分类号：TP331.1　　密　　级：公开
ＵＤＣ：　　单位代码：11646

宁波大学

硕士学位论文

论文题目：基于 AIG 的双逻辑面积优化技术

学　　号：1411082624
姓　　名：赵思思
专 业 名 称：电路与系统
学　　院：信息科学与工程学院
指 导 教 师：夏银水

论文提交日期：2017 年 05 月 27 日

A Thesis Submitted to Ningbo University for the Master's Degree

Area Optimization Technique Based on AIG for Dual-logic

Candidate： Sisi Zhao

Supervisors：(Associate) Professor Yinshui Xia

Faculty of Electrical Engineering and Computer Science
Ningbo University
Ningbo 315211， Zhejiang P.R.CHINA

May 27, 2017

基于 AIG 的双逻辑面积优化技术

摘　要

面积作为集成电路设计的首要指标，一直是逻辑综合和优化的主要目标。逻辑综合与所采用的逻辑密切相关，逻辑函数通常是基于传统的布尔逻辑(Traditional Boolean Logic，TBL)实现。研究表明 Reed-Muller 逻辑(RML)在表示一部分逻辑函数时，其速度、面积、功耗等性能相较于 TBL 表示有很好的优化，又由于单一逻辑表示逻辑函数进行逻辑优化的局限性，因此逻辑函数可以采用 TBL 和 RML 的双逻辑来表示。在逻辑综合与优化过程中，逻辑函数的表示方法同样重要。近年来，与非图(And-Inverter Graph，AIG)作为一种逻辑函数表示形式被广泛应用于逻辑综合与优化过程中。

本文从逻辑函数的图形表示 AIG 出发，提出了一种逻辑函数基于 AIG 的逻辑探测方法。又结合逻辑函数的双逻辑表示，提出了一种逻辑函数基于与/异或/非图(And-Xor-Inverter graph，AXIG)的双逻辑优化与映射方法。在映射过程中，提出了基于标准单元库映射方法。本文内容主要分为以下三个部分：

(1)基于 AIG 实现逻辑函数的探测。给定一个逻辑函数，通过探测 AIG 图中满足特定逻辑的结构，进行逻辑函数的探测。在探测过程中，根据 AIG 图中与门节点和反相器相结合可以实现节点类型的转换。将该方法运用到标准电路中，实验结果表明该方法可以有效的实现图形的压缩与逻辑函数的优化，为后续的逻辑优化奠定基础。

(2)基于双逻辑实现面积优化方法：对于逻辑函数的表示，在 AIG 基础上，引入了 XOR 节点，提出了一种新的数据结构 AXIG，即逻辑函数基于 AXIG 的双逻辑优化与映射方法：通过 AXIG 结构表示 TBL 和 RML，实现了逻辑函数的双逻辑图形表示。选择不同的 XOR 结构进行图压缩，并对映射过程中的局部逻辑结构重映射，最终实现逻辑函数的面积优化方法。实验结果表明，与学术界逻辑综合优化工具 ABC 相比，平均 AXIG 节点数明显减少，电路中的晶体管数具有一定改进。

(3)基于 AXIG 的工艺映射方法：讨论了现有的逻辑函数映射方法，本文将逻辑函数的 AXIG 图应用于已提出的极性图方法，根据目标单元库中的逻辑单

元，完成逻辑函数与工艺相关的映射，达到电路中反相器最小化的目的，最终实现电路面积优化技术。

关键词：双逻辑，　逻辑综合，　与工艺相关的映射，　与非图

Area Optimization Technique Based on AIG for Dual-logic

Abstract

Area，as the primary indicator of integrated circuit design, has been the main goal of logic synthesis and optimization. Logic synthesis is closely related to the logic representation and logic functions are usually represented by Traditional Boolean logic (TBL). Studies have shown that for some circuits, using Reed-Muller logic (RML) has better speed, area, power consumption than TBL. Since there is synthesis limitation of single logic representation, the logic function is represented in TBL and RML called dual logic. In logic synthesis and optimization process, the representation method of logic function is equally important. In recent years, And-Inverter graph (AIG) as a logic function of the representation form is widely used in logic synthesis and optimization process.

This paper starts from the representation method of logic function using AIG, and proposes a novel method of detecting the structure of XOR logic based on AIG. A new graph called And-Xor-Inverter graph (AXIG) is constructed to represent the dual logic form of logic function. In mapping process, a method based on standard cell library mapping is proposed. The content of this thesis is divided into the following three parts.

(1) Detection of logic functions based on AIG. Given a logic function, the logic function is detected by the AIG. In detection process, according to the gate node and the combination of the inverter can be achived node type conversion. The experimental results show that this method can effectively realize the logic optimization of graph compression which lays the foundation for the subsequent logic optimization.

(2) Achieved the area optimization based on the dual logic. A novel method is proposed for logic functions synthesis using And-Xor-Inverter graph (AXIG). First, logic function is represented in And-Inverter graph (AIG) and an optimized AXIG is constructed by detecting the structure of XOR logic; Second, transistor count is used to measure the area of the circuit in the process of technology mapping. Finally, the area optimization method based on AXIG is realized. The proposed method is implemented in C language and tested on MCNC benchmarks. Compared with state-of-the-art optimization tools, the experimental results show the efficiency of the proposed method.

(3) Applied to the proposed polarity graph method based on AXIG. According to the logic cell in the standard library, the technology mapping is completed with the logic function

implemented the circuit with the smallest inverters, and finally realized the circuit area optimization technology.

