申请上海交通大学硕士学位论文

符号化模拟电路敏感性的分析与应用[1]

学　　校：上海交通大学

院　　系：微电子学院

班　　级：　Z0621091

学　　号：1062109065

专　　业： 软件工程

硕 士 生： 孟晓旋

导　　师： 施国勇　教授

上海交通大学微电子学院

2008 年 12 月

此研究由上海市浦江人才基金资助（No.07pj14053）

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

Symbolic AC Sensitivity Analysis for Analog Design

Author: MENG, Xiaoxuan

Specialty: Software Engineering

Advisor: Prof. SHI, Guoyong

School of Microelectronics

Shanghai Jiao Tong University

December 7, 200

符号化模拟电路敏感性的分析与应用

摘 要

本文首先描述了一个基于拓扑方法的符号化模拟电路仿真器(GRASS),它通过一种新提出的图约化算法建立表达电路生成项的二分判定图。相比较于其它的符号化仿真器，它在电路处理规模和表达形式上都具有优势。本文在此基础上实现了电路元件交流灵敏度信息的提取，并应用于实际电路的分析中。

本文在第一章和第二章介绍了符号化分析方法和 GRASS 的分析原理和实现方法，第三章讲述灵敏度的概念和电路元件交流灵敏度提取的具体实现过程，第四章给出了实验的分析结果以及电路元件灵敏度信息在实际电路中的应用，最后对全文进行了总结和展望。

关键词：交流灵敏度，二分判定图，图约化算法，符号化分析

Symbolic AC Sensitivity Analysis for Analog Design

ABSTRACT

GRASS (graph reduction analog symbolic simulator) adopted a graph reduction approach based on a set of graph reduction rules developed recently. A Binary Decision Diagram (BDD) is used in the implementation of the symbolic circuit simulator. Comparing to other symbolic simulators, GRASS has advantages of analyzing larger analog circuits and one-to-one mapping from the circuit parameters to the GRASS symbols. The sensitivity information can be obtained easily and implemented to analog design.

The thesis is organized as follows. A brief introduction to symbolic analysis is in Chapter1. The graph reduction rules and the implementation of GRASS are presented in Chapter 2. The concept of sensitivity and details of computing AC sensitivity on GRASS are described in Chapter 3. Experimental results are reported and basic implements of sensitivity are discussed in Chapter 4. Conclusions and future work are reported in Chapter 5.

Keywords: AC Sensitivity, Binary Decision Diagram, Graph Reduction, Symbolic Analysis

目 录

图 目 录

第一章 绪论

随着以集成电路为核心的微电子技术的迅猛发展，电子信息产业已经成为全世界产业结构中最重要的、发展最迅速的一个部分。目前，随着集成电路的发展继续向高密度高速度的方向突飞猛进，超大规模集成电路(Very Large Scale Integrated Circuit, VLSI)已经称为当今媒介信息的主要载体。大规模集成电路技术的发展那还推动了片上系统(System on Chip, SoC)等大型系统芯片的诞生。随着计算机科学的不断发展，电路设计的自动化程度越来越高。由于数字电路设计目前设计自动的理论和技术都已比较成熟，而大部分的模拟电路设计依旧较大程度的依赖工程师本身，因此模拟电路设计自动化程度需要很大范围的提升。

如今，几乎所有的电子工程师和学生都在使用数值电路仿真器(SPICE【1】和 ELDO)。由于数值电路仿真器不能解决集成电路设计中的所需要的所有的分析功能。实际上，这些数值电路仿真器主要用来验证既定规模的电路性能。然而，设计者需要从设计参数和特性曲线的关系中预测不确定规模电路的性能。预测性能的关系主要包括传输函数、零点和极点、根轨迹、谐波失真系数等等。一直以来，这些关系主要依赖人工计算，自动计算这些关系的工具称之为符号化分析器。

从符号化分析的概念的提出到现在，已经有大量的符号化模拟电路仿真器出现，但很少有提供符号化分析灵敏度信息的概念，灵敏度尤其在电路优化设计中起着非常重要的作用。电路优化设计是 CAD 的一项非常重要的内容，它就是指在一定的容差范围之内，对电路整体性能进行优化设计从而达到设计的要求。由于工艺条件的限制和环境温度等因素的变化，元器件实际值和标称值之间总有误差，使得实际电路的性能与模拟值之间存在一些偏差，这就电路优化主要解决的问题，目前它的局限有很多，比如电路拓扑结构以及元件参数初值的选定多凭经验确定、电路优化可能陷入局部极小点导致失败和优化效率低等。对于实际的多变量优化问题，通过灵敏度分析可以决定优化变量的选择，减少计算工作量，提高优化效率。除此之外，对灵敏度的分析还可以预测随着元件

改变电路的性能变化，以及表达式线性化等等一些有利于电路设计和分析的辅助功能。因此如果能够快速准确的提供灵敏度信息，则会大大减轻模拟电路设计者的工作量。

1.1 电路分析方法

电路分析是电子电路工程的基础。一方面，它揭示了电路性能背后的复杂的机制，另一方面，设计者可以由此对电路进行建模，进而对电路的性能进行预测。电路分析主要研究电路行为特性和电路元件之间的依赖关系，主要有三种不同分析类型：量化分析、数值化分析和符号化分析。

量化分析主要从概念上定义输出和输入之间的关系，然后通过反馈信息帮助设计者在设计过程中判定合理的调整方向。

数值化分析指所有的输入信息，包括频率、导入模型的参数、输入幅度和相位都以数值的形式给出，分析结果也以数值出现，但是当任何一个电路参数发生改变时，需要通过重新执行来获得新的计算结果。因此数值型的电路仿真器虽然可以很好的验证电路设计，但很难预计电路参数改变后的性能情况。

符号化的分析方法是一种通过电路自变量（时间或频率），因变量（电压和电流）以及符号化的电路元件也形式化计算电路特性和行为的方法，最终分析的结果是符号化形式的公式，可以很明显找出决定电路行为和性能的参数。根据是否全部的电路参数以符号的形式出现，符号化分析结果有多种方式出现，如全符号化分析，即所有的电路参数都以相应的符号出现；半符号化分析，即在结果中，一部分电路参数以符号化的形式参与，另一部分以数值形式出现；限于频率的符号化分析，即频率 s 作为唯一的符号，其它全部以数值的形式出现；简化的符号化分析，即只有最重要的项会出现在最终的表达式中，其它的项被忽略。

符号化的分析方法在计算的精确度、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量化分析和数值化分析具有更大的优势【2】。

1.2 符号化电路仿真器的历史

伴随一些基本的符号化分析方法的提出，最早的符号化分析器出现在上世纪的 60 年代末期，作为最早被大家认识的 NASAP（1968）只是把复频域变量作为唯一的符号化参数， SNAP【3】则能够支持任意设定的符号化变量，随后出现的 NAPPE（1973）采用符号数值混合分析的方法，它将小部分的电路参数符号化，从而能够快速地分析规模较大的电路。但是由于低下的计算效率和不能够完全的满足电路设计者的需求，符号化仿真器没有在电路设计者中普遍使用，同时，在上世纪的 70 年代，缺乏对全定制集成电路的需求也导致符号化仿真器在应用效果上低于数值电路仿真器，随着 SPICE 的诞生，符号化电路分析的优势被 SPICE 准确和快速的求解大规模电路的特点所掩盖。上世纪的 80 年代，随着全定制模拟和混合信号集成电路需求的增加以及新的符号化分析方法学的提出，符号化电路分析方法开始焕发新的活力。SCAPP【5】采用了改进的节点分析方法（reduced MNA），这种方法能够大大的缩减节点矩阵，同时它实现了电路的层次化分析以便于求解大规模电路。通过将电路分块处理，它能够使得符号化的生成项随着电路的规模线性增长，而不是指数级的增长。之后的仿真器如 SCYMBAL【6】，SC【7】，SAPEC【8】等等在电路的可处理元件的范围和电路规模上都有改进和发展。SSPICE【9】已经可以处理晶体管级的电路而不仅仅是前面应用的电路级的电路，仿真器 ISSAAC【10】【11】可以获取简化的表达式，使人们可以更加迅速的了解电路元件和性能之间的联系，ASAP【12】【13】甚至提供零极点提取等一系列的信息。

在以上的这些工具中，最有效和最容易实现层次化分解以及其它应用信息提取的采用的方法是基于行列式的分析法和信号流图的方法。随着计算机性能的大幅度提高和符号化分析高效算法的发展，以及模拟集成电路设计对计算机辅助设计和设计自动化的迫切需求，符号化的分析方法在模拟电路的设计中逐渐开始发挥更重要的作用。

1.3 符号化分析方法

1.3.1 符号化分析方法的概念

传统意义上的符号化电路分析限于线性电路的频域分析，其中频率变量 s 出现在结果中。符号化分析主要是得到如下式所示的符号化传输函数的表达式：

$$H(s,X)=\frac{N(s,X)}{D(s,X)} \qquad X=[x_1,x_2,...,x_n] \qquad n \le n_{all} \tag{1-1}$$

表达式（1-1）是关于复数频域变量 s 的函数，变量 x_1 到 x_n 为电路元素变量，n 是电路元件变量的个数，n_{all} 是所有电路元件的个数。

对于一个完全符号化的电路，表达式中除了 s 其它全部由电路元件的符号组成。对于符号化和数值型的混合电路，一部分的电路元件符号为指定的数值，简化的符号化结果则为删除非重要项的表达式，下面给出一个电路的例子（图１），我们将依次给出上述三种形式的表达式。

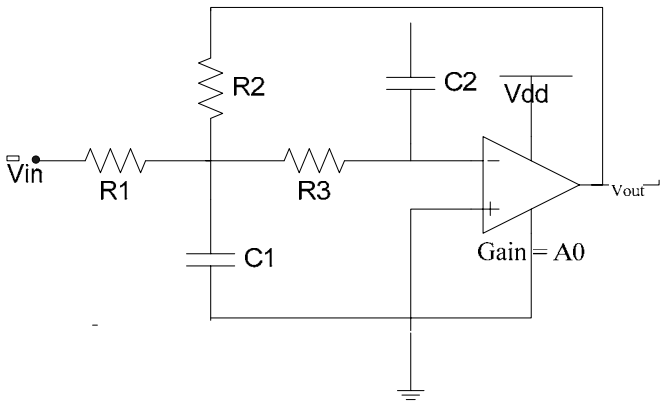


图1 符号化电路分析举例

Fig1 An Example for Symbolic Circuit Simulator

传输函数 $H(s)=Vout/Vin$ 的完全符号化表达式：

$$H(s)=\frac{V_{out}}{V_{in}}=\frac{-G_1G_3}{s^2[C_1C_2(1+1/A_0)]+s[C_2(G_1+G_2+G_3)(1+1/A_0)+C_1G_3/A_0]+G_2G_3(1+1/A_0)+G_1G_3/A_0}$$

(1-2)

谐振频率的半符号化表达式（保留放大器增益的符号）：

$$\omega^2 = (\frac{A_0 + 3}{A_0 + 1}) * (0.5 * 10^9) \tag{1-3}$$

假设放大器的增益 $A_0 >> 1$，则近似的传输函数表达式为：

$$H(s) = \frac{V_{out}}{V_{in}} = \frac{G_1 G_3}{s^2 C_1 C_2 + s[C_2(G_1 + G_2 + G_3)] + G_2 G_3} \tag{1-4}$$

在实际的符号化仿真器的处理中，如果电路出现了非线性元件或者是如图 1 所示的运算放大器，可选择不同的小信号线性模型进行转换再进行分析。

1.3.2 符号化分析方法的分类

一般来讲，符号化分析方法可以分为一下五种：

1）生成树枚举法

它可以分为两类：有向树枚举和无向树枚举。

有向树枚举法首先将初始电路稍作修改，建立一个符合分析算法的电路和与之相关的有向图，然后枚举图中所有的有向生成树。这些由导纳符号的乘积组成的生成树对应的项可以对应于节点导纳矩阵中的行列式和代数余因子，根据相应的对应关系得到符号化的传输函数表达式。对于一个有 n 个节点的电路，输入两端对应的节点分别为 1 和 n，输出两端对应节点分别为 2 和 n, 电路传输函数可以表示为如下的形式：

$$\begin{aligned} Zin &= \frac{V_1}{I_1} = \frac{\Delta_{11}}{\Delta} \\ \frac{V_o}{I_{in}} &= \frac{V_2}{I_1} = \frac{\Delta_{12}}{\Delta} \\ \frac{V_o}{V_{in}} &= \frac{V_2}{V_1} = \frac{\Delta_{12}}{\Delta_{11}} \end{aligned} \tag{1-5}$$

Δ 表示结点导纳矩阵 Y_n 的行列式，Δ_{ij} 表示 i 行 j 列元素对应的代数余子式。

对于任何的 RLCgm 电路，结点导纳矩阵的行列式等于相对于指定的根结点的所有

的有向生成树的导纳乘积的和。

无向树枚举法同时称为 2-图生成树枚举法，它首先将初始的电路图转化为两个无向图，分别为电压图和电流图，然后枚举两个图中所有共同的生成树。传输函数的表达式如 1-5 所示。结点导纳矩阵的行列式等于两个图中所有共同的生成树的导纳乘积的和，不过每一个乘积项要与它所对应的正负号相乘。即：

$$\Delta = \sum_{\text{共同的生成树}} \varepsilon_i * (\text{第i个相同的生成树导纳乘积项}) \tag{1-6}$$

ε_i 表示生成树对应的符号。

$$\varepsilon_i = |A_{Ii}| * |A_{Gi}| = \pm 1 \tag{1-7}$$

A_{Ii}，A_{Gi} 分别代表和第 i 个相同的生成树对应的电流图和电压图的约化入射矩阵。

生成树枚举法是最早的符号化分析方法，也是最早的符号化仿真器的基础。最初的仿真器只能处理 15 到 30 个节点的小电路，主要的原因在于产生的符号化生成项的个数随着电路的规模呈现指数级的增长，而且只能处理一种受控源—VCCS（电压控制电流源），而且这些方法也不能消除生成项相消的情况。随着新的算法的提出和近似等技术的应用，可处理电路的规模的类型都发生了极大的变化。

2) 符号流图法

它可以分为两种方法：Mason 信号流图和 Coates 图

两种方法都是根据明确定义的不同的法则来构建相应的信号流图和 Goates 图，然后分别应用 Mason 公式和 Coates 图对应的公式求解。

符号流图的分析方法是拓扑法进行符号化电路分析方法中被公认的最灵活、最有效的分析方法，但是该方法对于分析电路的规模还是有着比较大的限制。

3) 参数提取法

这种方法最适于分析电路中少数的元件以符号的形式出现，剩下的电路元件以数值代入计算，它的思想基于电路方程组对应行列式，它提供了从矩阵中提取符号化参数的机制，将矩阵分解为符号化参数的部分和数值化参数的部分，参数提取的规则依赖于符

号化分析矩阵的类型。

采用这种方法可以处理比完全符号化更大规模的电路，这种方法不仅仅限于代数方法分析电路，它还可以应用到拓扑方法。

4） 插入法

插入法最适合于频率变量 s 作为唯一的符号化变量，它基于电路的某些工作点的数值分析结果进行分析，这种方法需要计算不同频率点的行列式多项式系数的值，但是当电路超过 20 个结点时，利用频率变量的实数值计算会导致非常不精确的结果。

5）基于矩阵的方法

这种方法首先从电路描述中得到符号化的电路等式，然后将它们转化成为线性矩阵

$$Ax = b \tag{1-8}$$

A 代表 n*n 的符号化矩阵，x 是 n 个电路变量的向量，b 是包含 n 个元素的符号化的向量,n 是电路变量的数目，变量可以选择电压、电流、电荷、流量，进而对方程组进行求解。建立矩阵 A 的方法有很多种，最常采用的是改进的结点分析（MNA）以及在 MNA 上进行再次的改进。

1.3.3 符号化分析方法的优缺点

模拟电路有很大的设计空间，一方面，一种设计模块需要通过多次的电路设计图的修改才能够得到。另一方面，每一个电路设计图可以通过改变它的参数值来得到期望的性能规格。专业的设计者基于他们对电路知识的理解和掌握启发式的选择电路的拓扑结构和设计参数。然而，这些设计电路信息可以通过符号化的公式获取，而拓扑结构和设计参数的优化可以通过系统的评估这些公式的值获得。基于符号化分析的机构优化是通过不同的组合方式阻隔各个基础模块产生一系列交替的拓扑结构，针对于每一个产生的拓扑结构，和性能标准相关的公式可以自动的获取，同时，性能标准也可以通过重复的计算得到优化，电路的拓扑结构的优劣可以根据性能差异进行取舍。我们也可以采用相似的步骤进行电路元素尺寸的优化，因为符号化的表达式可以自动获取，设计这可以从中判别对影响电路性能的部分。除此之外，符号化分析还可以辅助模拟可测性分析和故

障检测上也具有明显的优势。

符号化分析同样也存在劣势，首先，目前提出的分析方法运算效率仍然比较低，而大多数的电路限于线性频域电路的分析，对于一些弱线性化或非线性化的电路没有提出很好的分析算法。其次，符号化分析的结果由一系列的符号化的生成项组成，对于符号化的分析方法来讲，这些生成项的数目随着电路规模的增加呈现指数增长，这些生成项的生成和存储都限制在计算机的运算和存储能力上，反之，由于生成项指数级的增长，使得使用于符号化分析的电路的尺寸也受到的限制。

1.4 符号化电路仿真器

一个符号化仿真器可以从电子电路中获得符号化的公式，并从这些公式中提取中要的重要的行为特征。不同的电路仿真器采用不同的输入和输出形式，大部分的符号化仿真器采用 SPICE 网表作为输入，处理的元件包括电阻、电容、电感、四种受控源、以及半导体器件和一些功能模块（如运算放大器），由于大多数的符号化分析方法都基于线性电路，对于电路中的非线性部分（如晶体管和运算放大器等）必须先转化满足精度的线性化的小信号等效模型，各种电路符号转化成频域上的导纳或阻抗的形式。包括简化功能的符号化仿真器时常根据经验给出一组电路元件的标称值进行近似，但这只能把近似公式的准确性限制到这个标称值附近的很小范围内，为改善这种情况，一些仿真器采用区间的概念进行近似分析。目前符号化仿真器的输出基本有两种形式，第一种称为顺次表达式(Sequence-of-expression, SOE)形式，这种形式中表达式以嵌套的形式表现以实现符号表示的重用，虽然他们的重复计算的效率较高，但一般比较难以揭示电路的性能；第二种形式是所谓的层次平展（Flat-nonhierarchical-expression, FNE）的形式，这种形式的表达式将 SOE 型中嵌套的子表达式展开，形成一个分子和分母中都存在很多公因子的表达式，其往往以关于频率变量 s 的多项式的形式出现。除了产生传输函数的公式，很多仿真器还提供了其它电路特征的符号化分析，如公式近似，参数不匹配处理，失真分析，零极点提取，参数灵敏度分析等等。

1.5 电路灵敏度分析

1.5.1 电路灵敏度的概念

网络理论是电子电路研究灵敏度理论的最早的一个分支，一般来讲，研究网络分析和综合的人们主要关注网络元素的精确度和这些元素的变动对这个网络的行为特性产生的影响。设计电路时，每一个电路的元件参数设定为一个标称值，而实际电路中的元件值都不可避免的存在误差。元件值得误差可能是在元件生产制造中所造成的，或者是由于温度变化、老化等环境条件变化引起的。元件参数值得误差必然会引起电路输出特性的变化对电路特性的影响，这些误差一般不能忽略，因此在设计电子电路时，研究电路中各元件参数的变化对电路特性的影响是很重要的问题。我们引入“ 灵敏度 ”的概念来表示这类变化关系的一种度量。

从原理上来讲，电路灵敏度理论和一般的动态系统灵敏度理论是一致的，但是这两个领域中的一个很重的区别在于研究方法的不同，电路灵敏度理论主要用于处理频域上的信息，而动态系统灵敏度理论在时域(time-domain)和频域(frequency-domain)上的信息都需要考虑。电路灵敏度理论由于它早期的发展和它的功能特性不同于系统理论而从系统灵敏度理论中分离出来称为一个单独的分支，信号流图在拓扑方法分析网络灵敏度上有很好的分析结果，但是我们之后采用的拓扑方法并不是基于信号流图。

灵敏度（sensitivity）是指网络函数对网络元器件参数的敏感程度。设 T 为网络函数，它可以使节点电位、支路电流等等；p 为网络中的元器件参数，如电导、电容、温度、晶体管模型参数等等；则“ 相对灵敏度定义 ”为：

$$S_p^T = \frac{\partial T}{\partial p}\frac{p}{T} = \frac{\partial T / T}{\partial p / p} \tag{1-9}$$

即网络函数 T 的相对变化量与网络参数 p 的相对变化量之比。此定义中假设变化量足够小。我们称（1-9）式定义的灵敏度为“ 归一化灵敏度 ”。当 T 或 p 为零时，就应采用“ 非归一化灵敏度 ”，它是网络函数 T 对网络参数 p 的偏导数，即

$$S_p^T = \frac{\partial T}{\partial p} \tag{1-10}$$

1.5.2 电路灵敏度的应用

在最早的电路灵敏度理论中，线性电路的灵敏度分析被延伸到零极点的灵敏度分析的角度，比如说，考察由于零点多项式系数的改变而导致零点的位置的改变，在【14】中给出了根轨迹变动处理的一般的方法。实际上，它引入了“大”参数变动的概念。但是，我们必须了解在等式 $a(s)+xb(s)=0$ 中，a 和 b 都是复频域变量 s 的多项式，x 作为唯一的变动的参数时是很容易考察根轨迹的。但是当可变参数边的更多时，处理变得复杂。【15】讨论了多输入系统的根轨迹。【16】给出了线性系统特征值和特征变量的求解理论。

在系统灵敏度的概念中，“大范围灵敏度”指当一个特定的系统的变量有一个较大的变动时，会导致系统的输出也有一个较大量的变动，用输出相对于原来的标准值的变动来测量系统的“大范围灵敏度”。这个概念来源类似于检测标准系统和受干扰系统之间的误差。如果因为系统变动引起的输入变动是在可允许的范围内的，则说明这个系统是鲁棒系统。理想情况是，系统的干扰对系统的输出没有任何的影响。一般来说，网络的“大范围灵敏度”分析将比普通系统要简化，因为网络分析中存在一些简单的关系，这可以简化“大范围灵敏度”分析。

在实际电路网络中往往存在有寄生参数，尤其是当电路工作在较高频率条件下时，寄生参数对电路性能的影响是不能忽略的，因此讨论寄生参数的灵敏度对电路分析有实际意义。假定在理想电路中寄生参数的标称值是 0，它对电路及电路方程本身不产生任何影响，但它的微小变化，有可能对电路性能产生很大影响。我们利用伴随网络法可以计算出寄生参数变化时，电路输出变量对寄生参数的灵敏度值。也就是说，在原网络中的一个零值的寄生参数可能有非零的灵敏度值。

