上海交通大学硕士学位论文

用于模拟电路模块设计的多层次符号化分析新算法研究与软件实现[1]

硕士研究生：徐辉

学　　号：1092109016

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：2012 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）赞助。

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

A Hierarchical Symbolic Simulator for Analog Circuit Design

Candidate：	Hui Xu
Student ID:	1092109016
Supervisor：	Prof. Guoyong Shi
Academic Degree Applied for：	Master of Science
Speciality：	Circuits and System
Affiliation：	School of Microelectronics
Date of Defence：	Jan, 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《用于模拟电路模块设计的多层次符号化分析新算法研究与软件实现》，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：

日期：　　年　　月　　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

用于模拟电路模块设计的多层次符号化分析新算法研究与软件实现

摘要

由于模拟集成电路设计的复杂性与经验性，其辅助设计软件的自动化程度一直不高，只能为设计者提供少量的反馈信息，导致设计者往往要通过反复的电路仿真才能确定电路参数，拉长了电路的设计周期。为此，本文提出了一种针对模拟集成电路的多层次符号化分析仿真器，旨在通过符号化的仿真信息进一步指导电路的设计，提高模拟电路设计的效率。

该仿真器结合了基于拓扑网络的符号化分析及基于行列式的符号化分析这两种方法，为每个子电路构建符号化的导纳矩阵，并符号化地求解顶层的电路系统矩阵，从而实现了电路层面的共享，进一步提升了仿真器的处理大型模拟电路的能力。

本文的第一章首先介绍课题背景，包括课题的由来，符号化仿真的现状；第二章详细介绍了多层次符号化分析的理论算法基础；第三章给出了符号化仿真器的具体实现，包括仿真器的架构，主要数据结构与算法的实现；第四章给出了仿真器性能的测试结果及一些应用实例；第五章总结全文，并展望了基于该仿真器的进一步改进与应用。

关键词：多层次符号化仿真、模拟集成电路、多端口导纳矩阵、二分决策图

A Hierarchical Symbolic Simulator for Analog Circuit Design

ABSTRACT

Highly automatic electronic design tools for analog circuit design are scare due to its great complexity and experience-based design flow. Repeated simulation iterations are usually involved for analog circuit designers to reach the design specification. On the other hand, the advent of SOC design results in an ever enlarging gap between designers' productivity and a shorter time to market requirement. To address this problem, a hierarchical symbolic analysis method is proposed in this paper, aiming to increase the design efficiency with the inherited merits of symbolic analysis.

The proposed method combined a topological symbolic approach with its algebraic counterpart on two hierarchies. Circuit regularity was fully utilized. Symbolic stamps of sub-circuits with different structure were constructed and assembled to the top circuit matrix, then solved by an algebraic symbolic solver. The simulator's capability of handling large scale analog circuit was improved greatly due to the hierarchical scheme and a compact data structure.

The thesis is organized as follows. A brief introduction is listed in Chapter 1. Algorithm and theory of hierarchical symbolic analysis are described in Chapter 2. Chapter 3 talks about the implementation details of the simulator. Chapter 4 presents the experimental results. Chapter 5 summarizes the paper.

KEY WORDS: Hierarchical Symbolic Simulation, Analog Circuit Design, Symbolic Stamp, Binary Decision Diagram

目录

第一章 绪论 ……1
1.1 数值化分析与符号化分析 ……1
1.2 符号化分析的产生背景 ……2
1.3 符号化分析的历史与现状 ……3
1.4 主要理论背景介绍 ……4
1.4.1 二分决策图 Binary Decision Diagram ……4
1.4.2 基于 Determinant Decision Diagram 的符号化分析 ……7
1.4.3 基于 Graph-Pair Decision Diagram 的符号化分析 ……9
1.4.4 DDD 与 GPDD 的区别 ……10
1.5 课题提出和研究意义 ……10
1.5.1 课题的提出与已有的研究成果 ……10
1.5.2 研究意义 ……12
第二章 多层次符号化分析算法 ……13
2.1 模拟电路的层次化特性 ……13
2.2 多端口网络分析及其符号化表示 ……14
2.2.1 多端口网络分析 ……14
2.2.2 基于 GPDD 的端口导纳求解 ……15
2.2.3 符号化导纳矩阵（Symbolic Stamp） ……21
2.3 多层次符号化分析架构与电路矩阵的符号化求解 ……22
2.4 本章小结 ……24
第三章 多层次符号化仿真器的实现 ……25
3.1 仿真器架构 ……25
3.2 网表分析器（Parser） ……26
3.3 子电路类（Subckt）与 X 器件类（XDev） ……27
3.4 符号化子电路导纳矩阵的实现 ……28
3.4.1 多根 GPDD 类 NPortGPDD ……28
3.4.2 初始约化图对的构造 ……31
3.4.3 根节点 1-边与 0-边的符号 ……35

3.5 电路矩阵的构造与符号化求解 ……37
3.6 本章小结 ……39
第四章 测试结果 ……40
4.1 仿真器工作流程 ……40
4.2 基于单个晶体管的分层符号化分析效率测试 ……41
4.2.1 Bipolar 电路的测试 ……41
4.2.2 MOSFET 电路的测试 ……44
4.3 电路划分策略测试 ……46
4.3.1 Bipolar 电路的划分测试 ……46
4.3.2 MOSFET 电路的划分测试 ……49
4.4 本章小结 ……55
第五章 结束语 ……57
5.1 主要工作与创新点 ……57
5.2 研究展望 ……58
参考文献 ……60
致谢 ……63
攻读硕士学位期间已发表或录用的论文 ……65

图录

图 1-1 数值化分析与符号化分析示意 ……1
图 1-2 BDD 表示的布尔函数 ……4
图 1-3 约化后的 BDD ……5
图 1-4 BDD 相同子图的共享 ……6
图 1-5 布尔函数 f(x1, …, x8) = x1x2 + x3x4 + x5x6 + x7x8 的 BDD 表示 ……6
图 1-6 行列式的 DDD 表示 ……7
图 1-7 DDD 的构造过程及共享 ……8
图 1-8 电路的 Modified Nodal Analysis（MNA）矩阵 ……9
图 1-9 电路的有向图表示 ……9
图 1-10 电路传输函数的 GPDD 表示 ……9
图 1-11 不同符号化仿真分析的电路处理能力（以所含晶体管数为指标） ……11
图 2-1 全摆幅 Cascode 放大器 ……14
图 2-2 二端口网络实例 ……15
图 2-3 导纳矩阵电路求解示例 ……16
图 2-4 初始有向图构建规则举例 ……18
图 2-5 导纳 y11 的符号化表示 ……18
图 2-6 GPDD 求值规则 ……20
图 2-7 导纳 y12 的符号化表示 ……20
图 2-8 多根 GPDD（multi-root graph-pair decision diagram）示例 ……22
图 2-9 多层次符号化分析架构 ……23
图 2-10 多层符号化分析举例 ……24
图 3-1 仿真器架构 ……25
图 3-2 子电路类（Subckt）基本结构 ……27
图 3-3 X 器件类（XDevice）基本结构 ……28
图 3-4 多根 GPDD 类（NPortGPDD）基本结构 ……29
图 3-5 边类 Edge 的数据结构 ……29
图 3-6 类 Subgraph 的数据结构 ……30
图 3-7 类 Subgraph 的主要成员函数与调用关系 ……30
图 3-8 类 Subgraph 的主要成员函数与调用关系 ……31

图 3-9 类 Subgraph 的主要成员函数与调用关系 ……32
图 3-10 类 Subgraph 的主要成员函数与调用关系 ……33
图 3-11 类 Subgraph 的主要成员函数与调用关系 ……33
图 3-12 二端口网络的 6 种左右子图 ……34
图 3-13 符号转换的三种情况 ……35
图 3-14 根节点 1-边正负号转换的三种情况 ……35
图 3-15 电路矩阵的构造与基于行列式的符号化分析 ……37
图 3-16 顶层 DDD 的符号列表 ……38
图 3-17 DDD 的构造与数据结构 ……38
图 4-1 仿真器工作流程 ……40
图 4-2 运算放大器 ua741 电路图 ……42
图 4-3 运算放大器 ua725 电路图 ……43
图 4-4 双极型晶体管的小信号模型 ……43
图 4-5 全摆幅 Cascode 放大器 ……44
图 4-6 运算放大器（44 管） ……45
图 4-7 MOS 管小信号模型 ……45
图 4-8 一种启发式的 ua725 电路划分 ……47
图 4-9 电路 3 划分一 ……49
图 4-10 电路 3 划分二 ……51
图 4-11 电路 3 划分三 ……53
图 4-12 电路 4 划分一 ……54

表录

表 2-1 不同类型元件边的约化操作 ……19
表 3-1 合法器件 ……26
表 3-2 合法命令 ……26
表 3-3 合法器件 ……36
表 4-1 ua741 测试结果（共享单个晶体管） ……41
表 4-2 ua725 测试结果（共享单个晶体管） ……44
表 4-3 电路三和电路四的测试结果（共享单个晶体管） ……46
表 4-4 电路 ua725 测试结果（启发式的划分） ……48
表 4-5 电路 3 测试结果（划分一） ……50
表 4-6 电路 3 测试结果（划分二） ……52
表 4-7 电路 3 测试结果（划分三） ……53
表 4-8 电路 4 测试结果（划分一） ……54

第一章 绪论

本章首先介绍了电路分析的两大基本方法：数值化分析与符号化分析，并着重阐述了符号化分析的优缺点及其产生的背景。第二部分叙述了符号化分析的现状，介绍了已有的一些成果与算法。最后给出了本课题的目的与意义。

1.1 数值化分析与符号化分析

电路分析目前主要可分为数值化分析和符号化分析两种。如图 1-1 所示，数值化分析将电路表示成矩阵的形式，电路的求解随之转化为一系列矩阵的数值操作，并求得系统的数值解。而符号化分析将电路方程以符号化电路参数的形式存储在特定的数据结构中，每次求解时只要电路拓扑结构没有改变，该数据结构就无需重新构造，只需替换参数的数值，就可以求解电路。

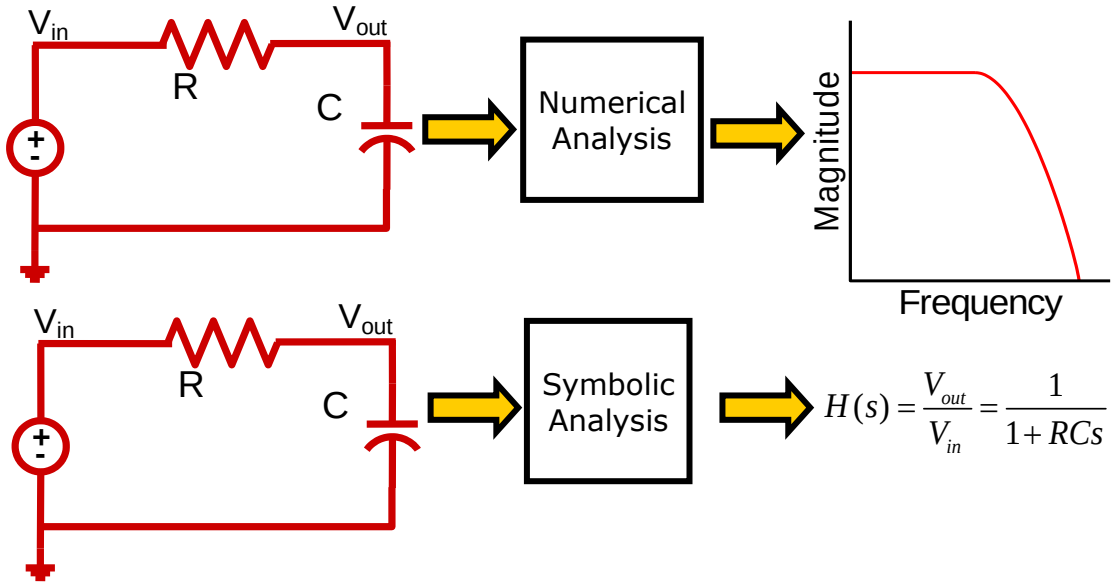


图 1-1 数值化分析与符号化分析示意

Fig.1-1 Comparison between numerical analysis and symbolic analysis

随着上世纪集成电路规模的不断扩大及矩阵理论与数值计算的飞速发展，加州大学伯克利分校(University of California, Berkeley)率先开发了数值电路仿真器 SPICE。此后基于 SPICE 的各种仿真器（如 Synopsys 的 HSPICE，Cadence 的

SPECTRE）成为了工业界的首选。SPICE 可以分析包括各种非线性的二极管及场效应管的电路，并可对电路的直流，交流小信号以及时间域传输特性进行分析。由于电路矩阵的稀疏性，以及矩阵求解等数值算法的飞速发展，数值化分析的速度有较大的提升，同时往往能求解较大的电路，这是数值化仿真器在工业界流行的主要原因。

而符号化仿真器的研究一致停留在学术界，主要原因是它可以处理电路的规模有限。随着电路的增大，符号化分析中的参数（symbol）数也急剧上升，而且其对应的数据结构往往随 Symbol 数指数上升，所以符号化仿真器难以处理较大规模的电路。

但相比数值化分析，由于符号化分析是一种以电路的自变量（时间与频率）、应变量（电压和电流）以及符号化电路中的元件参数来计算电路特性和行为的方法，可以得到符号形式的公式，清晰的揭示了电路特性。同时在该符号化公式的基础上，可以得到对某一电路参数的导数，从而优化或指导电路的设计。

1.2 符号化分析的产生背景

现代片上系统（System On Chip）一般即包含数字电路模块，也包含模拟电路模块。数字电路方面，计算机辅助设计软件（CAD）及自动化的设计流程已相当成熟，工程师可以用高级语言（如 VerilogHDL 硬件描述语言）根据产品参数在高层做功能的描述，然后计算机辅助设计软件可以将硬件语言描述的电路综合成门级网表，再经过自动的布局布线，最终可以得到电路的版图文件。高自动化计算机辅助设计软件的引入大大缩短了数字电路产品的开发周期。而模拟电路往往更为复杂精确，主要还是依靠工程师手工的精雕细琢，设计者需要有丰富的经验。因此数字模块与模拟模块在设计周期上的差距日益增大，无法满足片上系统及混合信号设计的要求。

而符号化分析在辅助模拟电路设计方面有其特有的优势。首先与数值化仿真器不同，符号化仿真器可以提供用户电路传输函数的符号化表示，从而进一步得到哪些参数对整个电路影响较大，某个性能指标与哪些参数有关，为设计者提供更多的参考；其次由于解析公式的获得，符号化分析可以在此基础上做一些电路的自动化设计或优化，如一定程度上的自动晶体管尺寸调整。再次对于那些需要反复求值的，但保持拓扑结构及模型不变的电路，符号化分析只需带入参数便可求值，而数值仿真器就必须重新进行矩阵的求解。对于这种应用，符号化分析在仿真时间上会有一定优势。另外符号化仿真器在最佳拓扑结构选择、设计空间拓

展、行为模型产生以及故障检测等方面比数值型的仿真器具有更大的优势[1]。因此，自上世纪六七十年代，人们就开始关注符号化分析的研究。

1.3 符号化分析的历史与现状

早在上世纪六七十年代，符号化分析就受到了广泛的关注。当时主要为了分析模拟滤波器，涌现出了一系列符号化仿真技术与软件。主要的分析方法有[1]：

a) 基于矩阵行列式的方法
b) 基于信号流图的方法
c) 生成树枚举的方法
d) 参数提取的方法
e) 插值法

其中基于信号流图与生成树枚举的方法在精确符号化分析中取得了较好的效果，但当时这些方法能处理的实际电路十分有限，规模仅为 15 个节点，30 条支路左右。这也反应了符号化分析最大的一个缺点：符号化分析的仿真时间以及内存的消耗随电路规模的上升而指数上升。

为了克服这一限制，人们在上世纪 80 年代提出了两种解决方案。第一种是近似得到的符号化解析公式（如文献[2-5]）。因为电路中的各种参数在数值上往往差别巨大，如 MOS 管的跨导与漏源电导相比。所以在最后的解析公式中很多项的系数是很小的，于是人们就想到去掉那些系数很小的项，从而减小符号化公式所占用的内存大小。当然近似分析会引入一定的误差，需要做误差控制。

另一种方法是层次化的分析[6-9]。电路会被划分成不同的模块层次，对每个小模块做符号化分析，再合并到一起得到整个电路的特性。因为整个电路特性函数由小模块嵌套组成，不再展开成积之和（Sum of Product）的形式，大大减小了符号化分析所需的存储空间。

后来也出现了一些结合近似化与层次化的符号化分析方法。但由于符号化分析本身数据结构的局限性及层次化模块共享机制的不完善，当时的层次化分析只能处理仅含十几个晶体管的电路。当然以上这两种方法在后面的研究过程中仍然起了重要的引导作用。

2000 年，X.D. Tan 等[10]提出了基于行列式决策图（Determinant decision diagram）的符号化分析，采用了二分决策图（Binary Decision Diagram[11]）这种数据结构，最大化解析公式中相同项的共享，在一定程度上解决了内存消耗增长过快的限制，实现了基于该技术的一个仿真器，可以分析具有 20 多个 BJT 的模拟电

路。

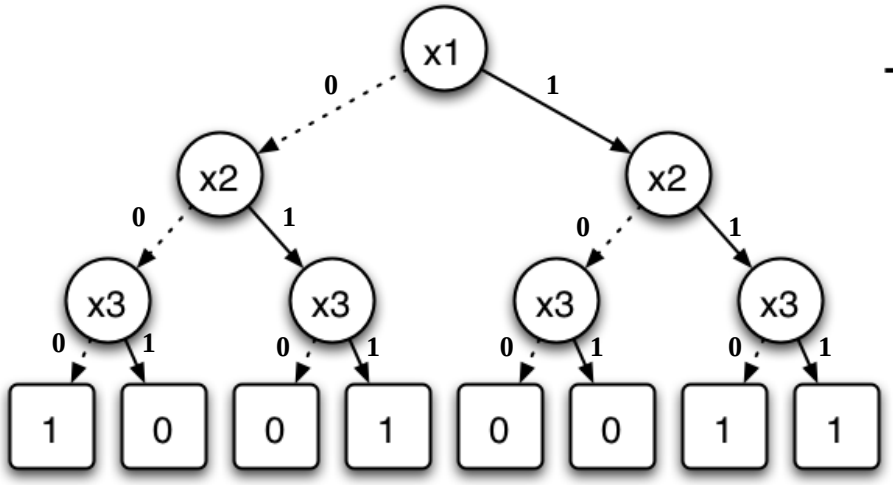
同样采用 Binary Decision Diagram (BDD)这种数据结构，G. Shi 等[12]提出了基于图约化（Graph reduction）的一种符号化分析方法，并实现了一个符号化仿真器 GRASS（Graph Reduction Analog Symbolic Simulator）[13]，也可以处理 20 几个晶体管规模的电路。

本文所述的符号化多层方法将结合以上两种方法的优点，进一步提升仿真器的性能。

1.4 主要理论背景介绍

1.4.1 Binary Decision Diagram

二分决策图（Binary Decision Diagram）是一种无环有向图。它有两个终结节点 0 和 1。每个非终结节点有 0 和 1 两条边，即两种不同决策。BDD 技术最早是为了解决布尔代数的求值与验证问题。在 BDD 提出之前，这类问题的求解往往是 NP 完全的。在 1986 年，Randal E. Bryant 首先提出了约化的二分决策图（Reduced order binary decision diagram, ROBDD）[11]的概念，使得人们能处理当时许多大型组合逻辑的数字电路。



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 1-2 BDD 表示的布尔函数（$f(x_1,x_2,x_3)=\overline{x}_1\overline{x}_2\overline{x}_3+\overline{x}_1x_2x_3+x_1x_2x_3+x_1x_2\overline{x}_3$）[14]

Fig.1-2 An example of a Boolean function in the form of BDD[14]

如图 1-2 所示，一个布尔函数可以方便的表示成 BDD 的形式。我们称从根节点出发到 1 这个终结节点的路径为 1-路径，如 f(0, 0, 0) = 1 就对应图中最左侧的一

条 1-路径，显然每条 1-路径就构成了一个乘积项（如$\overline{x}_1\overline{x}_2\overline{x}_3$），所有 1-路径的组合就构成了布尔函数和之积的形式。

显然图 1-2 中的 f(x1, x2, x3)可以化简成图 1-3 中的 f(x1, x2, x3)，对应的 R. Bryant 提出了 BDD 的共享与化简方法。如图 1-3 所示，ROBDD 只含有一个 0 和 1 终结节点，同时所有相同的子 BDD 会被共享到一起。所以通过对相同子图的共享，ROBDD 在一定程度上解决了复杂度随规模指数增长问题。

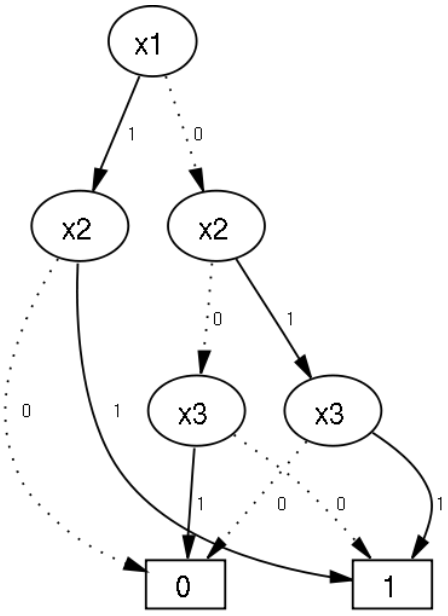


图 1-3 约化后的 BDD ($f(x_1,x_2,x_3)=\overline{x}_1\overline{x}_2\overline{x}_3+x_1x_2+x_2x_3$) [14]
Fig.1-3 An example of reduced ordered binary decision diagram[14]

自然地，相同子图的共享成为了 BDD 构造效率的一个重要因素，这里一般采用哈希技术（Hash）。可以证明在 ROBDD 中任何一个节点和其两个儿子节点构成的三元组是唯一的，因此我们将三元组的节点信息作为 Hash 函数的输入，得到 Hash 键值（Key），从而在 O(1)的时间复杂度内查出当前三元组是否可以被共享，这样在 BDD 自下而上的构造过程中就可以高效地实现共享（如图 1-4 所示，三元组 T1 与三元组 T2 将被共享）。

BDD 中另一个重要环节是符号的排序（ordering），排在前面的符号会先被处理（即更靠近根节点）。对于相同的符号排序，Bryant 指出 BDD 有其唯一的最简形式，即 ROBDD；但对于不同的符号排序，构造出的 BDD 大小差别巨大。如图 1-4 与 1-5 所示，两个 BDD 表示的都是 f(x1, …, x8) = x1x2 + x3x4 + x5x6 + x7x8，图 1-4(a)采用了一个较差的序列，图 1-4(b)采用了一个较优的排序，可以看到图 1-4(a)中 BDD 的大小明显要大很多。

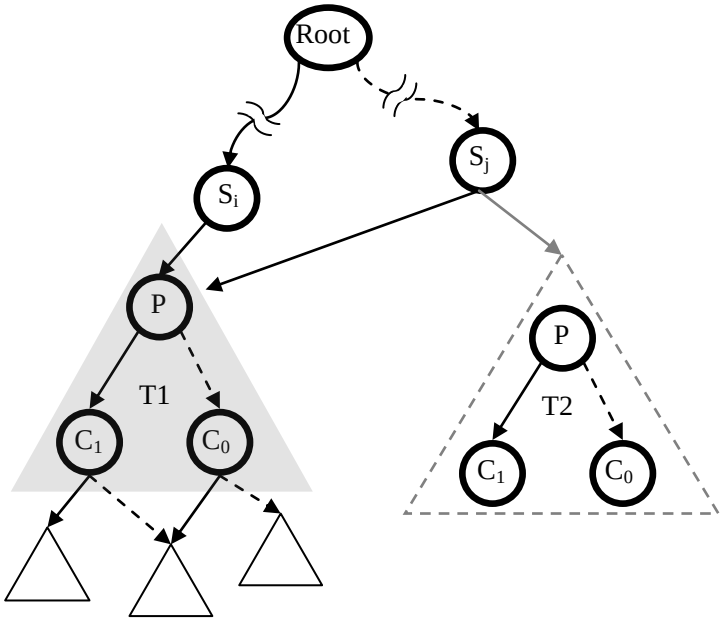
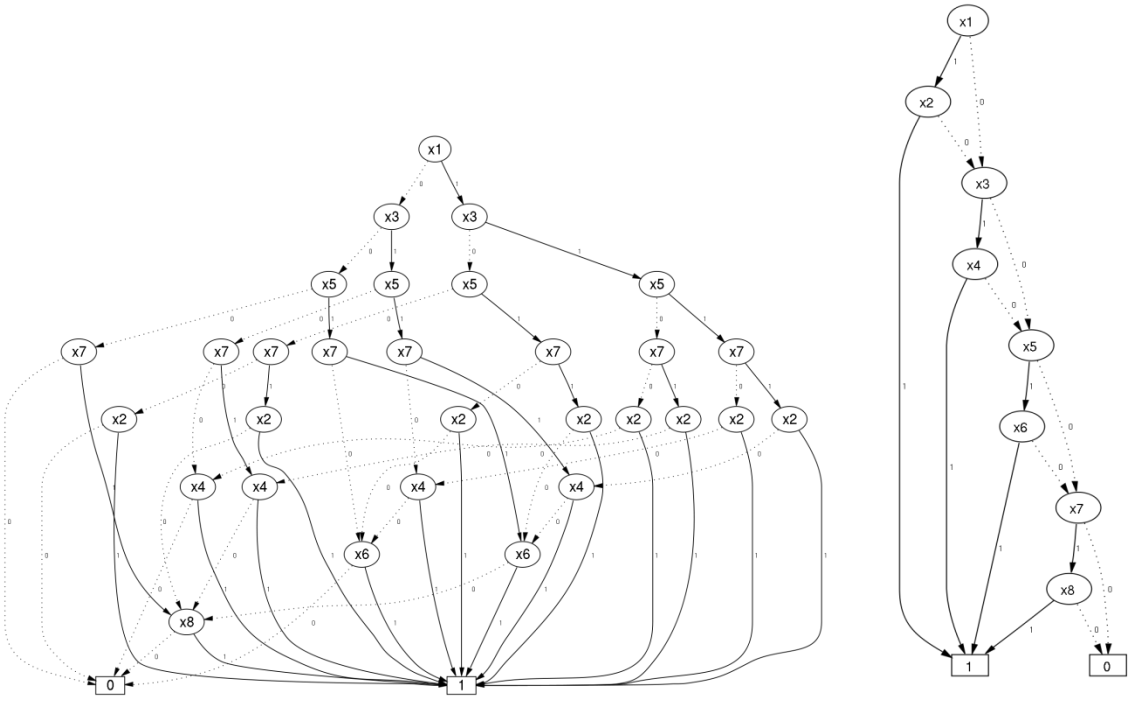


