申请上海交通大学硕士学位论文

符号化电路仿真器的一个层次化分析方法[1]

学　　校： 上海交通大学

院　　系： 微电子学院

班　　级： B0721091

学　　号： 1072109004

姓　　名： 陈硕

专　　业： 电路与系统

导　　师： 施国勇 教授

上海交通大学微电子学院

2009 年 12 月

[1] 此研究由上海市浦江人才基金（项目编号 07pj14053）和国家自然科学基金（项目编号 60876089）资助

A Dissertation Submitted to Shanghai Jiao Tong University for Master Degree of Science

A New Hierarchical Method for Symbolic Analog Circuit Simulation

Author： Chen, Shuo

Specialty: Circuits and System

Advisor: Prof. Shi, Guoyong

School of Microelectronics

Shanghai Jiao Tong University

Shanghai, P.R.China

December, 2009

符号化电路仿真器的一个层次化分析方法

摘 要

符号化模拟电路仿真器 GRASS（ Graph Reduction Analog Symbolic Simulator）应用拓扑方法分析电路。但由于不对电路进行分块，当电路规模增大到一定程度，GRASS 的性能遇到瓶颈。本文讨论用电路分解方法实现层次化分析。原电路的输入输出由子模块的频域传递函数的代数运算构成。实验证明这种电路分解方法可以有效解决符号化分析复杂度随电路规模指数增长问题，可以高效快速处理含有 44 个 MOS 管子的电路（等效小信号模型有 117 个电路节点，358 个元件），进一步延伸了符号化仿真器 GRASS 的仿真能力。

关键词：层次化分析，符号化仿真器，导纳矩阵，高斯消去

A New Hierarchical Method for Symbolic Analog Circuit Simulation

ABSTRACT

GRASS, (Graph Reduction Analog Symbolic Simulator) analyzes a linearized network topologically. Without partitioning the circuit, the performance of GRASS degrades when the circuit size exceeds certain level. This paper proposes a new circuit partitioning method that takes the advantage of topological analysis framework of GRASS so that the network function of the original network is composed algebraically of the network functions of the partitioned sub-circuits., which are solved by GRASS without running out of memory. Experimental results show that the new method is effective in alleviating the performance bottlenecks in symbolic circuit analysis, the method can be used to handle MOS circuits consist of 117 nodes and 358 elements effectively.

Key words- hierarchical symbolic analysis, symbolic simulator, nodal admittance matrix, Gaussian elimination

目 录

图 目 录

表 目 录

第一章 绪论

随着片上系统（SOC）设计逐渐成为大规模集成电路设计的主流，模拟、射频电路模块的有效实现也变得越来越重要。然而由于模拟电路的性能同时受电路，版图工艺等多参数的影响，使得模拟电路的自动化程度远远低于数字电路。

模拟电路仿真器目前主要有基于数值分析和基于符号化分析两大流派。工业界主要使用基于加州大学伯克利分校(University of California, Berkeley)开发的数值电路仿真器SPICE。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。从本质上讲，数值型仿真器验证预定尺寸的电路是否满足设计性能要求，因此，设计一个电路通常需要重复计算多次。

而对于符号化仿真器，由于采用的是对电路的自变量（时间与频率）、因变量（电压和时间）以及符号化电路中的元件来计算电路特性和行为的方法，来得到符号形式的公式，清晰的揭示了电路特性，无需多次迭代，同时也避免了数值计算过程中的舍入误差。另外，符号化模拟电路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]。

1.1 符号化分析的定义与基本方法

对电路进行符号化分析，意味着在整个的仿真过程中，电路中的元器件（部分或者全部）一直以符号化的形式表示（不代入具体的数值）。传统的符号化分析主要用于电路的复频域特性分析中。很多时候我们对一个电路进行符号化分析是为了得到它的符号化传输函数。对于一个线性，时不变网络，它的传输函数可以写成如下形式[7]：

$$H(s,P)=\frac{N(s,P)}{D(s,P)}=\frac{\sum_j b_j(P)s^j}{\sum_i a_i(P)s^i},P=\{p_1,p_2\cdots p_n\},\ n\le n_{all} \quad (1\text{-}1)$$

其中s是复数形式的频率变量，P是包含电路中的可变参数（如元器件的值）的集合，n是电路中可变参数的数量，n_{all} 表示电路中的所有元件数目。a_i 和 b_j 是由电路中可变参数组成的多项式。

如下图1所示的电路，传输函数的符号表达式为

$$H(s)=\frac{u_o}{u_s}=\frac{g_1}{g_1+sc_1} \qquad (1\text{-}2)$$

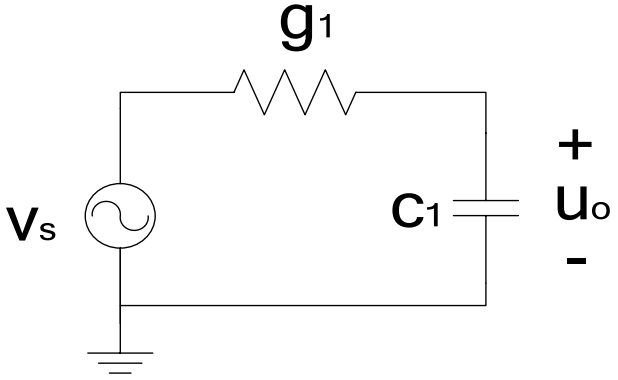


图 1 符号化分析电路举例

Fig.1 Symbolic analysis of a simple RC circuit

我们结合这个例子说明全符号化分析（fully symbolic analysis），部分（partial）符号分析以及代数(algebraic)分析[28]:

1) 如果分析电路给出的仿真结果是（1-2），那么进行的即是全符号化分析。此时电路中的所有元件均以符号（symbol）形式在传输函数中出现。
2) 电路中的部分元件是固定的数值，在传输函数中，我们只能以电路中的部分元件作为可变量分析，这即是部分符号化分析。例如，我们规定c_1为1f,只分析g_1对于传输函数的影响，那么此时的传输函数变成$H(s)=\frac{g_1}{g_1+s}$。
3) 电路中的所有元件均是固定的数值，例如图1电路中，假设g_1为1，c_1为0.5，那么传输函数则退化成只含有一个符号变量的多项式，此时$H(s)=\frac{1}{1+0.5s}$。

1.2 符号化分析的基本方法

目前具体的符号化电路分析算法大约可以分为五种：树枚举法、信号流图法、参数提取法、数值插入法和矩阵行列式法[1,28]。

1) 树枚举法的主要思想是应用了电路节点导纳矩阵的行列式等于电路图（此处

的电路图是指按照相应的构建规则生成的图）中所有生成树对应项的和，这些项等于每条树边对应权重的积。我们的GRASS仿真器即是属于这种方法（文献[8][10]）

2) 流图法主要有梅森（Messon）信号流图法，寻找电路方程中各阶项对应的路径和回路后，根据梅森公式计算可得指定分析结果[6]。仿真器例子有SCYMBAL[11]。
3) 参数提取法基于电路方程组所对应的行列式，通过递归分解行列式，例如每次将行列式分解成包含所需提取参数的部分和不包含所需参数的部分来得到最终的结果。适用于部分电路符号化的分析，仿真器例子有NAPPE[17]。
4) 数值插入法基于电路某些工作点的数值分析结果进行分析。适用于仅以频率变量为符号的分析以及规模较大的电路。仿真器例子有[18]。
5) 矩阵行列式法基于符号化计算行列式（例如使用符号化的高斯Gauss消去法或者递归使用拉普拉斯Laplace展开法），仿真器例子有ISAAC[12，13],DDD[16],ASAP[27]。

1.3 层次化符号分析的优势

层次化符号分析同时具有层次化分析和符号分析的优势。相对于数值仿真器，符号化仿真器有如下优势[1]：

1) 深刻揭示电路行为特性。符号化分析方法的结果是符号形式的，不仅能验证电路的特性是否符合设计要求（即通过代入具体的数值进行数值运算），并且能根据符号化的分析结果，给出具体的改善建议。例如图1所示的电路，当我们需要提高输出端电压值时，根据求得的符号表达式（1-2）知道减小电容C_1或者增大导纳G_1可以达到要求。
2) 电路参数迭代调整。由于在模拟电路的设计过程中，经常需要调整电路的参数以获取期望的性能，符号化的仿真器只需要将不同的数值代入表达式中即可快速得到分析结果，而无需重新分析电路。
3) 当从电路中删除元件时，符号化仿真器只需将该元件的值设为零代入原始的符号表达式中即可得到正确的数值分析结果，无需重新分析电路，数值仿真器则需要重新分析。

符号化分析同样有着缺点，由于采用的是符号化分析，需要存贮大量的符号化中间量，生成项，这也导致了运算效率会比较低，并且由于电路中的符号化生成项会随

着电路尺寸（电路中的元件个数，电路节点数）的增加指数增长，这使得适用于符号化分析的电路尺寸受到了限制。

部分符号化分析，符号化近似分析，和层次化分析电路[32]都可以有效解决上述瓶颈。由于在模拟电路中，只有部分元件对电路的性能影响较大[30],部分符号化分析在符号表达式中只有部分器件以符号的形式出现，使得求解问题规模降低，因此可以分析比较大规模的电路。符号化近似分析是通过舍弃数值幅度较小的项来实现的，这样做牺牲了精度，可分为计算后近似（Approximate After Computation, AAC）计算时近似（Approximate During Computation, ADC） [4，29,30]顾名思义，计算后近似方法是在已经得到符号化分析结果后进行近似分析的,近似时分析时在符号化分析的过程中进行近似分析，这样具有较好的近似意义，但是难度也更大，难点在于需要在符号分析产生的过程中就确定近似的分析机制已经差错控制机制。层次化符号分析也是采用将降低每次处理的问题规模（即将大电路划成相互独立的小块），分析小规模问题后，再合并分析得到完整的问题解，同时由于采用层次化符号分析后，每个小问题间都是独立的，这样有利于使用并行算法实现，使得运算时间所需更短，这是其他两种技术所不具备的优点。综上所述，即是层次化符号分析在具有了符号化分析的同时，也为并行算法的实现提供了可能。在下一节中，我们将介绍层次化符号分析的基本方法与分类。

1.4 层次化符号分析的基本方法与分类

使用层次化符号分析可以有效解决符号化分析在电路规模过大时所遭遇的性能瓶颈,下文将介绍电路划分-合并,或者采用嵌套表示表达式2种层次化分析技术[21]。

1.4.1 电路级分层

为了便于并行算法的实现及合并划分后的小电路块的分析结果,我们在划分电路时需要注意的是每个子模块之间必须是相互独立的(即不能有耦合作用或者控制关系等),这一条件是容易满足的,因为我们可以将相互关联的器件放入同一个模块内部,而不是放入不同模块。

划分完成后，电路将出现层次的概念，划分后的电路具有的层次如图2所示。

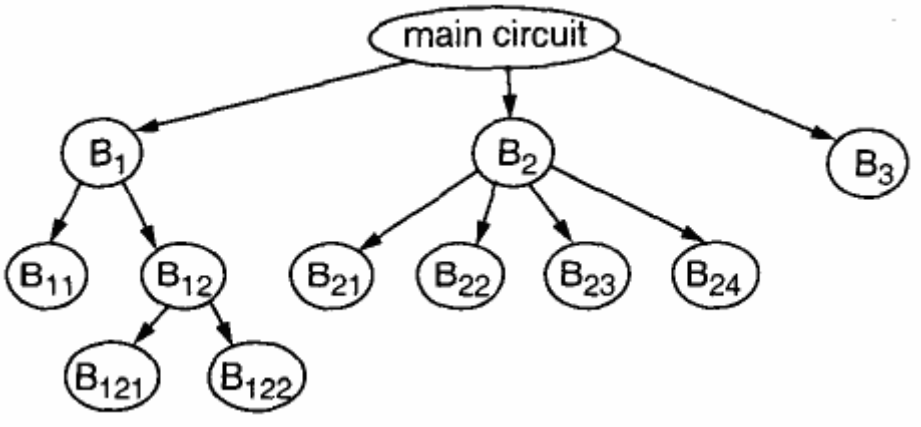


图 2 电路分层示意图

Fig.2 Hierarchy diagram of circuit

要将大电路转化成小电路，必须对电路进行撕裂，有基于边撕裂，基于节点的撕裂和混合撕裂[2]，基于边的撕裂就是 “撕去”某些支路，使得大的电路网络分解成多个小的相互独立的子网络，基于节点撕裂的特点是它仅以节点电位作为电路方程的未知变量。混合撕裂是为了减少撕裂数量，既有支路撕裂又有节点撕裂，下图3即是混合撕裂的例子，为了将图3中左图N_1、N_2和N_3三个网络撕开成3个相互独立的网络，需要进行的支路撕裂为{ b_5,b_6,b_7 }，节点撕裂为{ n_1,n_2 }，如果仅仅进行支路撕裂，需要撕裂的支路为{ b_1,b_2,b_3,b_4, b_5,b_6,b_7 } 共需要进行7次撕裂，如果仅进行节点撕裂，至少需要撕裂的节点为{ n_1,n_2,n_3,n_4,n_5}，需要进行5次节点撕裂。

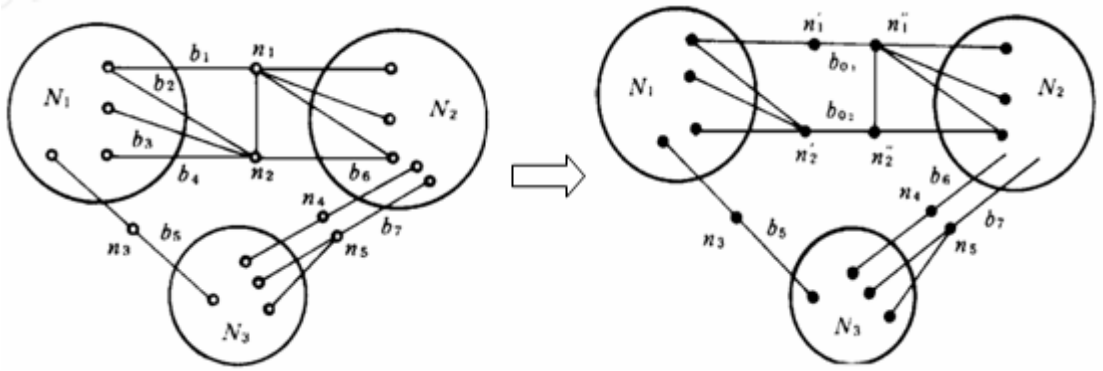


图 3 电路撕裂

Fig. 3 Tear of circuit

电路级分层求解电路的步骤为[21]：

1） 划分电路（用户划分，或者使用自动划分），出现终端电路块，如图2中的

B_{122}

2) 对每一终端电路块（例如图2中的B_{121}）运用相应的分析方法分析（例如使用基于代数方法，就可以得到RMNA矩阵（改进节点法矩阵），如果使用基于拓扑结构法，则需要得到相应的图）。

3) 合并子电路块，得到中层（Middle）块，例如图2中的B_{12}的分析结果

4) 不断重复上一过程，直至回到主电路层（例如图2中的 Main circuit）

在本文中，我们没有刻意区分我们所使用的是支路撕裂还是节点撕点，因为我们分析的时候使用模块时，将模块可看出是具有多端的元件，求得每个模块的信息后，可直接将他们贴起来（类似于元件的stamp），在2.2.1我们将会结合具体的电路给出由于分层而引入的一些概念。

1.4.2 表达式级分层

采用符号化分析大规模模拟电路的主要问题是由于传输函数表达式中符号化生成项随电路规模成指数增长，通过嵌套的形式表示以实现生成项表示的复用。但是这种方法较难计算敏感性分析，比较难揭示电路的特性。

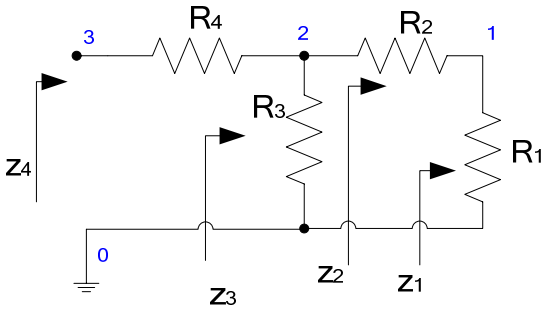


图 4 电阻阶梯电路[7]

Fig. 4 Resistive ladder circuit

文献[7]中给出了一个例子来说明使用嵌套表达式（Sequence of Expressions）的优点。

以图4中的电路为例，如果使用网络的输入阻抗这种方法来求Z_4，得到的展平的符号表达式为

$$Z_4 = U_{in} \Big/ I_{in} = \frac{R_4R_1 + R_4R_2 + R_4R_3 + R_3R_1 + R_3R_2}{R_1 + R_2 + R_3} \tag{1-3}$$

但是如果采用嵌套表达式，系列表达式如下：

$$\begin{aligned} Z_1 &= R_1 \\ Z_2 &= R_2 + Z_1 \\ Z_3 &= \frac{R_3 Z_2}{R_3 + Z_2} \\ Z_4 &= R_4 + Z_3 \end{aligned} \tag{1-4}$$

按照（1-4）系列形式存贮Z_4相比（1-3）需要的中间项少很多。

目前在已有的层次化分析电路的符号化仿真算法有以下几种

1） 基于行列式的算法。

算法的基础是改进节点法（Modified Nodal method），将电路变量分为内部变量，边界变量，外部变量，方程如下，由于每次的子电路块合并，导致的结果是原子电路块的部分外部节点变量变成了内部变量，因此每次通过高斯消去法消去会变成内部变量的矩阵块，从而得到只包含边界变量和外部变量的矩阵，利用拉普拉斯（Laplace）定理展开得到行列式的值，由于展开公式与布尔(bool)函数的展开有相同的形式[5]，因此采用了ROBDD来表示这一过程，（也正是由于采用了这一技术，使得DDD成为第一个即使不采用分层技术也可以精确分析ua741的符号化仿真器，最后利用克拉默（Cramer）法则来求解系统的传输函数[4]。

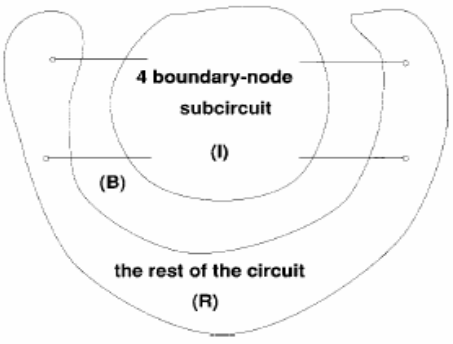


图 5 基于行列式的层次化分析

Fig.5 Hierarchical analysis based determination

电路中的结点总是可以分为内部节点和边界节点，当使用了层次化分析方法后，对于电路内的子模块节点，可以分为内部节点、边界节点和剩余节点，那么可以

将电路特性方程写成如下式（1-5）形式

$$\begin{bmatrix} A^{II} & A^{IB} & 0 \\ A^{BI} & A^{BB} & A^{BR} \\ 0 & A^{RB} & A^{RR} \end{bmatrix} \begin{bmatrix} x^{I} \\ x^{B} \\ x^{R} \end{bmatrix} = \begin{bmatrix} b^{I} \\ b^{B} \\ b^{R} \end{bmatrix} \qquad (1\text{-}5)$$

写成这种形式后，原理上都是使用高斯消去法消去内部节点变量，这样递归消去子模块的内部节点直至顶层电路，便只剩下顶层电路的边界节点信息。

2） 基于信号流图

对于每一块电路生成相应的信号流图，使用梅森（Mason）公式分析每一个子块，在合并子块时，按照连接关系使用相应的化简规则消去内部节点，只保留上一级的输入，输出节点[15]。

3） 双端口网络

电路总是能由一维的盒链（block-net）或者二维的盒网络（block-net）连接而成，这些盒子分别代表了单个的传输管（transistor）和传输管对（transistor-pair）[19],并且引入了A矩阵来表示输入端口的电压、电流与输入端口的电压，电流间的关系，枚举了盒（block）间所有可能的连接关系，对应的A矩阵的计算公式，来实现分层[20]。

4）基于电路拓扑结构

本文后面将要描述的层次化分析算法即是属于这种。利用GRASS（Graph Reduction Analog Symbolic Simulator ）[8,10]求解电路的约化节点导纳矩阵后结合高斯消去法，符号化消去内部节点得到指定的分析结果。

[24]中的算法基于群（cluster）技术，找出群中耦合（coupled）紧的元件组合成一个群，并且保证群中的元件数少于用户定义的上限，同时群间的连接最少。但是这种划分无法保证每块是均匀大小的。[22]通过对包含有互连（interconnect）信息的电路矩阵中的行交换排列来实现树的二分，同样无法保证划分是均与的，并且只能实现二分。[23]提出了一种基于DDD的的可均匀，多层次划分的算法，初始时基于连接关系进行划分，后来通过搬移节点评估收益函数来进行迭代的改进。