Keywords：dual-logic， logic synthesis， technology mapping， and-inverter graph

目　　录

引　言

随着集成电路技术的快速发展和工艺水平的不断提高，集成电路设计技术已由基于晶体管、逻辑单元设计发展到基于 IP 核的片上系统(System-on-Chip，SOC)设计，其中功耗、面积、速度和可复用性成为 IP 核设计的重要指标。在集成电路设计中，逻辑级是连接 RTL 级和电路级的关键一级，其中逻辑优化与映射又是最为关键的技术之一，这是由于更具优化的逻辑函数根据设计指标的不同可直接获得优化的 IP 核。

逻辑函数由逻辑变量及对变量进行逻辑运算的操作符组成。传统的布尔逻辑(Traditional Boolean Logic，TBL)采用与/或/非(AND/OR/NOT)作为基本操作符，而 Reed-Muller 逻辑(RML)则采用与/异或(AND/XOR)或者或/同或(OR/XNOR)作为基本操作符。操作符的选择对设计出的硬件电路质量有很大的影响，TBL 面向通用逻辑函数，而 RML 对于实现含有模 2 加和模 2 乘功能的通信电路、奇偶校验电路等算术电路，所得到的电路面积、速度和功耗等性能有较大改善[1]，这一特性使得研究者们提出结合 TBL 和 RML 进行双逻辑综合方法，以期面向通用逻辑实现优化[2]。

在逻辑综合与优化过程中，逻辑函数的表示方法显得尤为重要。逻辑函数的图形表示由于结构简单、与电路一一对应等优点受到重视。如 Lee 提出的二叉决策图(Binary Decision Diagram，BDD）常被用来解决逻辑优化的问题。但用 BDD 表示逻辑函数时，变量序与节点数之间关系密切。因此，Bryant 提出了一种称之为简化有序 BDD（Reduced Ordered BDD，ROBDD）[3]的数据结构，它是布尔函数的一种规范型，但在优化过程中其算法实现的时间复杂度和空间复杂度也受到变量序的影响。近年来提出的一种与非图（And-Inverter Graph，AIG），由于不受变量序等约束条件的影响，因此在表示逻辑函数时相比于 BDD 更高效而受到越来越多的关注[4]。

逻辑函数的映射是逻辑综合中的一个重要环节。目前大多数 EDA 工具是基于预先设计好的标准单元库来实现逻辑函数的映射。而标准单元库根据逻辑功能的不同，又可分为组合逻辑单元和时序逻辑单元。在标准单元库中大部分是由组合逻辑单元组成，如反相器（Inverter）、与非（NAND）门、或非（NOR）门等基本门，和具有复合功能的逻辑单元，如异或（XOR）门、同或（XNOR）门、与或非（AOI）门等复合门。工艺映射根据电路设计的需求不同，采用不同的逻辑单元实现。

本文提出了一种逻辑函数基于与/异或/非图（And-Xor-Inverter graph，AXIG）的双逻辑优化与映射方法。首先采用 AIG 表示逻辑函数并探测其中的 XOR 门，构建一种优化的 AXIG 实现逻辑函数的双逻辑表示；其次，在工艺映射过程中，通过选择优化的门电路单元进行映射，实现逻辑函数基于 AXIG 的面积优化方法。本文提出的方法用 C 语言编程实现，并应用于 MCNC benchmark 的组合电路进行验证。实验结果表明，与学术界逻辑综合优化工具 ABC 相比，平均 AXIG 节点数明显减少，电路中的晶体管数具有一定改进。

1 绪论

1.1 集成电路的设计

所谓定制集成电路是按用户的需要而专门设计制作的集成电路。通常集成电路的设计分为全定制设计、半定制设计和基于可编程器件的设计[5]。在全定制集成电路设计过程中，设计者需要完成所有晶体管和互连线的详细版图。而在半定制设计过程中，设计者是通过使用已经设计好的子电路来完成整个电路的设计，而这个设计好的子电路本身也是基于全定制设计实现的。由于全定制设计对于设计的精度要求很高，因此可以达到最大程度的芯片优化。但是这种设计方式在某种程度上比半定制的设计浪费更多的时间和精力，因此目前的集成电路设计都采用半定制设计和基于可编程器件的设计。而半定制设计受到电路优化的限制，这反而促进了电子设计自动化(EDA)的发展，为电路的优化提供了解决问题的空间。

在半定制集成电路设计过程中，最主要的设计是基于单元的设计。而标准单元库通常作为半定制设计的基本模块，所设计的逻辑结构通过标准单元库映射到合适的位置，之后进行内部的布局布线。由于单元库的提供者使用定制设计的方法完成单元库的开发，因此这种通过标准单元库实现电路的方法也被称为半定制设计。目前很多 EDA 工具可以自动生成库，并且也有通过使用自由库的方法提高电路的性能[6]。

在基于标准单元库设计过程中，通常选择一些简单的逻辑门作为标准单元库的组成部分，同时也会包含一些由简单逻辑门组成的复合门。在没有具体单元库的映射方法中，复合逻辑门的变化会更多，例如自由库的方法。自由库映射的方法主要是为了证明使用更复杂的逻辑单元比简单的逻辑单元更具有优势，而这些方法均是以达到减少所需要实现的集成电路的晶体管数为标准。大幅度减少晶体管数的方法对于设计的成本函数起到了一个积极的作用，因此最终电路的性能好坏是通过衡量电路中所需的晶体管数为标准的。

在超大规模集成电路设计过程中需要不同的成本函数，有些成本函数的约束条件并不是单一的，甚至是相互对立的。大部分使用成本函数进行优化的设计目标通常为芯片面积、电路延时、功耗、可测试性等。在最近几年，设计过程中的优化工作通常由 EDA 工具解决，正是因为 EDA 工具的使用，使得目前集成电路的设计取得更好的效果。而对于更复杂的具有多级电路的设计过程，