图 1-4 BDD 相同子图的共享
Fig.1-4 Common sub-BDD sharing



(a) 较差的符号排序
(a) A worse order

(b) 较优的符号排序
(b) A better order

图 1-5 布尔函数 f(x1, …, x8) = x1x2 + x3x4 + x5x6 + x7x8 的 BDD 表示[14]
Fig. 1-5 BDD for the Boolean function f(x1, …, x8) = x1x2 + x3x4 + x5x6 + x7x8[14]

然而对于一般问题的最优排序确定是 NP 完全的，实际应用中，往往采用启

发式（Heuristic）的方法，找到该特殊问题的一个较优的排序，从而将 BDD 的大小控制在合理的范围内。

1.4.2 基于 Determinant Decision Diagram 的符号化分析

Determinant Decision Diagram（DDD）[10]首次将 BDD 技术与矩阵的求解联系起来。在该数据结构中，BDD 所表达的不再是布尔函数，而是一个行列式的符号化公式。如图 1-6 所示，行列式 det(A)可以表示为右边的 DDD，只要遍历该 DDD 的所有 1-路径，就可以得到行列式 det(A)和之积（sum of product）的展开公式。

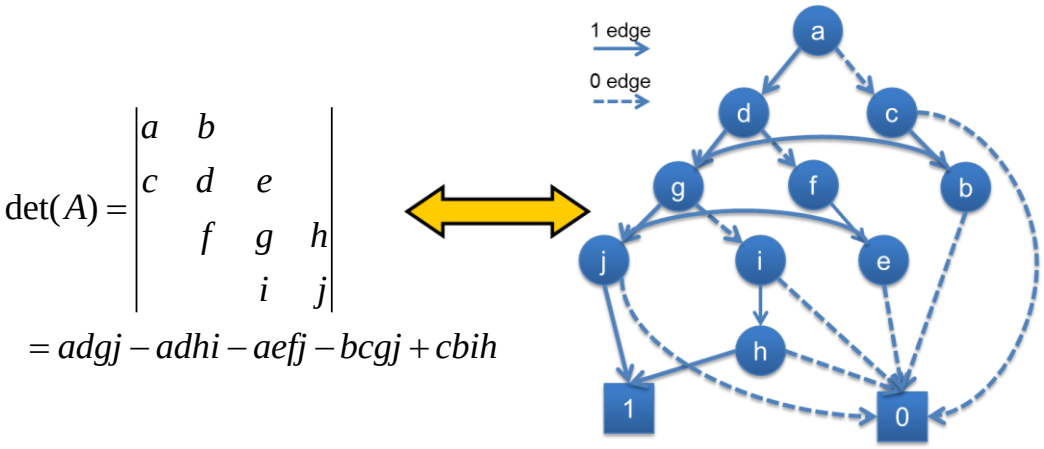


图 1-6 行列式的 DDD 表示[10]

Fig. 1-6 An example of representing determinant using DDD[10]

论文[10]中指出，DDD 的构造实际上是与行列式的拉普拉斯展开对应的。如图 1-7 所示，根节点 a 对应初始行列式 det(A)，DDD 中的符号即行列式中非零元素，按照排序，相继做拉普拉斯展开。如开始从 a 出发，拉普拉斯展开时会出现两个余子式，一个是 a 置 0 后得到的子式（称为 remainder），另一个划去 a 所在行列得到的余子式（称为 minor）。所以原本布尔符号的 0/1 两种取值的选择在 DDD 中变成了拉普拉斯展开时，符号置零与划去的两种选择。这样不断向下进行，直到行列式为 0（对应 DDD 中的 0 节点）或划去最后一个元素（对应 DDD 中的 1 节点）。

因为 DDD 本质上还是一种 BDD，所以在构造过程中共享与符号的排序仍是 DDD 效率的主要决定因素。论文[14]改进了论文[10]中的实现，首先在哈希上直接采用当前子 BDD 对应子行列式（即图 1-7 中每个节点对应的子行列式）的信息来计算哈希函数，而非基于以往 BDD 的三元组这一标准方法，提高了检测子图共

享的效率。其次论文还给出了满矩阵情况下的最优排序，虽然一般情况下并非最优排序，但使得 DDD 的构造变得十分容易，且具有较好的排序效果。

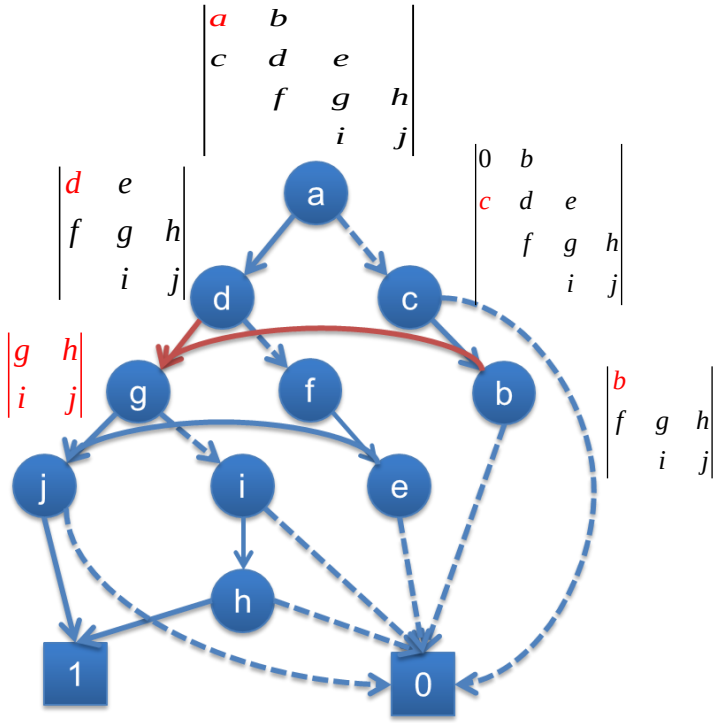


图 1-7 DDD 的构造过程及共享[10]

Fig. 1-7 The construction and sharing of DDD[10]

有了行列式的符号化表示，我们就可以进行矩阵的符号化求解。根据基尔霍夫电流与电压定律我们可以将电路表示成矩阵的形式，如图 1-8 中左侧电路可以表示成右侧的 Modified Nodal Analysis（MNA）[15]形式的矩阵方程。一般地，电路可以表示成如下矩阵方程：

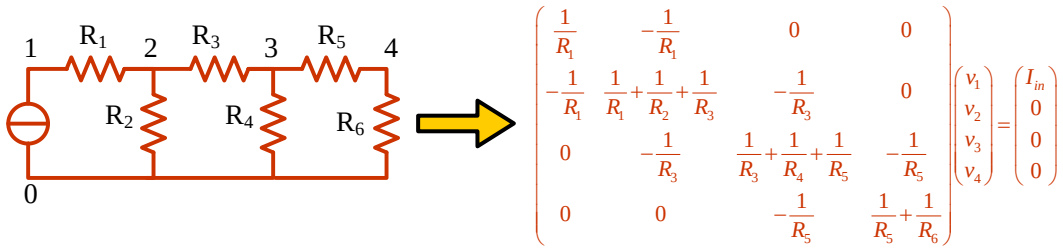
$$AX = B$$

$$X = (x_1, x_2, ..., x_n)^T, B = (b_1, b_2, ..., b_n)^T \qquad \text{（Eq. 1-1）}$$

$$x_k = \frac{\sum_{i=1}^{n} b_i (-1)^{i+k} \det(A_{i,k})}{\det(A)}$$

其中 x_k 是电路的未知变量，A 是电路的系统矩阵，B 是电路的输入矩阵。

而式（1-1）中的 x_k 是矩阵行列式之比，因此只要指定电路的输入输出我们就可以通过 DDD 得到电路传输函数的符号化表示。



$$\begin{pmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3} & -\frac{1}{R_3} & 0 \\ 0 & -\frac{1}{R_3} & \frac{1}{R_3}+\frac{1}{R_4}+\frac{1}{R_5} & -\frac{1}{R_5} \\ 0 & 0 & -\frac{1}{R_5} & \frac{1}{R_5}+\frac{1}{R_6} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} I_{in} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

图 1-8 电路的 Modified Nodal Analysis（MNA）矩阵
Fig. 1-8 An example of MNA matrix

1.4.3 基于 Graph-Pair Decision Diagram 的符号化分析

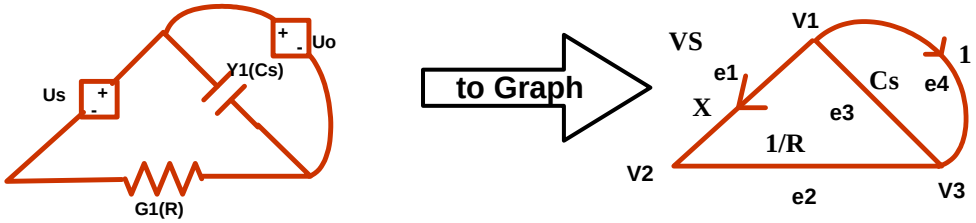


图 1-9 电路的有向图表示
Fig. 1-9 Circuit and its corresponding directed graph

与之前 DDD 不同，论文[12]从图的角度提出了另一种电路符号化的分析方法。如上图所示，线性电路可以转换为一个有向图的表示（如图 1-9）。论文[12]中给出了一系列证明及图的约化规则，在枚举该图生成树的过程中可以构造对应的 BDD。因为每个节点的建立与一对图有关，故称该 BDD 为双图决策图 Graph-Pair Decision Diagram（GPDD）。

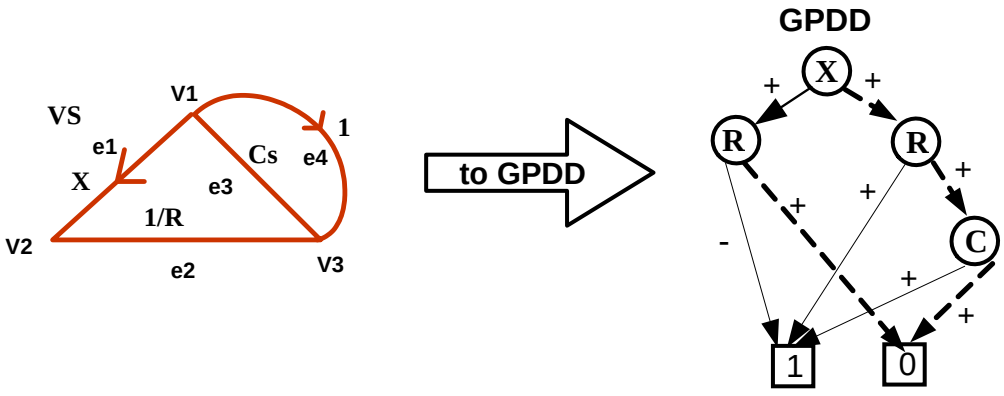


图 1-10 电路传输函数的 GPDD 表示
Fig. 1-10 Circuit and its corresponding directed graph

类似地，遍历该 GPDD 的 1-路径，便可以得到对应电路的传输函数。如上图中 GPDD 所表示的传输函数为：

$$-R^{-1}X+(R^{-1}+sC)=0$$
$$H(s)=1/X=\frac{R^{-1}}{R^{-1}+sC} \quad (1\text{-}2)$$

同样，GPDD 中也要处理共享与符号排序这两大问题。对于共享，因为在 GPDD 中每个节点对应一对图，论文[12]的作者采用了基于图的哈希算法，高效地进行图对的共享。对于排序，一般采用电路元件出现的自然顺序，因为往往相继出现的元件是相连的，所以在图的约化过程中能更快的减小图对的大小。对于较小的模拟电路，该启发式的排序都有较好的结果。

1.4.4 DDD 与 GPDD 的区别

DDD 从行列式的角度构造系统的传输函数，而 GPDD 从图生成树的角度构造传输函数，对于较好的排序，这两种方法都可以处理含有 20 多个晶体管的电路，可谓殊途同归。

但仍有一些不同点值得指出。DDD 中，矩阵的每个非零项被视为符号，从图 1.7 中可以看到，实际求解电路是，MNA 矩阵中的每一项与好几个电路参数有关，所以在进一步操作时（如求导）会带来一些额外的开销。

而从图 1-10 中我们发现每个符号与电路中的元件是一一对应的，所以在求导等操作时会十分方便，不需要额外的管理。

1.5 课题提出和研究意义

1.5.1 课题的提出与已有的研究成果

模拟电路设计一直以来是比较复杂的，设计者需要花大量的时间学习与实践以获得足够的设计经验，而符号化仿真从公式的角度出发，能为设计者提供更有针对性的信息。

总结前文所述的一系列符号化分析方法，大致可以以下四个标准对符号化分析进行划分[16]：

a) 问题的构造方式：行列式出发（代数）vs.有向图出发（拓扑）

b) 符号化传输函数的特性：精确 vs.近似

c) 层次特性：非层次化 vs.层次化

d) 数据结构种类：非 BDD vs. BDD

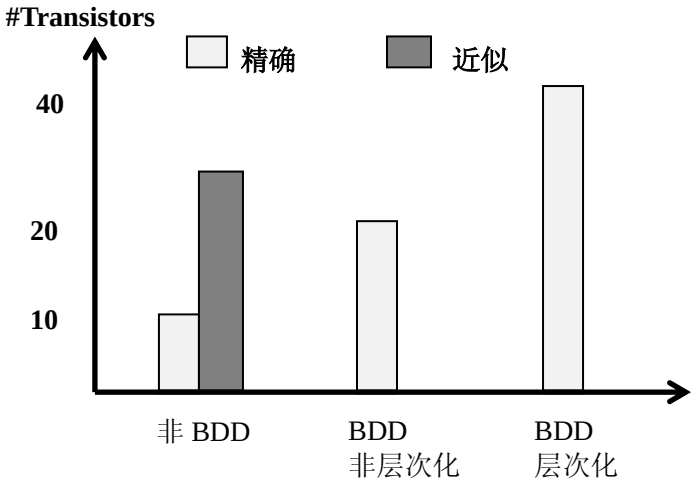


图 1-11 不同符号化仿真分析的电路处理能力（以所含晶体管数为指标）

Fig. 1-11 Comparison between different symbolic analysis approaches

由图 1-11 可以看到除了近似的符号化分析，一般非多层的符号化分析只能处理小于 40 个晶体管的模拟电路。而近似分析本身又存在一定的缺点。首先是精度问题，无论如何，近似化的分析总要牺牲一定精度；其次在该符号化公式的基础上进一步操作时（如求导），我们将遇到第二个更为严重的问题，由于原来的精确解析公式被化简了，虽然本身的误差并不大，但该公式对某一电路参数的导数值（即对该参数的灵敏度 sensitivity）的误差可能会因为公式的化简而被大大放大。

因此本文充分利用了 BDD 这一数据结构的紧凑性与电路的模块性，采用了精确的符号化分析，这样不仅提升了仿真器的分析能力，而且保证了可以进一步进行基于灵敏度的电路优化[17-20]。

本文所述工作参考了实验室之前两位同学的工作。陈薇薇同学[13, 21]在 2006 年实现了基于 GPDD 的仿真器 GRASS（Graph Reduction Analog Symbolic Simulator），由于采用了 BDD 这种充分共享的数据结构的符号化仿真器，相比之前的一些方法在运行时间与消耗内存方面有了很大的改善。但对于常用的一些 Bipolar 及 MOS 管小信号模型，GRASS 能处理的电路规模仍然有限，最大为 20 多个晶体管的电路。对于更复杂的一些放大器，滤波器等一些模拟电路，GRASS 仿真器也无能为力。

为了解决这一问题，陈硕同学[21]在 GRASS 的基础上采用了多层次的结构。首先对电路进行模块划分，再将每个模块传给 GRASS 获得其导纳矩阵，然后将

各个模块的导纳矩阵组装回整个电路的 MNA 矩阵，最后在顶层采用符号化的高斯消去法求解电路。论文[21]对多个电路做了不同的划分，并与非层次化的 GRASS 进行对比，讨论了各种划分的优劣，同时发现层次化的分析无论是在运行时间还是内存消耗方面都有很大的提高。因此这种层次化的分析能处理的电路规模也进一步扩大，论文[21]中最大的一个测试电路含有 44 个 MOS 管。

然而，论文[21]中上层 MNA 矩阵的求解采用的是符号化的高斯消去，这种方法并不能充分利用共享，效率仍有待提高，同时由于多层的嵌套，在该结构上的进一步操作，如求导等，代价很大，难以实现。另外电路的最佳划分与自动化，支持层次化网表的解析器等工作仍有待进一步研究与实现。

基于上面的分析，为了进一步提高符号化分析的效率，我们提出了“用于模拟电路模块设计的多层次符号化分析新算法研究与软件实现”这一课题，在前人的基础上研究新算法，新结构，开发一个功能更强大，更自动化的多层次符号化分析软件。

1.5.2 研究意义

如课题背景中所述，早在上世纪七八十年代就有人开始研究符号化分析，但一直以来符号化分析远未达到实用的要求。与此同时，市场对模拟电路设计周期的要求却越来越高。一个功能强大的模拟电路计算机辅助设计软件的出现迫在眉睫。

本课题旨在进一步提高符号化分析的能力，先解决符号化分析处理电路规模有限这一问题。同时尝试一些电路的优化，软件用户界面的设计，使得设计者能通过和软件的互动，进一步了解电路的特性，参数的重要程度等。为将来基于符号化分析的电路自动优化等研究课题打下基础。

第二章 多层次符号化分析算法

正如第一章所述，符号化分析最大的一个问题即算法的空间复杂度随问题规模上升指数增长，即使采用了二分判定图[11]这种紧凑的数据结构，也无法避免该问题。然而模拟电路本身往往包含诸多相同结构的电路模块，如电流镜，有源负载，差分对，基本运放等等。事实上，除了 BDD 这一数据结构本身带来的共享，在开发符号化仿真器的过程中，人们往往忽视了电路本身的内在层次性。本文就从该角度出发，充分利用电路结构的相似性，结合 BDD 这个紧凑的数据结构，大大提升仿真器的处理能力。

本章首先举例说明了电路的层次性，并回顾了基础多端口网络理论，在此基础上详细阐述了多层次的符号化分析算法。

2.1 模拟电路的层次化特性

模拟电路往往由很多基本模块所组成，而且是多层次的。一般可分为：

a) 系统层
b) 基本模块层（如运算放大器）
c) 器件层（如 MOSFET）

举例而言，比如一个模拟数字转换电路（ADC）一般包含很多相同结构的运算放大器，这是系统层面的结构相似性；同时在一个运算放大器的内部又包含很多基本单元，这就是基本模块层。例在如图 2-1 所示的全摆幅 Cascode 二级运算放大器中，M5，M7，M8 与 M6，M13，M14 便构成了两组结构一致的电流镜/有源负载电路，类似的 M1，M2 与 M3，M4 是结构一致的差分输入；最后一般同一个电路中，每个 MOSFET 对应的小信号模型也是相同的，构成了最底层的器件结构相似性。

然而非层次化的符号化分析没有利用模拟电路的这一特性。符号化分析处理的是线性电路的频率响应，所以对于含有非线性元件（如 MOSFET）的电路，我们先要做直流工作点的分析，然后提取所有电路元件的对应小信号模型，再构成一个新的线性电路。例如上图，若每个 MOSFET 对应小信号模型含有 12 个线性元件，那么生成的线性电路网表就包含超过 200 个线性元件，而且如果不做特殊处理，将丧失原来特有的层次化特性，导致电路规模的增大，限制了符号化分析

的处理能力。

为了保留这种层次化结构，本文对电路进行划分，对每个子块进行符号化分析，最后再符号化地处理整个网络，从而提升了该符号化分析算法的处理能力，可以精确的分析较大规模的模拟电路。

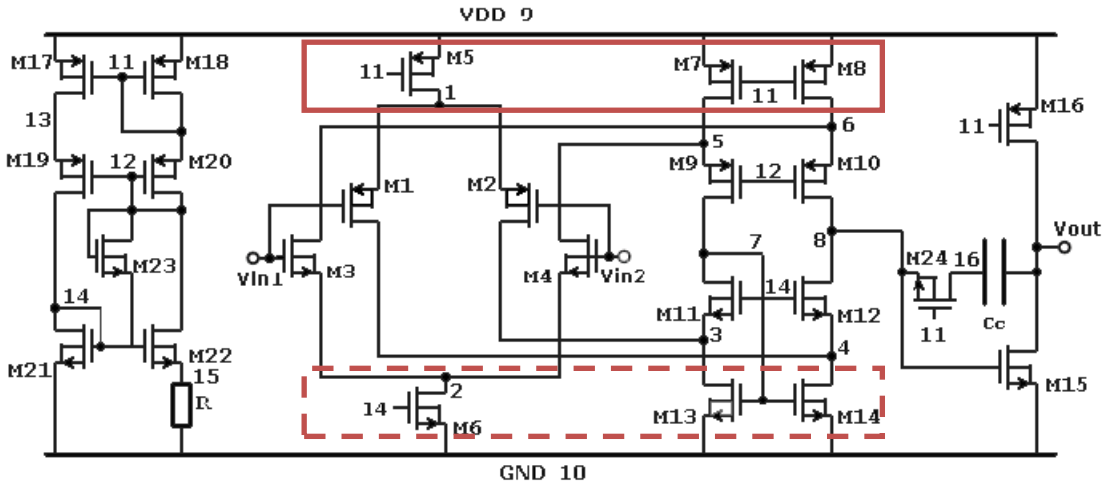


图 2-1 全摆幅 Cascode 放大器

Fig. 2-1 A rail to rail two-stage cascode amplifier

2.2 多端口网络分析及其符号化表示

事实上，任何一个电路或其子电路都可以看作是一个多端口的电路网络，如图 2-1 中方框中的电路，可以看作是一个 5 端口的网络。基于这种多端口网络的概念，我们就可以将一个大的电路划分为多个子网络，从而实现电路的层次化描述。

本小节将简要介绍多端口网络的理论分析，并着重描述如何符号化地表示一个多端口网络。

2.2.1 多端口网络分析

多端口网络分析是电路分析中的一个常用方法，通过描述一个子电路的端口特性，我们可以简化原电路问题：一方面，电路变得具有层次性；另一方面，在分析好每个子电路后，我们无需关心子电路的内部关系，对于顶层电路，它将成为一个黑盒子，只需知道其端口特性。

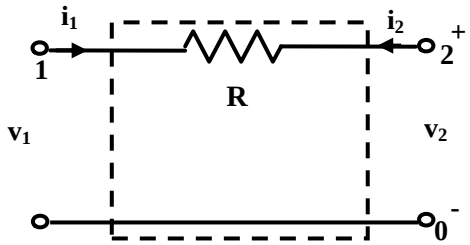


图 2-2 二端口网络实例
Fig. 2-2 A 2-port circuit example

多端口网络的一种表示是以导纳形式给出的，本文就采用了该描述方面，这里做简要介绍。如图 2-2 所示二端口网络，我们可以方便的通过下述矩阵来描述该网络的端口特性：

$$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{21} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 1/R & -1/R \\ -1/R & 1/R \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \quad \text{(Eq. 2-1)}$$

矩阵的每一行约束了该端口电流与所有端口电压的关系，因此对应的矩阵称为导纳矩阵。一般地，对于 N-端口网络，有：

$$\begin{bmatrix} i_1 \\ i_2 \\ \mathrm{M} \\ i_N \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & \Lambda & y_{1N} \\ y_{21} & y_{22} & \Lambda & y_{2N} \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ y_{N1} & y_{N2} & \Lambda & y_{NN} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \mathrm{M} \\ v_N \end{bmatrix} \quad \text{(Eq. 2-2)}$$

