申请上海交通大学工学硕士专业学位论文

符号化模拟电路仿真器的实现与应用

学　　校： 上海交通大学

院　　系： 微电子学院

班　　级： B0421091

学　　号： 1042109028

姓　　名： 陈微微

专业方向： 计算机系统结构

指导老师： 施国勇教授

导　　师： 付宇卓教授

上海交通大学微电子学院

2007 年 1 月

A Dissertation Submitted to Shanghai Jiao Tong University for

Master Degree of Science

A Symbolic Analog Circuit Simulator

Author：CHEN, Weiwei

Specialty: Computer System Organization

Advisor Ⅰ: Prof. SHI, Guoyong

Advisor Ⅱ: Prof. FU, Yuzhuo

School of Microelectronics

Shanghai Jiao Tong University

Shanghai, P.R.China

December, 2006

摘 要

本文论述了一个基于拓扑方法的符号化模拟电路仿真器。该仿真器基于一种全新的拓扑网络分析算法，通过电路子图约化的方式，建立表达电路生成项的二分判定图。通过子图和节点共享以及二分判定图的相关操作，获得理想的电路分析的时间和空间复杂度。

本文第一章和第二章首先介绍符号化分析方法，然后提出一组基于拓扑法的电路分析定理，第三章中具体介绍仿真器的实现，第四章报告了实验的分析结果，第五章讨论了符号化的近似分析方法，最后对全文进行总结和展望。

本文提出的仿真器是目前第一个可以成功地通过拓扑方法分析较大规模模拟测试电路(20 – 30 个传输晶体管)的符号化模拟电路仿真器。

关键词：符号化分析、二分判定图、图约化、启发式排序

ABSTRACT

Many topological approaches to symbolic network analysis have been proposed in the literature, but none are implemented ultimately as a simulator for large network analysis due to their complexity and exponentially increasing number of terms. A novel methodology adopted in this paper uses 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 described in this thesis. With sharing and other manipulations of BDD, high simulation and evaluation performance is achieved.

The thesis is organized as follows. A brief introduction to symbolic analysis is in Chapter 1. The graph reduction rules and corresponding algorithms are presented in Chapter 2. Implementation details on the symbolic simulator are discussed in Chapter 3. Experimental results are reported in Chapter 4. In Chapter 5, approximate circuit analysis approaches are introduced. Conclusions and future work are reported in Chapter 6.

The simulator in this thesis is probably the first one ever capable of analyzing large analog circuits directly from the circuit topology.

KEYWORDS: Symbolic Analysis, Binary Decision Diagram, Graph Reduction, Heuristic Ordering

目录

图目录

表格目录

1 引言

大规模集成电路技术的发展推动了片上系统(System on Chip, SoC)等大型系统级芯片的诞生。一个片上系统一般包括数字与模拟电路两个部分。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，数字电路设计目前已基本实现了设计过程的自动化，相应的理论与技术也已经比较成熟，已有大量成熟的工具运用于深亚微米数字电路设计，例如可以使数字设计获得良好性能的逻辑综合工具和物理版图设计工具。然而模拟电路设计工具的发展仍然面临许多挑战，由于模拟电路本身的复杂性以及对设计经验的依赖，市场对支持模拟电路设计的工具有很大的需求。

模拟电路仿真器作为一种主要的模拟电路设计验证自动化工具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[24]。SPICE 是一种通用的电路仿真器，它可以分析包括各种非线性的二极管及场效应管的电路，并可对电路的直流偏置特性，交流特性以及时间域传输特性进行分析。对于数值型电路仿真器，一般情况下，当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地验证电路设计，但很难预计电路参数改变后的性能情况。灵敏度分析法可以比较好的用于提高模拟电路的性能，这种方法主要是归一化计算电路元件参数对电路性能产生的影响，从而帮助设计者决定如何修改电路参数以获得理性的性能。但是对于一个电路而言计算所有元件的灵敏度往往是没有必要的，而且几乎不可能通过数值计算的方法来计算电路中所有元件之间的灵敏度问题。符号化的分析方法可以比较好地解决这个问题，电路级的符号化分析方法是一种通过电路自变量（时间与频率）、因变量（电压和电流）以及符号化的电路元件来形式化计算电路特性和行为的方法。由于最终分析的结果是符号形式的公式，可以很明显找出决定电路行为和性能的参数。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[2]。

随着全定制电路(ASIC)和超大规模集成电路(VLSI)设计周期的缩短和复杂度的提高，模拟电路的漫长设计周期及其设计的不稳定性越来越不能满足集成电路系统

的设计要求。为了能够追上数字设计的步伐，模拟电路设计从系统定义到流片的时间与代价也必须大幅下降。符号化的分析方法及相应的仿真工具将成为大大提高模拟电路设计能力的重要辅助手段，因为符号化的分析方法可以清晰地揭示电路的特性，同时它也可以与其他的技术例如电路尺寸调节、可测性分析相结合来实现电路设计的自动化。

1.1 符号化电路分析的历史

电路的符号化分析历史悠久。上世纪 60 年代末及 70 年代初期，随着计算机技术的进步，符号化电路分析方法成为研究的热点，其代表有 SNAP[37]和 NAPP[38]，他们主要用来分析模拟滤波器。基于图形的分析方法（例如生成树枚举法和符号流图法）被认为是最适合用于分析小规模整体均符号化的电路；若只将频率看作符号，可以采用数值的方法来分析规模较大的电路；将上述两种方法结合，又诞生了所谓的符号数值混合分析的方法，也就是将小部分的电路参数符号化，从而快速地分析规模较大的电路；随着 SPICE 的诞生，符号化电路分析的优势逐渐被 SPICE 准确和快速的数值结果所掩盖。到了上世纪 80 年代，为了克服符号化分析对电路规模的限制，层次化分解的分析方法被提出，电路符号表达式不再是通过展平整个电路来获得而是嵌套式地产生；同时基于矩阵行列式的分析方法得到发展，它们可以像图形分析方法一样对整体都符号化的电路进行有效的分析。从上世纪 80 年代后期开始，符号化电路分析方法开始焕发新的活力，主要是出于对于集成模拟电路分析的需求，诞生了许多成功的符号化仿真器，如 *ISAAC[6][7]，ASAP[8][9]，SYNAP[10][11]，SAPEC[11]，SSPICE[13]，SCYMBAL[14]，SCAPP[15]*和 *GASCAP[16]*。在这些工具中，最灵活和最有效的工具是其中基于行列式的分析法以及符号流图的方法。符号化电路分析方法重获新生主要有两大驱动力，首先是计算机计算性能的大幅提高以及相关高效算法的发展，其次是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。

1.2 符号化分析方法的定义

正如上文所述，电路级的符号化分析法是一种形式化的分析方法，符号化分析

研究主要针对线性电路的频域特性。一个集总 (lumped) 的线性时不变电路，对于已知输入，相应输出的传输函数可以表示成两个多项式相除的形式，其自变量为 x，如下式：

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

其中x是一个复数形式的频律变量，对应离散时域电路的z变量或连续时域电路的s变量；$p_1,\ldots,p_m$为电路元件符号；式中多项式系数$a_i(p_1,...,p_m)$及$b_i(p_1,...,p_m)$分别是由电路符号组成的多项式。对于完全符号化的电路，多项式系数完全由电路元件的符号组成；对于符号化和数值型混合电路，多项式系数一部分由电路元件符号组成，另一部分为具体数值。下面是一个电路的例子（图 1.1），符号化的电路仿真器可以得到如下的传输函数：

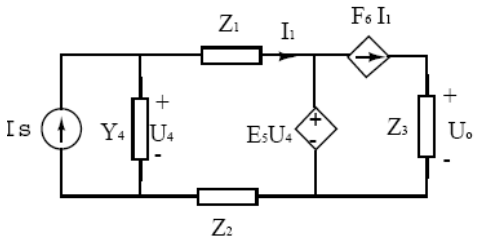


图 1 符号化电路分析举例

Fig.1 An Example for Symbolic Circuit Simulator

$$H(s)=\frac{U_o}{I_s}=-\frac{E_5Z_2^{-1}Z_1^{-1}F_6-Z_2^{-1}Z_1^{-1}F_6}{Z_2^{-1}Z_3^{-1}Y_4+Z_2^{-1}Z_1^{-1}Z_3^{-1}+Z_3^{-1}Y_4Z_1^{-1}-E_5Z_2^{-1}Z_3^{-1}Z_1^{-1}} \qquad (1\text{-}2)$$

电路中若出现非线性元件或者运算放大器，可选择不同的小信号线性模型转化后再进行分析。

1.3 符号化分析的基本方法与分类

符号化分析一般以描述电路的网表作为输入（例如最为著名的 *SPICE* 语法）。由于符号化分析方法基于线性电路，对于电路中的非线性部分（如二极管、场效应管、运算放大器等）必须先转换成线性化的小信号等效模型，电容 C 和电感 L 同样

地表示成频域上的阻抗与导纳的形式（*Cs* 或者 *Ls*）。经过上述转换后，即可以开始进行分析运算。

符号化分析方法大致可以分成两类：代数方法与拓扑方法。代数方法主要通过建立和求解电路方程来获得最终的分析结果（例如求描述电路行列式的代数余子式等）;拓扑方法一般无须建立电路方程,由于电路本身可以看作一张图或者一个网络，通过相应的图操作可以获得所需的符号化的分析结果（例如枚举电路中的基本回路或者是生成树）。

对应于上述分类，目前具体的符号化电路分析算法大约可以分为五种：矩阵行列式法、符号流图法、生成树枚举法、参数提取法和数值插入法。矩阵行列式法通过解电路方程组，例如使用符号化的 *Gauss* 消去法或者递归使用 *Laplace* 展开法来求解；符号流图法使用 Mason 公式，根据明确定义的法则来寻找表征电路方程中各阶项所对应的路径和回路来计算所需结果；生成树枚举法通过枚举电路图中的生成树，计算每棵生成树所对应的项来得到计算结果，其中电路图的每个分支都有相应的权重而生成树项由这棵树每个分支的权重所决定；参数提取法仍然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；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。可以说矩阵行列式法和参数提取法为代数型分析法，而符号流图法和生成树枚举法是基于拓扑分析的方法。这些具体的算法都有各自的优点和局限性，被公认为比较有效和灵活的方法主要是矩阵行列式法（仿真器例子有：*ISAAC[6][7]*、*SYNAP[10][11]*、*SAPEC[12]*和 *SSPICE[13]*）和符号流图法（仿真器例子有：*ASAP[8]*、*SCYMBAL[14]*和 *SCAPP[15]*）。生成树枚举法在处理各种类型受控源的问题上存在困难，并且难以避免产生可以消去的项，这些项的产生会占用多余的 CPU 时间并且会对后面符号化近似工作产生影响。参数提取法适用于部分电路符号化的分析。数值插入法适用于仅以频率变量为符号的分析以及规模较大的电路。

符号化分析算法的难度主要还是在于其算法时间和空间复杂度会随着电路结点和元件的增加而呈指数增长,从而导致符号化电路分析往往局限于规模较小的电路。

1.4 符号化模拟电路仿真器的潜在优势及与数值分析法的对比

符号化分析的潜在优势有：

1) 揭示电路行为特性。符号化的分析方法可以自动地提供电路特性符号化表达式，这些表达式不会随着电路参数的变化而改变，通过带入具体的数值或者忽略数值较小的部分来简化表达式都可以很快且准确的得到具体电路的计算结果。

2) 避免数值计算过程中的舍入误差以及计算收敛性问题，可以提高设计的可靠性和设计精度。

3) 电路分析模型的产生和电路尺寸的自动化调整。符号化的仿真器可以产生描述电路交流特性的分析模型，从而为电路尺寸调节提供依据。

4) 电路结构的交互拓展。电路拓扑结构调整后，符号化仿真器可以快速的产生新的电路特性表达式，交互式的界面可以便捷地绘制不同的电路结构，为电路结构的自动化调整提供基础。

5) 电路参数的迭代调整。电路结构不变时，符号化的分析结果不会发生改变。模拟电路设计常常进行多次电路参数迭代来获得理想的性能，符号化仿真器只需将不同的参数值代入电路表达式求值，而无须重新执行分析过程。

6) 自动生成运算放大器、滤波器等模拟电路模块的行为模型。无须使用具体器件来搭建这些模块，为快速进行电路系统的行为分析提供便利。

7) 辅助模拟可测性分析和故障检测。

相比较数值分析法，符号化分析法与数值分析方各有优势和缺点：

数值分析法的结果是描述电路特性的数据或者图表。由于数值分析理论的成熟发展，数值分析法的仿真速度比较快。数值分析法主要可以用来验证电路的功能特性，检验电路是否符合设计的要求。

数值分析法的缺点在于数值分析法无法非常直观地揭示哪个电路元件对电路性能起决定作用，使用数值分析法需要通过多次实验以及设计经验来找出关键元件；其二，正因为数值分析法提供的结果不是形式化的结果，所以无法自动化地提供给设计者相关的设计修改建议和电路所存在的设计问题；第三，就是每当电路中某个元件的数值发生变化，就必须重新运行数值分析法重新进行仿真。

符号化分析法的分析结果是以符号形式表示的电路特性表达式。一般而言，由于符号化电路分析方法分析的对象是线性电路，所以符号化分析主要面向电路的交流 (AC) 特性。只要电路的结构不发生变化，符号化表达式就保持不变。换句话说，即使电路元件参数值发生变化，符号化分析的结果也并不会发生变化。基于这个特性，符号化分析方法能给模拟电路设计提供很多的便捷，因为模拟电路设计通常需要调节电路元件的参数而不是电路的结构，这样对于符号化的表达式，只需代入不

同的数值进行计算就可，而不需要重新进行符号化分析。

符号化分析同样也有缺点。其一是目前提出的分析方法的运算效率仍然比较低；其二，符号化分析的结果—符号化电路表达式由一系列的符号化的生成项组成，而对于符号化的分析方法来说，这些生成项的数目随着电路尺寸（电路元件个数）的增加而呈指数增长，因此对于这些项的生成和存储都带来巨大的挑战；其三，由于生成项的这个特点，适用于符号化分析的电路的尺寸受到了限制。

1.5 基于拓扑法的符号化电路分析方法

本论文将着重讨论一个基于拓扑分析法的符号化电路仿真器的实现与应用。目前，拓扑法进行符号化电路分析方法被公认为最灵活、最有效的是基于符号流图的分析方法，但是该方法对于分析电路的规模还是有着比较大的限制[2]。[27][28]中提出了一种全新的拓扑分析方法，该方法发现组成电路表达式的生成项与电路网络的生成树有直接联系，且论证了对于一些特定电路元件的枚举方法，其局限有两处：一是[27][28]中所提出的方法适用于特定的受控源，对所有受控源和理想运算放大器没有很好的解决方法；二是论文没有提出比较好的计算机自动化分析的算法，在实现上有一定的困难。基于这种全新的方法和思想，我们对其进行了改进和严格的代数证明，将该理论进行了很好的总结，提出了非常清晰的生成树枚举理论，使该方法可以使用于较一般的线性电路；同时在本文中，我们将具体讨论该方法的计算机实现，使之可以真正运用于具体模拟电路模块的自动化分析。

1.6 引言小结

由上述的比较我们可以看出，符号化分析法本身具有很多简化模拟电路设计的特性，其主要的问题在于没有理想的高效算法适合目前计算机的计算能力；而对于数值分析法而言，虽然算法的效率高，对大规模的电路也有很好的支持，但是其本身的特点局限了其对模拟电路设计的自动化支持，尤其是自动提供电路优化和关键元件的功能。因此，如果我们希望可以找到一个相对理想的符号化模拟电路仿真算法，使其可以支持对较大规模模拟电路设计，从而由此获得符号化分析对于电路设计的天然的良好的支持。

本文将对一个全新的符号化模拟电路仿真算法及其实现细节进行讨论，为符号化模拟电路分析方法的研究探索一个新的发展可能。

2　基于拓扑法的符号化模拟电路分析定理与算法

本章中我们将基于[27][28]，提出一个完整符号化模拟电路分析方法（发表于[47]），并提出一种新的计算机实现算法。这种分析方法基于电路的拓扑结构，通过对相应由电路转化而成的有向图的操作得到最后的分析结果。

2.1　基于拓扑法的符号化模拟电路分析定理

首先讨论基于电路拓扑的符号化模拟电路分析定理，该定理基于严格的代数推导。本节主要介绍定理的内容，定理证明见附录 1。

本文所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，先要将电路转换构建成相应的有向图；然后根据分析定理进行运算。

2.1.1　基本前提条件

(i)．电路允许含有 5 类元件：阻抗（Z），导纳（Y），四种类型的受控源（压控电压源 VCVS， 流控电流源 CCCS，压控电流源 VCCS，流控电压源 CCVS），独立源以及理想运算放大器。

(ii)．电路中只含有一个独立源。由于线形电路具有叠加性的特点，即每个独立源对于电路的贡献可以通过简单相加得到。为了简化问题，我们提出这个前提条件。当电路中含有多个独立源时，只需对不同的独立源分别进行符号化分析，然后将结果叠加即可。

(iii)．每个控制源只控制一个受控源。

(iv)．每个受控源只受一个控制源控制。

(v)．电路中不含反馈电路。

如果待分析的电路符合 2.1.1 中所述的条件，那么就将电路根据以下规则转换成有向图，从而可以进行下一步分析。

2.1.2　有向电路的图的构建规则

电路图在分析前必须先转化成一个有向图，转换按照以下规则进行。图 2 是一个转化的例子，其从电路 2-1(a)到有向图 2-1(b)的例子。

(i). 受控源或者独立源转化成有向边。电压源的方向沿着正极（+）到负极（-），电流源的方向沿着电流的流向；

(ii)．在有向图中，对应电压控制支路添加一条有向边，例如图 2(b)中的U_4；对应电流控制支路添加一个结点和一条单独的有向边，例如图 2(b)中的I_1；

(iii)．所有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；

(iv)．理想运算放大器用零器（Nullor）建模，每个零器（Nullor）由一对零阻器（Nullator）和一个泛阻器（Norator）组成，见图 3；

(v)．将输入输出端（I/O）抽象成一个受控源， 例如图 2(b)中的U_o和I_{so}。

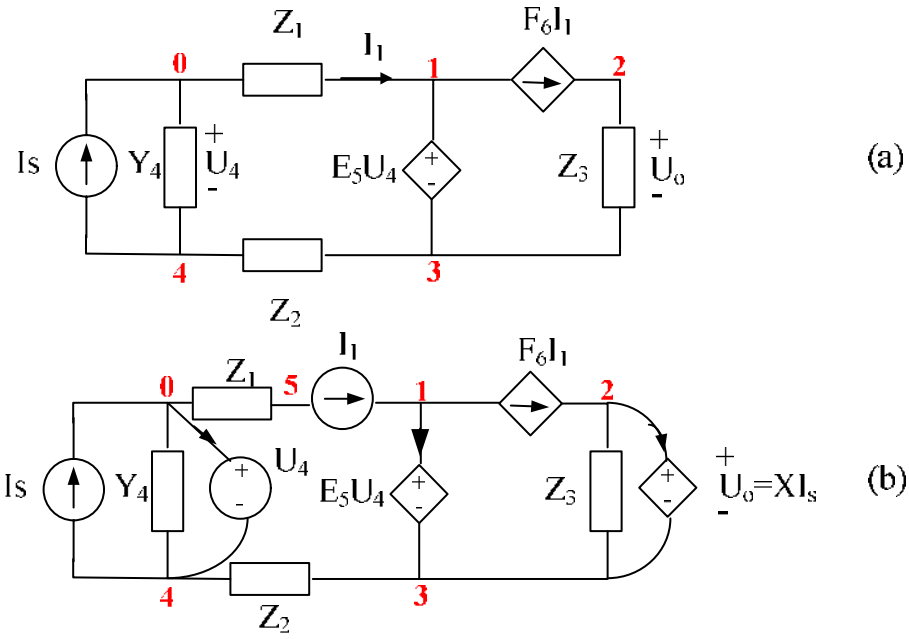


图 2 有向图构建规则举例。

(a) 待分析电路；(b) 转换后的电路。

Fig.2 A circuit example

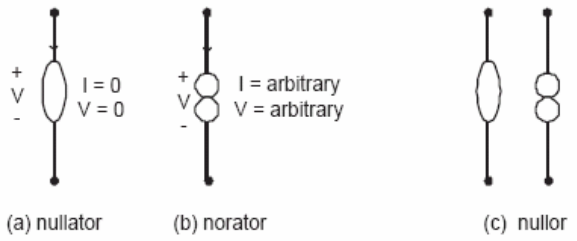


图 3 理想运算法大器模型

零阻器 (b) 泛阻器 (c)零器

Fig.3 Ideal operational amplifier model

规则（v）是算法中非常重要的一步转换。对于线形电路，当电路结构固定时，电路的输出取决于电路的输入，输入不同则输出值也不同。我们考察电路传输函数:

$$H(s)=\frac{Output}{Input}$$

我们把分母移到等式的左边有：$H(s)*Input=Output$。

发现，电路输出受电路输入控制。所以可以将输入输出建模成一个受控源，而电路传输函数就是这个受控源的增益，也就是我们需要得到的符号化分析结果；同样也可以将输入看作受输出的控制，而这个受控源的增益也就是传输函数的倒数。一般而言，对于输出电压（电流），当把它看作一个理想电源时，流过它的电流（电压）应该为零。而对于受控源而言，有如下规定：

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（VCCS）：$I_i=G_{i,j}U_j$，$I_j=0$;

