上海交通大学硕士学位论文

基于二分判定图的模拟电路层次化符号分析方法实现和性能比较

硕士研究生 ：宋阳

学　　号 ：1102109004

导　　师 ：施国勇教授

专　　业 ：电子科学与技术

所 在 单 位 ：微电子学院

答 辩 日 期 ：2012 年 12 月

授予学位单位 ：上海交通大学

---

1 本研究由国家自然科学基金（项目号 61176129）资助。

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

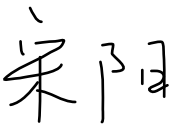
Implementation and Comparison of BDD-based Hierarchical Approaches to Symbolic Circuit Analysis

Candidate：	Yang Song
Student ID:	1102109004
Supervisor：	Prof. Guoyong Shi
Speciality：	Electronics Science and Technology
Affiliation：	School of Microelectronics
Date of Defence：	Dec. 2012
Degree-Conferring-Institution：	Shanghai Jiao Tong University

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文《基于二分判定图的模拟电路层次化符号分析方法实现和性能比较》，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名： 

日期： 2013 年 2 月 18 日

上海交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

保密□，在___年解密后适用本授权书。

不保密☑。

（请在以上方框内打“√”）

学位论文作者签名：宋阳　　指导教师签名：施国勇

日期：2013 年 2 月 18 日　　日期：2013 年 2 月 18 日

基于二分判定图的模拟电路层次化符号分析方法实现和性能比较

摘 要

符号化仿真器具备数值化仿真器所没有的特性，可以作为数值仿真器之外的重要补充。符号化仿真法基于不同的设计原理又可以分为基于代数方法和基于图的方法。基于图的符号化仿真器具有无对消项的优势。在这篇论文中，一种基于图约化的层次化符号分析方法将被提出。本方法使用二分判定图（Binary Decision Diagram）来实现内部的数据共享，是先前图对判定图（Graph-Pair Decision Diagram）的拓展。图对判定图是面向二端口受控源而设计的，而在新的方法中，我们将使用多端口受控源来构造图对。这样，大规模的电路就可以用层次化的方式进行分析。这个新的方法可以保证在每个电路层次中不存在对消项。根据我们的实验结果，该方法可以极大地降低电路分析的复杂度，并且仿真器的实现以及层次化电路的划分仍然可以很简单。本文进而在该方法与另一种基于代数的层次化符号法，层次化行列式判定图（Hierarchical Determinant Decision Diagram）之间进行实验对比。我们发现该方法在确保无对消项的前提下，可以取得与行列式判定图相当的性能。

关键词：模拟电路，二分判定图，无对消项符号法，图约化，层次化分析，多端口分析

IMPLEMENTATION AND COMPARISON OF BDD-BASED HIERARCHICAL APPROACHES TO SYMBOLIC CIRCUIT ANALYSISis

ABSTRACT

Parallel to algebraic methods, graphical methods for symbolic circuit analysis have the advantage of cancellation-free. This paper proposes a graph reduction method for hierarchical symbolic circuit analysis by applying a binary decision diagram (BDD) for data sharing. This method is extended from the Graph-Pair Decision Diagram (GPDD) method which was developed for two-port dependent sources. New graph construction rules for multiple-port dependent sources are introduced, with which large analog circuits can be analyzed hierarchically. The new hierarchical method guarantees the cancellation-free property at each layer of hierarchy. The BDD-based hierarchical analysis method can greatly reduce the analysis complexity of the entire circuit, while the software construction and circuit partition remain easy. The new method is compared to the algebraic hierarchical method based on DDD (Determinant Decision Diagram) which does not have the cancellation-free property.

KEY WORDS: Analog integrated circuits, binary decision diagram (BDD), cancellation-free symbolic analysis, graph reduction, hierarchical analysis, multiple-port analysis

目 录

图 录

表 录

第一章 绪论

1.1 数值化分析与符号化分析

SPICE(Simulation Program with Integrated Circuit Emphasis)是 EDA 领域最早出现的模拟电路仿真工具，提供多种电路分析功能，包括直流分析、交流分析、瞬态分析等等。它的架构在工业界和学术界一直沿用至今，已经成为电路仿真器中的经典。以 SPICE 为代表的数值化电路仿真器依据基尔霍夫电流定律（KCL）建立电路的矩阵方程，进而使用数值计算方法求解方程。由于目前数值计算方法的研究已经高度成熟，数值化仿真器可以取得让人满意的分析速度与效率。并且，随着计算机的进步，数值解也可以达到可观的精度。这是目前模拟电路设计者倾向于使用数值化仿真器的原因。

但从本质上来说，数值化仿真器只能仿真得到在某个特定输入信号的作用下电路的输出信号。设计者所得到的仿真波形不过是各种情况下（比如不同时刻、不同 DC 电压源），不同电路输出状态线性插值得到的结果。波形的变化与电路某个参数（比如电阻、晶体管宽度）之间的关系是被隐藏起来的。电路设计者需要一遍遍地改变电路参数，观察实验结果，并且根据设计经验才能够判断某些元器件对于电路的影响。数值化仿真器在这个方面的功能缺失导致电路设计者更加依赖于设计经验。对于大规模的集成电路设计，更新电路参数后每次都要重新进行矩阵运算，而每次运算都可能会花费几小时甚至几天的时间，大大制约了设计的效率。因此，为了够直观地揭示某个电路参数对电路输出的影响，甚至直接导出电路的传输函数表达式，符号化的分析方法以及相应的仿真器被开发出来了。

符号化仿真器在仿真过程中将电路参数（如电阻，电容）与传输函数之间数学关系保留下来，并导出电路传输函数的解析解。电路设计者可以从解析解得到很多有用信息，比如电路状态对于某个特定电路参数的灵敏度。并且，电路输出的解析式形式不受电路参数的取值的影响。这样就避免了电路参数更新后需要重新求解电路的问题。如图 1-1 所示，在数值法中，每次电路状态（如频率）更新后矩阵方程的构造和求解需要重新执行。而在符号法中，解析表达式的计算只需要执行一次，之后每次电路状态更新后只需要重新计算解析式的数值即可。这样被重复执行的只是一些运算复杂度较低的操作。这种特性使得符号仿真法适合用于需要大量重复运算的统计应用。

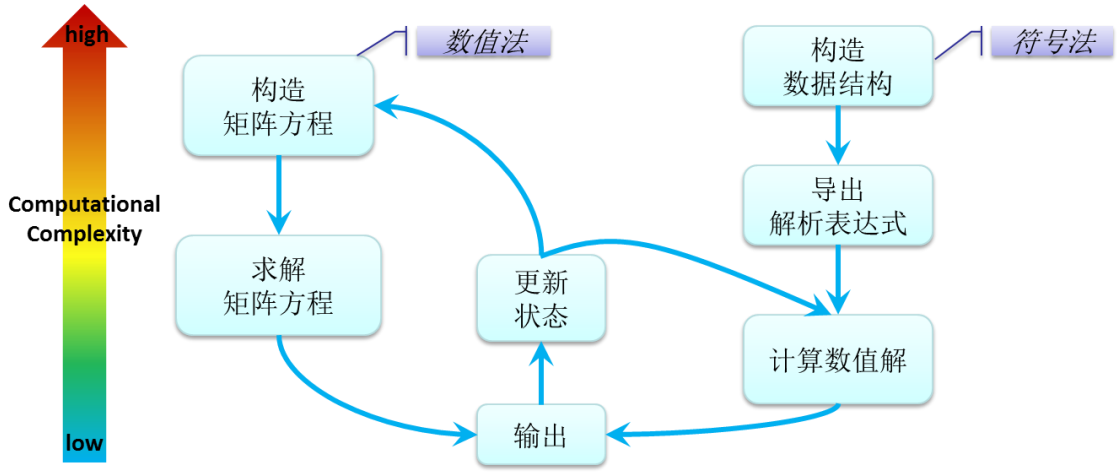


图 1-1 数值法与符号法分析流程

Fig.1-1 Flows of numerical and symbolic analyses

但是与数值化仿真器相同，符号化仿真器也面对着理论上和实现上的缺陷。与数值化仿真器不同的是，符号仿真器的发展并不成熟，因此符号化仿真器的设计面临更大的难度。本课题的研究目的就是克服当前符号化仿真器的性能缺陷，为符号化仿真器在大规模集成电路设计中的应用提供理论支持。

1.2 符号化分析现状

符号化仿真器通常以时间或者频率作为传输函数的自变量，通过读取网表文件建立数据结构，求解出电路输出信号的函数表达。电路中的节点电压、支路电流等都可当作传输函数的输出信号。

符号化仿真器通常用于交流信号的分析中。这是因为在交流小信号电路中，非线性器件将被线性化，于是小信号电路仅由受控源和线性阻抗构成。小信号电路的传输函数可以用整式相除的简单形式表达，这很大地降低了符号方法的实现难度。模拟电路的交流传输函数 $H(s)$ 可以写成如下形式：

$$H(s)=\frac{\sum_{i=0}^{m}a_i s^i}{\sum_{i=0}^{n}b_i s^i} \tag{1-1}$$

其中 $s=j\omega$。s 变量的系数 a_i 或 b_i 可表达为电路参数（如 R，L，C）的函数形式。以图 1-2 中的二阶 RC 梯形电路为例，假设输入信号 Vin = 1，输出信号为电容 C2 两端的交流信号，则此电路的传输函数为。

$$\frac{V_{out}}{V_{in}} = \frac{R_1^{-1}R_2^{-1}}{C_1C_2s^2 + [(R_1^{-1} + R_2^{-1})C_2 + R_2^{-1}C_1]s + R_1^{-1}R_2^{-1}} \tag{1-2}$$

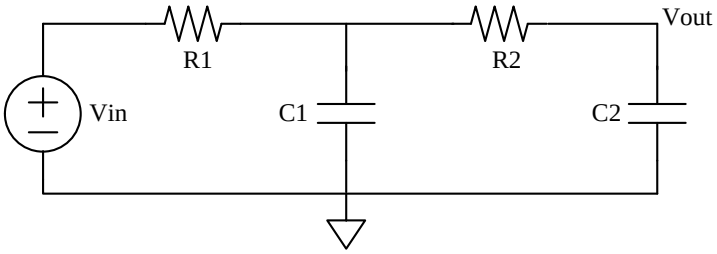


图 1-2 两级 RC 梯形电路
Fig.1-2 RC Ladder with two stages

符号仿真器可以直接得到传输函数（1-1）乃至其 s 系数的解析形式。符号化仿真的具体实现有多种途径，比如较为简单的方法利用了高斯（Gauss）消元过程。

$$\begin{bmatrix} R_1^{-1} & -R_1^{-1} & 0 & 1 \\ -R_1^{-1} & R_1^{-1} + R_2^{-1} + sC_1 & -R_2^{-1} & 0 \\ 0 & -R_2^{-1} & R_2^{-1} + sC_2 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ i_v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ v_{in} \end{bmatrix} \tag{1-3}$$

具体来说，首先根据基尔霍夫定律建立矩阵方程（1-3）。然后在使用高斯消元求解矩阵方程的过程中，我们使用特定的数据结构将矩阵元素的运算关系记录下来。这样我们在得到解的同时，也得到了解和矩阵元素之间的关系。方程解与电路参数之间的关系可以进一步得到。

至今为止，研究者已经对符号仿真器的开发做了多种尝试。但是符号仿真器的实现复杂度与运行效率一直以来困扰着研究者。以上文中基于高斯消元设计的符号仿真为例，理论上高斯消元法的复杂度为$O(n^3)$，因此大规模电路的高斯消元需要上万个迭代步骤。记录高斯消元过程会消耗很大的内存空间。另外，高斯消元过程中的行列置换操作使得符号仿真的实现更加复杂。

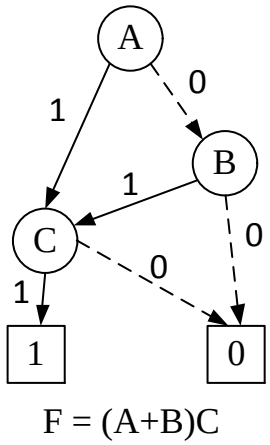
目前，两种基于二分判定图（Binary Decision Diagram）的符号仿真器取得了不错的性能，并且实现也较为简单。本研究将围绕二分判定图以及衍生出来的两种符号仿真器展开。

1.3 基于二分判定图的符号化分析法

1.3.1 二分判定图（Binary Decision Diagram）

二分判定图是一种类似于二叉树的数据结构。它最早由 Akers 在计算机领域

中提出[1]，旨在以较少的空间保存布尔公式。如图 1-3 所示，一个布尔公式可以表达为一个含有单个根节点，单方向且无环路的图。图中可包含多个决策点（Decision Nodes）和终点（Terminal Nodes）。终点又可以分为 0 和 1 两种。从根节点出发，位于同一个层次上的决策点代表同一个符号（symbol）。每个决策点指向两个子节点，分别由 0 型边（虚线）和 1 型边（实线）指向。从某个节点出发，选择 0 型边表示当前节点被赋为 0。相反，选择 1 型边就意味着当前节点值为 1。在这个判定图中，每一种从根节点到终点的遍历方式都对应一种符号取值的组合。如果遍历路径的终点值为 1 就意味着该符号取值组合所对应的布尔函数值为 1，反之则为 0。这样我们用二分判定图来隐式地表示一个布尔表达式或者一个真值表。在本文中，为了表述方便我们所提到的左子树（或节点）是指 1 型边指向的子树（或节点），右子树（或节点）指 0 型边指向的子树（或节点）。



F	A	B	C
0	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0

图 1-3 二分判定图
Fig.1-3 Binary Decision Diagram

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[2]（Reduced Ordered Binary Decision Diagram, ROBDD, 通常简称为 BDD），下文简称二分判定图。一个被简化的二分判定图需要满足两个条件：1）所有同构的子图共享；2）删除具有相同子节点的父节点。可以证明，在符号顺序确定的情况下，二分判定图的结构是唯一的。二分判定图的简化过程自下而上进行，每个节点与它的两个子节点可以唯一确定一个子图。因此这三个节点被用过构造过程中的哈希表键值（hash key）[2]。具体的简化过程不在这里赘述。

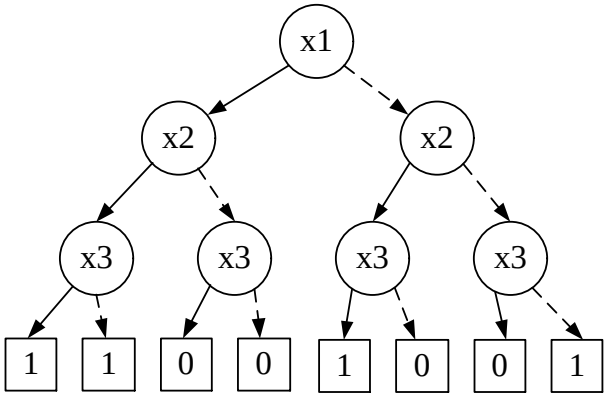


图 1-4 未简化的二分判定图

Fig.1-4 Binary Decision Diagram before reduced

经典二叉树的规模随着符号的增多以指数式递增，而经过简化的二分判定图可以将规模减小到可以接受的程度，甚至在理想的情况下可以获得线性增长速度。以图 1-4 为例，与符号 x3 对应的节点有四个，其中三个节点可被简化。按照从左至右的顺序，第一个和第二个节点的两条边都指向同样的终点，因此这两个节点的取值不会对经过的路径带来影响，它们可以被简化掉。第四个节点的 1 型边指向 0，意味着经过此节点的路径所代表的布尔表达式为零，因此也可以省略该节点。于是这三个节点都可以被移去。简化后的二分判定图见图 1-5。

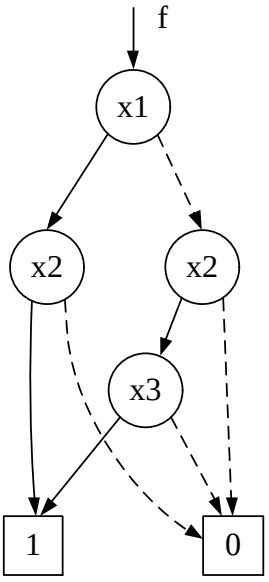


图 1-5 简化二分判定图

Fig.1-5 Reduced Ordered Binary Decision Diagram

二分判定图的缺陷在于，它的规模依赖于符号的排序（order）。一个理想的排序可以最大限度地共享子图，进而减少判定图的规模。而一个糟糕的排序可能导

致没有子图可以共享，在这种情况下，二分判定图就被构造成了一个二叉树。它的规模以指数式增长。对于符号排序的设计，目前尚没有一个明确的方法。研究者探索过一些启发式排序法，但是它们的效率视具体应用而定，因此不具备通用性。不过，对于大部分应用二分判定图还是可以取得不错的性能。

相对于真值表，二分判定图的低内存占用量使得它在计算机领域获得了广泛的应用。同时，由于二分判定图具有唯一性，它也适合行为验证以及电路逻辑综合等方面的应用。目前两种基于二分判定图的符号仿真器都取得了不错的效果，下面将具体介绍这两种仿真器。

1.3.2 行列式判定图（Determinant Decision Diagram）

行列式判定图（Determinant Decision Diagram，简称 DDD）在[3]中被提出。作为二分判定图的一种应用，行列式判定图将二分判定图用于行列式的表达。已知任意一个行列式可以通过拉普拉斯展开而表示成行列式元素的多项式函数。如（1-4）所示，一个 3 阶的行列式经过一次拉普拉斯展开后得到若干个 2 阶行列式。对于 n 阶大小的行列式，通过 n-1 次迭代展开就能够得到各行列式元素的乘加形式。