模拟电路的灵敏度是指微小的电路参数值的变动对性能尺度的数学意义上的度量，在模拟电路中，灵敏度分析在确定关键设计变量值中起着至关重要的作用，在布局中，

反复电路尺寸的灵敏度分析可以去定出关键的妨碍电路性能的寄生部分。优化电路时，灵敏度分析可以用来获取良好的性能。

除此之外，对灵敏度的分析还可以预测随着元件改变电路的性能变化，以及表达式线性化等等一些有利于电路设计和分析的辅助功能。因此如果能够快速准确的提供灵敏度信息，则会大大提升模拟电路设计效率。

1.6 本文主要的研究内容

本文介绍了一种居于拓扑分析法的符号化电路仿真器（GRASS）【17】【18】的原理和实现， GRASS 是目前第一个可以成功的通过拓扑方法分析较大规模模拟测试电路(20-30 个晶体管)的符号化模拟电路仿真器。然而当代的符号化模拟电路仿真器不能限于提供传输函数的表达式，它应该包括更多的有利于电路分析的功能，我们在它的基础上开发了灵敏度的提取，并应用灵敏度信息来分析实际电路。因为灵敏度能够在数值和方向上给出准确的信息和调整的方向，避免了多次电路性能仿真，从而缩减设计的时间。

第二章 GRASS 的介绍

GRASS（Graph Reduction Analog Symbolic Simulator）的实现基于【17】【18】提出的一个完成的符号化模拟电路分析方法，【18】给出了有效的计算机实现算法。

【17】提出的理论是一种网络拓扑分析方法，该方法运算的对象是电路网络的图本身，运算过程中使用的符号是网络元件的符号参数，较以往的网络拓扑方法，此种分析方法消除了运算结果中相互抵消的生成项。

【17】讨论的符号化分析方法主要应用于符合特定前提条件的电路；分析前将满足可分析条件的电路按照某种图构建规则转化成相应的有向图，基于这些有向图应用符号化模拟电路的分析定理。最后通过枚举有向图中满足分析定理的生成树来得到电路特性符号化表达式中的生成项。

GRASS 的实现采用二分判定图【19】与图简约方法，其中使用哈希（hash）结构实现存储数据的共享，从而大大增加了可处理电路的规模。此仿真器可以求解线性或可转化成线性电路的频率响应，并且得到精确的展平的非层次形式的表达式。

2.1 可分析电路基本条件

【17】提出的符号化分析方法主要适用于符合特定前提条件的电路。前提条件如下：

1）电路只允许含有 5 类分析元件：阻抗（Z），导纳（Y），四种类型的受控源（压控电压源 VCVS， 流控电流源 CCCS，压控电流源 VCCS，流控电压源 CCVS），独立源和理想运算放大器。

2）电路中只含有一个独立源。如果含有多个输入源，需要对不同的独立源分别进行符号化分析，然后进行线性叠加。

3）每个控制源只控制一个受控源。

4）一个受控源只受一个控制源控制。

2.2 图构建规则

当待分析的电路满足上述 4 个条件时，可以将电路根据如下的图构建规则转化成向图。

图 2 是一个转化的例子，其从电路 2－a 到有向图 2－b。

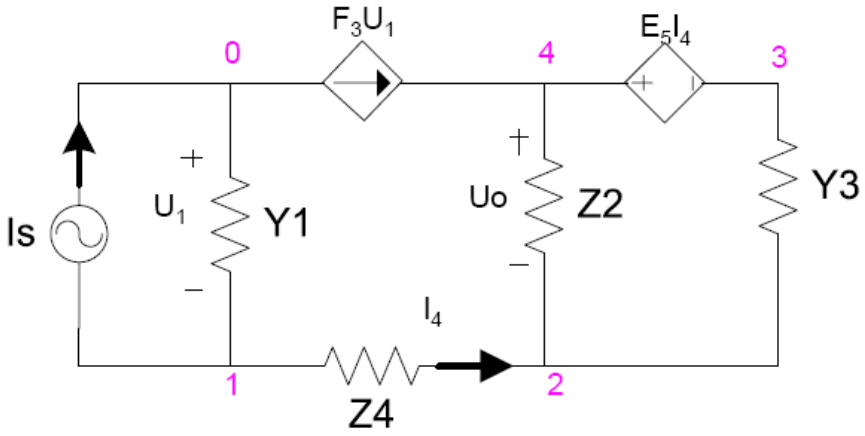


图 2-a 待分析电路

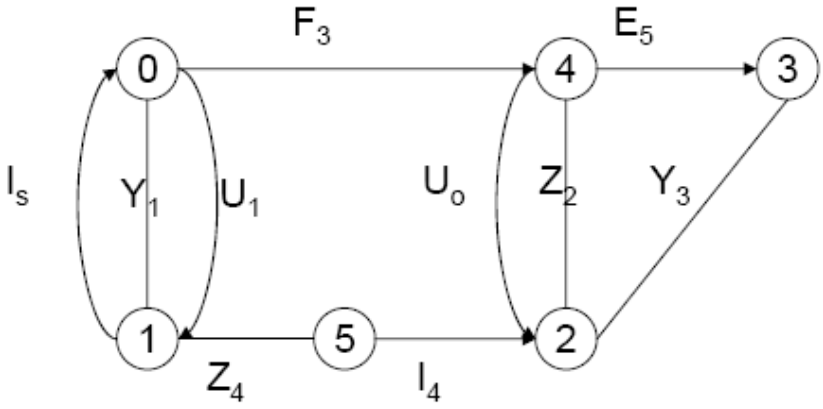


图 2-b 转换后的电路

图2 图构建规则转化举例

Fig.2 A circuit example

1） 所有的有向图中的边对应电路中的一个元件，每条边的权重为对应电路元件的名称，受控源为控制系数的符号。

2） 受控源或者独立源转化成有向边。电压源的方向从正极到负极，电流源的方向

沿着电流的流向，如 2-b 中的 F3 和 E5 对应的边

3） 电压控制支路添加一条有向边，例如 2-b 的 U1；对应电流控制支路添加一个家电和一条单独的有向边，如 2-b 中的节点 5 和 I4。

4） 输入和输出抽象成一个受控源，如 2-b 中的 Uo 和 Is，输入作为受控源，输出作为控制源，控制系数为 X。

5） 理想运算放大器用零器(Nullor)建模，每一个零器由一对零阻器(Nullator)和一个泛阻器(Norator)组成。

$H(s) = output / input$

由 $input = X * output$ 得到 $H(s) = 1/X$

所以输入输出所抽象的受控源的控制系数的倒数的表达式就是传输函数。

对于受控源，规定如下：

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

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

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

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

由上面可以看出，我们对控制源有着特别的要求，对于电压控制源，要求其流过的电流为零；对于电流控制源，要求其两端的电压为零。

2.3 符号化模拟电路分析定理

通过【17】提出的基于拓扑法的符号化模拟电路分析定理，对转化成有向图的电路进行分析，通过枚举有向图的生成树得到电路特性符号化表达式中的有效生成项。

值得注意的是，我们把所有的受控源的控制边和受控边分开分析，比如说把 CCVS 的 CC 代表的边和 VS 代表的边分开处理。但是 CCVS 依旧作为唯一的电路元件。

电路的拓扑法分析指通过枚举有向图的生成树来得到电路特性符号化表达式的生成项，但由于生成项出现相互抵消的的现象，会导致大量的冗余生成项，而【17】提出的

分析定理正是剔除了这些冗余项，保证了产生的生成项都是有效的。

*有效生成树存在定理：*有效生成树不包含 Nullor，不包含 VC 和 CS 的对应边，但是必须包含电路中所有的 CC 和 VS 的对应边。

*有效生成树对应生成项：*有效生成树对应的生成项由该生成树树边对应的元件符号的乘积组成。所有的 Y 和 Z-1 直接成为乘积项中的符号，所有的 CC 和 VS 的权重为 1，不出现在乘积项中。

需要注意的是，在所有的生成项中所有的阻抗以导纳的形式出现。

*有效生成树对存在定理：*有效生成树对由有效左生成树和有效右生成树组成。

1）所有的 Y 和 Z 边必须或同时出现在有效左、右生成树中，或在左、右生成树中均不出现。

2）所有的 NU 和 NO 边必须出现在每一对有效生成树对中，其中 NU 边出现在有效右生成树中, NO 边出现在有效左生成树边中。

3）所有的 CC 和 VS 边必须出现在有效生成树对中。CC 和 VS 边可以同时出现在有效左、右生成树中；CC 和 VS 边也可以成对分别出现在有效左、右生成树中，所谓成对出现就是 VS 边在有效左生成树中，其对应边在有效右生成树中，CC 边在有效右生成树中，其对应边在有效左生成树中。

4）所有的 VC 和 CS 边可以出现在有效生成树对中，也可以不出现。如果出现 VC 和 CS 边必须成对出现在有效生成树对中，所谓成对出现就是 CS 边在有效左生成树中，其对应边在有效右生成树中；VC 边在有效右生成树中，其对应边在有效左生成树中。

下面我们给出例子说明受控源元件的对应规则。

A） 流控电压源（CCVS），对应于有向图中的两条有向边，由有向生成树对存在定理的 3）看出，它们可以（CC，CC），（VS，VS），（VS，CC）三种选择方式，其中（a,b）表示 a 在有效左生成树中、b 在有效右生成树中。对于 CCCS，会有两种方式出现，一种方式是（CC，CC），（VS，VS），就是 CC 和 VS 边同时出现在左、右生成树中；另一种方式是（CC，VS），就是 VS 边在左生成树中，CC 出现在右生成树中。

B） 压控电流源 （VCCS），对应于有向图中的两条有向边，当考虑有效生成树对时，我们有（VC，VC），（CS，CS）和（CS，VC）三种组合。由有向生成树对存在定理的 4）可得，对于 VCCS 元件，（VC，VC）和（CS，CS）组合都是不合法的，只有（CS，VC）组合可以，也就是说要么 CS 出现在左生成树中、对应的 VC 出现在右生成树中；要么 VCCS 所对应的边都不出现在有效生成树对中。

C） 流控电流源（CCCS），对应于有向图中的两条有向边，当考虑有效生成树对时，我们有（CS，CS），（CC，CC）和（CS，CC）三种组合。由有向生成树对存在定理的 4）可得，对于 VCCS 元件，（CS，CS）组合都不合法的，也就是说要么 CS 不出现，（CC，CC）组合出现，也就是说 CC 边同时出现在左、右生成树中；要么（CS，CC）组合，也就是 CS 边出现在所左生出树，CC 边出现在右生成树。

D） 压控电压源（VCVS），对应于有向图中的两条有向边，当考虑有效生成树对时，我们有（VS，VS），（VC，VC）和（VS，VC）三种组合。由有向生成树对存在定理的 4）可得，对于 VCVS 元件，（VC，VC）组合都不合法的，也就是说要么 VC 不出现，（VS，VS）组合出现，也就是说 VS 边同时出现在左、右生成树中；要么（VS，VC）组合，也就是 VS 边出现在所左生出树，VC 边出现在右生成树。

值得一提的是，有效生成树其实是有效生成树对的一个特列，即左、右生成树完全一致。

由有效生成树对定理，我们可以得到一个生成树对中左、右生成树所含有边的类型。即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（不含 VC 和 NU 边）；右生成树只含有 Y，Z，VS，VC，CC 和 NU 类型的边（不含 CS 和 NO 边）。这对于我们进行计算机自动化算法的设计有很大的帮助，它规定了有效生成树所含边的类型，使我们在设计计算机算法时可以缩小运算的范围，提高运算效率。

*生成树对生成项定理：*由生成树对得到的生成项由下列四部分的乘积所得：

1） 生成项符号（+ 或者 -）；

2） 所有出现的 Y 和 Z 器件符号（Z 器件在最终表达式中以 Z-1 形式出现）；

3） 带符号的受控源增益。

对于在左、右生成树中均出现的 CC、VS 边均代表 1，故不影响生成项。

*受控源符号定理：*四种受控源（VCVS，CCCS，VCCS，CCVS）在生成项中不仅表示为符号化的增益，而且各自的符号也不同，具体如下：

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

只要 VS 边出现，增益前就要带上负号。

*生成树对生成项符号定理：*假设 A_L 和 A_R 为有效生成树对中表示左生成树和右生成树的约化入射矩阵(Reduced Incidence Matrix)。A_L 和 A_R 的行数相同（等于电路结点数减一）；它们的列树（生成树边数，也为结点数减一）也相同，并且按照如下的规则排列：若 Y、Z、CC、VS 边同时出现在左、右生成树中，则代表它们的列以相同列数分别出现在 A_L 和 A_R ；若两条边成对出现在 A_L 和 A_R ，且对应列数相同（控制边在 A_R 中，受控边在 A_L 中）。入射矩阵中所有边均是右向的，对应电路中的有向边（源器件对应边，VC、CS、VS、CC），其方向在矩阵中仍然保持不变；对应的无向边（Y、Z、NU、NO）则任意选择一个方向，但要保证在 A_L 和 A_R 中方向一致。有了上述定义后，生成树对所得的生成项符号就等于 A_L 和 A_R 行列式的乘积，即 $|A_L|*|A_R|(|A_L|、|A_R|=\pm 1)$

值得提出的是，约化入射矩阵的行代表每个电路结点（除去零点），其数目等于电路结点数减一；约化矩阵的列代表生成树的每条边，由生成树定义可知其边数等于电路结点数减一。所以约化入射矩阵为方阵，其维数等于电路结点数减一。约化入射矩阵为方阵，故其行列式存在，且行列式的值只可能为 1 或者-1（见[20]）。因此，代表有效生成

树对中左、右生成树的约化入射矩阵的行列式乘积只能为 1 或者-1，可以直接与组成生成项的其他部分相乘得到最终的结果。

*生成项加和定理：*所有生成项之和为零。（式 2-1）

它提出了如何获得最终计算结果的定理。由有向图构建规则可知，我们所要求的未知量）（电路传输函数）以受控源增益的形式出现；由前面的定理可知，该未知量是某些生成项的组成部分。由生成项加和定理可得关于未知量的一次方程，从而非常简单地得到结果（式 2-2）。

$$X\sum_{i=1}^{m1}t_i+\sum_{j=1}^{m2}T_j=0 \tag{2-1}$$

$$X=-\frac{\sum_{j=1}^{m2}T_j}{\sum_{i=1}^{m1}t_i} \tag{2-2}$$

由上述定理得到的生成项都是不可互相抵消的，即它们都是最终结果中必须出现的项，都不是冗余的，从而提高了分析的效率。

2.4 符号化模拟电路分析举例

我们给出一个例子来讲解上述的符号化分析过程。

我们分析 3 中的电路

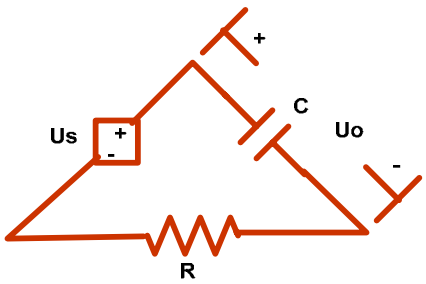


图3 拓扑分析法举例电路

Fig.3 A circuit for topological analysis

由上图可以看出它由一个电压源、一个电阻、一个电容串联而成，求电容两端的电

压值。由拉普拉斯变换，我们可以得到频域的传输函数为 $H(s)=\frac{1}{1+RCs}$。

由前面提到的分析方法分析可得，所有的电路元件符合我们的处理范围，按照图构建规则，我们可把 3 的分析电路转化成 4 的有向图。

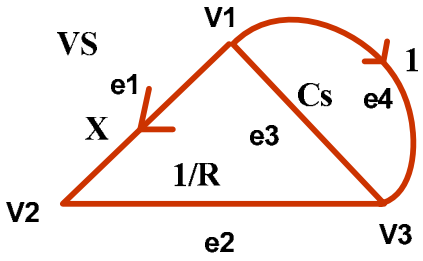
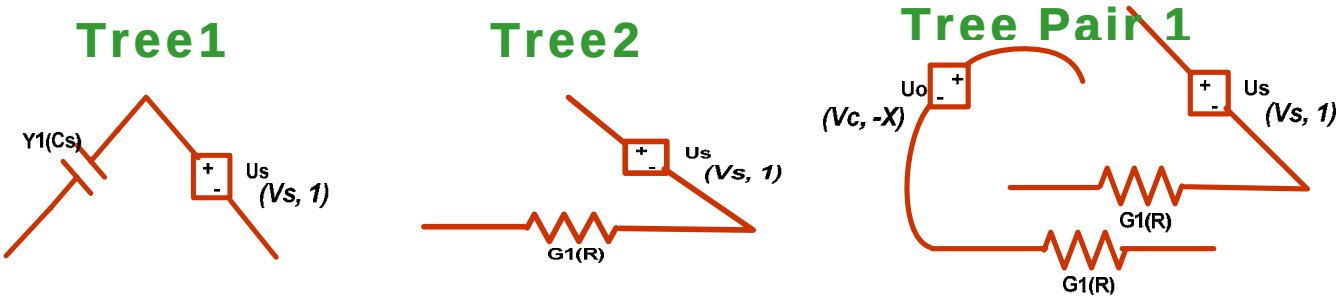


图4 待分析电路的对应有向图

Fig.4 Directed graph of the circuit

我们将输入输出看做一个压控电压源（VCVS），即输出源控制输入源。

根据分析定理，可以得到图 5 所示的两棵有效生出树和一对有效生成树对。



$Term1=Cs*1$ $Term2=1/R^{-1}*1$ $Term3=-1/R^{-1}*X$

图5 分析电路对应的有效生成树、生成树对、生成项

Fig.5 Admissible tree, admissible tree pair and corresponding product terms

两棵有效生成树和一对生成树对对应的生成项分别为：Cs*1，1/R*1 以及-1/R*X。

由式 2-1 得到 $Cs*1+1/R*1-1/R*X=0$

故得到传输函数 $H(s)=\frac{1}{1+RCs}$，和前面的计算结果是一致的。

2.5 符号化模拟电路算法分析

我们通过枚举生成树的方法得到传输函数的表达式，而生成项的个数与电路规模呈指数关系，当电路规模增大时，生成树枚举需要大量的运算时间和存储空间，【18】引入了二分判定图的数据结构，它的出现可以将一些原本需要指数复杂度计算和存储的问题转化为近似线性的结果。我们所引用的分析算法将生成树枚举变为有向图的约化过程，应用二分判定图的结构，在约化的过程中得到所有的有效生成树和生成树对，这个过程被称为有向图约化算法，而生成的二分判定图称为图约化判定图（Graph Reduction Decision Diagram, GRDD）。根据前面的分析方法，【18】提出计算机算法来实现分析的自动化。

下面我们首先给出有向图约化算法和生成项符号确定算法，然后在给出例子进行解释。

2.5.1 算法介绍

有向图约化算法：

Step1：初始化: 构建电路对应有向图，将该有向图拆分成左、右两个子图，分别删去两个子图中不允许存在的边。设定一个电路符号操作的顺序。

Step2：按照预设定的符号顺序对相应的边进行有向图约化（总是最先约化输入输出对应的符号）。按表格 1 所列举的操作选择或者排除电路元件符号所对应的有向图边。

Step3：检测是否满足终止条件：如果左、右子图均有生成树构成，则将判定图边指向终结点 1，然后转向执行 Step6；如果左、右子图均不可能得到生成树，则将判定图边指向终结点 0，然后转向执行 Step6；如前述两个条件均不满足，则继续执行 Step4。

Step4：依照生成项确定算法，分别确定判定图结点 1-分枝和 0-分枝对应的符号。

Step5：对约化子图进行同构测试，然后通过共享保证其唯一性。如果可以在已经生成的约化子图中可以找到同构的（包括相同）子图，则将对应的判定图分枝指向已生成约化子图对应的结点，同时删去新得到的约化子图；若否，则生成新的判定图结点，同时保存约化子图。

Step6：是否还有符号未处理？若是，则转向执行 Step2；否则，算法执行完毕。

表格1 不同类型元件边的约化操作

选取（Short）、移去（Open）

Table.1 Graph reduction manipulations

	选取元件		排除元件	
	左子图	右子图	左子图	右子图
VCVS	选择 VS	移去 VS 选择 VC	选择 VS	选择 VS 移去 VC
CCVS	选择 VS 移去 CC	选择 CC 移去 VS	选择 VS 选择 CC	选择 VS 选择 CC
VCCS	选择 CS	选择 VC	移去 CS	移去 VC
CCCS	选择 CS	选择 CC	选择 CC 移去 CS	选择 CC
Nullor	选择 NO	选择 NU	移去 NO	移去 NU
Y/Z	选择 Y/Z	选择 Y/Z	移去 Y/Z	移去 Y/Z

当约化过程结束时，我们构建了一个图结构。这是一个有向无环图，有一个根节点（X），两个终结点（0，1），所有的非结点均有两个分枝（实线 1-分枝，虚线 0-分枝）且均有一个代表符号，这正好是一个二份判定图。与二份判定图不同的是，该图的每条边都附带了一个符号。我们称这个由图约化而生成的二分判定结构为图约化判定图（GRDD），正是这个结构存储了所有的生成项。我们可以通过遍历该图来获得所有的生成项。遍历的规则是选择所有从根节点到终结点 1 的路径，若一个结点的 1-分枝在该路径上，则选取该结点所代表的符号，他们的乘积即位生成项的内容；所有该路径上所有分枝的符号的乘积，即为该生成项的符号。

生成项符号确定算法：

Step1：初始化：sign = 1。

Step2：如果要移除（Open）一条边，直接将该边从表示约化子图的数组中删去即可，sign 保持不变。如果要选择（Short）一条边（假设该边为（v1, v2），v1 和 v2 表示边的两个节点，其中 v1 < v2，即改变为正向），现将该边从表示约化子图的数组中删去（我

们把每一个约化子图存储在一个数组中，数组中包含了所有约化子图中的边，并每一条边的结点信息)，然后将数组中所有边结点为 v2 的点改成 v1。考察仍保存在数组中的结点，计算结点号小于 v2 的结点数目，如果该数为奇数，则 sign *= -1。

Step3：如果选择的边反向，则 sign *= -1。

Step4：如果选择的边类型为 VS，且 VS 与别的类型边成对出现，则 sign *= -1。

Step5：将 sign 关联到相应的 GRDD 结点分枝上。算法完毕。

2.5.2 算法举例

下面我们将根据图 3 的电路来说明如何通过有向图约化得到 GRDD，以及如何从 GRDD 得到电路的传输函数。约化的过程中，将有效生成树堪称由两棵相同的生成树组成的有效生成树对以方便统一处理。我们将按照前面给出的有向图约化算法进行。

A）初始化过程，由 2.4 中给出的例子，我们已经得通过图构建规则得到了 3 电路对应的有向图，将这个有向图拆分成左、右子图，并且删去两个子图中不允许存在的边。图 6 为由图 4 得到的两个子图，用于生成有效左、右生成树。