流控电压源（CCVS）：$U_i=E_{i,j}I_j$，$U_j=0$;

可以看出我们对控制源有特别的要求，对于电压（电流）控制源要求其流过的电流（电压）为零。基于以上规定，我们将输出作为控制电源而将输入看作受控电源，这样传输函数就是这对受控源增益的倒数。例如图 2 所示，我们将输入输出看作一个压控电流源（VCCS），其增益为 X，那么我们所要求的传输函数 $H(s)=\frac{1}{X}$。

2.1.3 基于拓扑法的符号化模拟电路分析定理

我们提出以下的定理，对转化成有向图的电路进行分析。该定理基于电路的拓扑结构，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。值得

一提的是，并不是所有的生成树都对应有效的生成项。本节所提出的定理将定义何种生成树是有效的，同时定义生成树如何对应成生成项，最终如何得到分析结果。

定理一（有效生成树定理）：一棵有效生成树不包含 Nullor，不包含 VC 和 CS 器件的对应边，但是必须包含所有的 CC 和 VS 器件的对应边。

定理二（生成树对应生成项定理）：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}器件（也就是所有的导纳）直接成为乘积项，而所有的CC和VS边权重为 1，均不在乘积项中出现。

定理二定义了如何由生成树得到的生成项。值得一提的是，Z 器件在生成树对应的生成项中以倒数的形式出现，也就是说要把阻抗变为导纳的形式。

定理三（有效生成树对定理）：有效生成树对由有效生左生成树和有效右生成树组成。

(i)．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；

(ii)．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；

(iii)．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。

(iv)．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。

上述定理定义了有效生成树对的规则，我们举例说明。

例如一个流控电压源（CCVS）元件，在有向图中对应两条有向边，当考虑有效生成树对时，我们有这样几种选择（CC, CC），（VS，VS）和（CC，VS），其中（a,b）表示 a 在有效左生成树中、b 在有效由生成树中。由定理三（iii），我们可得，对于 CCVS 元件，要么出现（CC，CC），（VS，VS）的组合，也就是 CC 和 VS 边同时出现在左、右生成树中；要么（VS，CC）组合，也就是 VS 边出现在左生成树

中、CC 边出现在右生成树中。

又如一个压控电流源 （VCCS）元件，同样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。由定理三（iv）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。

同理，对于 CCCS 元件，要么 CC 边同时出现在左、右生成树中，要么 CC、CS 边成对出现（CS 在左生成树、CC 在右生成树）；对于 VCVS 元件，要么 VS 边同时出现在左、右生成树中，要么 VC、VS 边成对出现（VS 在左生成树、VC 在右生成树）。

有效生成树其实是有效生成树对的一个特例，即左、右生成树完全一致，所以我们就将生成树对约化为一棵生成树。

由定理三，我们可以得到一个关于生成树对中左、右生成树所含边的类型的推论。

推论一（生成树边类型推论）：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）。

推论一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。

定理四（生成树对应生成项定理）：由生成树对得到的生成项由下列四部分的乘积所得：

(i)．生成项符号（+ 或者 -）；

(ii)．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）；

(iii)．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。生成项符号由定理五确定；带符号的受控源增益由定理六确定。

定理五（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：

$$VCVS \leftrightarrow -E_{j,k}$$

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

可以发现，只要 VS 边出现，增益前就要带上负号。

定理六（生成树对应生成项符号定理）：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)[1]。A_L和A_R的行数相同（等于电路结点数减一）；它们的列数（等于生成树边数，即结点数减一）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中；若两条边成对出现，则它们分别出现在A_L和A_R中，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有向的，对应电路图中的有向边（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$|A_L|*|A_R|$($|A_L|$、$|A_R| = \pm 1$, [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[1]）。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。定理六指出，生成项的符号由约化入射矩阵行列式的乘积决定，故需进行行列式计算，但在 2.2 中我们将提出一个简单的算法，在有效生成树对选择的过程中直接计算行列式值，而无须在选择完了之后（即得到A_L和A_R）再单独计算。

定理七（生成项零和定理）：所有生成项之和为零（式 2-1）。

定理七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；由定理三、四、五可知，该未知量是某些生成项的组成部分。由定理七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）。

$$X(\sum_{i=1}^{m_1} t_i) + (\sum_{j=1}^{m_2} T_j) = 0 \qquad (2\text{-}1)$$

$$X = -\frac{\sum_{j=1}^{m_2} T_j}{\sum_{i=1}^{m_1} t_i} \qquad (2\text{-}2)$$

对于上述定理得到的生成项，它们都是不可互相消去（cancellation-free）的，即它们都是最终结果中必须出现的项，都不是冗余的（证明见附录 1）。这是目前所有符号化模拟电路分析法都没有的特性，由于所有生成项都非冗余，所以没有不必要的开销用于冗余项的生成，从而提高了分析的效率。

2.1.4 举例

我们通过一个简单的例子来说明2.1.3所提出的分析定理。我们分析如图4(a) 中的电路，其由一个电压源、一个电阻和一个电容串联而成，求电容两端的电压值。由 Kirchoff 定理可以非常快地得到频域的传输函数 $H(s) = \frac{1}{1+RCs}$。

有上文提出的分析方法，我们先构建该电路对应的有向图 4(b)，其中我们将输入输出看作一个压控电压源（VCVS），输出控制输入。

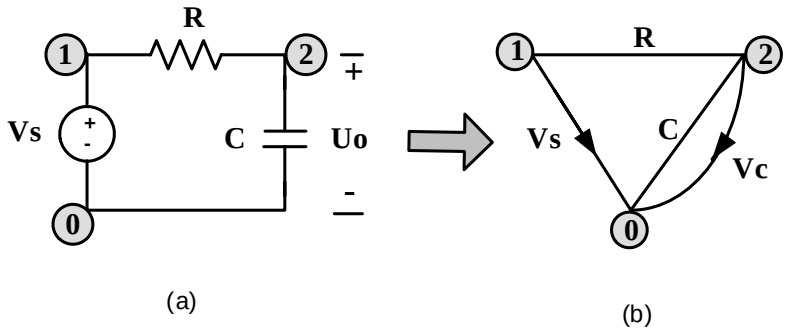


图 4 拓扑法符号化模拟电路分析法举例。
(a) 待分析电路 (b) 电路有向图

Fig4. An example for the enumeration rules

然后根据定理一、三，我们可以得到图 5 中所示的两棵有效生成树和一对有效生成树对。他们所对应的生成项分别为 Cs*1，1/R*1 以及-1/R*X。

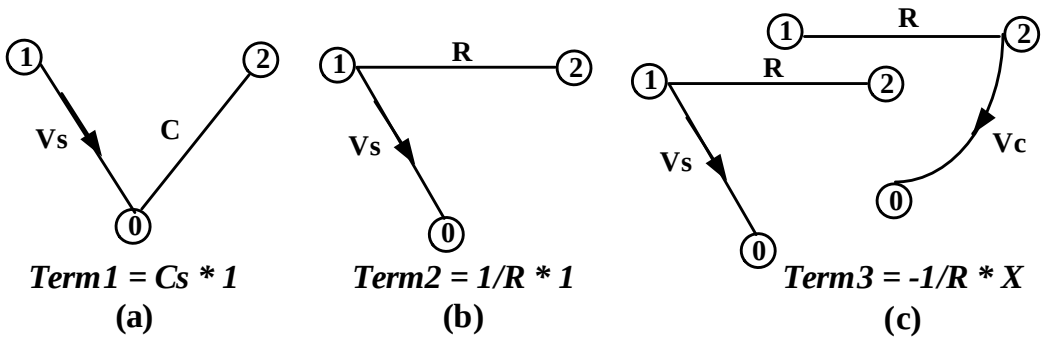


图 5 有效生成树、生成树对及对应的生成项

有效生成树 1 (b)有效生成树 2 (c)有效生成树对

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

由定理七，联立这三项得到等式 $Cs*1+1/R*1-1/R*X=0$。

解方程得到传输函数 $H(s)=\frac{1}{X}=\frac{1}{1+RCs}$，与我们观察的结果一致。

2.2 基于拓扑法的符号化模拟电路分析算法

2.1 节中我们给出了基于电路拓扑结构的符号化电路分析定理，按照这个定理，我们可以计算出电路传输函数的符号化表达式。从以上定理可以看出，最简单的实现方式是枚举出每一棵生成树，检验它们是否有效。但是，一般生成项的个数与电路规模呈指数关系，而生成树的数目就更多了，当电路规模变大时，生成树枚举需要大量的运算时间和存储空间（NP 问题），这也正是目前符号化电路分析的主要难点。

在本节中，我们将引入一个新的概念：二分判定图（Binary Decision Diagram）。二分判定图是现代电子设计自动化领域非常重要的一个数据结构，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形的结果。基于这个特性，我们设计了一个计算机算法，将生成树枚举变为有向图的约化过程，并在此过程中使用二分判定图，将指数级复杂度的问题简化，从而可以适应计算机的运算能力。

2.2.1 二分判定图

二分判定图由最初由 Aker 在[32]中提出，目的是为了将布尔函数转换成判定图，

Lee 在[33]中首先提出了一个直线化的转换程序。到了 1983 年，Byrant 将二分判定图概念有效地使用到门级电路的研究领域，提出了简化有序的二分判定图（Reduced Ordered Binary Decision Diagram, ROBDD, 往往被简称为 BDD，[34]）的数据结构并提出了基于这个数据结构的布尔函数操作的算法[35][36]。下列是一个表示布尔函数的二分判定图：

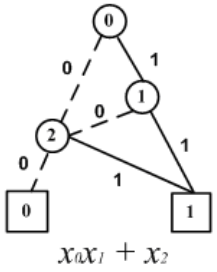


图 6 二分判定图

Fig.6 Binary Decision Diagram

这是一张有向无环图，根结点一个，终结点只有 0 和 1 两个（方型结点）。每条虚线（0-分枝）分枝表示该结点代表的变量取值为 0 后的布尔表达的结果，每条直线（1-分枝）分枝表示该结点代表的变量取值为 1 后的布尔表达的结果，从图上可见当$x_0=0$、$x_2=0$ 时布尔函数的值为 0；当$x_0=1$、$x_1=1$ 时，布尔函数的值为 1；而当$x_0=0$ 时，布尔函数的值取决于x_2的取值。从上例中可以看出，这个 3 个变量组成的布尔函数使用二分判定图只需 3 个结点，若使用真值表（truth table）将会有 $2^3=8$ 项。

二分判定图的结构具有以下几个优点：

1. 最大程度地实现子判定图结构的共享，否则这些子判定图结构的表达形式与问题规模呈指数增长的关系。

2. 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。

3. 所有的布尔操作可以通过简单的图论算法实现，这些算法的复杂度与问题规模呈多项式关系。

目前，二分判定图在数字集成电路计算机辅助设计工具中被广泛地使用，应用的领域包括电路行为验证、测试向量生成、故障仿真和逻辑综合[39]。

2.2.2 有向图约化算法说明

为了改进由生成树枚举带来的巨大运算量，我们提出一个有向图约化算法，在约化的过程中得到所有的有效生成树和生成树对，同时以二分判定图的方式进行存储，我们将这个过程称为有向图约化算法（Graph Reduction），将生成的二分判定图称为图约化判定图（Graph Reduction Decision Diagram, GRDD）。

我们用一个例子来说明如何通过有向图的约化得到 GRDD，以及如何从 GRDD 得到电路的传输函数。在这个例子中，我们将有效生成树看成由两棵相同生成树组成有效生成树对，这样有效生成树与有效生成树对可以统一进行处理。

考察图 4(a)中的电路，我们先构建有向图 4(b)。为了方便处理，我们将该有向图分成两个子图（见图 7），分别用于生成有效左、右生成树。由 2.1.3 节推论一我们可知，对于有效生成树对的左、右生成树，它们所含的边类型是固定的，所以我们在处理前先将不会出现的边从图中删去，这样可以使处理过程更为清晰。对于本例，有向图中含 Y、Z、VC 和 VS 四类边，原始图处理时，在左子图中删去 VC 边，而右自图则保持不变。

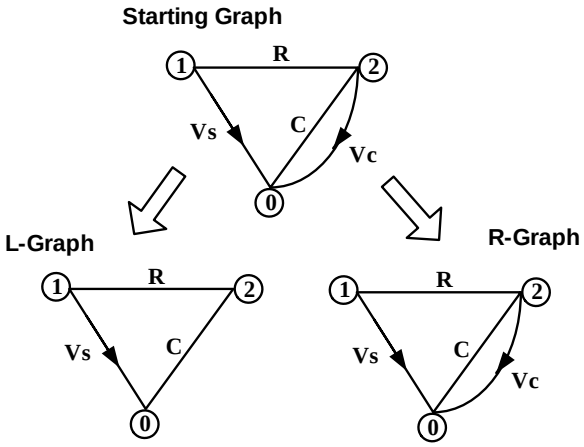


图 7 有向图拆分（原始图、左图、右图）

Fig.7 Graph Splitting

有向图约化过程从图 7 中的两个拆分后的子图开始。关于这个电路图，有三个符号，R 代表电阻，C 代表电容，X 代表输入输出受控源（VCVS）的增益，也就是我们要求的电路传输函数的倒数。在处理前，我们先给这些符号设定一个处理顺序，我们设 X < R < C，也就是先处理 X，然后 R，最后处理 C。

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，同时根结点有一个代表符号，我们最先处理 X，

所以根结点代表 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[33]。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。

考察本例，图 8 显示了整个有向图约化的过程，以及相应构成的 GRDD（见图 9）。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，另外我们在该边上关联上一个符号（+），这个符号将最终决定生成项的符号，具体如何确定该符号是正还是负，我们将在 2.2.4 中讨论；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，相关联的符号为-。处理完符号 X 后，接着处理符号 R，然后符号 C。表格 1 列举了取舍各种类型边相对应的操作。我们再来考察如何生成终结点，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。例如图 8 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，见[33]），

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。

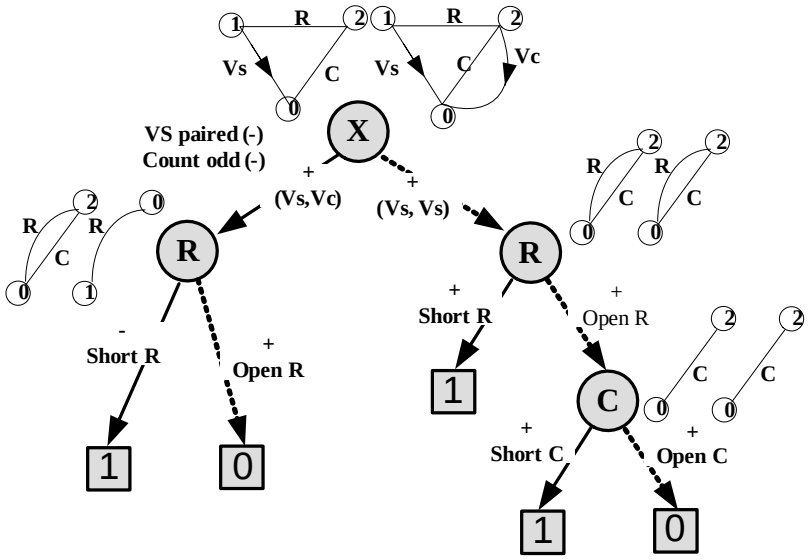


图 8 有向图约化 GRDD 构建过程

Fig.8 Construction of GRDD by graph reduction manipulation

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

表格 1 列举了不同类型元件边在子图中进行约化的具体操作。需要指出的是，对于受控源，在选择该元件时，其控制边和受控边会在约化子图中成对出现（分别出现在左、右约化子图中），所以，我们会在约化子图中分别选择对应边，但是对于 VS 和 CC 边，它们在左、右子图中都会出现在，而在它们与其他边成对出现时，它们只会在一个约化子图中出现，所以对于在另一个子图中的 VS 和 CC 边，必须移除它们，例如选取 VCVS 元件时对右子图的操作。

这一节论述了有向图约化（Graph Reduction）算法的基本思想，图约化判定图（GRDD）的构建以及如何从图约化判定图中获得电路符号化表达式生成项的方法。

2.2.3 图约化判定图

当约化过程结束时，我们构建成了一个图结构。这是一个有向无环图，有一个根节点（X），两个终结点（0，1），所有的非终结点均有两个分枝（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，该若一个结点的 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。如图 9(b)，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。根据定理七，我们就可以得出该电路的传输函数。

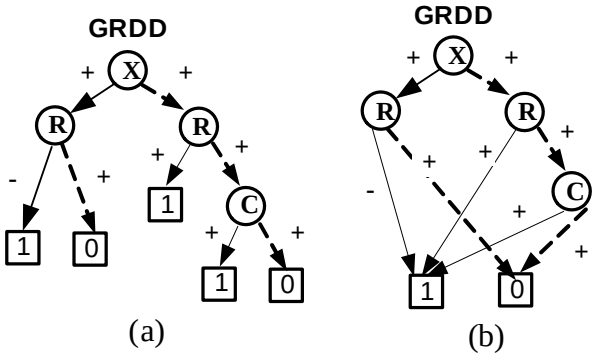


图 9 图约化判定图（GRDD）

(a) GRDD　(b)约化后的 GRDD

Fig.9 Graph Reduction Decision Diagram (GRDD)

2.2.4 有向图约化算法

根据上节论述的分析方法，我们提出计算机算法来实现分析的自动化。

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**（见 2.2.4），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。

我们之所以总是从输入输出符号进行操作，目的是要将这个我们要求的未知量符号（X）放在图约化判定图的根节点。这样，这个结点的 1-分枝所指向的判定子图代表所有包含 X 符号的生成项($\sum_{i=1}^{m_1} t_i$)，而 0-分枝所指向的判定子图代表所有不包含 X 符号的生成项($\sum_{j=1}^{m_2} T_j$)。由式 2-3，当图约化判定图构建完成后，我们可以非常简单地计算出未知量 X 的表达式。

$$X(\sum_{i=1}^{m_1} t_i) + (\sum_{j=1}^{m_2} T_j) = 0 \qquad (2\text{-}3)$$

该算法可以按照深度优先的顺序构建图约化判定图，这样可以比较简单、有效地通过计算机程序来实现分析的自动化。

2.2.5 生成项确定算法

在 2.1.3 中我们提到，每个生成项都有一个符号，这个符号由代表左、右生成树的入射矩阵的行列式值所决定。然而该符号的计算并不需要通过左、右生成树建立后，再计算相应的入射矩阵的行列式。我们发现，在图约化判定图（GRDD）的构建过程同时可以逐步地计算出生成项的符号。本节中我们提出生成项确定算法，该算法基于本分析法的理论推导（证明见附录 1），可以在图约化判定图构建算法的每一步中逐次执行而无需多余的运算。

在算法实现的过程中，我们采用一个 2 维数组来记录左、右约化子图中的每条边的两个结点（见图 10）。设 e(1)（第一维）、e(2)（第二维）分别为每条边 e 的两个端点，每条边都从 e(1)指向 e(2)，包括 Y/Z 类型边（固定一个方向）。当我们在选择一条边的时候，我们需要将两个边结点汇合成一个结点同时改变其他边中相应的结点。为了操作的方便，我们总是将结点号大的点汇合到结点号小的点，这样总是保持较小的点在子图中。我们称一条边的方向是正向的当且仅当 e(1) < e(2)；一条边是的方向是反向的当且仅当 e(1) > e(2)。我们提出如下的符号确定算法，应用于 GRDD 的构建过程之中。

e_1	e_2	e_3	e_4	e_5	
0	0	0	6	2	.....
2	3	4	2	5	.....