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} \qquad (1\text{-}4)$$

注意到在展开过程中，每一个被选中的行列式元素（符号）都将被展开成该符号（symbol）的与代数余子式（minor）的乘积与剩下的行列式（remainder）相加的形式，即 $symbol * \text{minor} + \text{remainder}$。二分判定图可以用来记录符号间的加法与乘法关系。具体来说，二分判定图中的每个节点代表被选中的符号。节点引申出的 1 型支路（实线）表示代数乘法，0 型支路（虚线）表示代数加法。因此 1 型支路所指向的子图就表示与该符号相乘的代数余子式，而 0 型支路所指向的子图就表示该符号置零后得到的行列式。随着拉普拉斯的逐层展开，行列式的解析式就被保存在一个二分判定图中了。值得注意的是，代数余子式的符号保存在行列式判定图的支路中。于是，1 型分支的数值可能为 1 或者-1，而 0 型分支的数值为 0。

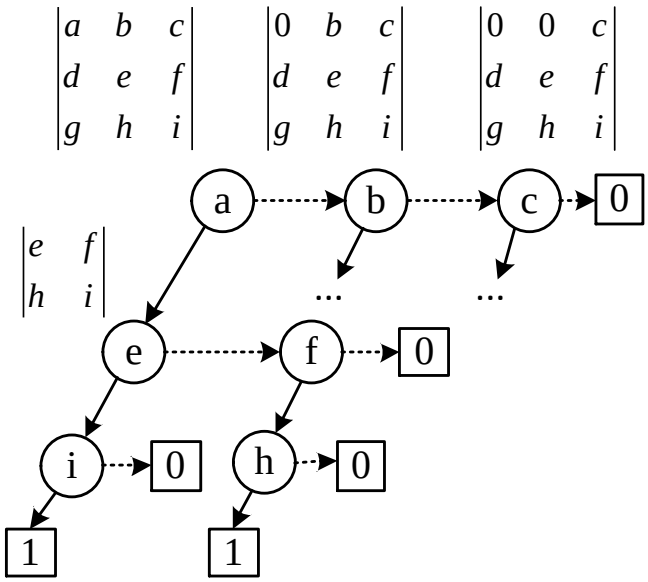


图 1-6 行列式判定图的构造
Fig.1-6 Construction of DDD

图 1-6 中一个 3 阶矩阵被展开，同时一个行列式判定图被构造出来。在这个行列式判定图中，节点 a 为根节点。从节点 a 出发遍历判定图，直到到达终点 1，其间每一种遍历路径都对应着一个行列式完全展开后的乘积项。从集合的角度来理解，行列式判定图其实是对乘积项的枚举过程。例如 aei 是行列式展开后得到的一个乘积项，若把 aei 视为一个集合，a、e 和 i 是其中的元素。位于集合中的元素用 1 来表示，不包含的元素用 0 来表示。那么这个枚举过程就可以用一个真值表或者布尔函数来表达。因此可以构造出一个二分判定图来表达这个枚举过程。

作为二分判定图的一种应用，行列式判定图的规模也受符号排序的影响。在这里，符号顺序也就是行列式元素展开的顺序。通常应用中符号展开的顺序遵循最小度数展开原则，即优先考虑包含非零元素个数最少的行或列，比如贪心排序法（greedy-labeling algorithm）。在目前没有更好算法的情况下，最小度展开实现简单而且大多数情况下有不错的性能，尤其对于稀疏矩阵的求解。

在传统二分判定图的简化过程中，三元组[2]被用做哈希表项的键值（key）。并且每次判定图在自上而下（自根节点至终点）构造完毕之后，还需要自下而上进行简化。这种简化方法虽然永远可行，但是时间开销较大，哈希表的实现也较为繁琐。在[4]中，作者发现了简化行列式判定图的简单方法。既然每个子图对应一个行列式，那么将行列式作为哈希表的键值即可。进而由于拉普拉斯构造过程中，所有出现的余子式都是原行列式的子式，那么只需要保存这个余子式在原行

列式中相对的行号和列号即可，行列式只需要保存一份。这种实现方法不仅简单，更重要的是在拉普拉斯展开的过程中每个余子式的行号和列号是已知的，因此判定图的简化可以同时进行。

根据 Cramer 法则，形如 $Ax=b$ 的线性方程的单个解可以表达为两个行列式相除的形式。基于行列式判定图的符号仿真器使用行列式判定图分别导出解的分母与分子解析式。这种仿真方法不仅简单而且取得了不错的效率。在使用[4]中哈希表构造方法的前提下，运算放大器如 uA741 可在几秒钟内分析完毕。

1.3.3 图对判定图（Graph-Pair Decision Diagram）

二分判定图不仅可以用于代数运算中，它也可以用于图的操作中。基于二分判定图，G. Minty 在[5]中提出了一种在图中枚举所有遍历树的算法。如图 1-7 所示，在 Minty 算法的每一步操作中，被选中的分支有两种可能的操作，分别为开路和短路。每次短路后都会有相邻的节点被合并。按照这种方式不断迭代下去，直到原图被约化为一个节点，或两个不连通的子图。若最终可以得到一个点，则说明被短路的分支可以构成一个遍历树。若图的连通性被破坏，则说明某个应该被遍历树包含的分支被移去，因此剩余的分支不能构成一棵遍历树。实际上这个约化的过程就是对遍历树树枝的枚举的过程，短路相当于选中当前的支路而开路相当于不包含该支路。

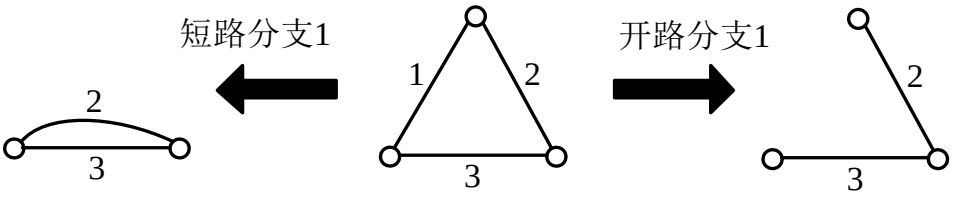


图 1-7 图的归约

Fig.1-7 Graph reduction

显而易见，二分判定图可以用于这个枚举的过程中。每条支路可被视为一个符号，由判定图中的节点表示。判定图中的 1 型分支表示选择该支路，0 型分支表示删除该支路。终点 1 代表原图被约化为一个点，而终点 0 表示原图被断开。在简化二分判定图时，归约图被用作哈希表的键值。对应相同归约图的子判定图被父节点共享。

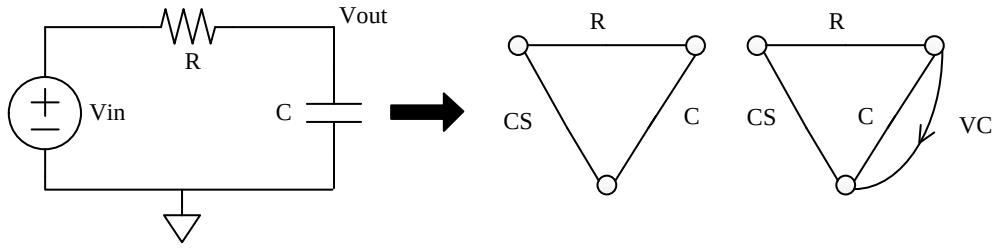


图 1-8 电路图的抽象
Fig.1-8 Abstract of a linear circuit

受 Minty 图约化算法的启发，在[6]中作者提出了基于图的电路符号仿真方法，即图对判定图（Graph-Pair Decision Diagram，简称 GPDD）。首先，小信号电路被抽象为一对有向图（见图 1-8），其中每个线性器件用一条支路表示，每个受控源由一对支路表示。在约化图对的过程中，每个器件被当做一个符号并且遵循相应的归约方法[6-8]。该符号的数值也就是相应线性器件的导纳，或者受控源的增益。电路的输出及输入端被视为一对控制与受控源，且作为归约过程中处理的第一个符号。根据作者在[6]中的证明，电路的传输函数即为根节点的左右子图之比的相反数。一个简单的低通滤波器的判定图构造见图 1-9，其中符号 X 代表输入端口和输出端口之间的等效受控源。X 的左子图的值为 R^{-1}，右子图的值为 $R^{-1}+sC$。因此该电路的传输函数为：

$$H(s)=\frac{R^{-1}}{R^{-1}+sC} \tag{1-5}$$

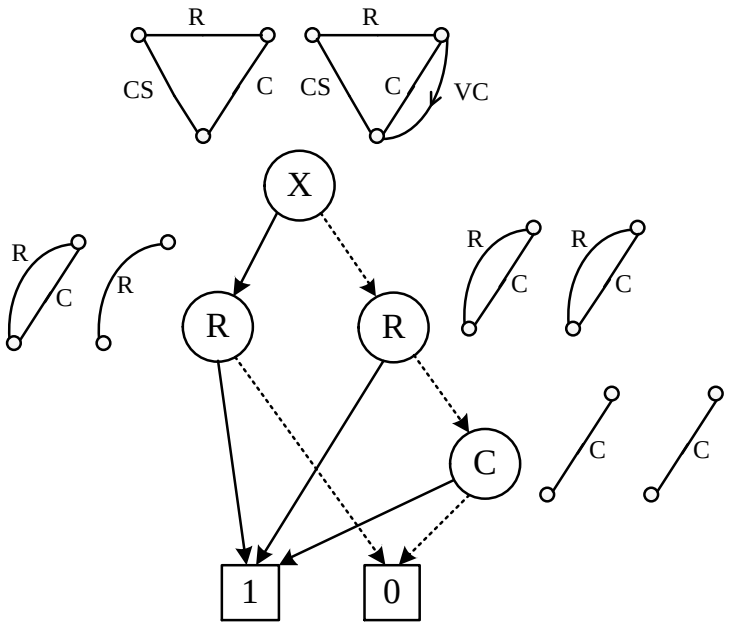


图 1-9 图对判定图的构造

Fig.1-9 Construction of GPDD

在近几年的发展中，行列式判定图与图对判定图都取得了不错的性能。它们都属于目前符号化仿真器中的佼佼者。但同时，它们也受到二分判定图缺点的限制，比如图规模对符号顺序的依赖。在本研究中，图对判定图将作为研究的终点，同时以行列式判定图为对照。因此，我们有必要在两者之间做一个简单的对比。与行列式判定图相比，图对判定图有如下几个特点：

- 更新快
  由于图对判定图中的符号对应电路中的器件，因此当电路的参数改变时，图对判定图可以立即更新符号数值。而行列式判定图中的符号对应着行列式中的元素，因此需要先更新行列式方可更新判定图。另外，这种符号与电路器件的直接联系也使得图对判定图适用于灵敏度分析[9]。

- 规模大
  图对判定图的大小决定于电路器件的个数，即电路图中支路的个数。而行列式判定图的规模主要取决于行列式的阶数，大致相当于电路中节点的个数。对于同一个电路，图对判定图自然会更大。

- 不稳定
  行列式判定图中，以最小度优先的行列展开方法可以取得不错的效果，尤其对于稀疏矩阵。大多数情况下可以得到较好的符号排序。而在图对判定图中，尚且没有一个较为可靠的启发式排序法。图对判定图的规模时常由于较差的排序而激增。

- 无对消项
  行列式判定图导出的解析式展开后往往会出现对消项（cancelling term），而图对判定图不存在对消项。对消项的概念将在下一小节详细展开。

1.3.4 对消项（Cancelling Term）

对消项是指在解析式的展开形式中，可以互相抵消的乘积项。仍以两级 RC 梯形电路为例（见图 1-2），首先对其系数矩阵 A 的行列式展开：

$$\begin{aligned}\det(A) &= -(R_1^{-1}+R_2^{-1}+sC_1)(R_2^{-1}+sC_2)+R_2^{-2} \\ &= -C_1C_2s^2 - R_1^{-1}C_2s - R_2^{-1}C_2s - R_2^{-1}C_1s - R_1^{-1}R_2^{-1} + R_2^{-2} - R_2^{-2} \\ &= -s^2C_1C_2 - [(R_1^{-1}+R_2^{-1})C_2 + R_2^{-1}C_1]s - R_1^{-1}R_2^{-1}\end{aligned} \tag{1-6}$$

注意到拉普拉斯展开后得到的解析式并不是最简形式。在化简的过程中，两个 R_2^{-2} 相互抵消。而对于更大规模的电路，这种对消项的出现更频繁。对消项出现

的原因在于，在构造 Modified Nodal Analysis (MNA)矩阵的过程中，每个线性器件都会最多将导纳填入 MNA 矩阵四次。因此在导出的解析式中，有可能会出现构成相同符号相反的乘积项，即对消项。而在图对判定图中，每个器件作为一个符号只在图对中出现一次。因此，不会出现构成相同的乘积项。图对判定图导出的解析式可以保证为最简形式。

尽管它们最终相互抵消，但是对消项的存在带来了额外的计算负担，还会引入计算误差。随着计算机硬件的进步，高精度计算保证由于多次计算而引入的计算误差被限制在很小的范围内。但是在某些考虑参数误差的计算中，比如区间计算，对消项的存在会导致过于保守的区间估计。比如，在[10]中，当把行列式判定图用于区间分析时，需要执行去消去项（de-cancellation）操作，以避免过于松散的区间估计。但是这样却引入了额外的操作，同时消耗了额外的时间。

1.4 基于二分判定图的层次化符号分析方法

两种基于二分判定图的符号仿真法各有自己的优势，但是他们的效率同时受到二分判定图符号顺序的影响。在符号顺序不理想的情况下，判定图的规模会呈指数式增长。在分析大电路时，判定图时常会得太大导致仿真程序消耗掉电脑的全部内存空间。因此，非层次化的判定图通常不能用来分析大规模的模拟电路。为了获得稳定的空间和时间消耗，研究者相继发展出层次化的符号仿真器。我们将在这一节详细介绍相关背景。

1.4.1 舒尔分解（Schur Decomposition）

对于一个大规模的矩阵，我们可以使用舒尔分解将其分解成若干个小矩阵。具体来说，对于任意给定的一次矩阵方程 $Ax=b$，我们假定它呈如下形式：

$$\begin{bmatrix} A^{II} & A^{IB} & 0 \\ A^{BI} & A^{BB} & A^{BR} \\ 0 & A^{RB} & A^{RR} \end{bmatrix} \begin{bmatrix} x^{I} \\ x^{B} \\ x^{R} \end{bmatrix} = \begin{bmatrix} b^{I} \\ b^{B} \\ b^{R} \end{bmatrix} \tag{1-7}$$

其中 A^{II} 为非奇异矩阵， A^{IB} 与 A^{BI} 包含非零元素。易证如果(1-7)成立，则下式也成立：

$$\begin{bmatrix} A^{BB}-A^{BI}(A^{II})^{-1}A^{IB} & A^{BR} \\ A^{RB} & A^{RR} \end{bmatrix} \begin{bmatrix} x^{B} \\ x^{R} \end{bmatrix} = \begin{bmatrix} b^{B}-A^{BI}(A^{II})^{-1}b^{I} \\ b^{R} \end{bmatrix} \tag{1-8}$$

这样子矩阵 A^{II} 及其所对应的行和列就被压缩掉了。受舒尔分解的启发，X.-D.

Tan 将电路分析过程中建立的 MNA 矩阵进行压缩 [11-12]。这样根据压缩后的矩阵所构造的行列式判定图不会太大。

使用舒尔分解要解决的一个关键问题在于如何选定被压缩的内部矩阵 A^{II} 。研究者发现电路中往往存在子电路模块，比如一个场效应管的内部等效电路，或者一个运算放大器的内部结构等。在电路分析时，这些子电路模块的内部状态，比如节点电压，并不是电路设计者所关心的问题。如果把所有的子电路展开，构造出一个大规模矩阵方程中，相当一部分行和列所对应的变量是没有必要求解的。因此，在[11-12]中作者将电路变量分为三种，分别为子电路内部变量 x^I ，子电路端口节点的变量 x^B 和剩余的变量 x^R 。而内部变量 x^I 所对应的行和列构成的矩阵即为内部矩阵 A^{II} 。在建立行列式判定图前，内部变量及其所对应的 A^{II} 及 b^I 都将被压缩掉。

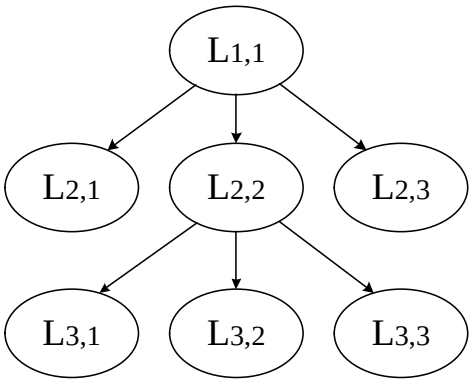


图 1-10 层次化电路

Fig.1-10 Hierarchical circuit modules

舒尔分解允许连续压缩同一个矩阵，也允许同时选取不同的内部矩阵进行压缩。因此，电路可以被分成不同的层次，而每个层次中又可以包含不同的电路子模块。如图（1-10）所示，这就是层次化的电路结构。注意到在执行舒尔分解（1-8）之前，需要首先计算得到内部矩阵的逆。于是在层次化电路的仿真过程中，底层电路首先被分析。

在使用舒尔分解之后，层次化的行列式判定图（Hierarchical Determinant Decision Diagram, 简称 HDDD）可分析的电路规模大大提高。根据[11-12]中的实验结果，晶体管运算放大器 uA725 可以在几秒钟之内仿真完毕。但是，舒尔分解属于矩阵操作，因此不适用于非基于矩阵的符号仿真法。下面我们将介绍另一种更为通用的层次化办法。