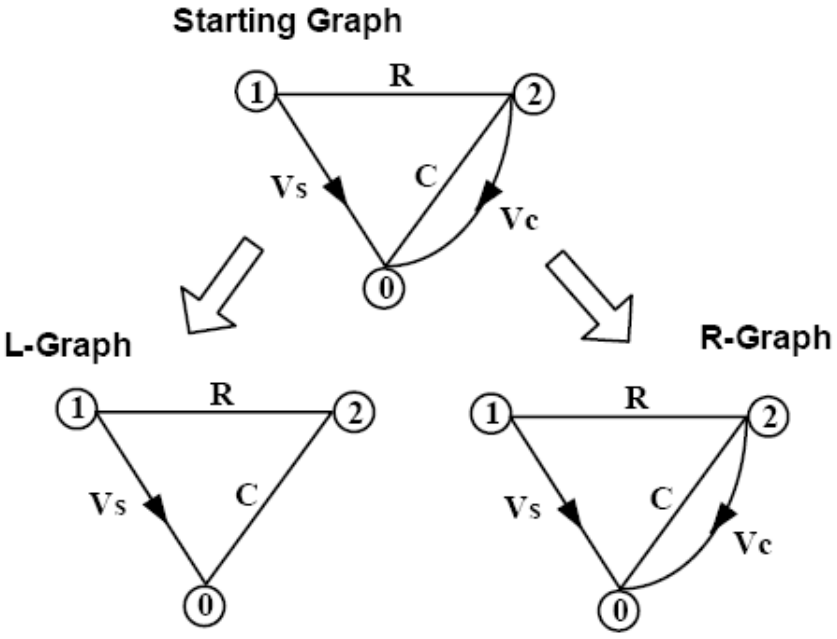


图6 有向图拆分（原始图、左图、右图）

Fig.6 Graph Splitting

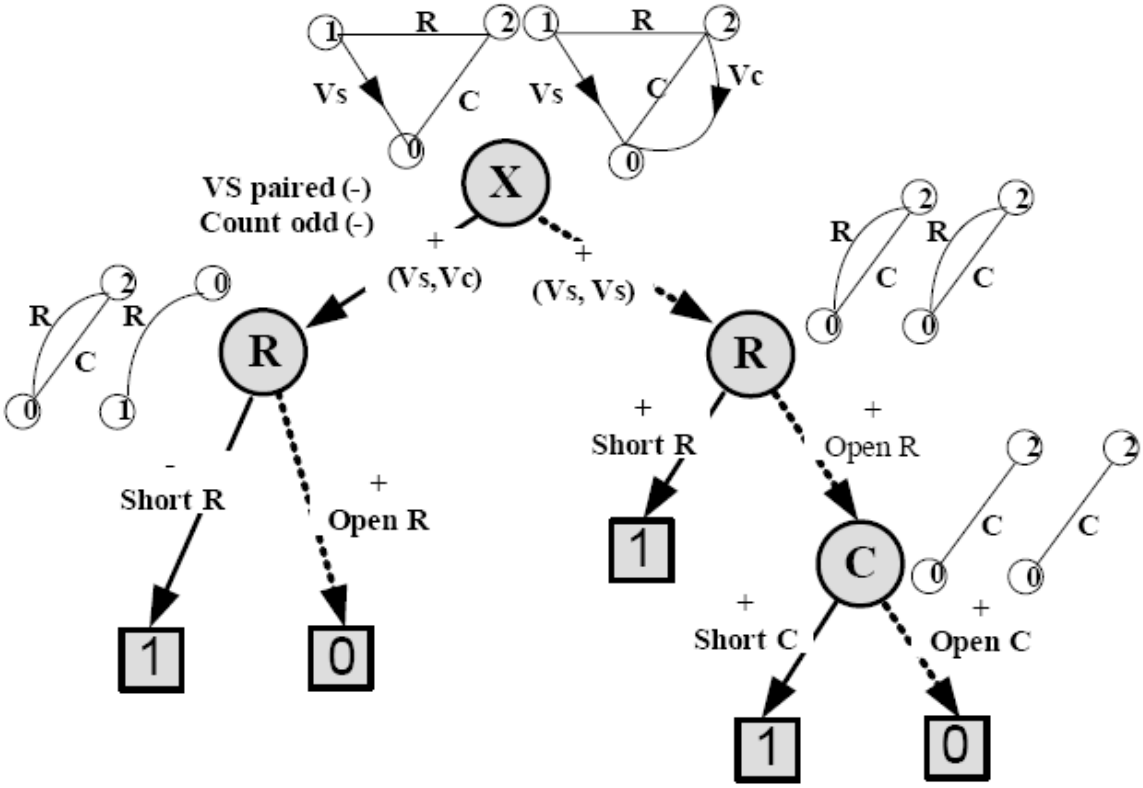


图7 有向图约化 GRDD 构建过程

Fig.7 Construction of GRDD by graph reduction manipulation

由前面的分析定理我们可以知道，对于有效生成树对的左、右生成树所含有边的类型是固定的，我们在处理前把不会出现的边从途中删去，如上图所示，VC 边将不在左图中出现，所以在左子图中删去 VC 边，右子图保持不变，而 Y、Z 和 VS 被允许出现的任意的两个途中，则在左右子图中保留。

因为我们的 GRDD 采用的是 BDD 数据结构，而简化有序的二分判定图是一个规范的数据结构，也就是说对于一个具体的问题，一旦确定一个转换的顺序，转换所得的结构是唯一的。所以我们在初始化的时候设定一个电路符号操作的顺序以保证生成结构的唯一性。在上述的电路图中有 3 个符号，电阻 R，电容 C 和输出控制输入的受控源增益 X。设定先处理 X，然后 R，最后处理 C。

B）我们开始按照预设定的符号顺序对，对相应的边进行有向图的约化。我们在进

行有向图约化的过程中构建一个二分判定图。该图有一个根结点，将拆分后的左、右子图与该结点关联，同时根结点代表符号 X,。然后我们对左、右子图进行约化操作，每一步约化都会得到两个新的有向图，我们构建一个新的判定图结点来与这两个新子图关联，同时用下一步要处理的符号来代表这个结点。这样按照预先设定的符号顺序对对应边进行处理直到所有边均处理完毕。我们处理有向图边，也就是处理相应的电路元件。对于生成项而言，元件或存在于一个生成项中，或不存在，这是一个二分的选择，所以我们使用二分判定图来记录整个约化的过程。在约化的同时，生成项也存储在了相应的判定图之中。由于该判定图是通过有向图约化的操作来建立的，所以我们称之为图约化判定图（Graph Reduction Decision Diagram，GRDD）。

通过图约化来得到生成树的思想来源于 G.Minty 的生成树枚举算法【21】。对于选择一个符号，相应的图约化操作就是选择该符号对应的边；而不选一个符号，相应的操作就是在图中移去相应的有向边。所谓选择一条边，在图操作中就是将该边的两个端点汇合成一个结点，为了计算的方便，我们总是将结点号大的点 Vmax 与结点号小的点 Vmin 汇合，也就是在移去边的同时将图中剩余边中端点为 Vmax 的点改为 Vmin；所谓排除一条边，只需要简单地将该边从有向图中删去即可。

在本例中，图 7 显示了整个有向图约化的过程，以及相应构成的 GRDD（图 8），。首先处理符号 X，X 对应输入输出受控源（VCVS）。选取 X，意味着 VC、VS 边在生成树对中成对出现，因此在左图中选取 VS 边、在右图中先移去 VS 边再选取 VC 边，这样得到位于根结点左侧的新结点，我们将新得到的约化子图与新结点相关联，，由于处理完符号 X 后接着处理的是符号 R，所以新结点以 R 表示，该新结点表示选取符号 X 的结果，我们用从结点 X 出发的实线箭头指向新结点。排除 X，意味着只有 VS 边作为公共边出现在在生成树对的两个生成树中，而 VC 并不出现，因此在左图中选取 VS 边、在右图中也选取 VS 边同时移去 VC 边，这样得到位于根结点右侧的新结点，这个结点与新的约化子图相关联，同时以符号 R 表示，该结点表示符号 X 不会出现在最终的生成项中，我们用从结点 X 出发的虚线箭头指向新结点。

C）然后我们检测处理完符号 X 的生成树对，发现左右子图均可以进一步的约化，

故不能直接将判定图直接指向终结点 1 或 0，进而依照生成项确定算法，分别确定判定图结点 X 的 1-分枝和 0-分枝的对应的符号。

如果我们通过约化来选择不同的电路符号，一旦发现左、右子图中都得到了生成树，那么我们就把相应的判定图箭头指向终结点 1；否则，若左、右子图进一步约化都不再可能得到生成树，则直接将箭头指向终结点 0。

D）对约化子图进行同构测试。如果可以在已经生成的约化子图中可以找到同构的（包括相同）子图，则将对应的判定图分枝指向已生成约化子图对应的结点，同时删去新得到的约化子图；若否，则生成新的判定图结点，同时保存约化子图。此例子中并没由发现此类图。

E）处理完符号 X 后，发现还有符号没有处理，接着处理符号 R，然后符号 C。如果已经没有未处理的符号，则此 GRDD 构建完毕。表格 1 列举了取舍各种类型边相对应的操作。

图 7 中最左面结点 R，当对其进行选择符号 R 的约化时（左、右子图选择 R 边），左、右子图都约化为一个结点（意味着得到生成树），所以结点 R 的 1-分枝（实线箭头）指向终结点 1；当进行排除符号 R 的约化时（左、右子图移去 R 边），右子图变成 2 个结点和 0 条边，产生孤立点，再无法得到生成树，所以该 R 结点的 0-分枝（虚线箭头）指向终结点 0。

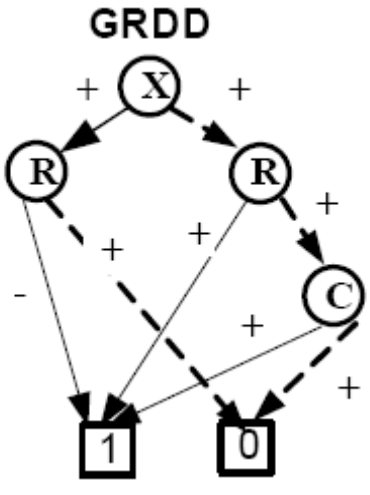


图8 分析电路的图约化判定图

Fig.8 Graph Reduction Decision Diagram(GRDD)

如图 8，我们可以得到 3 个生成项-XR-1，+R-1 和+Cs（电阻元件 R 作为阻抗以倒数形式出现），与我们先前的观察结果一致。根据生成项加和定理，我们就可以得出该电路的传输函数。

2.6 GRASS 结构和实现特点

2.6.1 符号化分析器结构

我们所基于的符号化仿真器 GRASS（Graph Reduction Analog Symbolic Simulator）和一般的符号化分析器结构一样，由三部分构成，包括网表解析器、符号化分析引擎和求值运算器。结构如图 9 所示：

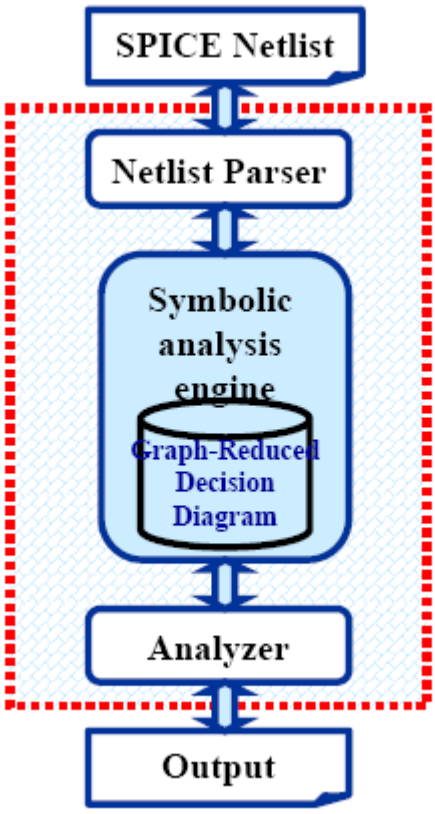


图9 符号化分析器基本结构

Fig.9 The Symbolic Circuit Simulator

仿真器的输入为 SPICE 网表，通过网表解析器获取电路信息。网表解析器由 PCCRTS（Purdue Compiler Compiler Tool Set）生成，它可以产生 C++代码的相关语法的解析器。经过网表解析器后，我们可以获得 SPICE 网表中描述的电路信息。

符号化分析引擎根据网表解析器提供的电路信息，建立电路对应的有向图，然后进行符号化分析，在内存中构建图约化判定图。

求值运算器可以提供各种数值分析结果，如符号化电路特性等。如果电路的结构不发生变化，GRDD 的结构不变，调制电路参数只需重新运行求职运算器。

2.6.2 GRASS 实现特点

GRASS 的实现采用二分判定图【19】与图简约方法，其中使用哈希（hash）结构实现存储数据的共享，从而大大增加了可处理电路的规模。共享时所有的二分判定图（BDD）相关问题共有的提升效率采用的方法，对于图约化判定图亦是如此。

在 GRDD 的构建过程中，每一步约化过程都会产生一对约化子图，如果新产生的约化子图与先前产生的约化子图相同，只是就可以将这两个相同结构的约化子图进行共享，即在内存中保留的一个副本。从图约化算法可以发现 GRDD 生成算法是以深度来构建 GRDD 的，因此一旦 GRDD 的节点可以共享，那么这个节点以下的所有节点都不需要再生成。

【18】介绍的方法是采用查找哈希表的方法来实现约化子图的共享，因为哈希表可以保证每一个存在其中的约化子图均是不同的，并且可以在比较短的时间内判断一个约化子图是否已经存在。分两步来实现约化子图的共享，哈希表是一组约化子图二分哈希树的集合，所谓二分哈希树就是使用二分查找树的方式来处理冲突的哈希表项。共享过程如下：

第一步：找出二分哈希树的入口（HashTreeID）。

第二步：如果第一步发生了冲突，即已经有约化子图的 HashTreeID 与目前使用的一

致，则建立二分哈希树，产生一个 hashID 用于寻找该约化子图的入口。

不同的约化子图可能产生相同的 hashID，所以我们要使用一个比较函数来判断两个具有相同 hashID 的约化子图是否具有相同的结构，如果它们的结构不同，则 hashID 将由指向描述图结构的数组的地址来代替（数组地址总是唯一的，故不会再产生冲突的问题）。

事实上，带有同构约化子图和项等价约化子图的节点也会引导相同结构的判定子图从而可以进行共享。它们的共享过程我们将不再一一介绍。

分析引擎中还有一些提升效率的侧率，例如并联元件的预处理和分离图的早期检测。

并联元件的预处理是将并联元件等价成一个元件，该元件的符号（数值）等于响应元件的并联值并联边的预处理可以很大程度上减少所需处理的元件个数，这对分析效率的提高有很大的帮助。

一旦一个图中不连通的两个分离部分，那么这张图就不会含有生成树。所以，如果在 GRDD 的构建过程中我们可以比较早地检测出约化子图含有分离的部分，那么一定不会有有效的生成树出现，因此可以直接将节点分枝指向终节点 0 从而结束该判定图分枝的构建。

在 GRASS 的符号化引擎中，被分析电路的数值频率响应可以很容易地从图约化判定图结构获得，只需把符号化的节点用相应频率点的数值量代替即可。一旦图约化判定图构建成功之后，如果电路的结构不发生变化，我们只需直接遍历图约化判定图就可得到电路的数值频率响应，每个图约化判定子图的数值响应值可以通过递归的方法得到，递归公式如下：

$$\begin{aligned} &Evaluate(vertex) = Evaluate(vertex->left)*V(symbol)*signL(vertex) \\ &+Evaluate(vertex->right)*signO(vertex) \end{aligned} \tag{2.3}$$

其中 $vertex$ 代表 GRDD 节点，$Evaluate(vertex)$ 代表节点 $vertex$ 引导的判定子图对应的数值响应，$vertex->left$ 代表节点 $vertex$ 1-分枝指向的节点，$vertex->right$ 代表节点 $vertex$ 0-分枝指向的节点，$signL$ 和 $signO$ 分别代表 1-分枝和 0-分枝上的符号，

$V(symbol)$ GRDD 节点 $vertex$ 的在某个频率点下所关联符号的数值值， 对于不同类型的符号，其对应的数值值如下：

$$\begin{aligned} &V(G) => G \\ &V(Z) => Z^{-1}(R->R^{-1}, L->(Ls)^{-1}->-j*(2*\pi*freq*L)^{-1} \\ &V(Y) => Y(C->Cs->j(2*\pi*freq*C)) \\ &V(Source) => Gain \end{aligned} \tag{2-4}$$

2.7 本章小结

本章介绍了一个基于拓扑分析法的符号化仿真器的分析原理、实现。该仿真器基于一个全新的拓扑网络分析理论，通过电路子图约化的算法在内存中构建二分判定图用以存储电路符号化表达式的所有有效生成项，然后通过对判定图的相关操作得到所需的符号化分析结果与相关数值结果。

它引用的算法基于严格的数学推理证明，通过采用二分判定图、子图和节点共享、判定图约化以及合理的符号分析顺序，构建比较清晰和有效的计算机算法，得以对较大规模的模拟电路（20-30 个传输晶体管）进行符号化的精确分析。数值分析的速度可以获得比 HSpice 仿真器更高的效率，算法的时、空复杂度都在一个比较理想的范围内。

但由于它只能得到精确的传输函数的信息，即只能观察频域中的频率响应，我们在它的基础上进行进一步的应用分析，即灵敏度的提取和应用。

第三章 符号化交流灵敏度分析

3.1 引言

模拟电路的灵敏度是指微小的电路参数值的变动对性能尺度的数学意义上的度量，在模拟电路中，灵敏度分析在确定关键设计变量值中起着至关重要的作用，在布局中，反复电路尺寸的灵敏度分析可以决定出关键的妨碍电路性能的寄生部分。优化电路时，灵敏度分析可以用来获取良好的性能。

符号化的灵敏度信息的提取可以消除数值化灵敏度分析工具（如 SPECTRE 和 SPICE）的不可洞察参数改变的缺点。符号化灵敏度分析是一种形式化的方法，它能够根据电路中符号化的电路参数和独立的频率变量得到电路性能的灵敏度，在符号化范围中，电路性能关于任意一个电路参数的一阶偏导数符号化灵敏度公式都可以精确的表示。但是一直都没有好的符号化分析方法有效的解决灵敏度计算的问题。因为灵敏度信息继承了顺序表达式的缺陷，即这些表达式中有大量的共享的因子。

随着层次化分解和顺序表达式概念的出现，大规模电路的符号化分析得到了极大的提升，同时对符号化表达式求导的灵敏度信息也得到了进一步的发展，

基于 DDD(determinant decision diagram）的符号化交流灵敏度提取算法在【4】提出，但是由于 DDD 是基于 MNA（modified nodal analysis）的方法，在 MNA 矩阵中的元素并不是代表电路中的某一个元件，它的映射机制使矩阵中的一个元素可能代表多个电路元件的关系，这可能使得灵敏度分析可能会依赖与 MNA 矩阵中的多个元素，利用链式求导法则进行求解，增加了分析方法的复杂度。

对于 GRASS，考虑到前面所述的算法分析，在图约化判定图中，如果不进行并联元件的预处理，电路元件和二分判定图中的结点是一一对应的，继而对应于生成项中的每一个元素。即使做了并联处理，由于并联元件都已加和的形式出现，并不影响求导的结果，所以我们依旧可以看做是电路元件和二分判定图结点一一对应的关系。我们将会在

图约化判定图的基础上进行灵敏度信息的提取。

3.2 行列式判定图（DDD）灵敏度的计算

下面我们将简单的介绍一种符号化灵敏度的计算方法，之后我们将会详细讲解我们的图约化算法灵敏度的计算，并且进行简单的比较。

3.2.1 行列式判定图 DDD

【4】提出了一种新的符号化分析方法来表达符号化的表达式，它主要把电路矩阵的符号化行列式表示成一张图的形式，称为行列式判定图 DDD（Determinant decision diagram），把对符号化行列式的计算转化称为对判定图的操作。使行列式的结果蕴含在 DDD 中。而 DDD 中节点的个数随着电路的规模呈现线性的增长，相较于随电路规模呈现指数增长的乘积项，大大减小了空间的复杂度。

它和我们应用的 GRDD 都借助于 BDD 的数据结构，不同的是，DDD 利用的是求解行列式的代数方法，而 GRDD 应用的是产生有效树的生成项的拓扑分析方法。

$$A=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \tag{3-1}$$

A 表示一个 n 行 n 列的方阵，a_{rc} 表示方阵中第 r 行和第 c 列的元素.