图 10 表示约化子图的二维数组

Fig.10 A subgraph represented in 2D-array

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（v_1, v_2），其中v_1 < v_2，即改变为正向），现将该边从表示约化子图的数组中删去，然后将数组中所有

边结点为v_2的点改成v_1。考察仍保存在数组中的结点，计算结点号小于v_2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。

该算法是通过用高斯消去法计算行列式值的方法推得，具体证明见附录 1。

2.3 本章小结

本章中我们具体论述了基于电路拓扑结构进行符号化模拟电路分析的方法及其相应定理，并提出了计算机实现的算法，同时引入约化子图判定图（Graph Reduction Decision Diagram）的数据结构用以辅助算法的执行和生成项的存储。本章主要介绍分析方法，该方法基于严格的代数证明，具体证明见附录 1。

3　符号化模拟电路仿真器的实现

本章将讨论模拟电路仿真器的实现细节，主要包括仿真器的结构，为提高效率所采用的图约化判定图的操作以及实现方法。通过这些方法，第二节中所提出的定理与算法可以有效地通过计算机程序得以实现。

3.1　仿真器结构

我们的符号化仿真器由三部分构成（见图 11），包括网表解析器、符号化分析引擎（内含图约化判定图 GRDD）和一个求值运算器（提供各种数值分析结果，符号化电路特性等）。

仿真器的输入为 SPICE 网表，通过网表解析器获取电路信息，然后符号化分析引擎按照第二节提出的算法构建分析电路相应的 GRDD，最后通过求值运算器得到所需要的分析结果。值得注意的是，如果电路的结构不发生变化，GRDD 的结构不变；调解电路参数只需重新运行求值运算器即可。

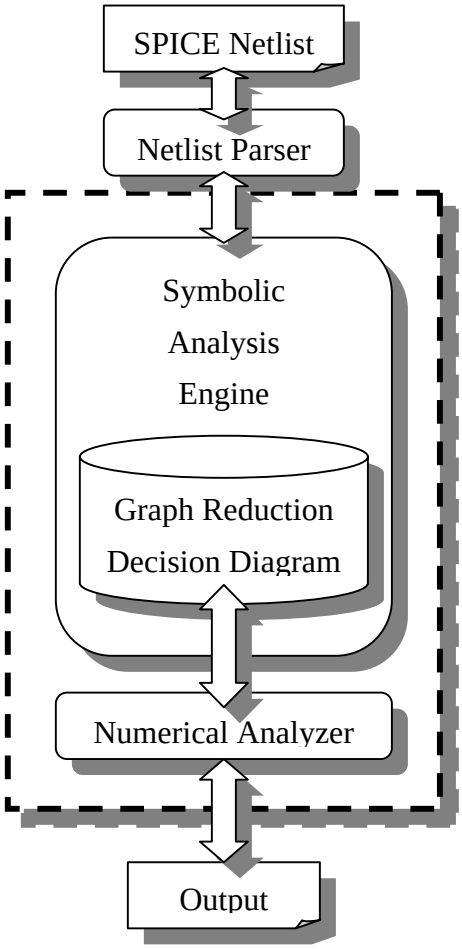


图 11 符号化分析器结构

Fig.11 The Symbolic Circuit Simulator

网表解析器由 PCCTS (Purdue Compiler Compiler Tool Set) 生成。PCCTS 具有类似于编译器生成工具 Lex 和 Yacc 的功能，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。

符号化分析引擎跟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。在图约化判定图的结构中，未知量 X 总是位于判定图的根节点位置（第一个处理的符号），这样处理的目的是一旦图约化判定图建立以后，所有含有未知量 X 的生成项将由根节点的 1-分枝引导，而所有不含未知量 X 的生成项则完全由根节点的 0-分枝引导，这样未知量可以通过分别计算根节点两个分枝所引导的子图约化判定图得到，(传输函数等于未知量倒数的负值，也即等于 1-分枝子图除以 0-分枝子图的负值)。如图 9（a）所示的 GRDD，

其包含了三个有效的生成项，它们可以构成一个方程（X 为未知量）：

$$-X \times R^{-1} + R^{-1} + Cs = 0$$

这样我们要求的传输函数表达式即为 $T(s) = \frac{1}{1+RCs}$。

数值分析器基于图约化判定图提供电路相应的特性结果，比如不同频率下传输函数的振幅和相位、符号化表达式所含的生成项数以及一些分析过程中的统计数据。

3.2 图约化判定图共享 （GRDD Sharing）

共享是所有二分判定图（BDD）相关问题所共有的提升效率采用的方法，对于图约化判定图亦如是。

3.2.1 图约化判定图共享如何提高算法效率

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，这时就可以将这两个相同结构的约化子图进行共享，即在内存中保留一个副本。

由于每个 GRDD 的节点都关联一对约化子图，而每个 GRDD 节点所引导的判定子图包含了对应约化子图中所有有效生成树对（注意这是局部子图中的有效生成树），所以一旦一个新生成的 GRDD 节点所关联的约化子图与一个已经生成的 GRDD 节点一致，那么这两个节点所引导的判定子图也一定具有相同的结构，因此在内存中可以共享所有的子判定图结构。

第二章中提出的 GRDD 生成算法是以深度优先来构建 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，这样就可以减少一定的运算量同时也节省了内存空间。

3.2.2 什么结构可以共享

在图约化判定图（GRDD）的构建过程中，所有的约化子图和 GRDD 节点均可以进行共享。

3.2.3 如何共享约化子图

在实现时，我们使用一个二维数组来存储每一个约化子图，其中每一维分别保存图中每条边的起点和终点，数组的维数与子图中的边数相等。若两个约化子图完

全相同，则记录它们结构的两个数组也完全一致。

我们使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。

我们分两步来实现约化子图的共享，所采用的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。为了减少冲突的可能性，哈希表的尺寸设为一个质数：

第一步：使用 HashTreeID()函数找出二分哈希树的入口。HashTreeID()使用约化子图所含的边数和点数来决定，具体见下：

```
unsigned int HashTreeID(int SubNumEdge, int SubNumVertex)
{
    return (((unsigned int )SubNumVertex - 1) * NumGraphEdge
            + (unsigned int) SubNumEdge - 1) % Prime_Number;
}；
```

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，则使用 Hash()函数来建立二分哈希树，Hash()函数的目的还是产生一个 ID 用于寻找该约化子图的入口，具体如下：

```
unsigned Hash (unsigned int i)      // generating the hashIDs
{
    for(j = 0; j < SubNumEdge; j ++)
    {
        if edge j is not self-loop
            result += (one node of edge j) ^ (the other node of edge j);
    }
   return ((result ^ i) %A_Prime_Number);
}
```

值得一提的是，Hash()函数仍可能对于不同的约化子图产生相同的 ID，所以我们要使用一个 compare()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），比较函数定义如下：

```
int compare(graph g1, graph g2)    // g1 and g2 are two graphs with same hashID.
{
```

```
for(j = 0; j < num of the edges; j ++)
{
    if(edge j of g1 != edge j of g2)
        return (&g1 - &g2);      // return when g1 and g2 are not identical
}
return 0;
}。
```

图 12 描述了如何进行约化子图共享的过程，约化子图哈希表的结构和决定约化子图哈希值的元素。

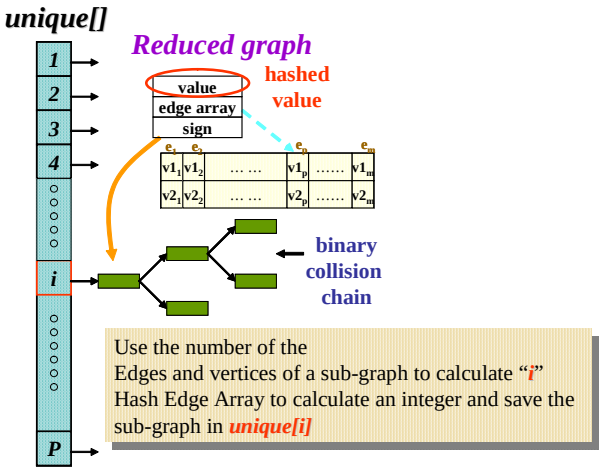


图 12 约化子图哈希策略

Fig.12 Reduced Graph Hashing Schemes

3.2.4 如何共享关联有相同约化子图的 GRDD 节点

每个 GRDD 节点由 3 部分组成：一对约化子图，一个代表的电路元件符号，GRDD 两条分枝关联的符号。

我们仍然使用哈希表来实现 GRDD 节点的共享。在比较两个 GRDD 的节点时，我们必须比较其各自的 3 个组成部分。由于约化子图已经进行了共享，所以只需比较指向约化子图的指针即可；电路元件符号及分枝符号分别由 3 个整型数表示，故可直接进行比较。我们仍然通过两步来进行 GRDD 节点的共享，哈希树入口由 HashTreeID()函数得到；节点哈希值 hashID 由 Hash()函数生成；compare()函数用来比较具有相同 hashID 的 GRDD 节点，这三个函数的具体定义如下：

```
unsigned tpddnode::HashTreeID()
{
    return ((((unsigned)pLTree) ^ ((unsigned)pRTree))) % TPDD_PRIME;
}

unsigned Hash(unsigned i)
{
    int value = ((((int)pLGraph << 3) + ((int)pRGraph << 15)) ^ (~symbIndex));
    return (value ^ i) % TPDD_PRIME;
}

integer Compare(sddnode * p, sddnode* n)
{
    if (p->index != n->index or p->pLGraph != n->pRGraph
            or p->pRGraph != n->pRGraph)
    {
        return (p - n);
    }
    return 0;
}
```

图 13 描述了如何进行约化 GRDD 节点共享的过程，节点哈希表的结构和决定 GRDD 节点哈希值的元素。

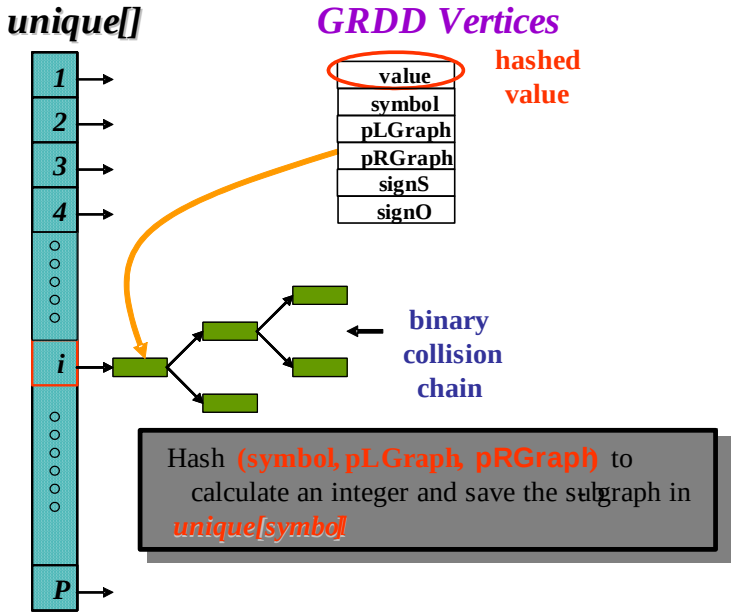


图 13 GRDD 节点哈希策略

Fig.13 GRDD Vertices Hashing Schemes

3.2.5 约化子图的同构和项等价约化子图

事实上不仅带有相同约化子图的 GRDD 节点因为引导相同结构的判定子图而可以进行共享，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。所谓的同构约化子同和项等价约化子图的定义如下：

约化子图同构：假设图 *G1 = (V1, E)*, 图 *G2 = (V2, E)*, 有集合 S1 和 S2，其中 *S1* = {*T* | *T* 为 *G1* 的生成树。}, *S2* = {*T* | *T* 为 *G2*’ 的生成树。}。图 *G1* 和 *G2* 同构当且仅当 *S1 = S2*。图 14 列举了同构子图的例子（其中图 G=(V, E), V 为图 G 的点集，E 为图 G 的边集。 V = {v0, v1, …, v|V|-1}, E = {e0, e1, …, e|E|-1}。边的处理顺序为 e0 > e1 > … > e|E|-1。因此子判定图以代表边 e0 的 GRDD 节点为根节点。)。为了说明的方便，我们在列举同构约化子图时只列举单独的同构子图而非同构子图对，所谓的同构子图对就是左、右子图分别同构的两对约化子图。

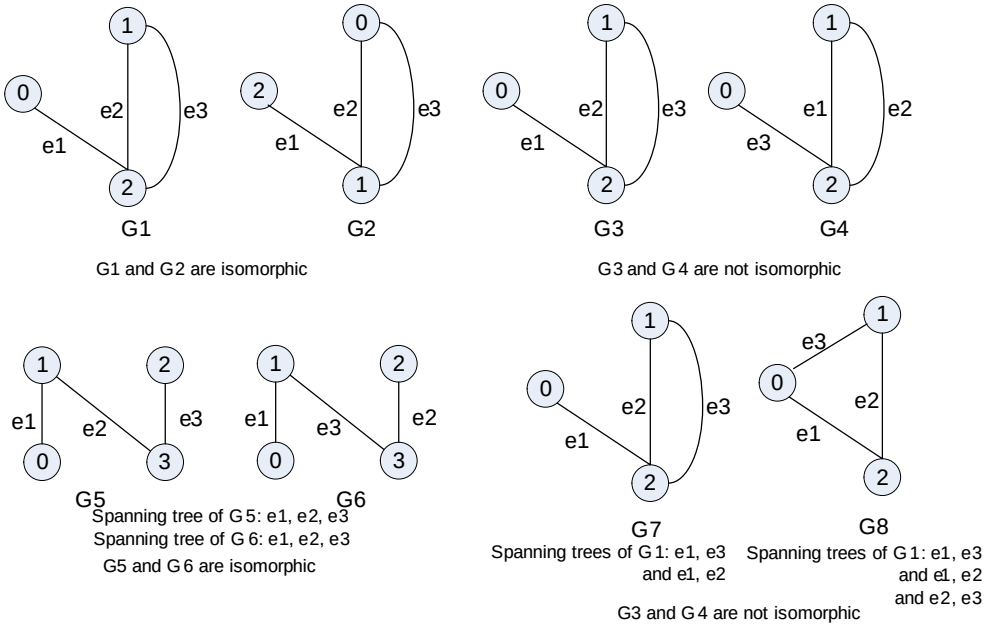


图 14 同构约化子图举例

Fig.14 Example for Isomorphic Reduced Graphs

图 15 说明了同构子图对如何生成相同结构的图约化判定图。

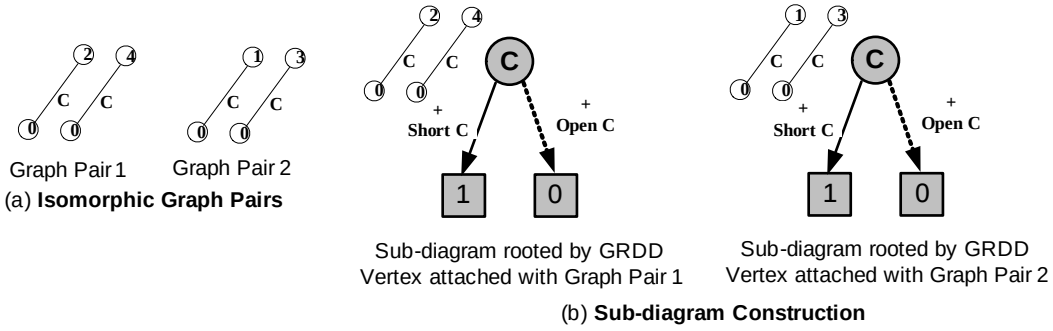


图 15 同构约化子图生成相同结构判定子图举例

Fig.15 Example for Same Sub-diagram generated by Isomorphic Reduced Graph Pairs

图 16 说明了如何共享 GRDD 节点。由虚线框所圈出的节点在进行了 GRDD 节点的共享之后可以被释放，事实上由于有了 GRDD 的共享，该结构除了其根节点会产生之外，其余的子节点都无需再生成。

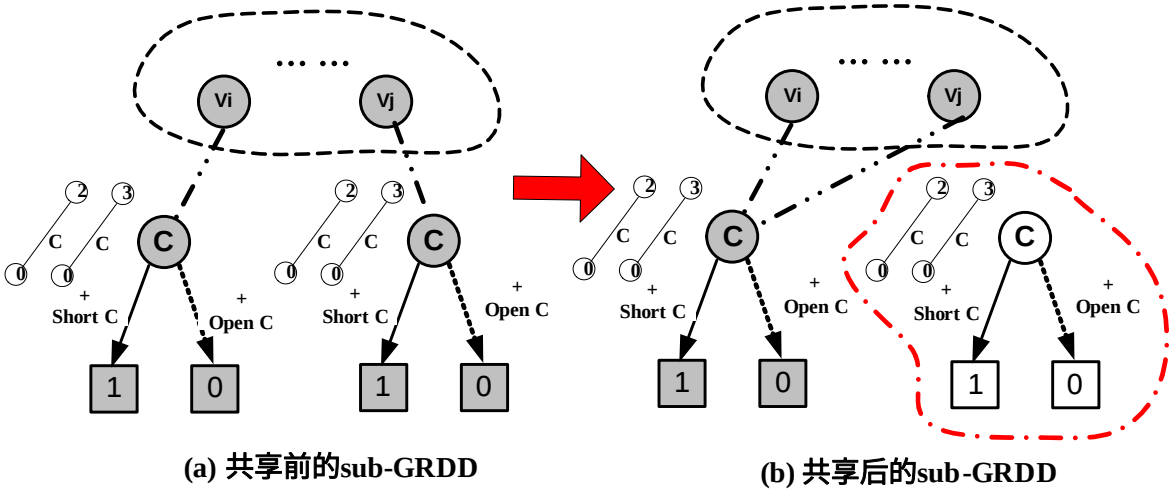


图 16 GRDD 节点共享举例

Fig.16 Example for GRDD Vertices Sharing

除了同构子图有可能产生结构完全一致的判定图之外，还有一种子图也有可能产生结构完全一致的判定图，从而得到同样的生成项，我们将其称为项等价约化子图，图 17 举例了这种子图。

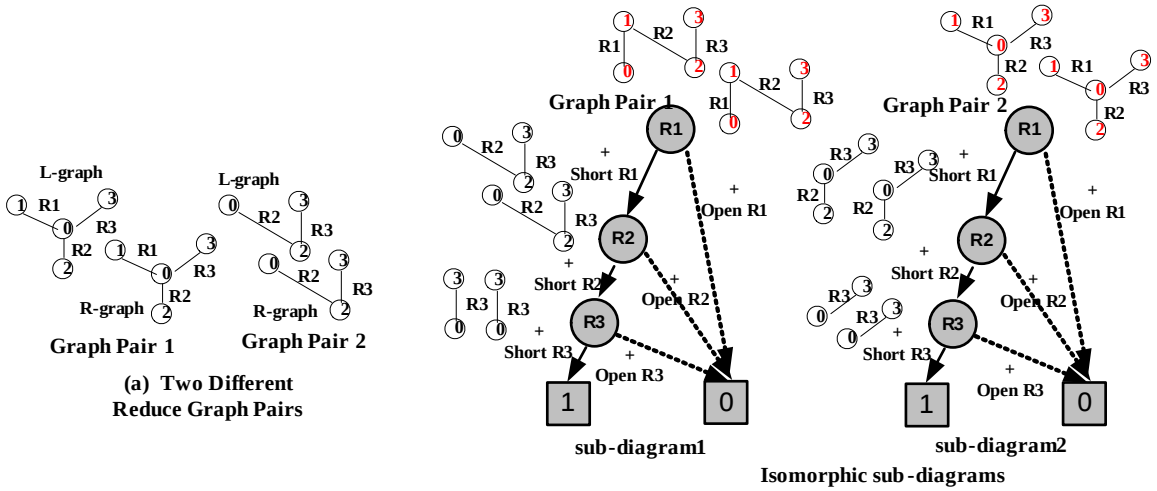


图 17 项等价约化子图举例

Fig.17 Example for Term-equivalent Reduced Graphs

我们可以发现，关联有同构约化子图或者项等价约化子图的 GRDD 节点所引导的判定子图结构完全一致，因此它们完全可以共享。但是对于关联有同构约化子图

或者项等价约化子图的 GRDD 节点其共享的方式将于关联有相同约化子图的 GRDD 节点不同。

3.2.6 如何共享关联有同构和项等价约化子图的 GRDD 结点

在共享具有相同结构的约化子图时，我们只需要比较代表的两个子图的数组即可。但是当比较同构的约化子图时，我们无法直接通过比较子图数组得到结果，比较算法如下：

子图同构比较算法：

设图 G1, G2 分别由数组 A1[n][2]和 A2[n][2]表示，其中 Ai[j][0]中保存图 Gi 第 j+1 条边的起点，Ai[j][1]保存图 Gi 第 j+1 条边的终点，n 为两张图的边数。且图 G1 和图 G2 的边数与点数均一致，否则 G1 与 G2 不可能同构。

Step1. 设置一个置换数组 Z[n], 初始化使 Z[i] = -1 (其中 i = 0, 1, …, n-1)。

Step2. for (i = 0; i < n; i ++)

```
{   if (Z[A1[i][0]] == -1)
        Z[A1[i][0]] = A2[i][0];
    else
    G1 与 G2 不同构。} 转到 Step3。
```

Step3. for (i = 0; i < n; i ++)

```
{   if (Z[A1[i][1]] != -1 && Z[A1[i][1]] == Z[A2[i][1]])
        continue;
    else
        G1 与 G2 不同构。}转到 Step4。
```

Step4. G1 与 G2 同构。

该算法在新建每个 GRDD 结点时都必须执行一次，并且每次执行时均必须分配存储空间给置换数组，会很大程度上影响算法的执行效率。

另外，对于项等价约化子图，我们几乎无法直接通过比较两个图来得到结论。由此对 GRDD 节点的共享带来了很大的困难。

虽然在 GRDD 生成的过程中无法完全实现节点的共享，我们仍然可以在 GRDD 生成后执行判定图约化操作删去结构重复的判定子图得到紧凑约化的图约化判定图，其子图中没有结构完全相同的存在。

3.3 图约化判定图约化 （GRDD Reduction）

GRDD 的约化主要是为了处理由同构约化子图和项等价约化子图所产生的相同结构的判定图。我们通过对判定图节点进行编号的方式来实现判定图约化。每个 GRDD 节点的编号由其子节点决定。我们设终节点 0 的编号为 0，终节点 1 的编号为 1，然后对每个非终节点进行重新编号，若当前要进行编号的节点的字节点与已编号节点的字节点编号相同，则其编号与已编号节点相同；否则其编号数顺次递增。此外，对于子节点均为终节点 0 的节点，说明其不会引导出有效的生成项，因此可以将该节点简单地替换为终节点 0，同时释放空间，以此类推。经过这两部分的操作，我们可以得到约化的判定图，对于一定的符号处理顺序，判定图的结构唯一。