1.4.2 符号化导纳矩阵（Symbolic Stamp）

符号化导纳矩阵即为导纳矩阵的符号表达。根据电路基本原理，对于任意一个有 n 个端口的线性器件或小信号电路，它在端口的电压电流关系可由一个 n 阶的导纳矩阵表达。一个 3 阶的导纳矩阵如下：

$$\begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} \tag{1-9}$$

在一个层次化电路中（如图 1-10），已知一个子电路的导纳矩阵，我们可以将其填入父级电路模块的 MNA 矩阵中。这样在保留子电路端口电压电流关系的同时，也避免了父级电路矩阵方程的变量太多。这就是基于符号化导纳矩阵的层次化方法。相对于舒尔分解，这种方法更容易理解。而且该方法也适用于基于图的符号仿真法。无论对于行列式判定图还是图对判定图，符号化导纳矩阵都可以用一个多端口的判定图表示。

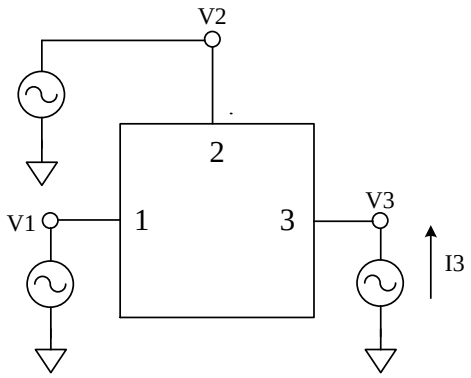


图 1-11 导纳测试电路

Fig.1-11 Test circuit for trans-admittance

导纳矩阵的计算可用测电流的方法取得。以一个三端口的电路模块为例（见图 1-11），若端口 2、3 的电压设为零，即 $v_2 = v_3 = 0$，且端口 1 的输入电压为 1，则易知流入端口 3 的电流即为（1-9）中的 y_{31}。这个测试电路的计算可以用（1-10）来描述：

$$\left[\begin{array}{c|c|c} A^{II} & A^{IB} & \\ \hline & & \begin{matrix}-1 & \\ & -1 \\ & & -1\end{matrix} \\ \hline A^{BI} & A^{BB} & \\ \hline & \begin{matrix}1 & \\ & 1 \\ & & 1\end{matrix} & \end{array}\right]\left[\begin{matrix}\vec{x}_I \\ v_1 \\ v_2 \\ v_3 \\ i_1 \\ i_2 \\ i_3\end{matrix}\right]=\vec{b} \tag{1-10}$$

其中，$\vec{x}_I$ 为电路内部变量构成的解向量，而 A^{II} 为相应的内部系数矩阵。$v_1 \sim v_3$ 与 $i_1 \sim i_3$ 分别表示端口电压和电流。通过改变端口施加的电压，并计算相应的电流变量 $i_1 \sim i_3$，我们就可以得到导纳矩阵中的任意元素。注意到在这个过程中，只有激励向量 $\vec{b}$ 是需要改变的，而 MNA 矩阵保持不变。根据 Cramer 法则，不同的电流变量解的分母都是 MNA 矩阵的行列式（见图 1-12）。于是不同电流解的分母相同。我们可以使用同一个行列式判定图来表达不同导纳的分母。另外不同电流解的分子部分虽然有些许不同，但是在拉普拉斯展开的过程中，时常会出现同样的余子式。这些余子式可以通过哈希表检测出，进而共享。于是，表示不同导纳的判定图相互共享，进而形成了一个多根节点（multi-root）的判定图。

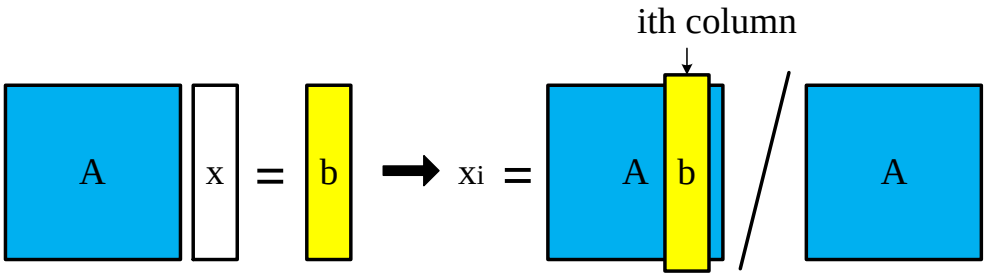


图 1-12 克莱姆法则
Fig.1-12 Cramer's rule

对于图对判定图，导纳矩阵也可以用一个多端口的判定图来表达。以一个二端口的电路模块为例（见图 1-13），已知输入激励为端口电压，输出为端口电流。跟据[6]中的图对构建规则，一个从输出口至输入口的受控源器件将被引入，即电流控制电压源（CCVS）。假设求解导纳 y_{12}，则把端口 2 作为输入端口，端口 1 接地并且流入端口 1 的电流作为输出电流。于是端口 1 被当做 CCVS 的电流控制源（CC），而端口 2 被当做 CCVS 的电压受控源（VS）。为了求解导纳 y_{12} 而建立的电路图见图 1-13 右上。根据图对构建规则，如果电路中不包含电压控制源（VC）或电流受控源（CS），那么一对图的结构相同。因此在这里，我们用一张图来表

示图对。导纳 y_{21} 所对应的图对构造过程类似（图 1-13 左上），因此不在这里赘述。

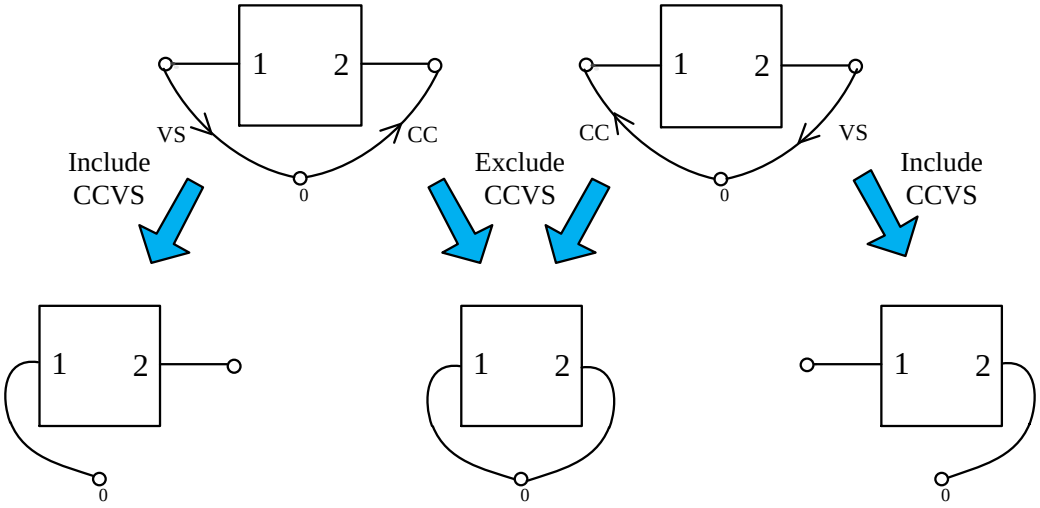


图 1-13 二端口电路图的归约
Fig.1-13 Graph reduction for a 2-port circuit

在图对构造好之后，就可以在进行图对判定图的构建了。端口处的 CCVS 器件将作为第一个符号处理。按照图对归约规则，在对 CCVS 器件进行移除（exclude）操作时，CC 与 VS 器件对应的支路都将被短路。于是所有端口都被接地。对于任意导纳所对应的图对，在移除 CCVS 之后得到的新图对是一样的（图 1-13 中下）。因此，表示这个新图对的判定图在不同导纳直接共享。整个符号化导纳矩阵的不同项由一个多根节点的图对判定图（图 1-14）来表达。

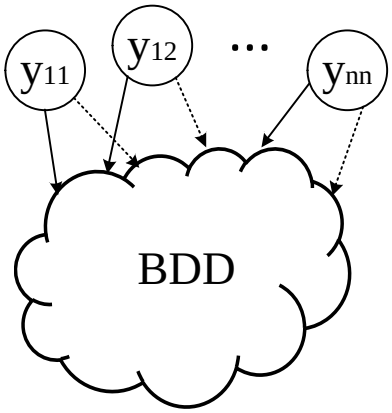


图 1-14 多端口判定图
Fig.1-14 Multi-root GPDD/DDD

在符号化导纳的基础上，[14-15]提出了层次化的判定图仿真器。这种层次化仿真器将电路分为主电路和子电路两个层次。行列式判定图和图对判定图被分别

用来求解主电路的输出与子电路的导纳矩阵。这种混合式的方法对两种判定图的融合做出了有意义的尝试，也取得了不错的效果。用于混合式层次化仿真器的测试电路至多可包含 44 个场效应管。在[14-15]中，作者还对于电路子模块的划分进行了探索。注意到 MOS 管中包含两个内部节点（图 1-15）。如果将这些节点除掉，那么大规模 MOS 管电路的 MNA 矩阵规模可以大幅度减小。因此在[14-15]中每个 MOS 管的等效电路被当做一个子电路模块来处理。再注意到每个 MOS 管等效电路的结构是相同的，因此我们可以使用同一个图对判定图来表示不同的 MOS 管，实现数据共享。在[16-17]中这种混合式层次化判定图被用在了灵敏度分析的应用中。

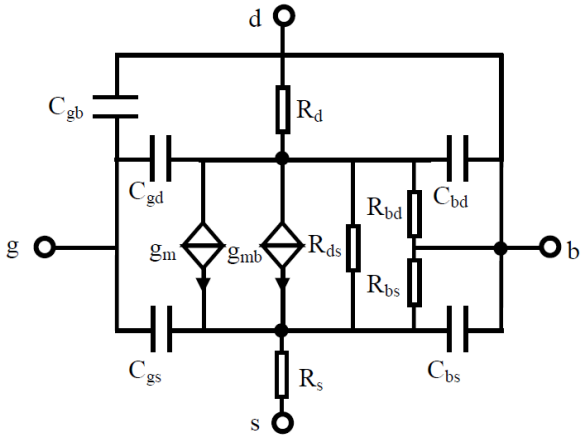


图 1-15 MOS 管等效电路图
Fig.1-15 Equivalent small signal circuit of MOS transistors

1.5 课题提出和研究意义

在[14-15]提出的层次化符号仿真器第一次将图对判定图运用于符号化仿真中。但是这种基于图对判定图的仿真器在充分利用图对判定图的数据共享的优势时，也存在两点缺陷：

- 可分析的电路规模有限

混合层次化仿真器只能将图对判定图运用于最底层的电路层次。在所使用的测试电路中，电路无一不被划分为两个层次。这种划分法对于更大规模的电路就可能不够了。

- 存在对消项

混合层次化仿真器使用行列式判定图分析上层电路。在求解上层电路 MNA

矩阵的时候，经常会出现对消项。而图对判定图无对消项的优势就无法体现出来

基于上述两个主要缺陷，我们提出了无对消项的层次化符号仿真器研究。旨在确保图对判定图无对消项的同时，提高图对判定图仿真器可处理的电路规模。无对消项的层次化图对判定图将是对符号化仿真器领域研究的重要补充。

第二章 无对消项层次化符号分析方法

在上一章中，我们介绍了基于二分判定图的两种符号仿真器的设计，以及在这个基础上层次化仿真器的发展。在本章，我们将详细介绍完全基于图对判定图的层次化仿真器的设计，主要包含三个方面：

首先，层次化的图对判定图仿真器原理及数学模型将做详细介绍。层次化的关键一步是多端口器件的引入。它成功将导纳矩阵映射入电路图中，并用于图对判定图的构建。

其次，合理有效的电路层次化方法可以降低仿真器的内存消耗，并且提高仿真速度。我们将举例说明适合于图对判定图仿真器的电路层次划分方式。层次化方法应当最大限度地利用图对判定图的特性。

最后，为了保证数学模型可以正确实现，自动化工具开发者需要对层次化图对仿真器设计合理的数据结构。并且我们将与其他两种层次化符号仿真器做对比。

2.1 多端口器件

根据上一章中对导纳矩阵的介绍，导纳矩阵可以用来表示子电路模块端口处的电压电流关系，这足以用来进行上层电路的仿真，同时避免了单个电路模块的规模太大。图对判定图的层次化设计也是基于这个想法。为了实现层次化，首先要解决的问题就是如何将子电路的导纳矩阵用于上层电路的仿真过程中。

在[6-8]中，一系列的图对操作规则被定义。可处理的元器件中既包括阻抗元件，也包括线性控制源和受控源。一个具有任意多个端口的子电路也可以被认为是一个多端口线性器件，但是与之匹配的图对操作规则没有在先前的研究成果中出现。因此多端口器件在图对操作中的引入是实现层次化的关键一步。

$$\begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} \tag{2-1}$$

还是以一个 3 端口的子电路模块为例，它的导纳矩阵见（1-10）。根据导纳矩阵，每一个端口电流都可以表示为如下形式：

$$i_k = y_{k,1}v_1 + y_{k,2}v_2 + y_{k,3}v_3, k = 1,2,3 \tag{2-2}$$

注意到每个端口电流都包含三个成分，分别为 $y_{k,1}v_1$， $y_{k,2}v_2$ 与 $y_{k,3}v_3$。这三部分分别与端口电压 v_1， v_2 和 v_3 呈常数倍数关系。因此我们可以使用三个 VCCS 器件来实现（2-2）中的物理关系式，这三个 VCCS 分别为 $i_{k,1}=y_{k,1}v_1$， $i_{k,2}=y_{k,2}v_2$，$i_{k,3}=y_{k,3}v_3$。我们进而使用三个分支对（edge pair）对在图中表示这三个 VCCS。以端口 1 为例，三个分支对如图 2-1 所示。

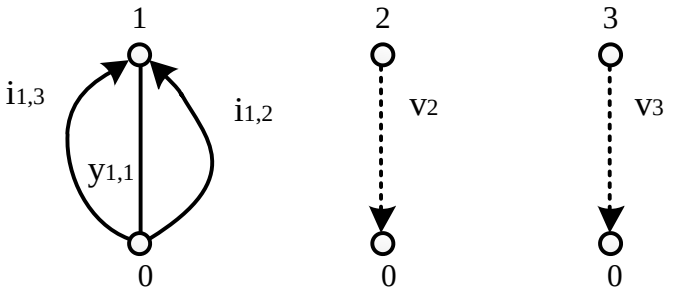


图 2-1 三端口模块图对示意图
Fig.2-1 Illustration of edge-pairs for a three-port module

注意到 $i_{1,1}=y_{1,1}v_1$ 中的电压电流位于同一个端口，因此我们可认为这是一个大小为 $y_{1,1}$ 的导纳器件。我们用一条支路表示导纳器件。而其余电流成分 $i_{1,2}$ 与 $i_{1,3}$ 为分别受 v_2 与 v_3 控制的电流源，增益为 $y_{1,2}$ 和 $y_{1,3}$。

在把每个端口电流分为三个部分后，我们将这个 3 阶的导纳矩阵用 9 个 VCCS 器件表示（如图 2-2）。这就相当于用这 9 个 VCCS 器件替代原子电路模块。同理，一个 n 端口的子电路可以使用 n x n 个 VCCS 替代。

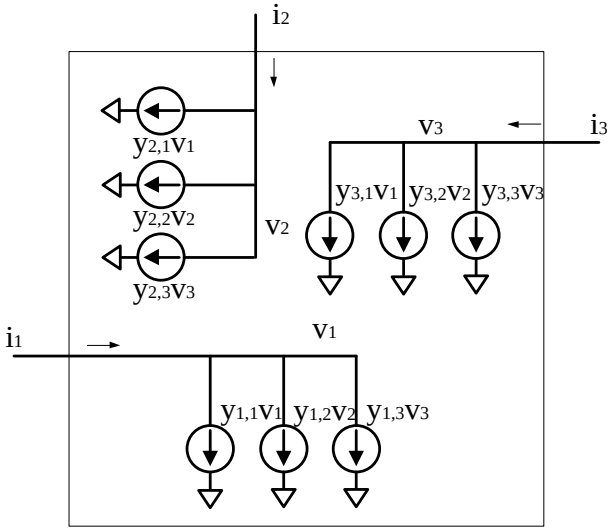


图 2-2 导纳矩阵等效电路图

Fig.2-2 Equivalent circuit of trans-admittance matrix

根据上文的论述，我们可以使用若干个 VCCS 器件来替代一个子电路模块。每个 VCCS 的增益系数对应着导纳矩阵中的某个元素。而符号化导纳矩阵可以通过对子电路建立图对判定图而得到。这样一个大电路就被划分成若干个小模块并单独分析，避免了单层电路的图对判定图太大的情况。多端口器件在图对处理中的引入是实现层次化的核心技巧。

2.2 层次化电路

在具备了层次化方法之后，下面就是如何将电路层次化的问题了。对于一般大规模电路，按照功能电路可以将其划分为若干个模块与层次。以一个有源滤波器为例（见图 2-3）。这个电路可以分为三个层次，每个层次分别包含一个电路模块：第一个层次由运算放大器和无源器件构成，它完成了电路的功能，即对输入信号进行滤波；第二个层次为运算放大器的内部电路，这个电路完成的功能即放大输入的差分信号；第三个层次为运算放大器内部的晶体管等效电路。

