1）只是 EKV 模型最简单的形式，为了保证模型的精度，还需

要考虑一些高阶的效应，诸如沟道长度调制、迁移率下降、速度饱和等因素。但由于公式形式稍显复杂，可参考文献[]，限于篇幅就不再展开了。

值得注意的是，EKV 模型实质上综合了 MOS 管在强反型区的平方律的 I-V 特性与弱反型区的指数律的 I-V 特性：当$q_{S,D}$ 1时，公式（5.1.2）中的 1 和对数项就可以忽略，（5.1.4）的电流方程就退化成了平方律共识；而当$q_{S,D}$ 1时（5.1.2）中的对数项占据了主导，代入（5.1.4）就可以得到一个近似指数函数的方程。

在实际的软件实现中，由于 EKV 模型的一些工艺参数是和 BSIM3 相同的，因此可以直接从 HSPICE 的库文件中提取这些参数的值（如载流子迁移率μ、阈值电压V_{th}、单位面积的氧化层电容C_{ox}等）。所以在软件实现的过程中，参考了 TSMC 的 0.18μm 工艺库的参数提取了 EKV 模型所需要的参数，并实现了 EKV 的模型方程，用于数值仿真引擎来求取电路的直流工作点和电路的小信号参数及其灵敏度。

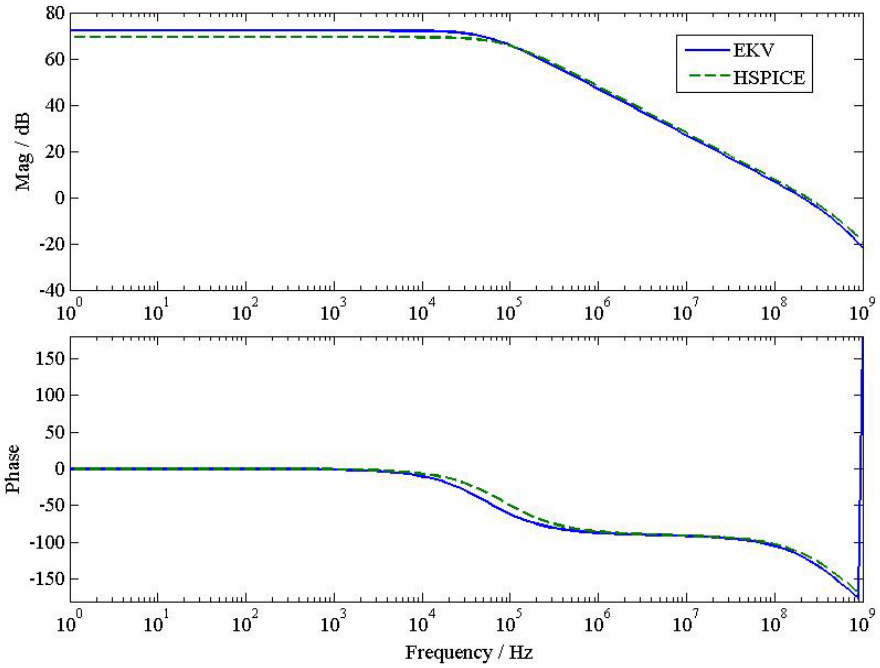


图 17：仿真结果和 HSPICE 的对比

Fig 17:Comparison with HSPICE Simulation Results

图 17 显示了用 EKV 模型与用 BSIM3.3 模型的 HSPICE 的仿真结果的对比，可以发现无论是幅度响应，还是相位的响应，本设计工具的结果在整个频域上都和 HSPICE 的结果比较接近，除了在直流增益和第一个极点的位置略有偏差。主

要原因还是限于模型较为简易，对 MOS 管的输出电阻 r_{ds} 的计算存在偏差。但对于一个用于运放设计的自动/互动设计工具而言，精度已经足够，在完成优化后只需要用诸如 HSPICE 的标准仿真工具检验微调下即可。

5.4 本章小结

本章主要介绍了基于 GPDD 的运算放大器设计工具的软件架构、主要的数据结构和算法以及仿真模型的提取。并提出实现了对 GPDD 算法的优化和改进，以方便后文对实验结果的描述和分析。

第六章 运算放大器电路设计应用实例

6.1 测试电路及测试环境

本运算放大器在 Linux 平台下使用 C++语言实现，并对以下的常见的运算放大器电路进行了测试，以验证算法的正确性、有效性以及执行的效率：

电路 1：二阶密勒补偿运算放大器（图 18）；

电路 2：套筒式共源共栅（Telescopic Cascode）运算放大器（图 19）；

电路 3：折叠式共源共栅（Folded Cascode）运算放大器（图 20）；

电路 1-3 是三种比较常用的运算放大器的拓扑结构，其中的所有偏置都使用了理想的电流源或者是电压源；来观察我们的设计工具对偏置电路设计的作用。

所有测试都在 EDA 实验室的服务器上完成，CPU 为 Intel Xeon 四核处理器，主频 1.8GHz，内存 16GB，操作系统为 RedHat Linux Enterprise 4。