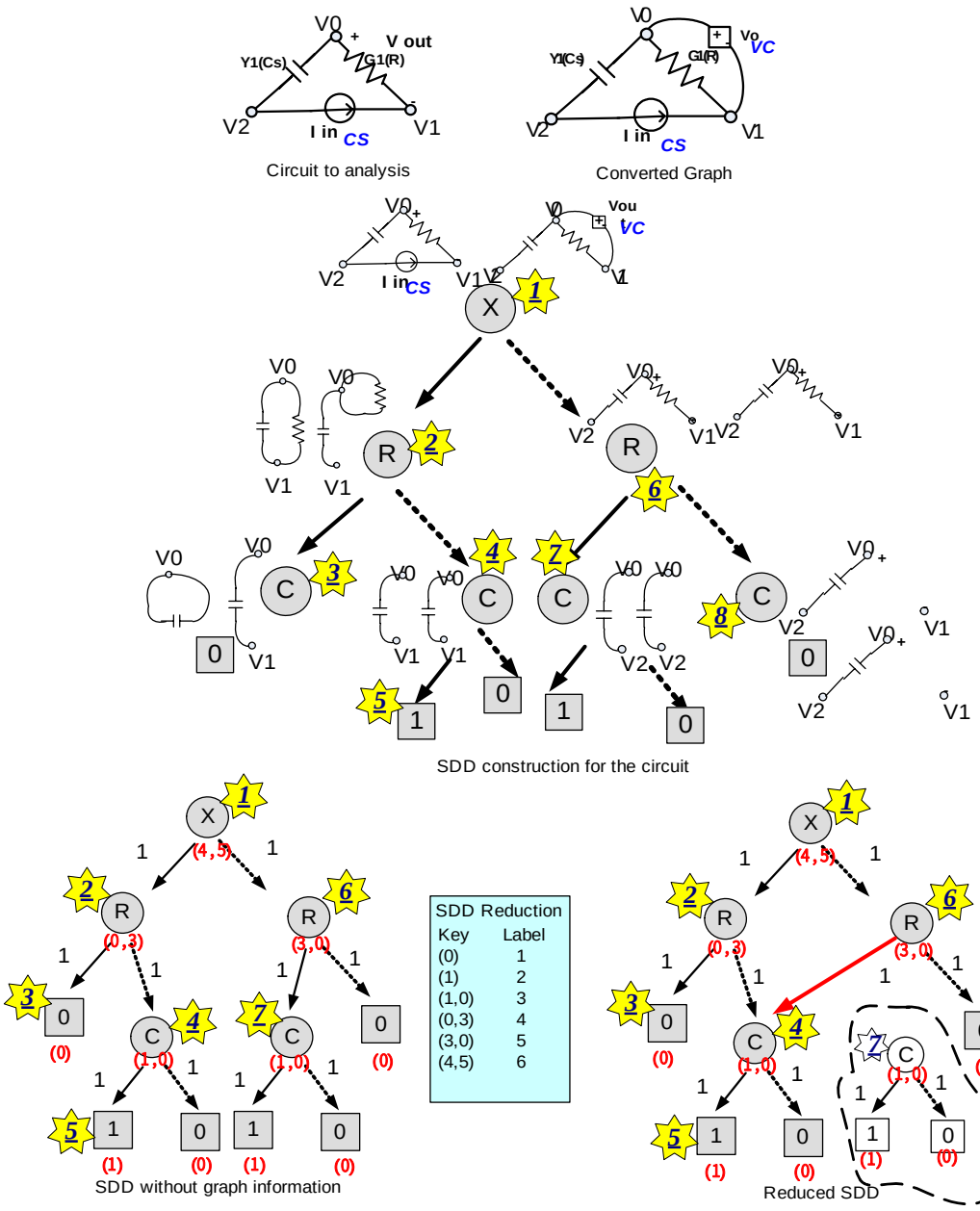


图 18 GRDD 约化举例

Fig.18 Example for GRDD Reduction

图 18 显示了判定图约化的过程。图中节点 4 和节点 7 分别引导的子判定图结构相同。因此节点 7 可以与节点 4 共享，而节点 7 以下的所有子节点均可以被删除。

约化操作执行后，我们执行垃圾收集的操作，将删除节点（仍存在于在哈希表中）的空间释放。

值得一提的是约化操作和垃圾收集操作都是二份判定图的标准操作，具体内容可见[36]。

3.4 启发式符号排序 （Heuristic Ordering）

我们在第二章中曾提到，在符号化分析开始前，我们需要先为符号设置一个处理顺序。事实上，符号处理的顺序在很大程度上将影响图约化判定图的尺寸，这也是所有判定图相关算法均会面临的关键问题。

在本文中我们对元件符号采用如下的排序方法：

1．输入输出受控源的符号总排在第一。

2．按照电路拓扑结构的连接关系进行排序。即从与输入相连的元件开始，然后是与这些元件相连的元件，然后再是与已排序元件相连的元件，… …，逐次以广度优先的方法进行排序。同时对于元件我确定一个优先级用于当某些元件同时与已排序元件相连时的情况，我们要求阻抗类元件优先于控制源类元件，对于同是阻抗类（控制源类）元件则采用随机的排序策略。

之所以使用这样的排序方式是为了在选择生成树时提前获得独立点（移除边而产生），从而结束操作（出现独立点则不会产生任何生成树）。

值得提出的是这个排序方法不是最优的方法，我们相信还有更好的排序方式获得更小尺寸的判定图。不同符号排序所得的实验结果详见 4.2。

3.5 图约化判定图求值 （GRDD Evaluation）

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。获得数值频率响应的效率由两部分组成：图约化判定图的构建效率和数值频响求值效率。3.4 中我们讨论了图约化判定图的构建效率主要受电路元件符号的操作顺序影响。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，其遍历的效率与判定图节点的个数呈线性关系。

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：

```
eval(vertex) = eval(vertex->left) * V(symbol) *sign1(vertex)
                    + eval(vertex->rigth) * sign0(vertex)。
```

其中 vertex 代表 GRDD 节点，eval(vertex)代表 vertex 节点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，sign1 和 sign0 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：

V(G) ==> G

V(Z) ==> Z^{-1} (R→R^{-1}, L→$(Ls)^{-1}$ → -j(2*∏*freq*L)$^{-1}$

V(Y) ==> Y (C→Cs→j(2*∏*freq*C))

V(Source) ==> Gain

除了数值频响之外，图约化判定图所对应的生成项个数也可以同过遍历判定图得到，而无需将所有生成项列出来计算。生成项个数递归计算公式如下：

T(vertex) = EvaluateTermNum(vertex->left) + EvaluateTermNum(vertex->right)。

其中 vertex 代表 GRDD 节点，T(vertex)代表 vertex 节点引导的判定子图所含的生成项个数， 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，且 T(0) = 0（即终节点 0 代表的生成项个数为 0），T(1) = 1（即终节点 1 代表的生成项个数为 1）。

3.6 其他效率提升策略 （Strategies for Efficiency）

3.6.1 并联元件预处理（Lumping Parallel Branches）

电路中的并联元件在有向图中表现为一组平行边(端点一致)。对于这些平行边，若其中一条会出现在某棵生成树中，则一定存在其他的生成树包含其余的平行边。在这里我们只考虑非受控源类器件，这样如果平行边中的一条边出现在某个有效生成树对中，那么其余的也会出现在某个有效生成树对中，而这些生成树对除了平行边之外，其余均相同。也就是说，所有的非受控源类的平行边在有效生成树对中是等价的。

基于这个事实，我们在构建 GRDD 之前对并联元件进行预处理。我们将并联元件等价成一个元件，该元件的符号（数值）等于相应元件的并联值（见图 19)。

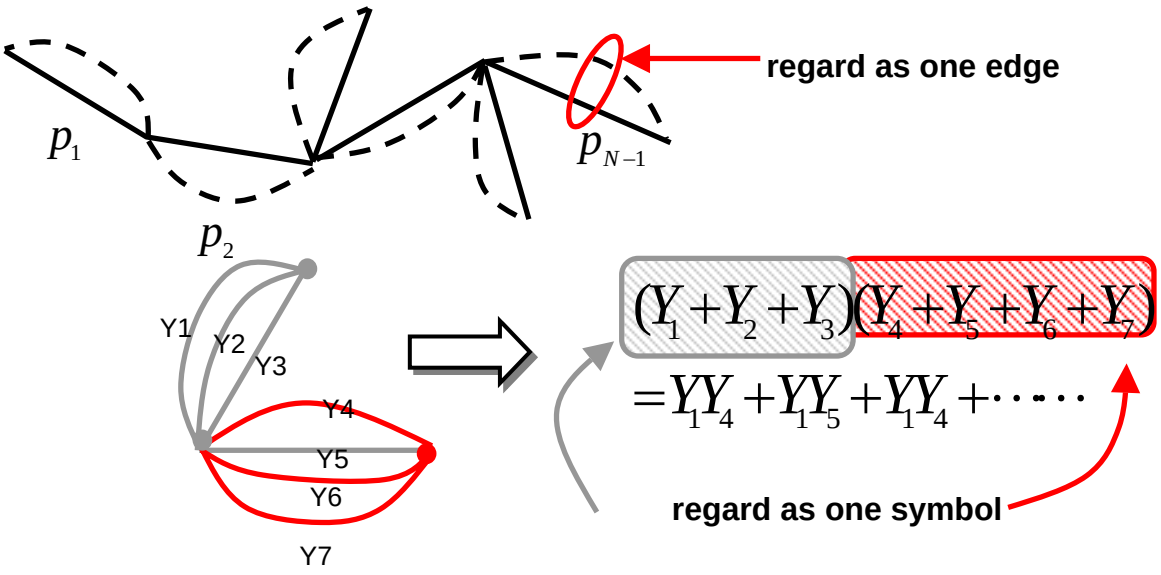


图 19 并联边预处理举例

Fig.19 Example for Lumping Parallel Branches

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，具体实验结果见 4.3。

3.6.2 分离图的早期检测（Early Disconnectivity Detection）

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。

图分离的状态如下：

1. 出现孤立点（见图 20 中的点 1）。

2. 有效边的个数比图中节点数少 2 或者更多，其中有效边计算时将平行边的边数计为 1，自环边的边数计为 0。如图 21 中，虽然原图含有 9 条边和 5 个节点，但事实上图中的有效边数为 3，而对于 5 个节点的图，3 条边是无法组成生成树（生成树边数= 节点数-1），即图中包含分离的部分。

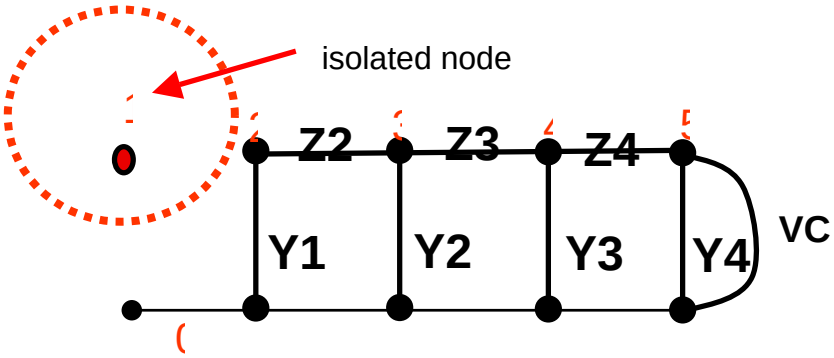
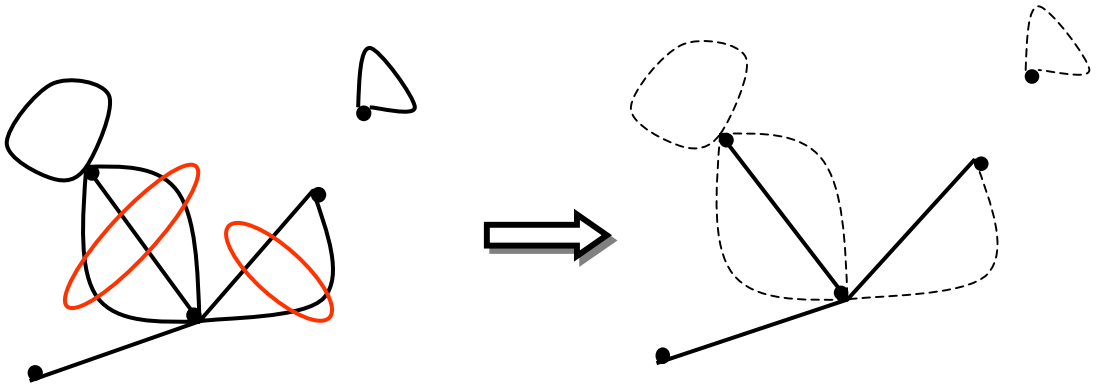


图 20 孤立点举例

Fig.20 Example for Isolation Node



(a) Original Graph

E = 9, V = 5

EA < V – 1

The graph is disconnected

(b) The graph with valid edge

EA = 3, V = 5

图 21 有效边计算举例

Fig.3-11 Example for Available Edge Counting

4　仿真器的应用与实验结果分析

4.1　基准电路试验结果

我们选取不同规模的基准电路对仿真器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 22）；

电路二：带通滤波器 1, 2 （图 23）；

电路三：μa741运算放大器（图 24）；

电路四：μa725 运算放大器（图 25）；

电路五：*MOSopamp* 运算放大器（图 27）；

其中电路一和电路二只包含无源器件电阻(R)、电容(C)和理想运算放大器；电路三和电路四包含有源二极管器件，电路三和电路四的规模不尽相同；电路五包含有源 MOS 管器件。