EDA 的使用变得越来越重要。随着 EDA 工具中所使用的算法不断变化，也是决定了一个电路优化效果的关键因素。

本课题基于以上背景，针对特定的单元库选择合适的逻辑门并结合目前学术界最先进的逻辑综合工具 ABC[7]进行电路优化的研究。经过 ABC 的优化可以提供节点数更少的初始电路，在这个初始电路的基础上，进行逻辑函数的重新组合，所获得的电路通过使用简单逻辑单元实现，相比于之前用更复杂的逻辑单元实现有较好的结果，或更进一步在简单门实现的基础上采用复合门实现，进行逻辑函数的重映射。这样做的目的就是结合 EDA 工具，通过优化中间电路，最终实现电路面积最小化技术。

1.2 研究目的与意义

本文主要研究数字集成电路的设计。数字集成电路的设计大都采用自上而下的设计模式。首先是行为级的设计，也就是将实际的电路功能通过 HDL 语言描述，形成寄存器传输级 RTL 代码，之后就是逻辑综合。所谓逻辑综合是将设计实现的 HDL 代码转换成门级网表形式。在逻辑综合过程中，需要设定约束条件，针对约束条件进行电路的设计使其满足面积、功耗或延时的标准。数字电路按照逻辑功能不同，又分为组合逻辑电路和时序逻辑电路[8]，而本文的逻辑综合主要是针对组合逻辑电路，这样不用考虑电路中的触发器等物理器件，只针对电路的功能进行逻辑函数的转化。

近几年，逻辑综合对于集成电路的设计显得尤为重要。目前，大多数的逻辑综合与优化方法均是基于 TBL 发展而来，如 MIS[9]，SIS[10]和 ABC[11]。MIS 是最早的逻辑综合工具，之后 SIS 是在 MIS 的基础上进行开发，主要针对多级组合逻辑电路的综合和优化。VIS[12]是逻辑验证领域的工具，它主要是基于 BDD[13]通过 CUDD[14]包开发而成。在 90 年代中期，SIS 的发展达到了瓶颈，而逻辑函数的表示又有了新的改进。在 2000 初，AIG 作为一种新的数据结构被提出用于解决逻辑函数的形式验证问题。同一时期，MVSIS[15]一种多值逻辑综合系统也被开发出来，主要用于操作多值逻辑间的关系。AIG 的提出代替了传统的逻辑函数表示和 BDD 进行逻辑函数的综合。SIS 和 ABC 最主要的不同在于 SIS 中的节点依然用与或式来表示，而 AIG 中的节点是二输入与门。一个 SIS 网络可以通过分解每个节点进行网络之间的转换。

由于上述的逻辑综合工具均是基于 TBL 发展而来，并未涉及 RML，研究表明双逻辑在表示电路时，可得到面积、功耗和延时等性能的提高，因此本文提出基于双逻辑的图形表示。而双逻辑的实现首先是逻辑函数的探测，目前已有

关于逻辑函数探测的研究，但所提出方法仍有一定局限性。文献[16]提出基于BDD 的 XOR 探测，但受到 BDD 变量序的限制。目前逻辑综合的优化方法大部分是在 AIG 的基础上完成的[17]。AIG 是一种表示逻辑函数的图形结构。它由二输入的与节点和节点与节点之间的连线构成。目前学术界著名的逻辑综合优化工具 ABC 就是在 AIG 的基础上进行的节点数优化。

逻辑函数的映射也是逻辑综合的一个重要环节，通常是通过标准单元库或者查找表 LUT 实现逻辑函数的映射。而在 AIG 图中与节点也可以直接由二输入的简单门实现，如 AND、OR、NAND、NOR 等。这些单元都是通过应用德摩根定理和反相器的结合由 AND 变化而得。在实际的电路实现中，通常以电路中的晶体管数作为衡量电路的一个标准。观察可知二输入 NAND 和二输入 NOR 单元用 CMOS 实现需四个晶体管，而二输入的 AND 和 OR 单元需要六个晶体管构成。因此，二输入的 NAND 和二输入的 NOR 在实际电路设计中更受欢迎。晶体管的数量可以通过选择所需晶体管数较少的逻辑单元来实现，而同时反相器的数量可能要增加[18]。

本文的目的是通过使用一系列的算法获得晶体管数最少的电路。首先提出了一种逻辑函数基于与/异或/非图（And-Xor-Inverter graph，AXIG）的双逻辑优化与映射方法:采用 AIG 表示逻辑函数并探测其中的 XOR 门，构建一种优化的 AXIG 实现逻辑函数的双逻辑表示；其次通过基于简单门的设计，进行图中节点类型的变换，实现逻辑函数基于 AXIG 的面积优化方法。这些算法主要集中在如何减少 AIG 图中的节点数，以及在映射成简单门的过程中，如何减少图中反相器的数量。

1.3 论文内容与结构

本文研究的主要内容是基于 AIG 的双逻辑面积优化技术。本文的研究工作分为以下三个部分：第一部分主要是研究基于 AIG 的逻辑探测，给定一个具体的逻辑函数表达式，通过在 AIG 图中探测满足具有特定功能的逻辑表达式结构，而这些特定的逻辑结构有利于逻辑优化而且便于后期逻辑映射的研究。第二部分主要是基于 AIG 的双逻辑表示，通过逻辑函数的探测，构建一种基于 AIG 的新型数据结构，即 AXIG。AXIG 实现了双逻辑的图形表示形式。由于 AIG 图中所表示的 XOR 结构不同，可选择的 XOR 结构亦不同，所生成的 AXIG 也不同。第三部分重点在于后期与工艺相关的逻辑映射，在 AXIG 图中通过标准单元库实现逻辑函数的映射，最终达到电路面积最小化的目的。