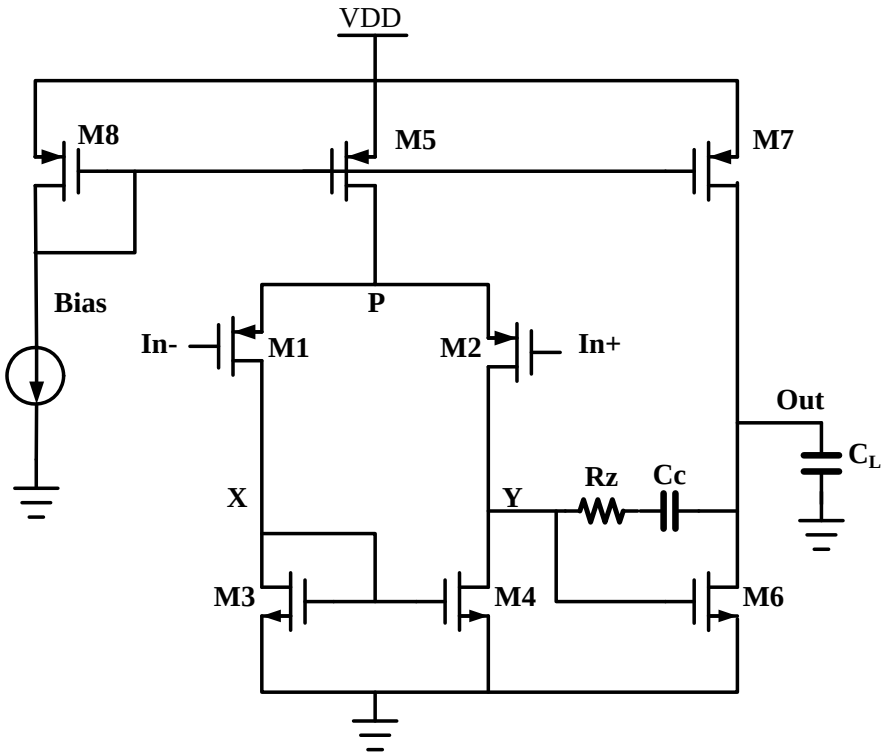


图 18：二阶密勒补偿运算放大器

Fig 18:Two-Stage Miller OPMAP

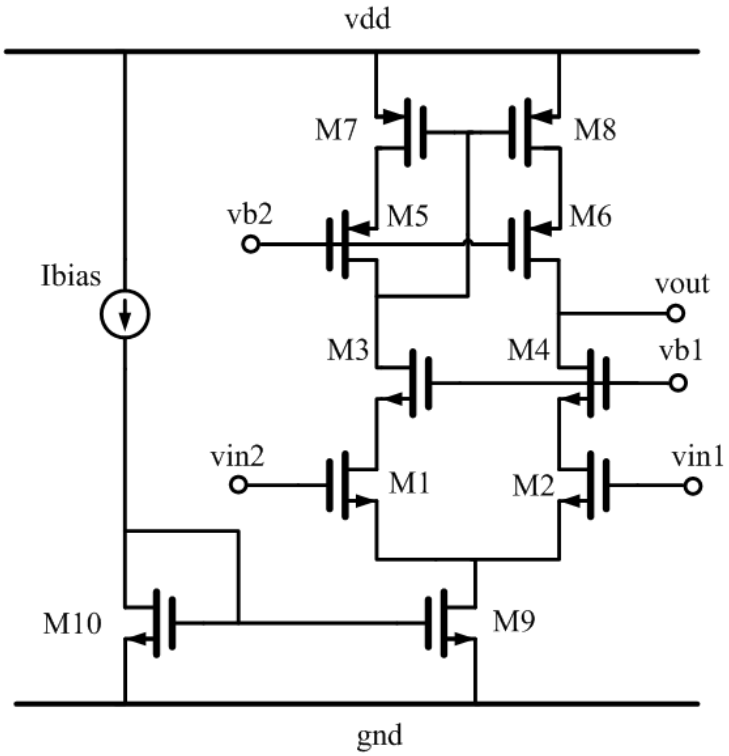


图 19：套筒式共源共栅运算放大器

Fig 19: Telescopic Cascode OPAMP

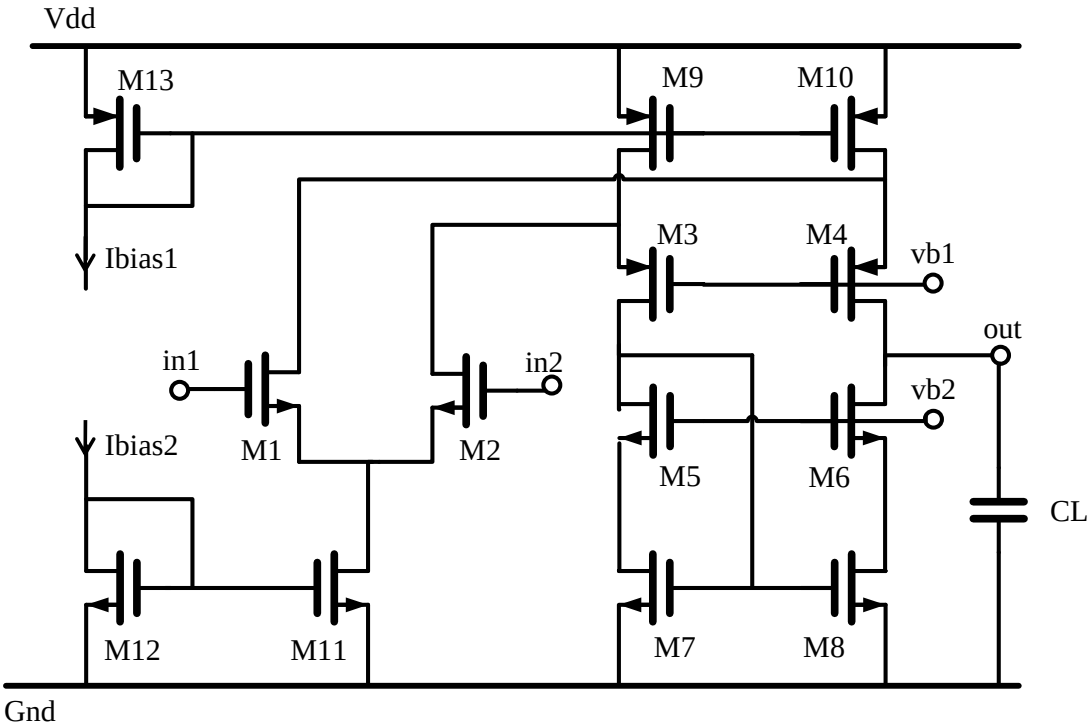


图 20：折叠式共源共栅运算放大器

Fig 20: Folded Cascode OPAMP

所谓的互动优化设计方法，就是在传统的设计方法的基础上，增加用符号化电路仿真器计算得到的灵敏度曲线所提供的直观的信息，来帮助模拟电路设计者找出影响电路性能的最主要的变量。电路设计者可以结合自身的经验和灵敏度曲线，来决定如何来对电路进行调整优化，使得电路能够达到性能指标的要求。

6.1.1 折叠式共源共栅运算放大器的互动设计

下面以通过灵敏度曲线互动优化的设计方法，来调整电路 3 中的折叠式共源共栅运算放大器（Folded Cascode OPAMP）的全过程来详细说明互动优化的设计方法学。

图 6.3 中所示的折叠式共源共栅运算放大其电路中，共有 13 个晶体管，除去用于拷贝理想电流源所使用的 M11 和 M12，有 11 个晶体管的尺寸可以用来调整。假设所有晶体管的沟道长度相同，并且所有差分晶体管对的尺寸也都相同，那么彼此独立的设计变量就只剩下 $W_1, W_3, W_5, W_7, W_9, W_{11}$ 这六个了。

对电路的指标设定如下：负载电容 $C_L = 2pF$，直流增益 $A_{DC} > 65dB$，单位增益带宽 $GBW > 200MHz$，相位裕度 $PM > 60°$，整个过程中电路的晶体管尺寸和指标的变化过程在表.2 中列出。下面就结合表.2 以及互动设计过程中所绘制的曲线，来详细说明通过灵敏度曲线的互动设计方法来调整优化电路的全过程。

	Step I	Step II	Step III	Step IV
$W_1 / \mu m$	20.0	30.0↑	45.0↑	40.0↓
$W_3 / \mu m$	2.0	5.0↑	15.0↑	15.0
$W_5 / \mu m$	2.0	3.0↑	3.0	3.0
$W_7 / \mu m$	2.0	3.0↑	3.0	3.0
$W_9' / \mu m$	15.0	10.0↓	10.0	12.0↑
$W_9 / \mu m$	45.0	50.0	55.0	54.5
$W_{11} / \mu m$	60.0	80.0↑	90.0↑	85.0↓
A_{DC} / dB	44.3	64.9	74.4	70.2
GBW / MHz	32.1	141	211	201
$PM / °$	87.4	67.2	55.7	60.9

表 2：互动优化过程中选定变量的变化过程

Table 2: The Process of Interactive Sizing

Step I：与传统方法类似，首先对电路进行简单的估算，得到晶体管的初始尺

寸，对其进行仿真，得到它的传输函数的响应曲线。如图 6.6 所示，并得到的电路的性能指标： $A_{DC}=44.3dB$， $GBW=32.1MHz$， $PM=87.4°$。

Step II：电路的直流增益和单位增益带宽的数值都没有达到性能指标的要求，并从仿真结果中发现，晶体管 $M_{9,10}$ 被偏置在了线性区，因而直流增益没有达到指标的要求。晶体管的偏置由其端口电压决定，并受周围晶体管尺寸的影响。对于 PMOS 晶体管 $M_{9,10}$，若需要将其偏置在饱和区，就需要降低它们的漏极（Drain）的电压 V_D，因此通过设计工具，绘制如图 22 所示的 V_D 关于各晶体管尺寸的灵敏度的柱状图。

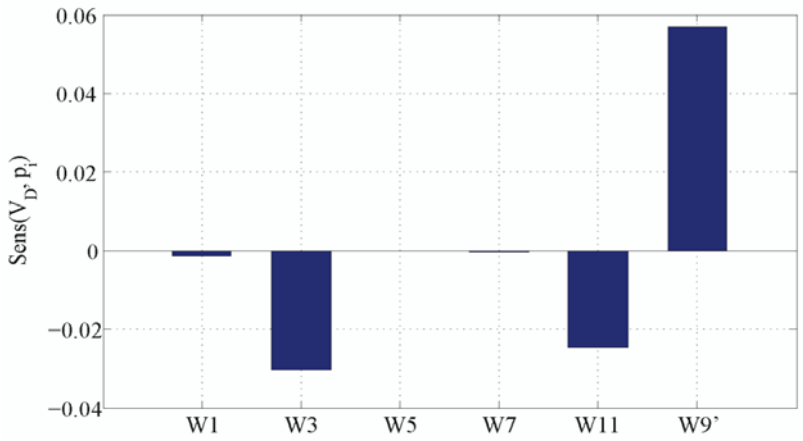


图 21：V_D 关于各晶体管尺寸的灵敏度的柱状图

Fig 21: Bar Chart of V_D with respect to All Variables

该柱状图的横坐标为被求取 DC 灵敏度的变量，纵坐标为灵敏度大小。柱状图的绝对值越大，则表示改变量对 V_D 的影响也就越大，反之则越小。柱状图的正负则表示这种影响的方向，正值表示正相关，负值为负相关。

从图 21 中可以观察到，V_D 关于变量 W_9' 的灵敏度最大，W_3 和 W_{11} 次之，而其他三个变量几乎没有影响。所以我们根据图 21 中的灵敏的数据和方向，决定增加 W_3 和 W_{11} 的值，减小 W_9' 的值。同时参考图 22 中绘制的电路的直流增益和各变量的灵敏度柱状图，发现上述变化的确是有利于提高电路的直流增益的，此外，增加 W_1 的数值也有利于提高电路的直流增益，因此我们得到了表 1 中第二列的数据。