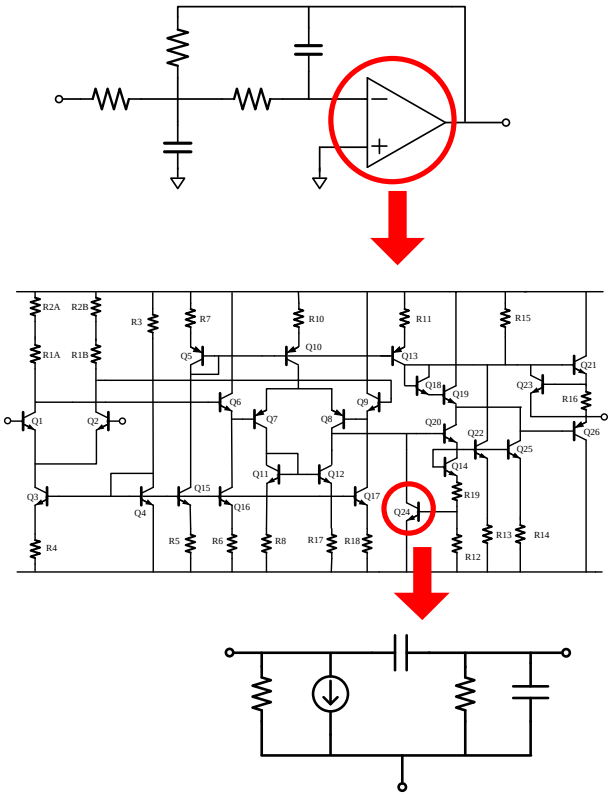


图 2-3 层次化电路示意图

Fig.2-3 Illustration of a hierarchical circuit

对于更为复杂的电路，可能会出现不相邻的电路层次之间的调用。比如在图 2-3 中的电路，若顶层电路模块中出现晶体管，那么顶层电路会直接调用最底层的电路。或者进一步对电路进行模块划分。图 2-3 中位于第二层的运算放大器可以分为不同的部分，如若干级放大电路模块，电流源模块等。这样，同一个层次中就会出现不同的电路模块。一个较为复杂的电路层次化方法可以得到图 2-4 中的电路结构。

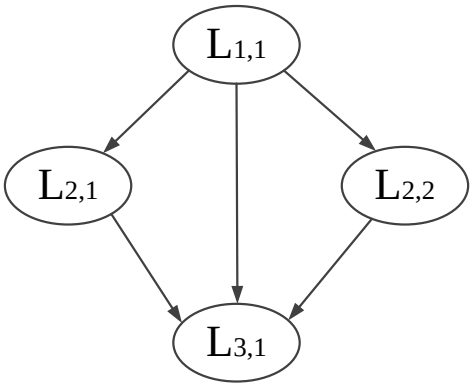


图 2-4 层次化电路模块
Fig.2-4 hierarchical circuit modules

这种层次化的电路中每个层次都具备相对独立的功能。在电路分析时一般只关注其中的某个模块，甚至常常只关注电路的最高层模块的输出，而其余的层次和模块是不需要观测的。按照功能划分电路的方式既符合电路设计者的需要，也有效避免了单层电路仿真的大量运算。

另外，这种按照功能划分的方式也有利于电路模块的共享。由于图对判定图的构造是基于电路结构的，因此对于结构相同但是电路参数不同的电路，比如处于不同状态中的 MOSFET 管，它们可以使用同一个判定图来表示。在仿真时只需要更新判定图的符号数值即可。这种同结构电路之间的数据共享是基于舒尔分解的层次化仿真器所不具备的。仍以图 2-4 中的电路结构为例，若使用基于舒尔分解的层次化符号仿真器来分析，父级电路不能共享子电路，因此图 2-4 中的子电路在每次被调用的时候都需要创建一个新的空间保存数据结构。这种不存在共享的情况下我们会得到图 2-5 中的模块结构。它比图 2-4 中的结构多出三个子电路空间。对于共享关系更为复杂的电路，这种数据共享前后的差别会更加明显。如果电路中包含若干个较大规模的子电路，如运算放大器，则图对判定图的数据共享功能可以大大减少空间消耗。

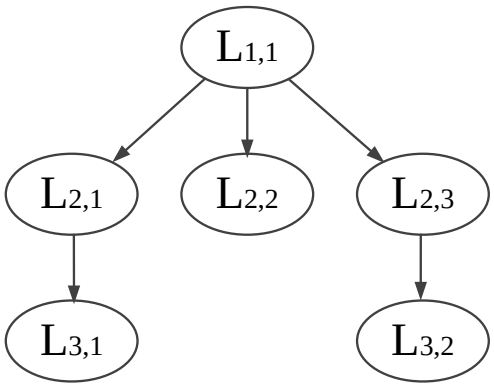


图 2-5 无共享层次化电路模块
Fig.2-5 hierarchical circuit modules without data sharing

2.3 层次化符号化分析

多层次符号化仿真方法可分为两大阶段，分别为构造（construction）过程和求值（evaluation）过程，如图 2-6 所示。

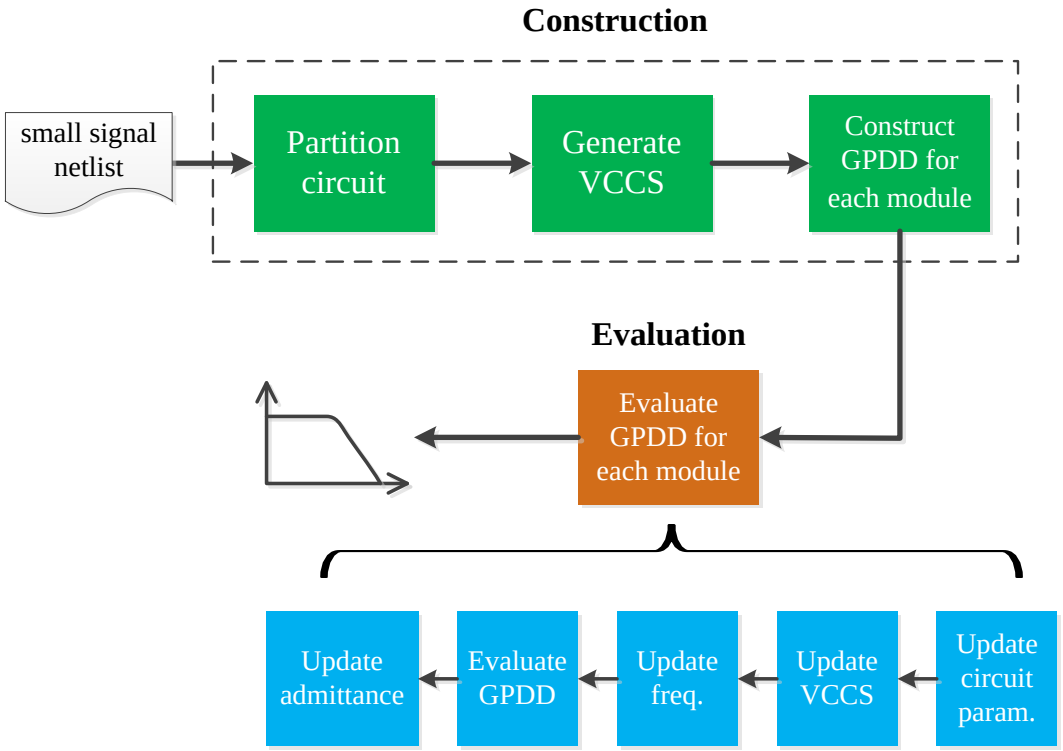


图 2-6 层次化图对仿真器分析流程
Fig.2-6 Analysis flow of hierarchical GPDD

在得到小信号电路的网表之后就可以开始图对判定图的创建了。层次化图对判定图构造过程一共包含三个步骤：

a) 电路划分，得到层次化网表。

b) 插入 VCCS 器件替代子电路模块。

c) 为每个电路模块构造图对判定图。

在进行电路划分时，我们按照电路的结构及功能将其分为若干个小模块。与[12-15]中的电路划分所不同的是，层次化图对仿真器将电路子模块划分出去的同时会引入 VCCS 器件。VCCS 器件的数量等于端口个数的平方。假如电路子模块中的器件个数少于生成的 VCCS 器件的个数，那么这种划分方式是适得其反的。以一个 MOS 管等效电路为例（见图 1-15），假如四个端口 d、g、s、b 分别连接在不同外部节点上，那么需要生成 16 个 VCCS 器件。而 MOSFET 等效电路内部只有 12 个器件。因此这种以一个 MOSFET 管为单位的子电路划分方式反而会增大上层电路的规模。在本文的第四章实验部分我们将会看到，测试电路都以较大的基本模块进行划分。每个基本模块都可以包含几个 MOSFET 管。

在电路划分方式确定后，就可以在每个父级电路模块中加入若干个 VCCS 器件。这些 VCCS 器件的数值由多端口子电路模块的导纳矩阵确定。在求值时 VCCS 的符号数值将被不断更新。

再插入 VCCS 器件之后，每个电路模块的网表就在保持原有的功能的前提下完全分离开。每个电路模块的网表是相对独立的。因此图对判定图的构造对于不同的电路模块也是独立进行的。不同电路模块的构造顺序可任意。

在构造过程结束后仿真流程就进入求值阶段。注意到同一个子电路模块可能会被不同的高层电路调用。根据上一小结的讨论，子电路模块在每次被调用时，内部的电路参数可能会有所不同。因此在对图对判定图求值之前，需要更新符号的数值。另外在交流分析中，每个频率点处的阻抗器件可能会有所不同，因此需要更新阻抗符号的数值。如果当前模块包含子电路的话，还需要先对子电路求值，才能得到相应的导纳矩阵及 VCCS 的符号数值。在上述步骤都完成后，就可以对当前电路求值了。如果上层还有电路模块，那么求得的结果将作为导纳写入导纳矩阵，进而将被上级电路调用。如果本模块已经是顶层模块，则直接将结果输出。

电路求值可以用一个递归过程来实现，每个电路模块的处理为一个循环，下层电路将被优先处理。基于上述讨论，每个递归循环都包含五个步骤，如图 2-7：

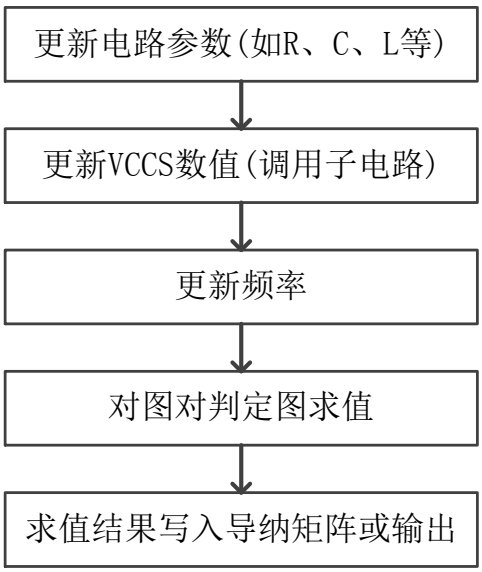


图 2-7 电路模块求值流程
Fig.2-7 Evaluation procedure for a circuit module

下面我们以图 2-8 中的层次化电路模块为例，介绍层次化图对判定图的求值过程。

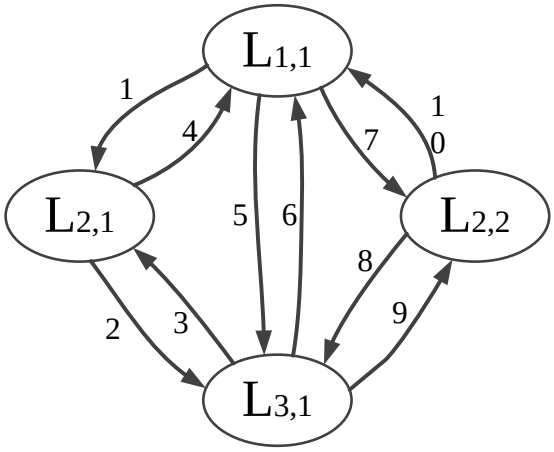


图 2-8 电路模块调用示意图
Fig.2-8 Traverse of circuit modules

在确定当前频率之后，仿真器第一步调用顶层模块 $L_{1,1}$ 进行求值。当更新 $L_{1,1}$ 中的 VCCS 器件时，需要访问子电路模块的导纳矩阵。首先被调用的是模块 $L_{2,1}$，为了得到 $L_{2,1}$ 中 VCCS 的数值，我们进而调用模块 $L_{3,1}$。在模块 $L_{3,1}$ 求值完毕后，它所对应的导纳矩阵被更新，并写回模块 $L_{2,1}$ 中的 VCCS。模块 $L_{2,1}$ 更新完毕后求得的值写回模块 $L_{1,1}$。接下来模块 $L_{1,1}$ 调用模块 $L_{2,2}$ 和 $L_{2,3}$ 来求解相应的 VCCS 器件。

电路模块的递归式访问过程在图 2-8 中描述。最底层的模块 $L_{3,1}$ 总共被访问了三次，每次在求值之前需要更新电路参数。虽然会有额外的更新操作，但是它减少了额外的空间消耗。

2.4 本章小结

本章介绍了基于图对判定图的多层次符号化分析方法的思想，原理以及设计。多端口器件处理规则的引入是实现图对判定图层次化的关键一步。有了多端口器件，导纳矩阵就可以被图对判定图处理。在层次化电路模块的基础上，子电路模块使用若干个 VCCS 器件替代，并分别建立图对判定图。在电路求值时，子电路的求值结果将用来更新导纳矩阵，进而写入位于上层电路模块中的 VCCS 器件中，作为上层电路求值的前提条件。层次化的图对判定图以这种方式组织在一起。并且，图对判定图优越的数据共享功能允许同样的电路功能模块可以共享一个数据空间。这进而减少了空间复杂度。最后，在层次化图对判定图中，每个器件符号只出现一次，因此无对消项存在于最后的解析式中。

第三章 无对消项层次化符号仿真器的实现

3.1 仿真器架构

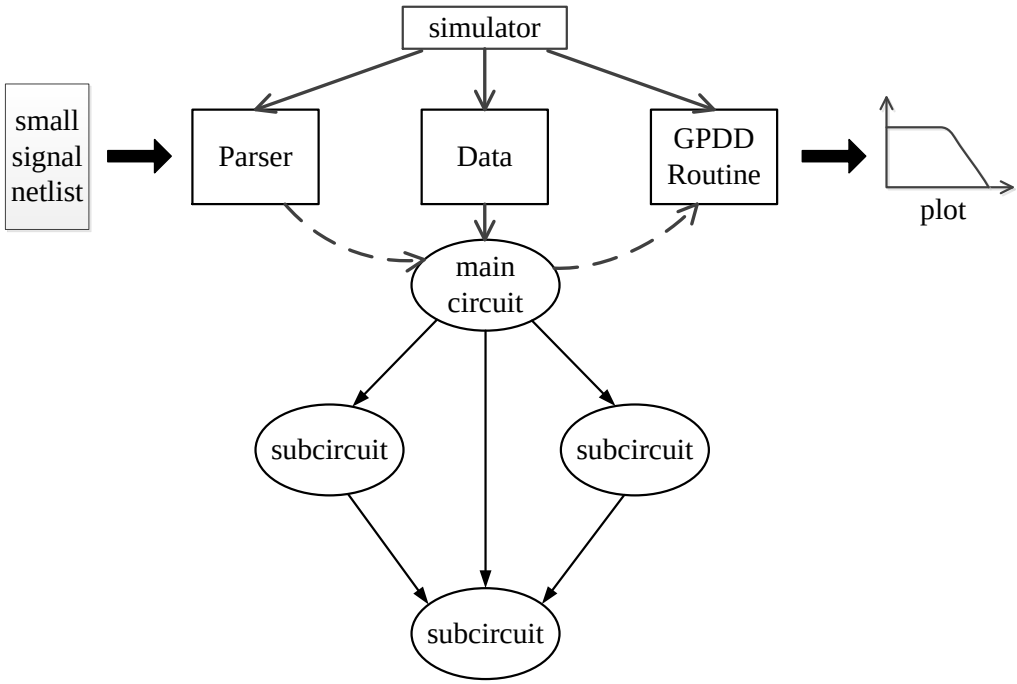


图 3-1 层次化图对判定图仿真器架构
Fig.3-1 Architecture of hierarchical GPDD simulator

本文提出的层次化图对判定图仿真器由 C++语言开发。其架构参见图 3-1。仿真器的输入为小信号网表，输出为频域仿真波形，或者也可以是电路传输函数的解析表达式。仿真器内部主要分为三个功能模块，分别为网表语法分析器（parser）、电路仿真数据（data）及图对判定图函数集。

仿真数据（data）主要包含电路参数、电路模块的层次信息及图对判定图。在不分层的符号化仿真器中，通常电路参数与判定图分开保存。但在我们层次化仿真器中，电路参数和判定图都以电路模块为单位，保存在一起。不同的电路模块所对应的参数及判定图分开保存。我们创建子电路类来保存每个电路模块的参数和判定图。子电路类以层次化的方式组织在一起。层次化的结构是根据电路层次而定的。

仿真器内部的数据结构在语法处理器读取网表时建立，同时电路参数被初始化。网表分析完毕后，仿真器就开始根据网表信息建立图对判定图。最后一步访问 GPDD 并导出电路解析式。

我们将在后面的内容详细讨论仿真器的设计细节。

3.2 层次化网表分析器

层次化仿真器的前端，网表语法分析器由 Flex[23]和 Bison[24]工具生成。Flex 和 Bison 作为 Unix 中的开源工具，有自己定义的高层逻辑语言。在 Flex 和 Bison 源文件中编写的语法树可编译生成 C/C++语言源代码。进而可以编译成二进制可执行文件[20]。

网表语言的语法定义依据经典仿真器 SPICE[21]的语法。因为本符号化仿真器用来执行交流分析，因此所配备的网表分析器只支持用于小信号分析的线性器件。该网表分析器所能接受的器件及其相应的语法规则见表 3-1。