1.5 本章小结

本章主要介绍了符号化分析定义及基本方法，层次化符号分析同时具备了符号

化分析的优势，并且只要保证划分时每个模块都是相互独立的，则可以使得每个子模块独立求解，使求解的问题规模小很多，从而提高了分析效率，同时为并行算法实现提供方便，最后介绍了符号化分析的基本方法和分类，并举例说明了采用嵌套表达式后，与使用项展平表达式相比，更节省空间。

第二章 基于电路拓扑结构的层次化分析方法

由于本文实现的层次化分析工作部分基于对 GRASS(Graph Reduction Analog Symbolic Simulator)的调用，因此本章首先将简要介绍 GRASS 的分析方法及实现，然后介绍基于电路拓扑的层次化分析方法。

2.1 GRASS 介绍

GRASS 分析基于电路的拓扑结构，在读入符合特定前提条件的电路网表文件后，按照构图规则将电路转换成有向图，然后只要按照文献[8]中提出的图处理规则进行约化操作，则可以保证不产生冗余项，这是其他基于拓扑结构符号化仿真器不具有的优点。

G. Minty 在文献[25]中提到可通过图约化来得到生成树。对于图中的一条边，总可以将该图的所有生成树分成 2 类：即包含该边和不包含该边。在图中直接删除该边后的子图生成树即是原图中所有不包含该边的生成树；在图中将该边短接后（即将该边的 2 个节点汇合成一个节点）所得的子图生成树加上该边后即是原图中所有包含该边的生成树，不断对得到的子图进行这类边操作，直至只剩下一个节点（此时意味着已经形成了生成树）。至此，图约化操作与二分判定图（Reduce Ordered Binary Decision Diagram; ROBDD）很好的结合起来（初始有向图对应着 ROBDD 的根节点，包含或者不包含边操作对应 ROBDD 的变量 1-分支和 0-分支，可形成生成树与不可形成树对应 ROBDD 的终结点 1 和 0）。文献[10]正是利用了这一特性，将图约化操作与二分判定图结合起来，从而将 GRASS 的仿真能力提高到可对含有 20-30 晶体管电路进行精确分析。同时由于二分判定图中的每一个节点均对应着电路中的元器件，因此 GRASS 还具有生成项展平（flat）的优点,这使得它能深入地揭示电路的特性。

2.1.1 可分析电路需满足的条件

文献[10]中提到如果要使用 GRASS 分析电路，该电路必须满足下列条件：

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。

2.1.2 有向图构建规则

对于受控源，有如下规定：

压控电压源（VCVS）：$U_i = E_{i,j}U_j, I_j = 0$

压控电流源（VCCS）：$I_i = G_{i,j}U_j, I_j = 0$

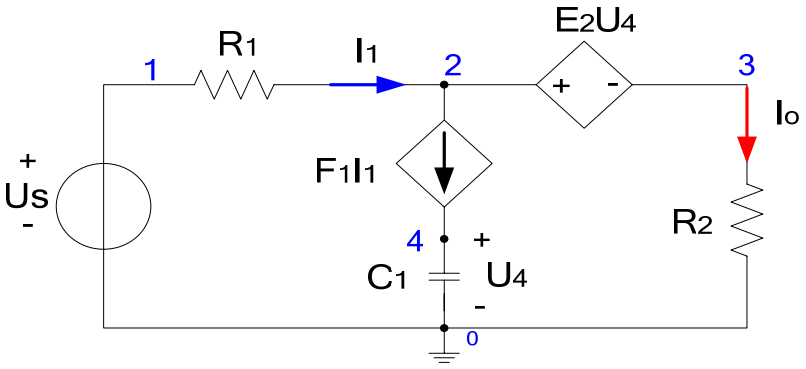
流控电压源（CCVS）：$U_i = H_{i,j}I_j, U_j = 0$

流控电流源（CCCS）：$I_i = F_{i,j}I_j, U_j = 0$

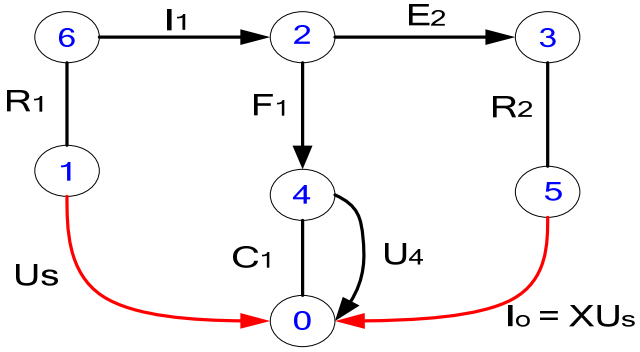
从上述规定中，可以看出处理受控源时有特别的要求，即如果是电压作为控制源，则要求流过它的电流为零；如果是电流作为控制源，则要求它两端的电压降为零。这些规定也决定了我们的有向图转换规则，下面即是电路转换成有向图的规则：

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，如图 6(b)中的 Us 边。
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），如图 6（b）中的 U_4。
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），如图 6（b）中的 I_1。
d) 所有的有向图中的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) 输入端抽象成受输出端控制的受控源，例如图 6（b）中的 U_s 和 I_o，控制系数为 X。



(a)



(b)

图 6 有向图构建规则举例

（a）待分析电路;（b）转换后的有向图

Fig. 6 A circuit example

根据电路传输函数公式： $H(s) = \dfrac{Output}{Input}$

整理该公式后有：$Input = \dfrac{1}{H(s)} \times Output$，之所以做这种处理，也是由于上文给出的理想受控源的规定。以图 6（a）中的电路为例，输入为理想电压源，求流经 R_2 的电流，如果按照通常的理解，输出端受输入控制，则应视为 VCCS，可是流经 Us 的电流并不为零，但是如果转而将输入看作受输出控制，则等效为 CCVS，对应的需满足的规定为电流控制源两端的电压降为 0，根据有向图转换规则（c），电流控制源对应的有向边是额外增加的，两端确实没有电压降，从而满足 CCVS 的

规定。因此在 GRASS 中，反而是输入受输出控制，并且这个受控源的系数就是传输函数的倒数，记为 X。

2.1.3 有向图约化规则及算法

GRASS 通过枚举有向图的生成树来得到电路特性符号化表达式中的生成项，由于不是所有的生成树都对应有效的生成项（有些生成项会相互抵消，可以被抵消的项即为冗余项），文献[8]给出了一套约化规则并给出了严格的数学证明，证明按照该规则约化图，则不会产生冗余的生成项。

下文会给出有效生成树对及有效生成树定义。

有效生成树对由满足下列条件的左树、右树组成：

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树对的左、右树中，或者均不出现。
c) 每一对 CC 和 VS 边可以同时出现在有效树对的左、右树中，或者同时不出现，或者成对出现在有效树对中，即 VS 边出现在左树中，CC 边出现右树中。
d) VC 和 CS 可以不出现在有效树对中，或者成对出现在有效树对中，并且 CS 边出现在左树中，VC 边出现在右树中。

当有效生成树对的左、右树完全一致时，生成树对退化成生成树。

根据以上条件，可知：在一个有效生成树对中，左生成树只含有 Y、Z、VS、CS、CC、和 NO 类型的边，右生成树中只含有 Y、Z、VS、VC、CC 和 NU 类型的边。因此在计算机实现由初始图得到初始左、右图时，在进行分析前，先删除左图中所有的 VC 边和 NU 边，删除右图中所有的 CS 边和 NO 边。图约化规则如下表 1 所示。

表 1 图约化规则[8]

Table. 1 Binary Operation For Graph Reduction

	选取(Include)元件		排除(Exclude)元件	
	左子图	右子图	左子图	右子图
VCVS	短接 VS	移除 VS 短接 VC	短接 VS	短接 VS 移除 VC
CCVS	短接 VS 移除 CC	移除 VS 短接 CC	短接 VS 短接 CC	短接 VS 短接 CC
VCCS	短接 CS	短接 VC	移除 CS	移除 VC
CCCS	短接 CS	短接 CC	移除 CS 短接 CC	短接 CC
Nullor	短接 NO	短接 NU	移除 NO	移除 NU
Y/Z	短接 Y/Z	短接 Y/Z	移除 Y/Z	移除 Y/Z

文献[8]给出的图约化算法如下（并按照深度优先的顺序构建图约化判定图[10]）：

Step 1. 根据有向图构建规则创建初始左、右图后，删除初始左图中所有的 VC 边，删除右图中所有的 CS 边。

Step 2. 按约定符号次序进行图约化操作，最先处理输入输出对应的符号（在 2.1.4 中我们将可以看到，这样处理便于求解传输函数），按照表 1 中的规则短接或删除图中的边。

Step 3. 检测是否满足终止条件。如果左、右图同时生成树，则将判定图的边指向终止节点 1，如果左图或者右图已不可能形成生成树，则指向终止节点 0，然后转向 Step 6，否则进入下一步。

Step 4. 根据生成项符号算法（2.1.4 将会介绍）计算当前判定图边的符号（+ 或 -）。

Step 5. 检测子图对是否已经存在于哈希表中，如果已经存在，证明可以共享（必

须满足子图对中的左（右）图与可共享节点的左（右）子图同构)，则将当前判定图的边指向该共享节点， 并终止对该子图对进行约化操作。

Step6. 如果仍有未处理的符号，则转向 Step 2，否则终止图约化算法。

2.1.4 GRASS 求传输函数

电路特性符号化表达式中的生成项由生成树中所有的元器件符号（所有的器件均是以导纳形式出现的，因此 Z 在表达式中以 Z^{-1} 的形式出现）和表示正负的符号乘积组成（只会是 + 或者 -)。

表正负的符号由受控源增益的符号和生成项的符号（+ 或者 -）共同决定。

四种受控源(VCVS、CCVS、VCCS、CCCS)在生成项中表现为带有符号的增益[8]，具体如下：

$$VCVS \leftrightarrow -E_{j,k}$$
$$CCVS \leftrightarrow -H_{j,k}$$
$$VCCS \leftrightarrow +G_{j,k}$$
$$CCCS \leftrightarrow +F_{j,k}$$

可以发现，当 VS 边出现时便会出现负号。

图的约化入射矩阵中行与电路结点一一对应（除去接地点)，其数目等于电路总节点数减一；约化矩阵的列表示生成树的边，由生成树的定义可知其边数等于电路总节点数减一。因此约化入射矩阵为方阵，行列式存在，且其行列式的值只能为 1 或者-1（见[26])。文献[8,10]中给出了生成项的符号算法，可以在有效树对选择过程中就直接计算行列式的值，而无需在选择完成后（即已形成生成树）再计算。

生成项符号确定算法如下：

Step 1. 初始化：sign = 1。

Step 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。

Step 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。

Step 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。

Step 5. 将 sign 关联到相应的 GRDD 边上。

文献[8]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

$$X(\sum_{i=1}^{m_1} s_i) + (\sum_{j=1}^{m_2} o_j) = 0 \tag{2-1}$$

$$X = -\frac{\sum_{j=1}^{m_2} o_j}{\sum_{i=1}^{m_1} s_i} \tag{2-2}$$

$$H(s) = \frac{1}{X} = -\frac{\sum_{i=1}^{m_1} s_i}{\sum_{j=1}^{m_2} o_j} \tag{2-3}$$

Si 表示选取 X 元件的生成项，Oj 表示排除了 X 元件的生成项，根据公式（2-3）可以得到传输函数的符号化表达式，并且传输函数的分子对应着所有包含 X 的项，传输函数分母对应着所有排除 X 的项（这里说的所有的项，是指所有通过约化规则得到的项），因此在进行图约化操作时，最先处理 X 器件，这样在图约化结束后，可以很方便地得到分析结果。

2.1.5 GRASS 分析电路举例

我们通过一个例子来说明 GRASS 分析电路的过程。待分析电路如图 7 所示，输入为电压源，输出为流经 R_2 的电流，因此 X 对应的器件类型为 CCVS。由电路基础理论可以快速得到该电路的频域传输函数 $H(s)=\frac{1}{R_1+R_2}$。

按照图转换规则生成初始有向图后，拆分得到左、右初始图，根据有向图约化规则及算法得到的分析结果如图 7 所示。图中的每一个节点与电路中的元件一一对应，并且含有左、右图对，每个节点是 BDD 结点，在文献[30]中称之为 tpddnode，这些 tpddnode 组成的分析结果（二叉决策图）称为 tpdd，本文继续沿用这两个名称。

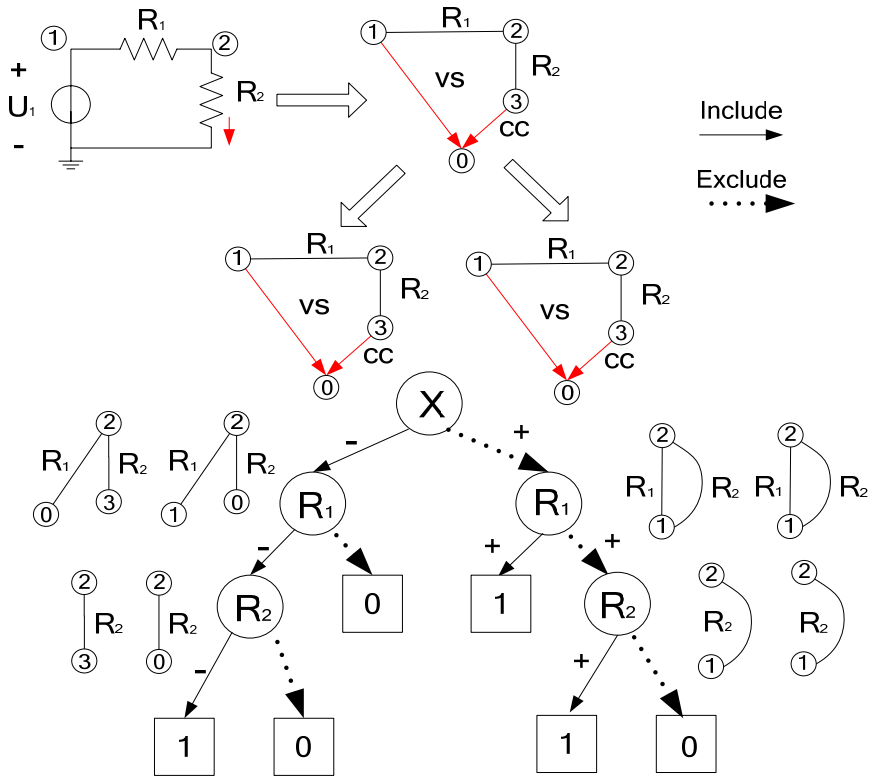


图 7 GRASS 分析电路举例

Fig. 7 A example of GRASS analysis circuit

tpdd 中由根节点出发至终止节点 1 的所有路径对应着有效生成项，共有三项，分别是

$t_1 = X*(-1*R_1^{-1})*(-1*R_2^{-1})*(-1) = -XR_1^{-1}R_2^{-1}$，对应的生成树对如图 8（a）所示。

$t_2 = R_1^{-1}; t_3 = R_2^{-1}$，$t_3$ 对应的生成树如图 8（b）所示,左、右树完全一样，此时有效生成树对退化成有效生成树。

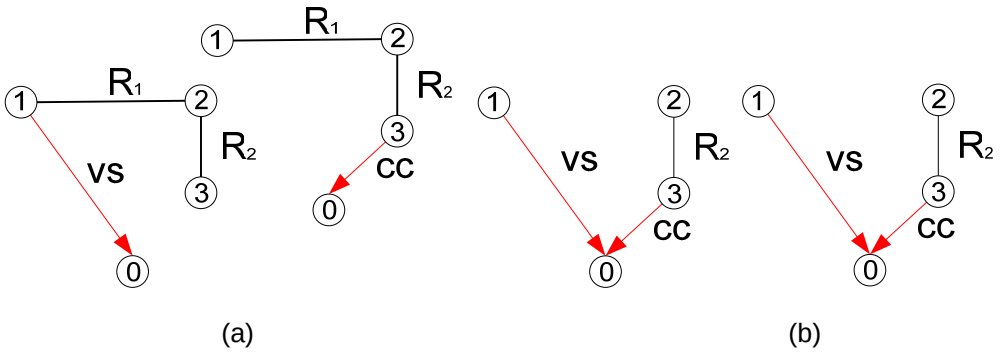


图 8 有效生成树对及对应项

(a) t_1 对应的有效树对；(b) t_3 对应的有效树

Fig. 8 Admissible tree pair and admissible tree

根据所有生成项之和为零，有 $-XR_1^{-1}R_2^{-1}+R_1^{-1}+R_2^{-1}=0$ 。

将 X 作为未知量解方程得 $X=\dfrac{R_1^{-1}+R_2^{-1}}{R_1^{-1}*R_2^{-1}}$，因此传输函数 $H(s)=\dfrac{1}{X}=\dfrac{R_1^{-1}*R_2^{-1}}{R_1^{-1}+R_2^{-1}}$，

整理后得 $H(s)=\dfrac{1}{R_1+R_2}$ 与前面依据电路理论知识的分析结果一致。

2.2 基于 GRASS 的层次化分析算法

由于符号化分析产生的项随着电路规模的增大成指数增长，这将极大限制符号化分析的应用，采用层次化分析可以很好得解决这一难题。层次化分析的思想是将大电路划分成若干小电路块后，通过对小电路的求解得到整个电路的分析结果。按照这一思想，层次化分析分为三部分：1）将电路划分成小块；2）对所有的小电路进行分析；3）根据小电路的解，按照一定规则求得整个电路的分析结果。

同时，由于在分析时，只需要用到网络的边界节点电压、电流信息，因此，只需要用约化节点导纳矩阵即可描述该网络[7]，对于一个有 n 个边界节点（除去地点）线性、时不变网络，它对应的约化导纳矩阵 Y 如下式（2-4），在本文中我们均以 Y 矩阵来简记约化节点导纳矩阵，并规定以电流流出边界节点为正方向。

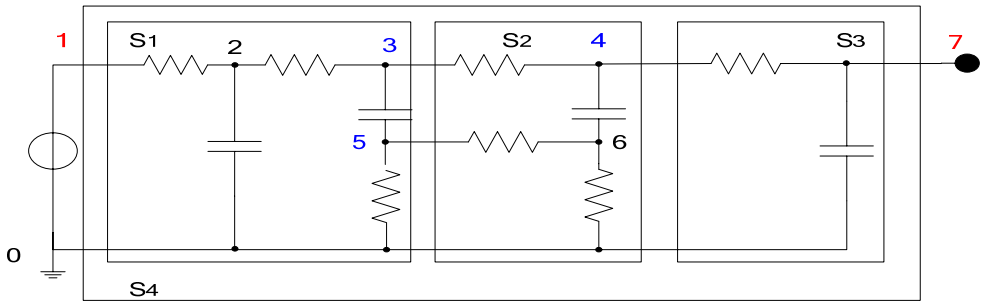
$$\begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_n \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \tag{2-4}$$

其中u_i是标号为 i 的边界节点电压（对地电压）。

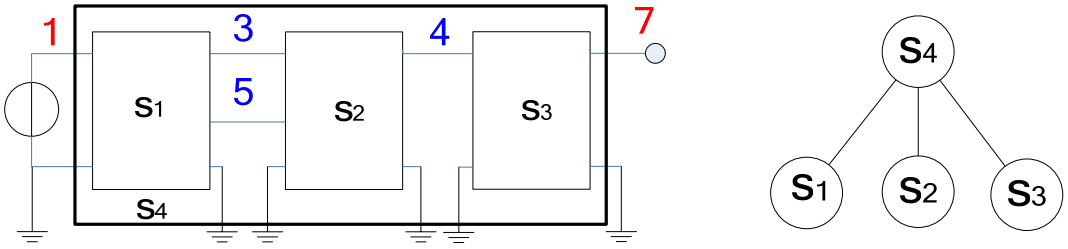
当我们对一个网络进行层次化分析的时候，所要做的就是消去子电路之间的连接节点，仅仅保留我们需要的上一层模块的端口信息（对于顶层电路来说，我们只需要保留输入输出端点即可），消去所有的内部节点后，自然可以获得顶层电路的 Y 矩阵。

2.2.1 电路划分

层次化分析的第一步是电路划分，如何对模块实现有效划分（例如在最短的时间内可将电路划分的块数最多等衡量标准），这是一个启发式（Heuristic）的过程，不在本文的讨论范围内。我们采用的是读入预定义的模块划分文件（模块间相互独立，不存在相互控制，耦合等情况），包含有模块的边界节点及组成模块的元器件。本节中，我们侧重介绍由电路划分引入的一些概念，如边界节点，内部节点，电路层次等。