方阵 A 的行列式可以表示为如下的形式。

$$\det(A)=a_{rc}(-1)^{r+c}\det(A_{a_{rc}})+\det(A_{\overline{a}_{rc}}) \tag{3-2}$$

$(-1)^{r+c}\det(A_{a_{rc}})$ 表示元素 a_{rc} 的代数余子式。

$\det(A_{\overline{a}_{rc}})$ 表示将矩阵 A 中第 r 行第 c 的元素替换为 0 后的行列式的值，称为 remainder。

因此（3-2）可以看做一种二分的处理方法，选择 a_{rc} 和不选择 a_{rc}，分别应于上述的两种结果。

对于一个线性时不变的模拟电路而言，它的系统等式可以由改进的结点分析

（modified nodal analysis, MNA）的方法得到如下的方程组：

$$Tx = w \tag{3-3}$$

X 表示未知的参数向量，如结点电压和电流；电路矩阵 T 是一个 n*n 的大型稀疏符号化矩阵，一般来说，只有很少的非零元素。

模拟电路的符号化分析可以转化为求解式（3-3），即获取电路未知参数的闭合表达式，由克拉默（Gramer）法则得到，向量 X 中第 k 行的未知参数 x_k 可以表达为：

$$x_k = \frac{\sum_{i=1}^{n} w_i (-1)^{i+k} \det(T_{t_i,k})}{\det(T)} \tag{3-4}$$

$(-1)^{i+k} \det(T_{t_i,k})$ 是行列式 $\det(T)$ 中元素 $t_{i,k}$ 的代数余子式。

而传输函数可以表达为未知参数 x_k 和输入 w 比值，因此只要得到 x_k 的表达式，传输函数的表达式就迎刃而解。而解电路方阵的问题也转化为求解行列式的问题。

因为电路矩阵一般都是稀疏的，所以由行列式得到的符号化的表达式往往都可以共享很多子表达式，而二分判定图可以很好的表达这种的特性。

我们给出一个简单的例子来说明 DDD 的构建过程。

如一个行列式为如下的形式;

$$\det(M) = \begin{vmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{vmatrix} = adgj - abhi - aefj - bcgj + cbih \tag{3-5}$$

从上面我们可以看出，很多相同的因子存在不同的乘积项中，如 ad, gj, hi 。

由 BDD 的有序性，我们需要把行列式中 10 个不同的元素进行排序。元素选择的顺序直接影响这 DDD 结点的规模，【4】给出了一种对于阶梯电路网络的一种比较号的排序。在此处我们设定顺序为：$a > c > b > d > f > e > g > i > h > j$ 。

我们按照设定的顺序建立如下的行列式判定图：

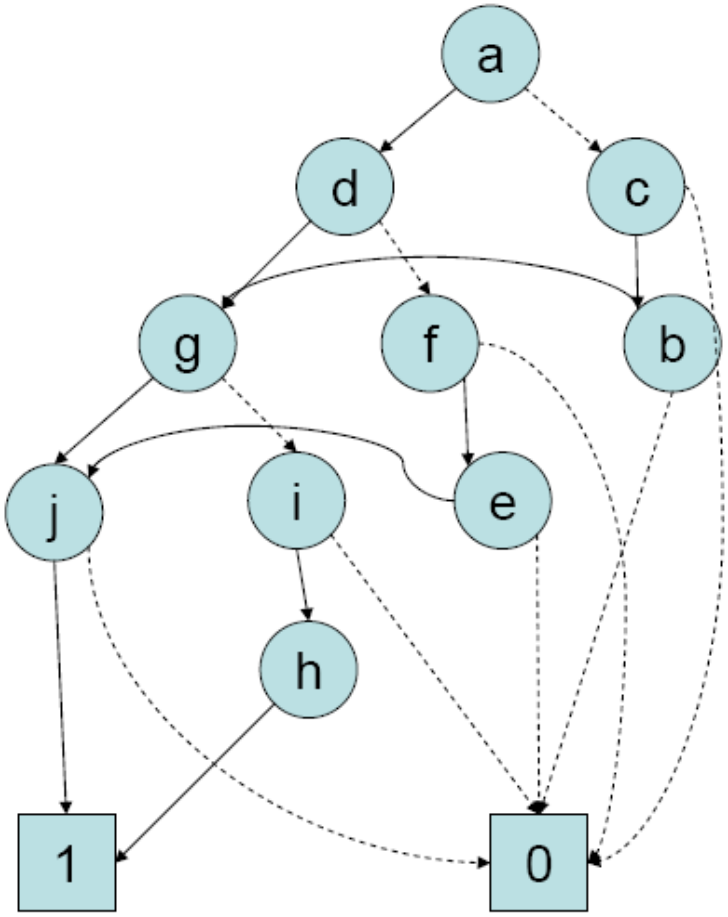


图10 行列式判定

Fig.10 Determinant Decision Diagram

我们把图 10 看做由一个迭代的应行列式展开公式（3.7-2）的生成的判定图，展开的顺序为 a,c b,d,f,e,g,I,h,j。每一个结点表达一个行列式的值，它的符号为待处理的一个元素，它的 1-分枝指向结点所表示元素的代数余子式的表达式，而 0-分支指向了 remainder。然后依次将行列式按照既定的顺序简化，直到行列式的值为 1 或 0，即指向终结点 1 或终结点 0。每一个 DDD 的结点关联一个符号，这个符号的产生【4】提出的算法产生。符号为 a_i 的值的结点代表的矩阵行列式 D 的值可以定义为如下形式迭代的得到：

1）如果结点是终结点 1，则 D = 1；

2）如果结点是终结点 0，则 D = 0；

3）如果结点为非终结点，$D = a_i s(a_i) D_{a_i} + D_{\bar{a}_i}$ 其中 $s(a_i)$ 为结点所关联的符号。D_{a_i} 和 $D_{\bar{a}_i}$ 分别为当前节点的 1-分枝和 0-分枝所指结点的行列式的值。

图 11 说明的图 10 的展开形式。

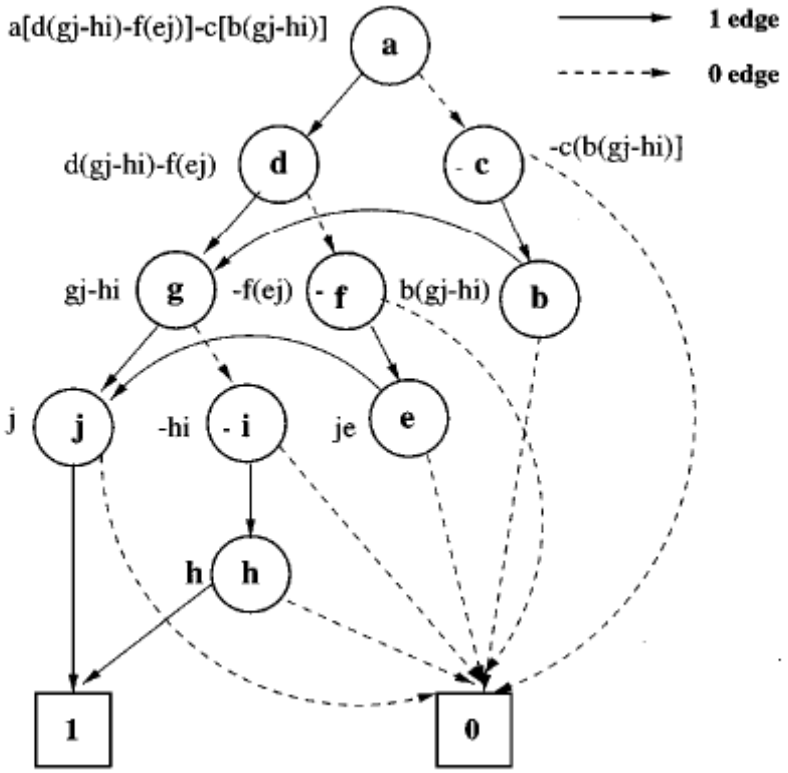


图11 矩阵（3-5）的行列式判定图

Fig.11 Determinant Decision Diagram for matrix (3-5)

而所有的起始于根结点的 1-路径的乘积项为此行列式的完全展开的符号化表达式。

3.2.2 DDD 的灵敏度求解

我们通过一个简单的例子来讲解如何运用基于 DDD 的代数余子式来计算电路传输函数的灵敏度。

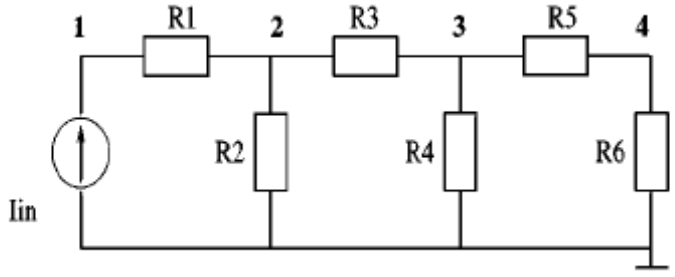


图12 灵敏度求解分析电路图

Fig.12 A circuit for computing sensitivity

描述上面的梯形电路的的等式如下：

$$\begin{pmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3} & -\frac{1}{R_3} & 0 \\ 0 & -\frac{1}{R_3} & \frac{1}{R_3}+\frac{1}{R_4}+\frac{1}{R_5} & -\frac{1}{R_5} \\ 0 & 0 & -\frac{1}{R_5} & \frac{1}{R_5}+\frac{1}{R_6} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} I_{in} \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3-6}$$

输入阻抗可以被定义为：$Z_{in} = \frac{v_1}{I_{in}}$ (3-7)

假设每一个矩阵的元素都别看做和其它不同的唯一的符号，电路矩阵的行列式则可以看做式（3-5）的形式。它的行列式判定图可以为 13 的形式，值得注意的是，在 13 中，终结点 0 以及和终结点 0 相关的分枝全部被移除。

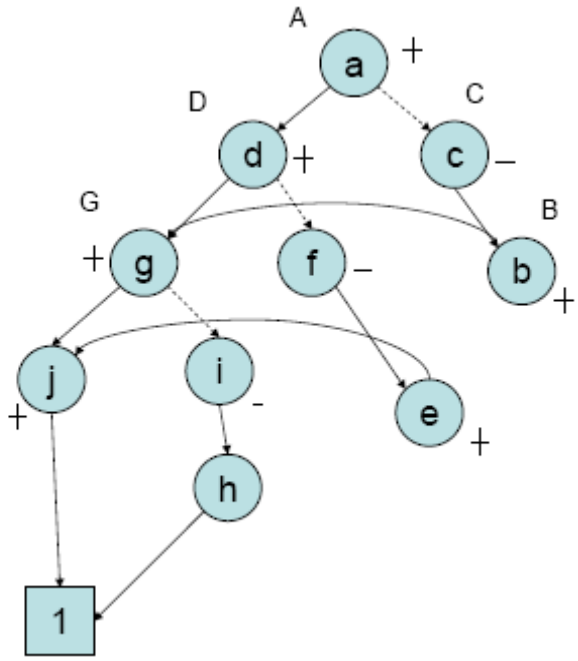


图13 移去 0 终点和相应边的行列式判定图

Fig.13 Determinant decision diagram

在 13 中，每一个结点被标记为一个小写的字母，我们用它们相应的大写字母来表示此结点的行列式，根结点代表电路矩阵的行列式 A。对应（3-6）和（3-5）中的矩阵元素，我们可以认为 $a=1/R_1, d=1/R_1+1/R_2+1/R_3$,由 Gramer 法则得到

$$v_1=\frac{I_{in}*Cofactor(A,a)}{A} \tag{3-8}$$

因此，

$$Z_{in}=\frac{Cofactor(A,a)}{A} \tag{3-9}$$

我们考查输入阻抗 Z_{in} 相对于电阻 R_2 的归一化灵敏度，即，

$$
\begin{aligned}
S_{R_2}^{Z_{in}} &= (\frac{R_2}{Z_{in}})(\frac{\partial Z_{in}}{\partial R_2}) \\
&= (\frac{R_2 A}{Cofactor(A,a)})\frac{\partial}{\partial d}(\frac{Cofactor(A,a)}{A})*(\frac{\partial d}{\partial R_2}) \\
&= (\frac{R_2 A}{Cofactor(A,a)})*(-\frac{Cofactor(A,a)}{A^2}\frac{\partial A}{\partial d}+\frac{1}{A}\frac{\partial Cofactor(A,a)}{\partial d})(-\frac{1}{R_2^2})
\end{aligned}
\tag{3-10}
$$

因为 $\frac{\partial A}{\partial d} = Cofactor(A,d)$，而且 $\frac{\partial Cofactor(A,a)}{\partial d} = Cofactor(Cofactor(A,a),d)$，

所以（3.7-10）可以化简为如下的形式：

$$
S_{R_2}^{Z_{in}} = (\frac{1}{R_2})(\frac{Cofactor(A,d)}{A}-\frac{Cofactor(Cofactor(A,a),d)}{Cofactor(A,a)}) \tag{3-11}
$$

3-11 里的三个代数余子式可以从 DDD 中求得。

因为电路矩阵的元素处理顺序已经确定，即 a>c>b>d>f>e>g>i>h>j，所以 Cofactor(A,a)就是根结点的左结点所代表的行列式的值，即 D，所以 Cofactor（A,a）直接指向符号为 d 的结点。同样，Cofactor((A,a),d)是结点 d 所代表的结点的左节点的行列式的值，为 G，所以 Cofactor((A,a),d)直接指向 G。

上面两个代数余子式的值可以直接从图中得到，而 Cofactor(A,d)则不能直接判断，因为行列式 A 不能直接和元素 d 相关联，我们将按照元素的处理顺序依次展开，直到待处理的结点的处理序号等于 d，然后返回。

最后得到的能表示各个代数余子式的判定图如下（图 14）：

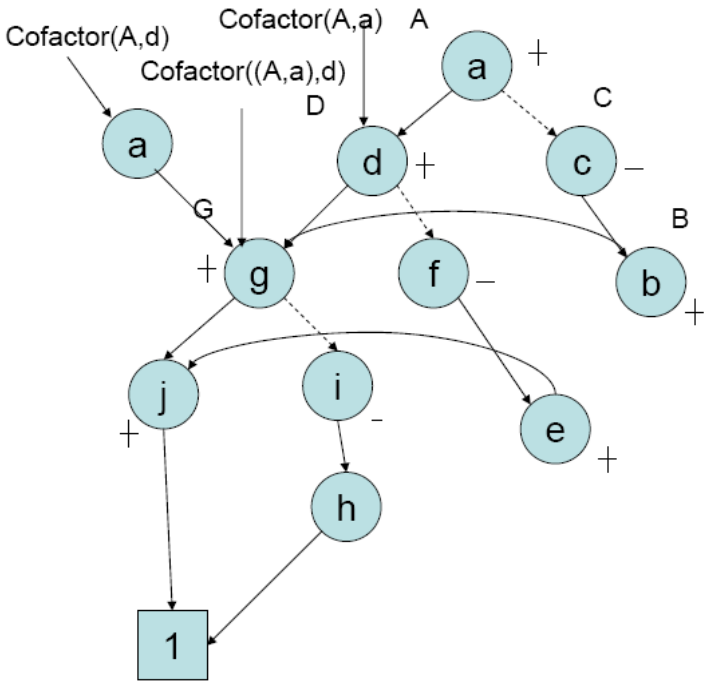


图14 基于行列式判定图的代数余子式

Fig.14 DDD-based derivation of cofactors

3.3 基于 GRDD 的符号化灵敏度

灵敏度分析主要集中应用于模拟电路设计中，假设 $H(s,x)$ 表示电路性能的传输函数，它相对于某一个电路参数 p（p 为电容、电感、电阻或受控源的控制系数）的归一化灵敏度定义为如下形式：

$$Sens(H(s,x),p) := \frac{p}{H(s,x)}\frac{\partial H(s,x)}{\partial p} = \frac{\partial \ln H(s,x)}{\partial \ln p} \tag{3-12}$$

绝对灵敏度为：

$$abs\_Sens(H(s,x),p) = \frac{\partial H(s,x)}{\partial p} \tag{3-13}$$

x 为电路中符号化参数向量。

一般的计算上述灵敏度信息的方法主要有网络干扰法【22】和邻接网络法【22】。

对于规模特别大的电路，灵敏度也可以从符号化的公式中计算出来，但是一般只是限于一个参数。

如果我们把公式 3-12 和 3-13 中的传输函数用 S-展开的形式表达，即

$$H(s,x)=\frac{N(s,x)}{D(s,x)}=\frac{\sum_{i=0}^{N}s^{i}f_{i}(x)}{\sum_{j=0}^{M}s^{j}g_{j}(x)} \tag{3-14}$$

所以归一化灵敏度公式（3-12）也可以表示为：

$$\begin{aligned}&Sens(H(s,x),p):=\frac{p}{H}\frac{\partial H}{\partial p}\\&=\frac{p}{H}\frac{\partial}{\partial p}(\frac{N}{D})=p(\frac{1}{N}\frac{\partial N}{\partial p}-\frac{1}{D}\frac{\partial D}{\partial p})\\&=\frac{p}{N}\sum_{i=0}^{N}\frac{\partial N}{\partial f_{i}}\frac{\partial f_{i}}{\partial p}-\frac{p}{D}\sum_{j=0}^{M}\frac{\partial D}{\partial g_{j}}\frac{\partial g_{j}}{\partial p}\\&=\frac{p}{N}\sum_{i=0}^{N}s^{i}\frac{\partial f_{i}}{\partial p}-\frac{p}{D}\sum_{j=0}^{M}s^{j}\frac{\partial g_{j}}{\partial p}\end{aligned} \tag{3-15}$$

由 3-14 可以得到：

$$Sens(H(s,x),p):=p(\frac{1}{N}\frac{\partial N}{\partial p}-\frac{1}{D}\frac{\partial D}{\partial p})=Sens(N(s,x),p)-Sens(D(s,x),p) \tag{3-16}$$

由式（3-16）知，我们可以传输函数对电路元素的归一化灵敏度转化为分子分母分别相对于电路元件的归一化灵敏度。

$$\begin{aligned}&abs\_Sens(H(s,x),p)=\frac{\partial H}{\partial p}=\frac{\partial}{\partial p}(\frac{N}{D})\\&=\frac{(-\partial \mathrm{N}/\partial \mathrm{p})\text{-}\mathrm{H}^{*}(\partial \mathrm{D}/\partial \mathrm{p})}{\mathrm{D}}=\frac{(-\partial \mathrm{N}/\partial \mathrm{p})\mathrm{D}+\mathrm{N}(\partial \mathrm{D}/\partial \mathrm{p})}{\mathrm{D}^{2}}\end{aligned} \tag{3-17}$$

由式（3-17）知，我们可以传输函数对电路元素的导数转化为分子分母分别相对于电路元件的绝对灵敏度和传输函数的分子分母的代数运算。

3.4 不同元件的归一化灵敏度公式

因为在 GRDD 中，所有除了控制参数，均以导纳的形式出现符号化表达式中。由于不同的电路参数按照不同的形式存储，分为以下的四种情况。

1）当电路元件为电阻时，设$G=1/R$，则

$$\frac{\partial H(s,x)}{\partial R}=\frac{\partial H(s,x)}{\partial G}*(-\frac{1}{R^2})=>\frac{R}{H(s,x)}\frac{\partial H(s,x)}{\partial R}=-\frac{\frac{1}{R}}{H(s,x)}\frac{\partial H(s,x)}{\partial G}$$

即 $Sens(H(s,x),R)=-Sens(H(s,x),G)$ (3-18)

在符号化分析定理中，$X=\frac{1}{H(s,x)}$

则

$$\frac{\partial X}{\partial p}=\frac{\partial X}{\partial H(s,x)}\frac{\partial H(s,x)}{\partial p}=>-\frac{1}{H^2(s,x)}\frac{\partial H(s,x)}{\partial p}=\frac{\partial X}{\partial p}$$

$$=>\frac{p}{H(s,x)}\frac{\partial H(s,x)}{\partial p}=\frac{p}{-\frac{1}{H(s,x)}}\frac{\partial X}{\partial p}$$

$$Sens(H(s,x),p)=-Sens(X,p)$$ (3-19)

有（3-12）和（3-13）得到

$$Sens(H,R)=Sens(X,G)=Sens(D(s,x),G)-Sens(N(s,x),G)$$ (3-20)

传输函数对电阻的归一化灵敏度则按照公式(3-20)计算。

2）当电路元件为电容时，

在 GRDD 的节点中，C 以 Cs 的形式存储，由

$$Sens(H(s,x),C)=\frac{C}{H(s,x)}\frac{\partial H(s,x)}{\partial C}=\frac{Cs}{H(s,x)}\frac{\partial H(s,x)}{\partial Cs}$$

得到

$$\begin{aligned} &Sens(H(s,x),C) = Sens(H(s,x),Cs) \\ &= Sens(N(s,x),Cs) - Sens(D(s,x),Cs) \end{aligned} \tag{3-21}$$

传输函数对电容的归一化灵敏度则按照公式(3-21)计算。

3）当电路元件为电感时，

在 GRDD 的结点中，L 以 1/Ls 的形式存储，由

$$\left.\begin{aligned} &Sens(H(s,x),L) = Sens(H(s,x),Ls) \\ &Sens(H(s,x),Ls) = Sens(X,1/Ls) \end{aligned}\right\}$$

$$=> Sens(H(s,x),L) = Sens(D(s,x),\frac{1}{Ls}) - Sens(N(s,x),\frac{1}{Ls}) \tag{3-22}$$

传输函数对电感的归一化灵敏度按照公式（3-22）计算。

4）选择参数为受控源的控制系数时，传输函数对其的归一化灵敏度按照公式（3-16）计算。

3.5 频率响应相对于电路元件的灵敏度计算

对于传输函数，我们一般考查它的幅度和相位的变化情况。如果我们能够得到幅度和相位随着某一个参数的变化趋势，对于设计电路将会极大的帮助。下面，我们简单推导了幅度和相位相对于某一电路参数的绝对灵敏度和相对灵敏度的计算公式。

3.5.1 归一化灵敏度

由公式 3-23

$$H(s,x) = |H(s,x)|e^{j\angle H(s,x)} \tag{3-23}$$

知

$$\begin{aligned} &Sens(H(s,x),p) \\ &= \frac{\partial \ln(|H(s,x)|e^{j\angle H(s,x)})}{\partial \ln p} = \frac{\partial \ln(|H(s,x)|)}{\partial \ln p} + \frac{\partial \ln(e^{j\angle H(s,x)})}{\partial \ln p} \\ &= \frac{\partial \ln(|H(s,x)|)}{\partial \ln p} + j\frac{\partial \angle H(s,x)}{\partial \ln p} \\ &= Sens(|H(s,x)|,p) + j\angle H(s,x)Sens(\angle H(s,x),p) \end{aligned} \tag{3-24}$$

由等式（3-24）得知，

幅度相对于电路元件的归一化灵敏度为传输函数相对于电路元件的归一化灵敏度的实部，即：

$$Sens(|H(s,x)|,p)=\mathrm{Re}(Sens(H(s,x),p)) \tag{3-25}$$

相位相对于电路元件的归一化灵敏度为传输函数相对于电路元件的归一化灵敏度的实部，即：

$$Sens(\angle H(s,x),p)=\frac{1}{\angle H(s,x)}\mathrm{Im}(Sens(H(s,x),p)) \tag{3-26}$$

3.5.2 绝对灵敏度

设 $H^*(s,x)$ 为 $H(s,x)$ 的共轭，

由 $|H(s,x)|^2=H(s,x)*H^*(s,x)$ 得知，

$$abs\_Sens(|H(s,x)|,p)=\frac{\partial|H(s,x)|}{\partial p}=\frac{\frac{\partial H(s,x)}{\partial p}H^*(s,x)+\frac{\partial H^*(s,x)}{\partial p}H(s,x)}{2|H(s,x)|} \tag{3-27}$$

而且

$$\frac{\partial H^*(s,x)}{\partial p}=\frac{\partial}{\partial p}(\frac{N^*}{D^*})$$
$$=\frac{(-\partial N^*/\partial p)D^*+N^*(\partial D^*/\partial \mathrm{p})}{(D^*)^2} \tag{3-28}$$

我们知道在 GRASS 中，表达式中，传输函数的分子和分母都是多项式乘积加和的形式，所以

$$\partial N^*/\partial p=(\partial N/\partial p)^*,\partial D^*/\partial p=(\partial D/\partial p)^*$$

所以由 3-17，3-27 和 3-28 得到幅度对电路元件的绝对灵敏度 $abs\_Sens(|H(s,x)|,p)$.