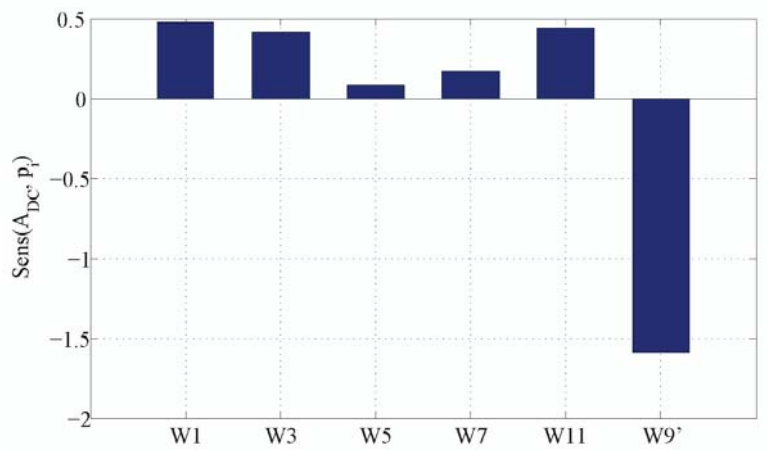


图 22：直流增益关于各量的灵敏度柱状图

Fig 22: Bar Chart of DC Gain with respect to all Variables

Step III：该电路的直流增益达到了指标的要求，但单位增益带宽 GBW 离 200MHz 的指标要求还有一定的差距。由于单位增益带宽 GBW 是一个频域上的指标，因此我们使用了设计工具中的数值及符号化仿真引擎，绘制了如图 24 的 AC 幅度关于个设计变量的灵敏度曲线：

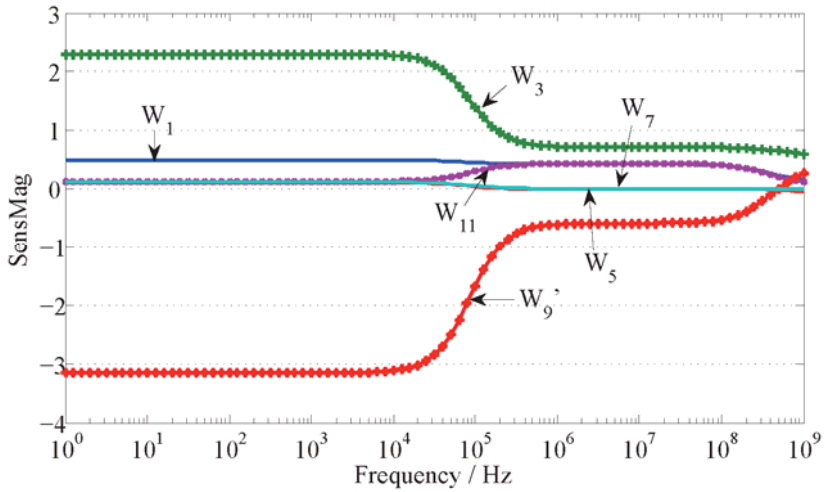


图 23：AC 幅度灵敏度曲线

Fig 23: AC Magnitude Sensitivity Curves

图 24 的 AC 幅度灵敏度曲线，横坐标是扫描的频率，纵坐标的绝对值反映了该变量在特定频率上对传输函数幅度的影响的大小，其符号也同样表示了这种影响的方向。

而电路得单位增益带宽 GBW 不满足指标要求，正是因为传输函数在高频段

的幅度太小所致。观察灵敏度曲线在高频度的响应，可以发现在高频段（100kHz—200MHz），传输函数的幅度相对变量w_3有较大的正相关的灵敏度，变量w_1和w_{11}次之，而变量w_5和w_7几乎没有影响；变量w_9'则有比较微弱的负相关性。因此需要提高电路的 GBW，在尽可能少的增加面积和功耗的情况下，就需要较大幅度地增加变量w_3的值，并适当增加变量w_1和w_{11}的值，稍稍减小变量w_9'的值，并基本维持w_5和w_7的值。由此我们得到了表.2 第三列中的晶体管尺寸和电路指标。发现直流增益和单位增益带宽都满足了指标的要求，但相位裕度反而没有能够达到指标的要求。

Step IV：为了使电路的相位裕度能够达到指标的要求，我们需要同时考虑电路的传输函数在高频段的幅度和相位。因此需要同时绘制幅度和相位的灵敏度曲线，如图 24 所示：

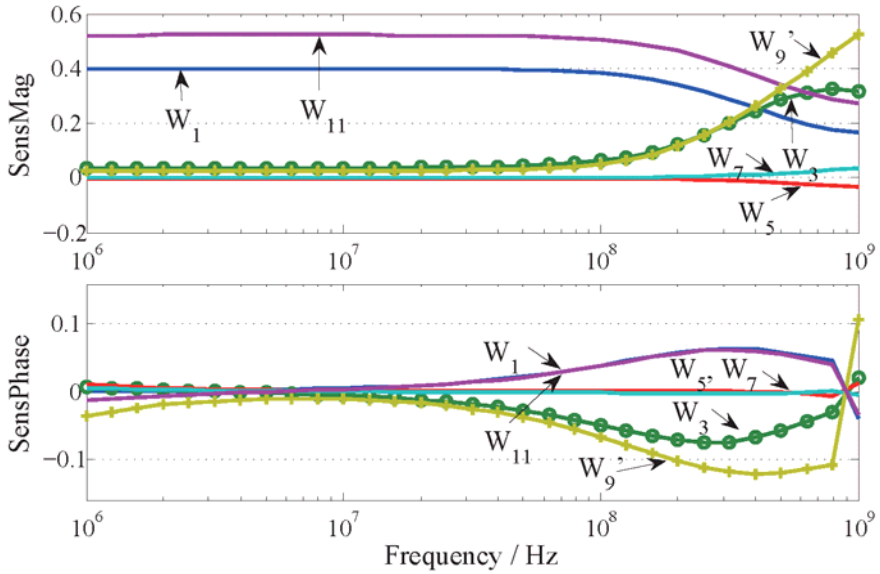


图 24：幅度和相位灵敏度曲线

Fig 24: Sensitivity Curves of Magnitude and Phase

由于电路的 GBW 已经超过了指标的要求，因此为了增加电路的相位裕度，可以适当的牺牲一些 GBW，以换取电路相位的增加。从图 24 中可以观察到，电路在高频段的相位和w_3,w_9'负相关，与w_1,w_{11}正相关；而在高频段的幅度又与这些变量都正相关，因此为了提高电路的相位裕度，需要减小电路的相位和幅度，根据曲线权衡后，就需要减小变量w_1,w_{11}的值，并适当增加变量w_3,w_9'的数值，这就得到了表 2 中第四列中的晶体管尺寸以及电路指标，发现电路指标都已满足。

6.1.2 其他案例及总结

表 2 展示了利用类似 6.2.1 节中所说的方法，结合模拟电路设计经验对 1-3 这三个电路进行互动设计后的结果的比较，可以发现互动设计很好的指导了模拟电路的设计者，使他们很好地发现影响电路性能的主要设计参数，提高电路设计优化的效率。

	Case I		Case II		Case III	
	Initial	Final	Initial	Final	Initial	Final
A_{DC} / dB	79.9	**75.4**	42.0	**67.3**	43.1	**69.1**
GBW / MHz	55.2	**196**	109	**203**	49.2	**199**
PM / °	53.7	**66.1**	**82.4**	**70.3**	**86.6**	**60.2**
$\sum W$ / μm	945.7	981.5	279.1	418.9	157.9	322.9
$I(vdd)$ / μA	172.2	613.7	484.9	444.7	461.8	772.8

表 3：互动设计的结果比较

Table 3: Results of Interactive Sizing

从表 3 中数据中可以得知，在这三个设计案例中，通过灵敏度曲线辅助互动设计，并结合设计者的经验，都使得电路能够在迅速达到指标的要求，并且面积和功耗都在可以接受的范围内。

事实上，灵敏度曲线设计的方法，和传统的通过小信号分析得到近似公式进行设计的方法是类似的，只是相比于经过近似的小信号分析的结果，灵敏度曲线提供了更多更精确的设计信息，来指导模拟电路设计者对电路进行优化。

例如通过传统的电路的小信号分析[7]，可以求得图 20 中的折叠式共源共栅电路的单位增益带宽 $GBW = g_{m1} / 2\pi C_L$，可以推断出通过增加晶体管 $M_{1,2}$ 的宽度或者偏置电流就可以增加电路的带宽。而图 23 中的响应曲线也反映了这一点。但从图 23 中还可以发现 W_3 对带宽也有很大的影响，主要就是由于在手工分析中被忽略的 cascode 的晶体管 $M_{3,4}$ 的寄生电容，起到了分离主极点和次极点的作用，这个被忽略的因素就在灵敏度曲线中显示了出来。

综上所述，基于灵敏度曲线的互动设计方法可以帮助模拟电路设计者更好的理解电路的结构以及一些寄生参数对电路的影响，结合传统的小信号分析以及设计者的经验，提高运算放大器电路设计的效率。