（a）



（b）

图 9 划分电路的层次图

（a）电路划分举例 ； （b）对应电路层次图

Fig. 9 Hierarchy of partioned circuit

以图 9（a）中的电路为例，完整的电路我们记为 s_4（包含整个电路的输入和输出端），被分成了 3 个模块 s_1、 s_2 和 s_3，本文中，我们均使用 2 层电路层次，即划分后的电路只会出现底层叶子电路模块（如 s_1）和顶层电路模块（如 s_4）。

图 9（b）右图即是对应电路的层次图。在电路层次中，只有一个顶层电路模块，但是可以多个底层叶子模块。这些底层电路成为该顶层模块的子模块（如 s_1 是 s_4 的子模块），顶层模块是所有的底层叶子模块的父模块（如 s_4 是 s_1 的父模块），叶子电路模块处于层次图的底端，没有子模块。

2.2.2 底层叶子电路分析

由于图约化算法天然消去了内部节点信息，仅保留最后的输入输出间的传输关系，并且图约化算法的生成项是展平的，且不产生冗余项，因此，我们使用图约化算法来计算叶子电路的约化节点导纳矩阵，这样可以保证叶子电路的约化节点导纳矩阵中的每一个元素都是非冗余的，并且是展平的。求解方法如下：要求解(2-4)中的 Y 矩阵，每次仅保留第 k 个输入，并且值为 1，其余的边界节点均短接到地，求解 $i_1, i_2, \cdots i_n$，就可以方便求得 Y 中的第 k 列元素。从传输函数的角度看，$y_{1k}, y_{2k}, \cdots y_{nk}$ 是 n 个传输函数，如 $y_{pk} = i_p / u_k (p, k = 1, 2, \cdots n)$，可看作 CCVS。每个 y_{pk} 由 GRASS 仿真器求得。只要底层叶子电路规模不是太大，GRASS 能非常快的完成符号化导纳矩阵的计算（为了便于计算机实现，我们规定每个底层叶子电路模块要包含地）。

2.2.3 顶层电路分析

顶层电路为同时含有指定分析的输入、输出端的最大模块，它只有 2 个边界节点，即电路的输入端和输出端，其他均为内部节点，但是所谓的顶层电路的内部节

点，只能是子模块的边界节点，例如s_4的内部节点只有节点 3、4、5,节点 2 是叶子电路s_1的内部节点，但是对顶层模块s_4不可见，不是s_4的内部节点。

顶层电路分析即是合并所有叶子节点的分析结果的过程。我们以Y_R^i标记标号为 i 的电路模块的约化导纳矩阵，V_e^i（J_e^i）标记标号为 i 的电路模块的边界节点电压（电流）矩阵。对于有 m 个子模块的电路，分别将这些子模块命名为$S_1, S_2 \cdots S_m$，每个模块的电压-电流关系如下列式子：

$$Y_R^1 \times V_e^1 = J_e^1 \tag{2-5}$$

$$Y_R^2 \times V_e^2 = J_e^2 \tag{2-6}$$

$$\cdots\cdots$$

$$Y_R^m \times V_e^m = J_e^m \tag{2-7}$$

我们先创建一个临时的导纳矩阵，记为Y_{R_t}，并且

$$Y_{R_t} = \begin{bmatrix} Y_R^1 & 0 & \cdots & 0 \\ 0 & Y_R^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Y_R^m \end{bmatrix} \tag{2-8}$$

此时等式右边的向量 RHS 有

$$RHS = \begin{bmatrix} J_e^1 & 0 & J_e^2 & 0 & \cdots & J_e^m & 0 \end{bmatrix} \tag{2-9}$$

如果子模块的边界节点间有连接关系，则将Y_{R_t}中对应的列相加（如图 10 左图所示，如果选择保留节点 3 编号，那么将Y_{R_t}中节点 5 对应的列以及节点 6 对应的列统统加到节点 3 所对应的列，这是由于将S_1的节点 3 与S_2中的节点 5 接在一起，从电路理论来讲，节点 3 和节点 5 一定具有相同的电压），同时将节点 5 和节点 6 对应的行加到节点 3 对应的行（由于矩阵中的每一行均代表着通过该节点流出电路块的电流和，因此当节点 3，5，6 连接在一起时，需要将他们代表的行相加），当节点 3 成为顶层电路的内部节点时，即不是顶层电路的输入、输出端时，节点 3，5，6 的电流和应该为零。

仍以图 10 中的左图为例，我们以 I_{ij} 表示从节点 i 流向节点 j 的电流（例如 I_{36} 表示从节点 3 流向节点 6 的电流），节点 3、5、6 接在一起后，成为了顶层电路模块的内部节点，为了便于分析，我们增加了一个虚拟节点 v（图 10（a）等效后如图 10（b）所示），同时为了不影响对电流的分析结果，我们规定没有从该虚拟节点流出的电流，并且有 $I_3 = I_{35} + I_{36}$，$I_5 = I_{53} + I_{56}$，$I_6 = I_{63} + I_{65}$，对节点 v 使用 KCL 定理，由于节点 v 没有流出的电流，因此有 $I_3 + I_5 + I_6 = 0$。

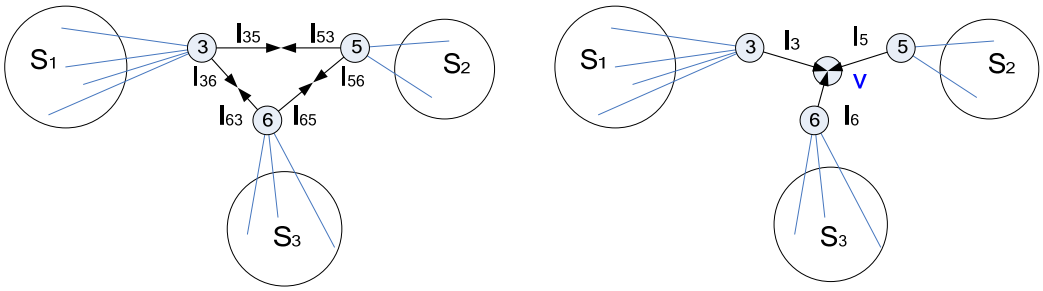


图 10 多子模块互联例子

（a）原始连接图；（b）加入虚拟节点 v 后的等效连接图

Fig. 10 An example of multi-connection between multi sub-block

根据相互连接的边界节点变成内部节点后，电流和为零这一条件，因此，只要使用这一条件，求解 Y_{R_t} 中变成内部节点对应的行组成的方程组，可以将内部电压变量用边界节点表示出来，回代入 Y_{R_t} 中内部节点电压代表的行，抽取对应的行和列，即可得到顶层电路的导纳矩阵。

得到分解图中的底层叶子电路模块的导纳矩阵后，根据各子模块的连接关系消去内部节点后可得到顶层电路块的导纳矩阵，并且根据二端口网络（two-port network）的 Y 矩阵与传输矩阵的关系，最终得到指定的传输函数的符号化表达式。如图 11 所示的二端口网络，对应的 Y 矩阵与传输矩阵的转换关系如表 2 所示。

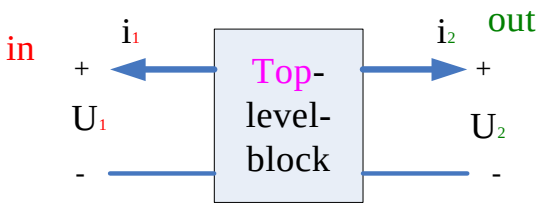


图 11 二端口网络

Fig. 11 Two-port network

二端口网络的导纳矩阵与传输矩阵（Transfer Matrix）的分别是式（2-10）和式（2-11），从（2-10）到（2-11）的推导见附录 1。

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{2-10}$$

$$\begin{bmatrix} U_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} U_2 \\ I_2 \end{bmatrix} \tag{2-11}$$

从 Y 矩阵推导得到的传输函数（Hs）如下表 2。

表 2 二端口网络 Y 矩阵与传输函数间的转换关系

Table 2 conversion table of Y matrix and transfer function

	Y 矩阵	传输函数（已转换为（1/t）
转换关系	$\begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix}$	$\begin{bmatrix} -\frac{y_{21}}{y_{22}} & -\frac{y_{11}y_{22}-y_{12}y_{21}}{y_{21}} \\ y_{21} & \frac{y_{21}}{y_{11}} \end{bmatrix}$

2.3 本章小结

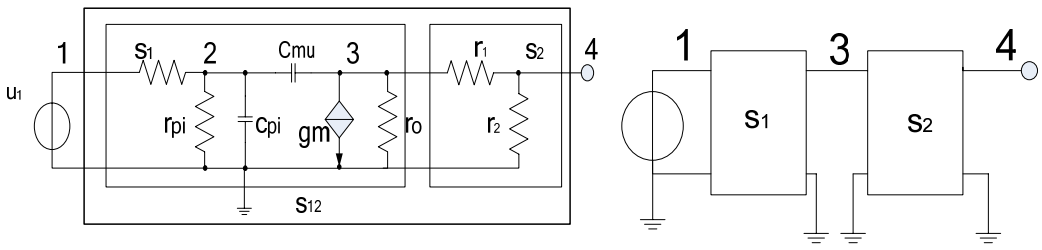
本章我们首先介绍了 GRASS 仿真器的适用条件以及 GRASSS 分析电路的算法，并举例说明了分析电路的过程，然后介绍了基于 GRASS 的层次化分析方法，不断消去内部节点变量后，对得到的顶层电路的导纳矩阵依照表 2 转换后，即可得到层次化分析的电路传输函数。

第三章 层次化分析方法实现

本章将讨论基于 GRASS 的层次化分析方法的实现细节，首先将举出一个例子来说明层次化分析的过程，然后给出层次化分析的基本流程以及使用高斯消去法求解顶层电路的传输函数和 GRASS 求解底层叶子电路 Y 矩阵（约化节点导纳矩阵）的实现方法，通过这些方法，第二章中提到的基于电路拓扑结构的层次化分析方法得以实现。

3.1 举例

在算法的早期验证阶段，我们正是通过例子才发现调整常规的层次化分析流程（即先分析底层叶子电路，再分析顶层电路）后，将可以减少冗余的分析，因此本节我们给出一个例子来说明这一过程及发现。



（a） （b)

图 12 层次化分析电路举例

(a) 电路原理图及划分图 (b) 模块连接框图

Fig. 12 An example of hierarchical analysis and it's block diagram

图 12（a）中所示电路中，设 S_1 的电压-电流关系为（设系数矩阵为 Y_1）

$$\begin{bmatrix} i_1 \\ i_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_3 \end{bmatrix} \tag{3-1}$$

设 S_2 的电压-电流关系为（设系数矩阵为 Y_2）

$$\begin{bmatrix} i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \times \begin{bmatrix} u_3 \\ u_4 \end{bmatrix} \tag{3-2}$$

将 S_1 和 S_2 按图 12(b)连接起来后，可得如下矩阵 O_3 为

$$\begin{bmatrix} i_1 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22}+b_{11} & b_{12} \\ 0 & b_{21} & b_{22} \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_3 \\ u_4 \end{bmatrix} \tag{3-3}$$

由于节点 3 变成了顶层模块的内部节点有 $i_3 = 0$，即

$$u_3 = -\frac{a_{21}}{a_{22}+b_{11}}u_1 - \frac{b_{12}}{a_{22}+b_{11}}u_4 \tag{3-4}$$

由此可消去(3-3)中的 u_3，得到一个 2*2 的约化节点导纳矩阵如下所示 Y_3

$$Y_3 = \begin{bmatrix} a_{11} - \frac{a_{12}a_{21}}{a_{22}+b_{11}} & -\frac{a_{12}b_{12}}{a_{22}+b_{11}} \\ -\frac{a_{21}b_{21}}{a_{22}+b_{11}} & -\frac{b_{12}b_{21}}{a_{22}+b_{11}} + b_{22} \end{bmatrix} \tag{3-5}$$

$$\begin{bmatrix} i_1 \\ i_4 \end{bmatrix} = Y_3 \times \begin{bmatrix} u_1 \\ u_4 \end{bmatrix} \tag{3-6}$$

如果我们要求 u_4/u_1，那么根据 Y 矩阵与传输函数矩阵的关系（附录中有详细的推导过程）有

$$H(s) = -\frac{y_{21}}{y_{22}} = -\frac{-\frac{a_{21}b_{21}}{a_{22}+b_{11}}}{-\frac{b_{12}b_{21}}{a_{22}+b_{11}} + b_{22}} \tag{3-7}$$

从上式（3-7）中可以看出，要求得传递函数，我们不需要使用 S_1 中的 a_{11}、a_{12}。因此我们的分析可分为两部分: 1）依据底层电路的连接关系，使用高斯消去法将顶层电路的传递函数表示为子电路输入输出导纳的代数表示；2）用 GRASS 分析各子电路导纳函数。

3.2 实现程序流图

算法的流程如下图 13 所示，符号化层次分析主要由 4 部分组成，如下图所示，读入排序且划分完毕的电路网表文件，构造符号化高斯分析模块，叶子电路块的 GRDD 符号分析和顶层电路的数值分析。

在 GRASS 中添加了层次化分析模块后，首先读入包含分块信息的网表文件，

根据分块信息以及底层模块的边界节点信息，建立包含所有叶子电路边界节点的原始矩阵（高斯消去前的矩阵）后， 根据网表中指定的输入、输出端及输出变量进行符号化高斯消去，得到传输函数关于叶子电路的导纳矩阵（Admittance matrix）中的元素的求解公式。

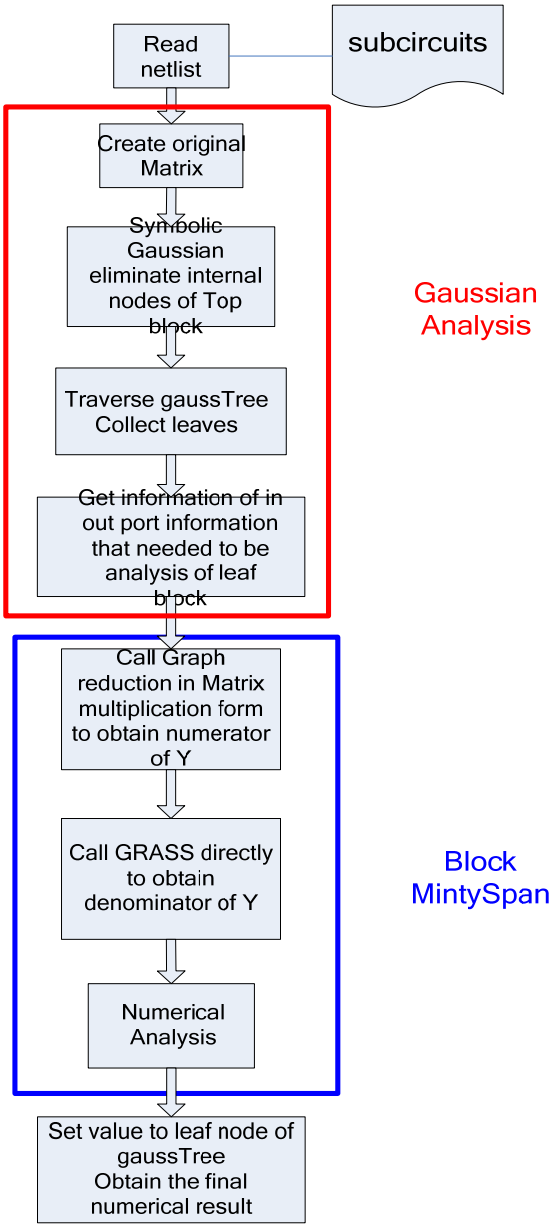


图 13 层次化分析程序流图

Fig.13 Flow graph of hierarchical analysis

3.3 高斯处理

高斯处理是为了从叶子电路块（leaf circuits）的导纳矩阵得到顶层电路的传输函数。

在进行高斯处理时，将会使用到电路层次的概念。首先需要生成所有的叶子电路的导纳矩阵，根据得到的导纳矩阵变换成需要求得的传输函数，然后对生成的符号化表达式（二叉树形存储）进行遍历，收集叶子节点（所有叶子节点将用 GRDD 进行符号化导纳计算），最后对收集到的叶子节点进行匹配查找（实现时是比较在哈希表中的索引号是否相等），即定位他们是属于哪一个叶子电路块的。。

3.3.1 高斯处理部分主要数据结构

为了保证层次化分析后，后续的分析（如敏感性分析）仍可以保持符号化，需要将高斯消去过程符号化表示。文献[9]使用了有向无环图（Directed Acyclic Graph）来表示高斯消去过程中得到的嵌套表达式，在本文我们称之为高斯树，其中的每一个节点称为高斯节点。高斯节点分为叶子节点和非叶子节点，其中的叶子节点对应的是子电路的导纳函数。

高斯消去时，每进行一次操作，都生成一个相应的节点，同时更新原始矩阵中的根节点。以图 12 中的电路为例，将矩阵 Y_1, Y_2 中元素依次标号为

$$Y_1 = \begin{bmatrix} g_1 & g_2 \\ g_3 & g_4 \end{bmatrix},\ Y_2 = \begin{bmatrix} g_5 & g_6 \\ g_7 & g_8 \end{bmatrix}$$

$g_i (i = 1, 2 \cdots 8)$ 是高斯树的叶子节点。将 Y_1 和 Y_2 按子电路连接关系组装成的导纳矩阵记为 O_3

$$O_3 = \begin{bmatrix} g_1 & g_2 & 0 \\ g_3 & g_9 & g_6 \\ 0 & g_7 & g_8 \end{bmatrix}$$

其中 $g_9 = g_4 + g_5$。

按照高斯消去的过程，我们首先选择 g_9 为主元，把 O_3 的第二行归一化，则 g_3 变为 $g_{10} = g_3 / g_9$，g_{10} 成为一个新的高斯节点，这时的高斯树如下图 14 所示。每次

进行一次算术运算，同时生成一个新的高斯节点来记录这一运算，这样在高斯消去结束后，我们由下往上即可得到最高层传递函数的符号化代数表示。接下来的任务是用 GRASS 求解所有需要的子电路导纳函数。

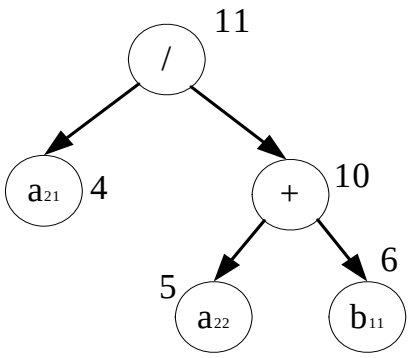


图 14 将 O_3 矩阵中 g_9 归一化的部分高斯树

Fig.14 GaussTree of unified g_9

3.3.2 高斯节点的哈希存储函数

为了避免重复冗余的操作，因此创建了一个哈希表 gcache 来管理 gaussnode 的存储，数据结构如下图 15 所示。

0	1	2	3	……	N-2	N-1
hash next =1 gaussnode*	hash next = 2 gaussnode*	hash next = 3 gaussnode*	hash next = 4 gaussnode*	……	hash next = N-1 gaussnode*	hash next = 0 gaussnode*

图 15 gcache 数据结构

Fig.15 data structure of gcache

首先创建一个一维数组，初始化时里面的每一个 hash 都设置成 0， 一维数组里的每一个单元通过 next 连接成一个循环数组（即长度为 N 的数组，最后一个单元的 next 值为 0，但是存贮数据后，next 的作用的是将除了哈希表入口外的所有共用该入口（称为“冲突”）的单元连接起来），哈希表创建完成后，在表头的前 2 个位置存储表示常量 0 和 1 的 2 个 gaussnode。

如果存储的是非叶子 gaussnode，则需要进行如下操作：

Step1：初步检测是否有必要进行运算（操作是否可以简化），例如当操作为相乘（*）

时，若其中的一个参数为常量 0，则直接返回常量 0。如果不能被简化，则进行下面的步骤。

Step 2：使用 Tripple（）函数寻找哈希表的入口，Tripple（）使用当前 2 个操作数在哈希表中的索引和操作符确定（如果是乘 * 或者加 + 操作，由于它们是可互换的操作，因此规定索引号小的作为 param1(参数 1)），具体如下：

```
int Tripple(int operator, int parm1, int parm2)
{
    return (operator + (parm1 + parm2) * (parm1 + parm2 + 1) / (2 + parm1)) *
        ( operator + (parm1 + parm2) * (parm1 + parm2 + 1) / (2 + parm1) + 1) /
        ( 2 + operator) ;
 }
```