我们的测试平台为 Intel Pentium 1G 内存，主频为 1.73GHz 的处理器。

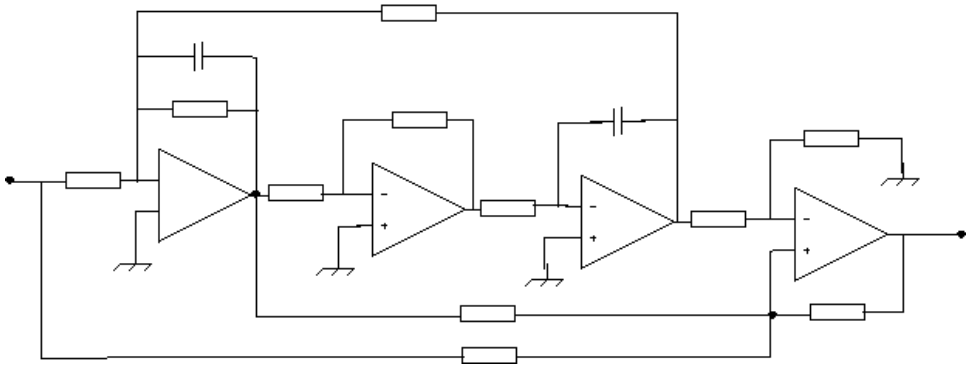


图 22 RC 滤波器

Fig. 22 Active RC Filter

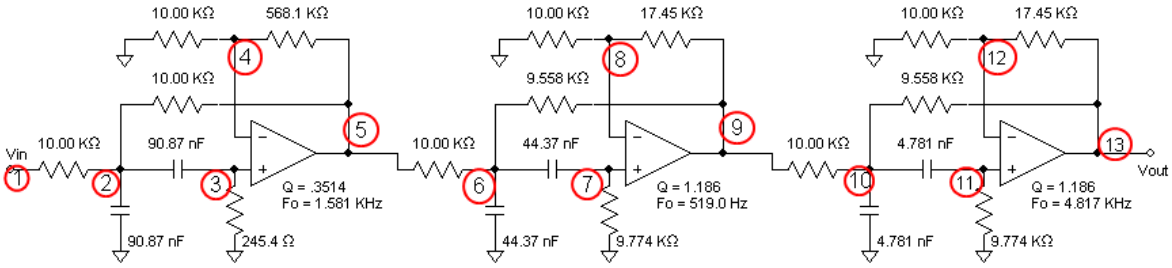


图 23(1) 带通滤波器

Fig. 23(1) BandPass Filter

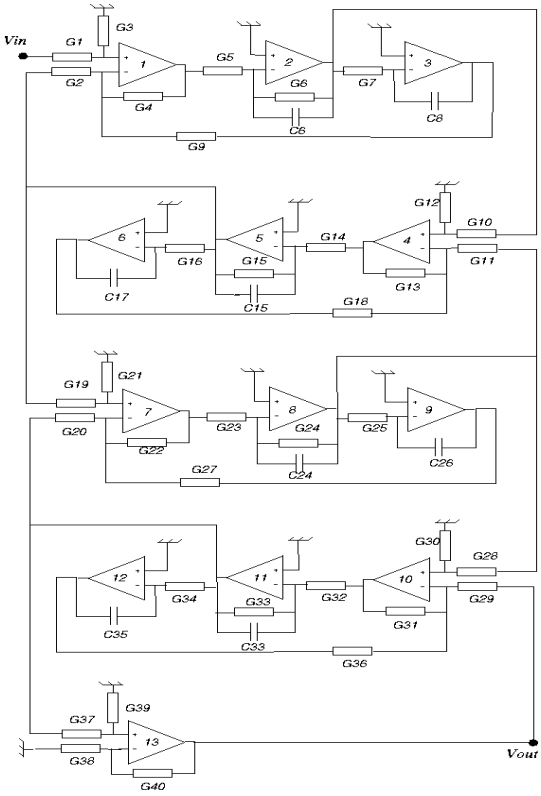


图 23 (2) 带通滤波器

Fig. 23(2) BandPass Filter

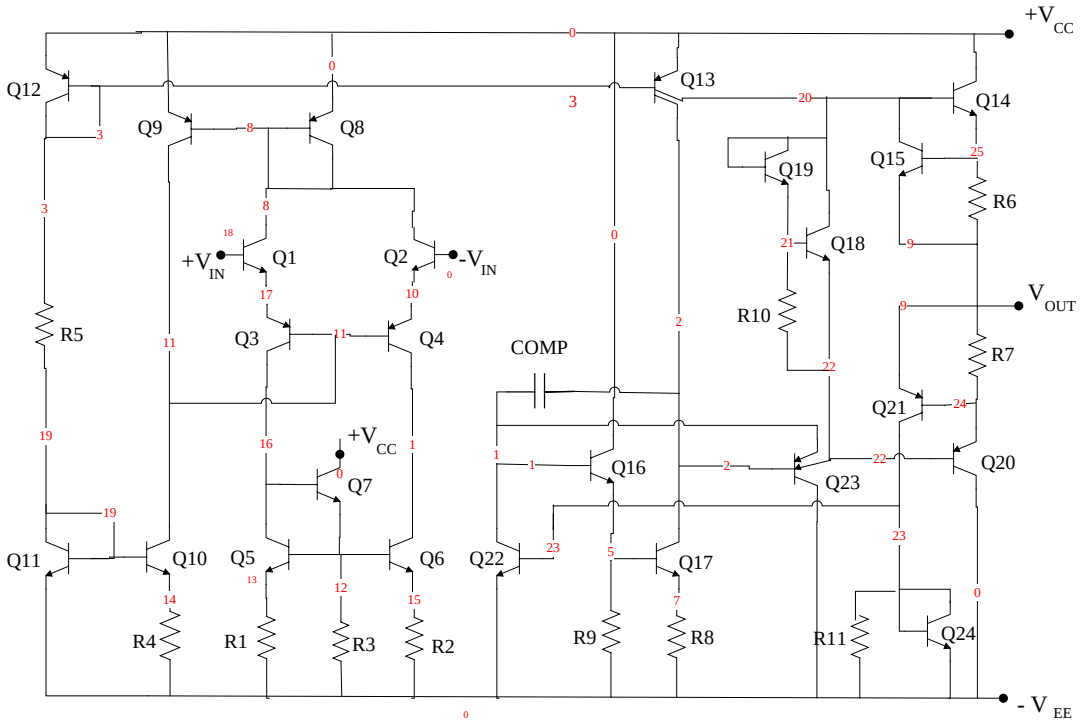


图 24 $\mu a741$运算放大器

Fig. 24 $\mu a741$ Amplifier

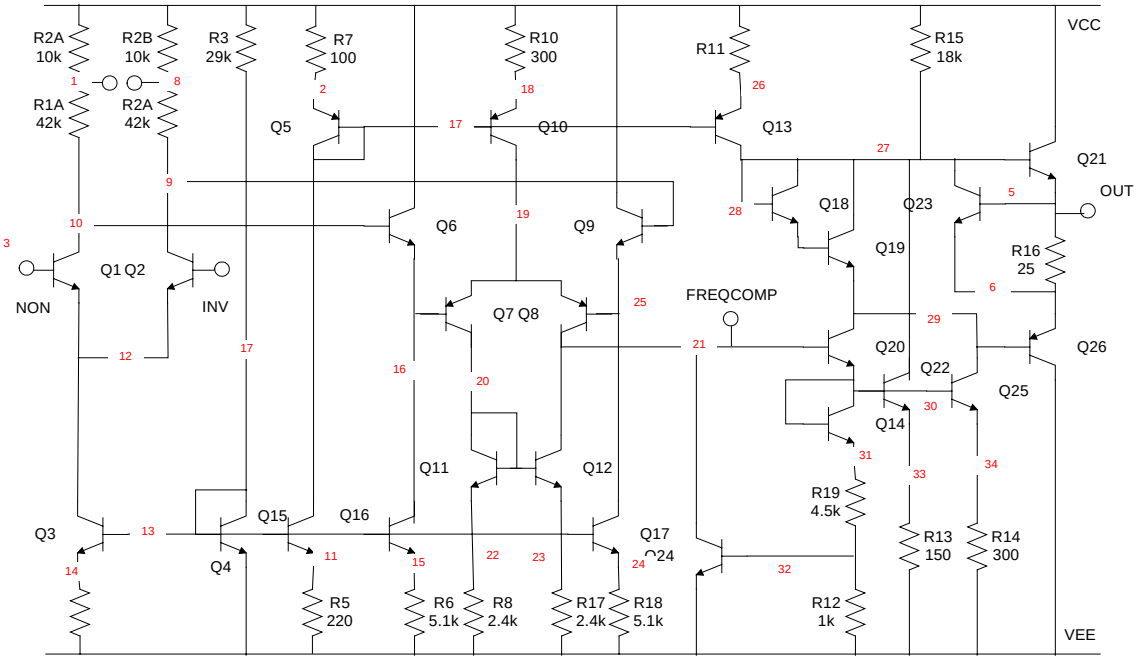


图 25 $\mu a725$运算放大器

Fig. 25 $\mu a725$ Amplifier

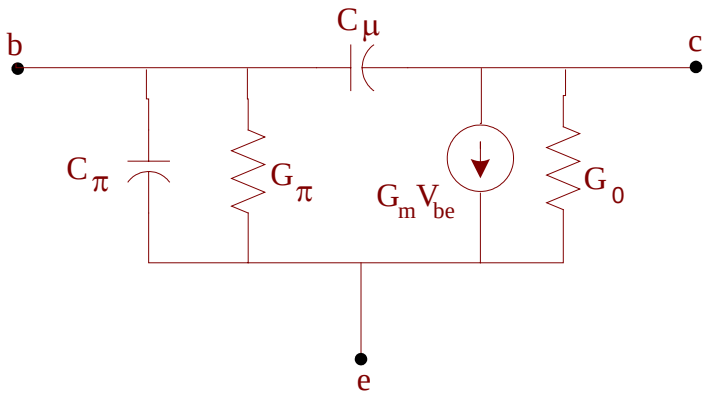


图 26 $\mu a741$，$\mu a725$运算放大器三极管小信号模型。

Fig.26 Small Signal Model for $\mu a741$ and $\mu a725$

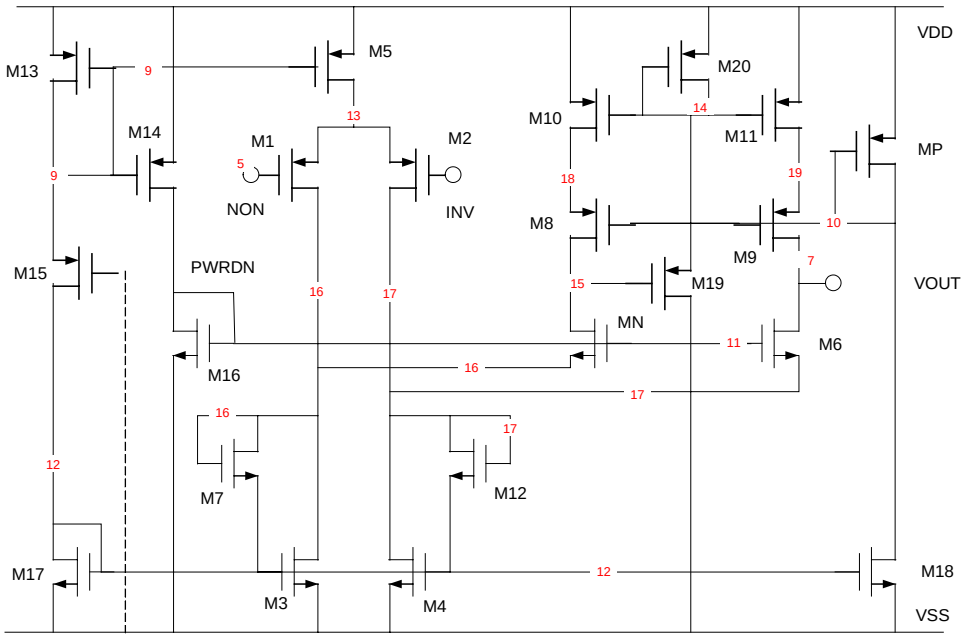


图 27 *MOSopamp* 运算放大器

Fig. 27 *MOSopamp* Amplifier

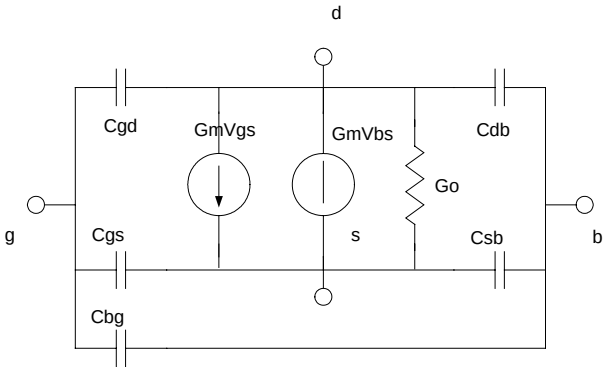


图 28 *MOSopamp* 运算放大器三极管小信号模型。

Fig. 28 Small Signal Model for *MOSopamp*

对于上述的测试电路集合，我们计算其传输函数的符号化表达式并作数值分析，仿真器的性能如表格 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；#Term 代表传输函数符号表达式所含生成项的数目；|GRDD|为图约化判定图所含节点的个数；Time 是仿真时间；Memory 为仿真所需的内存空间。|GRDD|的大小、仿真时间和内存与电路元件处理的顺序相关，但对于同一电路而言，#Term 总是相同的。

电路名称	#Edge	#Edge_lump	#Node	#Symb	#Term	|GRDD|	Time	Memory
RC Filter	23	22	11	17	13	420	0.1s	10.756MB
Bandpass Filter 1	29	29	14	25	9296	117	0.1s	10.664MB
Bandpass Filter 2	72	68	33	54	190,231,338	497	0.1s	11.676MB
$\mu a741$	160	103	24	81	1.39e+14	31887	1.9s	34.78MB
$\mu a725$	166	120	31	81	5.09e+17	53420	22.6s	358.7MB
MOSopamp	182	90	14	65	2.93e+09	154452	4.3s	84.52MB

表格 2 仿真器性能

Table.2 Performance of the Simulator

从仿真结果可以看出，我们提出的算法可以在一个比较理想的时间和内存下对电路进行仿真，尤其是测试电路三、四、五，其规模（20-30 个传输晶体管）要比目前所见的精确拓扑分析方法[3]、[4]、[6]、[7]研究结果所适用的对象规模大得多，可以说我们的仿真器是目前第一个可以通过拓扑分析方对模拟电路进行符号化分析

的工具。

图 29，图 30，图 31，图 32，图 33 显示了仿真器的频率响应（振幅 Mag 和相位 Phase）数值分析结果与 HSpice 的数值分析结果的比较，其中我们仿真器的数值结果是对电路元件符号赋予数值值并在不同频率点下进行求值，求值方法见 3.5。从这些频响图中我们可以看到，我们的仿真器分析的结果与 HSpice 仿真结果完全一致，而 HSpice 是工业界公认的标准的数值仿真器，所以可以表明我们的仿真器分析结果的正确性。

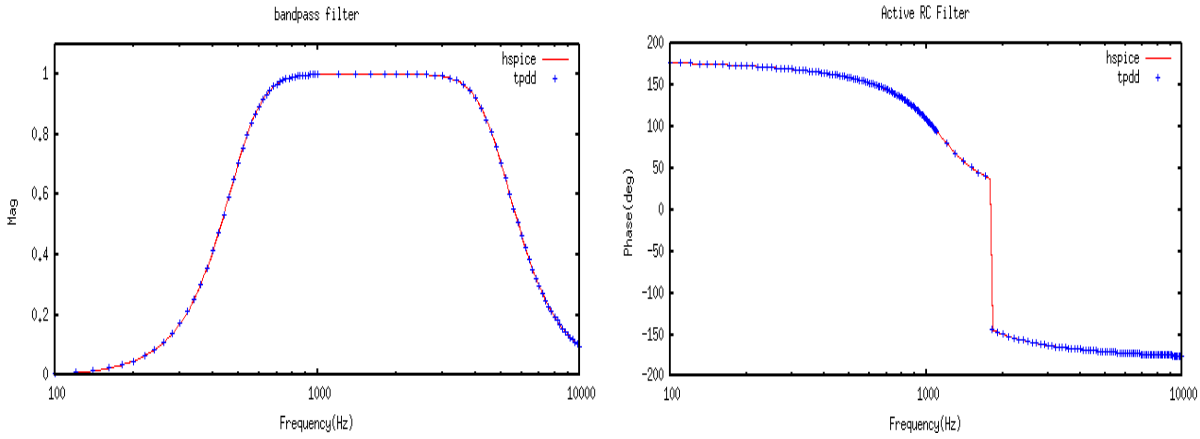


图 29 RC 滤波器频响

Fig.29 Frequency Response for Active RC Filter

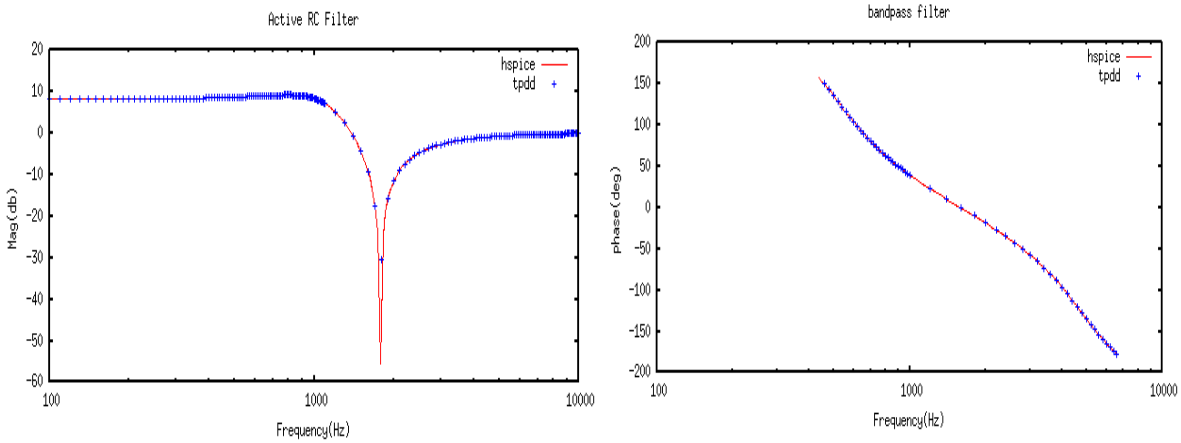


图 30(1) 带通滤波器频响

Fig. 30 (1) Frequency Response for BandPass Filter

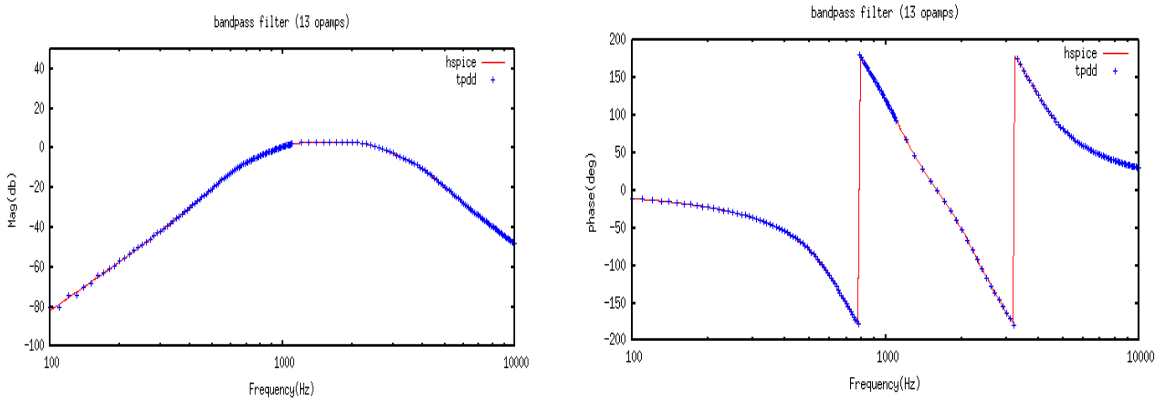


图 30 (2) 带通滤波器频响

Fig. 30 (2) Frequency Response for BandPass Filter

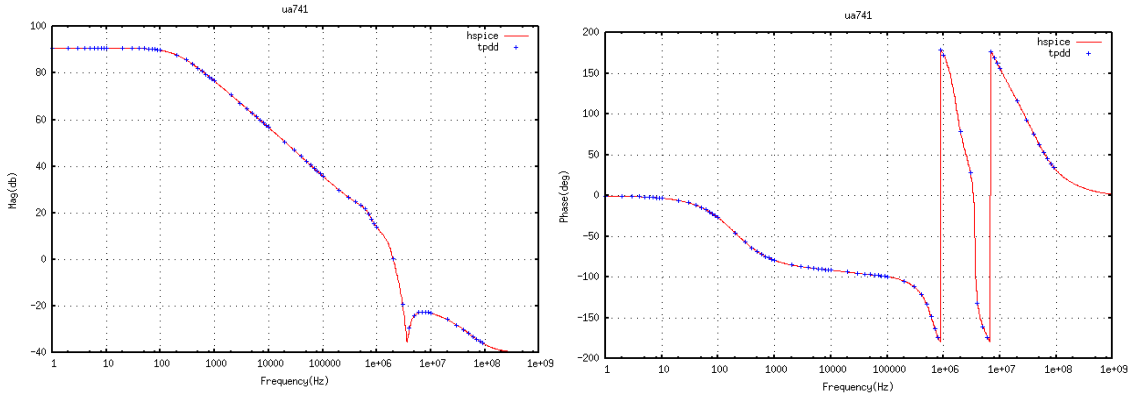


图 31 $\mu a741$ 运算放大器频响

Fig. 31 Frequency Response for $\mu a741$ Amplifier

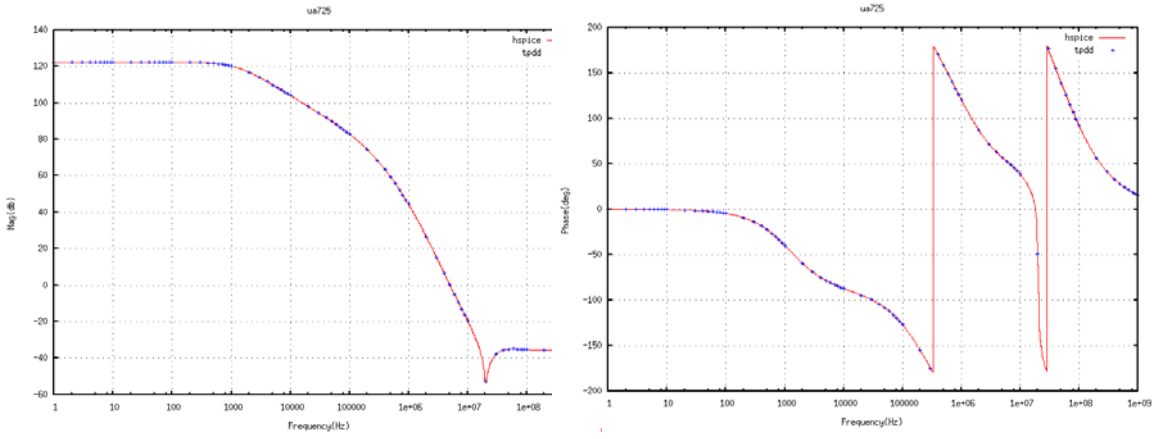


图 32 $\mu a725$ 运算放大器频响

Fig. 32 Frequency Response for $\mu a725$ Amplifier

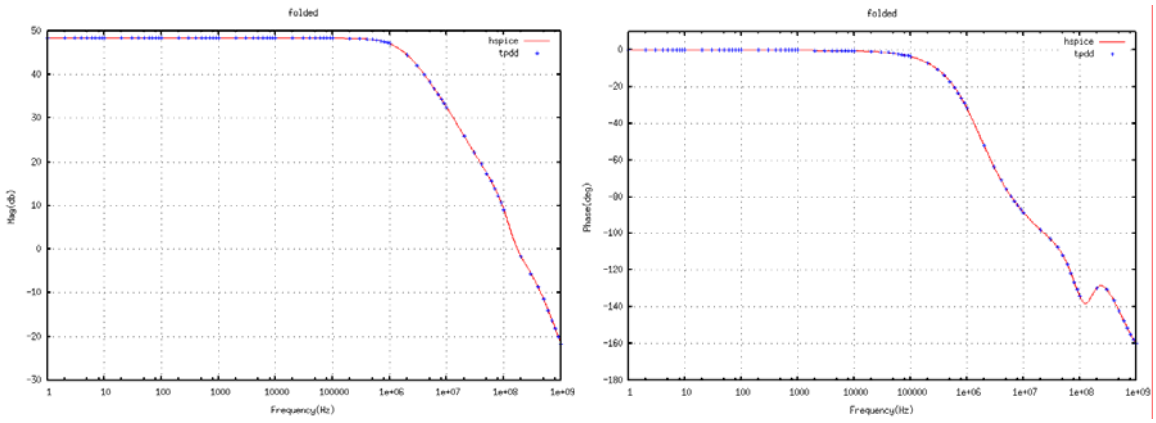


图 33 *MOSopamp* 运算放大器频响

Fig. 33 Frequency Response for *MOSopamp* Amplifier

4.2 不同符号顺序的试验结果

在 3.4 中我们提到，在仿真器内部所建立的图约化判定图的大小（仿真内存）与电路元件处理的顺序有很大的关系，本节中我们将通过举例来显示这种影响。

见图 34 的电路，一个n阶的RC阶梯网络，我们对不同阶数的网络进行分析，采取两种分析顺序：1. 先分析所有的R元件、再分析所有的C元件；2. 依照R、C间隔的顺序进行分析，这个顺序符合 3.4 种所描述的排序方式，具体顺序为R_1, C_1, R_2, C_2, …, R_n, C_n。从分析结果可以看出，两种分析顺序所需的时间和空间的复杂度有着巨大的差别，当我们选择第二种顺序的时候，无论是仿真时间还是内存，都与电路的尺寸呈线性关系，而第一种分析顺序所需的计算时空复杂据却与电路的尺寸呈指数关系。

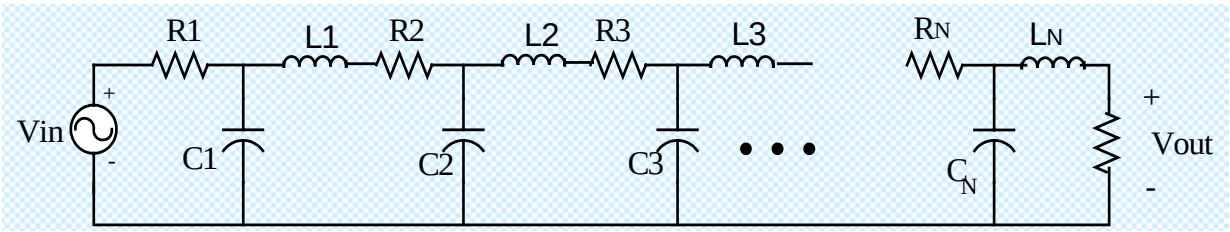


图 34 n 阶 RC 电路

Fig. 34 RC Ladder Circuit

	L25（25 阶）	L50（50 阶）	L75（75 阶）	L100（100 阶）
Time（顺序 1）	0.4s	4.2s	17.7s	N/A（超出运算能力）
Time（顺序 2）	0.2s	0.4s	0.5s	0.7s
#Terms	2.71e+14	5.41e+28	1.08e+43	2.14e+57
|GRDD|	228	453	678	903

表格 3 RC 阶梯电路不同符号处理顺序仿真结果

Table.3 Simulation Results for RC Ladder Circuits under Different Symbol Order.