6.2 自动优化设计方法

本次设计工具的软件实现的过程中，实现了基于灵敏度分析的自适应模拟退火算法，并分别针对三个测试电路进行了自动优化的测试，来检验自适应模拟退火算法的有效性。

在当前的软件实现中，为了便于测试，电路的性能指标暂时只考虑放大器的直流增益、单位增益带宽、相位裕度、面积以及功耗等五项指标。模拟退火算法中用来计算“能量”的公式也是考虑了这些因素后产生的，如公式 6.1 所示：、

$$\min E(x) = k_1 \frac{I_{vdd}}{I_{vdd}^{(0)}} + k_2 \frac{A}{A^{(0)}} + \sigma_1 P_1(x) + \sigma_2 P_2(x) + \sigma_3 P_3(x) \qquad (6.1)$$

其中前两项分别表示电路的功耗和面积与初始的功耗和面积的比值，k_1、k_2 分别表示了功耗优化和面积优化在优化过程中所占的权重，可由用户根据自身的要求设定；另外 $P_i(x)$ 表示优化过程中使用的外点惩罚函数（Penalty Function）[]，分别对应于运算放大器的直流增益、单位增益带宽和相位裕度的指标要求，一般为分段函数：

$$P_i(x) = \begin{cases} c_i - f_i(x), f_i(x) < c_i \\ 0, f_i(x) \geq c_i \end{cases} \qquad (6.2)$$

当电路的指标 $f_i(x)$ 没有达到放大器的性能指标 c_i 时，则计算它们之间的差值作为惩罚函数；而当该项指标达到要求时，惩罚函数则置为零。需要注意的是，惩罚函数的权重系数 σ_i 一般会取一些较大的数以使得电路尽可能快地达到性能指标的要求，也可由用户根据需求来指定。

还有一个用于控制模拟退火算法收敛性的变量就是优化过程中的“温度”：T。在实现中，每一次模拟退火迭代后，都会有一个在 $(0,1)$ 的系数乘上 T 来实现降温的过程，即 $T^{(n+1)} = \rho T^{(n)}, \rho \in (0,1)$。在实际使用中，$\rho$ 的取值要适当：ρ 太大算法地收敛会减慢，但优化的结果更接近于最优值；而 ρ 减小虽然能加快收敛，但往往算法终止时，目标函数值离最优值得距离会更远。

由于模拟退火算法是一个随机算法，在优化过程中的每一步都需要在原先的解向量的邻域内产生一个随机的新的解向量，因而每次模拟退火算法求出的最优解会互不相同，但可以渐进地依概率地收敛于一个全局的最优解，因此需要使用蒙特卡洛（Monte Carlo）的方法来比较自适应模拟退火算法和模拟退火算法的效率。