Step 3：如果第一步产生了冲突，即根据第一步计算的入口处非空，则检测 2 个 gaussnode 是否相同，比较函数的定义如下：如果不相同则沿着当前单元中的 next 指向的单元搜索。

```
int gaussnode::SameNode(int parm1_id, int parm2_id, int operator)
{
    if( operator are same && parm1 are same && parm2 are same)
        return 1;
    else
        return 0;
  }
```

Step 4：如果搜索失败（即不存在于哈希表中）或者不存在冲突，则将数据存贮在哈希表中的第一个空位处，同时移动空位标志到插入位置的 next，插入位置的 next 改为指向入口处的 hash 指向的单元，入口处的 hash 指向插入单元。

如果存储的是叶子节点，由于叶子节点没有左右子节点，因此直接将叶子节点按生成的先后顺序直接插入哈希表中的第一个空位处。

综上所述，gaussnode 哈希策略如下：

哈希表中的每一个单元中有 3 个数据，分别是 hash，next 和存储的数据指针。next 将所有共用同一个哈希表入口（除了入口单元）的单元连接起来，链表的表头为哈希表的入口单元，hash 为进入哈希表入口后提供跳转方向，如果是非哈希表入口，则沿着 next 查找。从链表的角度看，每次在入口处的后面插入存储单元，并且修改入口处的 hash，使其指向当前新插入的数据单元。

存储叶子节点是顺序产生，顺序存贮的，因此在后面的对应的叶子电路块中记录了当前模块的 gaussnode 的最大和最小的索引号，以便快速查找。

当高斯处理部分符号化分析结束时，可以得到所有的叶子电路的导纳矩阵中需要被分析的元素。

3.4 求解叶子电路约化导纳矩阵

例如我们要求 Y(i, j)时（Y 中第 i 行第 j 列的元素），对应的电路图及初始左、右图如图 16(a)所示，在第 j 个边界节点处施加一个单位电压源，求解第 i 个边界节点的到地电流，虚拟的 X 器件为 CCVS，根据 CCVS 器件的处理规则，分别得到了 Y(i, j)的分子部分子图如图 16(b)所示，分母部分子图如图 16(c)所示。

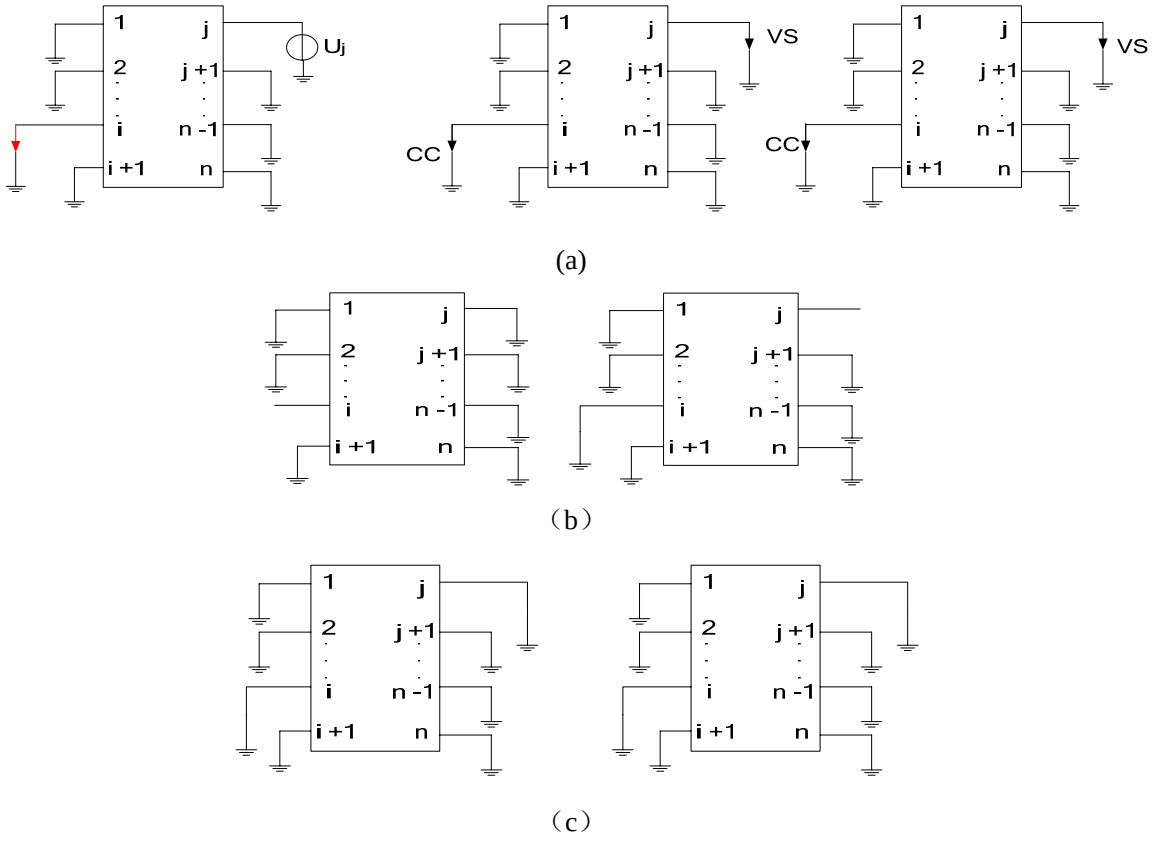
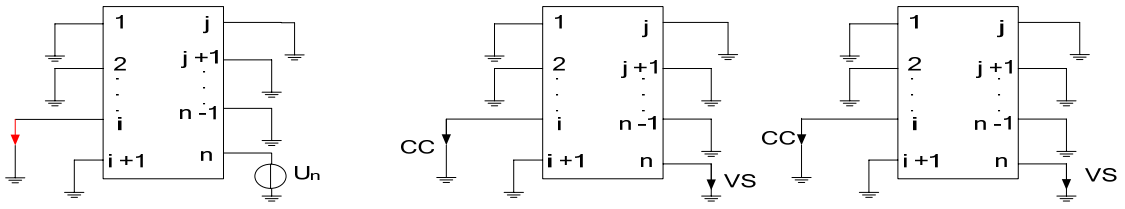


图 16 求解 Y(i, j) 电路图及选取 X 器件，排除器件后的子图对

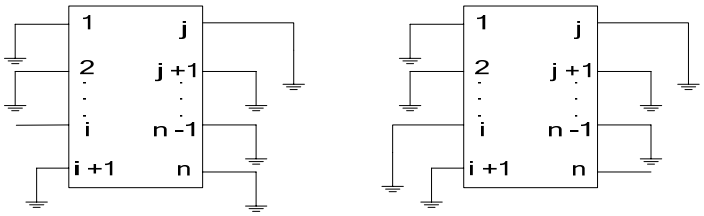
(a) Y(i,j) 电路图及初始左、右图；（b）选取 X 元件后的左、右子图；（c）排除 X 元件后的左、右子图

Fig.16 Starting graph pair of Y(i,j) and include X symbol exclude X symbol

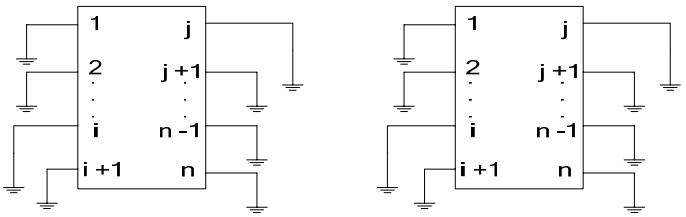
为了说明我们的算法，下面将给出求解与 Y(i, j)同一行的元素 Y(i, n)及同一列元素 Y(j, j) 的过程，其中 Y(i, n)对应的测试图如图 17(a)所示，根据 CCVS 处理规则包含 X 器件后的分子部分如图 17(b)所示，排除 X 器件后的图如图 17（c）所示。



（a）



（b）



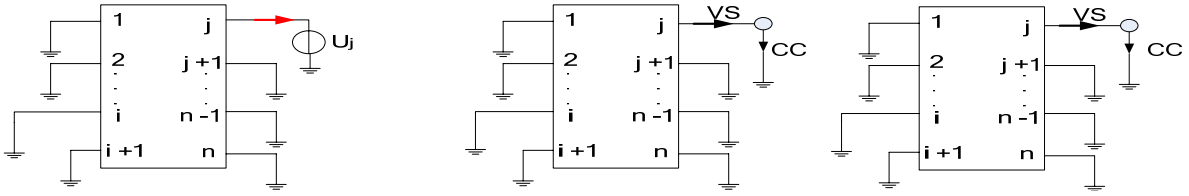
（c）

图 17 求解 Y(i, n) 电路图及选取 X 器件，排除器件后的子图对

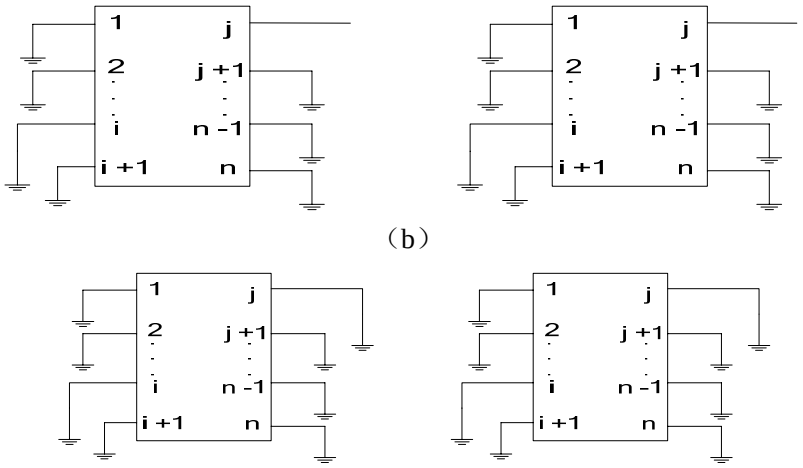
(a) Y(i, n) 电路图及初始左、右图；（b）选取 X 元件后的左、右子图；（c）排除 X 元件后的左、右子图

Fig.17 Starting graph pair of Y(i, n) and include X symbol exclude X symbol

求解 Y(j, j)对应的电路图如图 18(a)所示，按照 CCVS 约化规则包含 X 器件后的分子部分如图 18(b)所示，排除 X 器件后的图如图 18(c）所示。



（a）



（b）

（c）

图 18 求解 Y(j, j) 电路图及选取 X 器件，排除器件后的子图对

(a) Y(j, j) 电路图及初始左、右图；（b）选取 X 元件后的左、右子图；（c）排除 X 元件后的左、右子图

Fig.18 Starting graph pair of Y(j, j) and include X symbol exclude X symbol

Y 矩阵中的同一行元素，例如 Y(i, j)和 Y（i, n）, 从图 16(b)和图 17(b)的左图可以看出，选取 X 器件后，他们有相同的左子图（除边界节点 i 开路，其他的边界节点均短接到地）。Y 矩阵中的同一列元素，例如 Y（i, j）和 Y(j, j), 从图 17(b)和图 18(b)的右图可以看出，选取 X 器件后，他们有相同的右子图（除边界节点 j 开路，其他的边界节点均短接到地）。Y 矩阵中的所有元素，排除 X 器件后，他们的左、右子图均相同（左、右子图均是所有的边界节点短接到地），这意味着矩阵中的所有元素分母相同，这是拓扑层次化分析的优点。

如果我们在求解叶子电路的约化导纳矩阵时，只是简单调用 GRASS, 这样在计算该矩阵时就会导致重复展开部分左、右子图（例如当我们求解 Y（i, n）时 Y（i, j）的左子图就会重复约化操作），因此我们采用了一种新的数据结构来避免这些冗余操作，并且分别计算导纳矩阵的分子、分母部分，其中分子部分的左、右子图写成矩阵相乘形式，分母部分则直接调用 GRASS 计算。由于在求解导纳矩阵时用到了树对（Tree Pair），因此我们将求解导纳矩阵时的一种新的数据结构命名为 Ytpdd，其中的每一个单元命名为 Ytpddcell，后文将详细描述他们的数据结构。

3.4.1 分子部分算法实现

分子部分的左（L）图和右（R）图可写成矩阵相乘形式，图 12(a)中 S_2（边界节点重新依次编号为 1、2）的约化节点导纳矩阵分子部分左、右图可以写成如下图 19 的矩阵相乘形式（由它可表示四对图）。例如 S_2 中的 $Y_2(2,1)$的左图为 L[2](左图矩阵中的第 2 个图，在后文，我们以 L[i]表示左图矩阵中的第 i 个图)，右图为 R[1](右图矩阵中的第一个图，同样，我们以 R[j]表示右图矩阵中的第 j 个图)。采用这种形式，我们可以去除重复的图约化操作，为了便于程序实现，我们创建 YtpddCell 的数据结构来封装矩阵形式的图约化操作，其中除了左、右图矩阵外，还包含有一个二维矩阵 Ymat 用以存放约化处理当前元件过程中产生的需要继续处理的 tpddnode，并且规定终止结点 1、0 和指向哈希表中的结点（这 3 种节点我们称为无效结点，与之相对的是有效结点）不放入其中以方便判断是否终止图约化，为了进一步提升算法的执行效率，我们增加了两个一维数组 rowvalid（colvalid）来记录 Ymat 中的对应的行（列）中有效 tpddnode 的数目，我们以 rowvalid[i]表示 Ymat 中第 i 行中有效结点数，以 colvaild[j]表示 Ymat 中第 j 列中有效结点数，如果 rowvalid[i]为零，则无需对 L[i]进行图约化操作,同样，colvalid[j]如果为零则停止对 R[j]进行图约化。

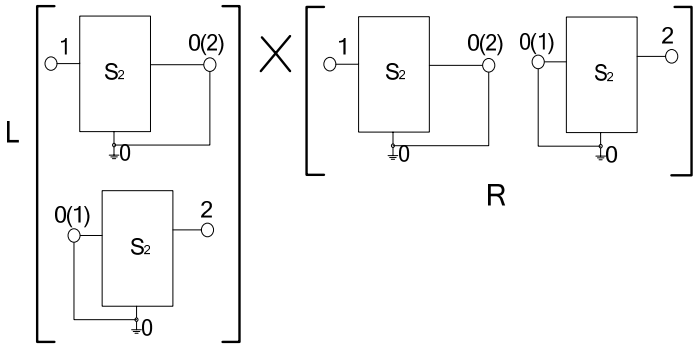


图 19 S_2 导纳矩阵分子部分初始图矩阵

Fig.19 starting graph matrix of numerator (S_2)

求解电路块约化节点导纳矩阵图约化算法归纳如下：

第 1 步初始化：根据读入的电路网表和构图规则创建内部器件的初始左图和初始右图。

第 2 步 分子矩阵初始化：分别依次使左图矩阵 L[i]和右图矩阵 R[i]中的第 i 个边界节点开路，其余边界节点短接到地。

第 3 步 根据当前符号类型确定所需的左、右图约化操作。同时完成图矩阵中的所有左、右图约化操作。如果 L[i]的图约化导致指向终止结点 0，那么 Ymat 的第 i 行所有元素均指向 0，如果 L[i]和右图矩阵中的 R[j]均导致指向终止结点 1，那么 Ymat[i][j]指向 1，否则存储新生成的节点，如果该节点已经存在于哈希表中，则将 Ymat[i][j]设置为空。检测 Ymat 矩阵中的非空元素数目（即仍需约化的图对数目），如果数目为 0，则终止当前元件的图矩阵约化。

第 4 步：计算当前节点的符号

第 5 步：若还有元件未处理完，则返回第 3 步，如果已处理完全，则终止。

3.4.2 求解分子部分举例

本节主要以图为主，介绍矩阵形式调用 GRASS 的图约化展开求解分子部分的过程。对 Ytpddcell 采取松散的封装后，可以方便的判断何时终止图展开(spanning)。将 Ytpddcell 按操作类型（short 或者 open）连接起来即可得到 Ytpdd，下图 21 是图 20(a)所示的电路对应的 Ytpdd 的展开举例。

图 20 说明了处理包含 MOS（或者三极管等非线性器件）的过程，划分电路后，将对应的模块转换成线性电路，然后根据 GRASS 构图规则生成初始有向图，如图 20（b）所示。

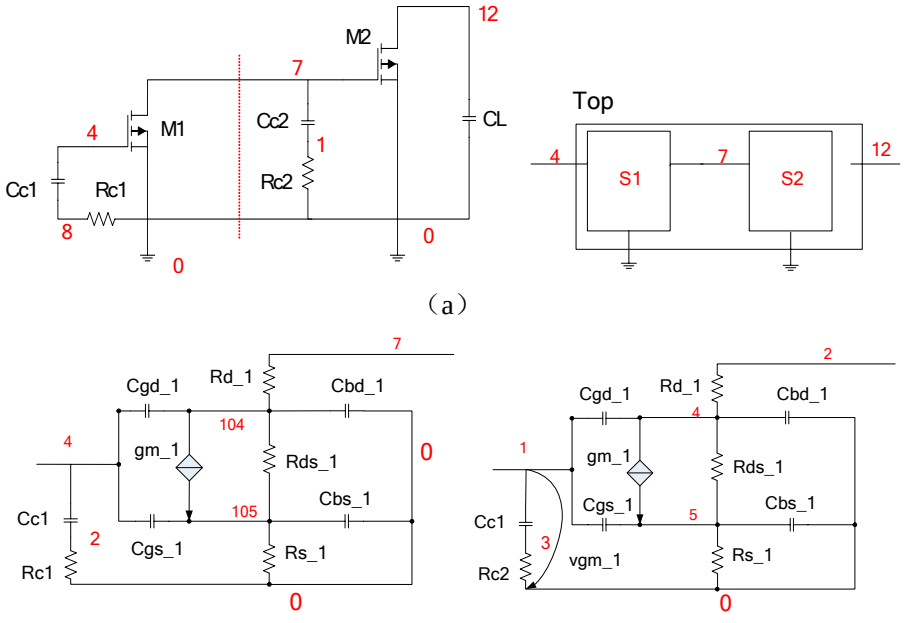
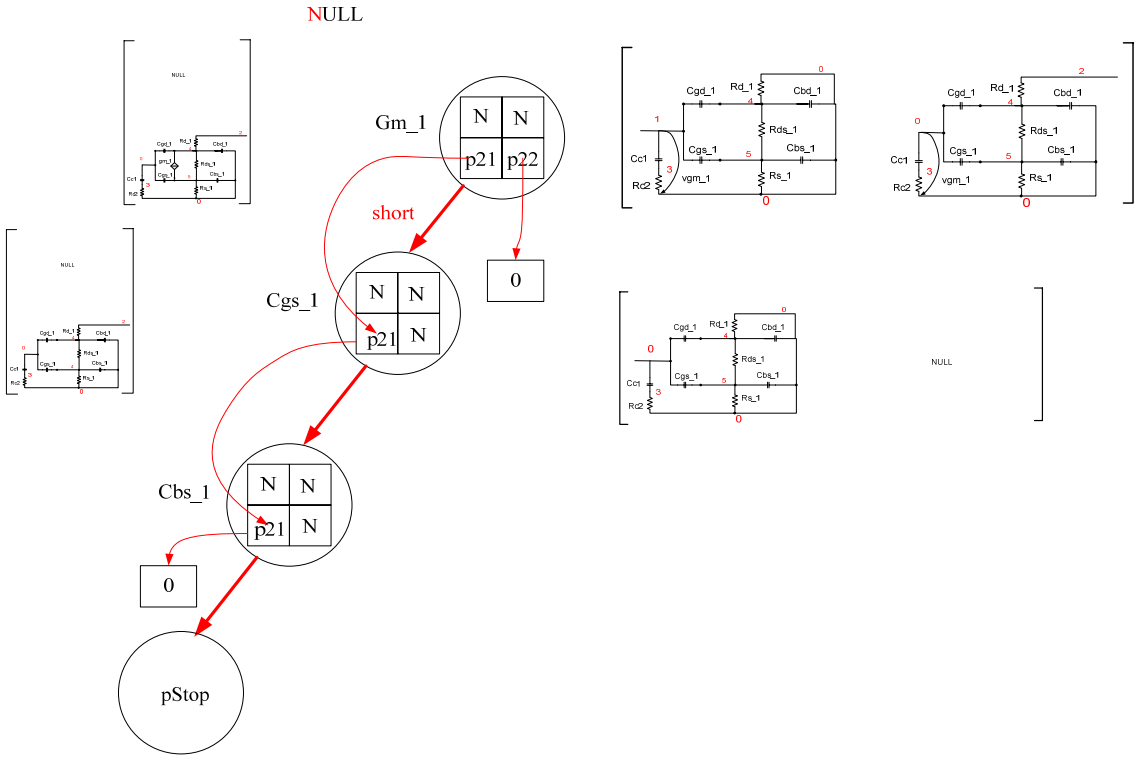