根据以上内容分析，论文共分成六个章节：

第一章为绪论，主要介绍了本文所研究的集成电路设计方法，以及选题的研究背景及意义，同时对本文所作的工作进行简单概述。

第二章是对数字电路设计流程的介绍，包括逻辑综合与优化的流程，以及实现该流程的两个步骤；TB 逻辑与 RM 逻辑相关的基本概念和简化规则以及双逻辑的定义和逻辑函数的图形表示。

第三章是基于 AIG 的逻辑探测，主要内容是根据 AIG 中 XOR 运算的特点，探测所有满足 XOR 运算结构的 AIG 子表达式，为后续构建一种新的数据结构 AXIG 的研究奠定了基础。

第四章是基于双逻辑的门级图形表示。构建一种新的数据结构 AXIG，实现双逻辑的门级图形表示；在构建 AXIG 过程中，考虑 XOR 结构在 AIG 图中的不同表示，优化构建的 AXIG；其次进行逻辑函数基于标准单元库的映射，采用由简单门和复合门构成的单元库，针对每个与门节点考虑可以映射的所需晶体管数最小的方式进行映射，最终实现电路面积最小化的目的。

第五章是基于 AXIG 提出反相器最小化算法。介绍了现有的基于 AIG 的图形映射方法，而本文根据标准单元库中特定的简单门单元，用极性图的方法实现反相器最小化。而对于 XOR 门，由于 XOR 运算中反相器伸缩性质，可以将图中的反相器推到 XOR 的周围，化简成无反相器的 XOR 门或者 XNOR 门，从而实现图中反相器最小化的目的。

第六章是总结本文的主要工作内容，并且指出了本文方法的优缺点，展望了基于 AIG 的双逻辑优化技术和有关反相器最小化的搜索方法的发展方向和前景。

2 数字电路设计基础

本章主要介绍数字电路设计过程中有关逻辑设计的基本技术背景，其中包括逻辑综合设计流程和双逻辑的基本概念与性质，以及用来表示逻辑函数的几种主要数据结构。

2.1 逻辑综合流程

逻辑综合是集成电路设计过程中用来实现电路内部结构的重要环节。多级逻辑综合是目前综合的主流。所谓多级逻辑是指电路从一个原始输入到一个原始输出的任意路径之间包含多个层级的逻辑门。逻辑综合主要有两个阶段:第一个阶段是执行自动翻译，将 RTL 转换成逻辑网表，通常用来将硬件描述语言，转化为相互关联的逻辑网表，最终通过与工艺相关的映射，实现具体逻辑门的电路实现；第二，在逻辑综合过程中会优化成本函数，如芯片面积、功耗、速度和可测试性等。针对不同的成本函数，所设计生成的电路也不同。而在实际的电路优化过程中，成本函数不单只是在一个方面进行限制，逻辑优化最终的结果往往有可能是多个约束条件共同作用下所生成的电路，而不是单纯的为了达到电路面积最小所生成的电路，或者为了实现电路速度最快而生成的电路，最终的结果往往是几个约束条件相互折中的实现。因此这样的逻辑优化被称为逻辑综合。逻辑综合流程通常分为两个阶段：与工艺无关的逻辑优化和与工艺相关的映射。

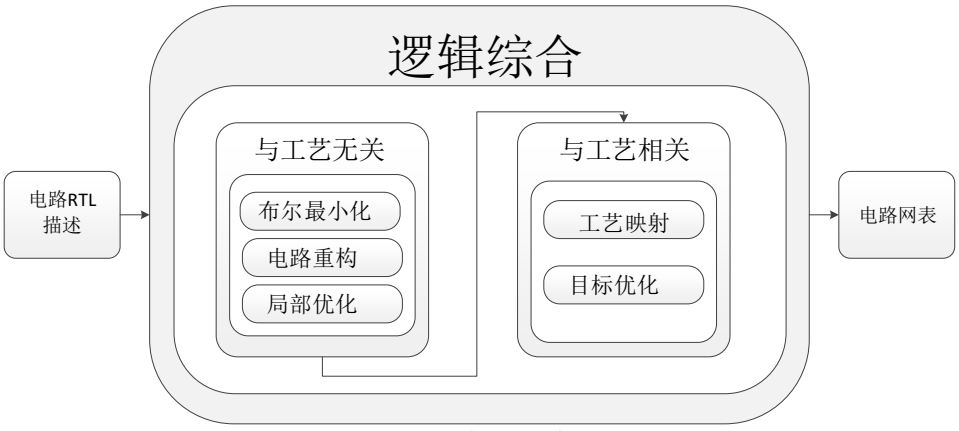


图 2.1 逻辑综合流程
Fig. 2.1 logic synthesis flow

图 2.1 描述了整个逻辑综合的过程。以下两个小节详细介绍了逻辑综合的两个阶段。

2.1.1 与工艺无关的逻辑优化

逻辑函数按照级数划分，可分为二级逻辑和多级逻辑。二级逻辑又包含两种规范式，分别称为与或两级规范形式(sum-of-products two-level form, SOP)和或与两级规范形式(product-of-sums form, POS)[19]。其中或与形式的第一级均为或项，第二级均为与项，而与或形式刚好相反。逻辑电路按照功能分为两种，分别是组合逻辑电路与时序逻辑电路。而本文的设计方法主要是针对组合逻辑电路进行的多级逻辑综合与优化。

逻辑函数优化时通常把文字数作为评估逻辑函数表达式复杂度的标准。文字数即为逻辑函数中组成每个乘积项的变量数之和。例如 E1 和 E2 是同一个逻辑函数的两个不同表达式，如果 E1 包含的文字数比 E2 包含的文字数少，则称 E1 比 E2 更简单，这就实现了电路中所谓的逻辑优化。