其中 $i_1, i_2, \ldots, i_N$ 为流出端口的电流，$v_1, v_2, \ldots, v_N$ 为端口电压，y_{ij}（i, j = 1, 2, …, N）为对应端口间的导纳。

为了求 y_{ij}，由式 2-2 可以看出，我们可以将 v_k (k = 1, 2, …, N 且 $k \neq j$)置为 0，将 v_j 置为 1，则此时 i_i 即为 y_{ij}。对应到电路上，即将除 j 端口以外所有端口短路，测流出 i 端口的电流。

另外对于一个包含 N + M 个节点的电路，其中 N 为我们所选取的 N 个端口，M 为剩余的内部节点，如果能方便的算出对应 N^2 个导纳，那么我们相当于将一个 (N+M) x (N+M)的电路导纳矩阵，降阶为一个 N x N 的电路导纳矩阵，即减少了顶层电路的节点个数。

2.2.2 基于 GPDD 的端口导纳求解

如上一小节所述，如果能高效地表示子电路的导纳矩阵就可以减小原问题的规模；另一方面，对于结构一致（具体元器件参数的数值可以不同）的子电路，

如图 2-1 中实线框与虚线框所示两个电路，其对应导纳矩阵必然也具有一样的结构，不同的只是矩阵元素的数值。因此，如果我们能构建符号化的导纳矩阵，就能进一步提高该结构的复用率。

论文[13, 20-21]中描述了一种基于图约化的符号化电路分析方法，该方法首先将电路转换为一个有向图，并指定一对输入输出，然后根据一系列图约化规则，构建所谓的双图决策图（Graph-Pair Decision Diagram），基于该判定图便可以符号化地求出对应输入输出间的传输函数。

本文拓展了这一结构，使其成为多根的双图决策图（Multi-root Graph-Pair Decision Diagram），用以表示导纳矩阵。对于图 2-2 中所示电路，为了求其导纳矩阵，根据上小节所述，我们可以构造如下四个测试电路（图 2-3）。如为了求 y_{11}，可以将端口 2 短路，并在端口 1 加一个测试电源 v_t，则流过该电源的电流 i_t 等于 y_{11}；为了求 y12，可以将端口 1 短路，并在端口 2 加一个测试源 v_t，则流过该电源的电流 i_t 等于 y_{12}。类似地，我们可以求得 y_{21} 和 y_{22}。

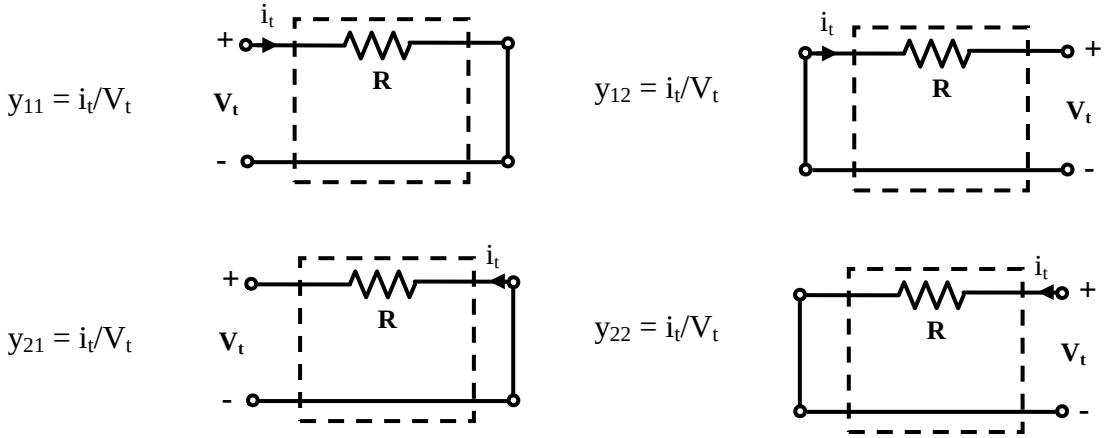


图 2-3 导纳矩阵电路求解示例

Fig. 2-3 Example of admittance matrix computation

因此，一个直观的方法就是利用第一章背景介绍中所述的基于拓扑的符号化分析方法，为每个导纳 y_{ij} 构造一个 Graph-Pair Decision Diagram，然后组装成一个符号化的导纳矩阵。为了阐明该算法过程，这里先回顾一下文献[12, 20, 21]中有关 GPDD 构造的部分规则与算法，详细证明及推导请参阅文献[20, 21]，这里不再赘述。

首先电路必须满足以下一些条件：

规则 2-1 符号化分析法适用电路特性

a) 只含有 5 类元件：阻抗（Z）、导纳（Y）、四种类型的受控源（压控电压源 VCVS、压控电流源 VCCS、流控电压源 CCVS、流控电流源 CCCS），独立电压源、独立电流源以及理想运算放大器。如果电路中含有非线性元件（如二极管、场效应管等）则转换成相应

的小信号模型;
b) 电路中只含有一个独立电压源或者独立电流源。当电路中含有多个独立源时，只需利用线性电路的叠加性，叠加各个独立源单独作用电路时的符号化分析结果即可;
c) 每个控制源只控制一个受控源;
d) 每个受控源只受一个控制源控制。

虽然对电路元件有一定限制，但该符号化分析还是能处理所有的线性电路，而对于非线性电路可以通过线性化后再进行处理。

满足上述条件后，我们就要进行原始电路到有向图的转换，该转换过程依循以下规则：

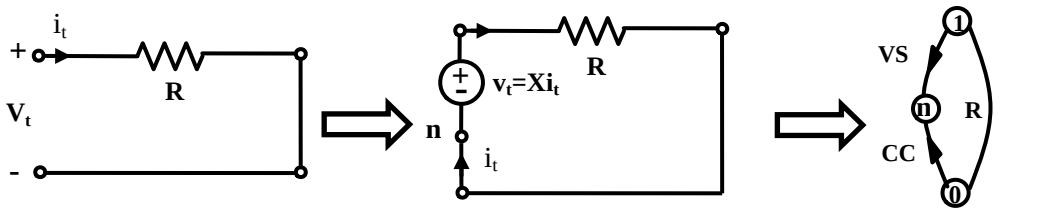
规则 2-2 有向图构建规则

a) 电路中的受控源或者独立源转换成有向边。电压源对应的有向边为正极指向负极，电流源对应的有向边沿电流方向;
b) 在有向图中，对应的电压控制支路增加一条并联的有向边（以满足流过理想电压控制源的电流为零）;
c) 在有向图中，对应的电流控制支路增加一条串联边（以满足理想电流控制源两端压降为零）;
d) 有向图中的每一条边对应着电路中的一个元件，权重为对应元件的名称;
e) 理想运算放大器使用零器（Nullor）建模，每个零器（Nullor）由一对零阻器（Nullator）和一个泛阻器（Norator）组成;
f) 将输入端抽象成受输出端控制的受控源，控制系数为 X。

其中规则（f）是算法中的一个重要环节。对于线性电路，其传输函数 H(s) 取决于输入输出端口的定义。而为了描述这种输入输出的定义且同时满足符号化分析对元件的限制，自然会想到利用受控源来描述输入输出关系，这里称该受控源为输入输出受控源。

但值得注意的是在加上该受控源时，不应改变原始电路的电气特性（节点电压，支路电流）。而理想受控源的电压控制端的电流为零，电流控制端的电压为零。所以为了不改变原始电路的特性，若原始电路为电流输出，则以该电流作为输入输出受控源的电流控制端；如果原始电路的输出是电压则以该电压作为输入输出受控源的电压控制端。这样原始电路的传输函数 H(s) = 1/X，X 为输入输出受控源的控制系数。

特别地，考虑端口 1 输入导纳的求解。因为输出信号是电流，输入信号是电压。所以我们引入流控电压源（CCVS）。又因为实际上求 y_{11} 时，输入输出是一个端口，为此在端口 1 处引入 n 这个内部节点。这样我们就得到了如图 2-4 所示的有向图。



（a）待分析电路　（b）转换后电路　（c）对应有向图

（a）Original circuit　（b）Transformed　（c）Directed graph

图 2-4 初始有向图构建规则举例

Fig. 2-4 Example of constructing initial directed graph from a 2-port circuit

得到初始有向图后，我们要进一步得到初始的有效生成图对（Valid Graph-Pair），如文献[21]所述，该初始图对需满足以下条件：

规则 2-3 有效生成图对定义

a) 只要初始图（starting graph）中含有 NU 和 NO 边，则左树中必须含有所有的 NO 边，同时右图含有所有的 NU 边。

b) 所有的 Y 和 Z 边或者同时出现在有效图对的左、右图中，或者均不出现。

c) 每一对 CC 和 VS 边可以同时出现在有效图对的左、右图中，或者同时不出现，或者成对出现在有效图对中，即 VS 边出现在左图中，CC 边出现右图中。

d) VC 和 CS 可以不出现在有效图对中，或者成对出现在有效图对中，并且 CS 边出现在左图中，VC 边出现在右图中。

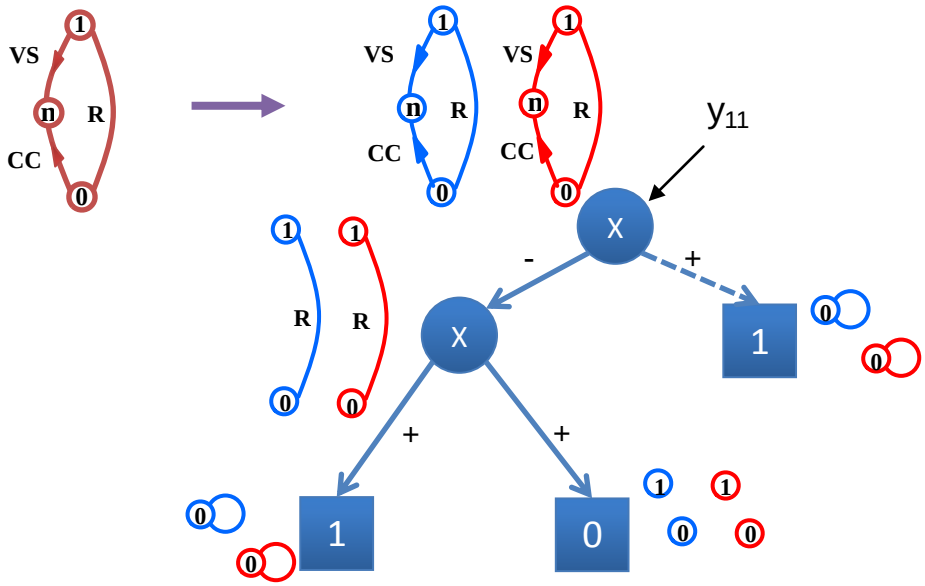


图 2-5 导纳 y_{11} 的符号化表示

Fig. 2-5 Symbolic admittance y_{11}

特别地，根据规则 2-3(c) CC 和 VS 边可以同时在初始左图与有图中出现，我们就可以得到图 2-4 中有向图的有效初始生成图对（如图 2-5 所示）。进一步根据

表 2-1 中所述规则可以得到图 2-5 中各个节点对应生成图对。当图对约化为单一节点时对应 GPDD 中的 1 节点，当图对不是有效生成图对时（如不连通）对应 GPDD 中的 0 节点。同时根据规则 2-4 可以确定对应边的符号。

表 2-1 不同类型元件边的约化操作
Tab. 2.1 Rules for graph reduction

	选取元件（Short，1-边）		排除元件（Open，0-边）	
	左子图	右子图	左子图	右子图
VCVS	短路 VS	移去 VS 短路 VC	短路 VS	短路 VS 移去 VC
CCVS	短路 VS 移去 CC	短路 CC 移去 VS	短路 VS 短路 CC	短路 VS 短路 CC
VCCS	短路 CS	短路 VC	移去 CS	移去 VC
CCCS	短路 CS	短路 CC	短路 CC 移去 CS	短路 CC
Nullor	短路 NO	短路 NU	移去 NO	移去 NU
Y/Z	短路 Y/Z	短路 Y/Z	移去 Y/Z	移去 Y/Z

规则 2-4 生成项符号判定规则

步骤 1 初始化：sign = 1;
步骤 2 如果操作为删除（open）边，则直接删除该边，而 sign 保持不变。如果操作为短接（short）边（v1,v2）且 v1 < v2，则删除该边后，将图中的 v2 节点改为 v1 节点，如果此时图中剩余节点小于 v2 的节点数目为奇数，则 sign *= -1。
步骤 3 如果短接的边反向（即 v1 > v2），则将该边表示成（v2, v1），同时 sign *= -1。
步骤 4 如果短接的边类型为 VS，且 VS 与别的类型边成对出现，则 sign *= -1。
步骤 5 将 sign 关联到相应的 GRDD 边上。

具体地，以图 2-5 为例，假设符号的优先级为 X > R（即先处理 X，再处理 R），那么从初始图对出发，以深度优先做图的约化（边的符号按规则 2-4 确定）：

步骤 1 图对 1→图对 2，采用了表 2-1 中 CCVS 行，第 1、2 列中所述规则（边短路时，统一保留节点数小的端点）;
步骤 2 图对 2→图对 3，采用了表 2-1 中 Y/Z 行，第 1、2 列中所述规则;
步骤 3 图对 2→图对 4，采用了表 2-1 中 Y/Z 行，第 3、4 列中所述规则;
步骤 4 图对 1→图对 5，采用了表 2-1 中 CCVS 行，第 3、4 列中所述规则。

文献[10]证明了所有的生成项之和等于零，根据该结论，我们可以获得最终的计算结果。即

$$X(\sum_{i=1}^{m} T_i^0) + \sum_{j=1}^{n} T_j^1 = 0 \qquad \text{(Eq. 2-3)}$$

为了方便求解，我们将 X 放在根结点，则根结点 X 实边（也称为 1-边，1-edge）

T_i^0

T_j^1

所指的子 GPDD 代表所有包含 X 元件的生成项之和，根结点 X 虚边（也称为 0-边，0-edge）所指的子 GPDD 代表所有不包含 X 元件的生成项之和，根据公式（2-3）就可以得到传输函数（1/X）的符号化表达式。

具体求值时，以深度优先遍历该 GPDD，如图 2-6 所示，以当前节点 P 为根结点的子 GPDD 的值 V(P)，等于其左儿子（1-边所指）的值 $V(C_1)$乘以 1-边上的正负号 Sign(1)，再乘以当前节点的对应符号（Symbol）P，加上其右儿子值 $V(C_0)$与 0-边上的正负号 Sign(0)。

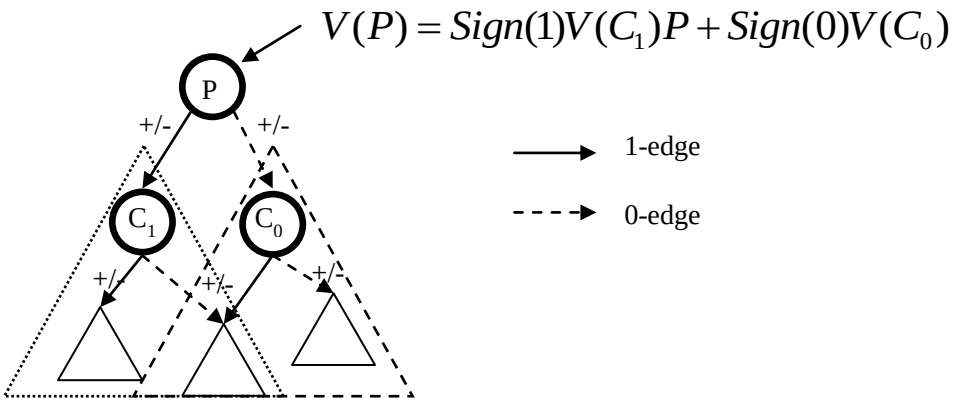


图 2-6 GPDD 求值规则
Fig. 2-6 Rule for GPDD evaluation

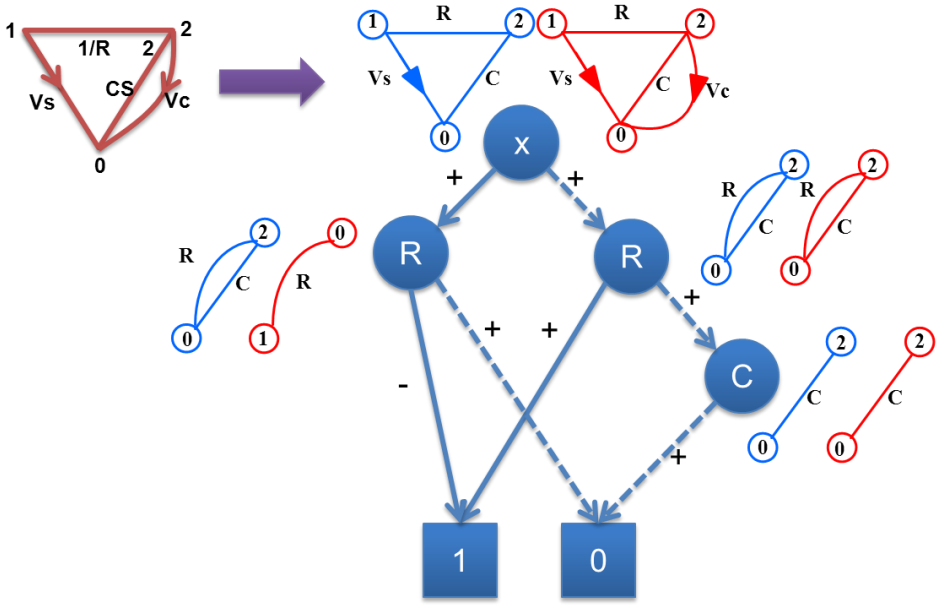


图 2-7 导纳 y_{12} 的符号化表示
Fig. 2-7 Symbolic admittance y_{12}

因此遍历图 2-5 中的 GPDD 就可以得到以下传输函数（端口 1 的导纳）：

$$\begin{aligned} -(1/R)X_{11}+1=0 \\ y_{11}=1/X_{11}=1/R \end{aligned} \qquad \text{(Eq. 2.4)}$$

这样就得到了最终要的导纳。

同理我们可以构造如图 2-6 所示 GPDD 表示 y_{12}，因为 y_{12} 实际上是端口 1 和端口 2 间的跨导，所以引入 CCVS，且 VS 和 CC 两条边分别位于端口 1 和端口 2 间。注意到这里 GPDD 中的所有节点共享 1-节点和 0-节点，事实上如果 GPDD 中存在相同子 GPDD，那么这些子 GPDD 也将被共享。

导纳 y21 和 y22 的构造方式类似，故不再赘述。

2.2.3 符号化导纳矩阵（Symbolic Stamp）

经过上一小节的讨论，实际上我们已经得到了一种获得符号化导纳矩阵（Symbolic Stamp[16]）的方法，即独立地为导纳矩阵中的每一个元素构造对应的 GPDD。一般地，对于 N 端口网络，我们可以通过构造 N^2 个 GPDD 来表示该网络 NxN 导纳矩阵中 N^2 个导纳。然而，这样的表示方法显然不够紧凑，如果子电路的规模较大，那么每个导纳对应 GPDD 本身就很大，再乘以 N^2 个，将消耗大量内存空间，直接限制了算法处理大型电路的能力。

事实上，上述独立的构造方法没有充分利用导纳求解电路的相似性和 GPDD 可以共享这一特点。因为导纳矩阵中的各个导纳项所对应的初始图对具有一定的相似性，在约化的过程中很可能约化为一致的图对，而由第一章的介绍可知图对与其对应的子 GPDD 是一一对应的，所以我们可以通过分析图对是否一致共享所有导纳的子 GPDD。如对于图 2-2 中的 2 端口电路，其四个符号化导纳显然是可以被共享在一个 GPDD 上的。直观上，由于其对称性，易得 y_{11} 等于 y_{22}，y_{12} 等于 y_{21}，因此这些导纳矩阵是可以被共享在一起的，另外根据 GPDD 的特性，一些结构一致的子 GPDD 也会自动被共享在一起。对于一般 N 端口子电路符号化导纳矩阵（Symbolic Stamp）的构造算法简述如下：

算法 2-1 基于多根 GPDD（multi-root graph-pair decision diagram）的符号化导纳矩阵构造算法

CREATE_MULTI-ROOT-GPDD( )

a) 定义 N 端口网络;
b) 生成一个全局存储图对的 Hash 表;
c) 对于每一个端口 i 执行过程 CREATE_GPDD(i)
d) 返回 N^2 个符号化导纳指针

CREATE_GPDD(i)

a) 创建求解端口 i 的导纳电路;
b) 按上一小节中所述方法构造单根 GPDD，但在构造过程中，每次生成一个新图对时就在全

局 Hash 表中检测是否已经存在，若存在则发生共享，否则继续构造子 GPDD;

c) 单根 GPDD 构造完毕后，返回该符号化导纳的 root 指针。

我们称所得的这种共享在一起的 GPDD 为多根 GPDD（multi-root graph-pair decision diagram），一般地，一个 N 端口网络有 N^2 个根，而这 N^2 个根所指向的 GPDD 是共享在一起的。这样一来，不仅节约了存储空间，更重要的是，加快了符号化导纳矩阵的求值过程。因为共享的存在，已遍历过的子 GPDD 就不需要再次遍历，所以遍历一遍多根 GPDD 即可以求出对应导纳矩阵中的所有导纳项。

图 2-8 给出了图 2-2 中 2 端口网络的多根 GPDD 导纳矩阵，显然比上一小节中所述方法更为紧凑，高效。

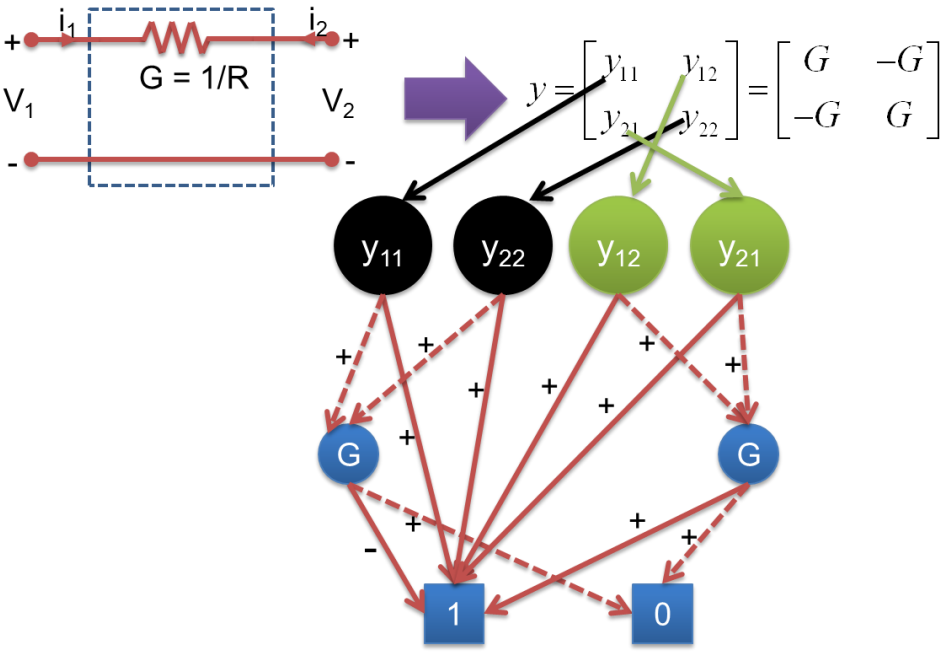


图 2-8 多根 GPDD（multi-root graph-pair decision diagram）示例

Fig. 2-8 An example of multi-root graph-pair decision diagram

2.3 多层次符号化分析架构与电路矩阵的符号化求解

在 SPICE 仿真器[15]中，每个元件都有一个特殊的“邮票”（Stamp），即该元件的导纳矩阵，然后再根据元件的节点位置将该 stamp 填充到顶层的电路矩阵中。本文所设计的多层次符号化分析方法也采用了这种电路矩阵的构造方法，只是现在每个子模块的导纳矩阵是由上一小节所述的多根双图决策图（Multi-root GPDD）所表示的符号化导纳矩阵（Symbolic Stamp）构成。