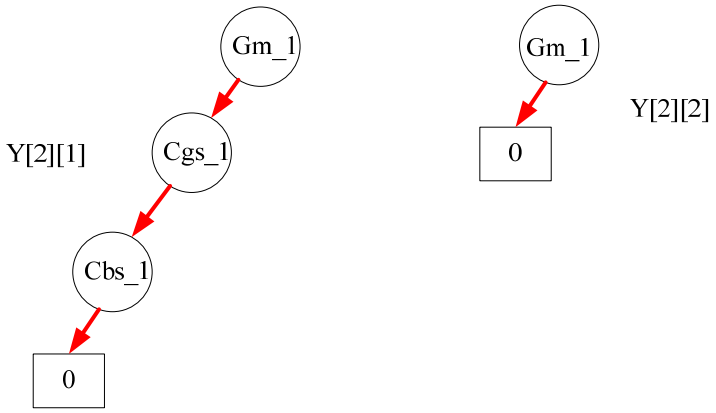
图 20 原始电路图及等效小信号模型构建的有向图

（a）划分前的完整电路及模块连接框图；　（b）S_1小信号模型电路及构建的有向图；

Fig.20 Small signal of the circuit and directed graph of the circuit



(a)



(b)

图 21 S_1模块的 Ytpdd 展开举例

（a）Ytpdd 前三次展开；　（b）前三次展开整理结果；

Fig.21 Ytpdd spanning examplex of S_1

图中的每个圆圈表示一个 ytpddcell，中间表示的是一个二维数组，维数由其边界节点数目决定，每次只对 cell 中的左图、右图中的一个元件进行约化操作，并连接二维数组中对应位置的 tpddnode，整理前三次的展开，对应的结果如图 21(b)所示。需要注意的是图 21（b）中的每一个节点都是 BDD 节点，而图 21（a）却不是。下一节，我们将比较两者间的异同。

3.4.3 Ytpdd 与 tpdd 的比较

从图 21 可以看出，Ytpdd 与 tpdd 的图展开十分相似，Ytpdd 与 tpdd 的相同点如下：

1) 同样有着 2 种操作，选择电路元件和不选择该器件，均是使用了 GRASS 的图约化规则。
2) 有 order（order 是指处理元件的次序），并且在同一个电路块中，order 与 tpdd（分母分析的模块）必须保持一致，以实现 BDD 的共享（使用同一张哈希表，同一哈希机制将可以实现自动共享）

Ytpdd 与 tpdd 的相似点：

1) Ytpdd 的图约化操作是针对左、右图矩阵进行的，而 tpdd 是针对左、右图进行的。即 Ytpdd 是完成当前 cell 中的左图矩阵图约化后才对右图矩阵进行操作，当前 cell 处理结束才会处理下一个元件，tpdd 是处理完每一个 tpddnode 后会继续一下个元件，而每一个 tpdddnode 只有一个左图和一个右图。这也使得导纳矩阵的分子部分是同时得到的。
2) Ytpdd 每次生成一个存有 tpddnode 的二维数组，tpdd 每次只生成一个 tpddnode。

Ytpdd 与 tpdd 的不同点

1) Ytpdd 只有一种终止结点 stop（当 cell 中的有效 tpddnode 数目为 0 时才会指向该终止结点。
2) Ytpdd 的结点单元 cell 本身不进行共享检测，但是 cell 内部的 tpddnode 却要检测，以避免重复冗余的展开操作，对 cell 不进行共享检测，是因为一个 cell 内部本身包含有 2 个图矩阵，如果可以共享，则需要对应的 cell 中左图矩阵、右图矩阵分别相同，这种概率比 tpddnode 低很多，当内部的 tpddnode 都可以共享时，我们可以通过减少 cell 的有效 tpddnode 结点计

数来控制，如果为 0，则证明内部所有的都可以共享或者 tpddnode 已经指向了终止 1 结点或终止 0 结点，无需再对当前 cell 进行图展开操作了，因此可以不用对 cell 进行共享检测。

3.4.4 分母部分算法实现

求解导纳矩阵的分母部分十分简单，只需要将初始左、右图中的所有边界节点短接到地后，直接调用 GRASS 分析即可（分析方法见 2.1.3）。

3.4.5 求解矩阵的符号规则

依然以有 N 个边界节点的网络为例，将它们依次标记为$1,2,\cdots N$，其他内部节点和内部由于构图规则增加的节点（由于有电流控制边而导致的串联边）则从$N+2$开始连续标记。求y_{pk}时，在第 k 个边界节点施加单位电压源，测量边界节点 p 的短路电流（除 k 以外所有的边界节点均短接到地），我们以 signL(sginR)标记左(右)子图操作产生的符号，此时对应的有 2 种情况，即：

1) $p=k$

此时 X 对应的 CCVS 边的连接情况如图 22（a）所示，网络外部增加节点标号为$N+1$，根据 CCVS 中选取元件的规则，左图进行的操作是需要短接 VS 边，根据生成项符号算法，有向图中节点编号比$N+1$小的有 2 个（分别是 0，k），算法的第二步执行完成时，signL 仍为 1。算法的第四步，短接的边是 VS，并且是以有效树对的形式出现的，此时$signL=-1$。同理，进行右图操作时，需要短接的边是 CC，第二步执行完成时，signR 同样为 1，不同的是由于选取的 CC 边与规定的有向边方向相反（CC 的节点 1 是$N+1$，大于节点 2 的编号），因此$signR=-1$。左、右图生成项算法执行完成后，最终的生成项符号$sign=signL*signR=(-1)*(-1)=1$。

2) $p\neq k$

此时对应的 CCVS 连接情况如图 22（b）所示，网络外部增加节点标号为 k，设

$p<k$，依据 CCVS 选取规则，符号确定算法第二步执行完的 signL 为-1（比节点 p 小的节点只有地点 0 一个），并且由于 VS 边是反向的，此时由要变一次符号，根据第四步，VS 以树对形式出现，符号再次变化，最终 $signL=-1$。右图稍有不同，按照算法，得到 $signR=(-1)^2*(-1)=-1$。因此最终的生成项符号 $sign=signL*signR=(-1)*(-1)=1$。

（a）　　（b）

图 22 VS 与 CC 连接情况

（a）$p=k$ 时 VS 与 CC 连接情况；　（b）$p\neq k$ VS 与 CC 连接情况

Fig.22 Different connection of edge between VS and CC

根据 CCVS 的排除规则，左右图均是短接 VS 边和 CC 边，此时导致 $signL$ 和 $signR$ 一定相等，因此导纳矩阵求解时排除 X 元件导致的符号为正。根据所有生成项和为零的特点，可知矩阵的分子、分母分别计算后，可以方便地合并得到最终的分析结果

$$y_{pk}=-\frac{VS_{pk}}{VO} \tag{3-8}$$

其中 VS_{pk} 表示约化导纳矩阵分子、分母分别求解后分子矩阵中对应的第 p 行第 k 列的元素，VO 则表示按照 3.4.2 中的算法得到的分母的分析结果。

3.5 分析结果求值

要对最终的分析结果进行数值计算，即是对顶层电路的分析量（本文中用到的

是求传输函数）所对应的高斯树进行数值分析。从高斯树的数据结构可知，要求得最终的数字结果，必须先完成高斯树叶子节点的数值化，这是通过先调用 GRASS 对底层叶子节点的导纳矩阵中的元素进行求值后再赋值给对应的高斯叶子节点完成的。

3.5.1GRASS 求值

文献[10]给出了如何对 GRASS 得到的图判定图进行求值。完成了图约化判定图的构建后，如果电路的拓扑结构不再发生变化（不增加元器件，却允许删除，因为我们只要将欲删除元件的值设为零即可使得 GRASS 符号化分析中所有包含该元件的项在数值分析时退化成零从而不影响数值分析的正确性，这也是符号化分析的一大优势，即删除元器件可以不用重新分析电路），只需要对图约化判定图进行中根遍历（即遍历顺序依次是根节点的左子树，根节点，根节点的右子树）就可以得到电路的数值频率响应，遍历效率与图中的节点个数成线性关系。

递归求图约化判定子图（Graph Reduction Decision Diagram，GRDD）的数值响应公式如下：

$$eval(v) = eval(v->left)*V(symbol)*signS(v) + eval(v->right)*signO(v)$$

其中的 v 代表 GRDD 中的节点，eval(v)代表以节点 v 作为根节点的判定子图对应的数值响应，v->left 代表节点 v 选取元件指向的节点，v->right 代表节点 v 排除元件指向的节点。signS 和 signO 分别代表选取元件和排除元件对应的分支符号，V（symbol）表示 GRDD 节点 v 在某个频率点下所关联符号的数值，对于不同类型的器件，其对应的数值如下所示：

$V(G) \Rightarrow G$

$V(R) \Rightarrow R^{-1}$

$V(L) \Rightarrow LS^{-1} \Rightarrow -j(2\pi * freq * L)^{-1}$

$V(C) \Rightarrow CS \Rightarrow j(2\pi * freq * C)$

$V(Source) \Rightarrow Gain$

通过上述公式，我们可以轻松获得 GRDD 的数值结果。

3.5.2 GAUSS 分析求值

对叶子电路导纳矩阵中的元素数值求值完成后，将高斯树中的叶子节点对应赋

值后，便可以按照后根遍历（遍历的次序依次是：根节点的左子树，根节点的右子树，根节点）的次序求得最终的电路特性数值结果。

递归求高斯树节点的数值公式如下：

$$eval(v) = eval(v->left) \quad \text{operator} \quad eval(v->right)$$

其中的 v 表示高斯树中的节点，eval(v)代表以节点 v 作为根节点的子树对应的数值，v->left 代表节点 v 代表的算术运算的参数 1，v->right 代表节点 v 代表的算术运算的参数 2，。operator 只能是加、减、乘、除四种算术运算（+、-、*、/）中的一种。

3.6 本章小结

本章首先通过一个例子来说明为什么不按常规的层次化分析顺序分析电路，而是采用先得到顶层电路传输函数关于底层叶子电路约化导纳矩阵中元素的求解公式，这样做可以提升算法效率，然后给出了符号化高斯消去的实现以及快速符号化分别求解叶子电路导纳矩阵分子、分母的算法实现，同时给出了一个例子来说明这一过程，并给出了如何合并分子、分母的分析结果，最后给出如何对符号化结果进行数值计算。

第四章 实验与结果分析

本文采用 C++编程语言实现算法，能够处理划分后具有任意数目边界节点的模块。读入包含划分模块信息的文件后，建立指定的传输函数关于底层叶子电路导纳矩阵中元素求解公式，调用 GRASS 分析底层叶子电路导纳矩阵中需要分析的元素后，代入传输函数的求解公式中，即可完成对电路的传输函数的求值，完成求值后，我们与 HSPICE 的仿真结果对比，如果 2 种不同的仿真方法得到的数值结果一致，我们则认为本文中的层次化分析算法是正确的。

我们选取了不同规模的基准电路验证仿真器的正确性和效率，测试电路组如下：

电路一：ua741 运算放大器（电路如图 23 所示）

电路二：ua725 运算放大器（电路如图 24 所示）

电路三：9_bw3 电路（11 个 MOS 管组成的电路，电路如图 26 所示）

电路四：Mos22 放大器电路（22 个 MOS 管组成的电路，电路如图 27 所示）

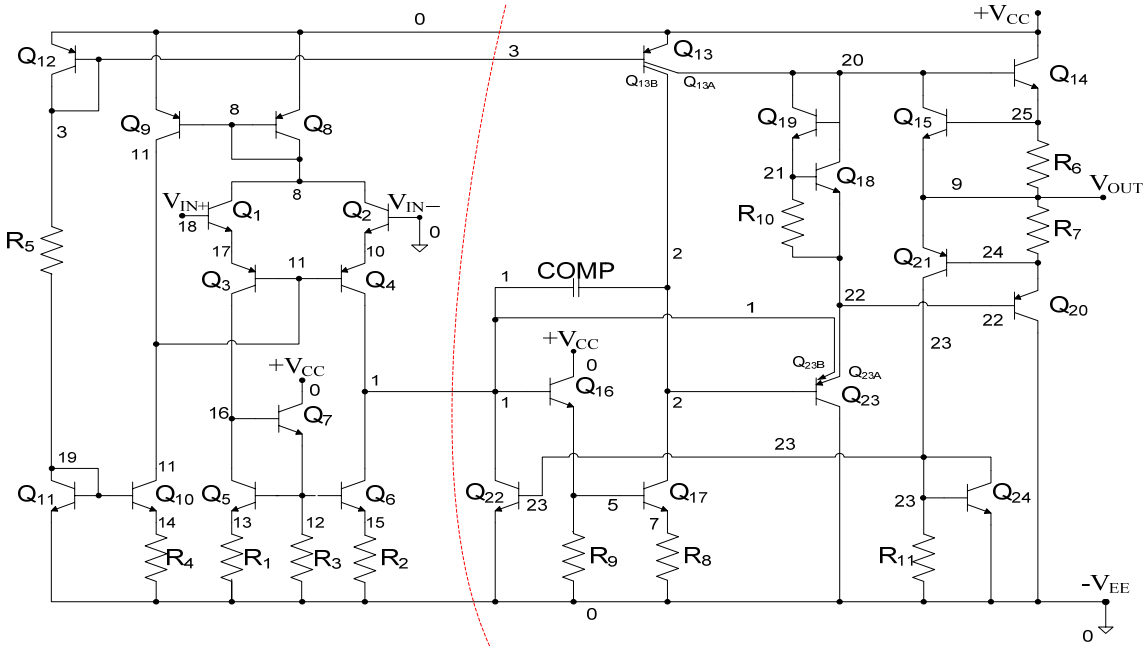
电路五：50t_opamp 电路（44 个 MOS 管组成的电路，电路如图 28 所示）

在进行测试时，分层和未采用分层分析的电路均经过相同的启发式算法。测试平台为测试平台为 Cygwin 下的 AMD Athlon 64*2 4400+ 双核 2G 内存,主频为 2.3GHZ 的处理器。

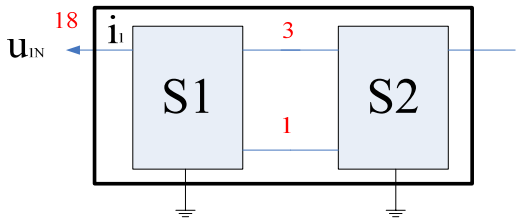
实验证明，本文的层次化分析算法是正确的，并且可以高效快速处理包含 44 个 MOS 管的电路（等效小信号模型有 117 个电路节点，358 个元件）。

4.1 算法正确性验证及效率测试

本节我们将依次测试前面 5 个电路,并将 HSPICE 得到的数值分析结果与我们的基于 GRASS 调用的层次化分析算法的数值计算结果（HSPICE 和层次化分析抽取的频率点不同），我们将两者的数据读入后使用 MATLAB 绘出幅度频率和相位频率响应曲线，以分析两种的曲线是否能够重合，如果可以重合，则表明我们的层次化符号分析算法是正确的。



（a）

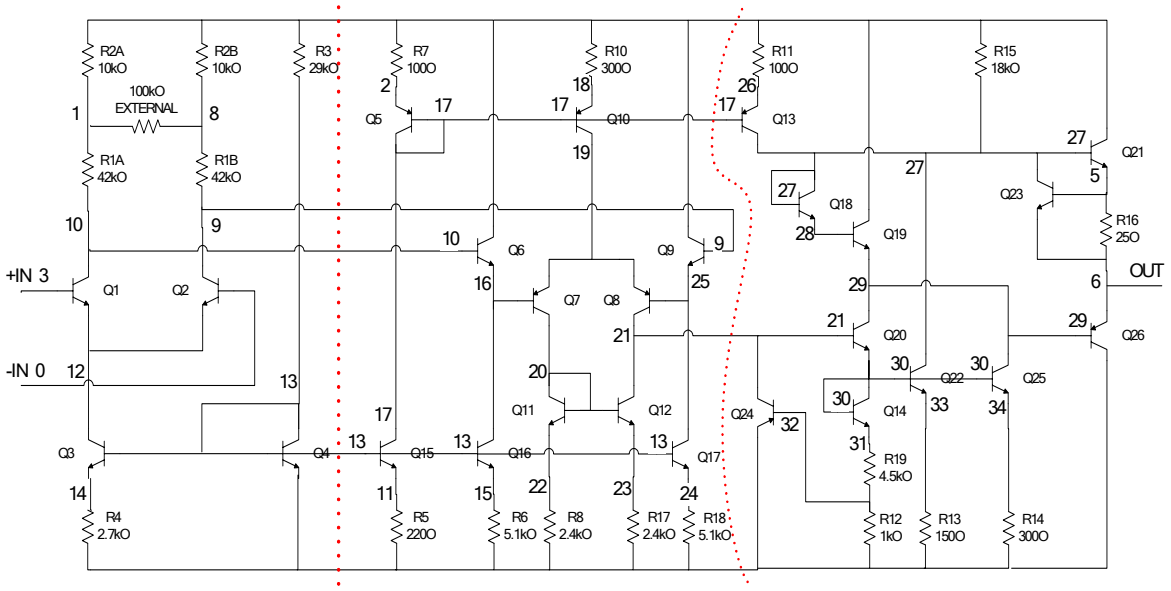


（b）

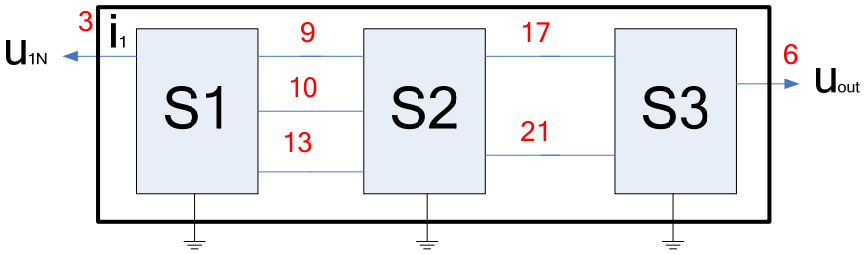
图 23 ua741 电路图及对应模块连接图

（a）电路图； （b）划分对应的模块连接图

Fig. 23 Schematics of ua741 and subblock diagram



（a）



（b）

图 24 ua725 电路图及对应模块连接图

（a）电路图；　（b）划分对应的模块连接图

Fig. 24 Schematics of ua725 and subblock diagram

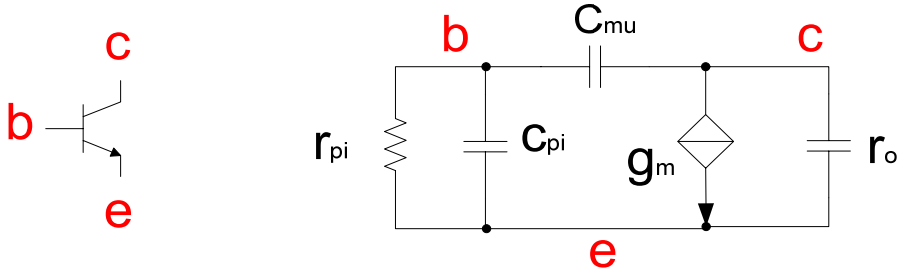
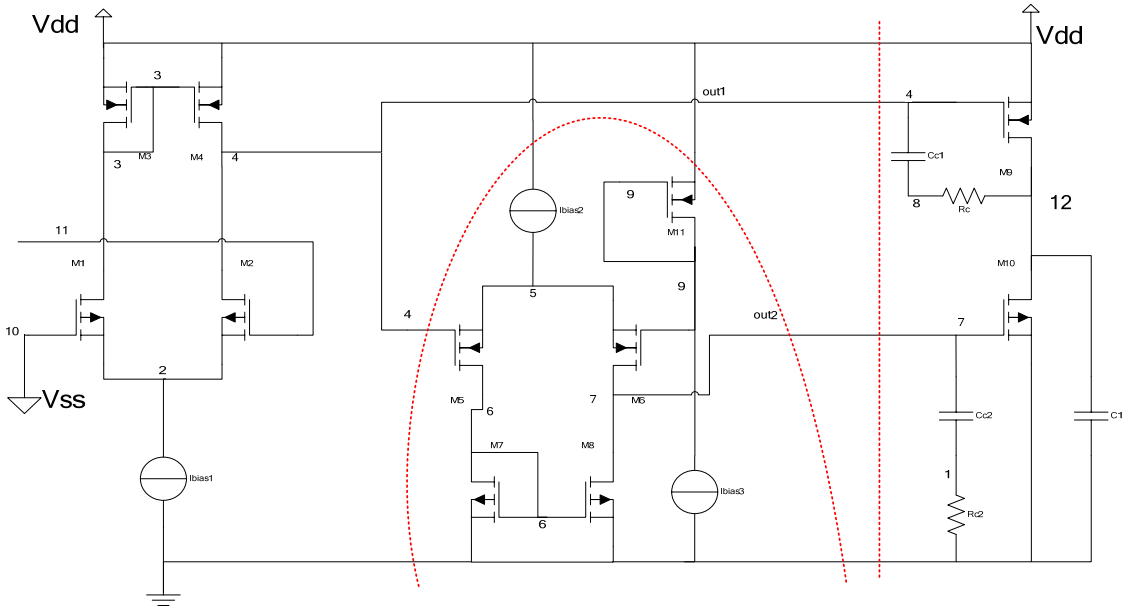
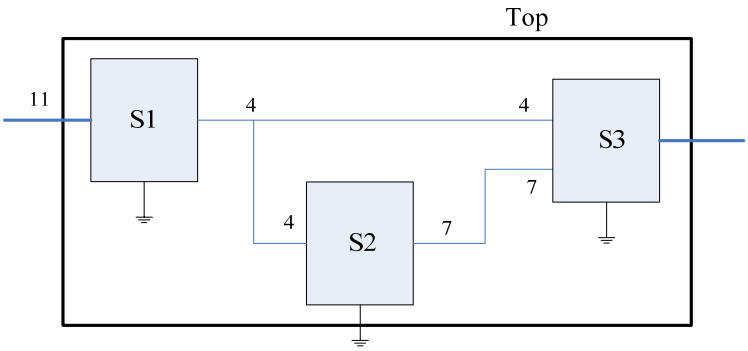