表 3-1 合法器件
Table 3-1 Valid devices

Device name	Grammar	Instance
Resistor	[R/r]xxx n1 n2 val	R1 1 2 100k
Capacitor	[C/c]xxx n1 n2 val	C1 1 2 100u
Inductor	[L/l]xxx n1 n2 val	L1 1 2 100n
VCVS	[E/e]xxx n+ n- nc+ nc- val	E1 1 2 a b 1k
VCCS	[G/g]xxx n+ n- nc+ nc- val	G1 1 2 a b 1k
CCVS	[H/h]xxx n+ n- vnam val	H1 1 2 V1 100
CCCS	[F/f]xxx n+ n- vnam val	F1 1 2 V1 100
Voltage source	[V/v]xxx n+ n- val	V1 1 2 5v
Current source	[I/i]xxx n+ n- val	I1 1 2 1mA
Sub-circuit	[X/x]xxx<nodelist><name><param=val>	X1 1 2 3 sub1 r1=100 c2=2p

以字母 X 开头的器件是子电路的实例化多端口器件。对于表 3-1 中的“X1 1 2 3 sub1 r1=100 c2=2p”语句，它所表示的是一个名为 X1 的子电路实例化多端口器件 sub1。该子电路包含三个端口，分别为 1、2、3。“r1=100 c2=2p”语句用于子电路实例内部参数的初始化。同一个子电路可以多次被实例化，并且每次实例化时的初始参数都允许有所不同。不同子电路实例的参数经网表分析器处理后分开保存，但是它们共享同一个电路结构。

根据[6]中的图对判定图建立规则，一个虚拟的受控源器件将在建立判定图时在输入端引入，替代输入电源。因此输入端的独立电压或电流源将被网表分析器转换为受控源器件。而控制端则为输出端，它在输出指令中被定义，见表 3-2。

表 3-2 合法指令
Table 3-2 Valid commands

Function	Grammar	Instance
AC analysis	.ac dec/oct/lin steps fstart fstop	.ac DEC 10 1 10K
Define output	.plot ac vr/vi/vm/vdb(n)	.plot AC Vdb(5)
Define Sub-circuit	.subckt name <node list> <subckt netlist> .ends	.subckt sub1 1 2 3 R1 1 2 1k C1 2 4 1p .subckt sub2 1 2 3 … .ends X 1 2 3 sub2 L1 2 3 1n .ends

“.subckt”指令用于子电路模块内部结构的定义，与 SPICE 标准语法一致。需要注意的是，网表分析器允许子电路定义的嵌套。如表 3-2 中的子电路定以示例，子电路模块 sub2 嵌套于 sub1 之中。如此嵌套式的语言定义了层次化的电路结构。在网表分析器实现中，嵌套语言的信息提取通过堆栈实现。当进入新的子电路定义模块时，当前电路模块数据结构的指针压栈，待到离开新电路模块时再出栈。

3.3 子电路数据结构

前文中已经提到过，每个电路模块所对应的数据结构分开保存。而层次化电路模块的组织方式通过层次化的网表语言定义。在网表分析器运行过程中，电路模块的数据结构被建立。在接下来的步骤中，每个电路模块将根据网表分析器提取的电路信息，构造自己的图对判定图。子电路数据结构示意图见图 3-2。不同电路通过指针串联在一起。其中每个电路模块的数据主要包含三个部分：

符号表（symbol list）——用于保存电路中的器件类型及连接关系。图对判定图建立在此基础上。在求值阶段，图对判定图需要访问符号表来获得符号数值。

电路参数（circuit parameters）——不同电路实例的参数保存在一起，由网表

分析器写入。在判定图求值阶段，电路参数将用于更新符号表中的符号数值。

图对判定图（GPDD）——仿真器的核心。

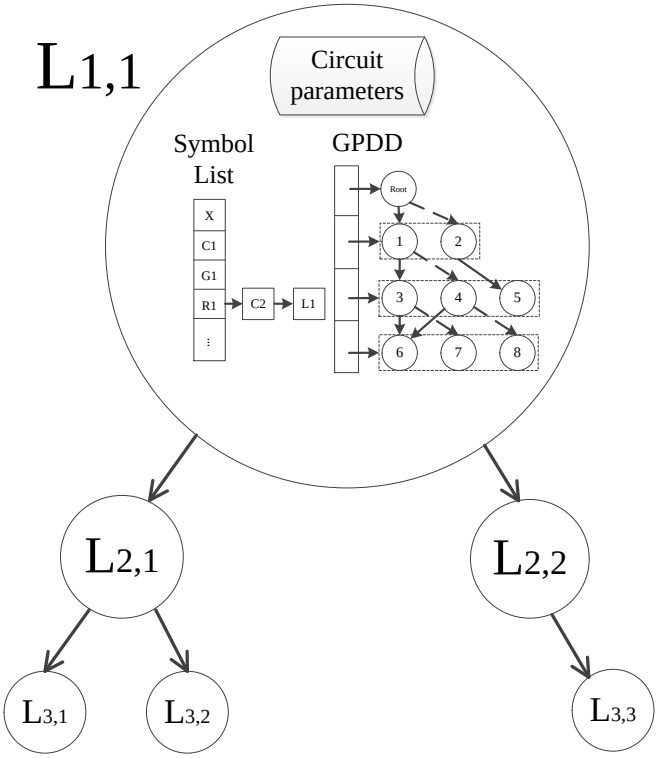


图 3-2 层次化数据结构

Fig.3-2 Hierarchical data structure

在数据结构的维护中，哈希表需要被多次用到。除去 GPDD 自身的构造过程中所需要的哈希表，我们还需要为电路实例参数和子电路的维护构造两个哈希表。首先，同一个子电路中可以存在不同的实例化参数，这些参数保存在一起。为了检索实例参数，我们需要引入一个实例表（instance table），以供在更新符号表时提供快速参数提取。另一方面，子电路模块数据结构的存储地址也保持在哈希表中，我们称之为子电路表（sub-circuit table）。通过按名字查找子电路表，我们可快速访问子电路。

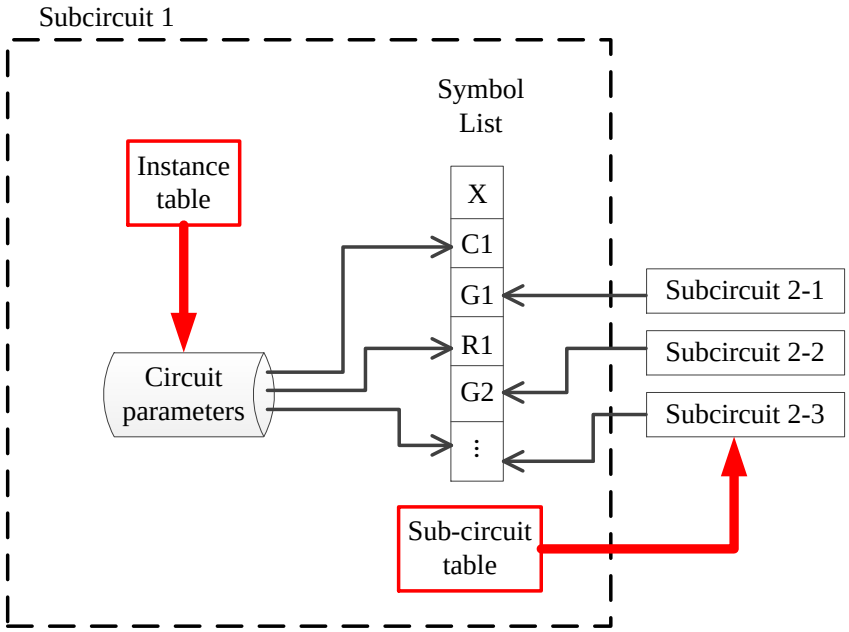


图 3-3 子电路中的哈希表

Fig.3-3 Hash tables in sub-circuit module

如图 3-3 所示，符号表的更新分为两步：首先，通过实例表调用当前实例的电路参数，并写入符号表中。然后，对于符号表中又子电路模块生成的 VCCS 器件，我们需要通过子电路表找到对应的子电路，并得到相应的 VCCS 符号数值。

每个电路模块中的图对判定图构成了我们层次化仿真器的核心部分。根据第一章的介绍，电路模块的符号导纳矩阵由一个多端口图对判定图来表达。每个判定图的根节点都对应着一个导纳矩阵的元素。对于一个 3 端口的子电路模块，我们需要 9 个根节点。多端口判定图根节点的详细构造过程参见[15]。

如图 3-4，多端口图对判定图的构建以广度优先（Breadth Search First，简称 BSF）的原则自上而下进行，即每个根节点衍生出的同一层子树平行展开。这种广度优先的构建法有助于不同根节点的子树共享。

判定图节点在以树的形式组织在一起的同时，我们另外还使用一个向量容器（vector）来按照符号编号保存，即符号编号一致的节点地址保存在一起。这种保存法有助于判定图的简化。

每个节点中都保存着所对应的图对的空间地址。图对作为节点哈希表的键值（key），被用来比较两个节点，以确定它们是否相同。节点的比较只在同一个判定图层次中展开。也就是说我们只对两个具有同样符号的节点进行哈希操作。当同一个符号所对应的节点都被构建后，该符号对应的哈希操作结束。因此我们可

以释放掉该层次的所有图对空间，以节省仿真器的空间消耗。

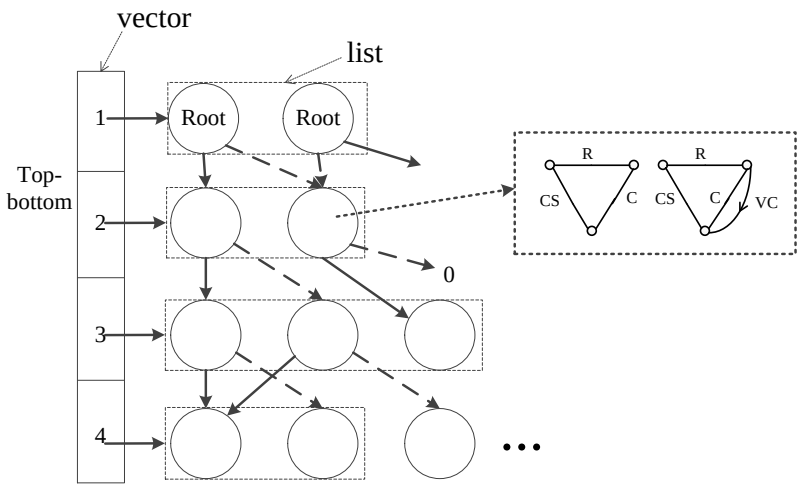


图 3-4 图对判定图的构建
Fig.3-4 Construction of GPDD

3.4 电路模块求值

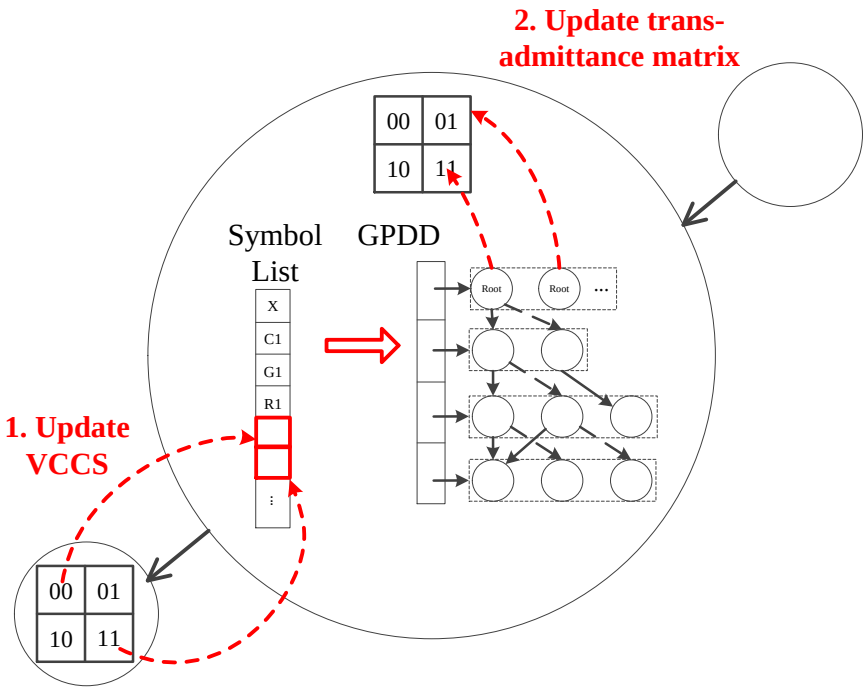


图 3-5 电路模块求值示意图
Fig.3-5 Evaluation of circuit modules

电路求值的迭代过程在 2.3 节中有了详细的介绍。在实现的时候，我们在每个电路模块的数据结构中再引入一个数值矩阵来保存当前导纳矩阵的数值。电路模块之间的数据传递通过导纳矩阵来实现，见图 3-5。

首先，在更新符号表（symbol list）时，需要通过子电路表（sub-circuit table）找到子电路的地址空间。假设子电路模块已经求值完毕，并且在当前频率的符号导纳矩阵数值已经写入导纳矩阵，于是读入该子电路的导纳并更新相应的 VCCS 器件。

在当前电路模块的图对判定图求值完毕后，判定图的根节点数值将填入导纳矩阵。父层电路将会在求值阶段来读取该电路的导纳数值。

3.5 层次化 S 系数展开

按照上文所述的方法构造出来的判定图，隐式地将以 s 为未知数的传输函数 H(s)保存在结构中，其中 $s = j\omega$。比如对于图 1-2 中的二阶 RC 滤波器，若规定符号的顺序为 $R_1 > R_2 > C_1 > C_2$（其中 $a > b$ 表示 a 优先于 b），则得到的展开式为：

$$H(s) = \frac{N(s)}{D(s)} = \frac{R_1^{-1}R_2^{-1}}{R_1^{-1}(R_2^{-1} + C_2 s) + R_2^{-1}(C_1 s + C_2 s) + C_1 C_2 s^2} \qquad (3\text{-}1)$$

注意到在传输函数的分母部分，s 的乘积项以嵌套的方式被表达。虽然对于这个简单的有理式，我们可以很轻松地手动将分母整式展开成（3-2）中的形式，即 s 的级数形式。但是当电路中的符号增多之后，手动展开就变的非常繁琐。当电路变得复杂化，我们就只能借助仿真器来提取 s 的幂级数系数。

$$H(s) = \frac{N(s)}{D(s)} = \frac{a_0 + a_1 s + ... + a_n s^n}{b_0 + b_1 s + ... + b_m s^m} \qquad (3\text{-}2)$$

s 的级数形式是一种很有用的表达方法。因为我们可以通过 s 展开后的系数计算系统的零极点，或者对系统进行模型降阶（model order reduction），进而得到一个简化的传输系统函数。我们也可以通过观测 s 系数与电路参数之间的解析关系，来了解参数对于电路性能的影响。

对于非层次化图对判定图，s 系数展开的基本思想是将 s 参数与其系数分离开。每个 s 幂级项的系数都由一个判定图来表示。一个整式的各个幂级数系数使用同一个多端口的判定图来表达。非层次化的 s 展开基本原理与实现在[8]中有详细的论述，本文就不再赘述。

但是对于层次化的图对判定图，s 系数的展开就会变得复杂。由于导纳矩阵中

的每个元素都呈分式的形式，因此子电路模块生成的 VCCS 器件符号需要用分式来表达。而二分判定图只支持加法和乘法的表达。分号或除号无法在判定图中出现，因此如何将 s 的除法运算在判定图中展开是层次化 s 展开所要解决的关键问题。

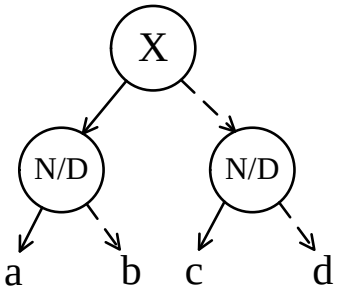


图 3-6 层次化 s 系数展开示意图
Fig.3-6 Illustration of hierarchical s-expansion

一个未展开的图对判定图见图 3-6。假设根节点 X 的左后两个子节点都对应着同一个导纳符号，且这个导纳的解析式为 N/D，我们可以很容易地得到这个判定图的解析式：

$$H(s)=-\frac{a\cdot N/D+b}{c\cdot N/D+d}=-\frac{a\cdot N+b\cdot D}{c\cdot N+d\cdot D} \qquad (3\text{-}3)$$

我们可以清楚地看到，X 左右两颗子树的解析表达式分母部分被抵消掉。于是，最终的解析式是两个整式相除的形式，而且分别是左右子树解析式的分子部分。

可以证明，以判定图的某一个节点出发，可以经过各种路径直到到达某个终点，其间经过的导纳符号相同。因此以该节点为根节点的左右子树所对应的解析式有相同的分母。最终这两个相同的分母在提取解析式时相消。因此我们只需要计算判定图左右两颗子树的解析式分子部分。

为了计算分式的分子部分，我们将通分运算融入图对判定图中。如图 3-7 所示，每个导纳符号都被展开成两个节点，分别对应导纳解析式的分子和分母。原符号的左子树成为分子符号的左子树；原符号的右子树成为分母符号的左子树。这样通分操作就隐式地在判定图中表达。

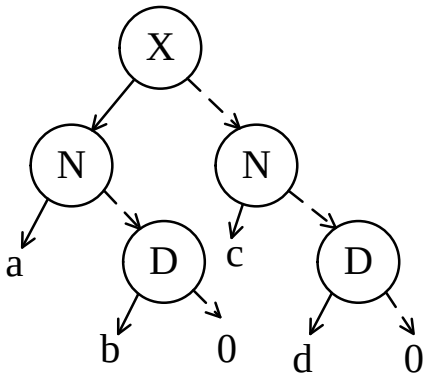


图 3-7 导纳符号的展开
Fig.3-7 Expansion of admittance symbols

下面我们以一个二阶低通滤波器（如图 3-8）为例介绍层次化 s 展开的过程。为简单起见，我们将第二级 RC 梯作为一个一端口的子电路，这个子电路模块将在主电路里生成一个 VCCS 器件。