多级逻辑的优化标准相对复杂了些，不比二级逻辑优化那样，总能把逻辑函数化简成二级逻辑的最简形式，即多个最小项的乘积项之和。较之于二级逻辑，多级逻辑每个门的扇入数减少，但是相应的也增加了电路的延时。若想降低逻辑门的级数，可以减少逻辑门个数，因为从输入到输出的路径上经过的门越少，延时越少。然而，达到最小延时标准的电路并不能得到最少数量的逻辑门实现，同时满足逻辑门数最少和整个电路延时最小的电路是不可能实现的。所以多级逻辑的逻辑优化是以增加电路延时为代价达到面积减少的目的。

通常多级逻辑的优化采用启发式方法[21]，这种方法在多数情况下是有效的，但这仍不能保证得到最简表达式。多级逻辑优化最根本的方法就是寻找可能存在的相同或相反项，实现最大程度的公共子表达式的共享，然后通过适当的变换，得到更符合设计的速度，面积，或可用单元的要求。通常多级逻辑的优化有五种与工艺无关的基本操作：分解、提取、分解因子、代入、压缩。这五种操作主要是通过运用表达式的基本定理和性质来实现的，最常用到的操作就是提取公因式，确定公共子表达式并代入相关表达式，实现表达式的反复利用，得到共享结构，达到结果优化的目的。

在与工艺无关的逻辑优化过程中，主要进行的是逻辑函数的最小化，电路结构的重新调整，以及局部电路的最优化。由于在此过程中逻辑优化和逻辑模块、物理器件无关，电路的优化过程不涉及到器件的任何相关信息，与最终实现网络所用的逻辑门类型无关，所以被称为与工艺无关的逻辑优化。

2.1.2 与工艺相关的映射

逻辑函数可以通过多种类型的逻辑门实现。这种处理被称为与工艺相关的优化，它对目标电路的逻辑门数，连线的复杂度，和延时性能都有很大的影响。

在大多数集成电路设计中，设计者可以利用预封装的函数库实现自己的设计。库单元包含所有主要的逻辑门，通常为二输入和三输入的各种类型逻辑门，还可能有更多输入的逻辑门，以及由一些基本逻辑门构造的更为复杂的逻辑门。逻辑函数可以通过映射技术互相连接逻辑门实现门和连线数目的最小化。衡量逻辑函数复杂度的一种标准是计算该逻辑函数所包含的文字数，而文字数提供了实现逻辑函数所需的晶体管数和连线数量的近似估计。由于电气特性和封装的需要，在给定的技术条件下，逻辑门通常只包含有限个数的输入，通常二输入、三输入、四输入的逻辑门很常见，所以执行逻辑最小化的一个主要原因是为了减少函数表达式中的文字数，从而减少逻辑门扇入的个数。逻辑门的数量越多，电路所占的面积越大。逻辑门的数目与实现设计所需的元件数量之间有很强的相关性，这里元件是指库模块或集成电路单元。用于制作的最简设计往往包含最少的逻辑门，而不是最少的文字数。

基本的逻辑模块有非门、与门、或门、与非门、或非门、异或门和同或门。一个逻辑函数可以出现多个等效的逻辑表达式，但规范形式的表达式只有一种。不同逻辑模块的面积和速度不同，因此对设计者的要求也具有很大的影响。

在与门和或门用 CMOS 实现时，由于与门和或门分别通过与非门和或非门添加反相器所得，因此它们所需的实际晶体管数要比与非门和或非门多出两个。因此对于逻辑与工艺相关的优化来讲，为了实现电路最终逻辑综合的优化，与非门和或非门在实际应用中要比与门和或门应用广泛得多。

在与工艺相关的逻辑优化过程中，将通用电路转换为基于预先设定好的含有具体逻辑门信息的门级网表为工艺映射。在这个过程中，所有的逻辑门的相关信息，如门的尺寸、关键路径的延时优化、缓冲器的插入和扇出限制等都将被用于实现最终电路以某性能为导向的最优化。通常，成本函数与最终实现的工艺相关，而这些均是来自预先设计好的单元库中读取的。工艺映射是在前期与工艺无关的逻辑优化基础上实现的，这个过程通常分为三个步骤：分解、匹配和覆盖。通过这三个步骤最复杂的逻辑函数都能够利用更基本的函数通过连接逻辑门实现，但最终实现的电路完全取决于预先设计的库中所包含的具体逻辑门。

2.2 双逻辑理论基础

所谓的双逻辑技术是由 TBL 和 RML 两种逻辑共同表示一个逻辑函数，其中变量与变量之间既包含 TBL 中的运算符又包含 RML 中的运算符。根据逻辑函数自身的特性，不同的逻辑函数自身特性不同，有的逻辑函数用单一的 TBL 来表示更适合逻辑优化，而有的逻辑函数中包含更多的适合 RML 实现的表达式，因此采用 RML 实现对最终电路的优化效果更好。下面主要介绍这两种逻辑的基本概念和主要性质，为后续的逻辑变换奠定基础。

2.2.1 TBL 的定义及性质

TBL 是由 AND、OR 和 NOT 三种运算组成。由于 AND、OR 和 NOT 三种运算构成完备集，因此，任意一个逻辑函数都可以转化成由这三种逻辑运算表示，相应地，电路最终均可以采用与门、或门、非门实现。

利用 AND、OR 和 NOT 运算建立的逻辑代数称之为 TB 代数。

以下为常用的 TB 代数的公式和定理。

(1) 与常量 0 和 1 的运算：$X+0=X \quad X+1=1 \quad X*1=X \quad X*0=0$

(2) 重叠定理：$X+X=X \quad X*X=X$

(3) 多变量有关公式：

交换律：$X+Y=Y+X \quad X*Y=Y*X$

或运算结合律：$(X+Y)+Z=X+(Y+Z)=X+Y+Z$

与运算结合律：$(X*Y)*Z=X*(Y*Z)=X*Y*Z$

分配律：$X*(Y+Z)=X*Y+X*Z \quad X+(Y*Z)=(X+Y)*(X+Z)$

(4) 德摩根定律：$\overline{X+Y}=\overline{X}*\overline{Y} \quad \overline{X*Y}=\overline{X}+\overline{Y}$