图 25 三极管对应的小信号模型

Fig. 25 small signal model of biopolar



（a） 9_bw3 电路图



(b)

图 26 9_bw3 电路图及对应模块连接图

（a）电路图； （b）划分对应的模块连接图

Fig. 26 Schematics of 9_bw3 and subblock diagram

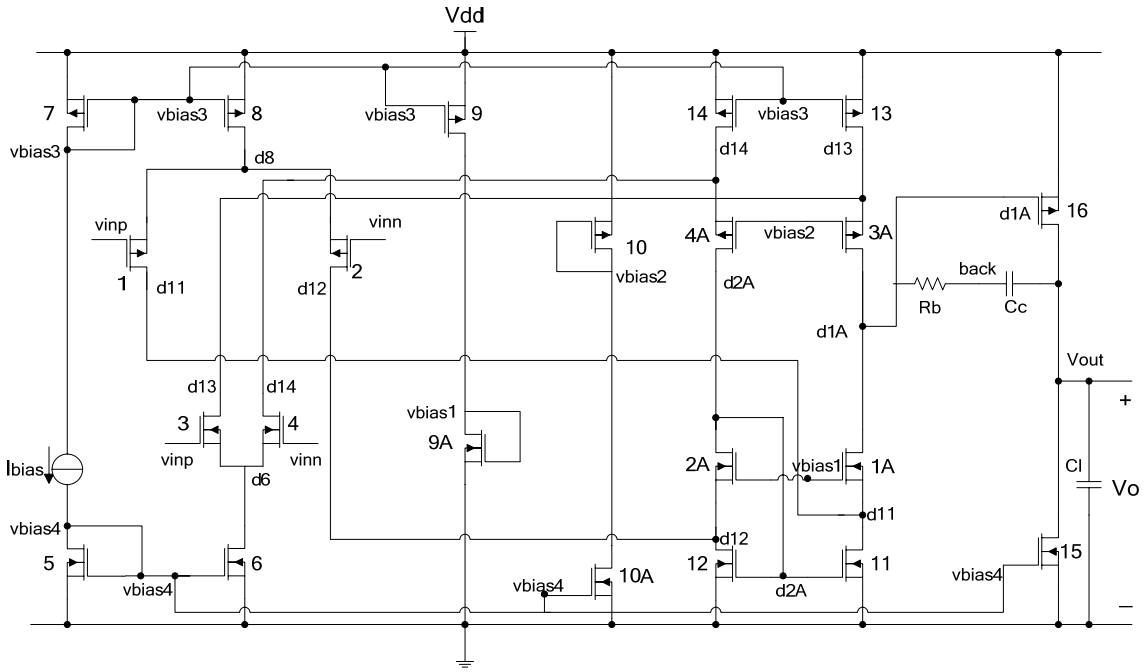


图 27 mos22 电路图

Fig.27 Schematics of mos22

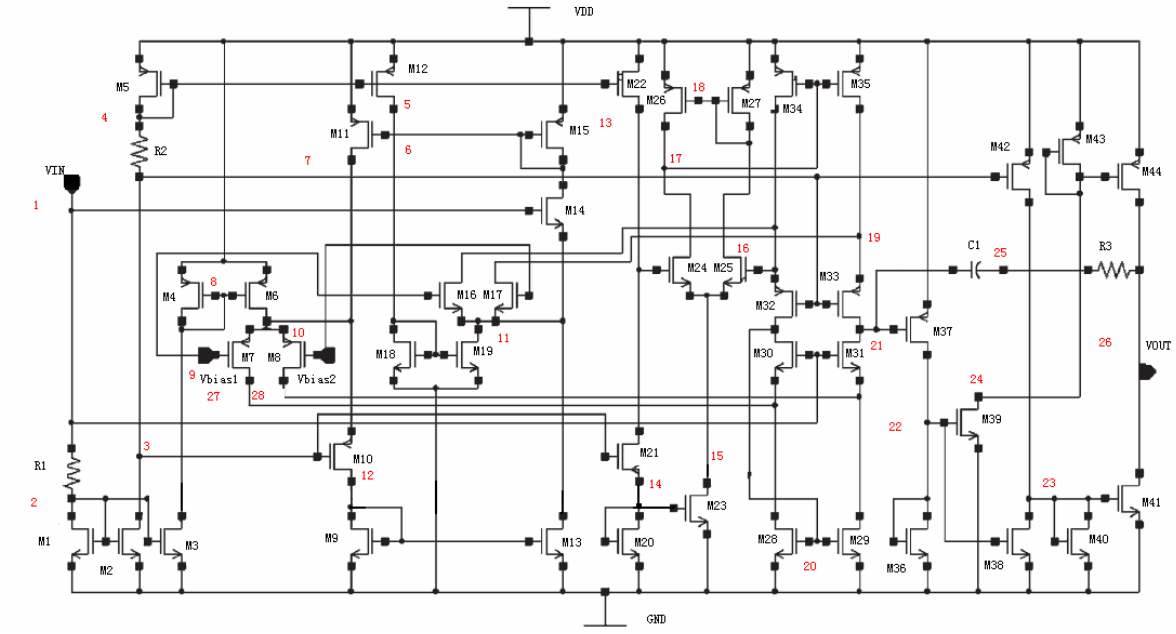


图 28 50t_opamp 电路图

Fig.28 Schematics of 50t_opamp

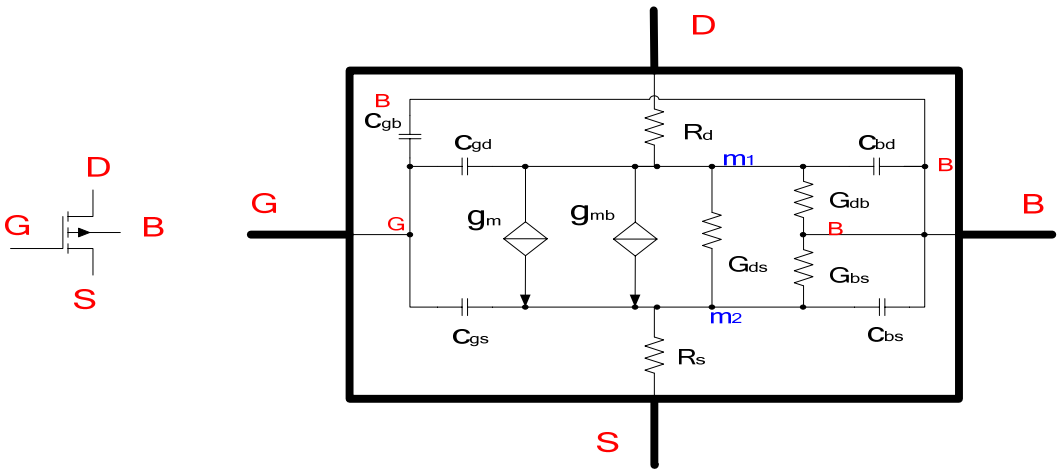


图 29 MOS 管小信号模型

Fig.29 small signal model of MOS

在测试时，我们统计了划分后每一子模块的电路节点数对应表中的第二行，模块内的元器件对应表中的第三行统计的 Symb 数（并联预处理后的数量，例如当电阻 R 与电容 C 并联时，我们的 Symb 数只增加 1），同时我们还统计了电路转换成有向图后，左（右）图的边数（依然是并联预处理后的统计数据，可以看出比 Symb 多很多，这是因为在三极管或者 MOS 管等效小信号模型中有许多的受控源，1 个 Symb 对应着 2 条边）。表中的第四行边界节点数是指模块除去接地点的边界节点数目。表中的第 7 行统计的是生成的 Tpddnodes 数，它和表中的子图数（从前面的 2.1.5 的例子可以看出，每一个 Tpddnode 的生产都伴随着约化子图的产生）一起表征了算法在调用 GRASS 时的空间消耗，同时 Tpddnodes 数和高斯结点数直接决定了我们对传输函数求值的效率（线性关系）。表中的 total 列表示是将划分后的子模块的各种统计信息简单相加得到的量，以方便和未层次化分析做对比。

表 3 ua741 分层前后对比测试

Table.3 Comparison of ua741 Numerical Analysis

ua741	层次化分析			非层次化分析
	S_1	S_2	total	
电路节点数	14	13	-	24
边数（内部）	71	87	-	160
Symb 数（内部）	40	41	-	81
边界节点数	3	3	-	2
高斯节点数	9	9	57	-
Tpddnodes 数	3，562	3，949	7，511	17,319
Reduce 后数目	1，868	2，280	4，148	10,271
子图数	2，424	2，313	4，737	9,506
时间（s）			0.532	1.844

表 3 的层次化分析是按照图 23（b）的方式进行划分的，共划分成了 2 块，并且层次化分析后的速度大约是原来的 3.5 倍。

表 4 ua725 分层前后对比测试

Table.4 Comparison of ua725 Numerical Analysis

ua725	层次化分析				非层次化分析
	S1	S2	S3	total	
电路节点数	7	17	14	-	31
边数	26	69	69	-	166
Symb 数	15	45	39	-	98
边界节点数	4	5	3	-	2
高斯节点数	16	25	9	237	-
Tpddnode 数	432	7,091	5,781	13,304	367,066
Reduce 后数	265	3,643	2,462	6,370	60,816
子图数	241	2,571	2,910	5,722	107,495
时间（s）				0.812	20.672

表 4 的层次化分析是按照图 24（b）的方式进行划分的，共划分成了 3 块，并且层次化分析后的速度大约是原来的 25 倍。

表 5 9_bw3 分层前后对比测试

Table.5 Comparison of 9_bw3 Numerical Analysis

9_bw3	层次化分析				非层次化分析
	S1	S2	S3	total	
电路节点数	13	16	10	-	34
边数	36	45	23	-	106
Symb 数	27	33	19	-	80
边界节点数	2	2	3	-	2
高斯节点数	4	4	9	57	-
Tpddnode 数	938	948	393	2,279	11,997
Reduce 后数	648	625	228	1,501	2,577
子图数	1,024	875	214	2,113	6,037
时间（s）				0.203	0.750

表 5 的层次化分析是按照图 20（b）的方式进行划分的，共划分成了 3 块，并且层次化分析后的速度大约是原来的 3.7 倍。

从这三个不同的电路的测试结果看来，即使电路规模不是太大时，分层算法，划分很少的块（2-3 块）也能取得很好的时间性能，特别是 ua725 这样规模的电路分层后，速度提高大约有 25 倍，因此，在一下节中，我们将选取更大规模的电路，来研究不同划分方式对时间性能的影响。

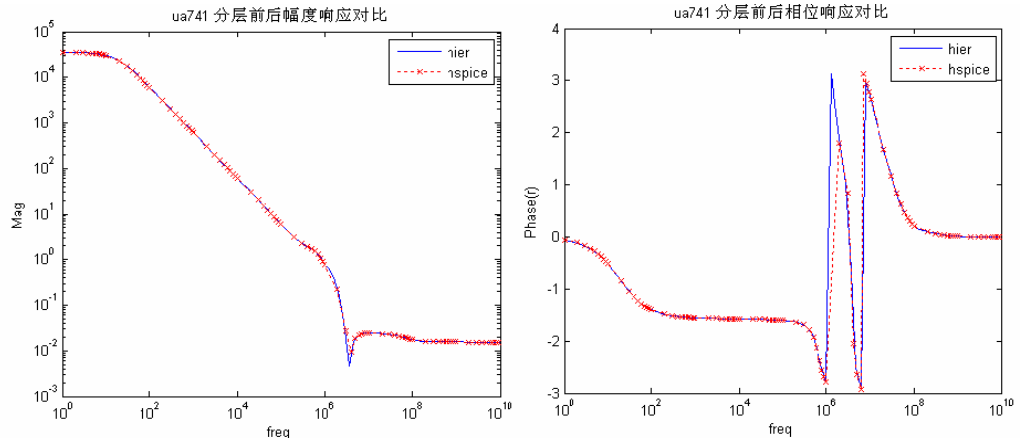


图 30 ua741 分层前后频率响应

Fig.30 Frequency Response of ua741

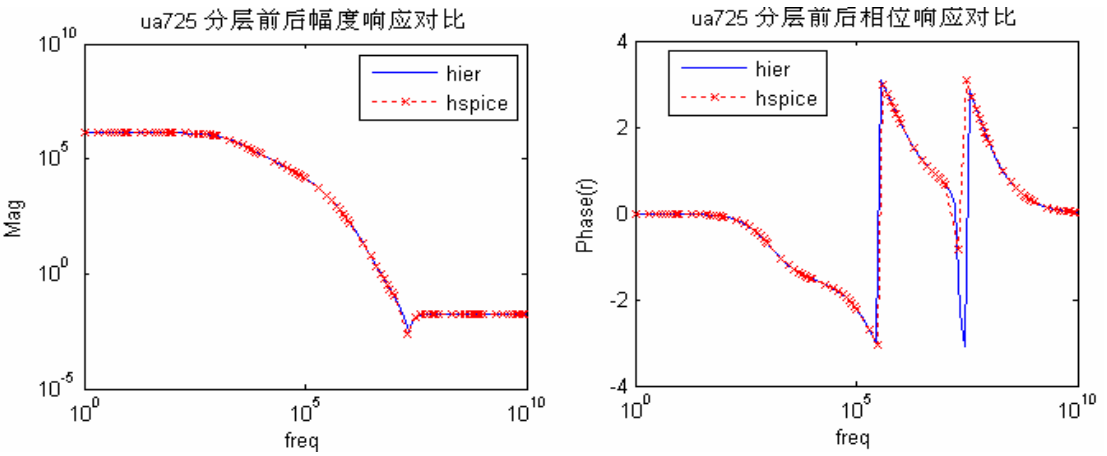


图 31 ua725 分层前后频率响应

Fig.31 Frequency Response of ua725

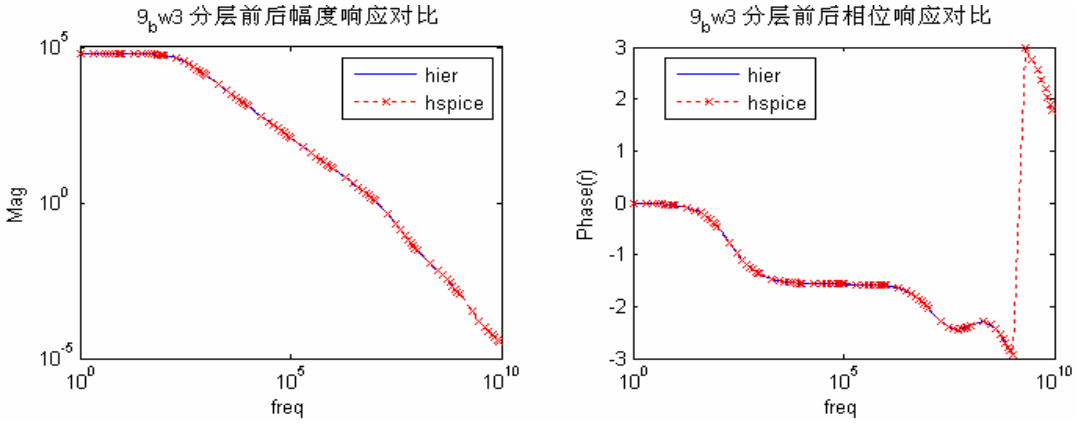


图 32 9_bw3 分层前后频率响应

Fig32　Frequency Response of 9_bw3

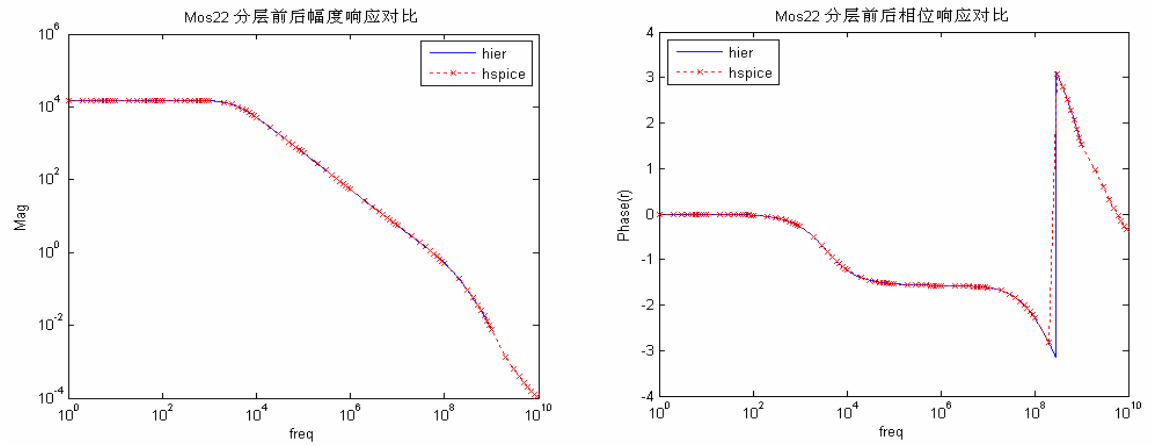


图　33 Mos22 分层前后频率响应

Fig.33　Frequency Response of Mos22

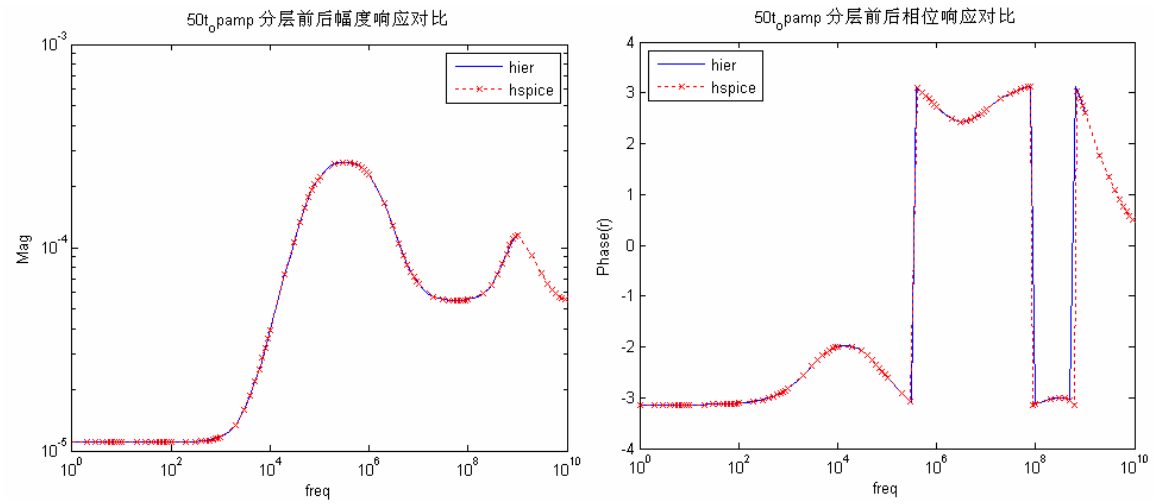


图　34　50t_opamp 分层前后频率响应

Fig.34　Frequency Response of　50t_opamp

4.2 不同划分方式测试

图 30，图 31，图 32，图 33，图 34 显示了仿真器采用了分层算法后仿真电路的频率响应（幅度 Mag 和相位），数值分析结果均与 HSPICE 的数值分析结果比较，在图 30 和图 31 中相位频率响应中，在 10^7 频率时，有 1-2 个点看起来误差稍大，