以下为蒙特卡罗测试的结果：

Case I：二阶密勒补偿运算放大器（Two-Stage Miller OPAMP）：

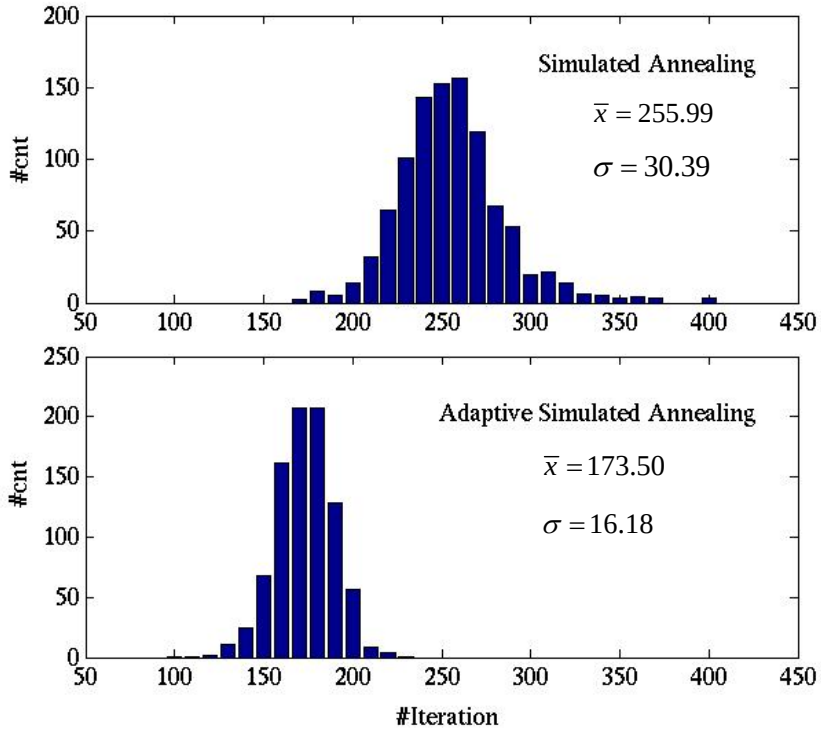


图 25：Case I：两种算法迭代次数的分布比较

Fig 25: Case I: The Distribution of Iteration Counts

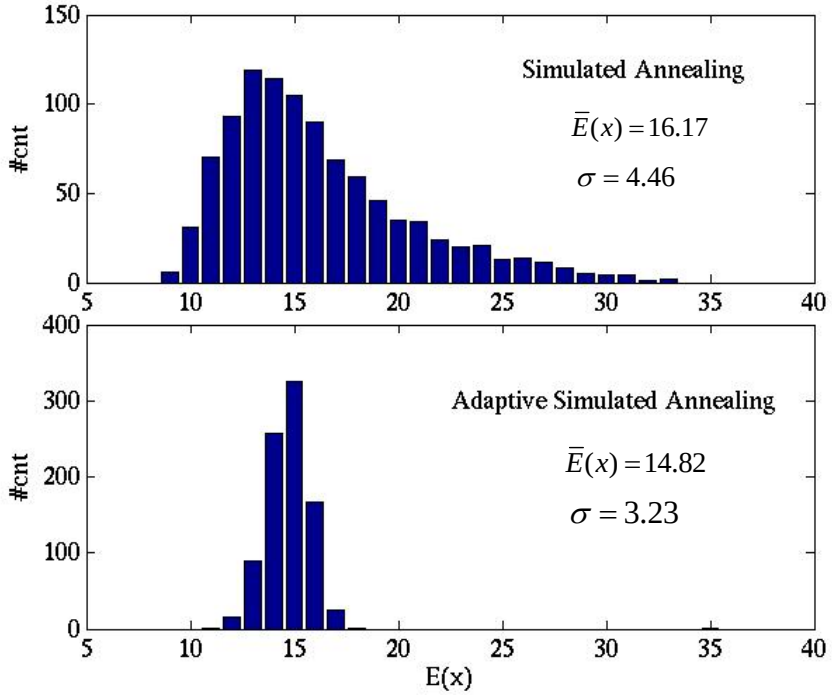


图 26：Case I：两种算法目标函数的分布比较

Fig 26: Case I: The Distribution of Final Target Function

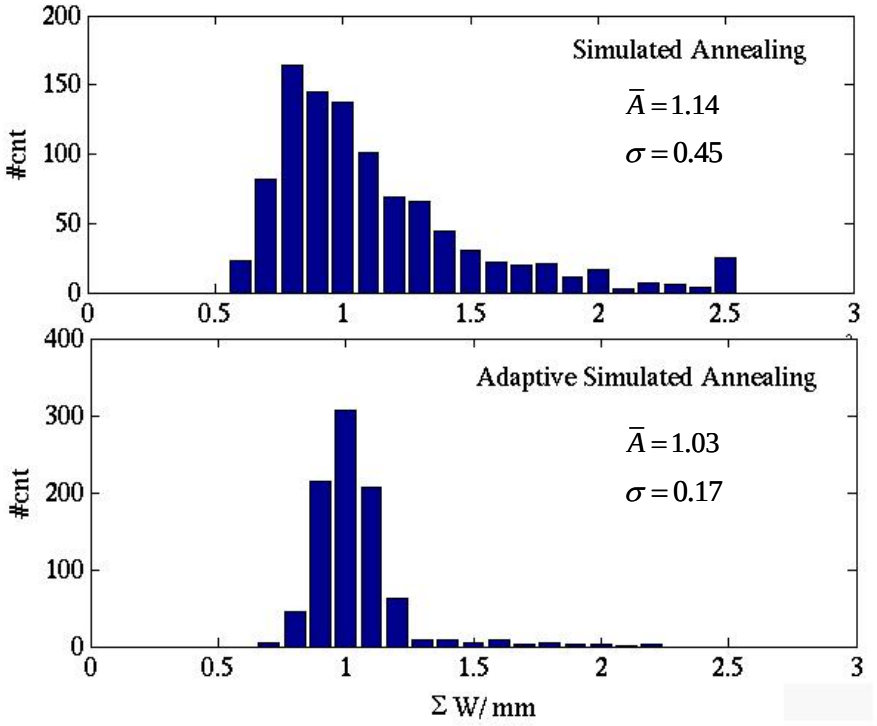


图 27：Case I：两种算法所得的电路晶体管长度和的分布比较

Fig 27:Case I: Distribution of Total Transistor Width

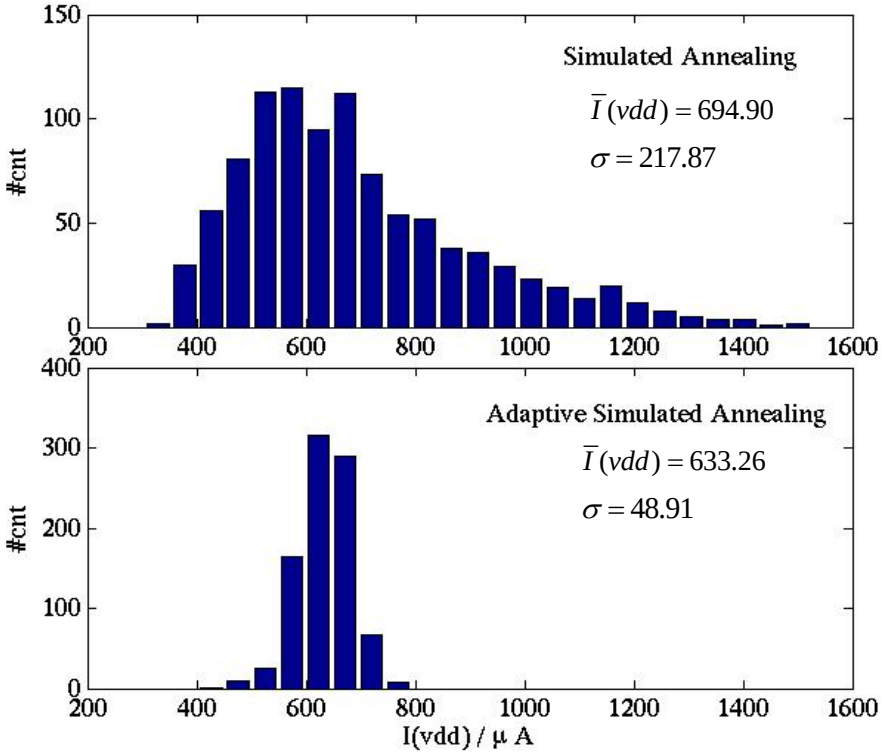


图 28：Case I：两种算法所得的电路功耗的分布比较

Fig 28: Case I: The Distribution of Power Dissipation

Case II：套筒式共源共栅运算放大器（Telescopic Cascode OPAMP）

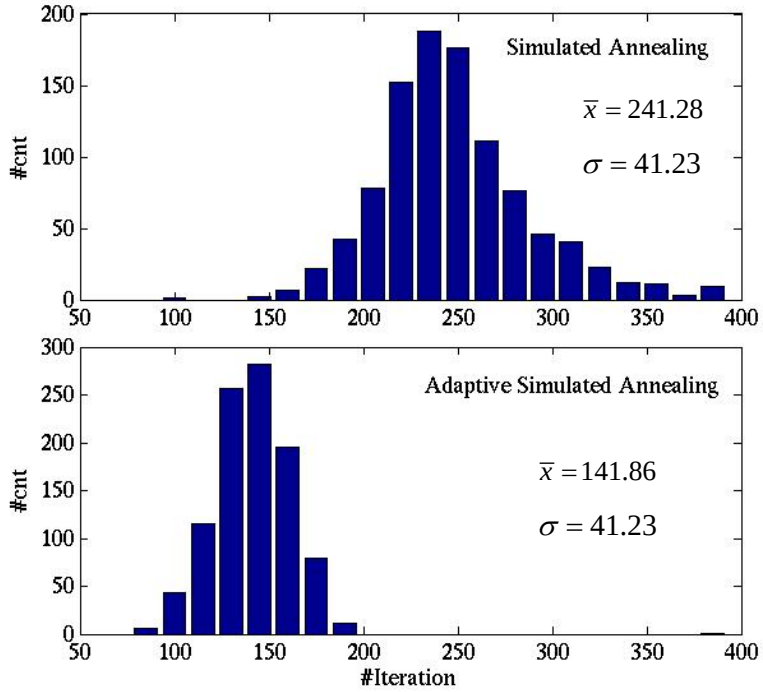


图 29：Case II：两种算法迭代次数的分布比较

Fig 29: Case II: The Distribution of Iteration Counts

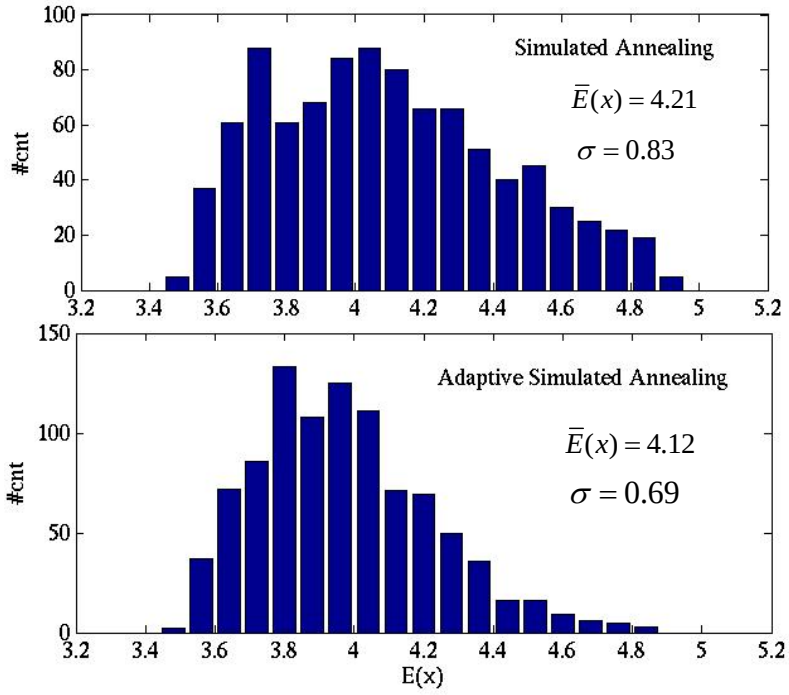


图 30：Case II：两种算法所得目标函数分布比较

Fig 30: Case II: Distribution of Final Target Function