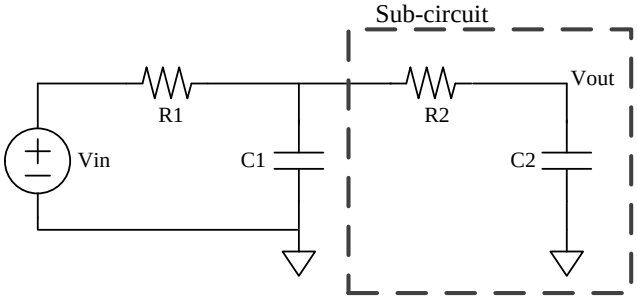


图 3-8 包含子电路的二阶 RC 滤波器
Fig.3-8 the 2nd order RC filter with a subcircuit

层次化的图对判定图其相应的 s 展开判定图如图 3-9 所示。在主电路模块的图对判定图中，符号 N 和 D 分别表示子电路导纳的分子和分母。导纳分子和分母的解析式进而分别由子电路判定图的左右子树表达。

每个土堆判定图的节点都包含一个指针，指向一组多端口的判定图。这个多端口判定图的每个根节点衍生出的子树代表了当前 s 展开式的某项系数。对于子电路模块，s 展开判定图当中的每个节点分别代表了一个电路参数符号（如图 3-9 右下）。而对于父层电路模块，s 展开判定图中的节点符号既可能为电路参数，也可能是表示子电路导纳的分母或分子中的某项系数（参见图 3-9 中的 d0 和 d1）。这个系数进而由子电路模块中的符号组成。

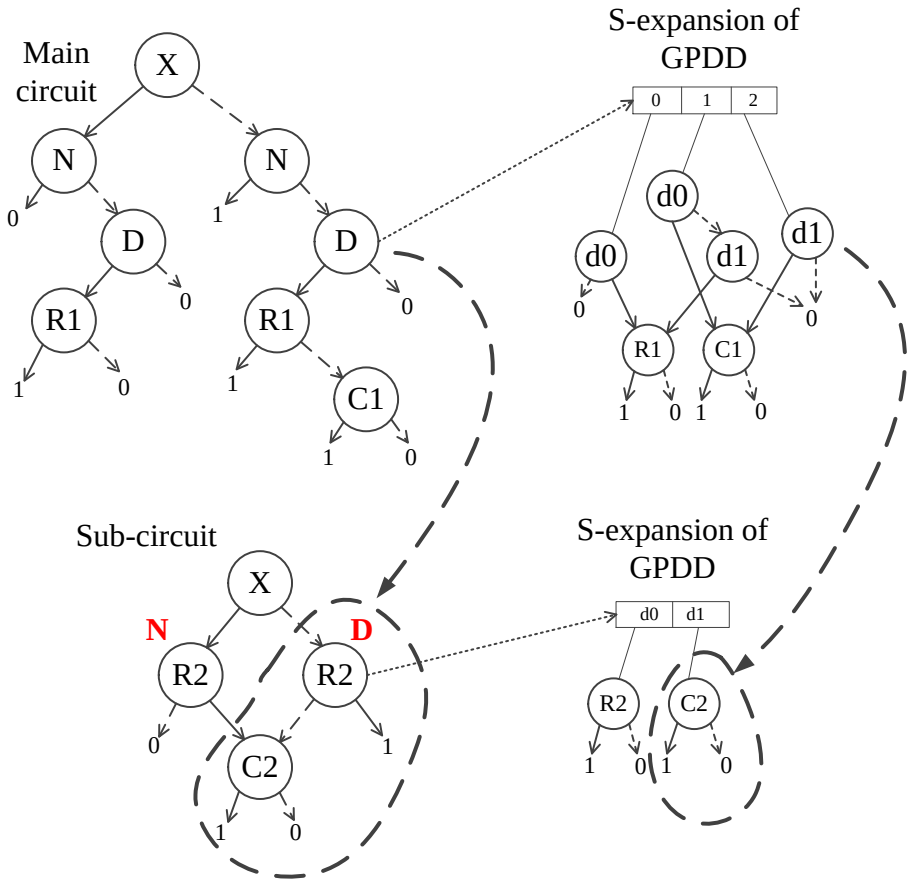


图 3-9 层次化 s 系数展开示意图

Fig.3-9 Illustration of hierarchical s-expansion

3.6 本章小结

本章详细介绍了层次化图对判定图符号仿真器的架构及实现流程。首先层次化仿真器包含三个模块，分别为网表分析器、层次化数据及图对判定图处理函数。电路层次通过层次化的网表语言定义，在此基础上我们建立了层次化的数据结构。每个电路模块的数据分开保存，这其中又主要包含三个部分：符号表，电路参数和图对判定图。此外还有若干哈希表，用来调用电路参数以及子电路的空间地址。不同电路模块通过哈希表连接在一起。在电路求值时，子电路的求值结果通过写入上层电路模块的 VCCS 符号中而传递到上一层。求值过程用一个递归过程实现。最后，我们还介绍了层次化 s 展开的原理及实现。

第四章 实验结果

本文所提出的层次化仿真器由 C++编写，在 Windows 7 环境下进行编译和测试。测试环境如下所示：

- CPU: Intel Core2 Duo T7100 1.80GHz
- Memory: 2GB memory
- Operating System: Win7

本章将给出仿真器效率的测试及不同分层方式所带来的不同结果，并与另一种层次化符号仿真器进行对比，最后给出一些基本结论。

4.1 仿真器工作流程

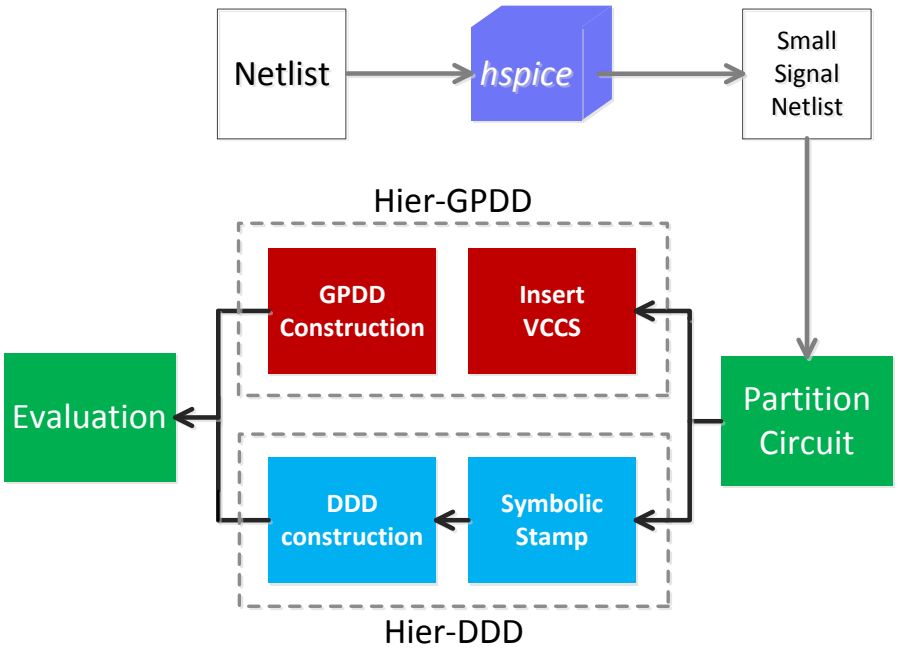


图 4-1 仿真流程图
Fig.4-1 Simulation flow

层次化仿真器的工作流程如图 4-1 所示。首先，含有非线性器件（如 MOSFET 管）的网表经过 hspice[22]的直流分析，转换为小信号电路。MOSFET 或 bipolar 晶体管由小信号等效电路代替。层次化符号仿真器使用小信号电路生成电路的传输函数解析式，并求值、输出仿真波形。

为了对本文的层次化图对判定图仿真器提供一个效率的参照，我们还实现了另一种基于行列式判定图的层次化仿真器。但是与[12-13]中提出的基于舒尔分解

的层次化仿真器不同，我们的对照组仿真器的层次化原理基于符号导纳矩阵。在用作对组照的层次化行列式判定图仿真器中，我们不需要构建图对，而是直接将子电路模块的导纳矩阵填入上层电路的 MNA 矩阵中，并使用行列式判定图对 MNA 矩阵求值。层次化行列式判定图的分析过程完全基于矩阵。

注意到两种基于二分判定图的仿真器对符号排序（symbol ordering）以及电路划分（circuit partitioning）有不同的要求。在这里，图对判定图采用随机的符号排序，而行列式判定图采用最小度展开原则排序。电路的划分主要根据层次化图对判定图的特点。我们已经在第二章做了相关的讨论。我们将在本章大致比较两种方法的效率差别。

4.2 交流分析测试

在本章节我们将使用三个运算放大器电路上测试我们的层次化图对判定图仿真器。这些测试电路分别为：

- 电路一：Bipolar 运算放大器 uA725
- 电路二：包含 24 个晶体管的 MOS 管两级全摆幅 cascode 放大器，在后文中简称运算放大器 1
- 电路三：包含 44 个晶体管的 MOS 管运算放大器，下文简称运算放大器 2

上述测试电路在前人的层次化仿真器研究中也有用到，因此具有一定代表性。而且，这些测试电路也是目前在已发表的论文中，符号化仿真器所能处理的最大规模的电路。其中最大的测试电路包含 44 个晶体管，在将每个 MOSFET 晶体管替换为等效小信号模型后，仅 MOSFET 晶体管就需要使用总共 528 个线性器件替代。由此可见这些电路规模足以用来验证层次化仿真器的仿真性能。

4.2.1 Bipolar 电路测试

我们使用的第一个测试电路为 uA725 放大器。uA725 被最早用于行列式判定图层次化研究中[12-13]。基于舒尔分解的层次化行列式判定图仿真器可以在较短时间内完成 uA725 的求值。非层次化的行列式判定图在使用[4]中简单高效的哈希机制后也可以在一定时间内完成 uA725。而非层次化的图对判定图很难完成 A725 的仿真。

uA725 电路中包含 25 个双极晶体管。在执行符号分析前，每个晶体管被替换为图 4-2 中的等效电路。

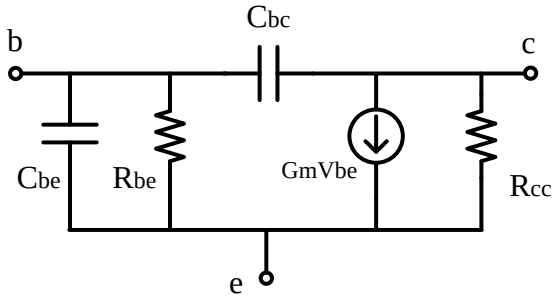


图 4-2 双极晶体管等效电路

Fig.4-2 Equivalent circuit of a bipolar transistor

如图 4-3 所示，我们首先尝试将 uA725 分成两个层次。注意到 uA725 的内部可以分为三个放大级，因此分别将 uA725 的前端，后端和中间部分各划分为一个子电路模块。这样 uA725 就成为一个含有两个层次的电路：底层电路模块有三个（$L_{2,1}, L_{2,2}, L_{2,3}$），分别是划分出来的三个子电路。顶层电路 $L_{1,1}$ 中包含三个多端口器件，分别对应这三个子电路。

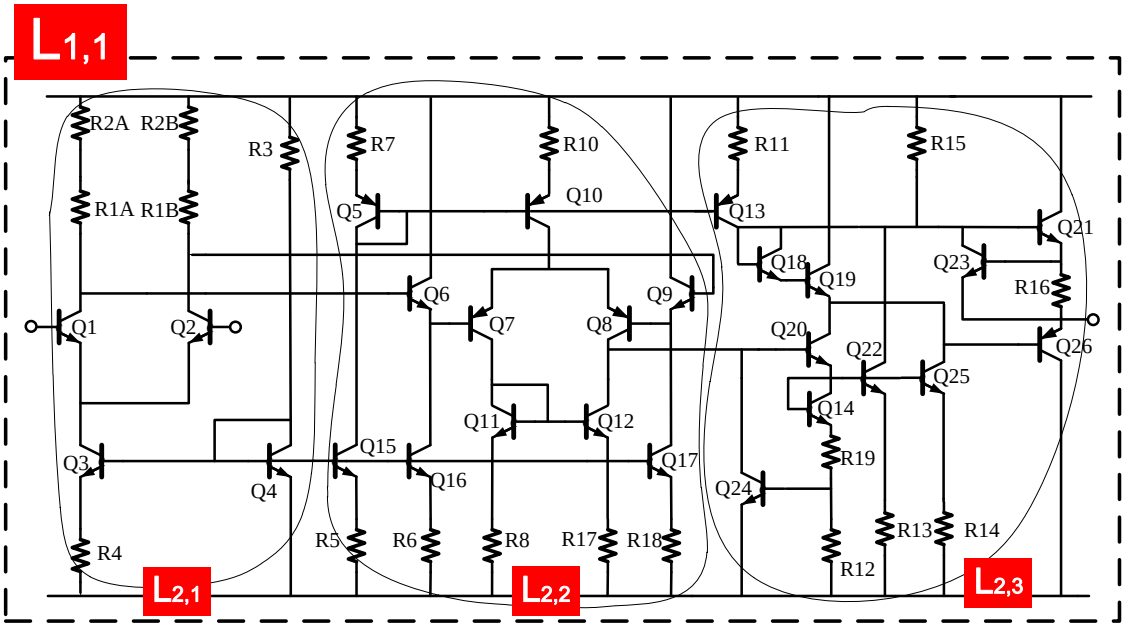


图 4-3 两层 uA725 电路图

Fig.4-3 Schematic of uA725 with two levels

详细的电路层次划分信息见表 4-1。层次化电路对两个层次的 uA725 电路的分析性能见表 4-2。其中，#Edges 表示电路模块中支路的个数，#Nodes 表示电路节点的个数，#Ports 表示电路模块的端口个数，|GPDD|表示电路模块所对应的图对判定图中的节点个数，GPDD Time 指图对判定图的构造时间，|DDD|与 DDD Time 指行列式判定图的规模及其构造时间。由于判定图的求值时间与图的规模基

本成正比，我们不再另外讨论求值时间。

根据实验结果，两种层次化仿真器都可以在几秒钟之内完成图对判定图的构建。对于本文的层次化图对判定图，uA725 只需要约 5 秒钟就可以构造好判定图。这是非层次化的符号仿真器所不能做到的。另外还可以看到，大部分的判定图构造时间都花在了 $L_{2,2}$ 和 $L_{2,3}$ 这两个子电路模块中。为了进一步减少判定图构建的时间消耗，我们下一步继续将这两个模块划分成更小的子模块。

表 4-1 两层 uA725 的层次划分表
Table 4-1 Partition list for uA725 with 2 levels

Module Index	Components
L1,1	L2,1 L2,2 L2,3
L2,1	Q1 Q2 Q3 Q4 R1 R2 R3 R4
L2,2	Q5 Q10 R7 R10 Q6 Q16 R6 Q7 Q8 Q11 Q12 R8 R17 Q9 Q17 R18 Q15 R5
L2,3	Q13 Q18 Q19 R11 Q21 Q23 Q26 R15 R16 Q14 Q20 Q22 Q24 Q25 R12 R13 R14 R19

表 4-2 两层 uA725 的仿真性能对比
Table 4-2 Performance comparison for uA725 with 2 levels

Module	#Edges	#Nodes	#Ports	|GPDD|	GPDD Time(s)	|DDD|	DDD Time (s)
L1,1	34	7	--	773	0.016	130	0.013
L2,1	26	8	4	548	0.046	341	0.063
L2,2	62	16	4	135,785	3.015	11870	1.999
L2,3	62	13	3	91,682	2.067	12022	0.687
Total	--	--	--	228,788	5.165	24363	2.781

如图 4-4 所示，我们根据电路的结构，将 $L_{2,2}$ 和 $L_{2,3}$ 中连接较为紧密的部分细划分为小的模块。这样我们得到了一个包含三个层次的电路：第一层依然包含三个放大级等效的多端口器件；第二级为任意一个放大级，它其中可能包含更小的多端口器件；第三层为各放大级中的子电路模块。划分为三个层次的 uA725 电路图见图 4-4。详细划分信息见表 4-3。

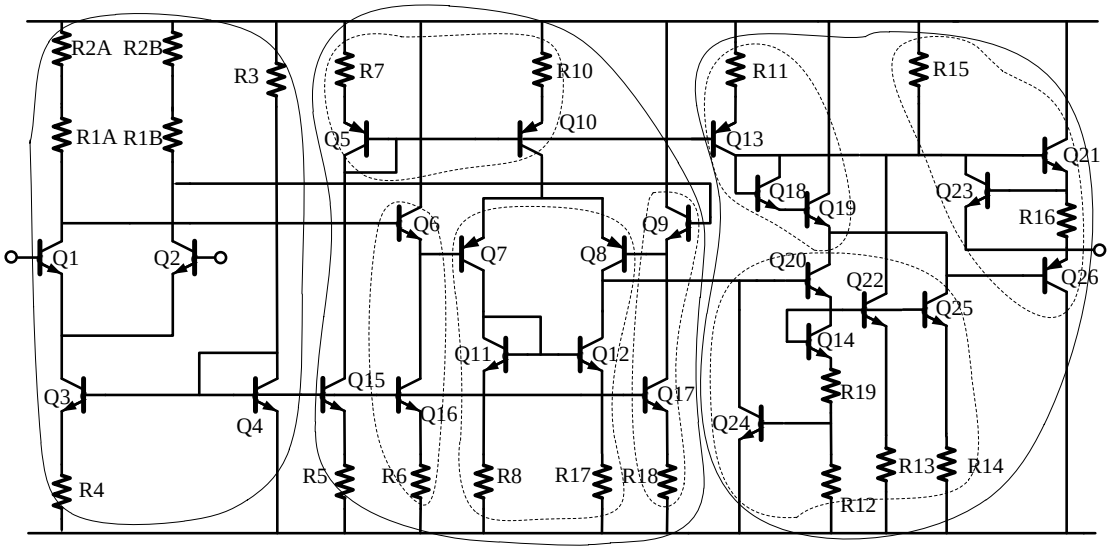


图 4-4 三层 uA725 电路图
Fig.4-4 Schematic of uA725 with three levels

表 4-3 三层 uA725 的层次划分表
Table 4-3 Partition list for uA725 with 3 levels

Module Index	Components
L1,1	L2,1 L2,2 L2,3
L2,1	Q1 Q2 Q3 Q4 R1 R2 R3 R4
L2,2	L3,1 L3,2 L3,3 L3.4 Q15 R5
L2,3	L3,5 L3,6 L3,7
L3,1	Q5 Q10 R7 R10
L3,2	Q6 Q16 R6
L3,3	Q7 Q8 Q11 Q12 R8 R17
L3,4	Q9 Q17 R18
L3,5	Q13 Q18 Q19 R11
L3,6	Q21 Q23 Q26 R15 R16
L3,7	Q14 Q20 Q22 Q24 Q25 R12 R13 R14 R19