在图 33 中的相位响应在 10^8 频率时，分层前后竟然是 2 条线，虽然表面看来误差较大，实际上这是由于我们的符号化仿真器在求值时，是按照线性频率抽样计算的，也就是说，在每个频率范围内，计算的点数是一样多的，而在 HSPICE 仿真时使用的是取对数后的区间平均取点，在频率高的时候，取的点反而多，那样导致我们的符号化仿真器在高频时计算的点数比 HSPICE 少，在作图时，将这些点分段连接起来，即使是少画一个点，都有可能会使最后的曲线轨迹稍有不同，我们选取的这五个电路都是频率高时变化快，而我们出现不同的点都是高频率变化快的时候出现的，因此，我们的仿真结果其实跟 HSPICE 是一致的，只是高频时的点数比较少，导致软件绘图时估计的趋势会稍有不同。在本节中，我们对电路 Mos22（22 个 MOS 管，HSPICE 统计该电路有 61 个电路节点，元件数为 188）和 50t_opamp（44 个 MOS 管，HSPICE 统计该电路有 117 个电路节点，元件数 350）测试划分方式对层次化分析算法效率的影响。由于测试时模块组成元件及边界节点都会影响到最后的仿真结果，因此我们同时给出了划分模块中的元件信息和边界节点信息，我们以 $S_{i,j}$ 表示第 i 种划分方式中编号为 j 的模块，$TN_{i,j}$ 表示对应模块的边界节点的集合，M_i 表示电路中的标号为 i 的 MOS 管，其他 RC 器件则是对应电路图的中电阻、电容器件。

表 6 MOS22 不同划分方式对比测试

Table.6 Comparison of MOS22 Numerical Analysis

MOS22	P_1	P_2	P_3	P_4	P_5	P_6	P_7
划分块数	3	4	4	5	5	11	18
高斯节点数	824	1,057	999	989	1,365	1,829	1，280
Tpddnode 数	73,496	26,116	25,441	19,981	11,797	5,348	2，816
Reduce 后数	30,563	10,434	10,297	8,524	5,192	3,286	1，929
子图数	19,341	9,668	9,654	8,578	5,974	3,543	2，479
时间(s)	3. 359	1. 25	1. 266	1. 078	0. 785	0. 562	0. 437

划分方式说明：

第一种划分方式：

$S_{1,1}=\{M_1,M_2,M_7,M_8,M_{11},M_{12}\}$， $TN_{1,1}=\{3,4,6,10,11,14\}$；

$S_{1,2}=\{M_3,M_4,M_5,M_6,M_{13},M_{14},C_2\}$， $TN_{1,2}=\{7,8,10,11,14,15\}$；

$S_{1,3}=\{M_9,M_{10},M_{15},M_{16},M_{17},M_{18},M_{19},M_{20},M_{21},M_{22},C_1,C_c,L_1,R_b\}$

$TN_{1,3}=\{3,4,6,7,8,10,14,15,20\}$。

由于第一种划分方法中$S_{1,3}$特别大，因此我们在第二种划分种将$S_{1,3}$划分成 2 个模块$S_{2,3}$和$S_{2,4}$，具体划分如下：

$S_{2,1}=S_{1,1}$， $TN_{2,1}=TN_{1,1}$； $S_{2,2}=S_{1,2}$， $TN_{2,2}=TN_{1,2}$；

$S_{2,3}=\{M_{10},M_{16},M_{19},M_{20},M_{22},C_2,C_c,L_1,R_b\}$， $TN_{2,3}=\{2,6,7,8,10,15,20\}$；

$S_{2,4}=\{M_9,M_{15},M_{17},M_{18},M_{21},C_1\}$， $TN_{2,4}=\{2,3,4,6,14,15,20\}$。

第三种划分方式，我们依然只对$S_{1,3}$进行细分，划分成 2 个模块$S_{3,3}$和$S_{3,4}$，具体划分如下：

$S_{3,1}=S_{1,1}$， $TN_{3,1}=TN_{1,1}$； $S_{3,2}=S_{1,2}$； $TN_{3,2}=TN_{1,2}$；

$S_{3,3}=\{M_{10},M_{15},M_{16},M_{19},M_{20},M_{22},C_1,C_c,L_1,R_b\}$， $TN_{3,3}=\{2,6,7,8,10,15,20\}$；

$S_{3,4}=\{M_9,M_{17},M_{18},M_{21}\}$， $TN_{3,4}=\{2,3,4,6,14\}$。

第四种划分，我们继续只对$S_{1,3}$细分，划分成 3 个模块$S_{3,3}$、$S_{3,4}$和$S_{3,5}$，具体划分如下：

$S_{4,1}=S_{1,1}$， $TN_{4,1}=TN_{1,1}$； $S_{4,2}=S_{1,2}$， $TN_{4,2}=TN_{1,2}$；

$S_{4,3}=\{M_{10},M_{19},M_{20},M_{22}\}$， $TN_{4,3}=\{2,6,7,8,15\}$；

$S_{4,4}=\{M_9,M_{17},M_{18},M_{21}\}$， $TN_{4,4}=\{2,3,4,6,14\}$；

$S_{4,5}=\{M_{15},M_{16},C_1,L_1\}$， $TN_{4,5}=\{2,10,15,20\}$。

第五种划分在第四种划分的基础下将 $S_{1,1}$ 细分成 $S_{5,1}$ 和 $S_{5,6}$， $S_{1,2}$ 细分成 $S_{5,2}$ 和 $S_{5,7}$，具体划分如下：

$S_{5,1}=\{M_1,M_2,M_{11},M_{12}\}$， $TN_{5,1}=\{3,4,5,6,10,11\}$；

$S_{5,2}=\{M_3,M_4,M_{13},M_{14},C_2\}$， $TN_{5,2}=\{7,8,10,11,14,17\}$；

$S_{5,3}=S_{4,3}$， $TN_{5,3}=TN_{4,3}$； $S_{5,4}=S_{4,4}$， $TN_{5,4}=TN_{4,4}$； $S_{5,5}=S_{4,5}$， $TN_{5,5}=TN_{4,5}$；

$S_{5,6}=\{M_7,M_8\}$， $TN_{5,6}=\{5,14\}$； $S_{5,7}=\{M_5,M_6\}$， $TN_{5,7}=\{15,17\}$。

第六种划分在第四种划分的基础下将 $S_{1,1}$ 细分成 $S_{5,1}$ 和 $S_{5,6}$， $S_{1,2}$ 细分成 $S_{5,2}$ 和 $S_{5,7}$，具体划分如下：

$S_{6,1}=\{M_1,M_2\}$， $TN_{5,1}=\{3,4,5,10,11\}$；

$S_{6,2}=\{M_3,M_4,C_2\}$， $TN_{6,2}=\{7,8,10,11,17\}$；

$S_{6,3}=\{M_{10},M_{22}\}$， $TN_{6,3}=\{13,15\}$； $S_{6,4}=\{M_9,M_{21}\}$， $TN_{6,4}=\{12,14\}$；

$S_{6,5}=S_{4,5}$， $TN_{6,5}=TN_{4,5}$； $S_{6,6}=\{M_7,M_8\}$， $TN_{6,6}=\{5,14\}$；

$S_{6,7}=\{M_5,M_6\}$， $TN_{6,7}=\{15,17\}$； $S_{6,8}=\{M_{11},M_{12}\}$， $TN_{6,8}=\{3,4,6\}$；

$S_{6,9}=\{M_{13},M_{14}\}$， $TN_{6,9}=\{7,8,14\}$； $S_{6,10}=\{M_{19},M_{20}\}$， $TN_{6,10}=\{2,6,7,8,13\}$；

$S_{6,11}=\{M_{17},M_{18}\}$， $TN_{6,11}=\{2,3,4,6,12\}$；

第七种划分进行的极限测试，在第六种划分的基础下，如果 MOS 没有接成二极管形式（即 MOS 管的漏端与源端没有接在一起），则将第六种划分的对应模块拆分成一个一个的 MOS 管，如果是接成了二极管形式，则保持不变（因为使用的是约化导纳矩阵，是除去地点，只含有边界节点的矩阵，因此除了地点，还需要 2 个边界节点，因此不能将接成二极管形式的 MOS 管单独放在一个模块里）

表 7 50t_opamp 不同划分对比测试

Table.7 Comparison of 50t_opamp Numerical Analysis

	P_1	P_2	P_3
划分块数	17	11	12
高斯节点数	3,779	5,573	2，813
Tpddnode 数	12,629	19,980	23，790
Reduce 后数	6,754	7,302	11，100
子图数	7,769	10,078	12，012
时间(s)	1.157	1.485	1.515

划分方式说明：

第一种划分采用的是随意划分，每一模块内含有 2-4 个 MOS 管，具体方式如下：

$S_{1,1}=\{M_{43},M_{44}\},TN_{1,1}=\{24,26\}$；

$S_{1,2}=\{M_{40},M_{41}\},TN_{1,2}=\{23,26\}$；

$S_{1,3}=\{M_{37},C_1,C_2,R_3\},TN_{1,3}=\{21,22,26\}$；

$S_{1,4}=\{M_{36},M_{39}\},TN_{1,4}=\{22,24\}$；

$S_{1,5}=\{M_{20},M_{23}\},TN_{1,5}=\{14,15\}$；

$S_{1,6}=\{M_{28},M_{29},M_{30},M_{31}\},TN_{1,6}=\{1,20,21,27,28\}$；

$S_{1,7}=\{M_{32},M_{33},M_{34},M_{35}\},TN_{1,7}=\{3,16,17,19,20,21\}$；

$S_{1,8}=\{M_{24},M_{25},M_{26},M_{27}\},TN_{1,8}=\{13,15,16,17\}$；

$S_{1,9}=\{M_4,M_6,M_7,M_8\},TN_{1,9}=\{7,8,9,10,27,28\}$；

$S_{1,10}=\{M_{16},M_{17},M_{18},M_{19}\},TN_{1,10}=\{5,9,10,11,16,19\}$；

$S_{1,11}=\{M_{11},M_{14},M_{15}\},TN_{1,11}=\{1,7,11\}$；

$S_{1,12}=\{M_{9},M_{10},M_{13}\},TN_{1,12}=\{3,7,11\}$；

$S_{1,13}=\{M_{5},M_{42},R_{2}\},TN_{1,13}=\{3,4,23\}$；

$S_{1,14}=\{M_{21},M_{22}\},TN_{1,14}=\{3,4,13,14\}$；

$S_{1,15}=\{M_{1},M_{2},M_{3},R_{1}\},TN_{1,15}=\{1,3,8\}$；

$S_{1,16}=\{M_{12}\},TN_{1,16}=\{4,5\}$；

$S_{1,17}=\{M_{38}\},TN_{1,17}=\{22,23\}$；

第二种划分是优先划分了接成差分对（其余则是从原理图中看起来连接比较紧密的就放在了一个模块），每个模块里面含有 4 个 MOS 管：

$S_{2,1}=\{M_{36},M_{39},M_{43},M_{44}\},TN_{2,1}=\{22,26\}$；

$S_{2,2}=\{M_{37},M_{38},M_{40},M_{41},C_{1},C_{2},R_{3}\},TN_{2,2}=\{21,22,23,26\}$；

$S_{2,3}=\{M_{20},M_{21},M_{22},M_{23}\},TN_{2,3}=\{3,4,13,15\}$；

$S_{2,4}=\{M_{32},M_{33},M_{34},M_{35}\},TN_{2,4}=\{3,16,17,19,20,21\}$；

$S_{2,5}=\{M_{24},M_{25},M_{26},M_{27}\},TN_{2,5}=\{13,15,16,17\}$；

$S_{2,6}=\{M_{28},M_{29},M_{30},M_{31}\},TN_{2,6}=\{1,20,21,27,28\}$；

$S_{2,7}=\{M_{4},M_{6},M_{7},M_{8}\},TN_{2,7}=\{7,8,9,10,27,28\}$；

$S_{2,8}=\{M_{16},M_{17},M_{18},M_{19}\},TN_{2,8}=\{5,9,10,11,16,19\}$；

$S_{2,9}=\{M_{9},M_{13},M_{14},M_{15}\},TN_{2,9}=\{1,6,11,12\}$；

$S_{2,10}=\{M_{5},M_{11},M_{12},M_{42},R_{2}\},TN_{2,10}=\{3,4,5,6,7,23\}$；

$S_{2,11}=\{M_{1},M_{2},M_{3},M_{10},R_{1}\},TN_{2,11}=\{1,3,7,8,12\}$；

第三种划分（容易识别的结构优先划分，剩余的任意划分，非均匀划分），具

体方式如下：

$S_{3,1}=\{M_3,M_4,M_6,M_7,M_8\},TN_{3,1}=\{2,7,9,10,27,28\}$；（该模块是电流源模块，由于 M7 和 M8 是差分输入管对，但是不含地，所以将他们和电流源模块放在了一起。）

$S_{3,2}=\{M_{12},M_{16},M_{17},M_{18},M_{19}\},TN_{3,2}=\{4,9,10,11,16,19\}$；（该模块与 $S_{3,1}$ 是对称的。）

$S_{3,3}=\{M_{38},M_{40},M_{42}\},TN_{3,3}=\{3,22,23\}$；（$M_{42}$ 与 M_{38} 是 2 个不同的放大电路，M40 作为 M_{42} 的负载，由于它们的规模都很小，我们也将它们放在了一个模块内）

$S_{3,4}=\{M_{39},M_{43}\},TN_{3,4}=\{22,24\}$；（放大电路）

$S_{3,5}=\{M_{41},M_{44}\},TN_{3,5}=\{23,24,26\}$；（2 个放大电路，但是由于只有 1 个 MOS 管，所以将它们放在了一个模块内）

$S_{3,6}=\{M_{36},M_{37},C_1,C_2,R_3\},TN_{3,6}=\{21,22,26\}$；（以节点 3，21，我们可以很容易的将电路分为 2 部分，后面的主要是由放大电路组成，将这些放大电路挑出后，剩余的组成该模块）

$S_{3,7}=\{M_9,M_{10},M_{11},M_{13},M_{14},M_{15}\},TN_{3,7}=\{1,3,7,11\}$；（互为电流镜，稳定节点 7，11 电压）

$S_{3,8}=\{M_{32},M_{33},M_{34},M_{35}\},TN_{3,8}=\{3,16,17,19,20,21\}$；（差分管 M_{18},M_{19} 的负载）

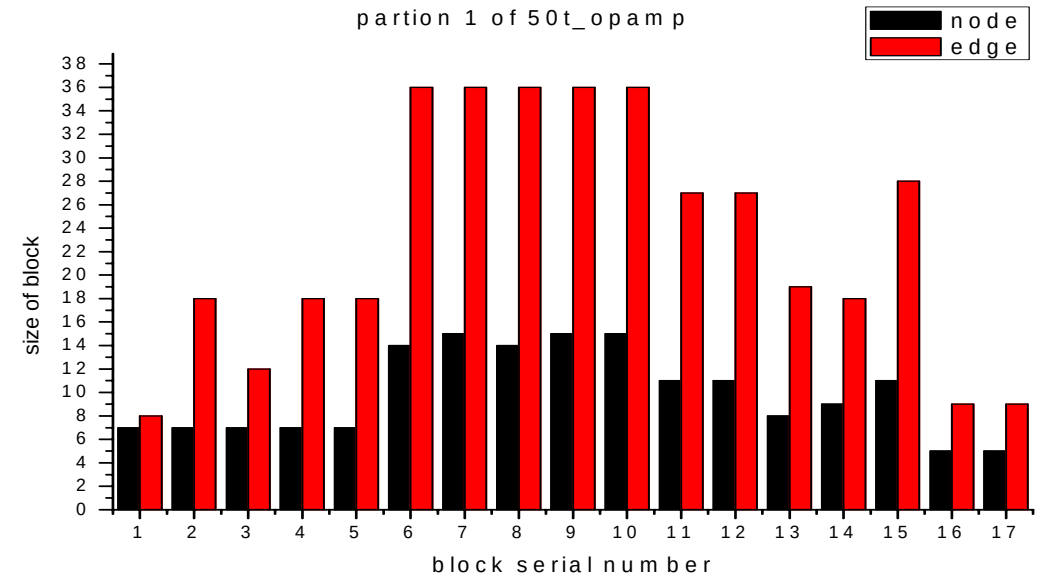
$S_{3,9}=\{M_{28},M_{29},M_{30},M_{31}\},TN_{3,9}=\{1,20,21,27,28\}$；（差分管 M_7,M_8 的负载）

$S_{3,10}=\{M_{24},M_{25},M_{26},M_{27}\},TN_{3,10}=\{13,15,16,17\}$；（图中看来较紧密）

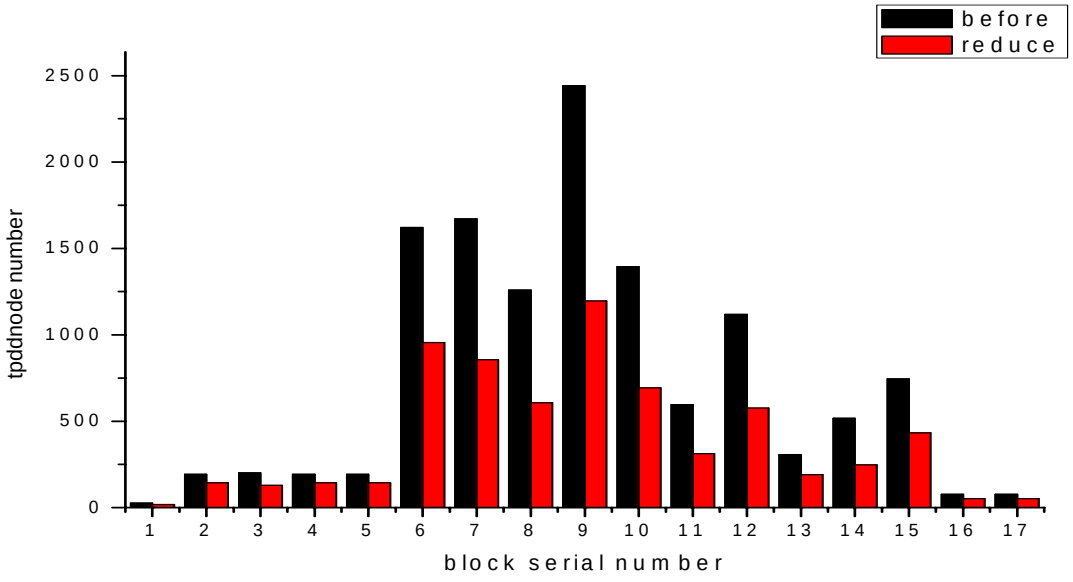
$S_{3,11}=\{M_{20},M_{21},M_{22},M_{23}\},TN_{3,11}=\{3,4,13,15\}$;

$S_{3,12}=\{M_1,M_2,M_5,R_1,R_2\},TN_{3,12}=\{1,2,3,4\}$。

表 7 中有一个十分有趣的现象，即虽然第三种划分（有 12 块），但是总共的空间消耗（即产生的高斯结点数与 tpddnode 之和）却比第二种划分（划分只有 11 块）的多。因此我们将给出每种划分的子模块所产生的结点的图表便于比较。我们以电路中的节点数和 GRASS 构建的有向图边数来衡量划分模块的规模。



（a）

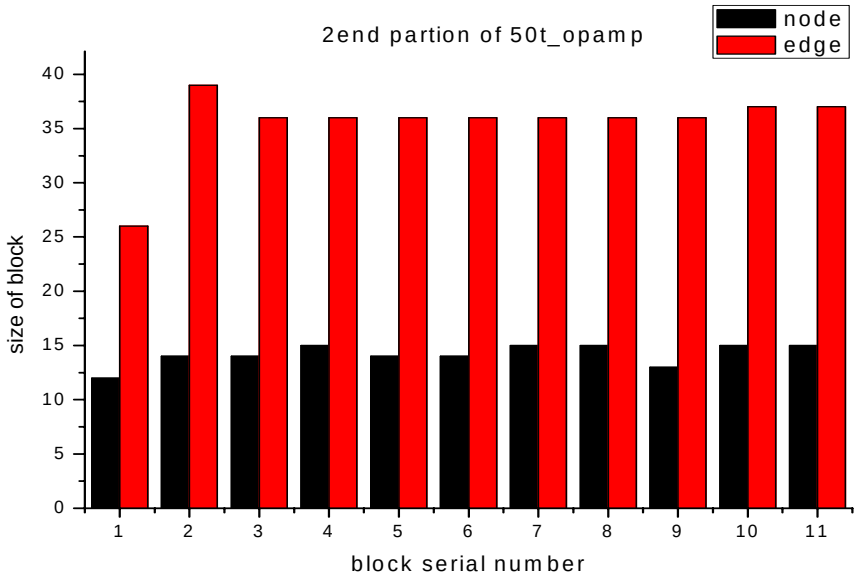


(b)