4.3 其他实验结果

表格 4 显示了 3.2、3.3 中所提到约化子图和 GRDD 节点共享以及图约化判定图约化这两步操作所提供的统计数据，从表中可以看出在 GRDD 的构建过程中，子图共享和节点共享的次数非常频繁，而判定图约化可以大大减少 GRDD 节点的个数。

电路名称	约化子图 共享次数	GRD 节点 共享次数	约化前 GRDD 节点数	约化后 GRDD 节点数
MOSopamp	81652	147337	334423	154452
$\mu a741$	209005	35870	85815	31887
$\mu a725$	4928831	858184	1753944	53420

表格 4 GRDD 共享和约化的统计数据

Table.4 Statistics for GRDD Sharing and Reduction

表格 5 显示了 3.6 中所提的效率提高策略：并联元件预处理和分离图早期检测所带来的效率提升，我们分析的对象为一个双极型的晶体管电路，具体统计数据如下：

	Time	#Symb	|GRDD|	#Terms
不进行并行符号与处理	0.2s	51	2877	131,812
进行并行符号与处理	0.1s	22	696	1077
分离图早期预测	0.1s	22	696	1077

表格 5 效率提高策略运用统计数据

Table.5 Statistics of Strategies for Efficiency

5　符号化近似分析

如前文所述，精确的符号化的模拟电路分析方法的分析效率受到分析方法本身的限制，目前还只适合于不是很大规模的模拟电路。主要的原因是组成符号化分析结果的生成项个数与电路的规模呈指数关系，一旦电路规模增大，分析的时间和空间复杂度都会剧增。

模拟电路的特性由许多的方面决定，除了电路的结构之外，实现工艺和环境参数都会对电路的性能造成影响，而符号化分析正可以对这些问题提供自动化优化的可能性。因此对于大规模模拟电路的符号化分析还是有很大需求的。

通过设计经验我们可以知道，模拟电路的性能事实上很大程度取决于电路的一些主要的部分，而这些部分相对于整个电路而言，往往是比较小规模的[43]，而符号化分析方法对于小规模的电路又有着比较好的分析能力，因此精确的模拟电路分析并不完全需要，往往一些近似的分析就可以很好地反映电路的特性。

由于近似分析所需要的时空复杂度会远远小于精确的电路分析，我们期望对近似模拟电路的分析方法进行探索，使之可以适应规模更大的电路。

假设电路通过符号化分析得到如下形式的传输函数：

$$H(s)=\frac{N(s,p_1,...,p_m)}{D(s,p_1,...,p_m)}=\frac{a_0+a_1s+...+a_ps^p}{b_0+b_1s+...+b_qs^q}$$

分子与分母多项式的阶数分别为 p 和 q。如果我们分析电路的特性可知，往往不需要分析较高的阶数就可以得到，所以一般的近似分析结果的表达式具有如下的形式：

$$H(s)=\frac{\tilde{N}(s,p_1,...,p_m)}{\tilde{D}(s,p_1,...,p_m)}=\frac{a_0+a_1s+...+a_{\tilde{p}}s^{\tilde{p}}}{b_0+b_1s+...+b_{\tilde{q}}s^{\tilde{q}}}$$

$$\tilde{p}\le p,\tilde{q}\le q$$

其中分子与分母多项式的阶数分别比精确表达式的分子与分母要低，也可以看

作是一种电路降阶的处理。

5.1 符号化近似分析基本方法

符号化模拟电路的近似分析方法大致可以分为三个大类

5.1.1 计算后近似 （Approximate After Computation, AAC）

计算后近似的方法一般先计算出电路符号化的表达式，然后通过对该表达式的近似分析得到电路的特性，如零极点、关键项，关键组成部分等[43]。

假设电路通过符号化分析得到如下形式的传输函数：

$$H(s)=\frac{N(s,p_1,...,p_m)}{D(s,p_1,...,p_m)}=\frac{a_0+a_1s+...+a_ps^p}{b_0+b_1s+...+b_qs^q} \tag{5.1}$$

我们可以通过根分解（root splitting）的方法来得到那些与其他点分离的零、极点的符号化表达式。分解方法如下：

设多项式 $f(s)=a_0+a_1s+...+a_{n-1}s^{n-1}+a_ns^n$ 的根 S_k 与其他根分离，即有 $|s_1|\le...|s_{k-1}|\le|s_k|<<|s_{k+1}|<<...\le|s_n|$。则 S_k 的近似表达式为：$s_k=-a_{k-1}/a_k$。

5.1.2 计算时近似 （Approximate During Computation, ADC）

计算时近似的方法顾名思义就是在符号化分析的过程中进行近似分析。可以说这是真正意义上的符号化近似分析，具有较好的近似的意义，可以很好地降低符号化分析的复杂度。

计算时近似的分析方法主要的难点在于如何在分析的过程中确定近似的分析机制和错误控制的机制。

图 35 描述了计算时近似的一般流程。从图中可以知道错误控制与数值参考点的生成是指导计算时近似最关键的部分，错误控制机制的好坏决定了近似结果是否接近正确值；而数值参考点主要用来指导近似方法的趋势，同时在某种程度上也揭示了计算时近似的方法不完全是符号化的分析法，而是数值与符号混合的分析方法。另外一般计算时近似需要进行多次循环计算使得近似结果跟接近精确值。

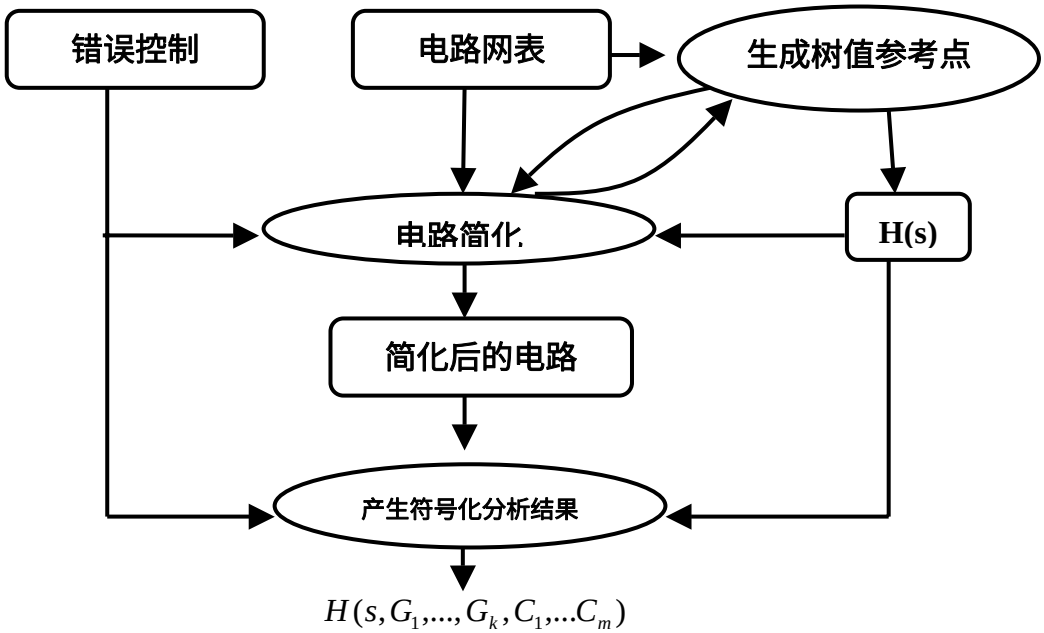


图 35 计算时近似分析流程

Fig.35 Process for Approximation During Computation

5.1.3 分解近似 （Decomposition Approach）

所谓分解近似就是将电路分解成规模较小的部分分别进行分析，然后将结果整合的近似分析方法。这种方法的主要关键是设计者必须对电路有比较好的认识，可以将电路分解成比较小的互不关联的部分。但是对于模拟电路而言，一般电路各部分的耦合性比较好，分解工作并不容易进行。

5.2 符号化近似分析方法应用

本文中我们主要进行计算后近似的分析，计算时分析和分解近似仍在探索过程中。

为了进行计算后近似的分析，我们需要对图约化判定图(GRDD)进行一些额外的操作，本节中我们将对这些操作进行具体的介绍。

5.2.1 符号化网络表达式形式

目前符号化网络表示式基本有两种形式[44]。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字

表示的重用，一般比较难揭示电路的特性。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进行展开，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，或者我们成为 S-展开的形式，具体形式如下：

$$T(s)=\frac{N(s)}{D(s)}=\frac{a_0+a_1s+...+a_ns^n}{b_0+b_1s+...+b_ds^d} \tag{5.2}$$

其中分子和分母多项式的系数由代表电路元件符号的乘积所做成的生成项的和组成，设 p_j 为电路元件符号，则：

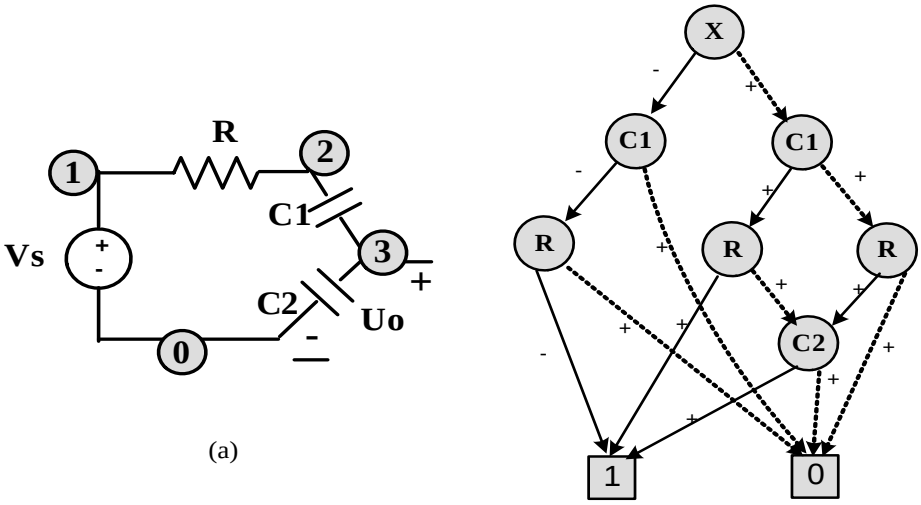
$$a_i \mid b_i=\sum_j(\prod p_j) \tag{5.3}$$

层次平展的表达式形式在分析是可以提供很多的便捷操作。例如，可以直接通过 $b_0,b_1,...,b_d$ 和 $a_0,a_1,...,a_n$ 得到对于主零、极点的非常好的近似。我们随后还会提到，这种表达形式对于 GRDD 的求值效率也有很大的帮助。

5.2.2 图约化判定图的 S-展开 (S-Expanded)

我们的仿真器通过对所分析电路执行图约化算法构建图约化判定图(GRDD)，然后直接分析 GRDD 来得到仿真的数据值。然而，直接分析 GRDD 得到的是 SOE 形式的符号化网络函数。例如式(5.4)，为图 36 所示电路通过 GRDD 分析所得的传输方程：

$$T(s)=\frac{1}{X}=\frac{C_1sR^{-1}}{C_1s(R^{-1}+C_2s)+C_2sR^{-1}} \tag{5.4}$$



(a) (b)

图 36 举例电路及其 GRDD

Fig.36 A circuit and its GRDD

由于层次展开的网络函数表达形式对分析有许多的帮助，所以我们希望可以获得这样的表达式。事实上，我们无须改动电路分析的算法，而只需对构建成的 GRDD 进行展开的操作，就可以获得 FNE 形式的网络函数表达式。具体展开方式与[45]中所述的方式相同。

我们考虑式(5.4)，可以将其重新写成 FNE 的形式：

$$T(s)=\frac{\sum_{i=1}^{1} N[i]s^{i}}{\sum_{i=1}^{2} D[i]s^{i}} \tag{5.5}$$

其中，

$$D[1]=C_1R^{-1}+C_2R^{-1}$$
$$D[2]=C_1C_2$$
$$N[1]=C_1R^{-1}$$

N[i]和 D[i]分别表示符号化传输函数分子分母多项式中对应第 i 阶变量 s 的系数。

我们提出一个新的概念：S-展开的 GRDD。S-展开的 GRDD 是一个有多个根节点的 GRDD，每个根节点代表一个表示多项式系数的判定图并且共享它们的子图。例如图 37 就是一个表示传输函数 FNE 形式(5.5)的 S-展开 GRDD（其中未连接的虚线边均指向 0 终节点）。

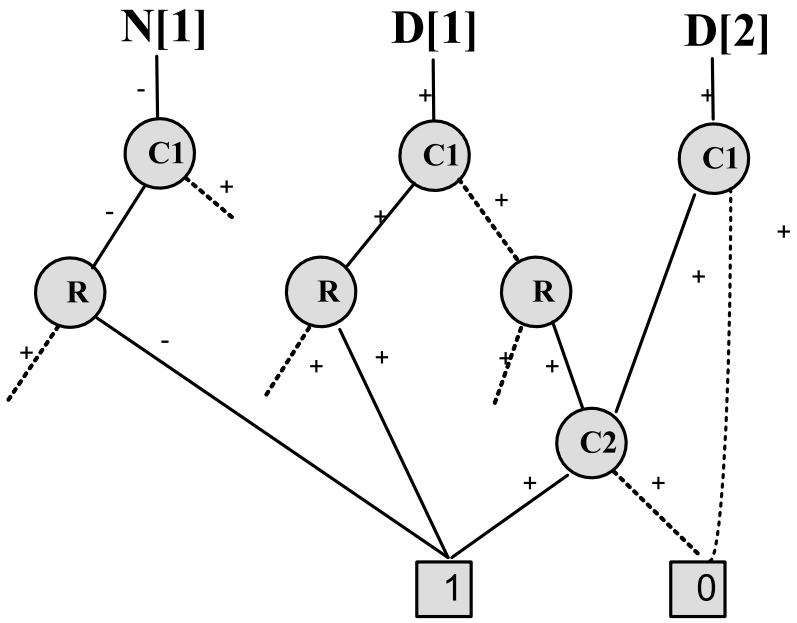


图 37 S-展开的 GRDD

Fig.37 S-Expanded GRDD

5.2.3 S-展开的图约化判定图构建

A．GRDD 分离操作

在对 GRDD 进行 S-展开前，需要先对 GRDD 中进行过并联预处理的符号做分离处理。如图 38(a)所示，假设节点 V 代表三个并联元件 R、C 和 L；V.Child1 和 V.Child0 分别为由 V 的 1-分枝和 0-分枝指向的 GRDD 节点， V.Sign1 和 V.Sign0 分别是关联在节点 V 的 1-分枝和 0-分枝的符号（见图 38(b)）。

分离节点 V 就是将 V 分别用代表节点 R、C 和 L 的节点代替。这三个新的节点通过各自的 0-分枝连接（见图 38(c)），最下面的一个节点的 0-分枝指向 V.Child0，所有节点的 1-分枝均指向 V.Child1。之所以这样连接的原因是 0-分枝连接的节点表示的是将这些节点代表的符号进行相加，而这正是并联元件在最终符号表达式中的表达形式。所有节点的 Sign1 均为 V.Sign1，最后一个节点的 Sign0 为 V.Sign0，而其余节点的 Sign0 均为正。

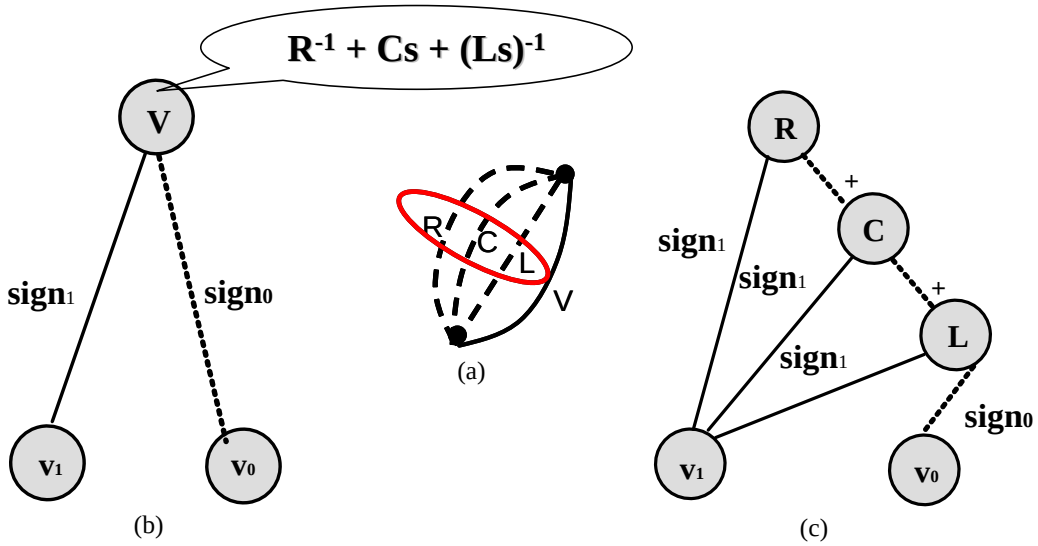


图 38 GRDD 分离操作

Fig.38 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。

B. GRDD 的 S-展开

构建了分离的 GRDD 之后，GRDD 中每个节点都只代表一个电路元件的符号。我们假设节点 V 所引导的判定图代表的多项式的展开形式为：

$$\sum_i p[i]s^i \qquad (5.6)$$

其中 $p[i]$ 可以是符号化的多项式也可以是 0。假设我们已经计算了 V.Child1 和 V.Child0 代表的多项式表达式，假设他们分别为 P1 和 P0，V.Symbol 为节点 V 所代表的电路元件符号，那么 V 的多项式可由下列的递归公式得到：

if V is 1- terminal, P[0] = 1;

else if V is 0-terminal, P[0] = 0;

else if V.Symbol is G or Z or Y,

P[i] = V * P1[i] * V.Sign1 + P0[i] * V.Sign0;

else if V.Symbol is C,

P[i] = V * P1[i + 1] * V.Sign1 + P0[i] * V.Sign0;

else if V.Symbol is L,

P[i] = V * P1[i - 1] * V.Sign1 + P0[i] * V.Sign0;

这样我们可以生成一个新的代表 P[i]的 S-展开的 GRDD 节点，它的符号为 V.Symbol，Sign1 为 V.Sign1，Sign0 为 V.Sign0，1-分枝和 0-分枝分别指向相应的 P1[j] 和 P0[k](j = i, i + 1, i – 1; k = i, i + 1, i – 1，具体值由 V.Symbol 的类型决定)。

GRDD 的 S-展开可以通过深度优先的方式执行。S-展开的 GRDD 节点也可以进行共享来节约存储空间，我们仍然使用哈希表来实现这些节点的共享。

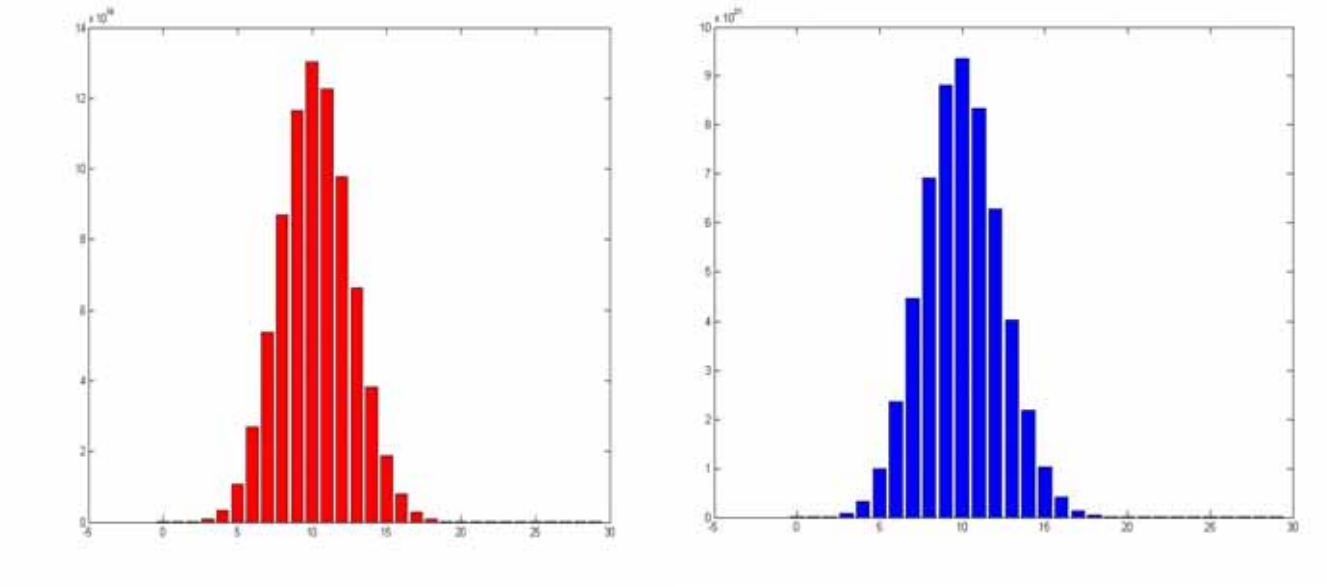
假设给定一个原始的具有|GRDD|个节点的 GRDD，要得到一个 S-展开的 GRDD 需要先进行 GRDD 分离操作，然后再进行 S-展开。S-展开的运算复杂度为 O((k+1)m|GRDD|)，S-展开的 GRDD 节点数不超过(k+1)m|GRDD|，其中 k 为 GRDD 所表示的多项式的最高阶数，m 为一组并联元件中元件个数的最大值。

5.3 近似分析结果

有了 S-展开的 GRDD，我们就可以做一些统计，求值和近似分析的工作。通过分析每一个系数 GRDD，我们可以很快地获得 FNE 形式的多项式系数信息。

A．不同阶 s 变量系数所含生成项统计结果：

我们遍历每个系数 GRDD，可以得到不同阶 s 变量系数所含生成项的个数，遍历的方式同一般的 GRDD 遍历。图 39(a)、(b)分别显示了 $\mu a725$ 电路符号化传输函数多项式不同阶 s 变量系数所含生成项个数的柱状图，生成项个数呈正态分布。



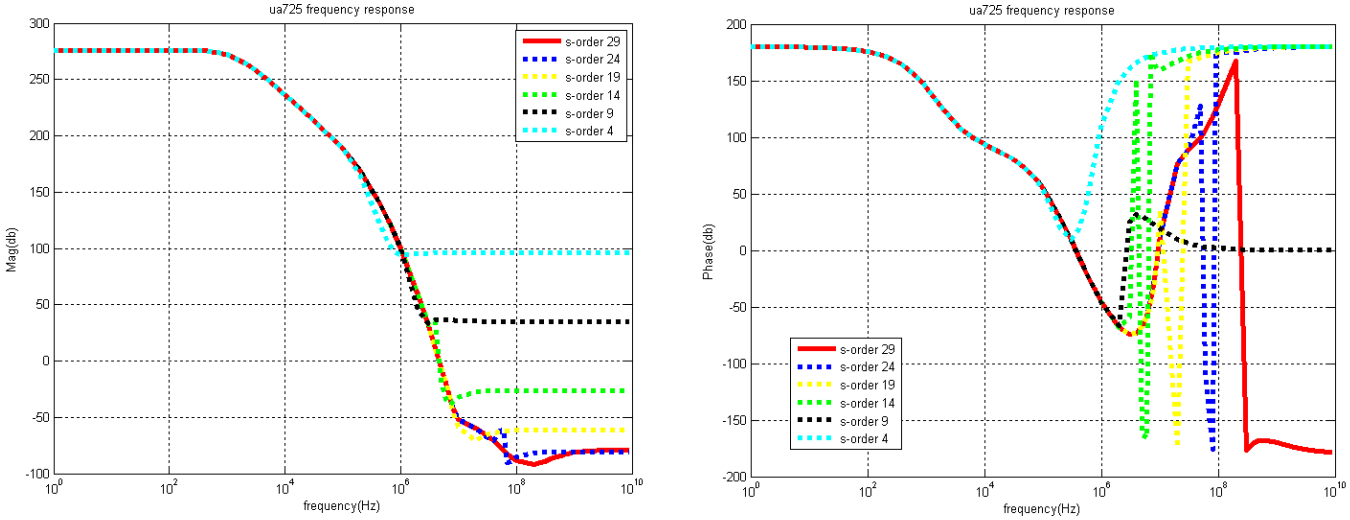
(a) $\mu a725$ 分子多项式系数生成项分布　(b) $\mu a725$ 分母多项式系数生成项分布