因为 $s=jw$，

由

$$H(jw) = |H(jw)|e^{j\theta(w)}$$
$$\partial H(jw)/\partial p = (\partial |H(w)|/\partial p) * e^{j\theta(w)} + |H(w)|e^{j\theta(w)}(j*\partial\theta(w)/\partial p) \quad (3\text{-}29)$$

得到

$$abs\_Sens(\theta(w), p)\partial\theta(w)/\partial p = \frac{\partial H(w)/\partial p - (\partial |H(w)|/\partial p) * e^{j\theta(w)}}{|H(w)|e^{j\theta(w)} * j}$$
$$= \frac{\partial H(w)/\partial p - (\partial |H(w)|/\partial p) * H(w)/|H(jw)|}{jH(w)} \quad (3\text{-}30)$$

由 3-29，3-30 得到相位相对于电路元件的绝对灵敏度 $abs\_Sens(\theta(w), p)$。

3.6 归一化灵敏度的求解步骤

下面我们通过两个简单的例子来说明符号化交流归一化灵敏度的求解步骤，以及如何从 GRDD 中获取电路的灵敏度信息。分析电路如下图所示：

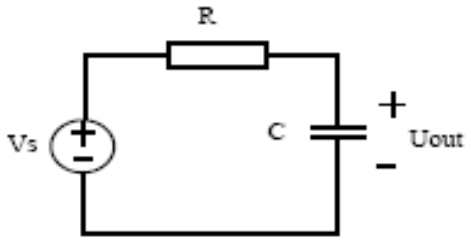


图15 求解灵敏度的分析电路

Fig.15 A circuit for sensitivity analysis

图 15 由一个电压源，一个电阻和一个电容串联而成，首先求解输出 U_{out} 对输入 V_s 的传输函数，然后再求解传输函数相对于电阻 R 的归一化灵敏度。

由 Kirchoff 定理可以非常的得到频域的输出 U_{out} 对输入 V_s 的传输函数：

$$H(s) = \frac{U_{out}}{V_s} = \frac{1}{1+RCs} \quad (3\text{-}31)$$

由式 3-31 计算得到传输函数相对于电阻 R 的归一化灵敏度为

$$Sens(H(s),R) = -\frac{RCs}{1+RCs} \tag{3-32}$$

显然，当 H（s）已知的时候，归一化灵敏度的符号化表达式的将依赖于 $\partial H(s)/\partial p$ 的表示，因为我们在第二章中得知 GRASS 已经能够把传输函数的符号化表达式精确的计算出来，因此我们的任务就是从传输函数中求得 $\partial H(s)/\partial p$ 的表达式。

由第二章的 GRASS 实现中，我们可以知道，上述电路可以表示成如下的形式如下的符号化判定图 SDD(Symbolic Decision Diagram)的形式（图 16）

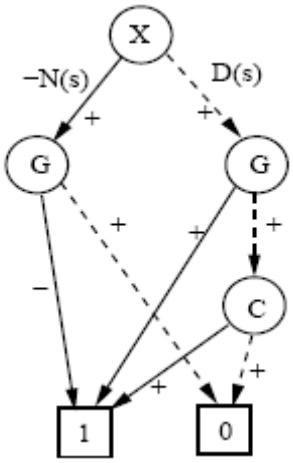


图16 待分析电路的 SDD 表示

Fig.16 Symbolic Decision Diagram

输入输出看做一个 VCVS（输出电压控制输入电压），控制系数为 X，然后按照先 X，再电阻 R，随后电容 C 的顺序建立符号化判定图。每一个起始于根节点的 1-edge 都是分析电路的一个有效生成树对应的生成项。由符号化分析定理中的的有效生成树生成项加和定理得到：

$$G + Cs + X(s)G = 0 \tag{3-33}$$

求解式（3-33）得到：

$$H(s) = \frac{1}{X(s)} = \frac{G}{G+Cs} \tag{3-34}$$

因为$G=1/R$

所以式（3-31）和式（3-34）是一致的。

而传输函数可以表达为如下的形式：

$$H(s)=\frac{N(s)}{D(s)} \tag{3-35}$$

所以

$$D(s)-X(s)N(s)=0 \tag{3-36}$$

由 BDD 的求值得到，D（s）就是根结点 X 的右结点的表达式，而$-N(s)$则为根结点 X 的左结点的表达式。

由图 16 得到：

$$\begin{aligned}&-N(s)=-G\\&D(s)=G+Cs\end{aligned} \tag{3-37}$$

所以

$$Sens(N(s),G)=\frac{G}{N(s)}\frac{\partial N(s)}{\partial G}=\frac{G}{G}\frac{\partial(G)}{\partial G}=1 \tag{3-38}$$

$$Sens(D(s),G)=\frac{G}{D(s)}\frac{\partial D(s)}{\partial G}=\frac{G}{G+Cs}\frac{\partial(G+Cs)}{\partial G}=\frac{G}{G+Cs} \tag{3-39}$$

由 3-38 和 3-39 得到：

$$Sens(X(s),G)=Sens(D(s),G)-Sens(N(s),G)=-\frac{Cs}{G+Cs} \tag{3-40}$$

有公式（3-20）得到

$$Sens(H,R)=Sens(X,G)=-\frac{Cs}{G+Cs} \tag{3-41}$$

显然式 3-32 和 3-41 是一致的。

因为在 BDD（Binary Decision Diagram）中，每一个 1-path 都可以看做是一个乘积

项，我们把导数也表示在 SDD 中。假设我们把初始的 SDD 根的左结点表示为$-N(s)$相对于某一个电路元件的归一化灵敏度，而初始的 SDD 的根的右结点表示为 D（s）的先归于某一个电路元件的归一化灵敏度，则初始的的 SDD 则转化为如下的表达形式(如图 17 所示)：

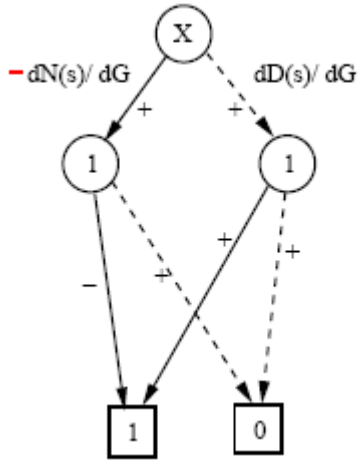


图17 待分析电路的灵敏度表示图

Fig.17 Symbolic Decision Diagram for Sensitivity Analysis

因为$-\frac{\partial N(s)}{\partial G}=1$，所以将 X 的左结点代表 G 的电路符号改为 1，并且将它的 1-edge 直接指向 1，终结点 1，0-edge 直接指向终结点 0。

而$\frac{\partial D(s)}{\partial G}=\frac{\partial(G+Cs)}{\partial G}=1$，所以将 X 的左结点代表 G 的电路符号改为 1，并且将它的 1-edge 直接指向终结点 1，0-edge 直接指向终结点 0。

值得注意的是，原本 SDD 的各个指向边所关联的符号不要改变，6-c 的根结点并不满足 BDD 的计算规则，即不能认为$-\frac{\partial N(s)}{\partial G}*X(s)+\frac{\partial D(s)}{\partial G}=0$。

由图 17 我们可以看到，灵敏度的表示图只会比原来的传输函数的符号化表示图更加简单，而不会额外增加空间的复杂度。

3.7 符号化交流灵敏度的具体实现

上面讲述了符号化交流归一化灵敏度的求解过程，并给出了一个灵敏度的表示图，接下来，我们将逐一分析它的具体实现过程。

从 16 的分析电路的例子中发现，因为 GRASS 已经可以实现电路的传输函数的表达式，所以，接下来的任务就是如何图约化判定图中把灵敏度信息提取出来。

3.7.1 符号化求导原理

传输函数可以表示为两个多项式相比的形式，而分子和分母能够从 GRASS 的符号化判定图中精确的表示出来，所以符号化交流灵敏度的求解过程最终转化为分别对分子和分母求偏导的过程。

由 GRASS 求得的传输的函数的分子和分母的表达式分别都是以 SOP（sum-of-products）的形式出现的。即分子和分母的每一个加和项分别是从根结点的左结点和右结点到终结点 1 的 1-edge【4】上的元器件符号的乘积，所以可以表示成如下的形式，假设待求导的一个参数为 p，值得注意的是，从符号化的分析定理中得知，在有效生成树对应的生成项中，所有的参数均以导纳的形式出现，而且，

1）当 p 为电阻 R 时，生成项中对应为电导 $p = G = \frac{1}{R}$。

2）当 p 为电容 C 时，生成项中对应于导纳 $p = Cs$。

3）当 p 为电感 L 时，生成项中对应与导纳 $p = \frac{1}{Ls}$。

4）当 p 为控制系数 Gain 时，生成项中依旧出现 $p = Gain$。

如果 $H(s) = \frac{N(s)}{D(s)}$，则分子分母分都可以表示为：

$$Q(s) = \sum_{i=0}^{sum1} f(i)p + \sum_{j=0}^{sum2} g(j) \qquad (3\text{-}42)$$

$\sum_{i=0}^{sum1} f(i)p$ 表示 N（s）或者 D（s）中含有参数 p 的乘积项的表达式，sum1 表示 N（s）或者 D（s）中含有参数 p 的表达式的个数；$\sum_{j=0}^{sum2} g(j)$ 表示 N（s）或 D(s)中不含有参数 p 的乘积项的表达式，sum2 表示 N（s）或者 D（s）中不含有参数 p 的表达式的个数。而且在所有的乘积项中，p 只有一次项和 0 此项，而不会出现高次项。

当 p 为电容（$p = Cs$）和控制系数（$p = Gain$）时，

$$\frac{\partial Q(s)}{\partial (Cs)} = \sum_{i=0}^{sum1} f(i) \tag{3-43}$$

当 p 控制系数（$p = Gain$）时，

$$\frac{\partial Q(s)}{\partial (Gain)} = \sum_{i=0}^{sum1} f(i) \tag{3-44}$$

当 p 为电感（$p = \frac{1}{Ls}$）时，

$$\frac{\partial Q(s)}{\partial (Ls)} = -\frac{\sum_{i=0}^{sum1} f(i)}{p^2} \tag{3-45}$$

当 p 为电阻（$p = G = \frac{1}{R}$）时，

$$\frac{\partial Q(s)}{\partial R} = -\frac{\sum_{i=0}^{sum1} f(i)}{p^2} \tag{3-46}$$

3.7.2 符号化求导步骤

我们给出一个例子来说明如果结合电路的约化判定图 GRDD 计算分子和分母相对于某一个电路元件的导数。电路图如下（图 18）

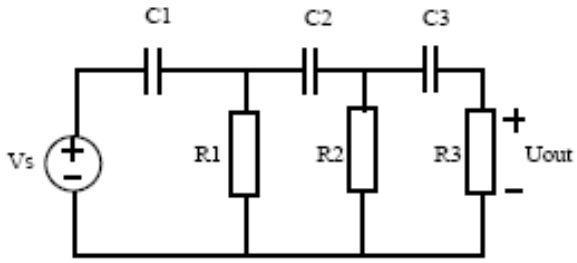


图18 待分析电路图

Fig18 A circuit for sensitivity analysis

由符号化模拟电路的分析算法，我们可以得到如下的符号化判定图（图 19）。

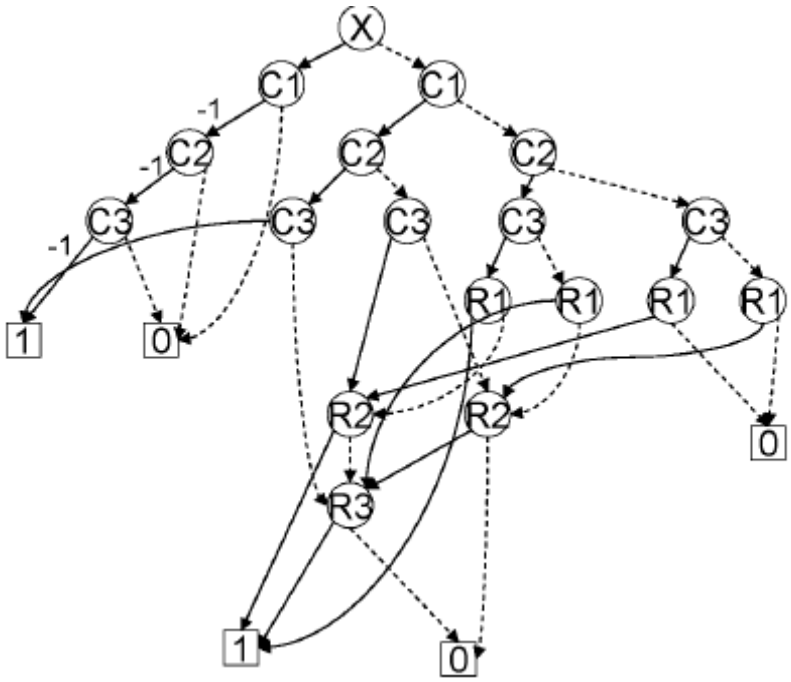


图19 待分析电路的符号化判定图

Fig.19 Symbolic Decision Diagram

观察符号化判定图，我们发现，相同排列序号的元素会分布在同一个层次上，如图 19 所示，电容 C1 全部分布在第 1 层，电容 C2 全部分布在第 2 层，电容 C3 全部分布在第 3 层，依次类推。这可以方面我们迅速判定带求导参数的位置。

由符号化分析定理，我们可以得到图 19 的传输函数的分子和分母的表达式的值分别为：

分子--$+C1C2C3$ （3-47）

$$C1C2C3s^3+C1C2G3s^2+C1C3G2s^2+C1C3G3s^2+$$
$$C1G2G3s+C2C3G1s^2+C2C3G2s^2+C2G1G3s+$$
$$C2G2G3s+C3G1G2s+G1G2G3$$

分母--$C1C2C3s^3+(C1C2G3+C1C3G2+C1C3G3+C2C3G1+C2C3G2)s^2$ （3-48）
$$+(C1G2G3+C2G1G3+C2G2G3+C3G1G2)s+G1G2G3$$
$$s+$$

在图 19 中，根结点 X 的左左结点代表传输函数的分子表达式的负值，而根结点 X 的右节点代表传输函数的分母的符号化表达式。

按照前面提出的分析步骤，我们将需要求解传输分子和分母分别对某一个参数的导数的符号化表达式。假设我们想要得到传输函数对电阻 R1 的导数。求导后的符号化判定图如图 20 所示：

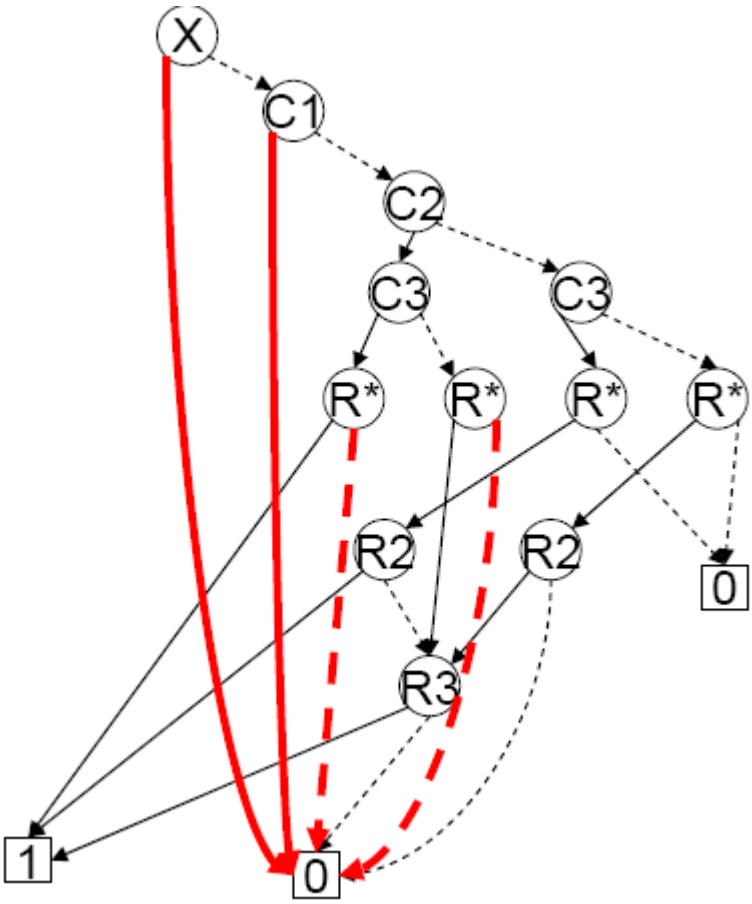


图20 待分析电路的最简形式灵敏度表示图

Fig.20 Symbolic Decision Diagram for sensitivity analysis

首先，在 GRASS 的实现算法介绍中得知，在进行图简约操作之前，按照特定的顺序排列所有的电路元件，将所有的电路元件参数，包括受控源按照顺序依次放在一个 symbol list 中。因而，每一个电路元件都对应与一个唯一的处理序号，我们首先搜索这个 symbol list，找到待求导数的元件参数的处理序号 element_order。并记录它的值。在 GRASS 中，这个顺序是按照读入网表的顺序建立的，而且输出对输入的控制系数 X 放在第一个位置上，即 X 对应的 element_order = 0。对于图 18 的处理电路中，从图 19 中可以直观的看到 R1 的序号为 4。

值得一提的是，因为在 GRASS 中，分析引擎为了提升效率，减少所需处理的元件的个数，在进行图简约操作之前，曾经对并联元件进行了预处理。它将并联元件等价成一个元件，该元件的符号（数值）等于响应元件的并联值，如下图 21 所式：

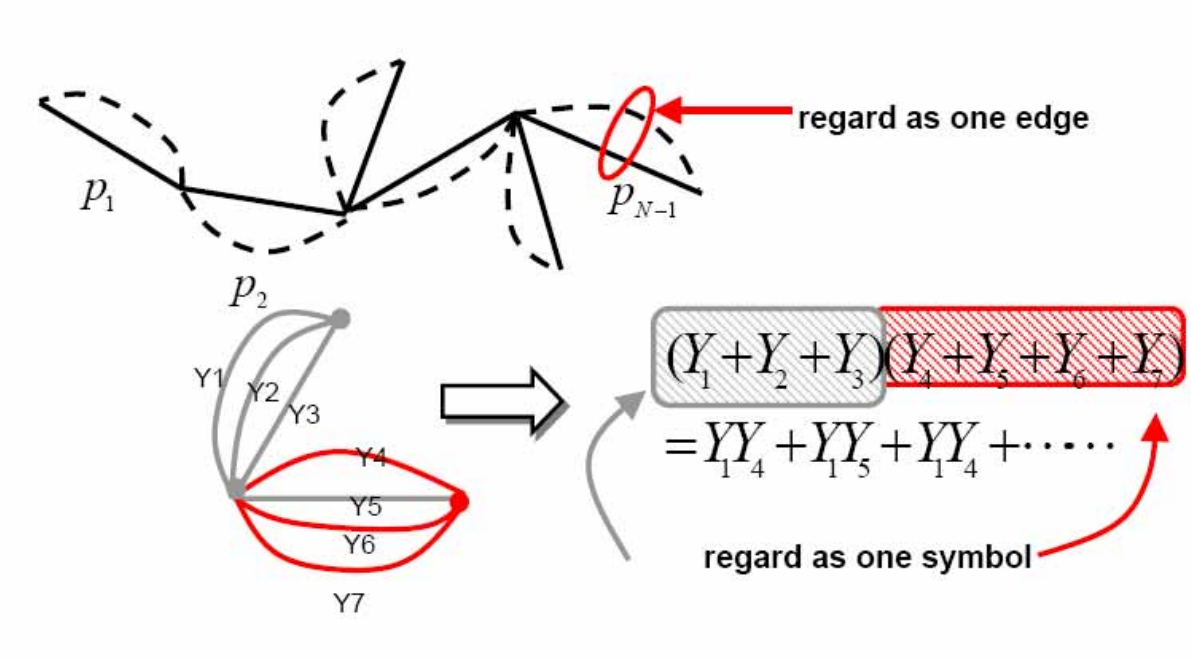


图21 并联边预处理举例

Fig.21 Example for Lumping Parallel Branches

如上图所示，我们将利用一个节点来同时代表 Y_1, Y_2, Y_3 三个元件符号的值，假设最早

读入的元件的Y_1，则以$\wedge Y_1$来代表三个并联元件，它的完整的符号表达式为（$Y_1+Y_2+Y_3$），因为全都是一次项加和的形式，所以只要找到带求导的电路元件所在的表示结点，可以把它当做所求电路元件的符号结点处理。而且它的排列序号为并联结点的序号。

完成上述操作以后，从根结点开始遍历符号化判定图 19, 因为在 GRASS 的分析引擎中，均严格按照排列顺序进行约化，所以我们预先知道 symbol list 中参数索引小与 element_order 的电路元件一定要比索引为 element_order 的元件提早分析，从符号化判定图中看即全部都建立在第 element_order 层的上层，虽然在最简化的符号化判定图中，会出现索引为小于 element_order 的结点指向索引为 element_order+m（m>=1）结点的情况，如图 20 中出现的一个符号为 C3 的右分支指向符号为 R3 的结点，这时说明此路径所引导的乘积项中不会存在元素索引为 element_order 的元件参数，因而此乘积项不会出现在求导记过中，此时我们修改当前这个 element_order-1 的结点的左结点，使其这节指向终结点 0。但是由于 GRASS 分析引擎本身的处理方法，不出现上述的情况，GRASS 得到的图约化判定图如图 22 所示，除非多个分支指向终结点，每一层的结点的数目严格为上一层结点数目的两倍（共享部分展开的情况下）。

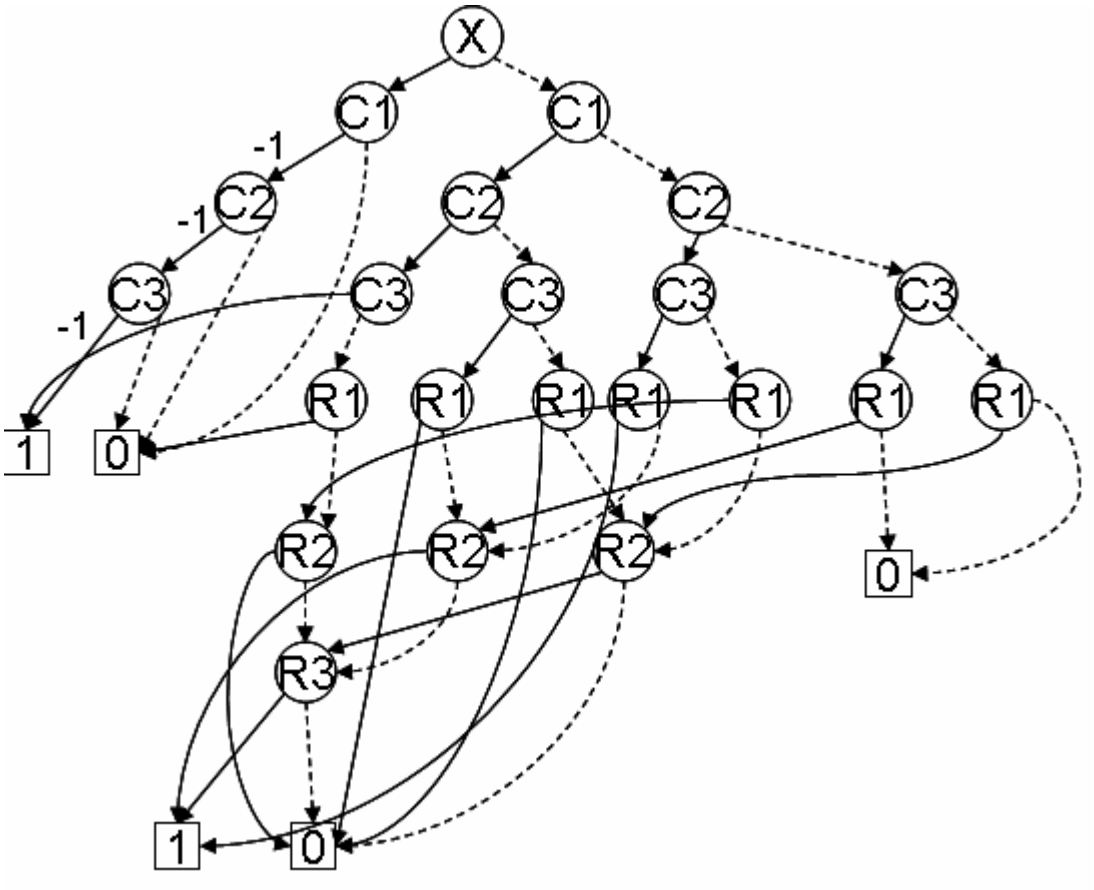


图22 由 GRASS 得到的图约化判定图

Fig.22 Graph Reduction Decision Diagram

如果存在一个结点 m, 它所代表的元件的索引比 element_order 小，并且它的左结点直接指向终结点 1, 说明此路径所引导的乘积项中不会存在元素索引为 element_order 的元件参数，故此乘积项不会出现在求导结果中，此时我们修改结点 m 的左结点，使其直接指向终结点 0。