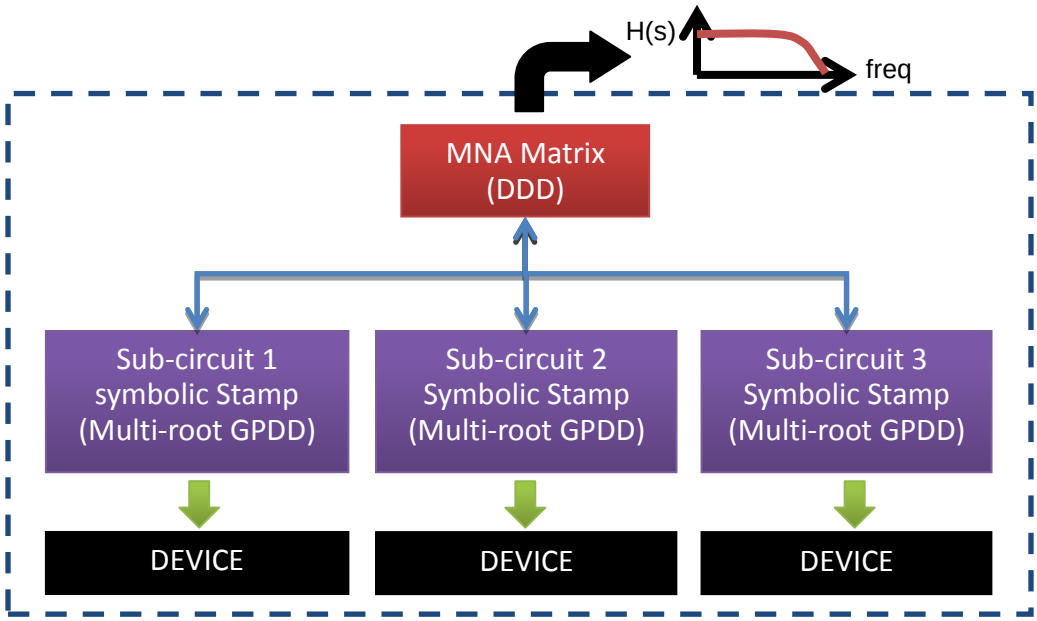


图 2-9 多层次符号化分析架构

Fig. 2-9 Hierarchical scheme of the proposed symbolic analysis algorithm

值得指出的是，与数值化导纳矩阵最大的不同是符号化的导纳矩阵可以复用。对于电路中相同结构的子电路，我们只需要构造一个符号化的导纳矩阵，只是在具体求值时，将带入不同器件的具体取值（如图 2-9 所示）。这样就可以充分利用电路本身的层次化特性。

对于顶层的电路矩阵（Modified Nodal Analysis Matrix[15]）本文采用了背景介绍中的基于行列式的符号化分析（Determinant Decision Diagram）。经过电路的划分，子电路中的内部节点已经不属于顶层电路，只剩下其端口节点，所以顶层电路的规模已经大大减小，从而可以使用基于行列式的符号化分析 DDD 方便地求出整个电路的传输函数（图 2-9）。

举例而言，假设共有两个子电路，Sub1 和 Sub2，它们分别对应符号化导纳矩阵 Mulit-root GPDD1 和 Multi-root GPDD2（如图 2-10 所示）。在顶层电路上有三个实例化模块 A，B，C。其中 A，B 具有相同的电路结构，对应子电路 1，这里就发生了电路结构层面的共享，在求值时他们两个实例化电路将分时复用这个符号化的导纳矩阵；C 是子电路 2 的实例化，它单独对应一个符号化导纳矩阵。无论 A，B，C 内部有多少节点，由于对于顶层电路只需考虑它们的端口，所以由它们组成的顶层电路 MNA 矩阵大小仅为 4x4，这就是层次化分析带来的另一个好处，减小了顶层电路的规模。在得到该顶层 MNA 矩阵后，再用矩阵的符号化分析方法得到一个 Determinant Decision Diagram，求得该电路的传输函数 H(s)。

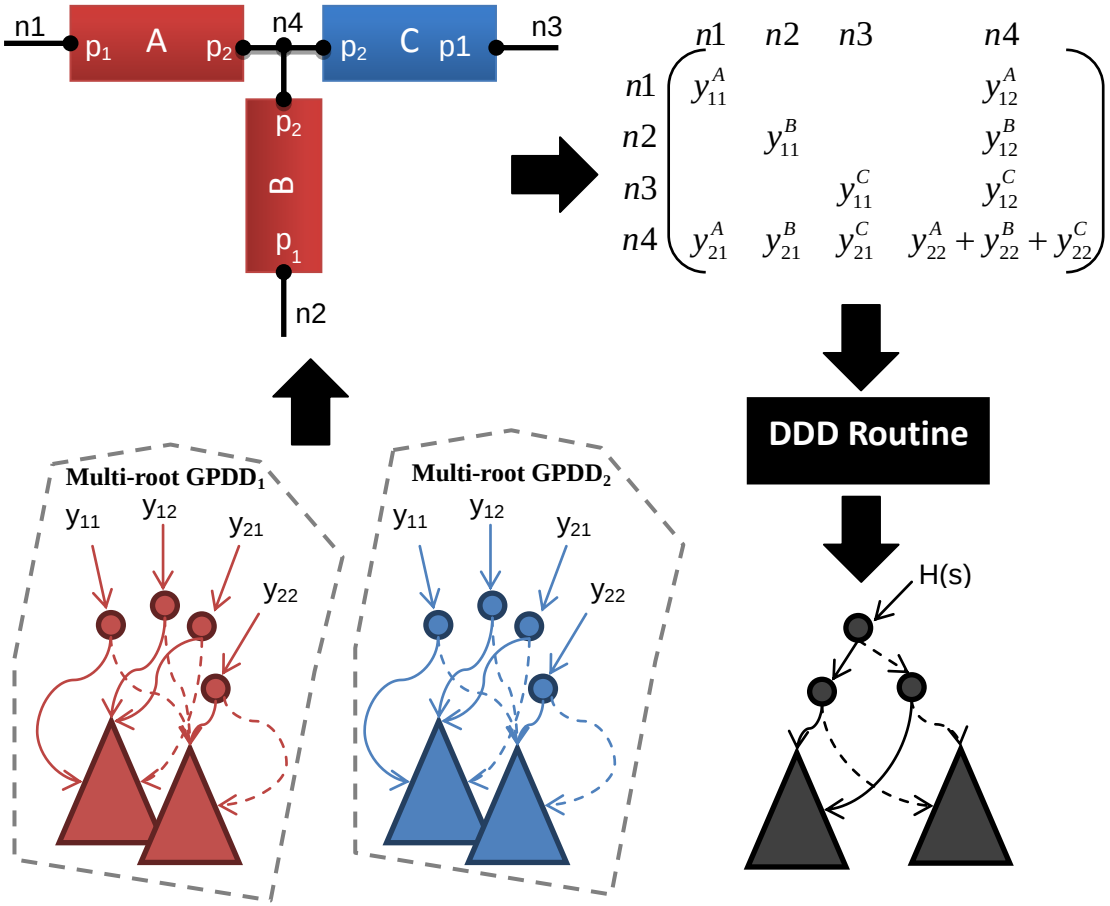


图 2-10 多层符号化分析举例

Fig. 2-10 Hierarchical analysis illustration

有关基于行列式展开的符号化求解代数矩阵的方法，即用行列式决策图（determinant decision diagram）求解矩阵方程，其基本原理已在第一章中做了简要介绍，这里不在赘述。其详细推导，与数据结构请参阅文献[10]。

2.4 本章小结

本章介绍了多层次符号化分析算法的基本思想、目的，与原理。利用电路本身的层次化特性，我们提出了以子电路为单位，构建符号化的导纳矩阵的方法，即所谓的多根双图决策图（multi-root graph-pair decision diagram）。这些符号化导纳矩阵将组装出顶层的电路 MNA（modified nodal analysis）矩阵，最后由基于行列式的符号化分析方法 DDD（determinant decision diagram）求解符号化的传输函数。

第三章 多层次符号化仿真器的实现

3.1 仿真器架构

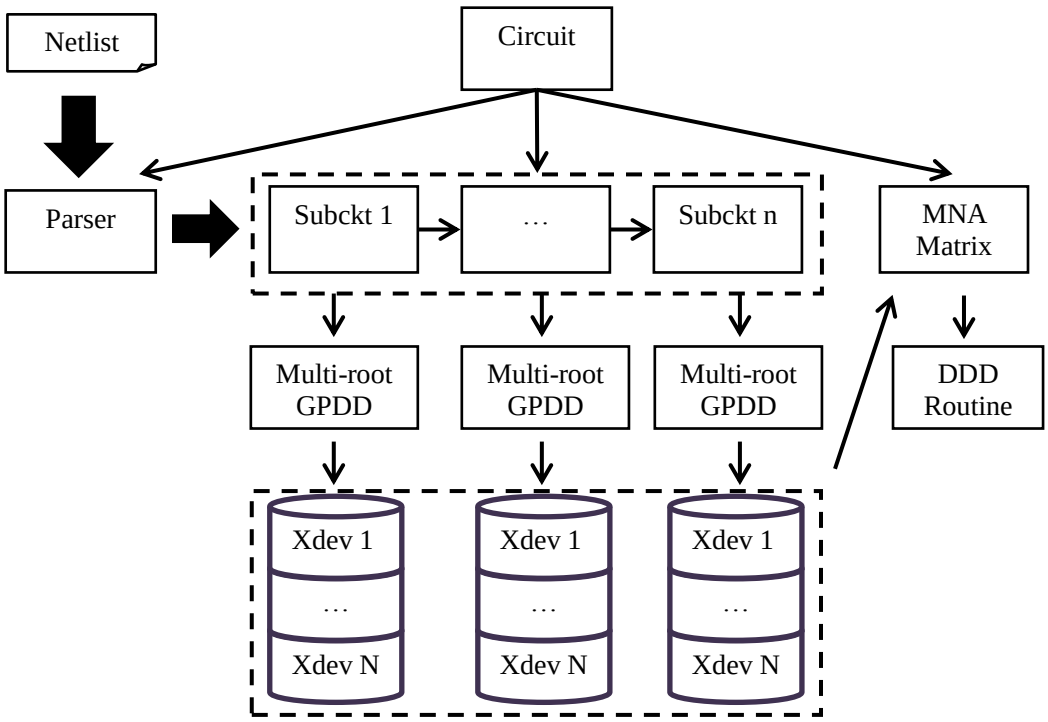


图 3-1 仿真器架构

Fig. 3-1 Simulator Architecture

本文所述的多层次符号化仿真器由 C++开发而成，顶层的类（Class）是 Circuit，主要由三大部分组成（如图 3-1 所示）：第一部分是网表分析器（Parser），完成多层次网表的读入，并构建子电路类与其对应的实例化器件；第二部分是子电路类，它由 Parser 构建出来，除了它对应的实例化器件链表，每个子电路还包含一个多根的 GPDD，用来表示其符号化导纳矩阵；第三部分是顶层电路的 MNA 矩阵，它由所有器件构造而成，同时每个器件的导纳矩阵是通过其对应子电路的符号化导纳矩阵求值而得，该 MNA 矩阵又由 DDD Routine 符号化地求解，最后得到电路的传输函数。

3.2 网表分析器（Parser）

本文使用 Flex[23]和 Bison[24]进行网表分析器的开发。该 Parser 的网表与标准 SPICE[15]网表基本类似。因为符号化仿真只支持线性器件，所以该网表分析器目前表 3-2 中所列器件，其中以字母“x”或“X”开头的是子电路实例化器件，如表 3-1 所列实例，表示该器件名为“Xdev1”，它有三个节点“1，2，3”，它对应的子电路是“sub1”。这样 Parser 就知道该器件是属于子短路“sub1”的，于是在新建该器件时会将其加入子电路“sub1”的实例化器件列表中。<*param=val*>表示参数负值列表，如例子中 r1=1k，即表示将 Xdev1 的参数 r1 负值为 1k，而 r1 应是子电路 sub1 网表中的一个电阻。

表 3-1 合法器件
Tab. 3-1 Valid Devices

器件名	**语法**	**实例**
电阻	[R|r]*xxx* n1 n2 val	R1 1 2 1k
电容	[C|c]*xxx* n1 n2 val	C1 1 2 1p
电感	[L|l]*xxx* n1 n2 val	L1 1 2 1n
VCVS	[E|e]*xxx* n+ n- in+ in- gain	E1 1 2 3 4 10
VCCS	[G|g]*xxx* n+ n- in+ in- transconductance	G1 1 2 3 4 1m
独立电压源	[V|v]*xxx* n1 n2 val	v1 1 2 1
独立电流源	[I|i]*xxx* n1 n2 val	i1 1 2 1
子电路实例化	[X|x]*xxx*<*node list*><*subckt name*><*param=val*>	Xdev1 1 2 3 sub1 r1=1kr2=1M

表 3-2 合法命令
Tab. 3-2Valid Commands

功能	**语法**	**实例**
AC 分析	.ac(.AC) *dec*steps startstop	.ac dec 10 1 1g
指定输出	.print [v|V](n)	.print v(10)
定义子电路	.subckt name <*node list*> <subckt netlist> .ends	.subckt sub1 1 2 3 R1 1 2 1k C1 2 4 1p L1 2 3 1n .ends

表 3-2 给出了 Parser 支持的命令。其中“.ac”规定了 AC 仿真的步长，起始点和终止点；“.print”命令规定了电路的输出，电路的输入是独立电源，所以传输函数就是这里定义的输出与独立电压之比；“.subckt”命令用来定义子电路，与 SPICE 标准语法一致，这样设计者就可以根据电路自身模块性定义子电路。

3.3 子电路类（Subckt）与 X 器件类（XDev）

从图 3-1 中我们就可以看出，仿真器中一个重要的类是子电路类（Subckt），它将了实例化器件，多根 GPDD 和电路矩阵关联在一起，也正是基于该类，实现了电路层面的共享——所有子电路的实例化共享一个子电路的符号化矩阵。因此这里给出子电路类的基本结构。

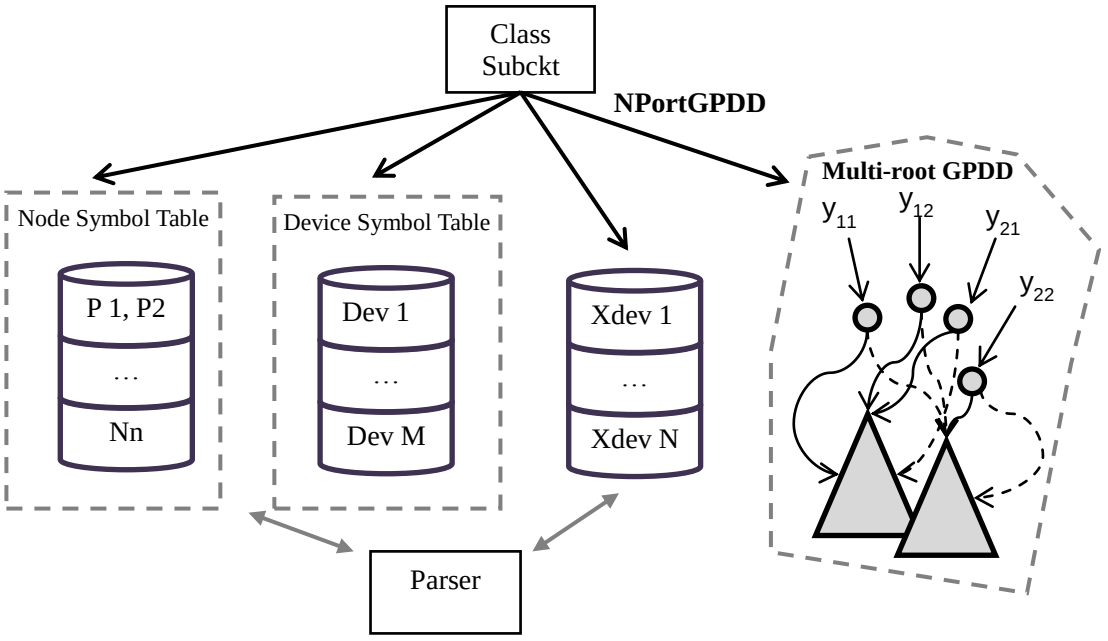


图 3-2 子电路类（*Subckt*）基本结构
Fig. 3-2 Data structure of the class *Subckt*

从图 3-2 可以看出子电路类主要包含四个部分：一是节点名的符号表（Node Symbol Table），二是所含器件名的符号表（Device Symbol Table），三是实例化 X 器件列表，最后是多根 GPDD。

节点名符号表和器件符号表是用于子电路网表的词法语法分析的。我们会将上一小节中所述“.subckt”命令中的<*node list*>先放入 Node symbol table，并标记为端口，其他出现在“.subckt”命令网表段的节点为内部节点。Device symbol table 用于检测该子电路内是否出现同名的器件（重复定义器件是非法的）。网表分析器（Parser）可以向这两个符号表添加元素，同时根据表中节点与器件名的信息构造 X 器件。

Subckt 类还会通过类 NPortGPDD（Device Symbol Table 中的器件作为其参数），生成一个多根的 GPDD，访问所有根指针，就可以得到符号化的导纳矩阵。

而 X 器件将自己的具体参数传给 Subckt 类，就可以计算出它实际的导纳矩阵的数值。

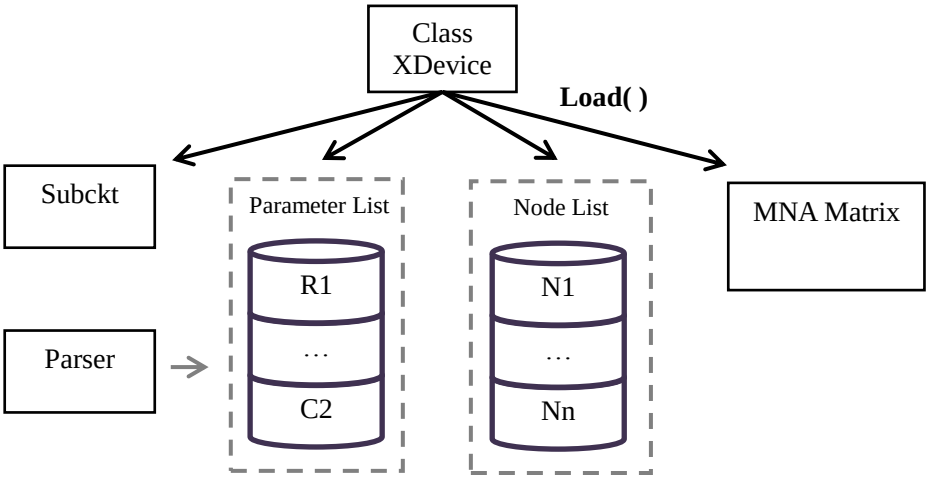


图 3-3 X 器件类（*XDevice*）基本结构
Fig. 3-3 Data structure of the class *XDevice*

图 3-3 给出了 X 器件类（XDevice）的主要数据结构与成员函数。首先 Parser 会将 X 器件类实例化时的电路参数传入 X 器件的参数表（Parameter List）。如果这些参数名与子电路中网表中的器件名一致，那么就记录下实例化的具体数值。节点列表（Node List）记录 X 器件的实际的端口节点。装载函数 Load( )主要复杂两项工作：一是将 X 器件的实际参数传回给 Subckt 类用于导纳矩阵的计算，然后根据 Node list 中的节点信息，将该 X 器件的导纳矩阵（stamp）填入顶层的 MNA 矩阵中。这样在程序的最顶层，调用一遍所有的器件的 Load 函数，就完成了顶层 MNA 矩阵的装载。

3.4 符号化子电路导纳矩阵的实现

本小节将详细叙述符号化子电路导纳矩阵求解的实现。这部分工作借用了陈薇薇同学[21]的部分代码，并基于此进行修改与再开发，进一步完成了多根 GPDD 的实现。

3.4.1 多根 GPDD 类 NPortGPDD

类 NPortGPDD 描述了一个多根 GPDD（multi-root graph-pair decision diagram），它是类 Subckt 的一个成员，Subckt 通过它进行符号化导纳矩阵的计算，

其基本结构如图 3-4 所示。

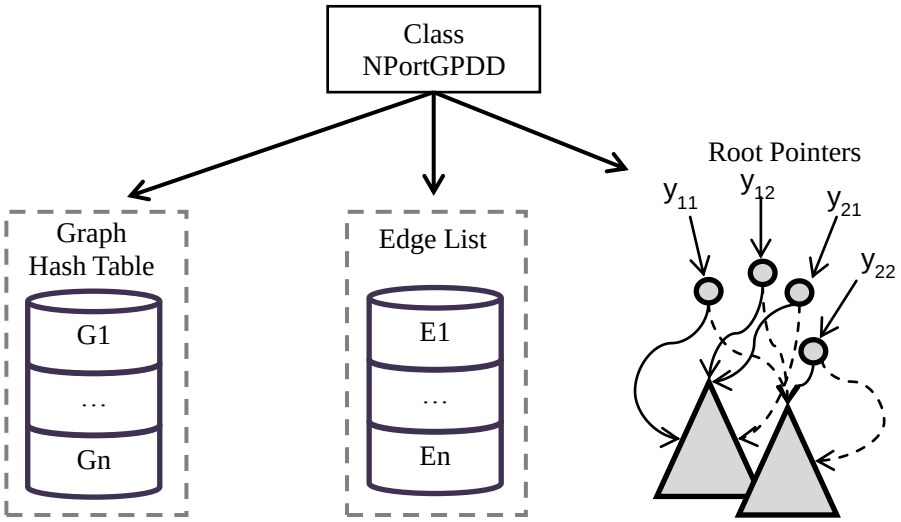


图 3-4 多根 GPDD 类（NPortGPDD）基本结构
Fig. 3-4 Data structure of the class *NPortGPDD*

其中 Edge List 描述了输入的多端口电路，Edge List 里的每个元素为类 Edge 的指针。Edge 类存有其对应线性元件的基本信息，如图 3-5 所示。成员 name 是该边对应线性电路元件（即符号化分析中的符号）的名称，node1 和 node2 是边的两个节点，type 指定了边的类型，当该边是受控源类型时，pname 指定其对边的名字，value 指定了该边的权重（电路元件值）。next 是链表指针，指向下一条边。

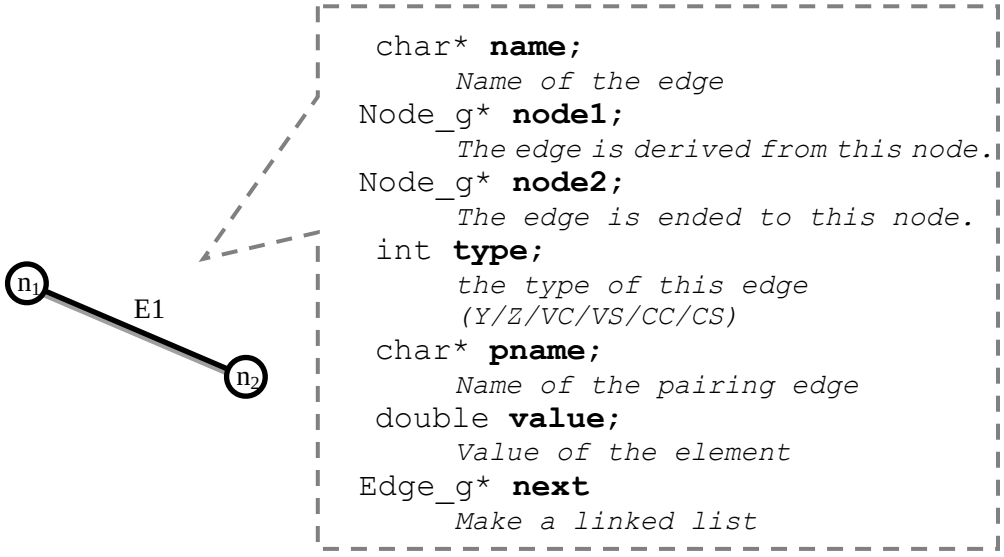


图 3-5 边类 Edge 的数据结构
Fig. 3-5 Data structure of the class *Edge*

图 3-4 中 Graph Hash Table 用来存放多根 GPDD 的所有不同的子图（类

Subgraph）。注意，这里 Hash Table 存放的是子图，而非子图对，而且所有不同的子图都只有这个存放在 Hash 表中的副本，当发生共享时只需用指针指向该副本即可（包括一对图中两张相同子图的共享）。另外，本文采用了文献[10]中的 Hash 方法，先根据当前子图的信息计算出哈希的键值，如果发生冲突再进行基于图的比较，确认两张图是否相同。详细算法请参阅文献[10]这里不再赘述。本文仅给出类 Subgraph 的基本成员与成员函数的调用关系，以供参考。如图 3-6，EdgeNum 是当前子图的总边数，NodeNum 是当前子图的总节点数，EdgeList 是一个 Edge 的数组，它记录当前子图所含的边。注意这三个成员在子图的约化过程中要随时更新，只保存当前剩余的边和节点。图 3-7 给出了 Subgraph 的一个重要成员函数 Compare，在哈希表发生冲突时，将调用这个函数，用来比较两张图的一致性（同构的两张图被认为是一张图）。