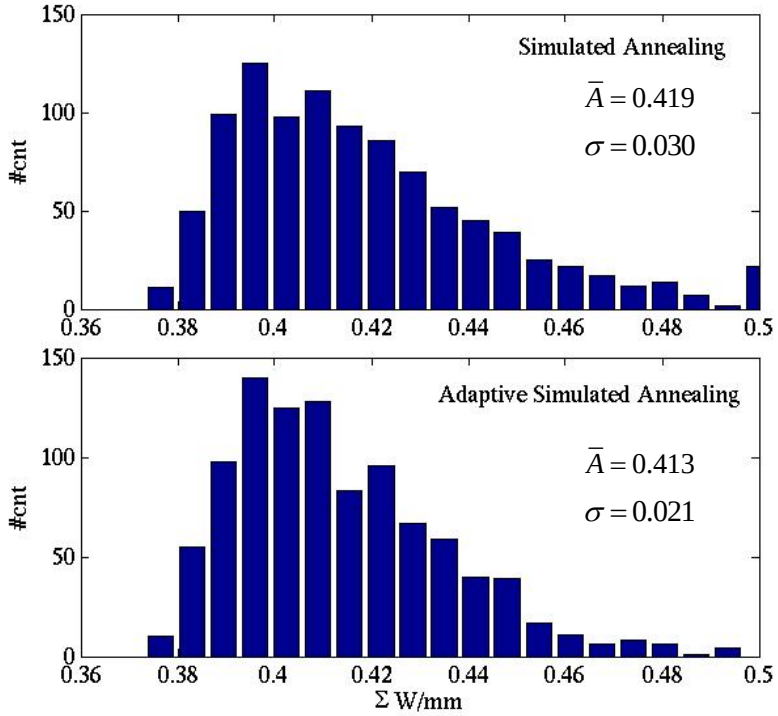


图 31：Case II：两种算法所得电路的面积分布比较

Fig 31: Case II: Distribution of Total Transtor Width

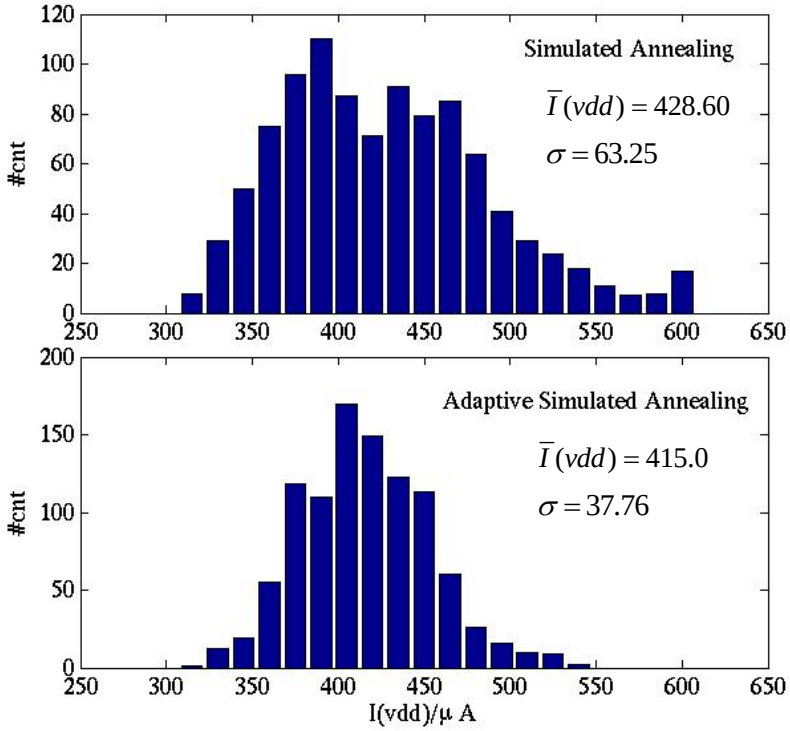


图 32：Case II：两种算法所得电路的功耗分布比较

Fig 32: Case II: Distribution of Power Dissipation

Case III：折叠式共源共栅运算放大器（Folded Cascode OPMAP）

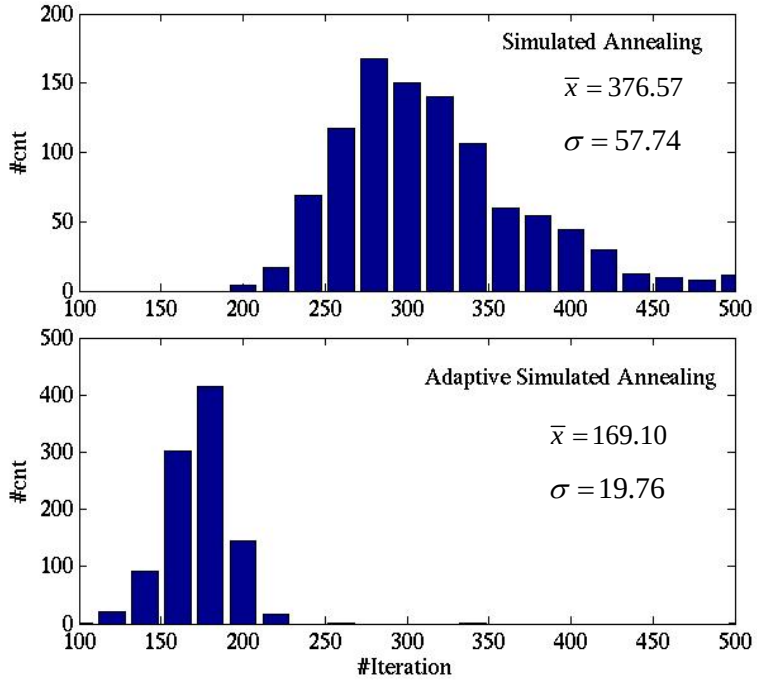


图 33：Case III：两种算法迭代次数的分布比较

Fig 33: Case III: Distribution of Iteration Count

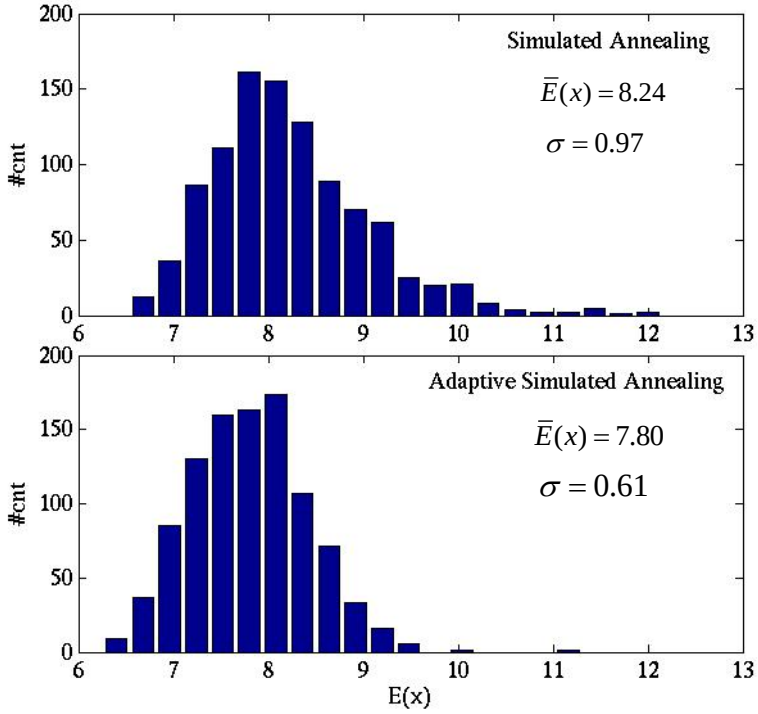


图 34：Case III：两种算法所得目标函数的分布比较

Fig 34: Case III: Distribution of Final Target Function

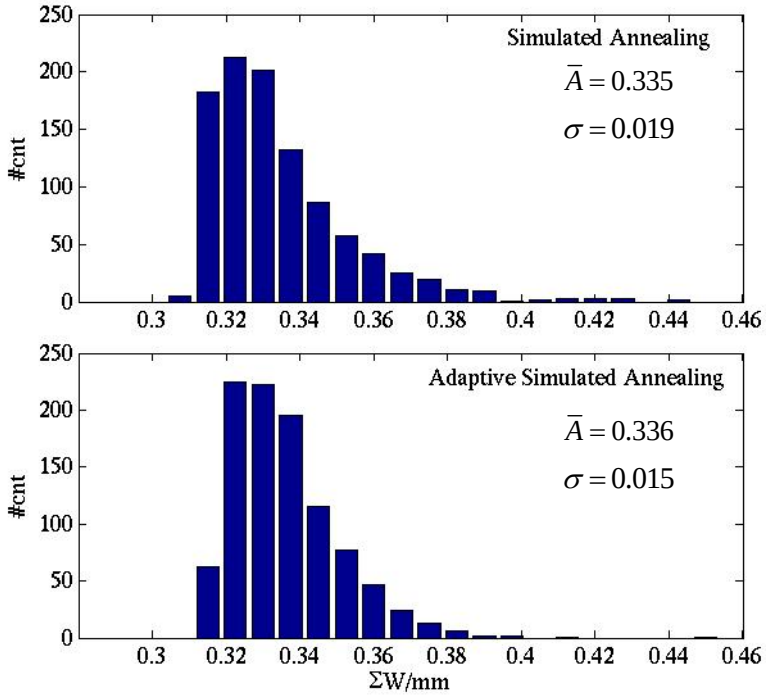


图 35：Case III：两种算法所得电路面积的分布比较

Fig 35: Case III: Distribution of Total Transistor Width

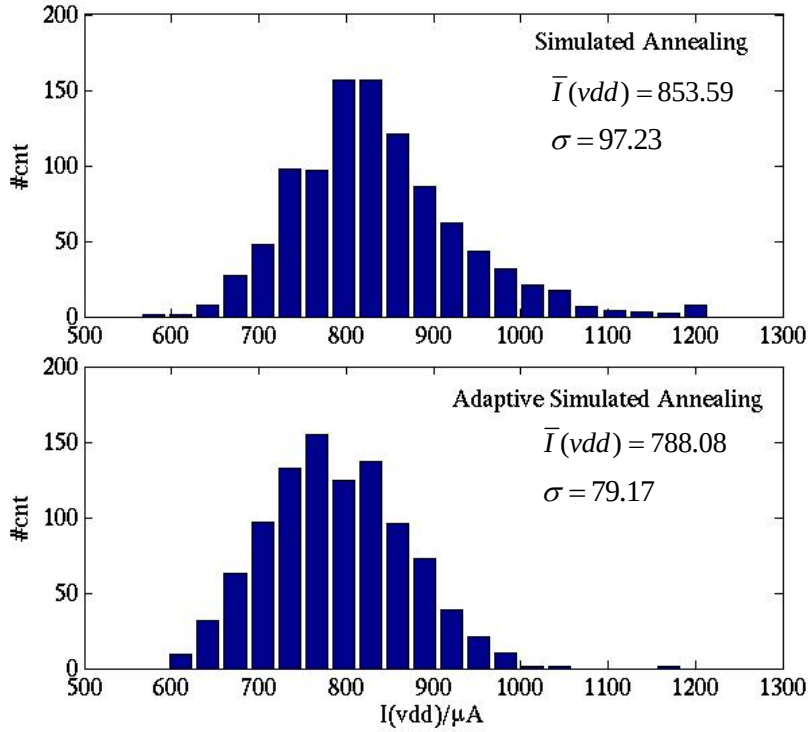


图 36：Case III：两种算法所得电路功耗的分布比较

Fig 36: Case III: Distribution of Power Dissipation

测试时，对每个测试电路分别使用模拟退火算法和自适应模拟退火算法，分别对测试电路进行 1000 次优化，观察最后收敛的结果中算法的迭代次数、电路面积、功耗和性能指标的分布状况，来比较和判断两种算法的优劣。