以上四种定律是 TBL 最基本的定律，并且这些公式都可以通过计算得到推理和验证。这些定理也是逻辑优化的基础。例如在一个表达式中，可以利用 TBL 中的分配律化简最初的逻辑函数，即 $x*y+x*z=x*(y+z)$，通过分配律的化简，最初的逻辑函数等号左边的四个文字数减少到化简之后等号右边的三个文字数。从逻辑表达式中可以看出等号右边的表达式比等号左边的表达式少了一个运算符，也就是在最终电路实现时，可以达到优化逻辑门个数的目的。TBL 中的交换律同样可以起到优化函数的作用，通过交换输入变量可以产生一个新的但功能上同最初等效的电路，利用交换律在实际电路实现过程中，调整局部逻辑变量的顺序，为变量之间共享或者构成可优化的表达式，提供了进一步优化的可能性。

2.2.2 RML 的定义及性质

RML 是由 AND 和 XOR 或者 OR 和 XNOR 构成的运算完备集。以下是 XOR 和 XNOR 运算用 TBL 实现：

$$x_1 \oplus x_2 = \bar{x}_1 x_2 + x_1 \bar{x}_2 \quad \text{（公式 2.1）}$$

$$x_1 \text{ e } x_2 = x_1 x_2 + \bar{x}_1 \bar{x}_2 \quad \text{（公式 2.2）}$$

从公式 2.1 和公式 2.2 中可以看出 XOR 运算和 XNOR 运算都是由两个变量乘积项的不同组合的与或形式表示，只是乘积项中取反的变量不同。

因为 RML 中的运算符也是一个完备集，所以任意逻辑函数都可以用 RML 实现。

给定一个逻辑函数 $f(x_1,\cdots x_{i-1},x_i,x_{i-1}\cdots,x_n)$，将其表示成 RML 的形式，如下所示：

$$\begin{aligned} f(x_1,\cdots x_{i-1},x_i,x_{i-1}\cdots,x_n) &= \bar{x}_i f(x_1,\cdots x_{i-1},0,x_{i-1}\cdots x_n) + x_i f(x_1,\cdots x_{i-1},1,x_{i-1},\cdots x_n) \\ &= (1\oplus x_i) f(x_1,\cdots x_{i-1},0,x_{i-1}\cdots x_n) + x_i f(x_1,\cdots x_{i-1},1,x_{i-1},\cdots x_n) \\ &= f(x_1,\cdots x_{i-1},0,x_{i-1}\cdots x_n) \oplus x_i[f(x_1,\cdots x_{i-1},0,x_{i-1}\cdots x_n) \\ &\quad \oplus f(x_1,\cdots x_{i-1},1,x_{i-1},\cdots x_n)] \\ &= f_0(x_1,\cdots x_{i-1},x_{i-1}\cdots x_n) \oplus f_1(x_1,\cdots x_{i-1},x_{i-1},\cdots x_n)x_i \end{aligned} \quad \text{（公式 2.3）}$$

其中 $f_1(x_1,\cdots x_{i-1},x_{i-1}\cdots x_n) = f(x_1,\cdots x_{i-1},0,x_{i-1}\cdots x_n) \oplus f(x_1,\cdots x_{i-1},1,x_{i-1}\cdots x_n)$，$f_0(x_1,\cdots x_{i-1},x_{i-1}\cdots x_n) = f(x_1,\cdots x_{i-1},0,x_{i-1}\cdots x_n)$。因此从公式中可以得出，任一逻辑函数都可以根据某个输入变量及反变量展开成 RML 表示形式。根据 RML 的运算性质，逻辑函数在展开的过程中，还有三个香农展开的变形，同样也是为了逻辑函数的优化而提出，如下所示：

$$f = x_i f_{x_i} \oplus \bar{x}_i f_{\bar{x}_i} \quad \text{（公式 2.4）}$$

$$f = f_{\bar{x}_i} \oplus x_i (f_{x_i} \oplus f_{\bar{x}_i}) \quad \text{（公式 2.5）}$$

$$f = f_{x_i} \oplus \bar{x}_i (f_{x_i} \oplus f_{\bar{x}_i}) \quad \text{（公式 2.6）}$$

其中 $f_{\bar{x}_i}=f(x_1,...,x_i=0,...,x_n)$ 和 $f_{x_i}=f(x_1,...,x_i=1,...,x_n)$。这三种不同的扩展方式通过运用 RML 中常量与变量之间运算的性质，为逻辑优化提供了更多的选择。最后 n 个变量逻辑函数转换成公式 2.7 所示：

$$f\left(x_{n-1},x_{n-2}...\right)=\oplus\sum_{i=0}^{2^n-1}b_i\pi_i \qquad \text{（公式 2.7）}$$