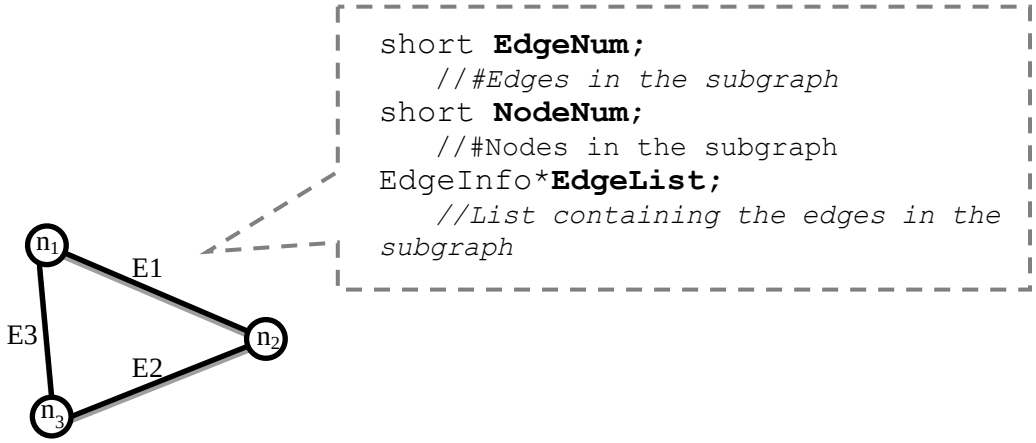


图 3-12 类 Subgraph 的数据结构
Fig. 3-6 Data structure of the class *Subgraph*

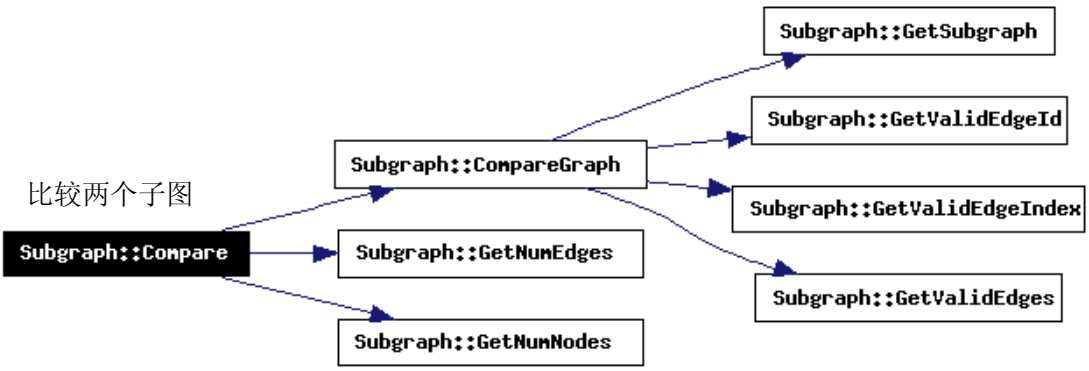


图 3-7 类 Subgraph 的主要成员函数与调用关系
Fig. 3-7 Member functions of the class *Subgraph*

NPortGPDD 最重要的一个部分就是新的多根 GPDD 这一个数据结构。如第三章所述，其构造方法与 GPDD 类似，但必须先构造初始电路，这部分会在下一小节详细叙述。对于包含 NPortGPDD 的 Subckt 类来说，NPortGPDD 只要提供其 N^2（对于 N 端口的子电路）个根节点指针即可（如图 3-4 所示）。

3.4.2 初始约化图对的构造

如第二章理论部分所述，对于多根 GPDD 的构造我们需要先构造 N^2 个端口测试电路（初始图对），然后根据表 2-1 进行下一步的约化。但实际实现过程中，由于我们引入的输入输出受控源是确定的，即 CCVS，利用其约化规则我们可以将初始子图的个数减小为 2N+2。为了方便叙述，下文以 2 端口电路为例说明该构造过程。

如图 3-8，为了计算导纳 Y_{11}，我们构造节点 X_{11} 对应的图对，值得注意的是根据规则 2-1 我们添加的输入输出受控源将同时出现在左图与右图中，而原电路内部元件可能会因为左右子图规则不同，操作（保留或去除）不同，导致左右子图可能不相同。

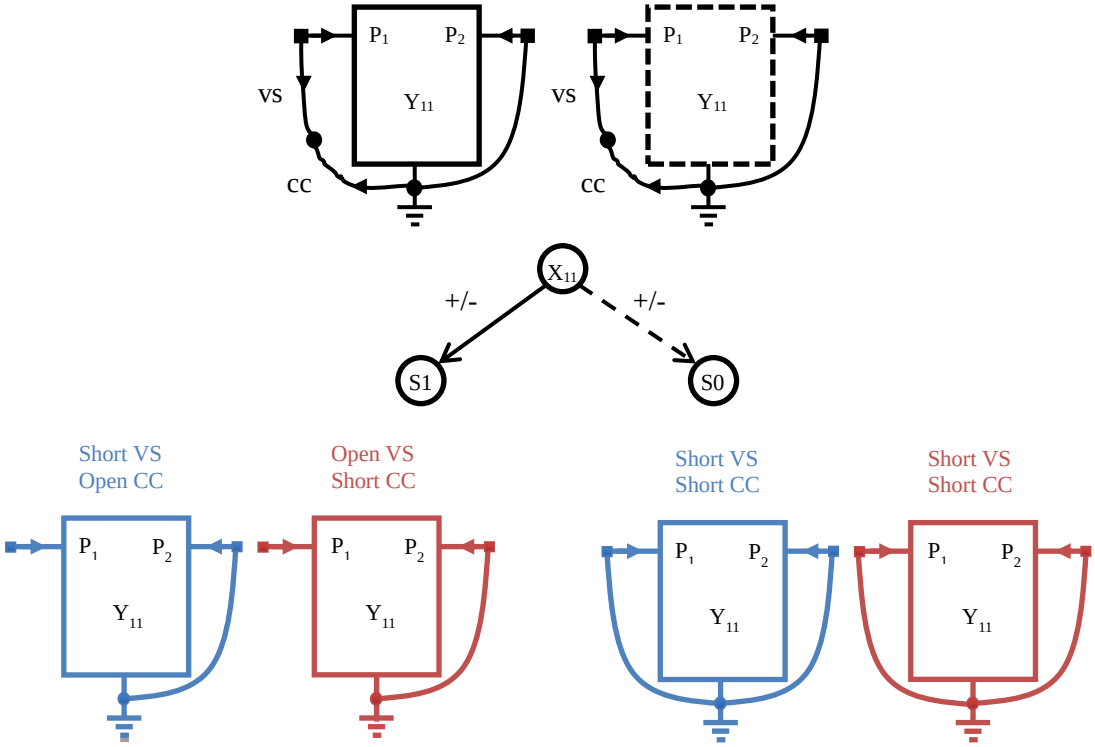


图 3-8 类 Subgraph 的主要成员函数与调用关系
Fig. 3-8 Member functions of the class *Subgraph*

现在我们考虑从根节点（我们总是先处理输入输出受控源）出发向左走，对于左子图其规则是先 Short VS，再 Open CC，所以端口 1（P_1）保持开路，而端口 2（P_2）短路到地。处理右子图时，先 Open VS，再 Short CC，虽然顺序不同但得到的端口连接结果相同，端口 1 开路，而端口 2 短路，同样由于初始的左子图与右子图可能不同，节点 S1 对应的两张子图内部也可能不一致。

如果我们从 X_{11} 向右走到 S0，其左子图的规则是先 Short VS 再 Short CC，所以端口 1 和端口 2 都会被短路到地；处理右子图是 CCVS 的规则是一样的（如图 3-8），所以得到同样的结果，类似地其端口连接致，但左右子图的内部电路可能不同。

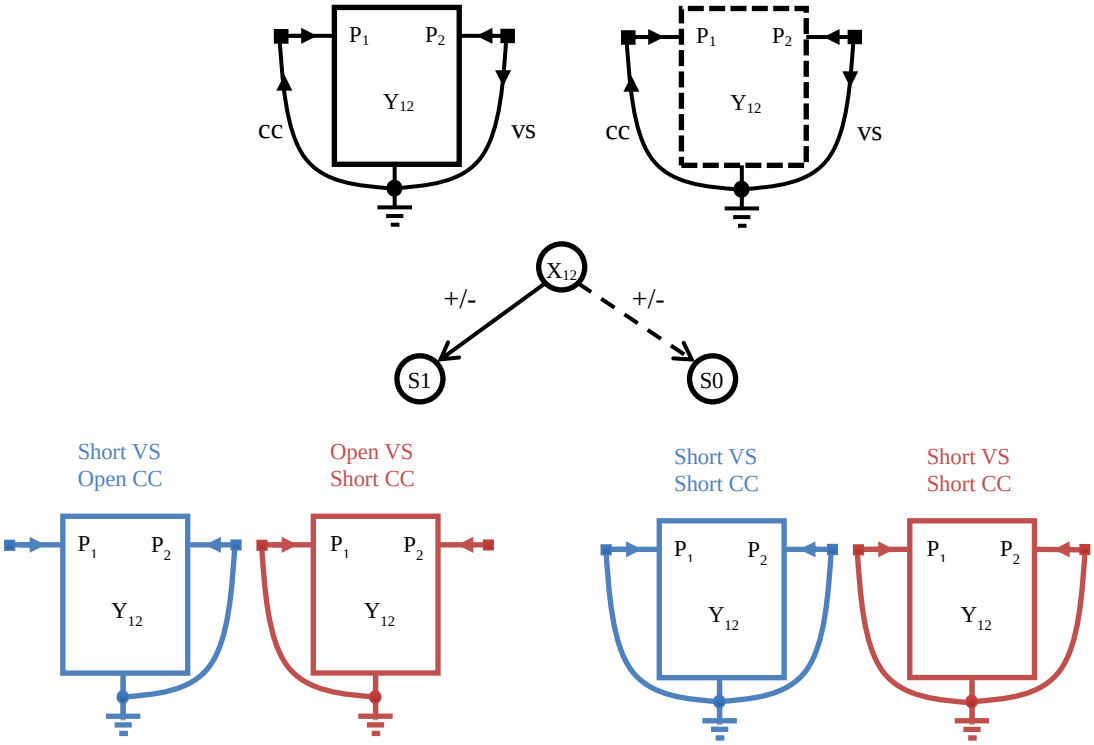


图 3-9 类 Subgraph 的主要成员函数与调用关系
Fig. 3-9 Member functions of the class *Subgraph*

当求解 Y_{12} 时，输入输出受控源的端口将变成如图 3-9 所示，类似图 3-8 的过程，向左走会得到 S1 对应的两张子图，向右走会得到节点 S0 对应的两张子图。同样即使端口连接关系一致，左右子图的内部可能不同。

因为求解 Y_{21} 的电路图是 Y_{12} 电路图的镜像对称，所以我们可以直观的得到图 3-10 中的图对，它们自然也是图 3-9 中对应图对的镜像对称（按所示规则可以验证）。同理，我们可以得到求解 Y_{22} 的过程，其图对是 Y_{11} 的镜像对称（图 3-11）。

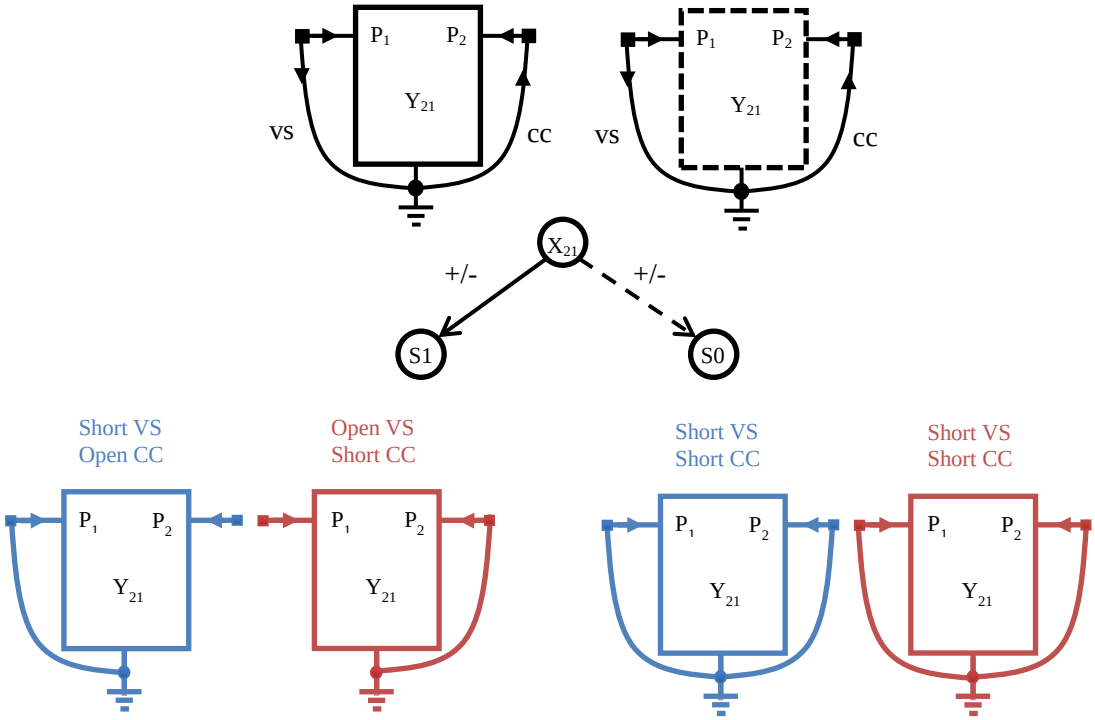


图 3-10 类 Subgraph 的主要成员函数与调用关系
Fig. 3-10 Member functions of the class *Subgraph*

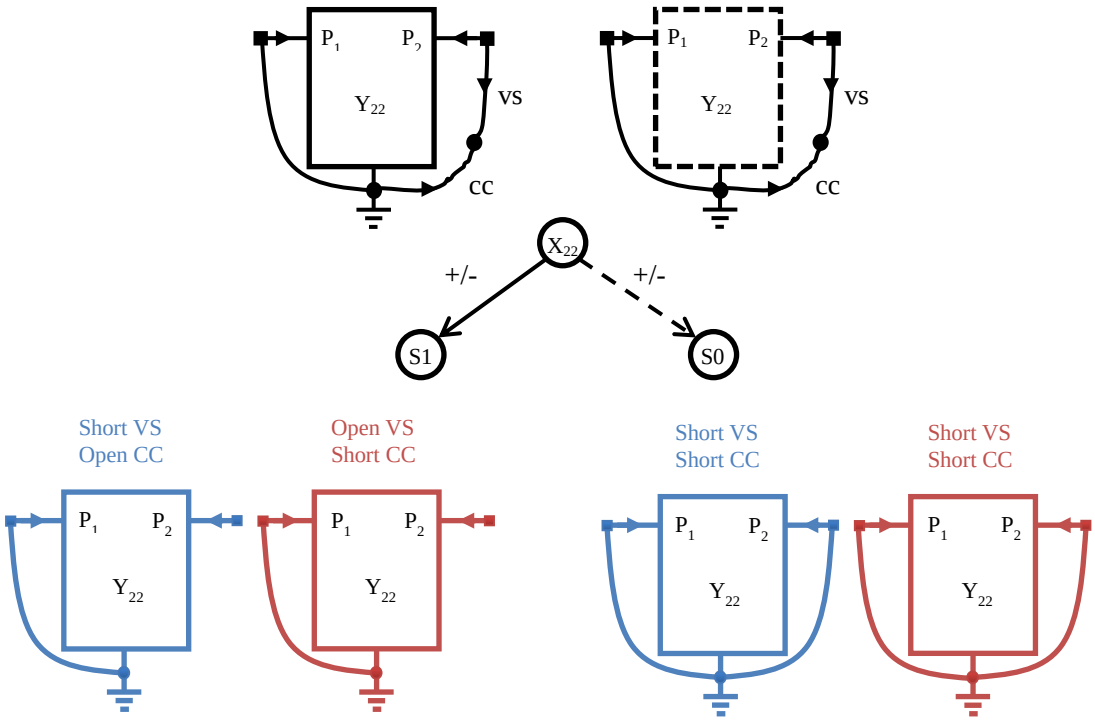


图 3-11 类 Subgraph 的主要成员函数与调用关系
Fig. 3-11 Member functions of the class *Subgraph*

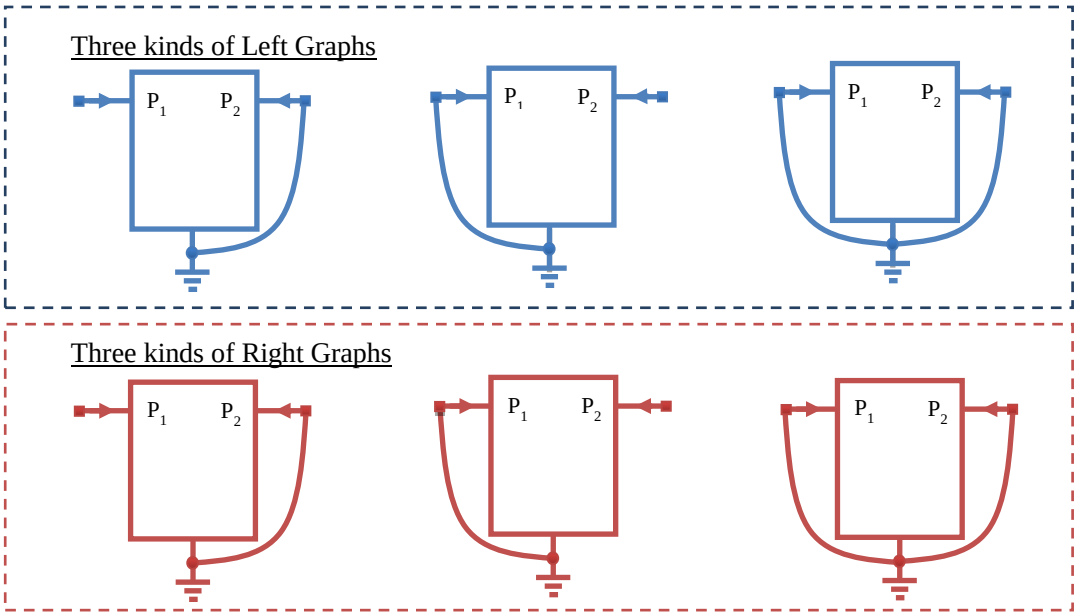


图 3-12 二端口网络的 6 种左右子图

Fig. 3-12 Six kinds of right subgraphs and left subgraphs of the 2-port network

观察图 3-8 到图 3-11，实际上各导纳求解初始图经一步约化后只产生了如图 3-12 所示的 6 种子图（左右子图外部连接一致，但内部可能不同，若相同，例如只含有 Y/Z 这些元件时，那么 6 张图将退化为 3 张图）。而且我们可以总结出一个规律，对于左子图或右子图，按端口是否开路可分为：

a) 端口 1 开路，其他短路;
b) 端口 2 开路，其他短路;
c) 所有端口短路，无开路。

所以对于 2 端口电路共有 2x (2+1) =6 种不同的初始子图。

对于一般 N 端口电路，同样我们容易得到其推广：N 端口电路共有 2 x (N+1) 种不同初始子图。

因此根据上述分析，在具体实现多根 GPDD 时，并非简单的按照第二章所述的算法实现，而是从经一步约化后的子图开始。换言之，我们并不需要构造不同的导纳测试电路，并真地添加一个输入输出受控源（即图 3-8 至 3-11 中根节点对应的图对是不必要的），而只需读入同一个 N 端口的子电路，根据上述方法生成 2N+2 种初始子图，并人工构造好初始的根节点。后续的构造就可以分别从图 3-8 至图 3-11 中的 S0 节点和 S1 节点开始，这样提高了代码的效率与可维护性。

3.4.3 根节点 1-边与 0-边的符号

上一小节确定了初始子图经一步约化后所得的子图，但未给出根节点到两个儿子节点 S0 和 S1 边的符号，本小节给出该符号的确定方法。

回顾规则 2-4，可以总结为图 3-13，即在三种情况下，符号需要变化（初始符号为正）。一是短接的边是 VS 边；二是短接方向（由短接边的较大节点指向较小节点）与有向边的方向相反时；三是原图中小于被短接节点的个数为奇数时（例子中被短接节点为 1，小于它的节点只有 0 这个节点）。

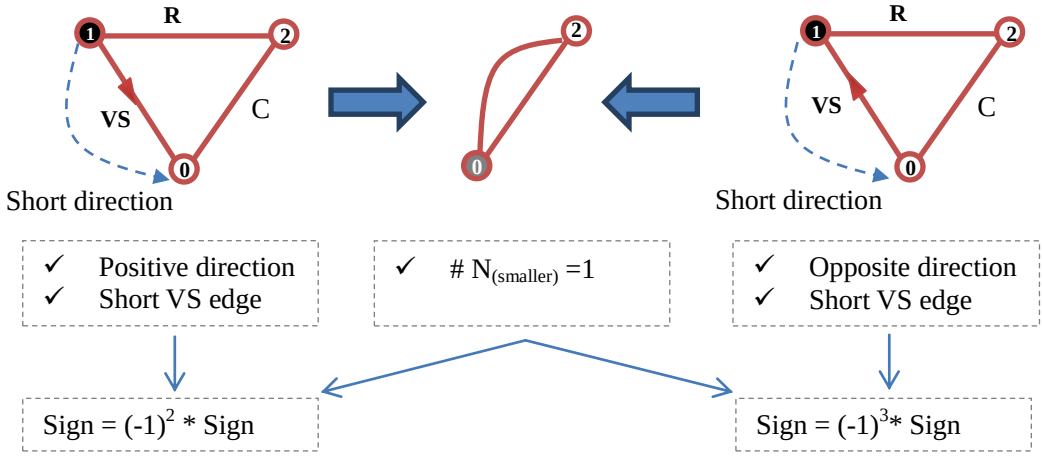


图 3-13 符号转换的三种情况
Fig. 3-13 Cases involving sign flip

现在我们根据上述规则判断由根节点出发两条边上的符号。这里我们假设当前的根节点对应第 i 个端口和第 j 个端口间的导纳 Y_{ij}。

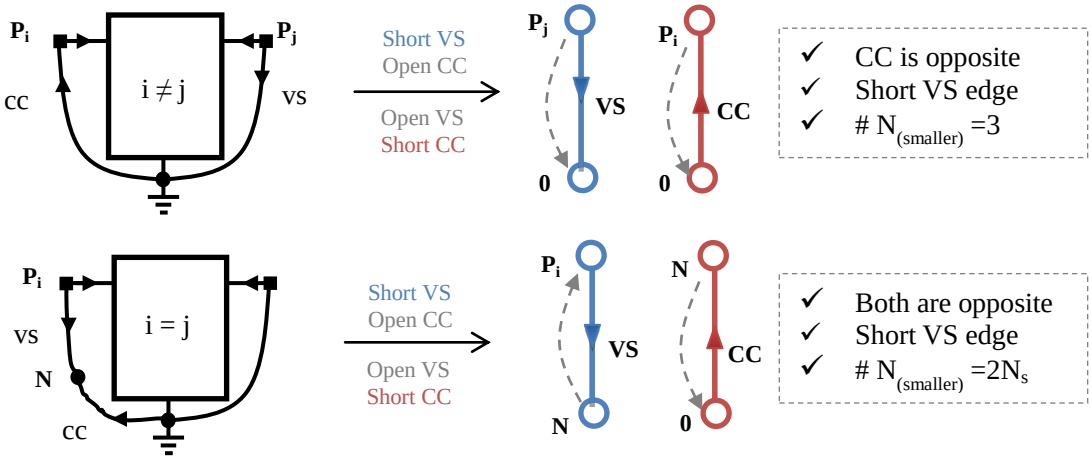


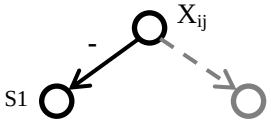
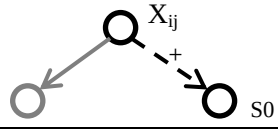
图 3-14 根节点 1-边正负号转换的三种情况
Fig. 3-14 Sign of the 1-edge

如图 3-14 所示，我们先考虑从根节点 X 出发 1-边的符号。此时分为两种情况，第一种是当端口 i 和端口 j 不一致时（$i \neq j$），注意只有短接某一条边时才会发生符号转变，所以我们只要分析左图中 short VS 的操作和右图中 short CC 的操作。又因为短接一条边时总是把大的节点改成小的节点，所以左图 short VS 时是正向的（positive direction），右图中 short CC 时是反向的（opposite direction）。N_{small} 表示比被短接节点 P_i 和 P_j 序号小的节点个数。假设该子电路共有 P 个端口，因为设置子电路时我们可以让端口节点从 1 开始到端口数 P，而保证内部节点都大于 P，比 P_i 小的节点数就只有 0,1, … ,P_i-1 这 P_i 个。又注意到在测端口 i 到端口 j 的导纳 y_{ij} 时，其他所有的端口 P_n （$n \neq i$ 或 j）都被短路到地，即节点 0。不失一般性，假设 $P_i<P_j$，这样的话比 P_i 小的节点数就只有 0 节点这一个，而比 P_j 小的有两个节点，地节点和 P_i。所以 $N_{small} = 3$。综上，short VS 改变一次符号，因为反向又改变一次，且 N_{small} 为偶数，因此该边的符号为负。

对于 i=j 的情况（同一端口的导纳），如图 3-14 所示，我们需要添加一个中间节点 N，显然若我们按序编号，N 要大于该子电路的所有节点。所以此时 N_{small} 为 $2(N-\#Port+1) = 2N_s$，必定为偶数，又 VS 和 CC 边都反向，因此对符号无影响，由于短接 VS 边还会引入一次符号变化，所以无论子电路结构如何，当 i=j 时，根节点的 1-边的正负号为负号。

对于从根节点出发的 0-边，由上一小节可知，其左图和右图的约化规则都是先 short VS，再 short CC。所以无论左图和右图最后是正是负，它们必定同号，因此整个图对的正负号一定是正号。因此求解任意符号化导纳，其根节点两条边的正负号是固定的，表 3-3 给出了根节点 1-边和 0-边正负号的取值。