层次化仿真器对三层 uA725 电路的分析性能见表 4-4 所示。通过进一步的划分，几乎所有底层电路模块的图对判定图构造时间都限制在了 0.01 秒之内。判定图的总构造时间甚至不到 1 秒钟。这种构建效率是在之前的研究成果中没有出现过的。

注意到每个电路模块的端口个数都限制在 4 个以内。这样做是为了避免生成过多的 VCCS 器件，导致电路规模反而增大。通过把每个电路模块的规模进行限制，判定图规模的指数式增长就被避免。于是对所有模块构建判定图的总时间消耗降低。

表 4-4 三层 uA725 的仿真性能对比
Table 4-4 Performance comparison for uA725 with 3 levels

Module Index	#Edges	#Nodes	#Ports	|GPDD|	GPDD Time (s)	|DDD|	DDD Time (s)
L1,1	34	7	--	773	0.016	130	0.014
L2,1	26	8	4	548	0.046	341	0.062
L2,2	44	16	4	2987	0.099	1667	0.238
L2,3	27	13	3	700	0.066	193	0.027
L3,1	12	4	2	99	0.038	52	0.007
L3,2	11	4	3	103	0.043	79	0.012
L3,3	22	7	4	476	0.045	366	0.062
L3,4	11	4	3	126	0.038	79	0.012
L3,5	16	5	3	133	0.05	99	0.017
L3,6	17	4	3	131	0.05	79	0.014
L3,7	29	8	3	4822	0.13	494	0.074
Total	--	--	--	10898	0.67	3579	0.586

当然，符号仿真器的精度也是值得考量的一个指标。我们特意使用 hspice 中的 AC 分析结果与层次化图对判定图的求值结果做了一个比较，见图 4-5。符号仿真器的求值输出波形（虚线）与 hspice 的仿真波形（实现）完全重合。我们的符号仿真器精度得到验证。

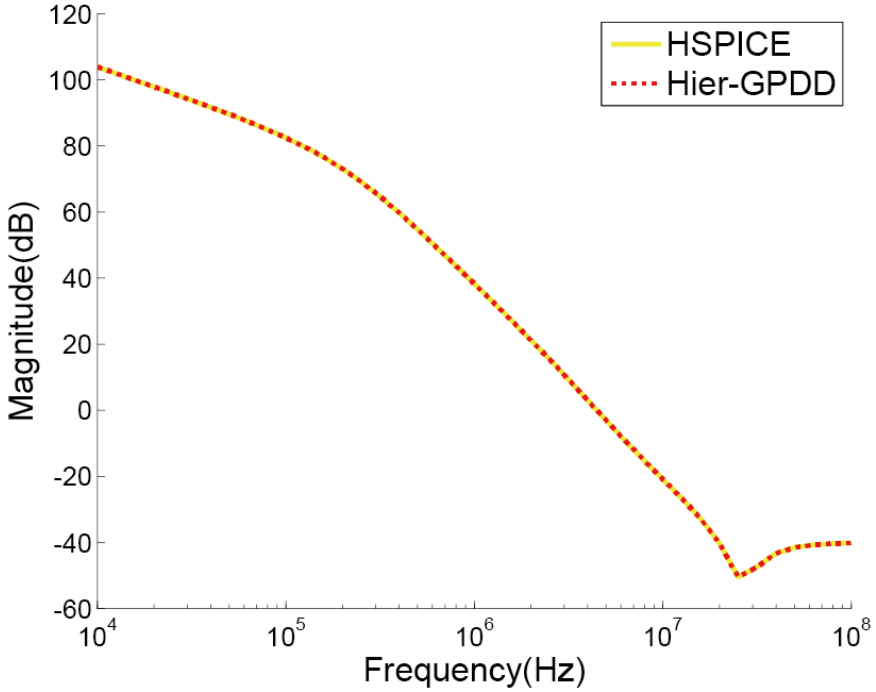


图 4-5 uA725 电路仿真输出
Fig.4-5 Simulation result of uA725

4.2.2 MOSFET 电路测试

下面我们将再使用两个 MOS 管构成的放大器电路进行测试。第一个放大器（见图 4-6）包含 24 个 MOSFET 晶体管。每个晶体管的等效电路见图 1-16。

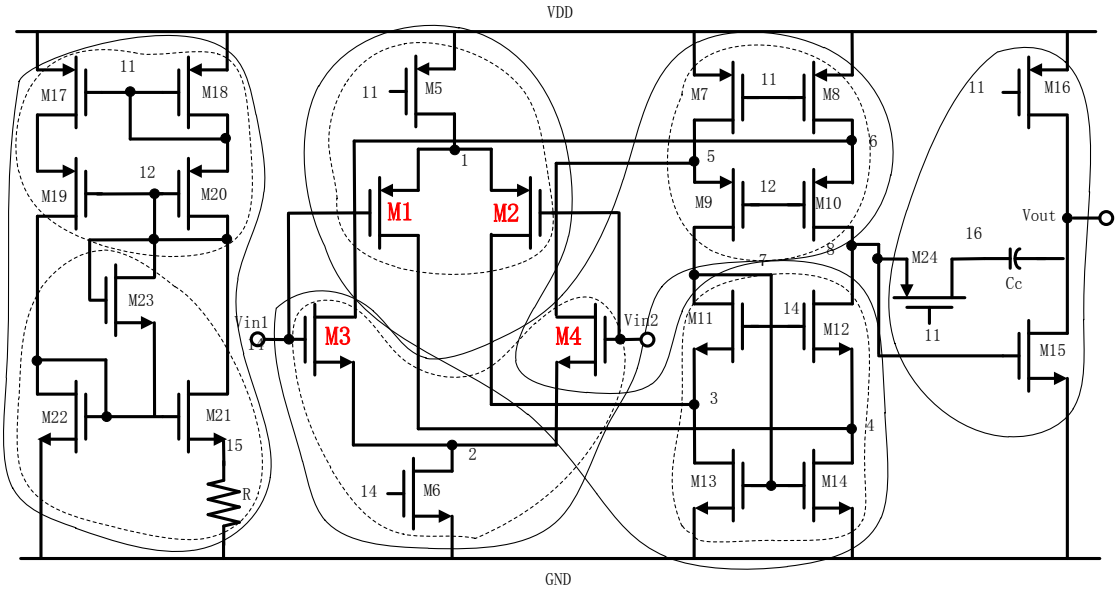


图 4-6 运算放大器 1（含 24 个晶体管）电路图
Fig.4-6 Schematic of Op-Amp I with 24 transistors

表 4-5 运算放大器 1 的层次划分表
Table 4-5 Partition list for OPAMP I

Module Index	Components
L1,1	L2,1 L2,2 L2,3 L2,4
L2,1	L3,1 L3,2
L2,2	L3,3 L3,6
L2,3	L3,4 L3,5
L2,4	M15 M16 M24 Cc
L3,1	M17 M18 M19 M20
L3,2	M21 M22 M23 R
L3,3	M1 M2 M5
L3,4	M3 M4 M6
L3,5	M7 M8 M9 M10
L3,6	M11 M12 M13 M14

根据电路的功能，此放大器可大致分为三个部分：电源电路，第一级差分转单端放大电路和第二级单端放大电路。注意到第一级放大电路实际包含两个互补的差分对，M1、M2 和 M3、M4。我们可以进而将这两个差分对拆开。这样我们就得到了四个模块。这四个模块构成了电路的中间层次。在这四个模块的内部，

我们进而划分出若干个子电路。最后得到一个三层的电路结构。

对三层放大器的仿真性能分析见表 4-6。同上一个电路，在细致划分后，几乎每个电路模块的判定图构造时间被限制在 0.1 秒以内。层次化图对判定图的构建时间（0.793）甚至要小于层次化行列式判定图的构造时间（2.042）。

表 4-6 运算放大器 1 的仿真性能对比
Table 4-6 Performance comparison for OPAMP I

Module Index	#Edges	#Nodes	#Ports	|GPDD|	GPDD Time (s)	|DDD|	DDD Time (s)
L1,1	63	6	--	5111	0.117	142	0.014
L2,1	13	3	2	68	0.039	41	0.005
L2,2	32	6	5	26	0.040	474	0.098
L2,3	32	6	5	26	0.044	474	0.099
L2,4	28	7	3	780	0.054	660	0.107
L3,1	36	12	3	1846	0.074	1006	0.165
L3,2	28	8	2	895	0.054	357	0.052
L3,3	27	11	4	1343	0.060	1437	0.239
L3,4	27	11	4	1343	0.061	1437	0.237
L3,5	36	14	4	3563	0.108	3341	0.602
L3,6	36	13	4	2487	0.092	2137	0.372
Total	--	--	--	17488	0.793	11506	2.042

我们使用的最后一个测试电路是包含 44 个 MOS 管的运算放大器电路，如图 4-7 所示。此电路曾在[14-15]中被使用。

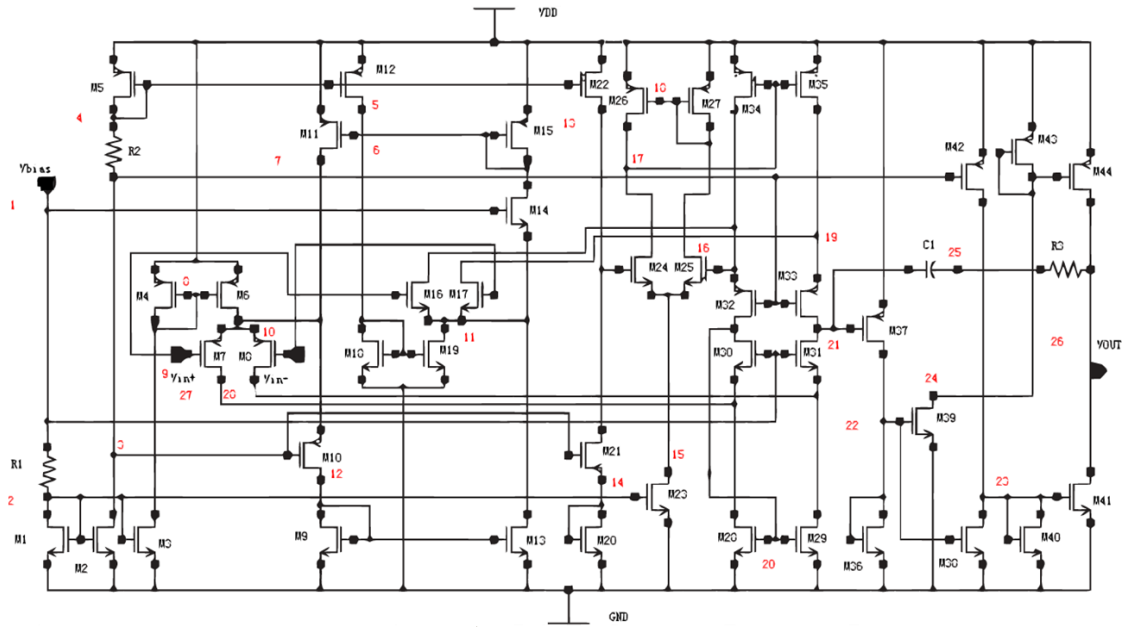


图 4-7 运算放大器 2（含 44 个晶体管）电路图
Fig.4-7 Schematic of Op-Amp II with 44 transistors

由于该电路规模很大，内部器件连接关系也很复杂，最终我们将其划分为 5 个层次（见表 4-7）。

表 4-7 运算放大器 2 的层次划分表
Table 4-7 Partition list for OPAMP II

Module Index	Components
L1,1	L2,1 L2,2
L2,1	L3,1 L3,2
L2,2	R3 C1 L3,3 L3,4 M36 M37
L3,1	L4,1 M32 M33
L3,2	L4,2 L4,3 M30 M31
L3,3	M39 M43 M44
L3,4	M38 M40 M41 M42
L4,1	L5,1 L5,2 L5,3 L5,4 L5,5
L4,2	M28 M29
L4,3	L5,6 L5,7 L5,8
L5,1	R2 M5 M12 M22
L5,2	M16 M17 M18 M19
L5,3	R1 M1 M2 M3 M23
L5,4	M20 M21
L5,5	M24 M25 M26 M27 M34 M35
L5,6	M11 M14 M15
L5,7	M4 M6 M7 M8
L5,8	M9 M10 M13

表 4-8 层次化符号仿真器性能总结
Table 4-8 Performance summary for hierarchical symbolic simulators

Circuit	#Edges	#Nodes	|GPDD|	GPDD Time (s)	|DDD|	DDD Time (s)
uA751	166	31	10432	0.682	3579	0.586
Op-Amp I	218	66	17488	0.793	11506	2.042
Op-Amp II	399	114	197274	6.771	62794	10.359

层次化仿真器对于各个测试电路的仿真性能总结见表 4-8。每个电路的判定图构建时间消耗分别可以控制在几秒钟之内。于是层次化的有效性得到了证明。此外，我们的层次化图对判定图仿真器取得了与基于行列式的层次化判定图仿真器近似的效率。