对于每一个设计案例，要分四个方面比较这两种算法运行的结果：迭代次数、算法终止时的目标函数值、电路的面积、电路的功耗这四个方面的分布状况。

每一个测试案例的四个结果图（图 23-36），分别展示了 1000 次蒙特卡洛运行后，算法的迭代次数、目标函数值、电路的面积以及功耗在相应区间上的分布。可以观察到，对于两种算法的这四个指标的分布都非常接近于正态分布，在引入灵敏度分析的自适应算法后，在其他条件（如降温系数、目标函数、收敛条件等）相同的情况下，相比简单的模拟退火算法有着更好的收敛性。并且算法收敛后得到的电路有更好的表现，平均来说，优化所用的时间更少，面积更小，功耗也更低，并且各次优化的收敛次数以及各次之间的性能指标间的标准差也更小，如表 4 中的总结：

	Case I		Case II		Case III	
	SA	ASA	SA	ASA	SA	ASA

#Iteration	255.99	**173.50**	241.28	**141.86**	316.57	**169.10**
Time / s	4.09	**1.96**	12.53	**9.97**	9.01	**4.87**
E(x)	16.17	**14.82**	4.21	**4.12**	8.24	**7.80**
$\Sigma W / \mu m$	1140	**1030**	419	**413**	**334.65**	336.02
$I(vdd) / \mu A$	694.90	**633.26**	428.6	**415.50**	835.59	**788.08**

表 4：SA 与 ASA 算法的执行结果比较

Table 4: Comparison of the Test Results of SA and ASA

由此可见，在模拟退火算法的基础上，引进符号化仿真器获得的灵敏度信息，来指导自适应模拟退火算法中新的随机解的产生，可以显著提高模拟退火算法的执行效率。即使自适应模拟退火算法中，计算电路的灵敏度需要花费一些时间，但当迭代次数显著减少以后，这些代价对优化时间的影响基本可以忽略不计，可见自适应模拟退火算法对时间效率的改善是很明显的。

注意到算法运行时间的加速比要大于迭代次数减少的比例，这可能的原因引入了灵敏度分析来指导新的随机解的生成，虽然需要一些额外的代价，但使用这种方法生成的电路的参数一般来说会走向更正确的方向，DC 解的牛顿迭代的收敛也会更容易些，因此算法的时间加速比超过了迭代次数的加速比。

不过，使用自适应地模拟退火算法，虽然效率有了很大的提高，但往往仍需要上百次的迭代才能找到全局的最优解。当电路比较小的时候，能够比较快的寻找到最优解；但当电路规模增大后，上百次的迭代会消耗非常多的时间，因此算法还需要一些改进。

6.3 两种设计方法比较

在本运算放大器设计工具中，在符号化电路仿真引擎进行灵敏度分析的基础上，实现并测试了基于灵敏度曲线的互动设计方法以及基于灵敏度分析的自适应模拟退火算法，都取得了不错的结果。

这是两种完全不同的设计方法。互动设计方法是在传统的模拟电路设计方法的基础上，增加了频域上的灵敏度曲线作为参考，来帮助设计者更好地了解影响电路的主要因素。相比经典的近似公式，灵敏度曲线提供了更丰富的信息，方法学上又与传统的方法相类似，比较容易被接受，对设计的效率有所提高，但提高得不多。而且当电路中变量增多以后，过于丰富的灵敏度信息不易进行人工分析，也会影响到设计的效率。

而自适应模拟退火算法的自动设计方法，直接集成了灵敏度分析，具有全局

的收敛性，可以“半智能”地用类似互动设计的方法自动地找出电路的最优解。电路设计者则只需要设定好各个指标之间的权重，给定一个比较正常的初值，算法就可以自动地找出一个近似最优解，在达到电路设计性能指标要求的同时，使得电路的面积和功耗达到最小。但这一过程又隐藏了许多电路设计中的细节，与一般的设计方法相差较远，不易被接受。

因此在本设计工具中，用户可以选择使用自动或者互动的设计方法，来决定如何优化电路：比如可以先用互动设计的方法找到一个比较好的初值，再利用自动优化方法找到一个更优的解；或者先用自动优化算法找到一个比较好的优化范围，再根据互动优化方法提供的曲线以及自己的经验进一步的优化电路。

6.4 本章小结

本章针对三个运算放大器的设计案例，分别使用设计工具中的互动优化的方法和自动的优化方法，对这三个案例进行了优化，比较了它们优化的结果和效率，验证了上述方法的有效性。

第七章 总结与展望

7.1 总结

本文在已有的符号化电路仿真器和简单的数值电路仿真器的基础上，通过符号化仿真器与数值仿真器相结合的方法，仿真器内部实现了 EKV 模型的方程，成功地实现了对电路中晶体管尺寸求取精确的灵敏度的算法。并在此基础上，实现了基于电路灵敏度曲线的互动设计方法，以及基于自适应模拟退火算法的自动优化算法，完成了一个简单的运算放大器的设计工具。

在设计工具完成后，分别针对三种常见的运算放大器的拓扑结构，分别进行了基于符号化灵敏度的互动设计和自动化的设计，取得了不错的结果。设计工具能够在可接受的时间内返回符合电路指标要求的电路，并能使得电路的面积或功耗尽可能的小。

7.2 不足与展望

本文对符号化电路仿真器在模拟电路设计工具中的应用做了一个初步的探索。由于符号化仿真器只能对线性电路进行仿真，因此还需要传统的 SPICE 仿真器相结合才能完成。无论是互动设计方法，还是自动的设计方法，都需要不断的对电路进行求值和求导操作，但对于运算放大器电路，GPDD 产生的节点相当之多，影响到了 GPDD 求值和求导的效率，在今后的设计中，可以考虑对 GPDD 的求值操作并行化，以进一步提高设计工具的效率。

对于互动设计方法，是在传统的设计方法上引入了电路的符号化 ac 灵敏度曲线作为参考，来帮助设计人员完成设计。然而，如果电路变得更为复杂，需要优化的变量变多时，灵敏度曲线的信息就会显得过于丰富了，会使得集成电路设计者无从下手。但是如果能通过某种内部的机制，过滤掉一些意义不大的曲线，才能使得灵敏度曲线更有意义。

对于自适应模拟退火的自动优化算法，虽然相比传统的模拟退火算法有所改进，但并效果不显著，只是减少了约一半的迭代时间，应该还有进一步提高优化的空间。此外，由于自动优化算法是一个黑盒，电路设计者无从知道其内部的优化策略，也无法得知其中蕴含的电路信息，不太容易被接受。如果能够有方法把

优化的过程形象化的表示出来，并且能够结合互动方法中的曲线所提供的信息，则能够在高效优化电路的同时揭示电路的本质，这样才能被更广泛地接受。

此外在数值仿真器方面，EKV 的模型方程存在一些问题，在某些特定条件（如极低电压下的若反型区）存在不收敛的状况，因而没有能够测试弱反型区的运算放大器与 g_m/I_D 方法作比较和结合。

总体来说，这次基于符号化仿真器的运算放大器设计工具的研究，并不是摆脱传统上的反复仿真迭代寻找最优的方法，而是利用符号化仿真器的特点，将电路的灵敏度分析的结果应用到这一过程中，企图提高运算放大器电路设计的效率。这次初步的尝试只考虑了电路的面积、功耗、直流增益、单位增益带宽和相位裕度，其他的一些指标诸如共模抑制比（CMRR），电源抑制比（PSRR），噪声（Noise），大信号特性（Slew Rate, Settling Time）等进行分析。今后应该在此基础上进行添加。使其成为一个更完整的设计工具。

参 考 文 献

[1] B. Martin, “Automation comes to analog”, *Spectrum of IEEE*, vol. 38, issue 6, pp. 70-75, Jun., 2001.