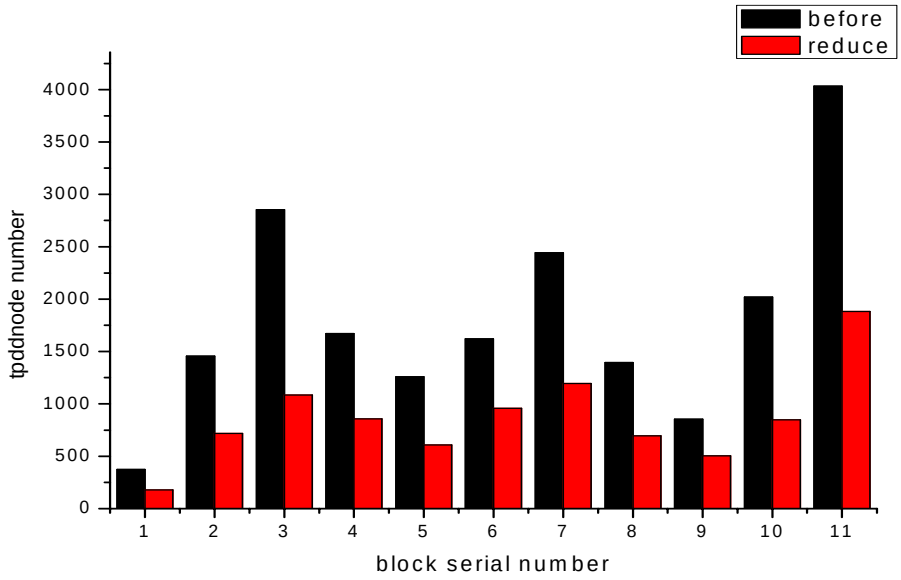
图 35 50t_opamp 第一种划分信息统计

(a)各模块电路规模；（b）对应模块产生的 tpddnode 数

Fig. 35 Statistics of 50t_opamp 1st partition



（a）

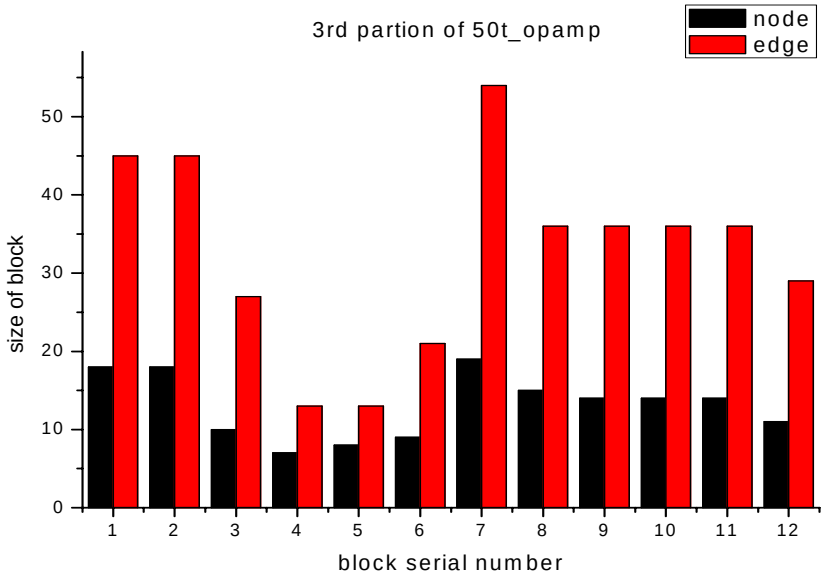


(b)

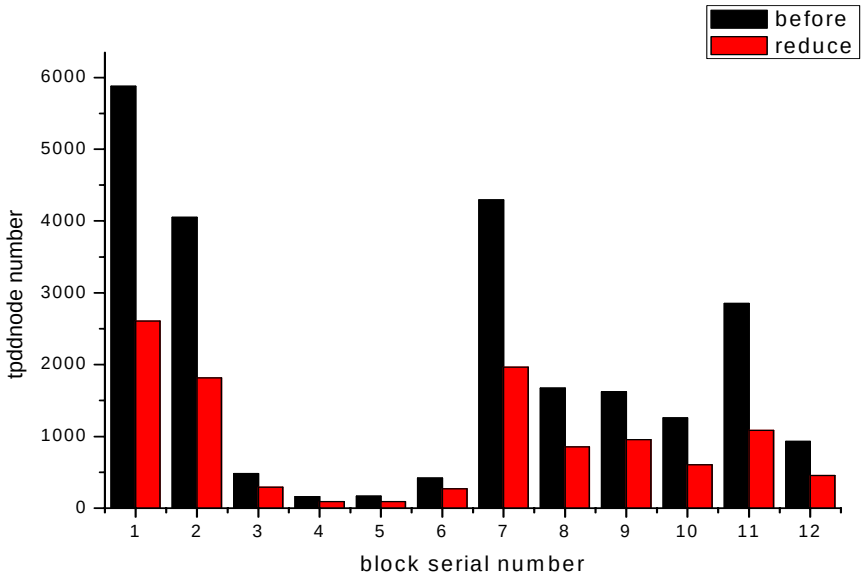
图 36 50t_opamp 第二种划分信息统计

(a)第二种划分各模块电路规模;（b）对应模块产生的 tpddnode 数

Fig. 36 Statistics of 50t_opamp 2nd t partition



（a）



(b)

图 37 50t_opamp 第三种划分统计信息

(a)第三种划分各模块电路规模;（b）对应模块产生的 tpddnode 数

Fig. 37 Statistics of 50t_opamp 3rd partition

从图 35、图 36、图 37 中我们可以看到，50t_opamp 的第二种划分方法是按照 MOS 管数目划分，每个模块控制在 4 个，最后产生的 tpddnode 分布也较第一种和第

三种划分均匀很多，速度比第三种划分快（功能块划分，从结果中可以看到，虽然产生的高斯节点数的确是最少的，但是分块的规模十分不均匀，而且速度反而是最慢的，因此，除非是工程师使用时指定必须按此划分，否则我们选择按照模块内的器件数目均匀划分倒是不错的选择，更主要的原因是这样便于计算机实现自动模块划分）。第一种划分的情况很极端，虽然也不均匀，但是如果我们以分组的情况来看，及 tpddnode 数目非常小的看为一组，较大的看为另一组，组内的变化要小很多，因此如果以后使用并行算法来仿真电路时，通过任务的合理组合，也许可以达到分析时间的较优，但是目前还无法确定，如果使用并行，是否第一种划分优于第二种划分。

4.3 本章小结

本章我们测试了 5 个不同规模的三极管电路和 CMOS 电路，在电路规模比较小（20-25 个三极管时），划分电路成 2-3 块就能很快的得到正确的仿真和数值结果，因此我们选择了大规模的电路来初步探索划分方式对层次化分析的效率影响。在电路规模很大时，我们可以将电路划分成很小的块（2-4 个 MOS 管）时，速度就可以很快，而且由于是任意端口划分的，不需要按功能块划分也能有很快的速度（这样更易于计算机实现），这说明我们的层次化分析从模块划分到仿真都是易实现的。

第五章 总结与展望

5.1 结论

本文提出了一个层次分析方法在拓扑符号化电路仿真器 GRASS 中的应用。使用 GRASS 仿真器快速求解规模不太大模块的端口导纳函数，结合高斯消去法在子电路间连接数较少的情况下，该方法能有效分析大规模放大器电路。实验测试结果验证了算法的有效性和正确性。在电路规模不太大时（例如 ua725），仅划分 2-3 块就可以很快得得到仿真结果，电路规模越大，采用层次化分析的时间性能，空间性能优势就越明显（如果不采用层次化分析方法，GRASS 无法完成个别电路的符号化分析）。

另外我们还初步探索了不同划分方式对层次化分析仿真效率的影响，通过对含有 44 个 MOS 管电路的 3 种不同划分方式测试，分别是任意划分、均匀划分和按功能块划分，测试结果表明，无论是哪种划分方式，我们都能快速得分析该电路。按照电路的功能模块划分，产生的高斯结点最少，但是 tpddnode 是三种划分之中最多的（这里面还有 order 的干扰），如果按照功能块划分，会导致各模块的规模极不均匀，不好的 order 对于规模大的电路影响更显著，但是采用按功能块划分，这样才更符合模拟工程师们的设计习惯，而且对于电路的层次化优化可能可取得更好的效果，因此目前我们还不能确定究竟是哪种划分方式更好，可能要看后续研究的期望目标，但是无论何种划分，我们的层次化分析都是高效快速的，这为以后的研究提供了可能。

5.2 研究展望

本文论述的内容对层次化分析在 GRASS 中的应用进行了初步探索，但是由于结合了高斯消去法后，以 GRASS 不产生冗余项和生成项展平为代价实现的，产生冗余项会对符号化近似分析产生干扰，而生成项展平，是可以更加清晰得揭示元器件对于电路特性的影响，因此未来的研究可以有以下几点：

1、探索采用层次化分析后，如何对电路进行敏感性分析，以深刻揭示对电路特性，并给出改善建议，文献[9]给出了如何对嵌套符号表达式进行敏感性分析的方法，由于在合并子模块过程使用的是符号化高斯消去法使得顶层电路指定的传递函数变成了非展平的（子模块越小，最终的分析结果嵌套越厉害），因此在未来可做这方面的结合，实现 GRASS 加入层次化分析功能后的敏感性分析；

2、目前实现的模块是每一块必须包含地，这样将会使得模块划分自由度降低，未来实现可不含地的层次化分析将可以为模块划分的研究提供更大的自由；

3、模块划分依据，在应用中我们可以根据需要灵活地选择划分规模。例如如果我们希望得到紧凑的表达式以实现快速计算，可以选择将电路划分成小模块（每一块都必须包含地），如果希望得到的生成项更加平坦，我们可以选择将电路划分成较大的模块，以充分发挥 GRASS 的优势。但是这之间应该有一个平衡，或者参考点，即我们希望得到的生成项更展平，但是同时希望分析也近可能快，这时每个模块最好是多大规模；或者结合以后可能进行的层次化优化研究，确定是按功能块划分还是可任意划分（因为按功能块划分的计算机实现比任意划分难度大很多）。

4 消除或减少冗余项，在 GRASS 模块中采用了层次化分析方法后，每一个模块的导纳矩阵是多个 BDD 组成的矩阵，这些矩阵间一定存在共享，同时由于 GRASS 在 BDD 的操作边上关联了正负号，使得部分共享节点可能意味着冗余项的产生，产生冗余项意味着有部分时间浪费在了计算无用的信息上，并且冗余项的产生会对符号化近似分析产生干扰；

5 高斯消去过程中的主元选择。在文献[3]中提到了为了避免除零，而使用先通分最后再相除的做法，并提出了一种主元选择策略，以期最少的算术运算实现高斯消去内部节点，这个可以跟我们的 GRASS 层次化分析的后续研究结合起来，同样实现最后一步才相除的功能；

6 建立符号化模块库。我们发现，在模拟电路中，特别是 CMOS 工艺的电路中，电路存在着极强的对称性，P 型器件和 N 型器件，当我们将这两种不同的类型的器件（具有相同的拓扑结构）划分入不同的模块时，显而易见的是，使用 GRASS 分析，得到的生成树应该是一样的，并且由于模拟电路中，一些结构单元是反复出现的，因此，如果能实现将这些经常出现的模块对应的符号化约化导纳矩阵建立成库，在后续的分析时，只许建立简单的映射关系（即器件符号映射），那么将可以节省很多时间（在层次化分析时，这种重复出现的概率比不采用层次化分析要大很多）；

7 并行算法研究。由于使用了层次化分析后，每个子模块间都是相互独立的，这为并行算法的研究提供了方便，如果可以使用并行来仿真电路，GRASS 可以取

得更好的时间性能。

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] F. LI and P. Y. Woo, "A new concept- the virtual circuit and its application in large network analysis with tearing," International Journal of Circuit Theory and Applications, 27, pp. 283-291, 1997.

[3] M. Pierzchala and B. Rodanski, "Generation of sequential symbolic network functions for large-scale networks by circuit reduction to a two-port." IEEE Trans.on Circuits and Systems—I: Fundamental Theory and Applications, vol. 48, no. 7, pp. 906–909, 2001

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

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

[6] J. A. Starzky and A. Konczykowska, "Flowgraph analysis of large electronic networks," IEEE Trans. Circuits Syst., vol. 33, no. 3, pp. 302–315, Mar. 1986.

[7] M.M. Hassoun and P.-M Lin, "A hierarchical network approach to symbolic analysis of large-scale networks," IEEE Trans. Circuits Syst., vol. 42, pp. 201–211, Apr. 1995.

[8] G. Shi, W. Chen and C.-J. Shi, "A Graph Reduction Approach to Symbolic Circuit Analysis," accepted by *12th Asia and South Pacific Design Automation Conference (ASP-DAC 2007), Yokohama, Japan*, pp. 197-202,Jan., 2007.

[9] J. Echtenkamp, M. Hassoun, R. S. Syst and N. Beach, "Implementation issues for symbolic sensitivity analysis," Proc. 39th Midwest symposium on Circuits and Systems (MWSCAS 1996), pp. 429-432, 1997.

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

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

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

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

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

[15] M. M. Hassoun and K. S. McCarville, "Symbolic Analysis of Large-Scale Networks Using a Hierarchical Signal Flowgraph Approach," Analog Integrated Circuits and Signal Processing vol. 3,

no.1,pp.31-42, Jan., 1993.
[16] 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.
[17] G. E. Alderson and P. M. Lin, “Computer generation of symbolic network functions – a new theory and implementation,” IEEE Trans. Circuit Theory, vol. CT-20, pp. 48-56, Jan., 1973.
[18] K. Singhal and J. Vlach, “Generation of immittance functions in symbolic form for lumped distributed active networks,” IEEE Trans. Circuits Syst., vol. CAS-21, pp. 57-67, Jan. 1974.
[19] X. Wang and L. Hedrich, “An approach to topology synthesis of analog circuits using hierarchical blocks and symbolic analysis”, Proc. Asia South Pacific Design Automation Conference, pp. 700-705, Jan. 2006.
[20] X. Wang and L. Hedrich, “Hierarchical Symbolic Analysis of Analog Circuits Using Two-Port Networks,” 6th WSEAS Int. Conf. on Circuits, Systems, Electronics, Control & Signal Processing, Cairo, Egypt, pp.21-26, Dec ., 2007.
[21] F. V. Fernindez, A. Rodriguez-Vizquez, J. L. Hertas and G. Gielen, Symbolic Analysis Techniques: Application to Analog Design Automation. IEEE Press, 1998.
[22] M.M. Hassoun and P.M. Lin, “An efficient partitioning algorithm for large-scale circuits,” in: Proceedings of the IEEE International Symposium on Circuits and Systems, pp. 2405–2408,1990.
[23] X.-D Tan and C.-J. Shi, “Balanced Multi-Level Multi-Way Partitioning of Large Analog Circuits for Hierarchical Symbolic Analysis”, Proc. Asia and South Pacific Design Automation Conference (ASP-DAC'99) , pp. 1-4, Jan., 1999.
[24] A. Sangiovanni-Vincentelli, L.-K. Chen and L.O. Chua, “An efficient heuristic cluster algorithm for tearing large-scale networks,” IEEE Trans. Circuits Systems CAS-24 (12) 709–717, 1977.
[25] G. Minty, “A simple algorithm for listing all the trees of a graph,” IEEE Trans. on Circuit Theory, vol. CT-12, p. 120, 1965.
[26] W. Chen, Applied Graph Theory-Graphs and Electrical Networks. Amsterdam: North-Holland, 1976.
[27] F. Femandez, A. Rodriguez-Vazquez, and J. Huertas, “A tool for symbolic analysis of analog integrated circuits including pole/zero extraction,” in *Proc. ECCTD*, pp.752-761,1991.
[28] C.-K. Cheng, Z. Qin and X. D. Tan, Symbolic analysis and reduction of VLSI circuits, Springer US , 2005.
[29] X.-D. Tan and C.-J. Richard Shi, “Efficient Approximation of Symbolic Expressions for Analog Behavioral Modeling and Analysis,” IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, Vol. 23, No. 6, pp.907- 918, Jun., 2004
[30] 陈微微. 符号化模拟电路仿真器的实现与应用[硕士论文].上海:上海交通大学. 2006.

附录 1 双端口 Y 矩阵求解系统传输函数

符号说明

u_1, i_1：表示输入端的电压、电流（规定流出端口为正）

u_2, i_2：表示输出端的电压、电流（规定流出端口为正）

则他们之间的关系可用如下表达式表示：

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} U_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} U_2 \\ I_2 \end{bmatrix} \tag{2}$$

1. 输出端变量为 u_2，我们理解节点 2 是开路的，因为此时没有流出端口的电流，此时有

$$\begin{cases} i_2 = 0 \\ u_1 = t_{11} u_2 \\ i_2 = y_{21} u_1 + y_{22} u_2 = 0 \\ i_1 = t_{21} u_2 \\ i_1 = y_{11} u_1 + y_{12} u_2 \end{cases} \tag{3}$$

1.1 当所求是的传输函数是 $\frac{u_2}{u_1}$ 时有：

$$\frac{u_2}{u_1} = \frac{1}{t_{11}} = -\frac{y_{21}}{y_{22}} \tag{4}$$

1.2 当所求是的传输函数是 $\frac{u_2}{i_1}$ 时，有：

$$\frac{u_2}{i_1} = \frac{1}{t_{21}} = -\frac{y_{11}y_{22} - y_{12}y_{21}}{y_{21}} = -\frac{\det(Y)}{y_{21}} \tag{5}$$

2. 输出端变量为 i_2，此时节点 2 是短路到地的，有

$$\begin{cases} u_2 = 0 \\ u_1 = t_{12} i_2 \\ i_1 = t_{22} i_2 \\ i_1 = y_{11} u_1 \\ i_2 = y_{21} u_1 \end{cases} \tag{6}$$

2.1 当所求是的传输函数是$\frac{i_2}{u_1}$时有：

$$\frac{i_2}{u_1} = \frac{1}{t_{12}} = y_{21} \tag{7}$$

2.2 当所求是的传输函数是$\frac{i_2}{i_1}$时，有：

$$\frac{i_2}{i_1} = \frac{1}{t_{22}} = \frac{y_{21}}{y_{11}} \tag{8}$$

附录 2 符号与标记

压控电压源	(Voltage control voltage source, VCVS）
流控电压源	(Current control voltage source, CCVS）
压控电流源	(Voltage control voltage source, VCCS)
流控电流源	(Current control voltage source, CCCS)
零器	(Nullor)
零阻器	(Nullator)
泛阻器	(Norator)
约化节点导纳矩阵	(Reduced Nodal Admittance Matrix, RMNA)
二叉决策图	(Binary Decision Diagram, BDD)
简化排序二叉决策图	(Reduced Order Binary Decision Diagram, ROBDD)

攻读研究生学位期间已发表或录用的论文

[1] 陈硕，“层次化分析在符号化模拟电路仿真器中的应用”，信息技术，2010 年第<3-5>期（已录用，论文原名《模拟电路层次化分析方法在符号化电路仿真中的应用》，由于论文题目超长，因此更名。）

致 谢

转眼间，两年半的研究生生涯即将结束，回首在交大的学习生活，收获颇丰。

首先要感谢我亲爱的妈妈，无论何时，您总是在背后支持我，鼓励我，给予我信心，给予我战胜困难的勇气和力量。

其次，要感谢我的导师施国勇教授，您渊博的学识，严谨的治学作风和踏实认真的态度让我受益匪浅。再要感谢的是李章全老师及模拟射频组的闫涛涛博士以及模拟射频组的韩世英、潘兆林、张立、王旭等同学，当我看着电路头皮发麻的时候，正是你们细致耐心的讲解，才让我当初的学习和研究层次化分析时划分电路的工作轻松了很多。另外还要特别感谢郝志刚博士以及 EDA 组里的每一位成员，于雪红师姐，李骥，陈安，王婷，谭锟元，曾媚，谢边村，黄伟坚，马迪铭，徐辉，李小鹏等，当我的代码出现 bug，郁闷得一塌糊涂的时候，正是你们提供各种帮助，让我顺利调通。感谢徐迪和刘安两位师兄，正是你们当初拖着我代表小组去参加学院组织的篮球赛，让我多了一次奇特、难忘又非常有趣的经历。

感谢学院里的每一位老师和同学，正是你们组成了一个如此温暖可爱又朝气蓬勃的集体，在这里，我衷心地祝愿大家事业有成，生活美满，学院也越来越好。

陈硕

2009 年 12 月 11 日