当遍历到索引为 element_order 的结点时，我们将此结点的符号表达式和数值全部修改为 1， $R^{*}=1$，左结点的指向不发生改变，右结点修改为终结点 0。由图约化判定图得到的关于灵敏度的二分判定图如图 23 所示。值得注意的是，如果单一的求解传输的分子或分母对某一个电路参数的导数的时候，在此过程中需要按照符号化求导原理中所介绍的，需要添加响应的修正因子。在图 18 中的例子，因为是对 R1 对偏导，最后乘上

一个$-\frac{1}{R1^2}$。

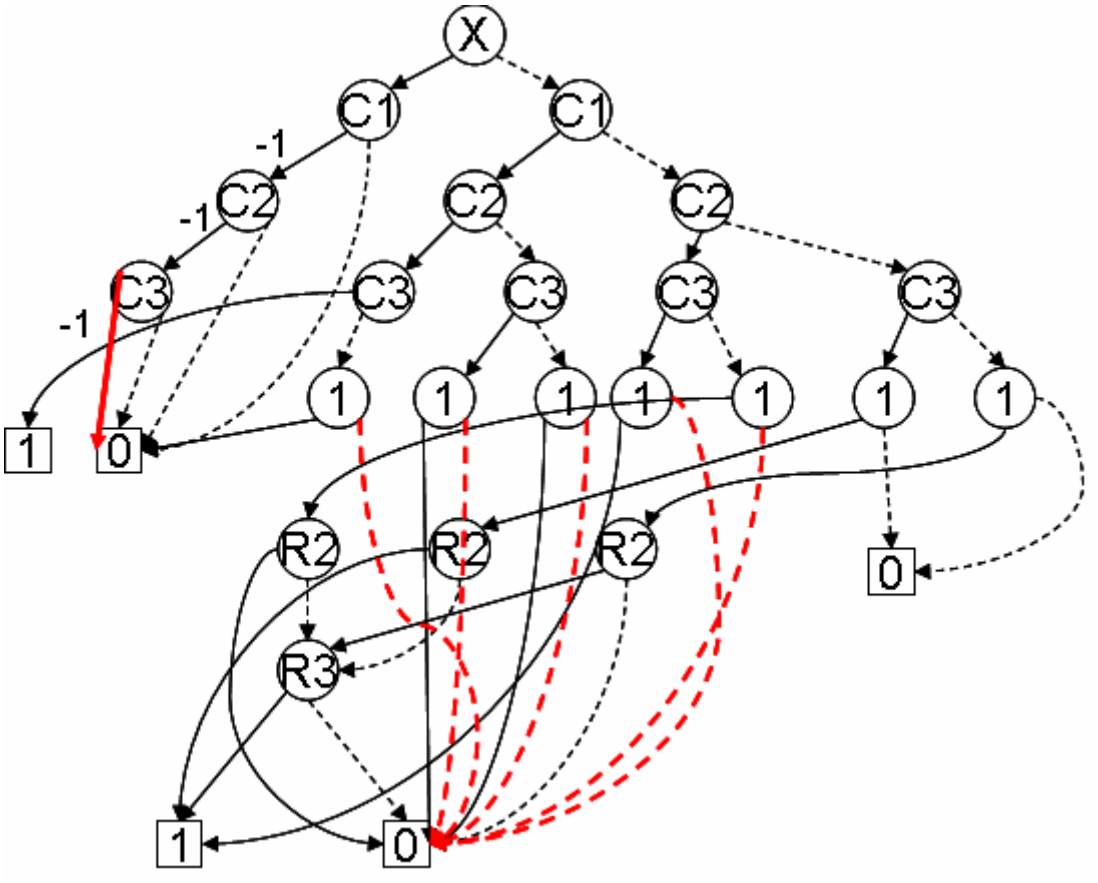


图23 由 GRDD 得到的灵敏度的二分判定图

Fig.23 Symbolic Decision Diagram from GRASS

虽然在求解归一化灵敏度的时候，我们可以按照前面所推导的公式避免了修正因子的处理，但是对与绝对的灵敏度求解，这些都是必须的。

同时要注意，原来分支上所关联的符号不需要修改。

完成上述步骤以后，根结点的左结点和有结点所代表的表达式就是$-\frac{\partial N(s)}{\partial p}$和$\frac{\partial D(s)}{\partial p}$。

从图 20 中得到的

$$-\frac{\partial N(s)}{\partial R1}=0$$

$$\frac{\partial D(s)}{\partial R1}=[C2sC3s*1+C2s*1*G3+C3s*1*G2+1*G2G3]*\frac{-1}{R1^2} \tag{3-49}$$

这和直接对 H（s）的分子分母求导得到一致的结果。

3.7.3 符号化判定图的 S-展开

因为我们建立的符号化判定图存在并联元件的预处理，而且在存在并联预处理的结点里会混合电容、电感或电容的元件，所以，每次求值的时候需要重新计算每一个和频率相关的结点的值，然后利用递归的求值法则进行计算，这样就大大的减慢了计算的速度，无法突出符号化计算速率提升的优点。

这种情况也会延续到灵敏度求值的过程中。因此【23】提出了 S-展开的概念。【24】将这个概念应用到了 GRDD 的求解中。我们也可以得到灵敏度的符号化判定图后应用 GRASS 的 S-展开的操作，以便更加迅速的得到运算结果。如下，我们将简单介绍一下 S-展开的概念和构建过程。

在我们分析原理中，结果由全部的生成树对应的生成项组成，由于电容和电感会引入频率的参数，而且它们分散在各个生成项中，所以不能够得到一个规则的表达式，而 S-展开的表达式如下所示：

$$H(s)=\frac{N(s)}{D(s)}=\frac{a_0+a_1s+...+a_ns^n}{b_0+b_1s+...+b_ns^n} \tag{3-50}$$

分子和分母的多项式系数由代表电路元件符号的乘积所对应的生成项的和组成，它把 s 系数的相同的生成项收集在一起，因此我们计算的时候只需要收集一次 s 的系数，剩下的工作仅仅是频率迭代的过程，而不需要每次都要对每一个结点求值和遍历整个判定图。

如公式 3-47 中，如果将分母转化成 s-展开的形式，则表达式可以表示为如下的形式：

$$\begin{aligned}&C1C2C3s^3+(C1C2G3+C1C3G2+C1C3G3+C2C3G1+C2C3G2)s^2\\&+(C1G2G3+C2G1G3+C2G2G3+C3G1G2)s+G1G2G3\end{aligned} \tag{3-51}$$

这种层次展开的表达式形式在分析中除了提升求值效率以外，还可以提供其它很多便捷的操作，例如主零、极点的近似等等。

GRASS 得到的表达式是 SOE 形式的符号化网络函数，【24】提出的结果方法不需要改变电路分析的算法，只需要对构建的图约化判定图进行展开的操作就可以获得 s-展开的表达形式。

如果想要得到每一个频率项的系数，需要把初始的根唯一的 GRDD 变为多根的 GRDD，每一个根结点分别代表了一个分子或分母的表示多项式系数的判定图，而且共享它们的子图。需要注意的是，即使这里依旧遵从了图约化分析算法中各种共享的原则，当电路变得复杂，而且 s 的阶数变高的时候，依旧会是电路初始的判定图成倍的增大，大大增加了空间的复杂度。

下面我们介绍一下【24】中 S-展开的图约化判定图构建的过程。

S-展开的图约化判定图分量部分来完成，一是 GRDD 的分离操作，另一部分是 GRDD 的展开操作。

1）GRDD 分离操作

所谓的 GRDD 分离操作，就是把 GRDD 中进行过并联预处理的符号做分离处理。

如图 24-a 所示，结点 V 由三个并联元件组成，分别为 R，C 和 L。V1 和 V0 分别为 V 的 1-分枝和 0-分枝指向的子节点，sign1 和 sign0 分别是关联在结点 V 的 1-分枝和 0-分枝上的符号（图 24-b 所示）。

分离结点 V 就是将结点 V 中并联元件单独分离成为 GRDD 中的结点，即用 R，L 和 C 三个结点代替原来的结点。如图 24-c 所示，R，L，C 这三个新的结点通过各自的 0-分枝相连，而其原来的 V 结点已经不存在，它由并联序列中第一个元件所代表的结点所取代，并联序列中的最后的一个元件所代表的结点的 0-分枝指向初始结点 V 的 0-分枝 V0，三个结点的 1-分枝全部指向初始 V 结点的 1-分枝。

因为 0-分枝连接的结点表示结点所代表的符号表达式相加，这正式并联预处理的时候 V 所代表的符号的表达形式。

所有的结点的 1-分枝所关联的符号仍旧是 sign1, 所有的并联序列中的元件所代表的

结点除了最后的一个元素的 0-分枝所关联的符号为初始结点 V 的 0-分枝关联的符号 sign0 外，其余结点的 0-分枝所关联的符号均为正。

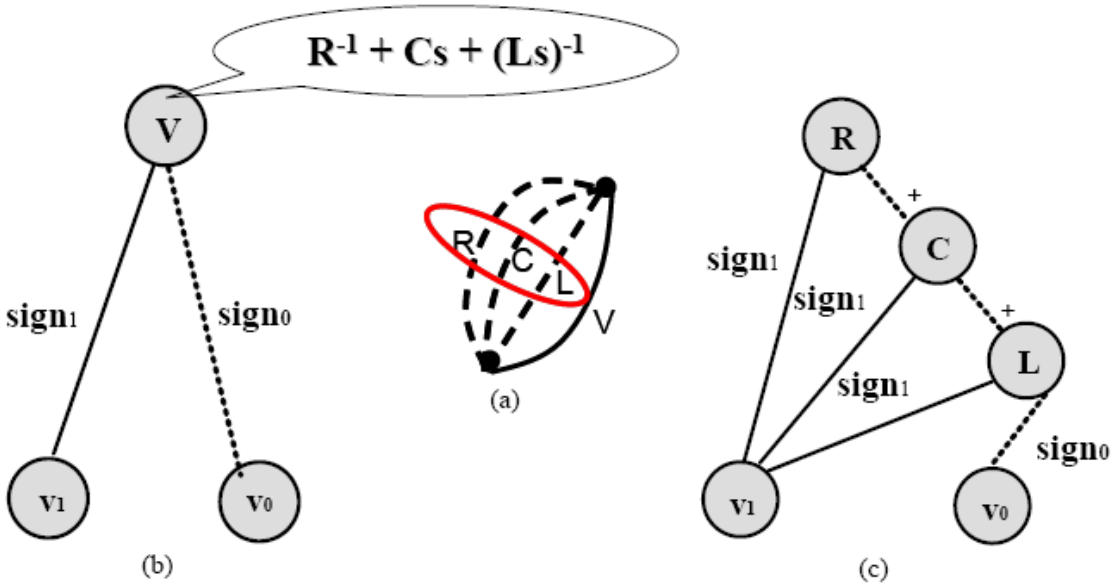


图24 GRDD 的分离操作

Fig.24 GRDD Splitting

假设一个 GRDD 含有|GRDD|个节点，GRDD 分离的操作所需的时间复杂度为 O(m|GRDD|)，一个分离过的 GRDD 所含的节点不会超过 m|GRDD|个，这里的 m 指的是一组并联元件中元件个数的最大值。

完成这一步以后，现在的 GRDD 每一个结点将会只对应于电路中的唯一的一个元素。

2）GRDD 的 S-展开的操作

假设结点 V 所引导的判定图代表的多项式的展开形式为：

$$\sum_i p[i]s^i \tag{3-52}$$

其中的 $p[i]$ 可以是符号化的多项式也可以是 0。V1 和 V0 分别为 V 的 1-分枝和 0-分枝指向的子节点，sign1 和 sign0 分别是关联在结点 V 的 1-分枝和 0-分枝上的符号。V1

和 V0 均是如（3-52）所示的多项表达式。加入 V1 和 V0 的多项表达式已知，分别为 p1 和 p0，

V_Symbol 是结点 V 所代表的电路元件符号，那么 V 的多项表达式为以下形式：

1）V 是终结点 1，则 p[0] = 1;

2）V 是终结点 0，则 p[0] = 0;

3) V_Symbol 是和频率无关的元件的符号，如（G 或者是 Z）

P[i] = V_Symbol * p1[i] * sign1 + p0[i] * sign0;

4) V_Symbol 是电容 C

P[i] = V_Symbol * p1[i + 1] * sign1 + p0[i] * sign0;

5) V_Symbol 是电感 L

P[i] = V_Symbol * p1[i - 1] * sign1 + p0[i] * sign0;

因为电阻不能引入频率变量 s，所以可以由子结点的相同的对应 s 的阶数的系数直接相加，而电感和电容分别引入了 s 的-1 阶和 1 阶，所以需要找到相应的 s 的阶数的系数参数运算。

假设给定一个原始的具有|GRDD|个节点的 GRDD，要得到一个 S-展开的 GRDD 需要先进行 GRDD 分离操作，然后再进行 S-展开。S-展开的运算复杂度为 O((k+1)m|GRDD|)，S-展开的 GRDD 节点数不超过(k+1)m|GRDD|，其中 k 为 GRDD 所表示的多项式的最高阶数，m 为一组并联元件中元件个数的最大值。

由此可见，当 K 的阶数高时，依旧会带来使运算的空间复杂度大大增加。

3.7.4 符号化交流灵敏度提取概述

符号化交流灵敏度信息提取的过程可以简单的论述如下：

Step1：确定待考察灵敏度信息的电路元件 p, 频率范围，以及希望得到的结果数量。我们将直接在网表中进行描述，添加一句.Sens element，提取出 element 的符号，频率范围和传输函数一致。

Step2：遍历因求解传输函数而建立的图约化判定图，并且修改它的结构以便得到灵

敏度信息。删除所有的不包含电路元件 p 的 1-路径上的节点，并且将所有删除后浮空的边直接指向终结点 0。然后将电路元件 p 的符号和数值全部修改为 1。

Step3：将已修改的结构进行 s-展开的处理，并利用求值规则进行求解，最后根据已得的数据画出灵敏度曲线。

3.8 灵敏度求解方法比较

我们详细描述了基于 GRDD 和 DDD 的符号化灵敏度的计算方法，相较于 DDD 符号化灵敏度的分析方法，我们的分析方法主要有一下的优点：

首先，我们的电路元件和符号化的判定图是一一对应的，在 GRDD 中，所有的相同的电路元件全部分布在相同的层次上，这将很容易进行辨别元素的位置；而在 DDD 中，由于 MNA 矩阵本身的构建方法，使得一个电路元件会同时出现在矩阵的不同元素中，如 R1 会同时存在在 a,b,c,d 这四个符号中，这中多对应的方法无形中增加了分析的复杂度。

其次，因为计算结果是枚举所有的有效的生成树，每一条起始于根结点的终结点为 1 的 1-path 都对应一个有效的乘积项，只需要找到待处理元件的层次然后一一按照前面的提出的计算方法直接修改判定图；对于 DDD，存在对同一个行列式不同元素求解代数余子式的问题，如图 12 的例子，需要同时求得 A 对符号为 a 和 d 的代数余子式，然而，并不是所有的代数余子式会对应于该行列式对应符号结点的左节点，即，所求的代数余子式不是行列式本身所对应的待处理符号，这需要递归的寻找此代数余子式的左右结点。

最后，GRDD 最终的和灵敏度相关的符号化判定图只可能在初始判定图的基础上进行简化，而不会有任何新的结点的出现，而 DDD 会引入新的结点指向对应于不同的行列式和待处理元素不对应的代数余子式，增加了空间的复杂度。

3.9 本章小结

本章中我们据论述了基于 GRASS 的进行符号化交流小信号灵敏度分析的方法，并

提出了计算机实现的方法。然后简略的介绍同样借助于二分判定图实现符号化模拟电路分析的行列式判定图（DDD）以及它提出的灵敏度求解的方法，最后针对两种方法进行了比较。

第四章 仿真结果和灵敏度应用分析

4.1 基准电路试验结果

我们将选取不同规模的基准电路和不同的分析元件针对灵敏度求解的正确性和仿真效率进行测试，测试电路组如下：

电路一：RC 滤波器（图 25）；

电路二：带通滤波器 1，2（图 26,27）；

电路三：$\mu a741$运算放大器（图 29）；

电路四：$\mu a725$运算放大器（图 29）；

但首先我们将给出$\mu a741$运算法大器的灵敏度的信息，并进行详细的分析来说明灵敏度的测试结果，之后在逐次给出其它电路的测试结果。

在上面六个电路中，电路一和电路二只包含无源器件电阻（R），电容（C）和理想放大器；电路三和电路四包含有源二极管器件，电路四的规模将大于电路三，其中的晶体管已经全部转化为相应的小信号模型；电路五包含有源的 MOS 管器件，电路三、四应用的晶体管小信号模型和电路五的小信号模型大不相同。