表 3-3 合法器件
Tab. 3-3 Valid Devices

1-边/0-边	Sign
	Negative
	Positive

至此为止，我们便完成了初始根节点与其两个儿子所构成的三元组的构造，包括 1-边和 0-边的正负号，以及两个儿子节点对应的初始约化图对。此后就可以以这两个儿子节点（S1 和 S0）为根节点按规则 2-4 构造整个 GPDD 即可。

3.5 电路矩阵的构造与符号化求解

经过上一小节的叙述，我们已经知道如何构造一个子电路的符号化导纳矩阵，下一步为了就求解整个电路的传输函数就要构造顶层的 MNA 矩阵，并调用 DDD 进行求解，本章将简要叙述这一过程。

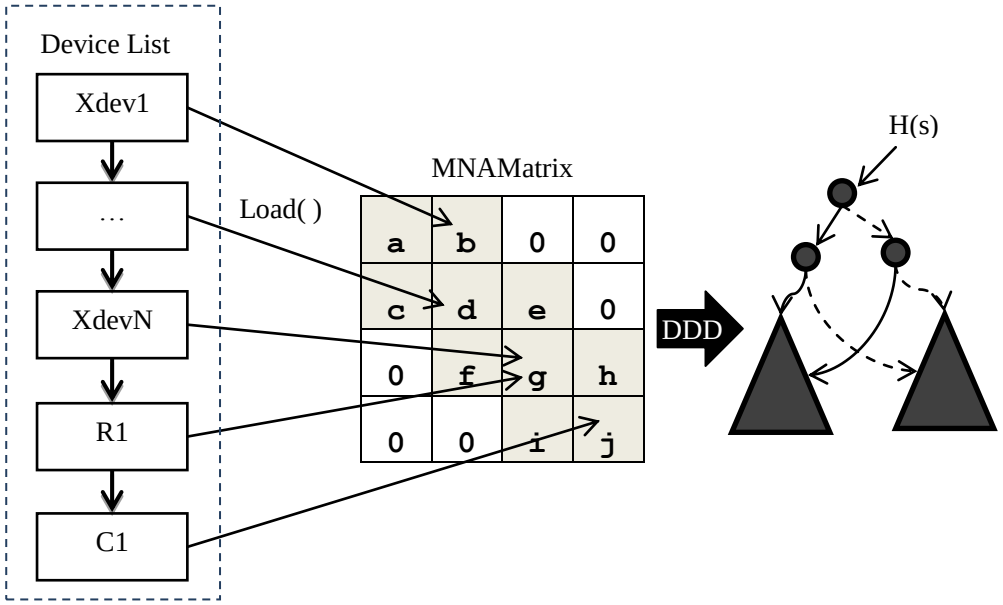


图 3-15 电路矩阵的构造与基于行列式的符号化分析
Fig. 3-15 MNA matrix construction and symbolic analysis based on DDD

如图 3-15 所示，顶层电路类 Circuit 包含一个器件链表（Device List），包括子电路的实例化 X 器件已经其它位于顶层电路的器件(可以是任何其它线性元件，如图中的 R1，C1)。而每个器件类都有一个 Load( )成员函数，该函数会将此器件的导纳矩阵（对于 X 器件将先对符号化的导纳矩阵求值）填入顶层的 MNA 矩阵。因此 Circuit 类只要遍历一遍器件链表并调用每个器件的 Load( )函数就可以构造出顶层电路的 MNA 矩阵。这个过程和数值仿真器 SPICE[15]是一致的。

不同的是，为了进行符号化分析（构造 determinant decision diagram），我们需要将 MNA 矩阵中的非零元素记为符号化分析中的符号（symbol）。为此我们构造了如图 3-16 中的一个数组来保存矩阵中的符号。每个符号含有以下几个域：一是符号的优先级（order），即在构造 DDD 过程中的处理顺序，优先级高的 Symbol 将被先处理，如第一章绪论中所述，order 的优劣对最终 DDD 的大小有巨大影响，本文实现的 order 算法参考了文献[10]中方法，这里不再赘述；另外还要记录该符号在矩阵中的位置（row，col）；最后一个域 val 记录了该符号的具体数值，这里

值得注意的是这个值可能是由多个器件的导纳相加而得。

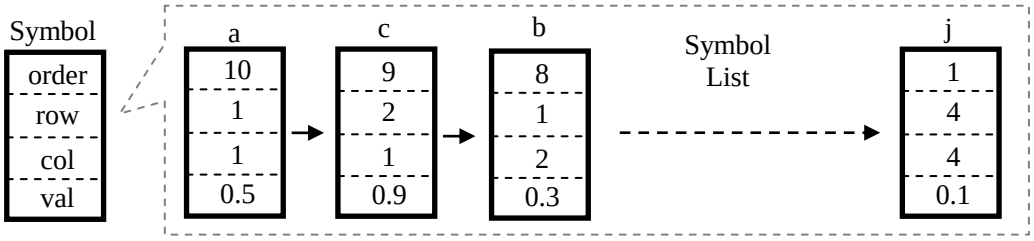


图 3-16 顶层 DDD 的符号列表
Fig. 3-16 Symbol list for DDD

与 GPDD 不同的是，在构造 DDD 的过程中我们约化的对象不再是图对，而是子行列式（minor 或 remainder[25]），如图 3-17 所示。

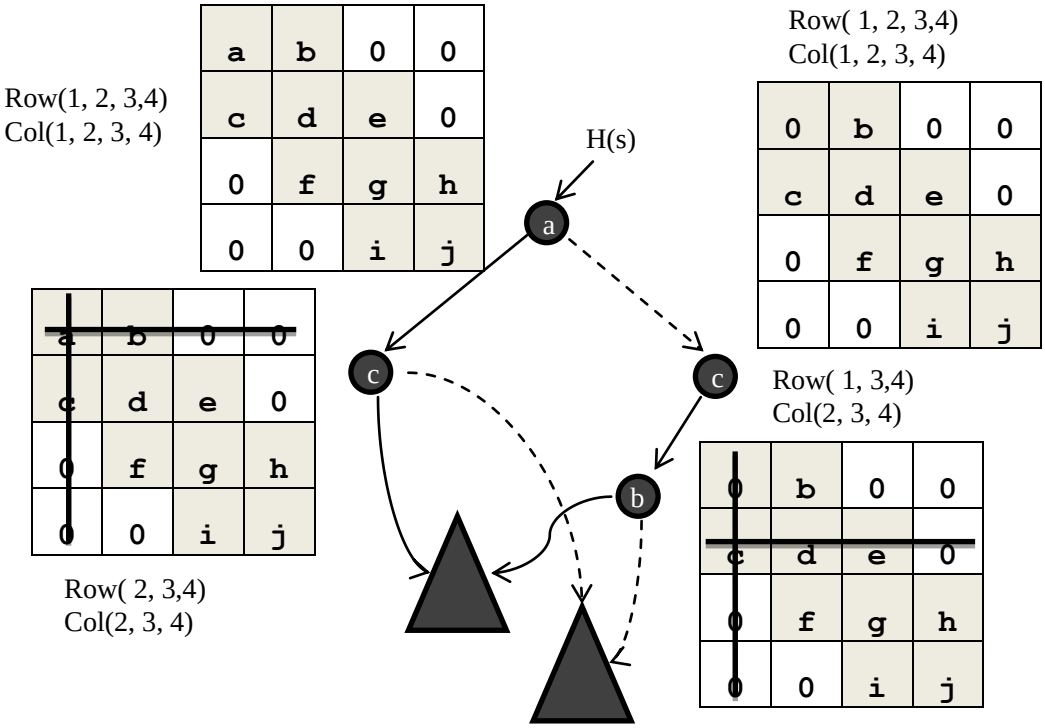


图 3-17 DDD 的构造与数据结构
Fig. 3-17 Date structure of minors and remainders

具体地，我们记录当前子行列式占原行列式的行号和列号。如从节点 a 出发，向左走，那么原始行列式划去 a 所在行和列可以得到约化后的行列式，我们以集合 Row(2, 3, 4)和集合 Col(2, 3, 4)表示该行列式占原始行列式行和列号。其它几个节点对应的标号类似。在共享时，我们正是基于这样的子行列式进行共享。首先根据 Row 和 Col 这两个集合计算出一个哈希键值，若发生冲突再进一步检查。这样就可以构造出整个行列式判别图。详细算法与求解请参阅文献[10]，这里不再赘述。

3.6 本章小结

本章详细介绍了多层次符号化仿真器的实现。包括放着器的架构，网表分析器的实现与网表语法的定义，并给出了三个重要类型：子电路类 Subckt，X 器件类 Xdev 以及多根 GPDD 类 NPortGPDD 的具体数据结构与实现，最后简要叙述了上层电路矩阵的构造与符号化求解。

整个仿真器采用 C++基于对象的开发方式，图 3-1 中仿真器的主要部分都对应着不同的类，类与类之间的调用都通过公有函数（public function）。对于上文所述的各种器件类，都继承了一个虚基类 AbstractDev，并采用了多态的技术，如 Load 函数就是一个虚函数，器件链表中只需存放 AbstractDev 的指针就可以访问各继承类中的 Load 函数，这样的开发方式可以大大缩短开发周期，方便调试与维护。

另外，对于初始图对特殊操作是对第二章中多根 GPDD 构造算法的简化，通过手工构造根节点三元组，减少了初始图对的个数，不仅调高了算法的效率，更方便实际的编程实现。

第四章 测试结果

本文所述的仿真器由 C++编写，在 linux 环境下进行编译，测试。所有测试结果均在实验室服务器上测得，具体测试环境如下所示：

- CPU: Intel Xeon CPU 2.83GHz
- Memory: 16 GB memory
- Operation System: Linux

本章将给出仿真器效率的测试及不同分层策略所导致的不同结果，并与以往文献进行对比，最后给出一些基本结论。

4.1 仿真器工作流程

整个仿真器工作流程如图 4-1 所示。首先将原始网表传给 HSPICE[26]，利用 HSPICE 分析该电路的工作点，然后通过 Perl 脚本，从 HSPICE 的输出文件中提取该电路的小信号模型，将金属氧化物场效应晶体管（MOSFET）或双极型晶体管（BJT）线性化，组成新的网表，该网表满足第三章第二小节所述语法。这个新的线性化网表将被传给本文所述符号化仿真器，最后得到关于频率 S 和各线性电路元件 P 的传输函数，最后经求值得到数值化的曲线。

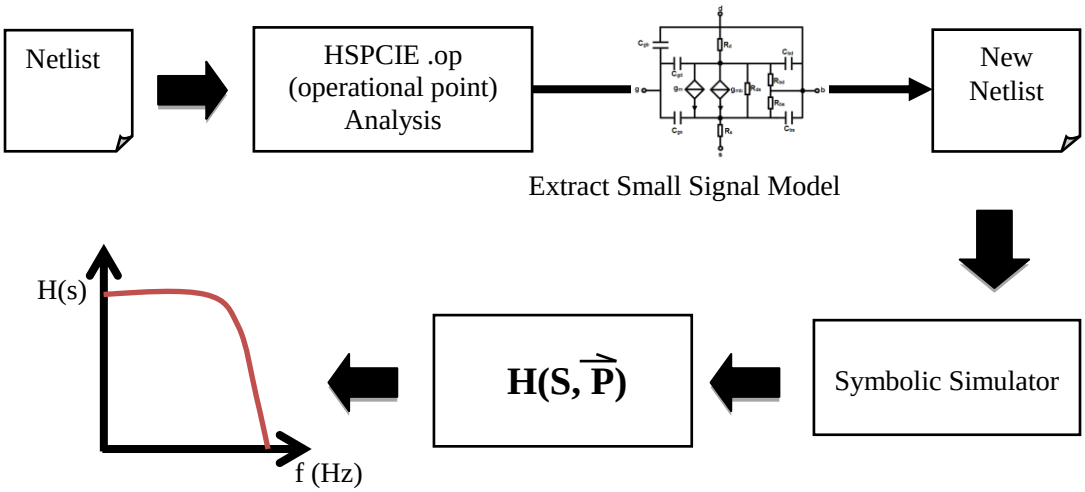


图 4-1 仿真器工作流程
Fig. 4-1 Simulation flow

4.2 基于单个晶体管的分层符号化分析效率测试

模拟集成电路中往往最多的器件就是晶体管，而同一电路中一般所有的晶体管又都使用相同的小信号模型，因此最直观的一种电路结构一致性即单个晶体管小信号模型的一致性。本小节就利用了这一点，将晶体管的小信号模型作为一个基本的子电路，并为其建立符号化的导纳矩阵，而电路中所有晶体管将共享这个符号化的导纳矩阵，从而增强了仿真器的分析能力。

这里共测试了四个具有一定规模的模拟放大器电路：

- 电路一：Bipolar 运算放大器 ua741（如图 4-2 所示）；
- 电路二：Bipolar 运算放大器 ua725（如图 4-3 所示）；
- 电路三：MOS 两级全摆幅 cascode 放大器（如图 4-5 所示）；
- 电路四：MOS 管运算放大器电路（如图 4-6 所示）。

以下小节将按 Bipolar 电路和 MOSFET 电路分别叙述。

4.2.1 Bipolar 电路的测试

前两个测试电路时是双极性模拟运放电路 ua741（图 4-2）和 ua725（图 4-3）。最早的一些非 BDD 结构的符号化分析方法都无法分析这种规模的电路。文献[10]首次利用基于行列式的符号化分析方法（DDD）分析了 ua741 电路，随后文献[12]进一步分析了 ua725 电路。

为了线性化晶体管，在仿真时，我们采用了如图 4-4 所示的小信号电路代替测试电路中所有的晶体管。表 4-1 总结了这两个测试电路的符号化分析的结果。

表 4-1 ua741 测试结果（共享单个晶体管）
Tab. 4-1 Experimental result of ua741 (based on single-transistor sharing)

Works	GPDD 符号数	DDD 符号数	MNA Size	|GPDD|	|DDD|	Run Time	Mem Cost
HybridSim[21]	5	105	24	33	1,755	0.19 s	57 MB
GRASS[12]	81	/	/	31,887	/	1.9 s	35 MB
DDD	/	105	24	/	1,755	0.15 s	10 MB

表中第一行所列的 HybridSim 仿真器即本文所述的符号化多层次的仿真器。

第二行 GRASS 是文献[12]中所述仿真器，第三行 DDD 是我们基于文献[10]的方法自己实现的基于行列式的符号法仿真器，以便测试比较。表 4-1 的前两列分别是 GPDD 和 DDD 所含的符号个数。第 3 列给出了上层电路 MNA 矩阵（方阵）的大小。|GPDD| 和 |DDD| 分别表示双图决策图和行列式决策图所含的节点个数，这两个参数直接影响仿真的效率与内存的消耗。

可以看到，相对于 GRASS[12]，本文所述方法和重新实现的 DDD 都是要快一些的。从具体数据可以看出，虽然 GRASS 所含的 GPDD 符号数仅为 81 个，而本文所述仿真器和 DDD 都有 105 个 DDD 的符号（即在顶层 MNA 矩阵中有 105 个非零元素），但最后得到的 BDD 结构却更小：GRASS 的 GPDD 有 31,887 个节点，Hybridsim 有 33 个 GPDD 节点和 1,755 个 DDD 节点，而 DDD 只有 1,755 个 DDD 节点，导致了三种方法仿真时间与内存消耗的不同。这可能是由于 105 个非零元素对 24x24 的矩阵来说，元素的分布是比较稀疏的，所以 DDD 并不会很大。而对于 GRASS 来说，注意到图 4-4 中共有 5 个元件，所以若采用展平的网表，ua741 电路将含有 100 多个线性元器件，由于采用了复合并联边（lump）的技术[12]，其符号数降为 81 个，但相比矩阵的规模，仍然较大。

对比表中的第一行和第三行，我们会发现另一个结论：基于单个晶体管的划分对图 4-4 所示 Bipolar 模型并无益处。

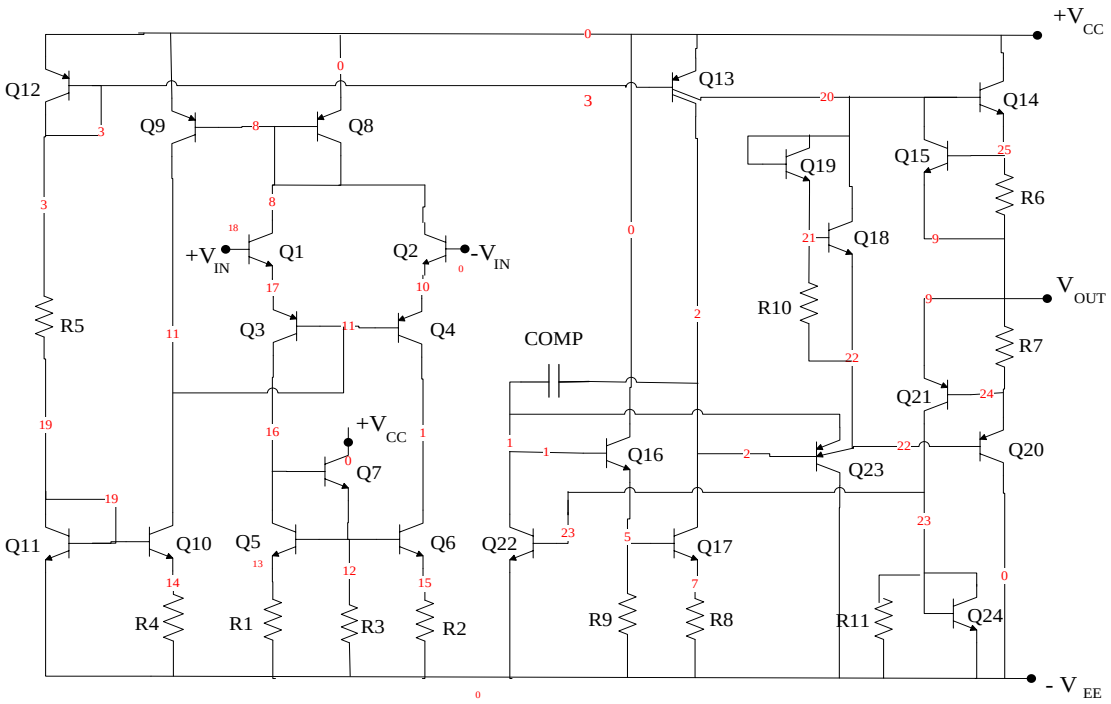


图 4-2 运算放大器 ua741 电路图[10]
Fig. 4-2 Schematic of ua741[10]

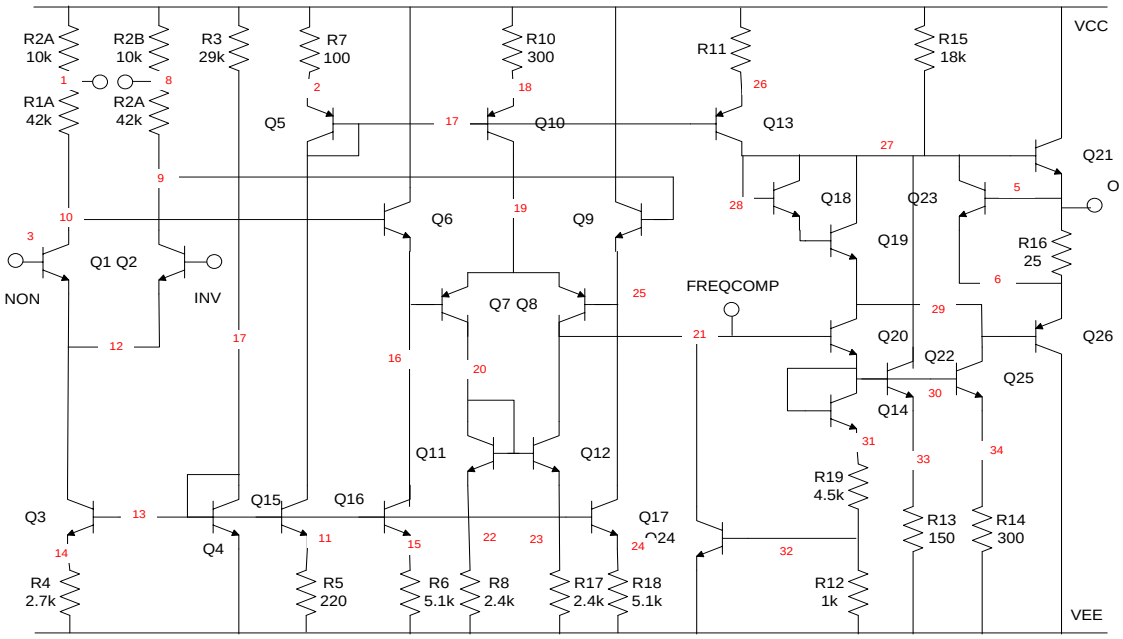


图 4-3 运算放大器 ua725 电路图[10]
Fig. 4-3 Schematic of ua725 [10]

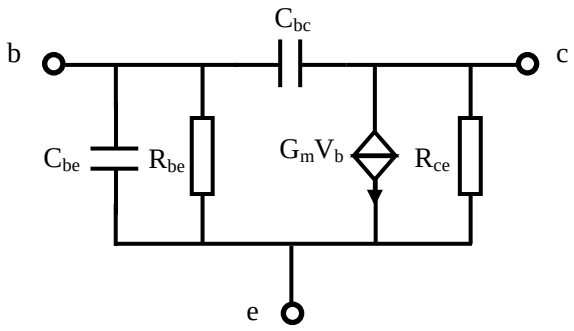


图 4-4 双极型晶体管的小信号模型
Fig. 4-4 Bipolar small signal model

因为图 4-4 中的小信号模型并不包含内部节点，即该子电路的所有节点均为端口节点，所以对于顶层电路而言，并起不到第二章与第三章中所述的降阶的效果。这就是为何表 4-1 中为何本文所述的分层次的符号化分析和非多层的 DDD 会得到一样的 MNA 矩阵大小。因为构造子电路的符号化导纳矩阵还要额外的开销，所以对于该电路，这种多层次分析反而没有单层的 DDD 分析效率高。

表 4-2 给出了 ua725 的测试结果，可以看到，同样的第一行和第三行的 MNA 矩阵大小是一样的，并没有起到预想的效果。但他们都比非层次化的 GPDD 分析效率高

表 4-2 ua725 测试结果（共享单个晶体管）
Tab. 4-2Experimental result of ua725 (based on single-transistor sharing)

Works	GPDD 符号数	DDD 符号数	MNA Size	|GPDD|	|DDD|	Run Time	Mem Cost
HybridSim[21]	5	136	31	33	282,784	2.87 s	137 MB
GRASS[12]	98	/	/	53,420	/	22.6 s	359 MB
DDD	/	136	31	/	282,784	2.81 s	130 MB

4.2.2 MOSFET 电路的测试

第三个和第四个测试电路如图 4-5 和 4-6 所示，均为模拟放大器电路。为了进行符号化分析，需要将 MOS 管展开为对应线性小信号电路，本文所用的 MOSFET 模型如图 4-6 所示，即 SPICE Level 3 模型[27]。

注意到该小信号模型含有 12 个线性的电路元件，因此对于测试电路三（24 个 MOS 管），展平的线性网表将含有约 300 个线性元件，即符号化仿真的符号。而电路四（44 个 MOS 管）将含有约 500 个符号，这远远超过了文献[10, 12]中符号化仿真器的电路处理能力。具体测试时，即使服务器 16G 的内存消耗殆尽，都无法构造出完整的 BDD 结构，这就说明了，如果不采用多层次的结构，虽然 BDD 本身的共享特性能延缓符号化分析规模的指数增长，但可能只是减小了底数，并未改变其指数增长的本质。