[2] L.T. Pliiage, R.A. Rohrer, C. Visweswariah, Electronic Circuit and System Simulation Methods, McGraww-Hill, New York, 1994.

[3] G. Gielen, H. Walscharts and W.M. Sansen, “Analog circuit design optimization based on symbolic simulation and simulated annealing”, *IEEE Journal of Solid-State Circuits*, vol. 25, no. 3, pp. 707-713, Jun., 1990.

[4] J. R. Koza, F.H. Bennett III, D. Andre and M. A. Keane, “Automated design of both topology and sizing of analog electrical circuits using genetic programming”, *Proceedings of the First Annual Conference on Genetic Programming*, 1996.

[5] M. del M. Henderson, S.P. Boyd, T. H. Lee, “Optimal design of a CMOS op-amp via geometry programming”, *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no.1, pp. 1-21, Jan., 2001.

[6] R. Schwencker, J. Eckmueller, H. Graeb and K. Antreich, “Automating the sizing of analog CMOS circuits by consideration of structure constraints”, In *Design Automation and Test in Eurpe, Munchen, Germany*, 1999, pp. 323-327.

[7] B.Razavi, 陈贵灿等译, Design of Analog CMOS Integrated Circuits, 西安：西安交通大学出版社, 2002

[8] 陈微微，“符号化模拟电路仿真器的实现与应用”，上海交通大学微电子学院硕士论文，2007 年 3 月。

[9] G. Shi and X. Meng, “Variational analog integrated circuit design by symbolic sensitivity analysis”, In *Proceedings of the international symposium on circuits and systems (ISCAS)*, pp. 3002-3005, Taiwan, China, May, 2009.

[10] H. Yang, A. Agarwal, and R. Vemuri, “Fast analog circuit synthesis using multi-parameter sensitivity analysis based on element-coefficient diagrams”, In *Proceedings of the IEEE computer society annual symposium on VLSI*, pp. 71-76, Tampa, Florida, USA., 2005.

[11] D. Ma, G. Shi, A. Lee, “A design platform for analog device size sensitivity analysis and visualization”, In *Proceedings of Asia Pacific conference on circuits and systems (APCCAS)*, pp. 48-51, Malaysia, Dec. 2010.

[12] X. Li, H. Xu, G. Shi, and A. Tai, “Hierarchical symbolic sensitivity computation with applications to large amplifier circuits”, In *Proceedings of International Conference on Circuits and Systems (ISCAS)*, pp. 2733-2736, Rio de Janeiro, Brazil, May., 2011.

[13] 陈家俊，”用于电路仿真器的现代迭代求解器的实现”，上海交通大学微电子学院学士论文，2010 年 6 月。

[14] 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, Feb., 1994.

[15] 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, Jan., 2000.

[16] R. E. Bryant, “Graph-based algorithms for Boolean function manipulation,” *IEEE Trans. Comput.*, vol. C-37, pp. 677-691, Aug., 1986.

[17] G. Shi, W. Chen and C.-J. Shi, “A Graph Reduction Approach to Symbolic Circuit Analysis,” in *Proc. Asia and South-Pacific Design Automation Conference (ASPDAC)*, Yokohama, Japan, pp. 197-202, Jan., 2007.

[18] W. Chen and G. Shi, “Implementation of a Symbolic Circuit Simulator for Topological Network Analysis,” in *Proc. IEEE Asia Pacific Conference on Circuit and System (APCCAS)*, Singapore, pp.1327-1331, Dec., 2006.

[19] H. Xu, G. Shi, and X. Li, “Hierarchical exact symbolic analysis of large analog integrated circuits by symbolic stamps”, In *Proceedings of the Asia South-Pacific Design Automation Conference (ASPDAC)*, pp. 19-24, Yokohama, Japan, Jan., 2011.

[20] 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.

[21] S. Minato, “Zero-suppressed BDDs for set manipulation in combinatorial problems”, In *Proceedings of the 30th IEEE/ACM Design Automation Conference*, pp. 272-277, Dallas, TX, 1993.

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

[23] G. Shi, “A survey on binary decision diagram approaches to symbolic analysis of analog integrated circuits”, In *Analog integrated circuits and signal processing*, Sep., 2011.

[24] H. Yang, M. Ranjan, W. Verhaegen, M. Ding, R. Vemuri and G. Gielen, “Efficient

symbolic sensitivity analysis of analog circuits using element-coefficient diagrams", In *Proceedings of the Asia South-Pacific design automation conference (ASPDAC)*, pp. 230-235, Jan., 2005.

[25] A. Giradi, F. P. Cortes, S. Bampi, "A tool for automatic design of analog circuits based on gm/Id Methodology", In *Proceedings of IEEE International Symposium on Circuit and System(ISCAS)*, vol., no., pp.4 pp., 21-24 May 2006

[26] 陈宝林，"最优化理论与算法（第二版）"，北京：清华大学出版社，2005

[27] S. Kirkpatrick, "Optimization by simulated annealing: quantitative studies", Journal of Statistical Physics, vol. 34, No. 5, 1984

[28] L. Ingber, "Adaptive simulated annealing: lesson learned", *Control and Cybernetics*, vol. 25, pp. 33-54, 1996

[29] A.V. Aho, R. Sethi, J. D. Ullman, *Compilers Principles, Techniques and Tools*, Addison-Wesley, Reading, MA, 1986

[30] Y. Saad, M.H. Schultz, "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems", *SIAM J. Sci. Statist. Comput.* vol. 7 pp. 856–869, 1986

[31] G. Shi, "Introduction to EDA", Courseware of School of Micro-electronics, SJTU

[32] C. Enz, F. Krummenachaer, A. Vittoz, "An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications", In *Analog integrated circuits and signal processing*, pp, 83-114, 1995.

[33] J.R. Brew, "A charge sheet model for the MOSFET", *Solid State Electronics*, vol. 21, issue 21, pp.345-355

[34] P. G. A. Jesper, The gm/Id Methodology, A Sizing Tool for Low-Voltage Analog CMOS Circuit, Springer, New York, 2010.

致　谢

时光荏苒，转眼间又一次迎来了毕业的季节，而我已经在交大微电子学院度过了近七年时光，从本科到研究生，在收获颇丰的同时，也不禁感叹时光的飞逝。

首先要感谢父母这二十多年的养育之恩，以及在大学的这些年的全力地帮助和支持。给予了我一个坚强的后盾，能够全心全意地攻读学位，做自己喜欢的事情。

当然最要感谢的就是我研究生学习研究阶段的导师施国勇教授。他治学严谨，勤奋务实的作风令我映像深刻。从本科四年级的 EDA 引论课开始，一步步把我带入 EDA 的美妙世界，学习了解了各种电路仿真优化的算法，指导我开展研究，帮助我修改论文并投稿，参加学术会议，并完成了最后的毕业设计，使我第一次理解到一个学者是该如何做研究的。同时我在毕业后能够去 Synopsys 公司工作，继续在 EDA 领域进行软件开发与研究，也离不开他的推荐和帮助。

还要感谢微电子学院的各位老师们，这些年来在各门课程上对我的指导。让我对 EDA 方向以外的知识有了更深入全面地了解，帮助我能够更全面地看待、处理研究上碰到的各种问题。印象尤其深刻的是李章全老师的专业英语课，通过电影配音练习，对我们口语和演讲技巧的提高有一定的帮助，并且能够结合本专业的知识和自己的传奇故事来讲解，让人受益匪浅；还有体系结构的祝永新教授、模拟射频实验室的周建军教授、数字电路实验室的何卫峰、谢憬老师等。

当然也不会忘记在微电子学院一起成长的同学们，你们的陪伴使我的大学以及研究生的生活充满了各种乐趣，一起讨论问题攻克难关、一起运动、一起腐败还有晚上学习工作结束后各种欢乐的三国杀及电子游戏活动，这七年的学习生活会是一生的财富。

在闵行近七年的学习生活即将进入尾声，2012 不会成为世界末日，而会成为一个新的开始。感谢交大，感谢我的父母、导师和同学们，让我能在毕业后以一个很好的起点，我会在今后的工作学习生活中继续努力，创造一个更美好的未来。

攻读硕士学位期间已发表或录用的论文

[1] Jiajun Chen, Guoyong Shi, Andy Tai and Frank Lee, “A size sensitivity method for interactive MOS circuit sizing”, *New Circuits and Systems Conference (NEWCAS), 2012 IEEE 10th International* , vol., no., pp.169-172, 17-20 June 2012