其中 $i=\left(i_{n-1},i_{n-2},...,i_0\right)$，是一组二进制序列。$b\in\{0,1\}$，$\pi_i=x_{n-1}x_{n-2}\cdots x_0$ 称之为 $\pi_{-\text{term}}$，x 可以是原变量 x 或者反变量 $\bar{x}$。

$$x_j=\begin{cases}1, & i_j=0\\ x_j, & i_j=1\end{cases} \qquad \text{（公式 2.8）}$$

以下是关于 RML 运算的相关性质。

（1） 常量之间的异或运算：$0\oplus0=0\quad 0\oplus1=1\quad 1\oplus0=1\quad 1\oplus1=1$；

（2） 常量与变量之间的异或运算：

$0\oplus A=A\quad 1\oplus A=\bar{A}\quad A\oplus A=0\quad A\oplus\bar{A}=1$；

（3） 变量与变量之间的异或运算：

交换律：$A\oplus B=B\oplus A$；

结合律：$A\oplus(B\oplus C)=(A\oplus B)\oplus C$；

分配律：$A(B\oplus C)=AB\oplus AC$；

（4） 求反运算：

De Morgan 定理：$\overline{A\oplus B}=\bar{A}\,\varepsilon\,\bar{B}$；

反演律：$\overline{f(A,B,\cdots;\gamma,+,\varepsilon,\oplus)}=f(\bar{A},\bar{B},\cdots;+,\gamma,\oplus,\varepsilon)$；

（5） 化简的公式：

$A(A\oplus B)=A\bar{B}$；

$A+\bar{A}B=A+B$；

$(\bar{A}\oplus B)(A\oplus C)(B\oplus C)=(\bar{A}\oplus B)(A\oplus C)$；

（6） 特殊性质：

变量位置变换：若 $A\oplus B=C$，则 $A=B\oplus C$；

非号位置变换：

$$\bar{A}\oplus B\oplus C = A\oplus \bar{B}\oplus C = A\oplus B\oplus \bar{C} = \overline{A\oplus B}\oplus C = \overline{A\oplus B\oplus C}\ 。$$

逻辑函数通过上述的 RML 运算性质进行逻辑函数的优化和映射。

2.3 逻辑函数的表示形式

逻辑函数可以通过多种方式来实现，其中包含的描述方式有布尔代数表达式、以表格形式记录电路输入与输出值的真值表、直观反映乘积项之间关系的二维平面卡诺图和具有 n 维的 TB 立方体等[18]。真值表、卡诺图、立方体在描述小型电路的逻辑功能时比较有效，但当逻辑函数的输入个数逐渐增多时，它们的规模均是随着输入个数呈指数级增长的。

对于逻辑级优化，目前最多的是选择 EDA 工具进行逻辑综合和优化。而目前的 EDA 工具均是基于特定的数据结构进行逻辑函数的表示及优化。以下是几种典型的数据结构表现形式。

2.3.1 二元决策图

二元决策图 BDD 是布尔函数的一种图形表示。BDD 实际上是由一组二进制值决定每个变量的图形结构，以终端的常量节点标志该乘积项的结束，是一个有向无环图，它由内部节点和两个终端节点以及节点与节点之间的实线或者虚线相连。BDD 这种数据结构在很早之前就已经被提出[13]，后来 Bryant 提出了基于 BDD 的一种可以作为布尔函数标准型的数据结构受到研究者们的关注，即 ROBDD[3]。这是因为 ROBDD 预先定义了变量的顺序，同时就确定了该逻辑函数的唯一实现，即可以得到逻辑函数的标准型。这对于同一个逻辑函数，具有多种不同的逻辑实现而言，使得优化变得更简单。但由于它是在变量顺序确定好之后进行的表示，因此又受到变量顺序的限制，不同的变量顺序，所生成的 BDD 图的时间和空间规模还是存在很大差距的。

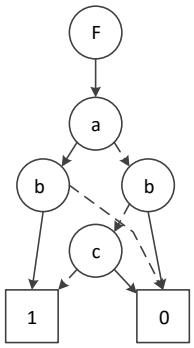


图 2.2 函数 $F = ab + \bar{a}\bar{b}\bar{c}$ 的 BDD 表示形式

Fig. 2.2 BDD of the function $F = ab + \bar{a}\bar{b}\bar{c}$

图 2.2 描述了逻辑函数 $F=ab+\overline{abc}$ 的 BDD 表示形式。其中节点 F 表示该逻辑函数，F 节点没有输入，只有一个输出边，而内部的其它节点表示该逻辑函数的变量。从图中可以看出每个内部节点从上到下都具有两个输出边，一条实线边和一条虚线边。实线边表示该变量的输出为原变量，虚线边表示该变量的输出为反变量，图中从上到下直到终端节点 1 和 0，最终将所有指向终端节点 1 的乘积项相或，得到该逻辑函数。在这个 BDD 图中，变量的顺序已提前设定为 a，b，c。

为了获得逻辑函数的标准型，通常逻辑函数采用 ROBDD 形式表示，又由于函数变量的顺序也要提前设定，而 BDD 的规模通常也和变量的排列顺序相关，这反而限制了 BDD 的优化。当输入变量个数增加，BDD 的规模呈指数型增长。

2.3.2 有向无环图

DAG 是一种有向无环图（Directed Acyclic Graph，DAG）。早期逻辑函数之间的门级关系，也尝试映射到图中或树中实现，DAG 就是这样一种数据结构。图中的节点是 TBL 的操作符。最底端是逻辑函数的输入，除了非节点的单输入，其他内部节点均是二输入节点。逻辑函数中变量之间的逻辑关系均可以从图中直观的显示出来。函数 $F=ab+\overline{abc}$ 的 DAG 表示如下所示：

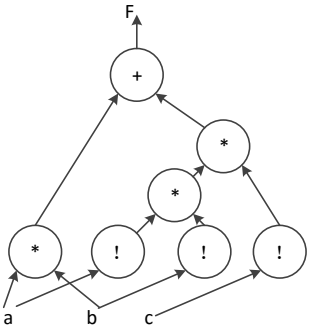


图 2.3 函数 $F=ab+\overline{abc}$ 的 DAG 表示形式

Fig. 2.3 DAG of the function $F=ab+\overline{abc}$

图 2.3 所示是逻辑函数 $F=ab+\overline{abc}$ 的 DAG 表示，图中从输入到输出，通过变量之间的逻辑与、或、非运算可以得到该逻辑函数。图中关于变量的取反操作也是通过单一输入的内部非节点表示，这样对于有关非运算的一些伸缩性质并不能很好的展现出来，也并不能直观的看出有关非运算的变化。因此对于 DAG 图的使用也渐渐的被一种新的数据结构替代，即与非图 AIG。