我们的测试平台为 Intel Pentium 1G 内存，主频为 1.8GHz 的处理器。

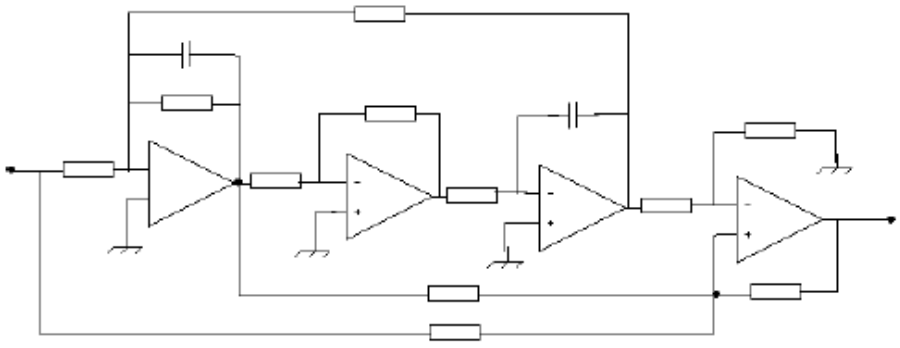


图25 Active RC Filter

Fig. 25 Active RC Filter

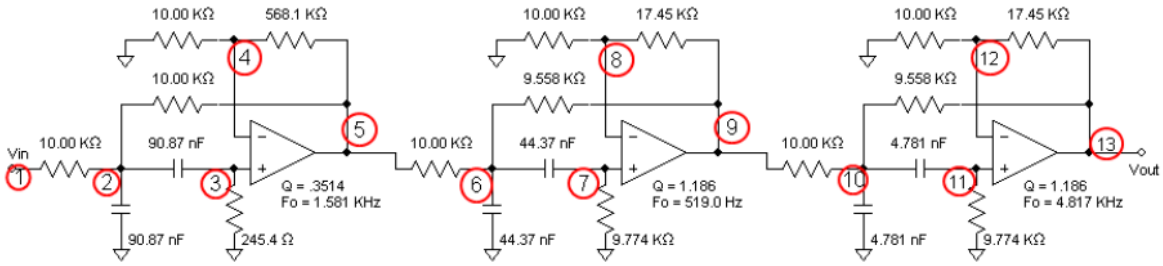


图26 带通滤波器

Fig. 26 BandPass Filter

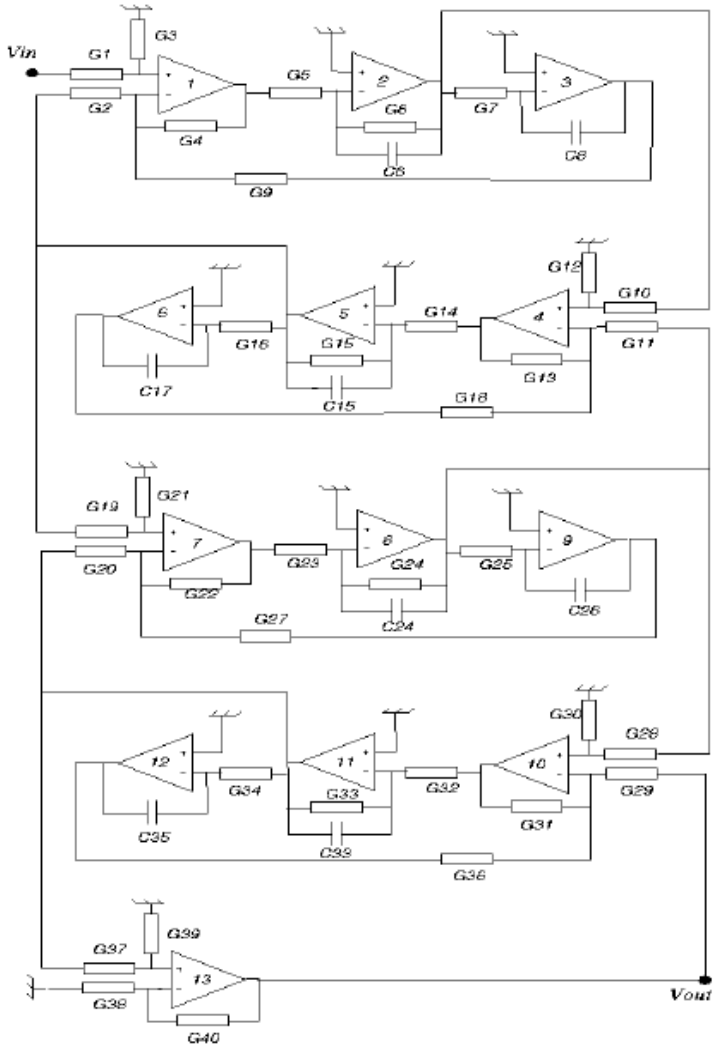


图27 带通滤波器

Fig. 27 BandPass Filter

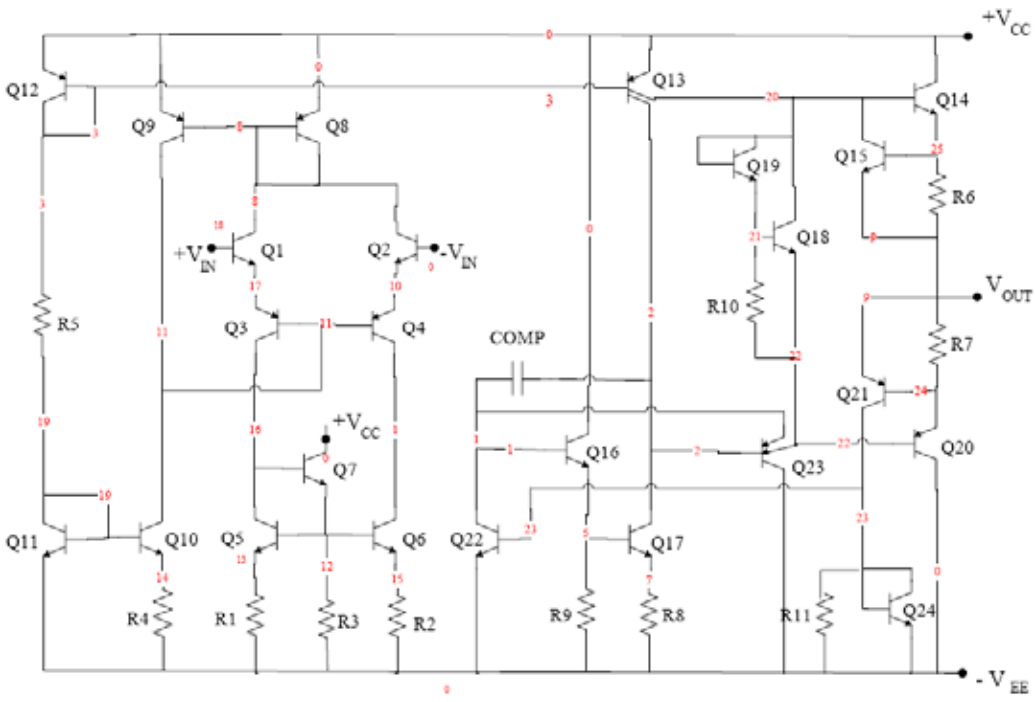


图28 $\mu a741$运算放大器

Fig. 28 $\mu a741$ Amplifier

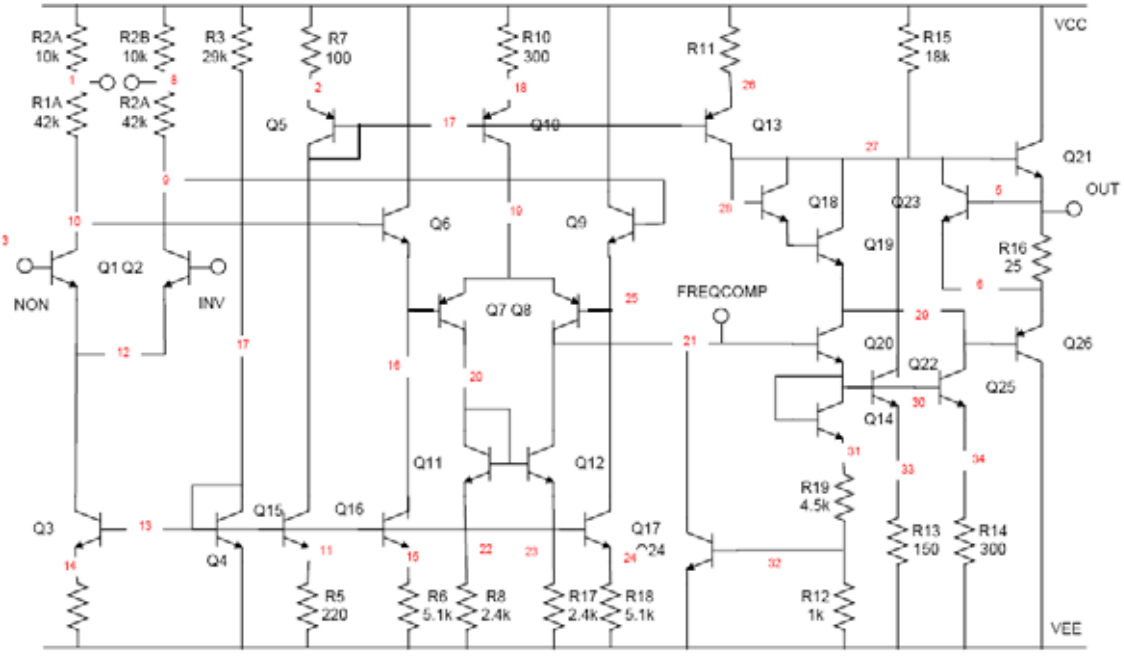


图29 $\mu a725$ 运算放大器

Fig. 29 $\mu a725$ Amplifier

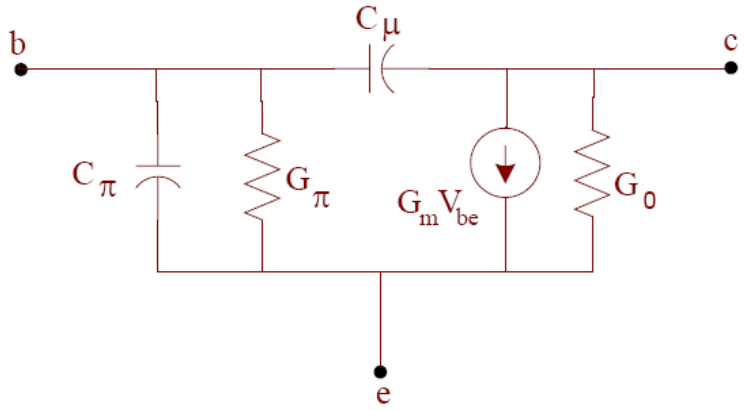


图30 $\mu a741$，$\mu a725$ 运算放大器三极管小信号模型

Fig.30 Small Signal Model for $\mu a741$，$\mu a725$

1 首先给出测试电路的基本信息如表格 2：

其中#Edge 代表电路所含的分支数；#Edge_lump 代表将预处理并联元件后电路所含的分支数； #Node 代表电路所含的节点数；#Symb 代表电路所含的符号数；

表格2 电路信息

Table.2 circuit information

电路名称	#Edge	#Edge_lump	#Node	#Symb
RC Filter	23	22	11	17
Bandpass Filter1	29	29	14	25
Bandpass Filter1	72	68	33	54
$\mu a741$	160	103	24	81
$\mu a725$	166	120	31	81

2 需要注意的是，因为在【24】已经验证了 GRASS 测试结果和 HSpice 之间的严格对应，所以在此我们将不再进行对比两者的频率响应，然而我们求解的是交流灵敏度的

信息，而 Spice 中的交流灵敏度信息仅仅是输出的变化率对输入的变化率，和我们提出的概念是不一致的，所以不能将我们的灵敏度信息和 spice 求解的灵敏度信息去比较。

下面我们首先给出$\mu a741$的一系列结果，并给出具体的讲解，其它电路的相应结果将会依次给出。

从图 31 中我们可以看出，在频率范围为 10hz 到 1000000hz 时，放大器的幅度随着电容的增大而减小因此幅度补偿电容的的一阶导数应该为负值，而且，从图 31 中发现，不同频率点的灵敏度是不同的，例如，在上述频段内灵敏度较高，而在其它的范围内灵敏度较低，从图 32 中可以发现，我们仿真结果完全符合分析过程。图 32 给出了电容在 30pf 附近的幅度对电容的灵敏度信息，而图 35 给出了几个单一的频率时电容在较大范围内的灵敏度信息，从图 33 可以显示出在不同的频率电容相对于幅度的的变化依旧是单调的。图 33 给出了在整个考察的频域上不同的电容相对于幅度的变化，说明在整个频段上幅度相对于电容的变化率基本上是相似的变化曲线。

由图 33 看到在频率较高的时候，幅度的灵敏度相对很大，说明幅度在频率较高的的地方改随着电容的改变变化都很大，而由图 38 看以看出，除了在低频，相位相对于电容的灵敏度信息都趋近于零，所以相位随着电容的改变变化很小。

如果我们想要改变带宽，发现随着补偿电容的减小带宽增大，所以可以通过减小补偿电容来增大带宽，而我们发现在幅度为 0db 出的相位裕度大约 30 度，而相位随着补偿电容的变化在中高频段改变并不明显，而对幅度的改变却非常明显，所以我们可以通过调节电容来增加电路的稳定性。

我们还可以按照灵敏度的大小，符号预测描述电路性能的曲线。灵敏度为正，说明变化特性随着元素的增大而增大，灵敏度的绝对值越大，说明敏感程度越高，灵敏度为负，说明电话特性随着元素的增大而减小。