图 39 $\mu a725$ 传输函数多项式系数生成项个数分布

Fig.39 Distribution of the Numbers of the Product Terms for each Coefficient in the Transfer Function of $\mu a725$

B. 近似降阶分析

由于我们对不同阶 s-变量的系数都进行了 GRDD 的表达，所以我们可以对传输函数的数值值进行任意阶数的分析。例如，对于一个分子和分母多项式最高为 25 阶的传输函数，我们可以只分析 0-5 阶 s-变量的系数，从而得到传输函数的一个低阶的近似表达式。图 40(a)、(b)分别显示了对 $\mu a725$ 电路传输函数的振幅和相位的不同阶数的低阶近似。从图中可以看出，对于低频部分，虽然 $\mu a725$ 传输函数的分子与分母多项式的最高阶为 29，但只要分析其 0-4 阶就可以得到很到的近似；当然近似阶越高，近似结果就越准确。这种近似的优势在于第一、其近似的精确度较高；第二、如需要精确计算 $\mu a725$ 电路的传输函数数值值需要对 30 (29+1) 个系数 GRDD 进行求值，而做了近似可以大大减少求值的次数。



(a) (b)

图 40 $\mu a725$ 近似分析结果

Fig.40 Approximate Analysis of $\mu a725$

C．GRDD 快速求值。

S-展开的 GRDD 对于传输函数的精确求值效率的提高有很大的帮助。如 3.5 中所述的求值方式，由不同的频率点对于电容 C 和电感 L 类的元件其数值值是不同的，在进行 GRDD 求值时，每个频率点都必须重新执行遍历求值的操作，这种重复遍历的操作大大影响了求值的过程。但是对于 S-展开的 GRDD，每个系数 GRDD 的数值值与频率点无关，若先约求出了 s-变量的系数，对于不同的频率点，只需带入相应的 s 值于 FNE 表达式中即可，运算效率可以大大提高。表格 6 显示了 GRDD S-展开操作的性能。表格 7 显示了 S-展开 GRDD 的求值效率与 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的分析，频率点个数选取 1000 个。可以看到通过 S-展开 GRDD，我们仿真器的求值效率比 HSpice 的数值分析快近 3 倍，很好地显示出了本仿真器的分析优势。

ckt name	|GRDD|	Split |GRDD|	S-展开 GRDD 节点个数	S 变量的最高阶数
MOSopamp	55992	86928	638714	12
ua741	31887	42088	452610	22
ua725	53420	58796	1172667	29
ckt name	GRDD 构建时间	GRDD 分离时间	GRDD S-展开时间	Total Memory

MOSopamp	2.5s	0.1s	4.3s	123.31M
ua741	1.6s	< 0.1s	1.5s	92.75M
ua725	28.9s	0.1s	1.7s	458.06M

表格 6 GRDD S-展开的效率

Table.6 Performance for S-Expanded GRDD

ckt name	Hspice	基于原始 GRDD 的数值分析	基于 S-展开 GRDD 的数值分析
MOSopamp	0.14s	34.0s	0.04s
ua741	0.18s	19.9s	0.05s
ua725	0.18s	32.1s	0.05s

表格 7 GRDD 与 Hspice 数值分析效率对比

Table.7 Comparison of Numerical Analysis between GRDD and HSpice

6 结论与研究展望

6.1 结论

本文论述了一个基于拓扑分析法的符号化仿真器的分析原理、实现与应用。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。

本文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。

6.2 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，另外算法分析电路的复杂度与符号处理的顺序有密切的关系，而目前我们还没有提出这种最好顺序的方法。因此未来的研究可以有这样几点：1、探索符号处理顺序的通用方法，来降低算法的时、空复杂度，从而实现更大规模模拟电路的仿真。2、模拟电路的电路特征往往由一些关键部分所决定，一些合理的近似分析就可以很好地给出电路性能的提示，而精确仿真有的时候比较复杂，所以符号化的近似分析，甚至是数值和符号化混合的近似分析将是提高仿真器分析大规模电路能力的一种有效的方法，尤其是计算时近似，可以大大提高符号化分析的效率同时也能够给设计者提供完善的设计提示；3、本文所提出的算法只适用于线性时不变电路，但目前模拟电路设计中，尤其是高频设计立，为了更符合实际情况会使用一些在频率域仍为非线性的模型，所以对于非线性电路的符号化分析也是一个很好的研究方向。4、对于

符号化分析方法作用的探索也是我们想要继续研究的一个课题，形式化的分析方法本身可以给问题的自动化解决提供很好条件（例如可以提供符号化的主零、极点表达式等），我们需要进一步的探索，使我们的分析结果可以提供更多的设计提示、揭示电路的性能。

附录

定理证明（附录 1）

我们在这里给出第二章中所提定理的理论证明。该证明主要基于图的矩阵代数图表（matrix algebra tableau）和 Binet-Cauchy 定理，下面将给出简要的证明过程。该证明的概要已发表于[47]。

在证明定理之前我们先介绍一个代数操作来辅助证明。

1．0 的 ε 替代。

为了证明的方便，我们引入一个非常小的量 ε 来代替矩阵中的 0 元素。

例如：行列式
$$\begin{vmatrix} a & b \\ c & 0 \end{vmatrix} = -bc \tag{1}$$

我们用 ε 来替代 0 元素，则行列式变为
$$\begin{vmatrix} a & b \\ c & \varepsilon \end{vmatrix} = \varepsilon(a - b\varepsilon^{-1}c) = a\varepsilon - bc \tag{2}$$

其中当 $\varepsilon \to 0$ 时，有 $\begin{vmatrix} a & b \\ c & \varepsilon \end{vmatrix} = a\varepsilon - bc \to -bc$

2．分块矩阵行列式操作

对于分块矩阵我们采用同样的方式，假设采用一个较小的变量来替代 0 块，则要求分块矩阵的行列式值，可以得到如下公式：

$$\begin{vmatrix} A & B \\ C & D \end{vmatrix} = |A|\left|D - CA^{-1}B\right| \tag{3a}$$

$$\begin{vmatrix} A & B \\ C & D \end{vmatrix} = |D|\left|A - BD^{-1}C\right| \tag{3b}$$

由于使用变量来替代了 0 块，所以分块 A 和 D 都是非奇异的，故上述公式可以成立。

3．建立电路矩阵代数图表。

对于一个电路，我们采用印记（stamp）的方式来建立描述该电路的矩阵代数图表，矩阵代数图表将以数学表达式来描述一个电路。

首先对于一个电路，由 kirchhoff 定理可以得到如下的电路方程组：

$$ZI + YU = 0 \tag{4}$$

其中 Z，Y 为电路的阻抗和导纳矩阵，I 和 U 分别为表示电流分支和电压分支的向量

$$I^T = [I_1 \quad \cdots\cdots \quad I_E]$$

$$U^T = [U_1 \quad \cdots\cdots \quad U_E]$$

其中 E 为电路所转换成的有向图的边数，转换规则见 2.1.2。

电路允许含有 5 类元件：阻抗（Z），导纳（Y），四种类型的受控源（压控电压源 VCVS， 流控电流源 CCCS，压控电流源 VCCS，流控电压源 CCVS），独立源以及理想运算放大器。对于不同的元件，其支路的电流电压遵循下列方程：

$$U_i = Z_i I_i \quad \text{（Z 元件）} \tag{5a}$$

$$I_i = Y_i U_i \quad \text{（Y 元件）} \tag{5b}$$

$$U_i = E_{i,j} U_j, I_j = 0 \quad \text{（VCVS 元件）} \tag{5c}$$

$$I_i = F_{i,j} I_j, U_j = 0 \quad \text{（CCCS 元件）} \tag{5d}$$

$$I_i = G_{i,j} U_j, I_j = 0 \quad \text{（VCCS 元件）} \tag{5e}$$

$$U_i = H_{i,j} I_j, U_j = 0 \quad \text{（CCVS 元件）} \tag{5f}$$

$$U_j = 0, I_j = 0 \quad \text{（NU 元件）} \tag{5g}$$

其中受控源的增益采用 SPICE 语法描述的方式。(NO，NU)可以看作 VCVS 的增益为无穷时的一个特列，由于对于 NO 元件其电流和电压为任意值，所以无须列出相应的方程。另外为了证明的方便，我们将所有的 Z 元件都看作 Y 元件，这样(5a)可以写作 $I_i = Z_i^{-1} U_i$，为了运算的方便我们在操作 R 和 L 类的 Z 元件时仍然使用原来

的形式，只是在做求值运算的时候以倒数 Z_i^{-1} 来代替即可。

有了电流和电压方程，我们可以得到电路 Z 和 Y 矩阵图表的印记方法：

Z 矩阵印记 ↔ Y 矩阵印记

$$\begin{bmatrix} \ddots & & \\ & 1 & \\ & & \ddots \end{bmatrix} \leftrightarrow \begin{bmatrix} \ddots & & \\ & -Y_i & \\ & & \ddots \end{bmatrix}$$ （Y 元件） (6a)

$$\begin{bmatrix} \ddots & & \\ & 1 & \\ & & \ddots \end{bmatrix} \leftrightarrow \begin{bmatrix} \ddots & & \\ & -Z_i^{-1} & \\ & & \ddots \end{bmatrix}$$ （Z 元件） (6b)

$$\begin{bmatrix} \varepsilon & & \\ & \ddots & 1 \\ & & \ddots \end{bmatrix} \leftrightarrow \begin{bmatrix} -1 & \cdots & E_{i,j} \\ & \ddots & \vdots \\ & & 0 \end{bmatrix}$$ （VCVS 元件） (6c)

$$\begin{bmatrix} 1 & \cdots & -F_{i,j} \\ & \ddots & \vdots \\ & & \varepsilon \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 & & \\ & \ddots & \\ & & -1 \end{bmatrix}$$ （CCCS 元件） (6d)

$$\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 & \cdots & -G_{i,j} \\ & \ddots & \vdots \\ & & 0 \end{bmatrix}$$ （VCCS 元件） (6e)

$$\begin{bmatrix} \varepsilon & \cdots & H_{i,j} \\ & \ddots & \vdots \\ & & \varepsilon \end{bmatrix} \leftrightarrow \begin{bmatrix} -1 & & \\ & \ddots & \\ & & -1 \end{bmatrix}$$ （CCVS 元件） (6f)

$$\begin{bmatrix} \varepsilon & & \\ & \ddots & \\ & & 1 \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 & \cdots & 1 \\ & \ddots & \vdots \\ & & 0 \end{bmatrix}$$ （Nullor 元件） (6g)

在做印记的过程中我们假设所有的控制边的标号比受控边的标号大（i < j），这样所有的印记矩阵都会以上三角矩阵的形式出现。我们将所有 Y 矩阵的对角线上的元素都加上一个负号（-），将所有 Z 矩阵对角线中的 0 元素由 ε 代替，这样 Z 矩阵就为非奇异矩阵，因为（$\varepsilon > 0$），但到最后我们必须将 $\varepsilon \to 0$ 来计算最终值。对于 Nullor 元件，还可以有如下的印记矩阵图表：

$$\begin{bmatrix} \varepsilon & \cdots & 1 \\ & \ddots & \vdots \\ & & \varepsilon \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 & \cdots & 0 \\ & \ddots & \vdots \\ & & 1 \end{bmatrix}$$

（Nullor 元件）。

我们使用行变换来改变 CCCS 和 CCVS 对应的印记矩阵图表，将对角线上的 ε 变换到非对角线的位置，它们分别变为：

$$\begin{bmatrix} 1 & \cdots & 0 \\ & \ddots & \vdots \\ & & \varepsilon \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 & & -\varepsilon^{-1}F_{i,j} \\ & \ddots & \\ & & -1 \end{bmatrix}$$

（CCCS 元件） (6d')

$$\begin{bmatrix} \varepsilon & \cdots & 0 \\ & \ddots & \vdots \\ & & \varepsilon \end{bmatrix} \leftrightarrow \begin{bmatrix} -1 & & \varepsilon^{-1}H_{i,j} \\ & \ddots & \\ & & -1 \end{bmatrix}$$

（CCVS 元件） (6f')

从印记矩阵中我们大概可以看出生成树枚举的大致规则：

（a）Z-印记矩阵对角线上的 ε 只对应于 CC 和 VS 分枝，除了零器（Nullor）；

（b）Y-印记矩阵对角线上的 0 只对应于 VC 和 CS 分枝，除了零器（Nullor）；

（c）零器（Nullor）在 Y-印记矩阵上会同时出现 0 和 ε 相关联的 NO 边；

(d) 与符号项 $E_{i,j}$ 和 $\varepsilon^{-1}H_{i,j}$ 相关联的 VS 边的符号总是正的 ;而与 CS 相关联的 $\varepsilon^{-1}F_{i,j}$ 和 $G_{i,j}$ 的符号总是负的。

4．枚举定理的代数证明：

接下来我们给出第二章中枚举定理的代数证明。

我们将一个连通图任意移去一棵生成树后得到的子图称为共轭树（Cotree），假设 I_t

（U_t）和I_c（U_c）分别为表示生成树分枝和共轭树分枝的电流（电压）向量。在矩阵向量图表中，电路网络方程可以写成如下的分块矩阵方程形式：

$$\begin{bmatrix} A_t & & & A_c \\ & B_c & B_t & \\ Z_{(1)} & Y_{(2)} & Y_{(1)} & Z_{(2)} \\ Z_{(3)} & Y_{(4)} & Y_{(3)} & Z_{(4)} \end{bmatrix} \begin{bmatrix} I_t \\ U_c \\ U_t \\ I_c \end{bmatrix} = 0 \tag{7}$$

其中$A=[A_t \quad A_c]$为约化入射矩阵，A_t代表预选的生成树的入射矩阵；$B=[B_c \quad B_t]$为环路矩阵，B_c为共轭树的环路举证（见[1]），其中A_t和B_c的行列式 $\det|A_t|=\pm 1$，$|B_c|=\pm 1$。空矩阵块代表全 0 阵。

设$I^T=[I_t^T \quad I_c^T]$，$U^T=[U_t^T \quad U_c^T]$，分块矩阵(7)中的第三、四行块由(4)式表述的分支方程经过分块得到：

$$\begin{bmatrix} Z_{(1)} & Z_{(2)} \\ Z_{(3)} & Z_{(4)} \end{bmatrix} \begin{bmatrix} I_t \\ I_c \end{bmatrix} + \begin{bmatrix} Y_{(1)} & Y_{(2)} \\ Y_{(3)} & Y_{(4)} \end{bmatrix} \begin{bmatrix} U_t \\ U_c \end{bmatrix} = 0 \tag{8}$$

由公式组(6)可以看出，Z 矩阵经过特定的变换和 0 的ε替代后都可以变成对角线矩阵的形式，这样公式(9)可以变成如下形式：

$$\begin{bmatrix} \hat{Z}_{(1)} & \\ & \hat{Z}_{(4)} \end{bmatrix} \begin{bmatrix} I_t \\ I_c \end{bmatrix} + \begin{bmatrix} \tilde{Y}_{(1)} & \tilde{Y}_{(2)} \\ \tilde{Y}_{(3)} & \tilde{Y}_{(4)} \end{bmatrix} \begin{bmatrix} U_t \\ U_c \end{bmatrix} = 0 \tag{9}$$

其中$\hat{Z}_{(1)}$和$\hat{Z}_{(4)}$都为对角矩阵，并且对角线上的 0 元素都替换成了ε，$\tilde{Y}$矩阵也做了同样替换。假设 M 为公式(7)的系数方程，并且有

$$\det|M|=\begin{vmatrix} A_t & & & A_c \\ & B_c & B_t & \\ Z_{(1)} & Y_{(2)} & Y_{(1)} & Z_{(2)} \\ Z_{(3)} & Y_{(4)} & Y_{(3)} & Z_{(4)} \end{vmatrix} \tag{10}$$

经过公式(9)的替换可得：

$$\det|M|=\begin{vmatrix} A_t & & & A_c \\ & B_c & B_t & \\ Z_{(1)} & Y_{(2)} & Y_{(1)} & Z_{(2)} \\ Z_{(3)} & Y_{(4)} & Y_{(3)} & Z_{(4)} \end{vmatrix}=\begin{vmatrix} A_t & & & A_c \\ & B_c & B_t & \\ \hat{Z}_{(1)} & \tilde{Y}_{(2)} & \tilde{Y}_{(1)} & \\ & \tilde{Y}_{(4)} & \tilde{Y}_{(3)} & \hat{Z}_{(4)} \end{vmatrix}$$

$$=\det|A_t|\det|B_c|\det|\hat{Z}_{(1)}|\det|\hat{Z}_{(4)}|\times$$

$$=\begin{vmatrix} I & & & Q \\ & I & -Q^T & \\ I & \hat{Z}_{(1)}^{-1}\tilde{Y}_{(2)} & \hat{Z}_{(1)}^{-1}\tilde{Y}_{(1)} & \\ & \hat{Z}_{(4)}^{-1}\tilde{Y}_{(4)} & \hat{Z}_{(4)}^{-1}\tilde{Y}_{(3)} & I \end{vmatrix} \qquad (11)$$

其中 $Q=A_t^{-1}A_c$，$B_c^{-1}B_t=-Q^T$。

设 $\det|M_1|=\begin{vmatrix} I & & & Q \\ & I & -Q^T & \\ I & \hat{Z}_{(1)}^{-1}\tilde{Y}_{(2)} & \hat{Z}_{(1)}^{-1}\tilde{Y}_{(1)} & \\ & \hat{Z}_{(4)}^{-1}\tilde{Y}_{(4)} & \hat{Z}_{(4)}^{-1}\tilde{Y}_{(3)} & I \end{vmatrix}$

$$
\begin{aligned}
\det|M_1| &= \left| \begin{bmatrix} \hat{Z}_{(1)}^{-1}\tilde{Y}_{(1)} & \\ \hat{Z}_{(4)}^{-1}\tilde{Y}_{(3)} & I \end{bmatrix} - \begin{bmatrix} I & \hat{Z}_{(1)}^{-1}\tilde{Y}_{(2)} \\ & \hat{Z}_{(4)}^{-1}\tilde{Y}_{(4)} \end{bmatrix} \begin{bmatrix} & Q \\ -Q^T & \end{bmatrix} \right| \\
&= \left| \begin{matrix} \hat{Z}_{(1)}^{-1}(\tilde{Y}_{(1)} + \tilde{Y}_{(2)}Q^T) & -Q \\ \hat{Z}_{(4)}^{-1}(\tilde{Y}_{(3)} + \tilde{Y}_{(4)}Q^T) & I \end{matrix} \right| \\
&= \left| \hat{Z}_{(1)}^{-1}(\tilde{Y}_{(1)} + \tilde{Y}_{(2)}Q^T) + Q\hat{Z}_{(4)}^{-1}(\tilde{Y}_{(3)} + \tilde{Y}_{(4)}Q^T) \right| \\
&= \left| \hat{Z}_{(1)}^{-1}\tilde{Y}_{(1)} + \hat{Z}_{(1)}^{-1}\tilde{Y}_{(2)}Q^T + Q\hat{Z}_{(4)}^{-1}\tilde{Y}_{(3)} + Q\hat{Z}_{(4)}^{-1}\tilde{Y}_{(4)}Q^T) \right| \\
&= \left| \begin{bmatrix} I & Q \end{bmatrix} \begin{bmatrix} \hat{Z}_{(1)}^{-1} & \\ & \hat{Z}_{(4)}^{-1} \end{bmatrix} \begin{bmatrix} \tilde{Y}_{(1)} & \tilde{Y}_{(2)} \\ \tilde{Y}_{(3)} & \tilde{Y}_{(4)} \end{bmatrix} \begin{bmatrix} I \\ Q^T \end{bmatrix} \right| \\
&= \left| \begin{bmatrix} A_t & A_c \end{bmatrix} \begin{bmatrix} \hat{Z}_{(1)}^{-1} & \\ & \hat{Z}_{(4)}^{-1} \end{bmatrix} \begin{bmatrix} \tilde{Y}_{(1)} & \tilde{Y}_{(2)} \\ \tilde{Y}_{(3)} & \tilde{Y}_{(4)} \end{bmatrix} \begin{bmatrix} A_t^T \\ A_c^T \end{bmatrix} \right|
\end{aligned} \tag{12}
$$

由于 $|A_t||B_c| = \pm 1$，且 $\det|M| = 0$ 等价于 $\det|\hat{Z}_{(1)}|\det|\hat{Z}_{(4)}|\det|M_1| = 0$