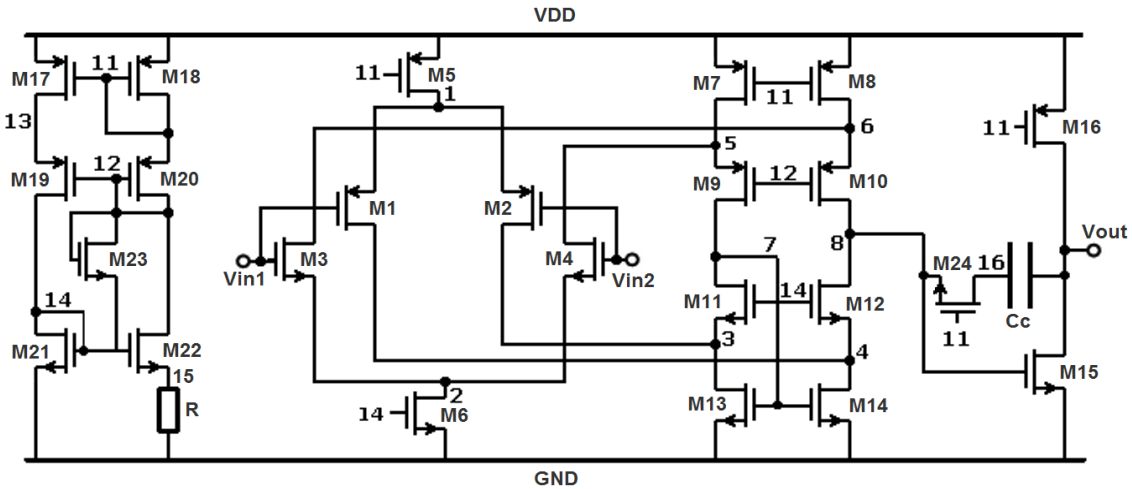


图 4-5 全摆幅 Cascode 放大器
Fig. 4-5 Rail to rail cascode amplier

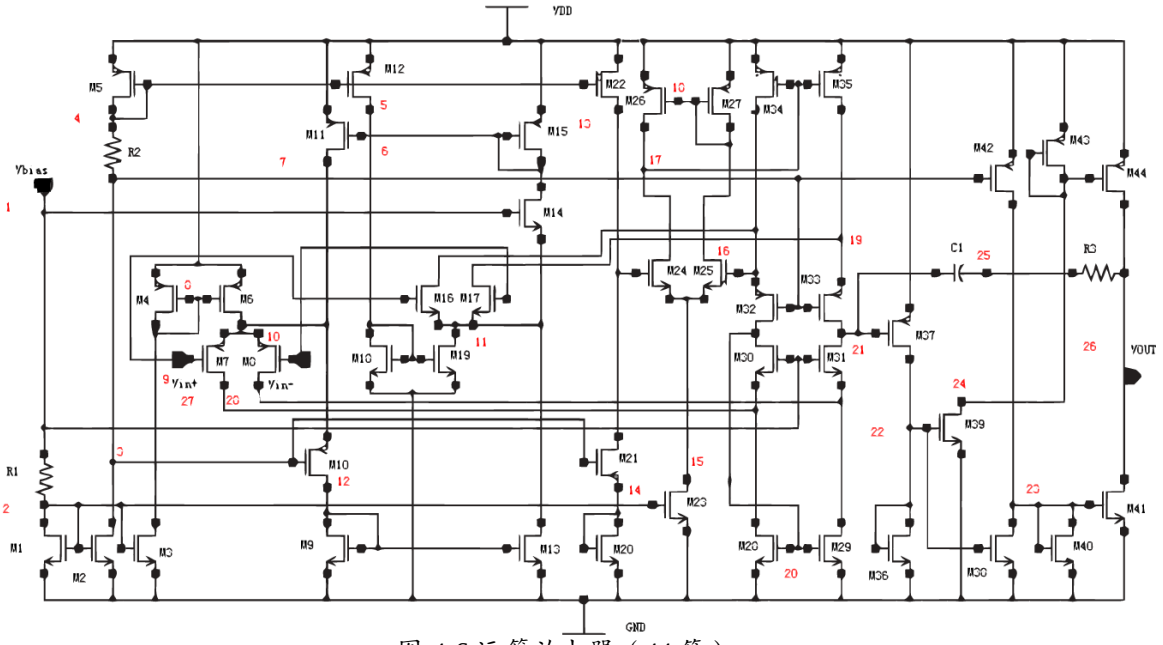


图 4-6 运算放大器（44 管）

Fig. 4-6 MOSFET operational amplifier (44 transistors)

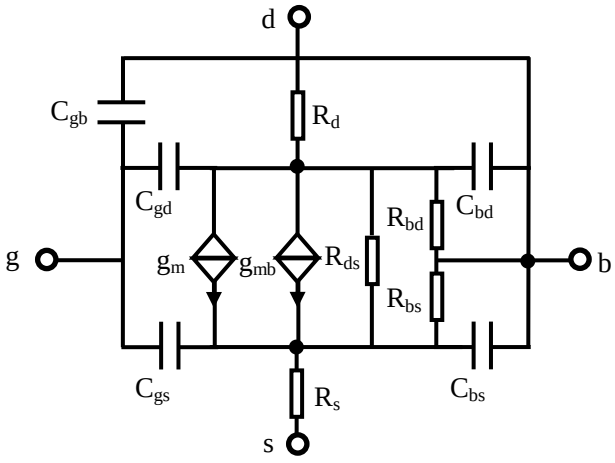


图 4-7 MOS 管小信号模型[27]

Fig. 4-7 MOSFET small signal model[27]

但若将图 4-7 中的小信号模型看做一个子电路，那么情况就会有大大的改善。与 Bipolar 的小信号模型不同，这里使用的 MOS 管小信号模型有 3 个内部节点，因此对于测试电路三的顶层电路就会减少了 24x3=72 个内部节点，对于测试电路四就会减少更多的内部节点。而本身对于图 4-7 而言，其规模很小，GPDD 有足够的能力求解其符号化的矩阵。并且其端口数为 4，虽然多跟 GPDD（multi-root GPDD）的构造时间基本是端口数的平方级的，但由于一般子短路的端口数有限，也不会影响其效率。

表 4-3 电路三和电路四的测试结果（共享单个晶体管）
Tab. 4-3Experimental results of test case 3 and case 4 (based on single-transistor sharing)

Op-amp Circuit	#MOS	GPDD 符号数	DDD 符号数	MNA 矩阵大小	GPDD 节点数	DDD 节点数	仿真时间 (秒)	内存消耗 (MB)
电路三	24	12	104	18x18	234	70,129	1.81	70
电路四	44	12	140	28x28	234	45,716	1.50	91

表 4-2 给出了电路三和电路四的测试结果。因为图 4-6 所示小信号模型有 12 个元件，所以对应 GPDD 有 12 个符号，对应 GPDD 节点数为 234（去除冗余节点后）；测试电路三对应的 MNA 矩阵大小为 18x18，对应 DDD 节点数为 70,129，仿真从 1Hz 到 1GHz，每十倍频 10 个点进行 AC 分析，总时间（包括网表的读取，符号化导纳矩阵的构造，顶层 MNA 矩阵的符号化求解）为 1.81 秒，内存消耗为 70MB。

测试电路四共有 44 个 MOS 管，由于采用同一个小信号模型，所以对应的 GPDD 是一致的；测试电路四对应的 MNA 矩阵大小为 28x28，对应 DDD 节点数为 45,716，AC 分析的总时间为 1.50 秒，内存消耗为 91MB。

从上面的结果可以看出一个有趣的现象，电路四比电路三大，但所得到的 DDD 却更小，导致仿真时间也比较短。这是因为二分判定图的大小受符号处理顺序（order）的影响很大，同时电路矩阵的稀疏度不同也可能导致 DDD 的大小相差很大。从另一角度来看，本文提出的分层符号化分析相比文献[2, 4]中的非多层的符号化分析进一步缓解了 BDD 大小随电路规模增长而剧烈增长的问题。

4.3 电路划分策略测试

上一小节中我们采用的划分策略是基于直觉的，那么这种以单个晶体管为单位的划分策略是否是一种好的策略呢？另外模拟电路往往有其本身的结构性[28]，为此，本小节进行了多种划分的测试，并进行了比较。

4.3.1 Bipolar 电路的划分测试

如上小节所述，图 4-4 中的 Bipolar 小信号电路并不含有内部节点，虽然可以最大程度实现共享（所有双极性晶体管都可以共享一个符号化的导纳矩阵），但无

法减小上层电路的规模，所以导致行列式决策图（DDD）的大小与划分前室一样的，求解速度没有改进。

为此这里尝试了一种人工的划分方法，其基本策略如下：

a) 尽量使用端口数较小的子电路;
b) 子电路的规模不可过大;
c) 尽量利用电路结构的一致性，即每一个子电路被尽可能多的实例化模块共享;
d) 尽量使子电路包含更多的内部节点。

策略（a）考虑的是构造多根 GPDD 的效率，因为符号化导纳的个数正比于子电路端口数的平方，如果子电路端口数很大，这一时间空间开销将会较大，好在一般模拟电路中，我们可以方便的找到一些端口数不大的子电路。

策略（b）考虑的是 GPDD 处理能力的限制，如果子电路过大，那么很可能光一个子电路的多根 GPDD 就耗尽了内存，过大的子电路也就起不到分层效果。

策略（c）是提升符号化导纳矩阵的复用率，从另一个角度而言，就是减少了整个电路中所含子电路的个数，从而使得总的 GPDD 节点保持在一个合理范围之内。

策略（d）的目的较为明显，如果子电路包含更多的内部节点，那么上层电路的降阶效果就越明显。但值得注意的是，这点往往与其他策略是矛盾的，如策略（b），（c），所以实际划分时一定是这四个策略的一个折中，没有一个最优的方案。

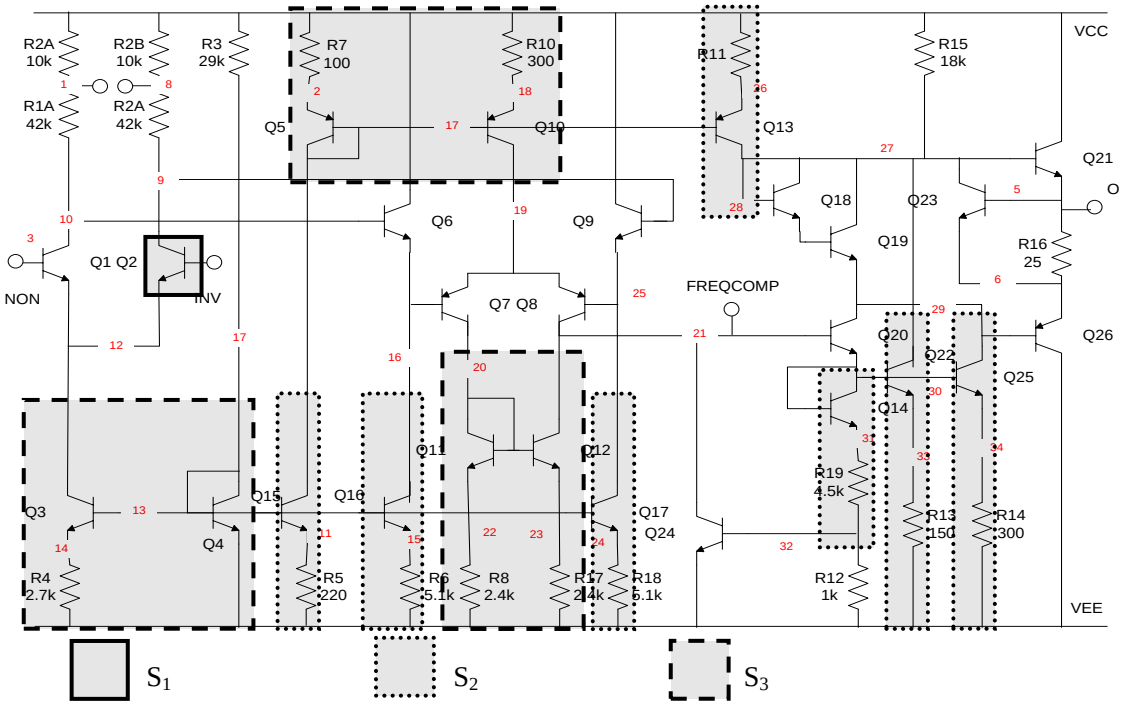


图 4-8 一种启发式的 ua725 电路划分

Fig. 4-8 A heuristic partition of ua725

如图 4-8 所示，根据上述四种策略，我们对 ua725 电路（ua741 电路规模较小故不再尝试不同划分）进行了划分，共包含 3 个子电路，为了描述一种划分，我们以 S_i 表示第 i 个子电路（sub-circuit），并以 X_i 表示第 i 个实例化 X 器件（即第三章所述 XDev 类）。另记 $S_k\{X_1, X_2, \ldots, X_n\}$表示 X_1 到 X_n 是子电路 S_k 的实例化，$X_1(dev_1, dev_2, \ldots, dev_n)$表示实例化 X 器件中包含原顶层电路中的器件 $dev_1, dev_2, \ldots, dev_n$。为了方便，也可简写为 $S_k\{ (dev_1, dev_2), (dev_4, dev_8) \}$，表示子电路有 $X(dev_1, dev_2)$和 $X(dev_4, dev_8)$着两个实例化。

故图 4-8 中的划分描述如下：

- 子电路 S_1：$S_1\{ (Q1), (Q2), (Q6), (Q7), (Q8), (Q9), (Q18), (Q19), (Q20), (Q21), (Q23), (Q24), (Q26) \}$，即一个双极性晶体管的小信号模型，与之前单管划分一致，共有 13 个实例化 X 器件共享 S_1，但每个 X 器件不含内部节点；
- 子电路 S_2：$S_2\{ (Q15, R5), (Q16, R6), (Q17, R18), (Q14, R19), (Q22, R13), (Q25, R14), (Q13, R11) \}$，共有 7 个实例化 X 器件共享 S_2，每个 X 器件含有 1 个内部节点；
- 子电路 S_3：$S_3\{ (Q3, Q4, R4), (Q11, Q12, R8, R17), (Q5, Q10, R7, R10) \}$，共含有 3 个 X 器件，而每个 X 器件含有 2 个内部节点。值得注意的是 S_3 中的 $X_1(Q3, Q4, R4)$，比另两个 X 器件少了一个电阻，但正由于 S_3 的导纳矩阵是符号化的，在求 X_1 的数值导纳时，只需将对应的那个电阻置零，充分利用符号化的优点，从而增加子电路的复用。

表 4-4 电路 ua725 测试结果（启发式的划分）

Tab. 4-4Experimental results of ua725 (based on heuristic partition)

Sub-circuit	**S_1**	**S_2**	**S_3**	**顶层 DDD**
#Symbol	5	11	6	80
#Nodes	33	120	50	12,977
MNA Size	19x19			
T_{all} (s)	0.27			

表 4-4 给出了这种划分下的测试结果。我们可以看到 S_1 所含符号数（#Symbol）为 5，对应 GPDD 节点（#Nodes）为 33 个；S_2 所含符号数为 11，对应 GPDD 节点为 120 个；S_3 所含符号数为 6，对应 GPDD 节点为 50 个，可以看到三个子电路规模都不大，三个多根 GPDD 加起来只有 203 个节点。

另外注意到 MNA 矩阵大小（MNA Size）由表 4-2 中的 31x31 减小到了 19x19，

非零元素由 136(符号数)减小到了 80，从而使得 DDD 的大小也变小了，由 282,784 个节点变为 12,977，只为原来 1/20 不到。所以符号化分析的时间仅为 0.27s（T_{all} 包括网表读取，符号化数据结构的建立，以及 1Hz 到 1GHz，90 个频率点的 AC 响应求值过程）。

综上所述，这种尝试性的划分大大提升了符号化分析的效率。

4.3.2 MOSFET 电路的划分测试

上一小节中，对 ua725 重新进行划分后得到了较好的效果，这一小节尝试对测试电路三和测试电路四做不同划分，并给出更详细的数据，以供对比。

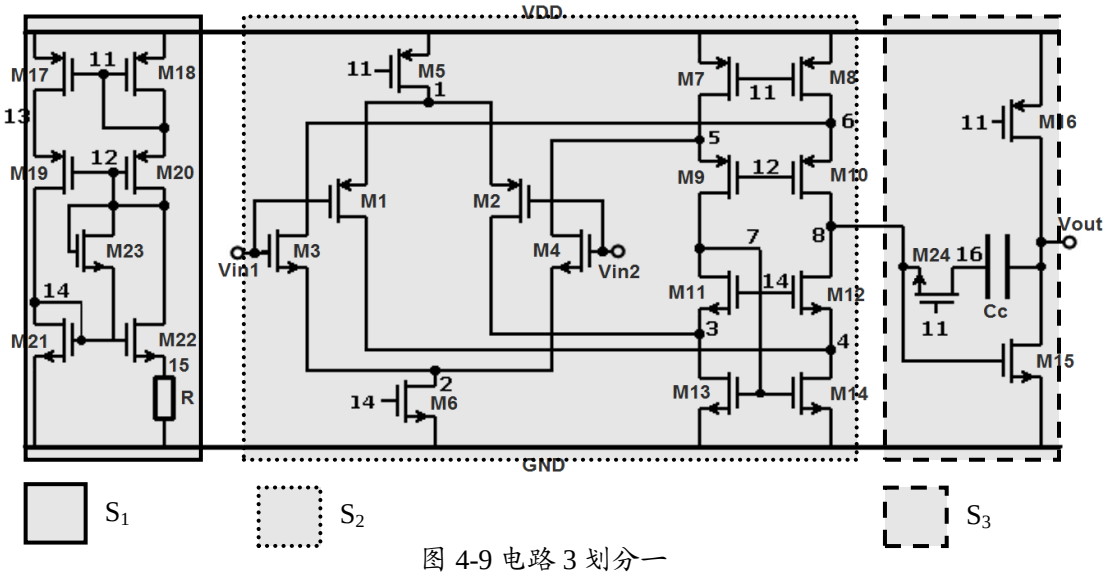


图 4-9 电路 3 划分一

Fig. 4-9 Partition 1 of test case 3

测试电路三划分一：

- 子电路 S_1：S1{ (M17, M18, M19, M20, M21, M22, M23) }，即放大电路的偏置电路；
- 子电路 S_2：S_2{ (M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14) }，即放大电路的第一级。
- 子电路 S_3：S_3{ (M11, M16, M15, Cc) }，即放大器的第二级电路。虽然只有 X(M11, M16, M15, Cc)这一个实例化，但该 X 器件内部包含 10 个内部节点(每个 MOS 管有 3 个，还有图中的节点 16)

与上一小节不同的是，这里尝试根据电路本身的功能模块进行划分，将整个放大电路分为偏置级，第一级与第二级子电路，与设计者习惯相符。

表 4-5 电路 3 测试结果（划分一）
Tab. 4-5 Experimental results of test case 3 (partition one)

Sub Circuits	S_1	S_2	S_3
#Ports	4	5	3
#Symbol	11	154	34
#GPDD Nodes	388	5,867,224	943
#GPDD Nodes Reduced	154	5,176,979	421
#GPDD Nodes Remaining	234	690,272	522
#GPDD nodes sharing	92	2,030,916	323
#Sub-graph sharing	721	15,130,547	1,807
MNA Size	7		
#Symbol	32		
#DDD Nodes	108		
#DDD Reduced	0		
#DDD Remaining	108		
T_{DDD} (s)	0.01		
T_{GRASS} (s)	64.52		
T_{Eval} (s)	20.06		
T_{all}(s)	84.59		

表 4-5 给出了这种划分的结果。表中#Ports 表示子电路的端口数，#Symbol 表示子电路所含的符号数，由于本文所述的 GPDD 也是一种第一章所述 ROBDD（Reduced Ordered Binary Decision Diagram），在完成初步构造后，可以根据三元组进一步去除一些冗余的节点，而这些去除的节点即表中#GPDD Nodes Reduced 一项，而剩余的 GPDD 节点记为#GPDD Nodes Remaining。可以看到通过这一步骤，可以进一步减小 GPDD 的节点数。#GPDD nodes sharing 和#Sub-graph sharing 分别指 GPDD 节点的共享和每个节点对应图的共享，从表 4-5 中可以看出，三个子电路对应 GPDD 中有存在大量的共享，说明 GPDD 如第三章所述，的却十分紧凑。

对于顶层 MNA 矩阵及 DDD 的统计信息，主要有以下几项：MNA Size 表示顶层电路矩阵的大小，与 GPDD 类似，#DDD Nodes 表示在去除冗余节点之前的节点数，#DDD Reduced 表示冗余节点的个数，#DDD Remaining 表示最后剩余的总节点数。

仿真效率分析由三个时间组成：T_{DDD} 表示顶层 DDD 的构造时间，T_{GRASS} 表示所有子电路的多根 GPDD 的构造时间之和，T_{Eval} 表示求值时间（从 1Hz 到 1GHz，

90 个频率点）。

与表 4-3 相比，表 4-5 中所记录的仿真时间大大加长，为 84.59 秒。我们容易发现，子电路 S_1 和 S_3 所对应的多根 GPDD 都较小，而顶层的 DDD 由于三个子电路包含了大量的内部节点，其规模较表 4-3 中结果更是大幅减小，矩阵大小仅为 7x7，DDD 的节点数仅为 108。唯一节点数庞大的是子电路 S_2 对应的多根 GPDD，其节点数为 690,272，远远超过其他所有判定图节点之和。这也解释了为何 T_{GRASS} 占据了 T_{all} 的大部分时间。显然这种划分并非一个好的尝试，因为它违反了上一小节中的划分策略（b），子电路 S_2 的规模太大。

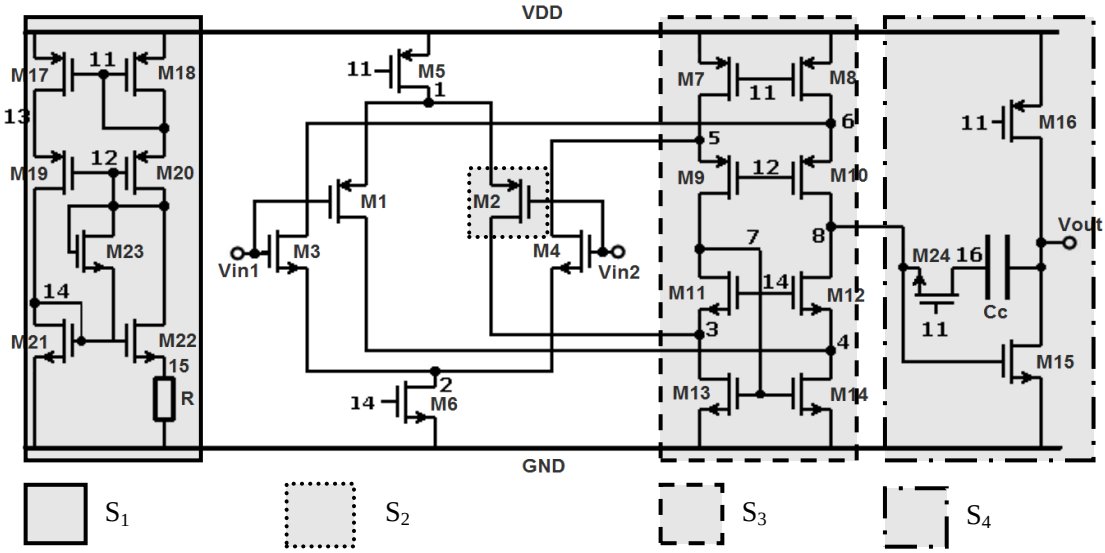


图 4-10 电路 3 划分二

Fig. 4-10 Partition 2 of test case 3

为了改善划分一的结果，我尝试了第二种划分方法，如图 4-10 所示。S_1 和 S_4 分别是划分一中的 S_1 与 S_3，即偏置电路和第二级放大电路。而子电路 S_2 和 S_3 将原来的 S_2 拆开：

- 子电路 S_2：S_2\{ (M1), (M2), (M3), (M4), (M5), (M6) \}，即 MOS 管的小信号电路，所以输入差分对中的每个 MOS 管 M1 到 M6 都共享这一个子电路。
- 子电路 S_3：S_3\{ M7, M8, M9, M10, M11, M12, M13, M14) \}，即第一级电路中的 cascode 部分。

这样一来原本的第一级子电路被拆成了很多块，减小了电路规模。表 4-6 给出了这种划分的测试结果，与预期的一致，仿真效率大幅提高，总时间仅为 0.75s，比基于单管的划分还要快。可以看到虽然由于上层节点的增多，导致顶层电路矩阵的大小变大（14x14），DDD 的大小变大为 30,197，但仍在合理范围内，而最大

的多根 GPDD（S_3）为 6,216，较划分一种的 690,272 减小了两个数量级。所以 GPDD 的构造时间（0.25s）和电路求值时间（0.42s）大大缩短。

表 4-2 电路 3 测试结果（划分二）
Tab. 4-6 Experimental results of test case 3 (partition two)

Sub Circuits	S_1	S_2	S_3	S_4
#Ports	3	4	8	3
#Symbol	78	11	88	34
#GPDD Nodes	3,903	388	14,935	943
#GPDD Nodes Reduced	1,812	154	8,785	421
#GPDD Nodes Remaining	2,102	234	6,216	522
#GPDD nodes sharing	1,266	92	4,680	323
#Sub-graph sharing	7,830	721	34,033	1,807
MNA Size	14			
#Symbol	103			
#DDD Nodes	114,693			
#DDD Reduced	84,496			
#DDD Remaining	30,197			
T_{DDD} (s)	0.08			
T_{GRASS} (s)	0.25			
T_{Eval} (s)	0.42			
T_{all}(s)	0.75			

但有一点值得注意的是，S_3 端口数很多，并且除了这些端口和节点 7，减少的内部节点都是 MOS 管的内部节点，因此若把整个 S_3 都拆成单管，对顶层电路而言，节点数只增加 1。子电路 S_1 虽然只有三个端口，但情况与 S_3 类似，若拆成单管，只会增加节点 15。因此我进一步尝试了划分三，如图 4-11 所示。

- 子电路 S_1 即以图 4-7 中一个 MOSFET 的小信号模型为子电路，这样除了子电路 2 中的 MOS 管外其他 MOS 管都共享这个子电路；
- 子电路 S_2：S_2{ (M11, M16, M15, Cc) }，即放大器的第二级电路。虽然只有 X(M11, M16, M15, Cc)这一个实例化，但该 X 器件内部包含 10 个内部节点（每个 MOS 管有 3 个，还有图中的节点 16）

表 4-7 给出了这种划分的测试结果，如上述分析，顶层的 MNA 矩阵由 14x14 变大为 16x16，但是其符号数由 103 减小为 87，所以划分三的 MNA 矩阵更为稀疏，这也使得对应的 DDD 反而更小，仅含有 9,301 个节点。同时子电路 S_1 和 S_2 也都很小，因此总的仿真时间是三种划分中最小的，仅耗时 0.2 秒。