最后，我们的层次化仿真器对两个 MOS 运算放大器的输出波形与 hspice 输

出曲线比较见图 4-8 和图 4-9。两者的仿真结果高度一致。

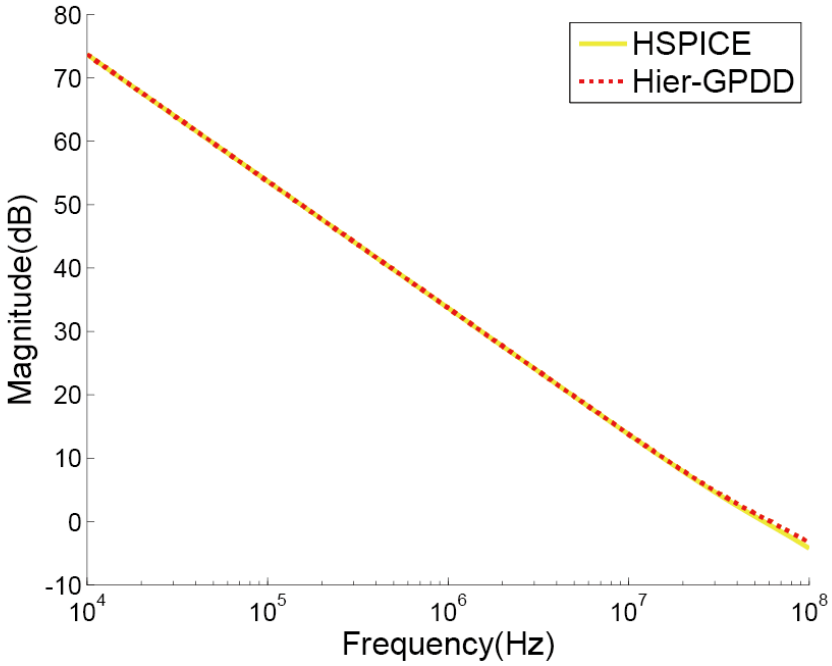


图 4-8 运算放大器 1 仿真输出

Fig.4-8 Simulation result of Op-Amp I

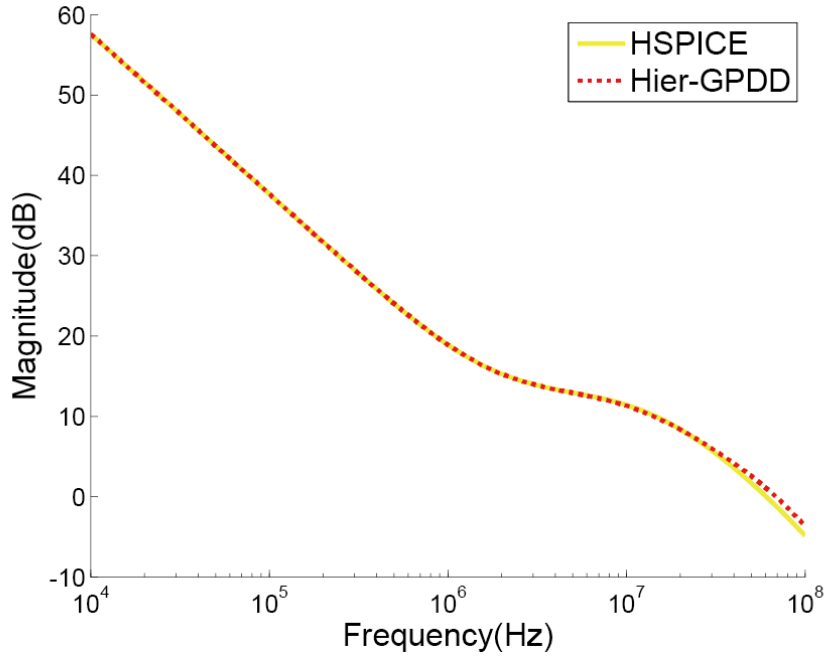


图 4-9 运算放大器 2 仿真输出

Fig.4-9 Simulation result of Op-Amp II

4.3 S 系数展开测试

在本章的末尾，我们对层次化图对判定图的 s 展开进行验证。

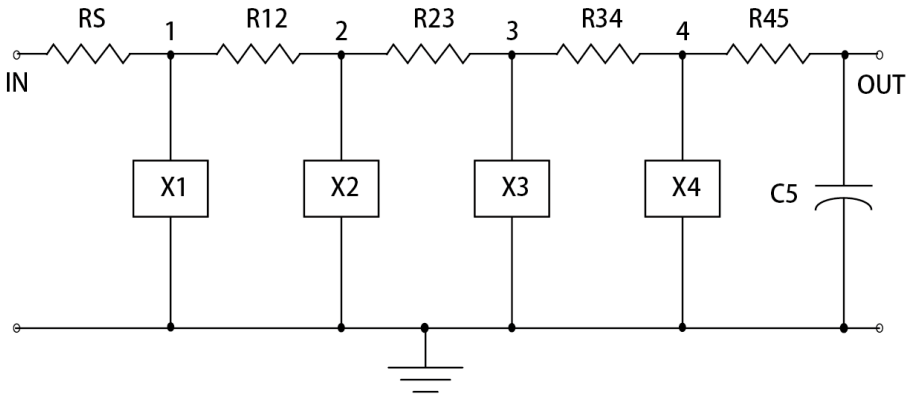


图 4-10 九阶低通滤波器
Fig.4-10 Ninth Order Low-Pass Filter

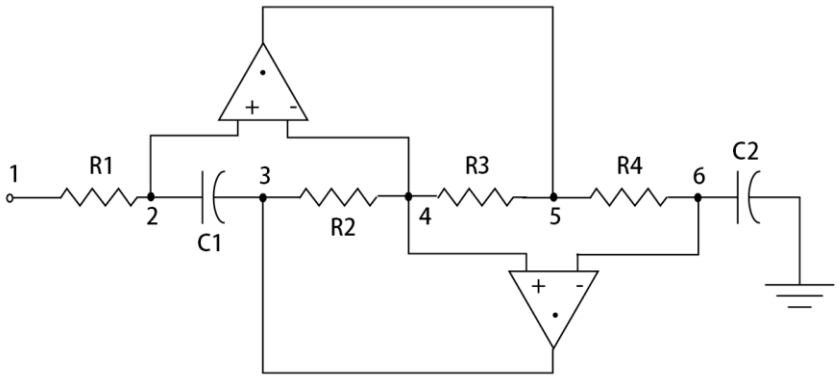


图 4-11 FDNR 子电路
Fig.4-11 The FDNR Subcircuit

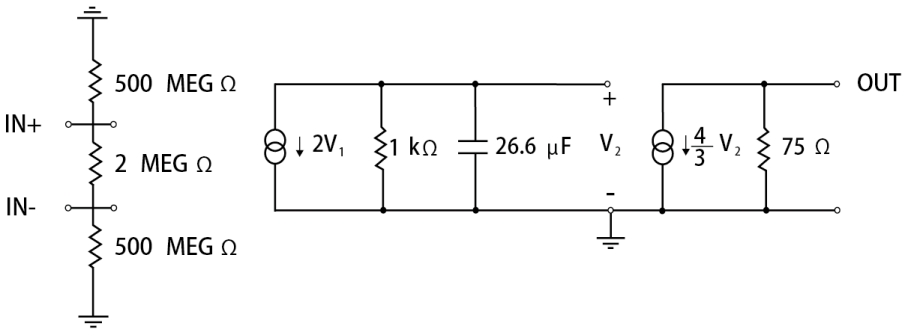


图 4-12 运算放大器 741C 线性模型
Fig.4-12 Linear Model of the 741C Op-Amp

S 展开的测试电路是一个 9 阶的有源低通滤波器，取自[23]。它包含三个层次，

从上到下依次见图 4-10,4-11 和 4-12。在使用第三章描述的方法进行层次化 s 展开后，我们得到的 s 展开系数见表 4-9。

表 4-9 九阶低通滤波器传输函数 s 展开系数
Table 4-9 Coefficients of s-expansion of 9th order low-pass filter

Order	N(s)	D(s)
0	1.33116e-057	1.33129e-057
1	1.42427e-063	2.40979e-061
2	6.50579e-066	2.81558e-065
3	6.96996e-072	2.35078e-069
4	1.1218e-074	1.47871e-073
5	1.20309e-080	7.69006e-078
6	7.85324e-084	2.93819e-082
7	8.4262e-090	1.00574e-086
8	1.76899e-093	2.01977e-091
9	1.89538e-099	4.31035e-096
10	1.01386e-105	4.52842e-102
11	3.38657e-112	2.40967e-108
12	7.68341e-119	8.04029e-115
13	1.20527e-125	1.82524e-121
14	1.28409e-132	2.86745e-128
15	0.54069e-140	3.06094e-135
16	2.8323e-147	2.04049e-142
17	--	6.78208e-150

由于电路中的导纳，电感和电容往往很小，这就导致最后 s 展开得到的系数也变得非常小。对于大规模的电路，s 的系数小到甚至超出双精度浮点数所能表示的范围。因此在实现的时候我们需要使用特别的技巧，比如系数的指数与小数部分分开保存。

为了验证 s 展开式的正确性，我们将其导入至 Matlab 中，画出传输函数在给定的 s 范围内的曲线。进而我们使用非 s 展开的图对判定图的仿真曲线做对比，见图 4-13。

通过对比可以证明，s 展开前后的输出函数保持一致。层次化 s 展开可以正常工作。

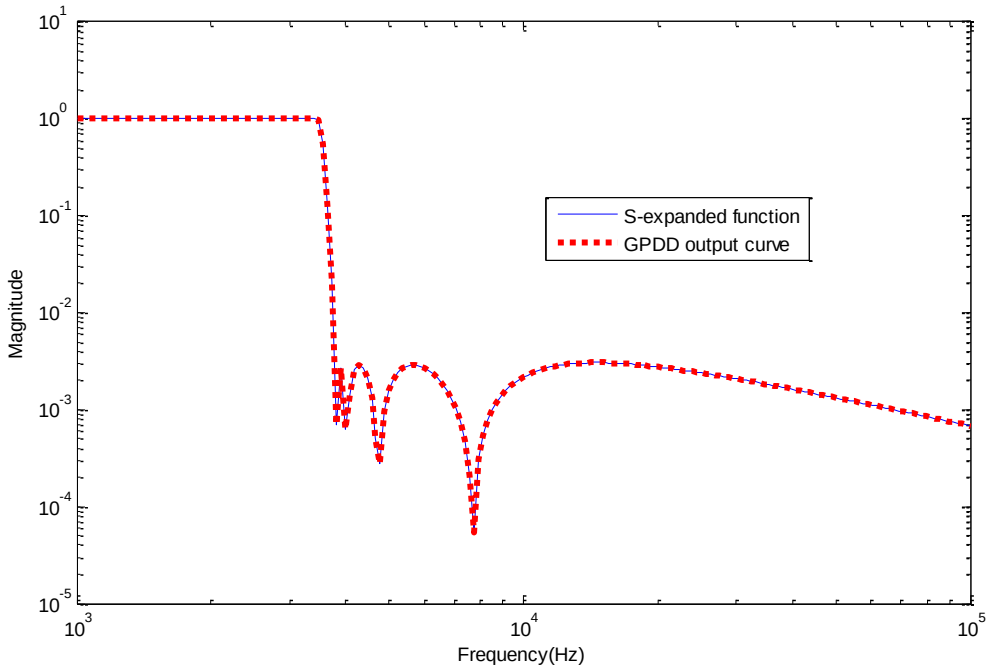


图 4-13 层次化 s 展开验证
Fig.4-13 Verification of hierarchical s-expansion

4.4 本章小结

在本章中，我们测试了层次化图对判定图仿真器的性能，并验证了仿真结果的正确性。作为对比，我们还实现了一个基于符号导纳矩阵的层次化行列式判定图。我们选取了三种模拟运算放大电路作为测试电路，包括 bipolar 电路 uA725、包含 24 个和 44 个管子的 MOS 管放大器。通过尝试不同的层次划分，层次化的有效性得到证明。在合适的电路划分方法下，两种仿真器都可以在几秒钟之内完成大规模电路的判定图构建。另外图对判定图还具备无对消项的优势，这使得它成为符号化仿真器的探索过程中的一个重要进步。在本章的最后，层次化的 s 展开也得到了验证。层次化 s 展开的实现将进一步促进层次化图对判定图在未来的应用。

第五章 结束语

5.1 主要工作与创新点

基于二分判定图的符号化仿真器一直受到指数式规模增长的影响，因而很难在大规模电路上应用。在近几年来的研究成果中，层次化仿真的概念被提出。层次化的符号仿真器都分别取得了不错的效果。

基于图对判定图的符号仿真器具备了行列式判定图仿真器没有的一些特性。比如器件参数的快速更新，无对消项等。但是，由于之前没有层次化的图对判被开发出来，图对判定图的应用只能被限制在小电路中。虽然在[14,15]中，研究者引入了基于图对判定图的混合式层次化仿真器，但是这种仿真器损失了无对消项的特性。本文的主要工作是开发出一种完全基于图的层次化符号仿真器。这个仿真器在保证了图对判定图的无对消项特性的同时，还具备了优秀的仿真效率。

我们提出的层次化图对判定图仿真器是基于符号化导纳矩阵实现的。在导纳矩阵的基础上，要实现层次化的关键问题是如何使用图对判定图来处理导纳矩阵。而解决这个问题的方法就是在原有的图对构建规则中引入了多端口器件的处理。于是，我们将子电路的符号导纳矩阵映射成一组 VCCS 器件，并替换掉原子电路。这样做的结果是一个大的电路被分割成若干个模块，每个被分割掉的模块使用一系列 VCCS 器件来替代。既然每个电路的规模可以被限制在有限范围内，判定图的指数式增长就可以避免了。

为了实现层次化仿真器，本文还设计了层次化的数据结构用来维护来自不同电路模块的数据。每个模块的求值结果通过写入导纳矩阵进而传递到上层电路模块的 VCCS 中，最终用在上次电路的求值过程中。

此外，本文还实现了基于图对判定图的层次化的 s 展开。层次化 s 展开的难点在于如何将包含分母的复合符号在判定图中展开。而我们的做法是将导纳符号展开成通分的形式，并且只展开通分后的分子多项式。这样就有效避免了除法运算。

通过实验数据的采集，我们证明本文提出的层次化图对多项式具备与层次化行列式相当的效率。它可以在几秒钟之内完成大规模电路的分析。其中最大规模的测试电路包含 44 个 MOS 管。并且，我们的符号仿真器的输出波形与 hspice 一致。其仿真的准确性也得到了验证。

5.2 后续研究工作

后续的研究工作应当解决两个方向的问题，即进一步提高层次化仿真器的效率以及寻找更多的应用来推广符号化仿真。

首先，为了进一步提高仿真器的性能，我们可以探索适合图对判定图的符号排序方式。对于层次化的仿真器，电路的划分也是一个值得探讨的话题。合适的划分可以很大改善仿真效率。而在第二章中也提到过，图对判定图的电路方式有别于行列式判定图。由于需要考虑额外的 VCCS 器件，子电路的端口数通常要控制在很小的数量内。先前常用的划分电路方法（例如以一个 MOS 管的等效电路为一个独立电路模块）可能不再适用了。与此同时，电路划分仍然需要考虑到如何利用图对判定图可以共享同构电路的特点。

其次，目前层次化符号仿真器都已经可以分析规模可观的电路。寻找适合符号仿真器的应用是另一个重要方向，比如零极点的符号式计算。

此外，层次化 s 展开的实现需要进一步改进。由于电路中的导纳和电容等参数绝对值很小，s 展开后的系数常常太小以至于实现时发生浮点数下溢。对于规模较大的电路，这种情况尤为明显。潜在的改进方法包括使用缩放（scaling）技术，通过一个公共系数来调节电路参数的数量级，进而调整 s 系数的范围。

参 考 文 献

[1] S. B. Akers. “Binary Decision Diagrams,” IEEE Trans. on Computers, C-27(6):509–516, June 1978.

[2] R. E. Bryant, “Graph-based algorithms for Boolean function manipulation,” IEEE Trans. on Computers, C-35(8), pp. 677-691, August, 1986.

[3] C.-J. R. Shi and X.-D. Tan, “Canonical symbolic analysis of large analog circuits with determinant decision diagram,” IEEE Trans. CAD, vol. 19, no.1, Jan 2000.

[4] G. Shi, “A simple implementation of determinant decision diagram,” in IEEE International Conference on Computer-Aided Design (ICCAD), 2010, pp. 70-76.

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

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

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

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

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

[10] Z. Hao, X.-D. Tan, R. Shen and G. Shi, “Performance bound analysis of analog circuits considering process variations,” in ACM/EDAC/IEEE Design Automation Conference (DAC), June 2011.

[11] R. A. Horn and C. R. Johnson, Matrix Analysis, Sections 2.3 and further, Cambridge University Press, 1985.

[12] X.-D. Tan and C.-J. Shi, “Hierarchical symbolic analysis of large analog circuits via determinant decision diagrams,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 4, pp. 401–412, April 2000.

[13] X.-D. Tan, W. Guo and Z. Qi, "Hierarchical approach to exact symbolic analysis of large analog circuits," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.24, no.8, pp. 1241- 1250, Aug. 2005.

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

[15] 徐辉,“用于模拟电路模块设计的多层次符号化分析新算法研究与软件实现,”上海交通大学微电子学院硕士论文，2012 年 1 月。

[16] 李小鹏，“用于模拟电路模块设计的多层次符号化分析新算法研究与软件实现，”上海交通大学微电子学院硕士论文，2012 年 1 月。

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

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

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

[20] J. Levine, “flex & bison,” O'Reilly Media. pp. 304. ISBN 978-0-596-15597-1.

[21] L. W. Nagel, “SPICE2: A computer program to simulate semiconductor circuits,” Ph.D. Dissertation, University of California, Berkeley, CA, May 1975.

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

[23] Star-Hspice Manual, Avant! Corporation, Jan. 1999

致　谢

时光荏苒，两年半的硕士学习已然走到尽头。而我在上海交通大学的六年时间也已经悄然流逝。在母校的日子里，有奋发的日夜，也有虚度的光阴。我还记得文科图书馆日语里的大桌子，那是我曾经每周三写高数作业的地方。我也忘不了东区宿舍的可乐亭，那是我们结伴顶着寒风买夜宵的去处。这是我的人生中一段漫长、曲折而又充满希望的日子。有太多值得纪念的片段，也有很多值得感谢的人。

首先，我要感谢我的导师，施国勇教授。第一次接触到我的导师是在本科的自动化设计引论课上。很多本科同学告诉我这门课不易，于是我就抱着挑战一下的心态选了这门课。后来，我不但完成了一直引以为傲的课程作品，并且第一次对学习产生了考试分数以外的热情。这个经历使我最终决定投入施老师门下。其中的过程曲折，而我很庆幸自己在关键时刻做了正确的选择。在我的硕士学习期间，施老师给予了很多耐心的指导。他严谨的治学态度是我学习的典范。他的认可一直是我研究的一项动力。我也很感激能够得到施老师给予的机会参与亚洲顶级会议。这段经历无疑是我求学生涯中的一个难忘片段。无论将来我在何处继续我的工作和学业，我将谨记施老师的教诲。

其次，我要感谢我的学长徐辉。在我刚进实验室的时候，对研究工作并不熟悉。每当我遇到问题，徐辉总是我第一个去请教的人，而他总是很耐心的帮助我。在学习研究上，辉哥给学弟学妹们树立了一个标杆。我一直暗暗希望自己在学术上超越辉哥，而在做人做事上我还有太多东西要向辉哥学习。

此外我还要感谢朝夕相处的同学们。栗念龙是我从本科第一天就住在一起的室友，六年的相处让我们相互之间不能更熟悉。我很高兴他拿到了心仪的 offer，相信他的能力不久会在新的环境中再次被证明，同时也提前祝福他和未婚妻的幸福生活。李金波从本科开始就是我在学习上的竞争对手，而私下里也是经常混在一起。如今金波也决定出国继续学业。希望我们在大洋彼岸常联系。卢志坚一直是一个强力的竞争对手，而我发自内心地佩服他淡然的心态。他让我看到很多自己的不足之处。我相信若干年后他会在自己喜欢的领域里开拓出一片天地。还有同实验室的同学们，我很佩服朱彦在科研上的那股钻研劲头，感谢她在我准备出国材料时提供的帮助与鼓励，希望我们都能拿到名校的 offer。谢谢做事认真为人热心的董兰兰在平日里提供的帮助，还有擅于编程热爱摄影的陈家俊，做事有想

法的孙涛，以及下一届学弟程建东，实验室的将来可要靠你了。

最后我要感谢我的父母，感谢他们在我求学期间提供的精神与物质支持。没有父母创造的条件就不会有我现在的学习成果。

我们交大微电子学院是个小地方，但是并不缺少人才。作为她从本科培养起来的第三届学生，希望在将来有一天我能为这个学院的发展做出自己的一份贡献。

攻读硕士学位期间已发表或录用的论文

[1] Yang Song and Guoyong Shi, Hierarchical Graph Reduction Approach to Symbolic Circuit Analysis with Data Sharing and Cancellation-free Properties. In ACM/IEEE Asia South-Pacific Design Automation Conference (ASP-DAC), pp.541-546, Jan. 2012.