设

$$
\hat{Z} = \hat{Z}(\varepsilon) = \begin{bmatrix} \hat{Z}_{(1)}(\varepsilon) & \\ & \hat{Z}_{(4)}(\varepsilon) \end{bmatrix}
$$

$$
S = |\hat{Z}(\varepsilon)| \left| A\hat{Z}^{-1}(\varepsilon)\tilde{Y}(\varepsilon)A^T \right| \tag{13}
$$

则 $\det|M| = 0$ 等价于 $S = 0$。

我们对 S 使用 Binet-Cauchy 定理进行行列式展开可以得到

$$
\begin{aligned}
S &= \left| \hat{Z} \right| \sum_{j_1,\ldots,j_n} \left[A\hat{Z}^{-1} \right] \begin{pmatrix} 1 & \ldots & n \\ j_1 & \ldots & j_n \end{pmatrix} \left[\tilde{Y} A^T \right] \begin{pmatrix} j_1 & \ldots & j_n \\ 1 & \ldots & n \end{pmatrix} \\
&= \left| \hat{Z} \right| \sum_{\substack{j_1,\ldots,j_n \\ k_1,\ldots,k_n}} \left(\sum_{i=1}^{n} \hat{Z}_{j_i}^{-1} \right) A \begin{pmatrix} 1 & \ldots & n \\ j_1 & \ldots & j_n \end{pmatrix} \tilde{Y} \begin{pmatrix} j_1 & \ldots & j_n \\ k_1 & \ldots & k_n \end{pmatrix} A^T \begin{pmatrix} k_1 & \ldots & k_n \\ 1 & \ldots & n \end{pmatrix}
\end{aligned} \tag{14}
$$

其中 n 为电路网络中的节点数（除去接地零点）。$A\begin{pmatrix}1 & \dots & n\\ j_1 & \cdots & j_n\end{pmatrix}$代表由矩阵$A$的行{1, …, n}和列{j1, …, jn}组成的约化入射矩阵。求和符下标中的{j1, …, jn}默认为 j1 < j2 < … < jn。

从(14)式我们可以观察到，求和公式中要产生一个非零项，当且仅当

$A\begin{pmatrix}1 & \dots & n\\ j_1 & \dots & j_n\end{pmatrix}$，$\tilde{\mathrm{Y}}\begin{pmatrix}j_1 & \dots & j_1\\ k_1 & \dots & k_n\end{pmatrix}$，$A^T\begin{pmatrix}k_1 & \dots & k_n\\ 1 & \dots & n\end{pmatrix}$均为非零。

对于一个电路网络而言，一个非零的约化入射行列式正好代表电路网络的一棵生成树。当{j1, …, jn}和{k1, …, kn}完全一致时，$A\begin{pmatrix}1 & \dots & n\\ j_1 & \cdots & j_n\end{pmatrix}$和$A^T\begin{pmatrix}k_1 & \dots & k_n\\ 1 & \dots & n\end{pmatrix}$代表一棵有效生成树；当{j1, …, jn}和{k1, …, kn}不完全一致时，$A\begin{pmatrix}1 & \dots & n\\ j_1 & \cdots & j_n\end{pmatrix}$和$A^T\begin{pmatrix}k_1 & \dots & k_n\\ 1 & \dots & n\end{pmatrix}$代表一对有效生成树对，其中{j1, …, jn}代表有效左生成树的树边在电路网络中的分支序号，{k1, …, kn}代表有效右生成树的树边在电路网络中的分支序号。为了枚举出有效的生成树（cancellation-free）,除了$A\begin{pmatrix}1 & \dots & n\\ j_1 & \cdots & j_n\end{pmatrix}$和$A^T\begin{pmatrix}k_1 & \dots & k_n\\ 1 & \dots & n\end{pmatrix}$的行列式不为零之外,还需要适当地选取$\hat{Z}^{-1}_{j_i}$和$\tilde{\mathrm{Y}}\begin{pmatrix}j_1 & \cdots & j_1\\ k_1 & \dots & k_n\end{pmatrix}$来得到有效的非零项，而选取方式的确定正好符号第二章所述的枚举规则。值得提出的是，$\tilde{\mathrm{Y}}\begin{pmatrix}j_1 & \cdots & j_1\\ k_1 & \dots & k_n\end{pmatrix}$是上对角矩阵，其行列式等于对角线元素的乘积，而这些元素中的 0 均被替换成了符号ε，$\hat{Z}^{-1}_{j_i}$中的 0 得倒数被替换成了ε^{-1}，因此一旦$\left|\hat{\mathrm{Z}}\right|$的对角线上出

现了若干个ε，则需要在$\tilde{\mathrm{Y}}\begin{pmatrix} j_1 & \cdots & j_1 \\ k_1 & \cdots & k_n \end{pmatrix}$和$\hat{Z}_{j_i}^{-1}$中选择同样个数的为$\varepsilon^{-1}$来抵消这些回趋向于 0 的$\varepsilon$。例如 CCVS 元件，Z 矩阵对角线上的有两个$\varepsilon$，当选取 Z 矩阵上的两个为$\varepsilon^{-1}$的$\hat{Z}_{j_i}^{-1}$时正好可以抵消，这时正好有{j1, …, jn}={k1, …, kn}，即 CC 和 VS 边以有效生成树边的形式出现；当只选取一个为ε^{-1}的$\hat{Z}_{j_i}^{-1}$时，就需要同时选取$\tilde{\mathrm{Y}}\begin{pmatrix} j_1 & \cdots & j_1 \\ k_1 & \cdots & k_n \end{pmatrix}$中的$\varepsilon^{-1}H_{i,j}$元素来抵消，这时{j1, …, jn}≠{k1, …, kn}，即 CC 和 VS 边分别出现在一对有效生成树对中。同理，对于其他的元件，我们可以逐步推出第二章中的枚举规则。

对于生成项的符号，同时也由$A\begin{pmatrix} 1 & \cdots & n \\ j_1 & \cdots & j_n \end{pmatrix}$，$\tilde{\mathrm{Y}}\begin{pmatrix} j_1 & \cdots & j_1 \\ k_1 & \cdots & k_n \end{pmatrix}$，$A^T\begin{pmatrix} k_1 & \cdots & k_n \\ 1 & \cdots & n \end{pmatrix}$的行列式符号和$\hat{Z}_{j_i}^{-1}$符号的乘积决定。对于生成树而言，$A\begin{pmatrix} 1 & \cdots & n \\ j_1 & \cdots & j_n \end{pmatrix}$和$A^T\begin{pmatrix} k_1 & \cdots & k_n \\ 1 & \cdots & n \end{pmatrix}$行列式值永远相同，又$\det A\begin{pmatrix} 1 & \cdots & n \\ j_1 & \cdots & j_n \end{pmatrix} = \pm 1$，故$A\begin{pmatrix} 1 & \cdots & n \\ j_1 & \cdots & j_n \end{pmatrix}$和$A^T\begin{pmatrix} k_1 & \cdots & k_n \\ 1 & \cdots & n \end{pmatrix}$的乘积总为 1，不影响最后的生成项符号；而$\tilde{\mathrm{Y}}\begin{pmatrix} j_1 & \cdots & j_1 \\ k_1 & \cdots & k_n \end{pmatrix}$对角线上的元素符号总为负，但是对于有效生成树对应的生成项，其符号无论是正还是负，总是一致的，而一个公因子式的符号对于 det|M|=0 而言并无影响，故可以忽略，这样对于生成树对应的生成项，其符号总为正；

对于有效生成树对对应的生成项，其符号中的$\tilde{\mathrm{Y}}\begin{pmatrix} j_1 & \cdots & j_1 \\ k_1 & \cdots & k_n \end{pmatrix}$贡献的部分总与生

成树对应的生成项一致，而 $\hat{Z}_{j_i}^{-1}$ 总为正，$\left|\hat{Z}\right|$ 又是共有的部分，所以最终的生成项符号直接由 $A\begin{pmatrix}1 & \cdots & n\\ j_1 & \cdots & j_n\end{pmatrix}$ 和 $A^T\begin{pmatrix}k_1 & \cdots & k_n\\ 1 & \cdots & n\end{pmatrix}$ 两个约化入射矩阵的行列式值来决定。

5．生成项符号算法证明：

我们使用高斯消去法(Gause Elimination)来计算行列式的值，即保留每一列的一个非零元素，同时将其余的非零元素通过线性变换消去。对于约化入射矩阵而言，每一列只有两个非零元素 1 或者-1，我们只需通过将包含着两个非零元素的行相加就可以消去其中的一个非零元素。每次进行的这样的线性变换在消去某一列的一个非零元素 v2 的同时也将其余边与保留下的元素 v1 行号一致的非零元素变换到了与 v2 所对应的行上，这一步执行过程其实就是图约化过程中的选择符号的操作（将两个节点合并改变相应结点号）。且这样的线性操作不会产生全零列，如果出现的话，行列式就存在线性相关的列，即对应的生成树中会出现环路，其行列式值为 0，这与生成树的定义以及约化入射矩阵行列式只为 ± 1[1]的性质矛盾。通过这样的线性变换，我们可以获得一个每一列和每一行均有且只有一个非零元素的方阵，若我们每次均用 1 去消去-1（反之也可），则这些非零元素只可能为 1。要计算这样的行列式值，我们递归地使用 Laplace 列（行）展开来计算。

由 Laplace 按第 i 行展开，我们有：

设 $A=(a_{ij})$ 为 n 阶方阵，A_{ij} 为 a_{ij} 的代数余子式，则：

$$\det A=\sum_{j=1}^{n}(-1)^{i+j}a_{ij}A_{ij} \qquad (15)$$

对于我们所讨论的进行了线性变换后的约化入射矩阵，我们有

$$A=(a_{ij}) \qquad a_{ij}=\begin{cases}1 & i=j\\ 0 & i\neq j\end{cases} \qquad (16)$$

于是由(15)可以得到如下的递归公式：

$$\begin{aligned}&\det A=(-1)^{i1+j1}1\det A^{c}\begin{pmatrix}i_1\\ j_1\end{pmatrix}\\ &\det A^{c}\begin{pmatrix}i_1\cdots i_k\\ j_1\cdots j_k\end{pmatrix}=(-1)^{i_{k+1}+j_{k+1}}1\det A^{c}_{k+!}\begin{pmatrix}i_1\cdots i_{k+!}\\ j_1\cdots j_{k+1}\end{pmatrix} \qquad k=1,2,\ldots,n-1\end{aligned} \qquad (15)$$

其中 $\det A^{c}\begin{pmatrix} i_1 \cdots i_k \\ j_1 \cdots j_k \end{pmatrix}$ 代表从矩阵 A 中删去第 $i_1,...,i_k$ 行和第 $j_1,...,j_k$ 列所余下元素按原序排列成的方阵的行列式，即 $A\begin{pmatrix} i_1 \cdots i_k \\ j_1 \cdots j_k \end{pmatrix}$ 的余子方阵的行列式。$i_k^{'}$ 和 $j_k^{'}$ 代表展开行在相应余子式中所在的行号和列号。按照我们处理的顺序，每次总是展开方阵的第一列，因此我们只需考虑展开元素的列号即可。

由于我们总是将节点号大的点汇合成节点号小的店，所以有可能消去的是 1 而剩下的是-1，故有 ***Step3***：*如果选择的边反向，则 sign *= -1。*

由此我们可以得到 2.2.5 中所提出的生成项符号的算法。

由于我们要知道的是 $\det A_L \times \det A_R$ 的值，所以我们无需计算其单独的值，对于(-1)的次数，我们只需考虑展开元素在 A_L 和 A_R 相应余子式中列号之和的奇偶性即可，故有***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（v_1, v_2），其中 $v_1 < v_2$，即改变为正向），现将该边从表示约化子图的数组中删去，然后将数组中所有边结点为v_2的点改成v_1。考察仍保存在数组中的结点，计算结点号小于v_2的结点数目，如果该数为奇数，则sign *= -1。*

由印记矩阵及其变换形式可以看到与符号项 $E_{i,j}$ 和 $\varepsilon^{-1}H_{i,j}$ 相关联的 VS 边的符号总是正的；而与 CS 相关联的 $\varepsilon^{-1}F_{i,j}$ 和 $G_{i,j}$ 的符号总是负的。故有 ***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。*

证毕。

符号与标记（附录 2）

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)
主子行列式	(major)
余子式	(Complementary minor)

参 考 文 献

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

[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] 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.

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

[5] P. Wambacq, G. Gielen, and W. Sansen, "Symbolic Network Analysis Methods for Practical Analog Integrated Circuits: A Survey", *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: ANALOG AND DIGITAL SIGNAL PROCESSING*, VOL. 45, NO. 10, October 1998, pp. 1331-1341.

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

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

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

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

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

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

[12] S. Manetti, "New approaches to automatic symbolic analysis of electric

circuits," *Proc. Inst. Elec. Eng. pt. G*, pp. 22-28, February, 1991.

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

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

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

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

[17] C. Coates, "General topological formulas for linear network functions," *IRE Trans. on Circuit Theory*, vol. CT-5, no. 1, pp. 42-54, March, 1958.

[18] D. Brown, "New topological formulas for linear networks," *IEEE Trans. Circuit Theory*, vol. CT-12, pp. 358-365, September. 1965.

[19] W. Chen, "Topological analysis for active networks," *IEEE Trans. on Circuit Theory*, vol. CT-12, pp. 85-91, 1965.

[20] 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.

[21] M. Hassoun and K. McCarville, "Symbolic analysis of largescale networks using a hierarchical signal flowgraph approach," Kluwer .I. *Analog Integrated Circuits and Signal Prnress*., vol.3, no. 1, pp. 31-42, January. 1993.

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

[23] A. Talbot, "Topological analysis of general linear networks," *IEEE Trans. on Circuit Theory*, vol. CT-12, no. 2, pp. 170-180, June 1965.

[24] 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.

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

[26] M. Sharif-Bakhtiar and M. A. Ahmad, "Symbolic analysis of electronic circuits

based on a tree enumeration technique," *Proceedings IEE*, pt. G, vol. 140, pp. 68-74, Febuary, 1993.

[27] 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.

[28] X. Li and Z. Yin, "The algorithm and program scheme to find out all valid trees and valid tree-pairs," in *Proc. 5th Int'l Conference on ASIC*, (Beijing, China), pp. 298-301, 2003.

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

[30] 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.

[31] 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.

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

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

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

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

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

[37] 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.

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

272-277, 1993.

[39] L. Chua and P. Lin, Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques. Englewood Cliffs, NJ: Rentice-Hall, 1975, ch. 14.

[40] P. Lin, Symbolic Network Analysis. Amsterdam, The Netherlands: Elsevier, 1991.

[41] R．E．Bryant，"Formal Method for Functional Verification".

[42] D. O.Pederson, "A Historical Review of Circuit Simulation", *IEEE Trans. on Circuit and* Systems, Vol. CAS-31, No1, pp.103-111, January, 1984.

[43] X.-D. Tan, 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, June, 2004.

[44] 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.

[45] 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.

[46] Weiwei Chen, Guoyong Shi, "Implementation of a Symbolic Circuit Simulator for Topological Network Analysis", in *Proc. IEEE Asia Pacific Conference on Circuit and System 2006, Singapore,* pp.1327 – 1331, December 2006.

[47] Guoyong Shi, Weiwei Chen, C.-J. Richard Shi, "A Graph Reduction Approach to Symbolic Circuit Analysis", accepted by *12th Asia and South Pacific Design Automation Conference (ASP-DAC 2007), Yokohama, Japan,* January, 2007.

致谢

硕士研究生的学习是一段极其快乐且又收益颇丰的时光。当这篇硕士学位论文即将完成付梓的时刻，回首在交大七年的学习生涯，无不让人感叹成长的时光如白驹过隙于弹指一挥之间。

首先要感谢的就是我亲爱的爸爸妈妈，无论在任何时候都对我无限支持和无微不至的关怀，给了我充分自由的空间去选择自己的发展方向。你们出色的后勤保障工作、轻松温暖的家庭氛围和坚定的信任，让我可以完全没有后顾之忧地做自己喜欢的事情，这种心灵上的自如是你们给我的一笔极宝贵的财富。

学习和研究的工作中最要感谢的是我硕士论文的指导老师施国勇教授。您的悉心指导和帮助让我从论题选择的迷茫之中明确了自己工作的方向，开始逐步学习如何进展研究工作同时领略到世界一流水平的科研工作的巨大乐趣。正是有了您提出的如此优美的生成树枚举理论和对二分判定图的介绍，让我可以顺利地完成这篇论文的工作；也正是有了您不厌其烦地给我修改英语论文，让我学习了如何撰写学术论文，向国际学术界靠拢。更要感谢您对我个人学习规划的大力支持，让我坚定信心又充满自信地去迎接未来的挑战。在您身上，我看到了一个学者的谦虚、勤奋和智慧，我期待有一天我可以像您一样，传播知识、作育英才。再要感谢的是体系结构方向的祝永新副教授。虽然您不是我的任课老师，也不是我论文的负责老师，但是您热情地让我参与您的研究小组，对我的工作提出意见和指导，让我可以在自己的专业领域有具体的涉猎和工作，弥补了毕设课题与专业方向稍有不同的缺憾，也为我今后的研究方向的选择提供了一个很好的基础。和您一起学习和探讨真是一种愉快的享受，您的乐天和真诚无不感染着我们每一个人，作为交大计算机系的同门，您是我学习的楷模。然后就是谢凯年副教授、韩泽耀、赵峰三位老师，在 Nightingale 项目组里共事的三个月让我从你们身上体验到了一个工程师的优秀品质和投入精神。你们的指导是对我参与工程项目的一个完美的启蒙，同时和你们天马行空的交谈大大拓宽了我的视野和思路，你们对我平等和亲切的态度让我倍感珍惜。还要感谢的是我的导师付宇卓教授，虽然由于您工作非常繁忙所以我们的交流相对较少，但是您每次及时地回复我的各种要求的信件，在百忙之中和我讨论研究方向和论文，让我感受到了您对学生的关心和重视；还有各位任课老师，郭炜老师、程秀兰老师、汪辉老师、黄其煜老师、汪宁老师、李章全老师，你们的教学为我打下了扎实的知识基础。另外，学院的胡薇薇老师、杨薛雯老师、郁美娟老师、林佳老师、栾瑞老

师以及我们的班主任董宣如老师，感谢你们平时的辛勤的行政工作，为我们的学习开展和生活的安排付出了巨大的努力，还有在我生病时对我的关心让我体会到了学校集体的温暖。

不得不提的还有我们 04 级微院所有的同学：天真烂漫的贺献辉、一年四季穿裙子的老同学周文婷、热衷填字游戏和九宫格的室友李萱和严青、年纪最大但个子最小的郑秋华、一直和我海阔天空神侃的林明亮和冯鑫、一天一个进步的孙传名、和我热烈讨论儿时动画片的马潇、人不胖却被称作小胖的彭瑞华、皮鞋总是锃亮的邓力群、老是笑眯眯的洪杰、永远踢后卫的钱锋，有着东南亚风情的李景琼，外号“ 小花 ”的姜智华，安排表妹住在我们寝室的谢俊，文笔很好的张晖，一直挺神秘的编译器大师董峰，等等一干众同班同学，虽然大家的专业方向不同，但是有幸可以在一个班级学习、在同一个实验室干活，大家天天生活工作在一起，友谊弥足珍贵。还有李宇飞、徐如淏、张维、陈亮、谢憬几位师兄，在项目中给了我很多的帮助分享给了我很多的经验。当然还有和我一起种 bug、高频率讨论技术走势的金涛师弟和体系结构研究组中做出突出贡献的钟荣荣小师妹，和你们一起的时光充满了欢乐和前进的动力，让我觉得自己更加年轻了。我一直觉得微电子学院是一个生机勃勃、和乐融融、充满希望的集体，师生间的融洽、同学们的勤奋都让我觉得三年前从计算机系选择到这里继续研究生的学习是一个非常正确而明智的决定。

思源湖畔七年的学习生活就要接近尾声，而我们的人生可以说才刚刚起步。感谢大家对我的帮助和支持，更让我满怀期待地憧憬未来，追求卓越，勤俭谦诚，去描绘浓墨重彩的人生蓝图。

攻读硕士学位期间已发表或录用的论文

[1] **Weiwei CHEN**, Guoyong SHI, "Implementation of a Symbolic Circuit Simulator for Topological Network Analysis", Proc. *IEEE Asia Pacific Conference on Circuit and System 2006 (APCCAS 2006), Singapore*, December, 2006, pp.1327-1331.

[2] Guoyong SHI, **Weiwei CHEN**, C.-J. Richard SHI, "A Graph Reduction Approach to Symbolic Circuit Analysis", accepted by *12th Asia and South Pacific Design Automation Conference (ASP-DAC 2007), Yokohama, Japan*. （被 2007 亚洲与南太平洋设计自动化会议录用）

[3] **陈微微**，付宇卓，赵峰，《嵌入式通用处理器的 MP3 解码软件优化方法》，小型微型计算机系统杂志录用。

[4] **Weiwei CHEN**, "Symbolic Analysis of Analog Integrated Circuits", *Embedded Systems and Materials - Research for Advanced Applications, Proceedings of the 1st Chinese-German Summer School in Shanghai, September 18th-28th, 2006*. pp. 150-166, ISBN-10: 3-00-019576-9 / ISBN-13: 978-3-00-019576-1. （第一届中德暑期学校通讯收录）