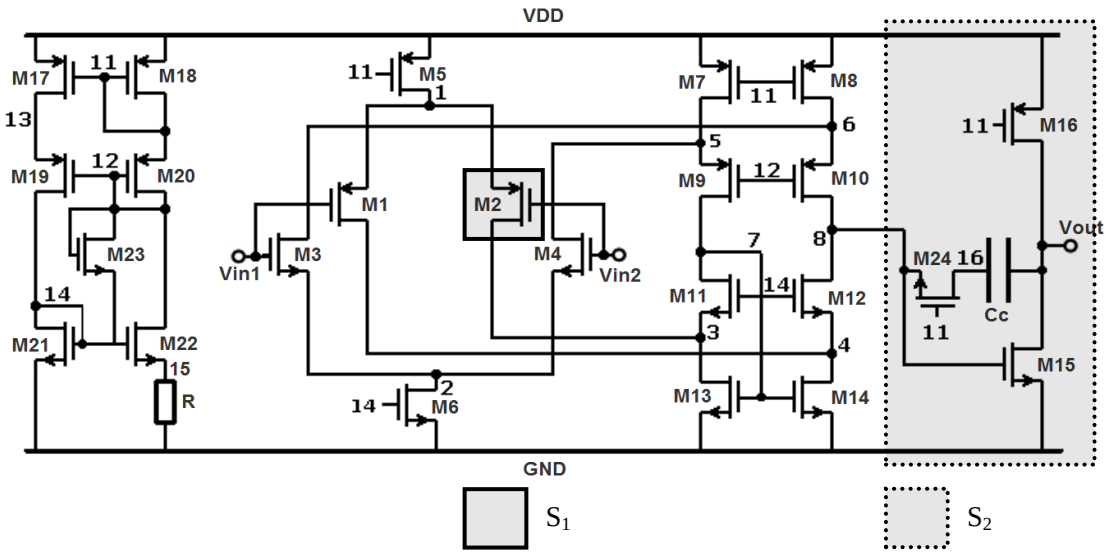


图 4-11 电路 3 划分三

Fig. 4-11 Partition3 of test case 3

表 4-7 电路 3 测试结果（划分三）

Tab. 4-7 Experimental results of test case 3 (partition three)

Sub Circuits	S_1	S_2
#Ports	4	3
#Symbol	11	34
#GPDD Nodes	388	943
#GPDD Nodes Reduced	154	421
#GPDD Nodes Remaining	234	522
#GPDD nodes sharing	92	323
#Sub-graph sharing	721	1,807
MNA Size	16	
#Symbol	87	
#DDD Nodes	16,989	
#DDD Reduced	7,688	
#DDD Remaining	9,301	
T_{DDD} (s)	0.02	
T_{GRASS} (s)	0.07	
T_{Eval} (s)	0.11	
T_{all}(s)	0.2	

对于测试电路 4，我们尝试了类似的一种划分，如图 4-12 所示：

- 子电路 S1：以 MOS 管的小信号电路为一个子电路，所以除了子电路 S2 中的 MOS 管，电路中所有其他的 MOS 管都共享这一个子电路。

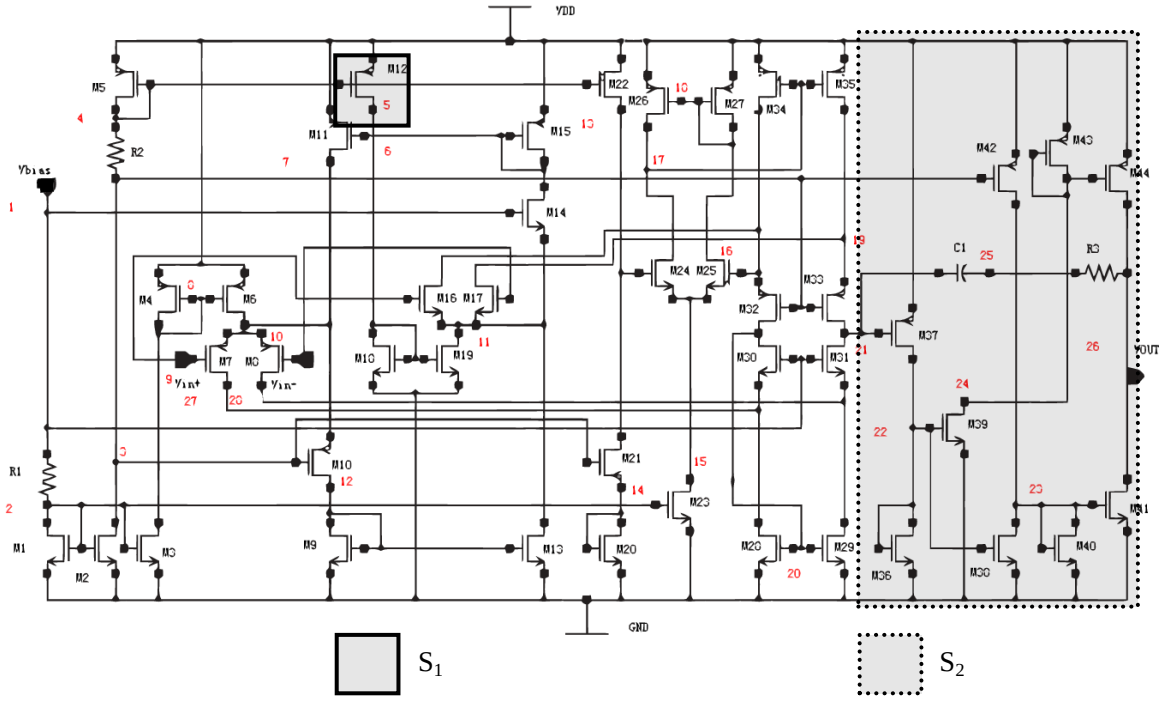


图 4-12 电路 4 划分一

Fig. 4-12 Partition 1 of test case 4

- 子电路 S_2：S_2\{ M36, M37, M38, M39, M40, M41, M42, M43, M44, C1, R3) \}，即输出级和第二级电路。

这样划分的原因是 S_2 所含端口较少，仅为 3 个（节点 VDD 和 GND 在 AC 分析中都接地，不作为端口），且包含一定数量的内部节点。而电路中的其他模块连接都较紧密，若划成一个模块，所包含的内部节点大多为 MOS 管的内部节点，因此不再做划分。

该划分结果如表 4-8 所示，可以看到 S_2 对应的多根 GPDD 节点数也仅为 8,921，而通过划分，上层电路的节点数由 28（表 4-3）减少为 23 个，DDD 的大小也由 45,716 减小为 8,992，因此仿真时间也对应减小，由 1.5 秒变为 0.74 秒。

表 4-8 电路 4 测试结果（划分一）

Tab. 4-8Experimental results of test case 4(partition one)

Sub Circuits	S_1	S_2
#Ports	4	3
#Symbol	11	101
#GPDD Nodes	388	31,695
#GPDD Nodes Reduced	154	22,774
#GPDD Nodes Remaining	234	8,921

表 4-2（续）
Tab. 4-8 (continued)

Sub Circuits	S1	S2
#GPDD nodes sharing	92	11,217
#Sub-graph sharing	721	74,453
MNA Size	23	
#Symbol	114	
#DDD Nodes	37,487	
#DDD Reduced	28,495	
#DDD Remaining	8,992	
T_{DDD} (s)	0.04	
T_{GRASS} (s)	0.24	
T_{Eval} (s)	0.46	
T_{all}(s)	0.74	

4.4 本章小结

本章给出了本文所述的多层次符号化仿真器的测试结果。首先采用基于单管的划分方法与以往的一些符号化分析方法进行了对比。对于双极性电路而言，由于这里采用的小信号模型不含有内部节点，所以这种划分方法并没有取得预想的降阶效果；而对于 MOS 管电路，其小信号模型更为复杂，含有三个内部节点，其降阶效果明显。基于电路这种本身的结构性，单管划分大大减小了 BDD 结构（包括多根 GPDD 和顶层的 DDD）的大小，从而提升了仿真器处理大型模拟运算放大器的能力，而以往仅利用 BDD 本身共享特性的仿真器都无法分析像测试电路三和测试电路四这样大小的电路。

本章的第二小节进行了更多的划分尝试，并给出了一种直观的人工划分策略。从几种划分可以看出，划分的好坏对仿真速度有极大的影响。同时，基于单管的划分是一个不错的选择，虽然在上述几种划分中它不是最快的，但事实上，另外几种都是在其基础上，再根据模块和端口数划分成一些更大一些的模块。我们可以得到一个基本结论，如果模块电路的连接较为紧密，我们不妨保留该模块基于单晶体管的划分，如果能找到符合划分策略的电路模块，那么我们才构建更大的划分。

另一个结论是顶层 DDD 的大小并不简单的取决于矩阵的大小，还和矩阵的稠

密程度有关，如果矩阵维数增大，而符合数却增大不多甚至减小，那么对应的 DDD 反而会变小。

最后将来可以继续研究的是划分的自动化，正如上面所讨论的，我们可以用图论的方法，分析一个图块的连接紧密性，将端口数小，内部节点较多（图稀疏）的模块划分为一个子电路，而那些紧密的图仍保持单管划分。

第五章 结束语

5.1 主要工作与创新点

本文结合了文献[10]和文献[12]的方法，提出了一种新的多层次的符号化分析方法，旨在提高符号化分析处理大型模拟电路的能力。

近年来，基于二分划分图的符号化分析[10, 12]大大拓展了符号化分析的处理能力，已经可以处理像 ua741 和 ua725 这样的基本模拟放大电路，但是对于小信号模型更为复杂和更大规模的电路仍然无能为力。本文所述的多层次的符号化方法进一步拓展了仿真器的能力，使得可以处理较大规模的模拟运放电路，并为进一步的运用打下基础。

本文首次提出了符号化导纳（Symbolic Stamp）的概念，在传统的数值化分析中，我们可以用每个器件的导纳矩阵（Stamp）组装出整个电路的 MNA 矩阵，然后再调用矩阵求解器进行求解。类似地，本文也利用了这一处理方法，不同的是导纳矩阵和顶层的 MNA 矩阵都是符号化的求解的。而符号化会带一个优点，导纳矩阵的基本单位将不再仅是一个器件（如一个电阻或电容），而可以是任意端口的子电路。同时，模拟电路中有大量具有相同结构的电路模块，这样这些模块就可以同时共享这一个符号化的导纳矩阵。

正如上面所述，相比以往基于 BDD 的符号化分析，本文所提出的多层次的符号化分析除了继承了 BDD 结构天生的共享特性，还自然地利用了电路的结构特性，使得整个结构更为紧凑，多了一个电路层面上的共享，这也是为何该方法能大大提升符号化仿真器处理大型模拟电路的能力。

对于每个子电路的符号化导纳矩阵，本文采用了一种基于图约化的方法。每个节点对应一对图，初始图是由电路按一定规则变化而得，电路中的线性元件即图中的边，从根节点出发，在图约化的过程中，就可以构造出一个双图决策图（Graph-Pair Decision Diagram），遍历该判定图中的 1 边，即可求得一对输入输出的传输函数。文献[GPDD]完成了上述这些工作，并给出了实现与理论基础。但由于该方法求的只是一对输入输出的关系，而不能简单地直接用于求解 N 端口电路的 N^2 个导纳，本文在其基础上进行了改动，构造了一种多根的双图决策图（Multi-root Graph-Pair Decision Diagram）来表示一个符号化的矩阵，并且这多个根指向的 GPDD 是共享在一起的，这样使得整个符号化矩阵仍然是一个很紧凑的

结构，充分利用 BDD 的共享机制。

上层 MNA 电路的求解是基于行列式决策图（Determinant Decision Diagram）的，其基本理论在文献[10]中有详细叙述，本文重新实现了一遍，并做了一定的改进。DDD 中每个节点会对应一个行列式，所以在实现 DDD 结构共享时，我们直接采用该行列式的信息来计算哈希函数的键值，进一步提高了构造效率。

通过多种测试，我们可以发现本文所述的多层次符号化分析的性能大大优于以往的符号化分析方法。如第四章中的测试电路三和测试电路四，以往的符号化分析方法根本无法处理，而现在本文所述仿真器可以在 1 秒内得出结果。另外，通过对同一电路不同划分的测试，我们发现基于单个晶体管的自然划分是一种较优的划分，而在此基础上，可以将一些连接较稀疏的电路元件划为同一模块，而保持连接较紧密处的单管划分，这样得到划分也有较好的仿真结果。上述的一些尝试可以为将来划分的自动化提供重要参考数据。

5.2 研究展望

本文的一个终极目标是简化模拟电路的设计。早在我学习模拟电路的时候就深感模拟电路设计之精深，不经过些锤炼，必定云里雾里，更别谈设计。很多时候只是盲目地反复通过数值仿真器进行仿真，与预订指标进行比较，然后再回头更改电路参数，这个设计循环负责漫长，关键是仿真器为设计者提供的指导信息实在有限。因此我们的最终目标是构造一个更智能的仿真器，能为广大学习模拟电路的学生或是实际的模电电路工程师提供更简单更有效的设计平台。

事实上，本仿真器只是该远达目标的第一步，但也是重要一步。不能处理较大规模的模拟电路一直以来是符号化仿真的通病，而本文所述方法在一定程度上解决了这个问题，能够精确的得出电路的符号化传输函数。因此就可以基于该符号化结构进一步求解电路传输函数对应电路中某一个参数的敏感度（即归一化的导数），而基于敏感度就可以对目标性能进行有方向性的优化。虽然由于一般模拟电路的设计自由度和解空间十分巨大，即使有了优化方向，也不一定能在这么大的一个空间中找到最优解，但相比其他一些模拟电路综合的方法（如模拟退火，遗传算法等），基于敏感度的方法不失为一种好的尝试。基于本仿真器的一些最新的运用，读者可以参阅文献[20]，我们进行了一些基于电路元件敏感度的电路优化。

另外，结合友善的图形界面（Graphic User Interface），我们可以为设计者提供根据互动性的设计环境。只要电路的拓扑结构不发生变化，而仅仅电路元件值改变，将不影响符号化传输函数的结构，符号化仿真器可以快速地算出新的电路响

应。因此，当用户调整某一电路参数时，我们可以尝试实时地显示输出响应波形的变化，这样设计者可以直观地看到所调整的电路参数对于整个电路的影响。这是传统数值化仿真器无法达到的。

相信通过未来的继续努力，本文所述的多层次符号化分析方法一定能为设计者带来更多的便利与帮助。

参考文献

[1] 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.

[2]Q. Yu and C. Sechen, "A unified approach to the approximate symbolic analysis of large analog integrated circuits," *IEEE Trans. on Circuits and Systems–I: Fundamental Theory and Applications*, vol. 43, no. 8,pp. 656 – 669, 1996.

[3] P. Wambacq, R. Fernández, G. E. Gielen, W. Sansen, and A. Rodríguez-Vázquez, "Efficient symbolic computation of approximated small-signal characteristics," *IEEE J. Solid-State Circuit*, vol. 30, no. 3, pp. 327–330,1995.

[4]P. Wambacq, R. Fernández, G. E. Gielen, W. Sansen, and A. Rodríguez-Vázquez, "A family of matroid intersection algorithms for the computation of approximated symbolic network functions," *in Proc. Int'l Symposium on Circuits and Systems*, 1996, pp. 806–809.

[5] O. Guerra, E. Roca, F. V. Fernández, and A. Rodríguez-Vázquez, "Approximate symbolic analysis of hierarchically decomposed analog circuits," *Analog Integrated Circuits and Signal Processing*, vol. 31, pp.131-145, 2002.

[6] S. X. D. Tan, W. Guo, and Z. Qi, "Hierarchical approach to exact symbolic analysis of large analog circuits," *in Proc. Design Automation Conference*, 2004, pp. 860–863.

[7] A. Doboli and R. Vemuri, "A regularity-based hierarchical symbolic analysis methods for large-scale analog networks," *IEEE Trans. On Circuits and Systems – II: Analog and Digital Signal Processing*, vol. CAS-48, no. 11, pp. 1054–1068, 2001.

[8] M. M. Hassoun and P. M. Lin, "A hierarchical network approach to symbolic analysis of large-scale networks," *IEEE Trans. on Circuits and Systems – I: Fundamental Theory and Applications*, vol. 42, no. 2, pp.201–211, 1995.

[9] X. D. Tan and C.-J. R. Shi, "Hierarchical symbolic analysis of analog integrated circuits via determinant decision diagrams," *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems*, vol. 19,no. 4, pp. 401–412, April 2000.

[10] C.-J. Shi and X.-D. Tan, "Canonical symbolic analysis of large analog circuits with determinant decision diagrams," *IEEE Trans. on Computer-Aided Design*, vol. 19, no. 1, pp. 1-18, Jan., 2000.

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

[12] 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.

[13] 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.

[14] Online: http://en.wikipedia.org/wiki/Binary_decision_diagram

[15]L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Ph.D. dissertation, University of California, Berkeley, CA, May1975.

[16] H. Xu, G. Shi, and X. Li, "Hierarchical exact symbolic analysis of large analog integrated circuits by symbolic stamps," *in Proc. ACM/IEEE Asia South-Pacific Design Automation Conference (ASP-DAC)*, Yokohama, Japan, Jan. 2011, pp. 19-24.

[17] J. Y. Lee, X. Huang, and R. A. Rohrer, "Pole and zero sensitivity calculation in asymptotic waveform evaluation," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 5, pp. 586–597, May 1992.

[18] G. Shi and X. Meng, "Variational analog integrated circuit design by symbolic sensitivity analysis," *in Proc. International Symposium on Circuits and Systems (ISCAS)*, Taiwan, China, May 2009, pp. 3002–3005.

[19] D. Ma, G. Shi, and A. Lee,"A design platform for analog device size sensitivity analysis and visualization," *in Proc. Asia Pacific Conference on Circuits and Systems (APCCAS)*, Malaysia, Dec. 2010.

[20] X. Li, H. Xu, G. Shi, and A. Tai, "Hierarchical Symbolic Sensitivity Computation with Applications to Large Amplifier Circuit Design," *in Proc. IEEE International Symposium of Circuit and System (ISCAS)*, Brazil, May 2011, pp. 2733-2736.

[21]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。

[22] 陈硕，"模拟电路层次化分析方法在符号化电路仿真中的应用，"上海交通大学微电子学院硕士论文，2010 年 3 月。

[23] Online: http://flex.sourceforge.net/

[24] Online: http:// www.gnu.org/s/bison/

[25] G. Shi, "A simple implementation of determinant decision diagram," *in Proc. International Conf. on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2010.

[26] Online: http://www.synopsys.com

[27] A. Vladimirescu and S. Liu, "The simulation of MOS integrated circuits using SPICE2," EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL

M80/7, 1980.
[28] T. McConaghy and G. E. Gielen, “Globally reliable variation-aware sizing of analog integrated circuits via response surfaces and structural homotopy,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 11, pp. 1627–1640, Nov. 2009.

致谢

思源湖、篮球场、实验室、圣南区、二食，6 年的时光早已让交大成为我最熟悉的地方。因为这里有一路支持我，聆听我，指导我的老师和朋友们，谢谢你们，是你们让校园变得那样熟悉、自然，那样朴实、固执。

我的指导教师施国勇教授便是这样一位有些固执的人，至少在很多人眼中如此。一大早，他的车总是孤零零地停在学院停车场上，回邮件，写代码，准备课件，读论文，每天如此，专心学术。施老师时不时还会亲自跑到实验室，和我们讨论代码的具体实现，研究课题的设想等等。“这没什么值得尊敬的，因为敬业是任何人最基本的义务，在美国如此，在日本也如此。”施老师曾经和我们这样说过。但我不得不敬佩施老师能多年如一日的坚持，况且在这个浮华的时代，能坚持自己的理想实为不易。这也是为何我们常开玩笑说，您走路的时候有股强大的气场，让人不禁敬畏三分。很荣幸我能成为您的学生，我不光学会了做研究的方法，从课题的提出、调研，到算法的设计，从代码的实现到论文的撰写，更重要的是您严谨认真的作风潜移默化地感染着我。谢谢您多年的指导，相信无论在什么领域，这份做事业的固执将令我终身受用。

另一位我要特别感谢的老师是赵峰老师，虽然您已经离开学院，但我仍然记得您“赵大侠”的风采，上课风趣幽默，为人豪爽，典型的北方哥们。在研究上，我要感谢您对我们上海市创新项目的指导，这也是我的第一个正式科研项目，第一次发表论文。更要感谢您让我参与了人脸检测的项目，让我有幸经历了一次甜蜜的法国之旅。那时候我在准备 GRE，学校的出国手续又一如既往的麻烦，我都一度想放弃，是您给了我很大的支持，还陪我一起去银行汇款，交会议注册费。赴法当天由于会议地点不在巴黎，不能直飞，我一路折腾，飞机加高铁，超过了 24 小时才到达位于昂热的酒店，好在演讲很成功，教授们听得饶有兴趣。而在我办理签证时邂逅了我以前的高中同学，现在的女友 Minzhi，不经感慨赵老师的英明。

当然要比洒脱，英明的赵大侠还是比不过我们敬爱的李爷的。现在回想起来，早在我大一的年代，我们已领教了他的狂放不羁。那天李爷给我们大一新生做演讲，他带了个棒球帽（后来知道这顶帽子出自大名鼎鼎的 UC Berkeley），穿着运动衣，一屁股坐在讲台上给我们讲课，着实把我们给镇住了。虽然您不是我的指导教授，我还是从您的那些传奇经历中学到了很多。一人勇斗两名歹徒的是您，

为学生讨公道上访教育部的是您，美网上加油声震慑李娜对手的球迷是您，我看到的不仅是一位教授，更是一位直率的长者。您的呐喊让我们这些年轻人自愧不如，在您的字典里没有妥协，唯有勇往直前。

当然在长长的感谢中，我不会忘了 EDA 实验室的所有同学们。晓鹏是我的 partner，感谢你和我一起在实验室埋头写代码的每个夜晚，没有你的帮助，我们不会这么快有研究成果。也要感谢翁彬彬和张贺两位女生，很开心能与你们共事。感谢郝志刚博士，作为在实验室呆得最久的男人，你给了我很多的帮助，也恭喜你即将顺利毕业，希望你未来的事业和生活蒸蒸日上。感谢上一届的两位大牛，黄伟坚和马迪铭学长，我在你们身上看到了智慧与热情。希望黄伟坚学长创业成功，马迪铭学长在 Synopsys 有所作为。还要感谢上上界的李骥学长，骥哥的称号名副其实，对实验室的所有人都很照顾。同时也要谢谢温柔的大学姐王婷和爱哭的小学姐陈硕，你们是当时实验室最认真的，我的研究课题也受到了陈硕学姐的启发，祝愿你们二位事业有成，早日成家。最后谢谢所有学弟学妹们，谢谢熬夜看球还能精力充沛的陈家骏，谢谢和施老师一样认真严谨的朱彦，谢谢写代码飞快的宋阳，谢谢认真的董兰兰、孙涛、程建东，希望你们每个人都能再接再厉，学业有成。特别地，祝愿朱彦和宋阳能拿到心仪的美国名校 offer，程建东能肩负起 EDA 实验室的大旗，任重而道远。

最后我要感谢我生命中最重要的一些人，感谢我的父母，你们总是默默付出，聆听我的想法，支持我，关怀我。而我却很少注意到你们日益增多的白发和皱纹，一声谢谢在父母的伟大面前其实十分无力，但我还是要借此机会向你们道一声谢谢，我会在以后的时光中多多关心你们，照顾你们。另外我要感谢 Minzhi，给了我相识十年后的一次美丽邂逅，为我分忧，给我快乐，希望我能与你一起保持那份天真，一起看世界。

交大是我最熟悉的一个地方，毕业十分，难免不舍，愿母校保持那份朴实与固执，愿所有交大人都事业有成，待来年，饮水思源。

攻读硕士学位期间已发表或录用的论文

[1] Xiaopeng Li, Hui Xu, Guoyong Shi, and Andy Tai, "Hierarchical Symbolic Sensitivity Computation with Applications to Large Amplifier Circuit Design," *in Proc. IEEE International Symposium of Circuit and System (ISCAS)*, Brazil, May 2011, pp. 2733-2736.

[2] Hui Xu, Guoyong Shi, and Xiaopeng Li, "Hierarchical exact symbolic analysis of large analog integrated circuits by symbolic stamps," *in Proc. ACM/IEEE Asia South-Pacific Design Automation Conference (ASP-DAC)*, Yokohama, Japan, Jan. 2011, pp. 19-24.

[3] Hui Xu, Ran Ju, and Feng Zhao, "Hardware Architecture for Object Detection Based on AdaBoost Algorithm," *in Proc. International Conf. on Computer Vision Theory and Applications (VISAPP)*, Angers, France, May 2010, pp. 420-424.

[4] Hui Xu, Yifan Qu, Yan Zhang, and Feng Zhao, "FPGA Based Parallel Thinning for Binary Fingerprint Image," *in Proc. of the 2009 Chinese Conf. on Pattern Recognition. (CCPR 2009) and the First CJK Joint Workshop on Pattern Recognition (CJKPR)*, 4 pp., Nov 2009.