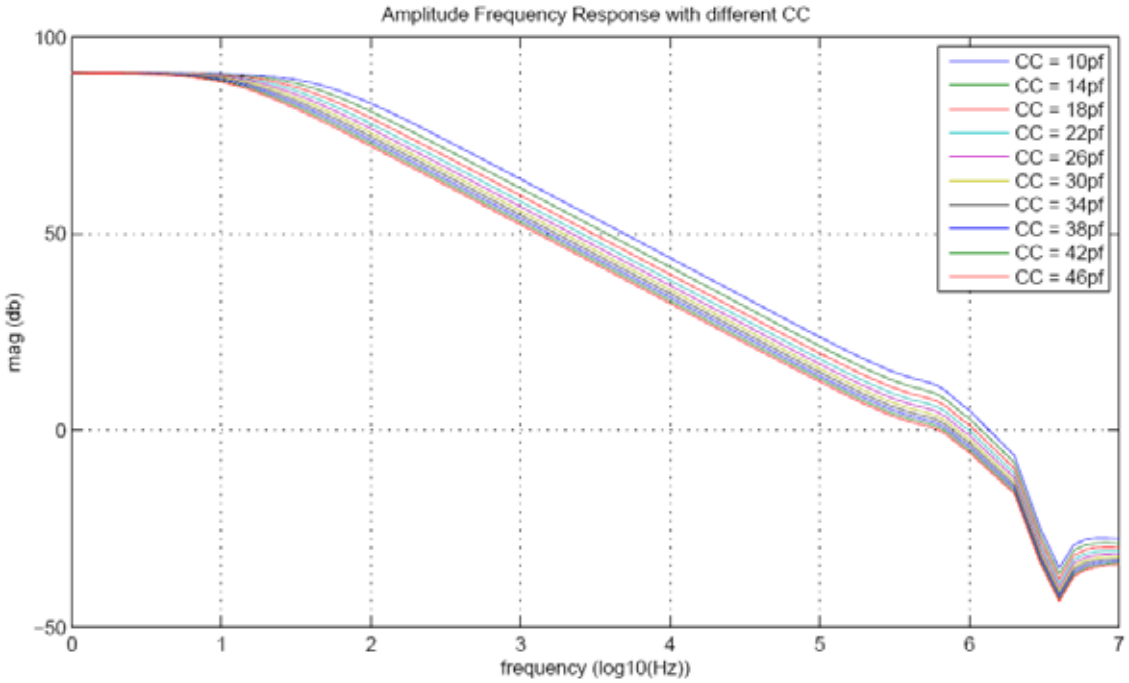


图31 幅频响应曲线

Fig.31 Amplitude Frequency Response

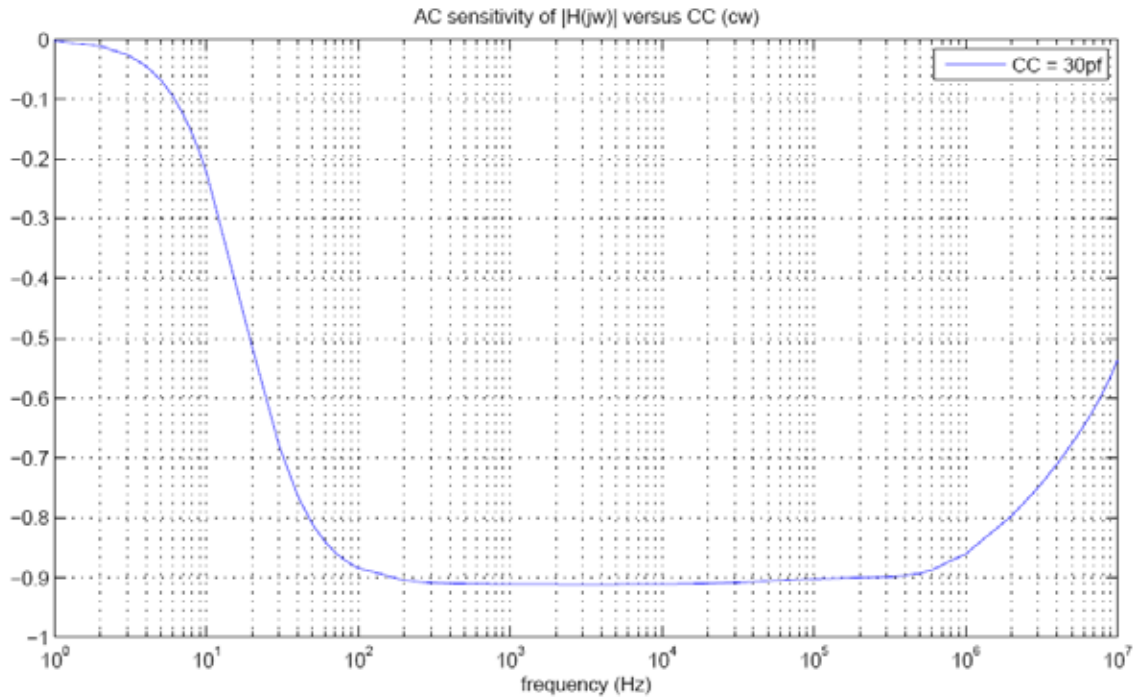


图32 幅度关于 CC 的交流灵敏度

Fig.32 AC Sensitivity of Amplitude versus CC

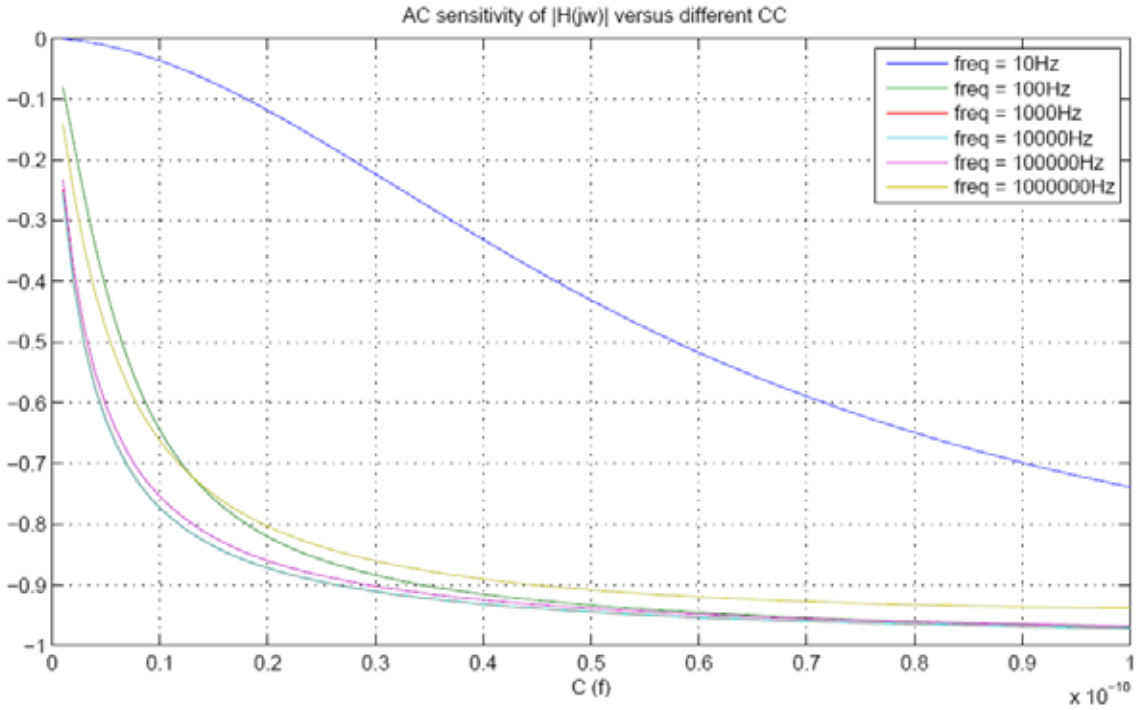


图33　幅度关于 CC 的交流灵敏度

Fig.33 AC Sensitivity of Amplitude versus different CC

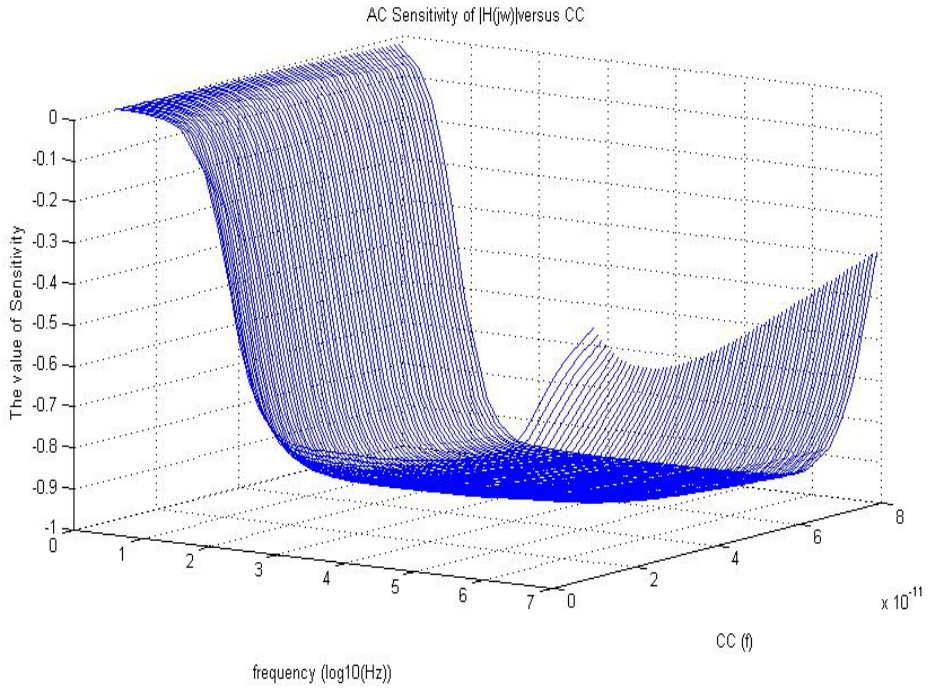


图34 幅度关于 CC 的交流灵敏度

Fig.34 AC Sensitivity of Amplitude versus CC

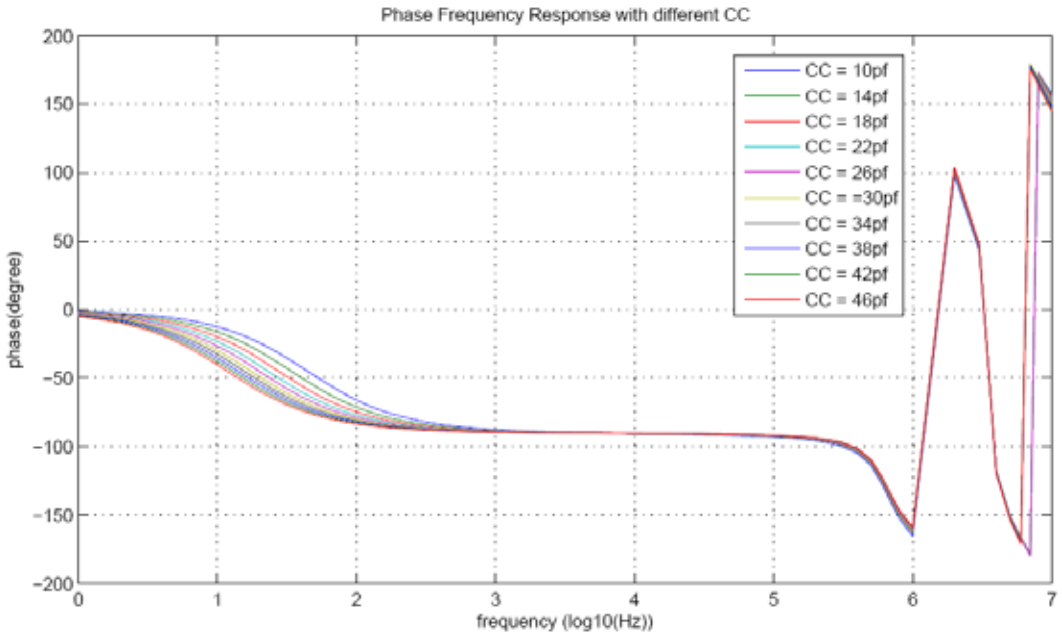


图35 相位响应曲线

Fig.35 Phase Frequency Response

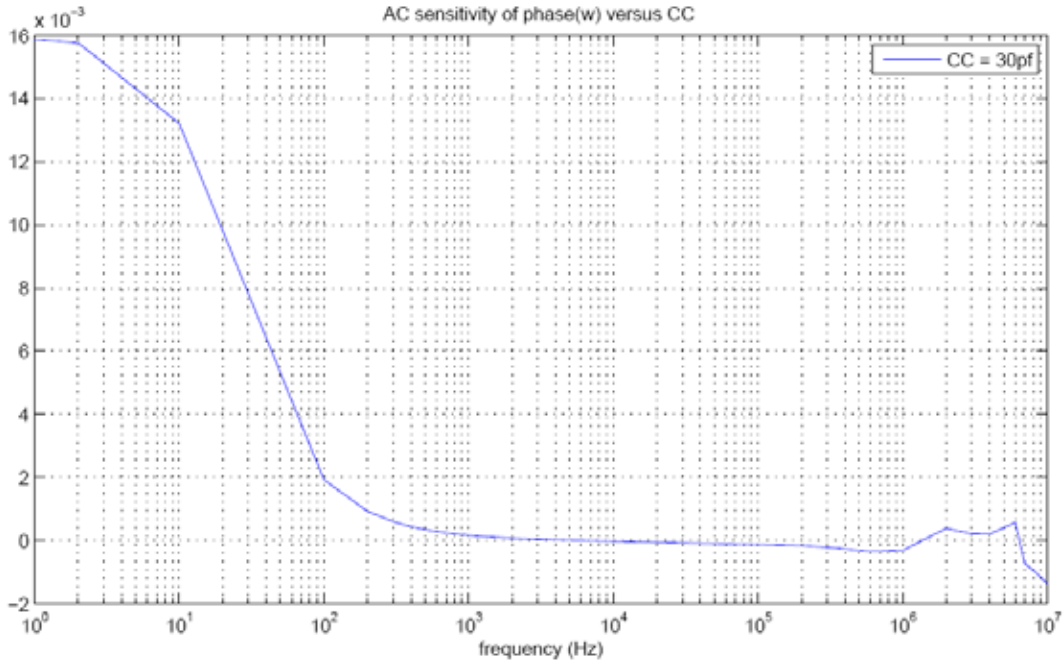


图36 相位关于 CC 的交流灵敏度

Fig.36 AC Sensitivity of phase versus CC

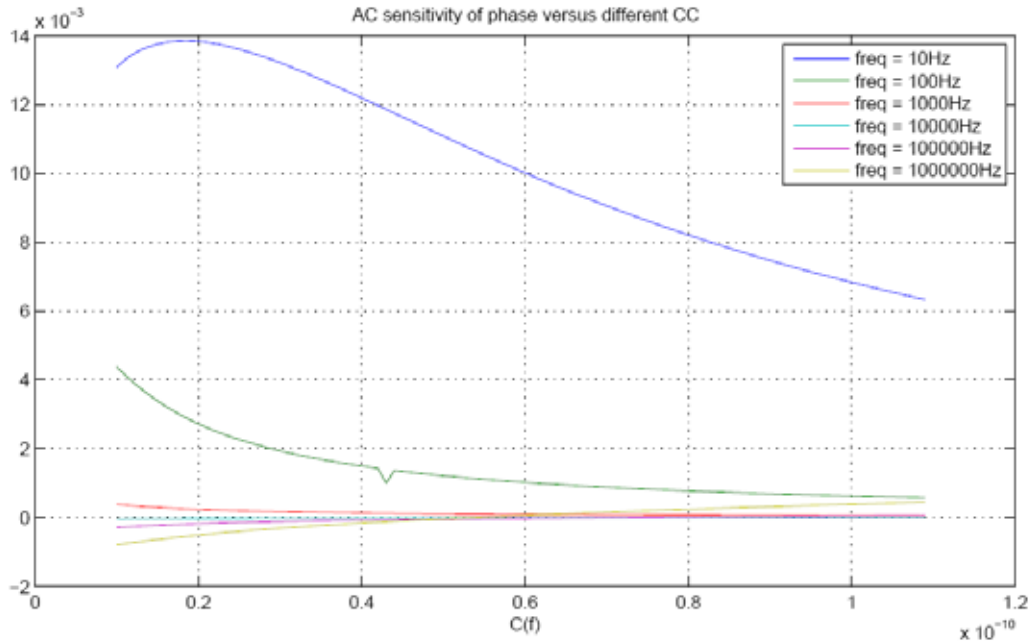


图37 幅度关于 CC 的交流灵敏度

Fig.37 AC Sensitivity of phase versus different CC

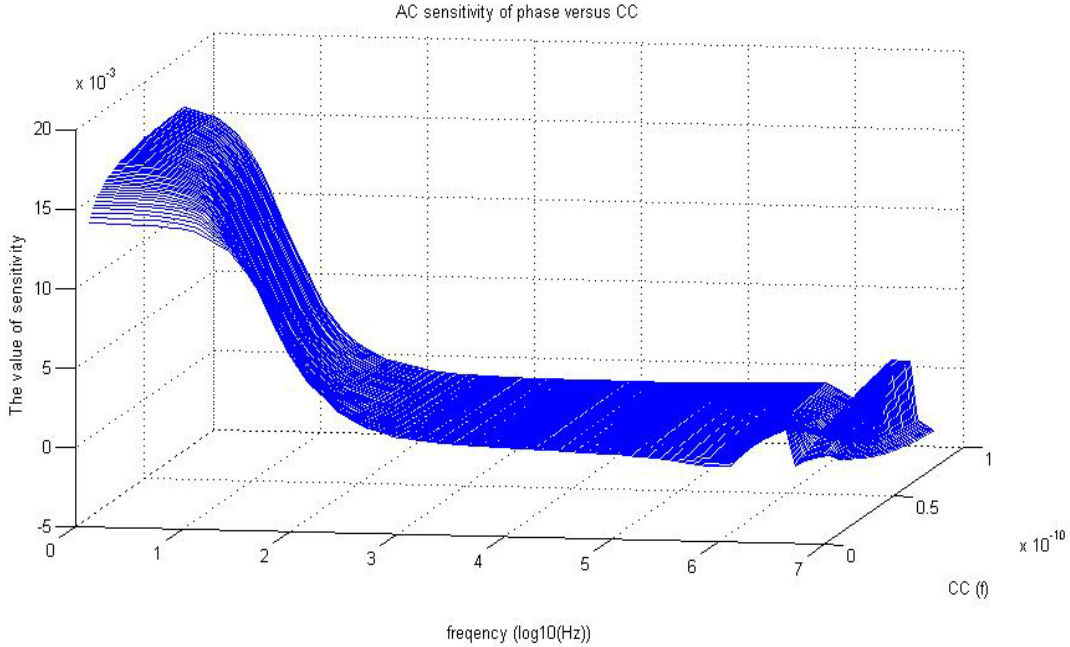
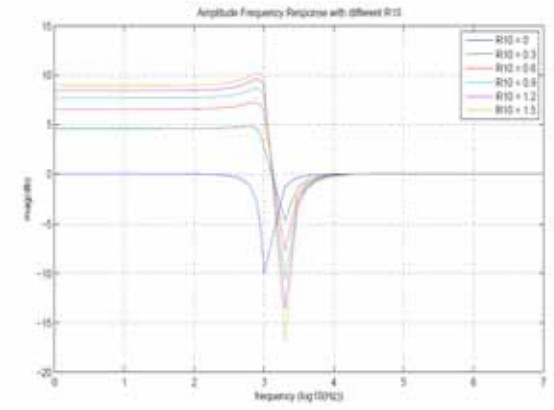


图38 幅度关于 CC 的交流灵敏度图

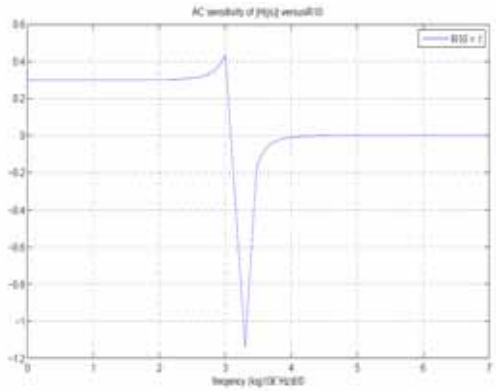
Fig.38 AC Sensitivity of phase versus CC

下面我们给出其它电路的一些仿真结果

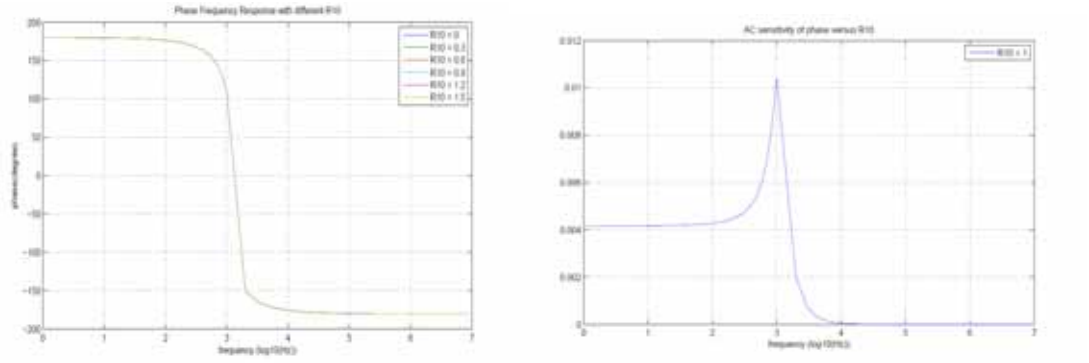
（active RC 01）



（a）幅度响应曲线



(b)幅度关于电阻 R10（=1）时的灵敏度曲线



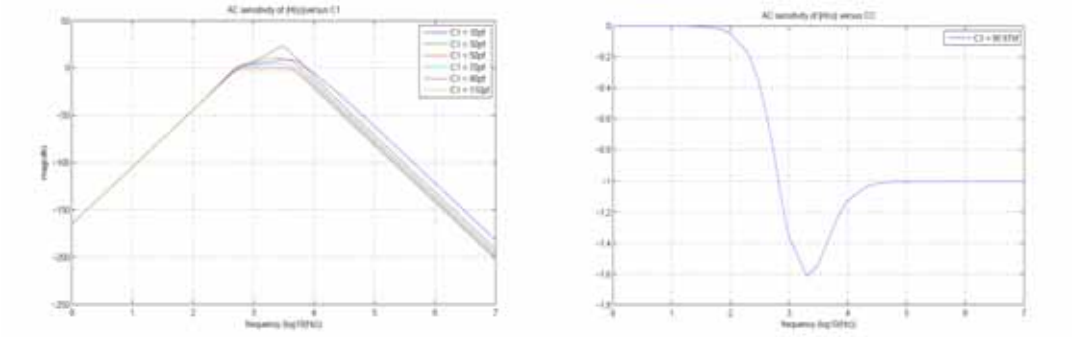
（c）相位响应曲线　　(d)相位关于电阻 R10（=1）时的灵敏度曲线

图39　RC 滤波器频率响应和灵敏度信息

Fig.39　Frequency Response and Sensitivity information for Active AC Filter

由图 39 可以看出，RC 滤波器的幅度在 0-1000Hz 左右随着电阻 R10 的增大而增大，之后随着电阻的增大而减小，而相位相对于电阻的改变较小。说明反馈电阻对电路的幅度影响很大。

bandpass1



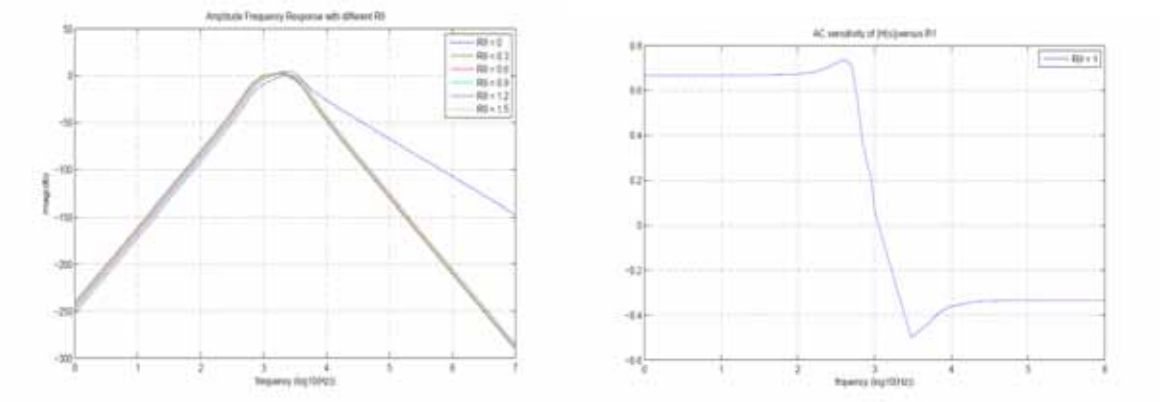
（a）幅度响应曲线　　(b)幅度关于电容 C1（=90.87nf）时的灵敏度曲线

图40 带通滤波器幅度频率响应和灵敏度信息

Fig.40 Amplitude Frequency Response and Sensitivity information for BandPass Filter

由图 40 可以看出，电容 C1 对带通滤波器的上边带影响严重，由电路知识得到，电容 C1 影响低通滤波器的截止频率，所以对上边带影响严重，可以通过调整 C1 来调整截止频率。

Bandpass_2



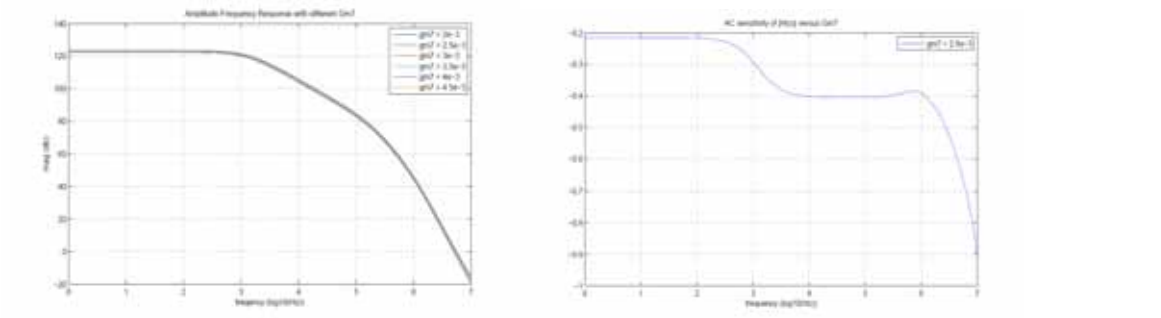
（a）幅度响应曲线　　(b)幅度关于电阻 R9（=1）时的灵敏度曲线

图41 带通滤波器幅度频率响应和灵敏度信息

Fig.41 Amplitude Frequency Response and Sensitivity information for BandPass Filter

由上图可以发现，反馈电阻 R9 对幅度影响相对与截止频率是呈现对称的，当频率小于上边带时，幅度随着电阻的增大而增大，而当频率大于下边带时，幅度随着电阻的增大而减小。

$\mu a725$



（a）幅度响应曲线　　(b)幅度关于电阻 Gm17（=2.6e-3）时的灵敏度曲线

图42 $\mu a725$ 幅度频率响应和灵敏度信息

Fig.42 Amplitude Frequency Response and Sensitivity information for $\mu a725$

4.2 基准电路试验时间信息

表格 3 显示了 S-展开后的 GRDD 的求值效率和 HSpice 求值效率的比较。我们对所

分析电路的传输函数进行频率响应以及相应导数的分析。频率点个数选取 1000 个。可以看到无论是传输函数还是导数的求解，求值效率都会比 HSpice 的数值分析快，所以很好的显示了 GRASS 的分析优势。

表格3 GRASS 与 Hspice 数值分析效率对比

Tablet.3 Comparison of Numerical Analysis between GRASS and HSpice

电路名称	Construction of GRDD	Evaluate of H(s)	HSpice-2005 (ac-analysis)	Evaluate of dN(s)/dp and dD(s)/dp
RC Filter	0.06s	0.002s	0.06s	0.002s
Bandpass Filter1	0.05s	0.004s	0.08s	0.004s
Bandpass Filter2	0.13s	0.007s	0.08s	0.006s
$\mu a741$	5.9s	0.03s	0.16s	0.03s
$\mu a725$	29.25s	0.05s	0.2s	0.03s

4.3 交流灵敏度的应用

微电子技术和微电子系统设计的发展给电路仿真带来了巨大的挑战，尤其是随着系统复杂性的增加，在电路仿真和验证过程中消耗了过多的时间。为了获取好的电路结构和优良的性能，需要对不断修改电路元件的参数，伴随参数值的改变从而进行一次又一次的仿真。为了提高基于电路参数改变而仿真电路的效率，以及扩大分析电路的范围，我们尝试结合电路参数的敏感性分析。

工程师可以按照利用前面给出的信息进行参数的调整，如相位裕度、带宽等等。

但参数的改变足够小时，我们可以利用灵敏度的信息进行现行近似，传输函数可以表达为如下的形式：

$$H(s)_{new} = H(s)|_{p=p_0} + (\partial H(s)/\partial p|_{p=p_0}) * (\triangle p)$$

结合区间的概念和基于敏感性的方法，我们也可以计算出 worst-case response:

$$f_0 - \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i} \right| |\phi x_i| \le f_0 \le f_0 + \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i} \right| |\phi x_i|$$

但是由这个结果我们时常得到一些范围过于大的结果。进入深亚微米级以后，电路的某些参数微小的变动会直接影响电路的稳定性以及优化的效果。由前面的例子可以得到，大多数的参数在主频域上是单调的，所以可以利用灵敏度的信息进行预测。

第五章 全文总结

5.1 主要结论

本文介绍了一个基于拓扑结构的符号化模拟电路仿真器GRASS的基本算法和实现，并在此基础上实现了灵敏度信息的提取，灵敏度信息的提取没有导致多余的空间复杂度，最后得到符号化分析结果和相关曲线。

本文的算法经过严格的数学推理，由符号化公式到处的数值曲线可以应用到电路设计、工艺等多个领域，而且数值结果具有高效和直观性。而且数值分析的速度可以获得比HSPICE仿真器更高的效率，算法的时、空复杂度都在一个比较理想的范围中。

5.2 研究展望

本文论述的内容对符号化的电路分析进行了初步的探索，由于符号化分析算法本身的复杂性，本文所提出的仿真器对于目前较大模拟电路的分析能力还是有限的。未来的研究可以有如下的部分：

对于灵敏度而言，由于大多数的放大器以及一些滤波器的设计需要零点和极点的信息，如果能够得到传输函数零极点和电路的对应关系，甚至是零极点相对于电路元件的灵敏度信息都会为设计带来极大的方便。除此之外，灵敏度的信息可以应用在更多的领域中。而且在应用方面，可以借助与现在的结构进行更多的应用探索从而揭示电路更多的性能。

为了提高可处理电路的规模，我们希望对电路进行层次化处理，即将一个大规模的电路逐次分成若干个独立的模块，这样可以大大提高电路的处理规模。

由于模拟电路的特征往往由一些关键部分决定，而精确仿真结果有时候过于复杂，所以有效的近似可以提供给设计者准确而简化的信息。

因为算法分析电路的复杂度与符号处理顺序由密切的关系，需要探索符号处理顺序的通用方法来减低算法的复杂度，甚至实现更大规模年模拟电路的仿真。

参 考 文 献

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

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

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

[4] C.-J. Shi and X.-D. Tan, “Canonical symbolic analysis of large analog circuits with determinant decision diagrams,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 1, pp. 1–18, January 2000.

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

[6] E. Wehrhahn, “Evaluation of transfer functions of ideal S.C. networks in z-domain using standard linear symbolic or semisymbolic network analysis programs,” Electron. Lett., vol. 16, no. 21, pp. 801-802, October 1980.

[7] A. Konczykowska and M. Bon, “SCYMBAL2: A portable computer program for efficient all-symbolic hierarchical analysis of large multiphase wsitched capacitor networks,” Proc. European Conf. Circuit Theory Design, Stuttgart, pp. 375-378, 1983

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

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

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

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

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

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

[14] H. Ur. “Root locus properties and sensitivity relations in control systems,” IRE Trans. Automat. Conrr. vol. AC-5, pp. 57-65. Jan. 1960.

[15] A. G. J. MacFarlane, “Multivariable Nyquist-Bode and multivariable root-locus techniques,” in Proc. IEEE Conf. Decision Control,1976，pp.342-347

[16] M. Eslami, An Introduction to Theory of Sensitivity of Dynamic Systems. In preparation

[17] G. Shi, W. Chen, and C.-J. R. Shi, “A graph reduction approach to symbolic circuit analysis,” in Proc. Asia South-Pacific Design Automation Conference (ASPDAC), Yokohama, Japan, Jan. 2007.

[18] W. Chen and G. Shi, “Implementation of a symbolic circuit simulator for topological network analysis,” in Proc. Asia Pacific Conference on Circuits and Systems (APCCAS), Singapore, Dec. 2006, pp. 1327–1331.

[19] R．E．Bryant，"Formal Method for Functional Verification". http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.3464

[20] W. Chen, Applied Graph Theory - Graphs and Electrical Networks. Amsterdam: North-Holland, 1976.

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

[22] 汪蕙，王志华. 电子电路的计算机辅助分析与设计方法. 北京：清华大学出版社 2006：160-188

[23] C.-J. Shi and X.-D. Tan, “ Compact Representation and Efficient Generation of s-Expanded Symbolic Network Functions Computer-Aided Analog Circuit Design” IEEE Trans. on Computer-Aided Design, vol. 23, no. 6, pp. 907–918, June 2004.

[24] 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.

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

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

[27] 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.

[28] 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.

[29] G. Gielen and W. Sansen, Symbolic Analysis for- Automated Design of Analog InteRrated Circuits. Norwell, MA: Kluwer, 1991.

[30] S.-D. Shieu and S.-P. Chan, "Topological formulation of symbolic network functions and sensitivity analysis of active networks," IEEE Trans. on Circuits and Systems, vol. CAS-21, no. 1, pp. 39-45, 1974.

[31] Z. Yin, "Symbolic network analysis with the valid trees and the valid tree-pairs," in IEEE Int'l Symposium on Circuit and Systems, (Sydney, Australia), pp. 335-338, 2001.

[32] S.-D. Shieu and S.-P. Chan, “Topological formulation of symbolic network functions and sensitivity analysis of active networks,” IEEE Trans. on Circuits and Systems, vol. CAS-21, no. 1, pp. 39–45, 1974.

[33] S.-D. Tan and C.-J. 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, June 2004.

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

[35] W. Verhaegen, G. Gielen “Efficient DDD-Based Symbolic Analysis of Linear Analog Circuits,” IEEE Tran. on Circuit and Systems – II: Analog and Digital Signal Processing, volL. 49, no. 7, pp.474–487, July 2002.

[36] S. W. Director and R. A. Rohrer, “The generalized adjoint network and network sensitivities,” IEEE Trans. Circuit Theory, vol.CT-16,pp. 318-323, Aug. 1969.

[37] D. A. Hocevar, P. Yang, T. N. Trick, and B. D. Epler, “Transient sensitivity computation for MOSFET circuits,” IEEE Trans. Computer-Aided Design, vol. CAD-4, pp. 609-620, Oct. 1985.

[38] T. N. Trick, F. R. Colon, and S. P. Fan, “Computation of capacitor voltage and inductor current sensitivities with respect to initial conditions for the steady-state analysis of nonlinear periodic circuits,” IEEE Trans. Circuits Syst., vol. CAS-22, pp. 391-396, May 1975.

[39] T. Nguyen, P. Feldmann, S. W. Director, and R. A. Rohrer, “SPECS simulation validation with efficient transient sensitivity computation,” in Proc. IEEE In?. Con$ on CAD, Nov. 1989, pp.252-255

[40] F. Yuan and A. Opal, “Adjoint network of periodically switched linear circuits,” in Proc. IEEE ISCAS, vol. 6, Monterey, CA, May 1998, pp. 298–301.

致 谢

转眼，到了第三个冬天，在我的 22 的岁的尾巴上，我的研究生生活马上也就要画上句号了。

夏天开始的课题即将打下最后的句号，跨越季节，才发现有那么多的地方需要弥补，只是转角的选择，让我来不及让它精致。在不停的更换的思路中，我悄然的成长。也许今后我会选择其它的专业作为继续深造的方向，这意味着我在 EDA 的路上渐行渐远，但我依旧不后悔以在交大微电子学院的这两年作为我青春的开始。

在交大这两年的时光，感谢导师施国勇教授对我的影响。他带给我了一个走进 EDA 软件开发的窗口，带给我了一个执着于学术的精神，带给我了一个谨慎、不轻易放弃的精神鼓励。严谨、执著、奉献，这六个字的在他的身上得到了完整的体现。感谢两年来给予我帮助的老师的和同学，你们让我觉得自己是幸运的。

爸爸妈妈永远都是我最安全的港湾，脆弱的时候的支持我，彷徨的时候鼓励我，我会因为它们的爱而泪流满面，更因为“女儿”的称谓而更加努力。

附录一 符号与标记

导纳 (Admittance)

阻抗 (Resistance)

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 current source, VCCS)

流控电流源 (Current control current source, CCCS)

零器 (Nullor)

零阻器 (Nullator)

泛阻器 (Norator))

入射矩阵 (incidence matrix)

约化入射矩阵 (reduced incidence matrix)

攻读硕士学位期间已发表或录用的论文

【1】孟晓旋，“ 符号化交流灵敏度信息的提取 ”，现代电子技术