2.3.3 与非图

与非图作为学术界著名的逻辑综合工具 ABC 所使用的基本数据结构，被用来实现逻辑函数的优化。AIG 在早期就已经被用来解决逻辑函数的验证问题，并且针对 AIG 的结构，提出了一种描述 AIG 结构的文件格式为 aiger[22]，这是一种类似于描述逻辑级的文件格式 BLIF[23]，它可以用于逻辑综合问题中的电路原始输入或输出。AIG 也是一种有向无环图，其中包含二输入的与节点，和输入节点以及输出节点。二输入与节点是一个具有二输入实现与操作的节点。在节点和节点之间也是由两种类型的连线连接，类似于 BDD 中的正边和负边，分别用实线或者虚线相连。一个变量通过实线到达内部节点，则节点的输入为该变量，如果变量是通过虚线连接到内部节点，则该节点的输入为此变量的反变量。

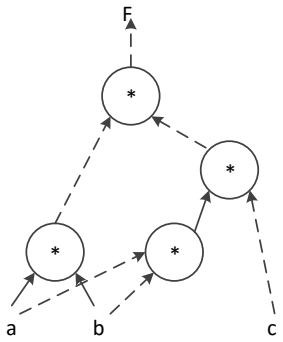


图 2.4 函数 $F = ab + \bar{a}\bar{b}\bar{c}$ 的 AIG 表示形式

Fig. 2.4 AIG of the function $F = ab + \bar{a}\bar{b}\bar{c}$

如图 2.4 所示，是逻辑函数 $F = ab + \bar{a}\bar{b}\bar{c}$ 的 AIG 表示形式。从图中可以看出反变量均是通过虚线表示，相比于 DAG 图节点数减少了，并且反相器不计入电路的内部节点。ABC 用 AIG 实现电路，在生成 AIG 的同时，也进行了逻辑函数的优化，从图中表现为节点数的减少。由于每个节点都是一个与门，而通过改变节点两个扇入的极性和扇出的极性，可以实现节点逻辑功能的转换，因此 AIG 图具有可拓展性，这在接下来的章节进行深入研究。

3 逻辑探测的研究

本章主要讨论基于 AIG 进行逻辑函数的探测。逻辑函数的表示既可以用 TBL 表示又可以用 RML 表示，也可以用双逻辑实现。而为了实现逻辑函数的双逻辑表示形式，首先进行逻辑函数的探测。探测最初的逻辑函数中是否包含有 XOR 运算的结构，其次创建一个新类型的节点代替图中满足 XOR 的 AIG 结构，最终实现逻辑函数的探测和双逻辑的实现。对于逻辑函数的探测，不仅可以搜索基于双逻辑的 XOR 运算，同时可以在 AIG 图中探测更复杂的逻辑运算，如数据选择器 MUX 等。这样可以通过减少 AIG 图中节点数实现逻辑函数的优化。

与、或、非运算是逻辑函数的完备集，而其中的或运算也可以通过与和非运算实现，因此，与非也能组成逻辑函数的完备集，用来表示任意布尔函数，而 AIG 图就是由与节点和非组成的图形结构。

3.1 理论基础

目前，关于 RML 探测已有大量研究。如文献[24]提出了一种用于探测 Pure RM 逻辑的算法，即对于给定的逻辑函数只用 XOR 门实现，由于大多数电路的逻辑函数不能单纯的采用 XOR 门来实现，所以该算法并没有很好的应用到逻辑优化中。文献[25]也对此问题进行了探讨，提出了一种从 XOR 运算特点出发，通过计算逻辑函数最小项之间的汉明距离来判断逻辑函数是否适于双逻辑实现，但该方法提出的条件只是探测 XOR 运算的充分条件，对于一些含有 XOR 运算的函数，虽然不符合条件，但是也同样可以用双逻辑实现；并且该方法仅对于单输出电路有效，对于多输入多输出电路，并不能进行有效的探测。

文献[26]和文献[27]的探测方法均是根据乘积项海明距的特点出发，但文献[26]是在最简项的基础上进行的搜索，而文献[27]是在不相交乘积项上进行的 XOR 运算搜索。对于最简项的逻辑搜索，逻辑函数优化的目的就是达到逻辑函数的最小化，这本身就是一个很难实现的目标。文献[28]提出的判断条件不能很好的探测 XOR 门，并且这些判断 XOR 门的方法都是基于真值表和卡诺图深入的。当电路的输入数量增大时，不利于大电路的搜索和实现。本章根据逻辑函数表示方法的不同，提出了一种有效的图形表示方法来描述电路，在 AIG 的基础上可以直观的对 XOR 结构进行逻辑函数的局部优化。

为了实现逻辑函数的优化与压缩，本章通过探测 AIG 图中可能出现的 XOR 结构，构建一种新的类型节点，达到逻辑压缩的目的。

3.2 问题定义及基本思路

3.2.1 问题定义

本章所要解决的问题如下所述：给定一个 n 变量逻辑函数的 AIG 表示形式。由于 AIG 本身具有可延展性，它是所有图形结构变换的基础，因此针对具体的逻辑，探测 AIG 中符合所要搜索的具体逻辑的 AIG 结构，进行图形的变换与比较。

3.2.2 基本思路

根据问题的定义，提出所要解决问题的具体实施方案如下：逻辑函数的表示形式有多种，研究表明采用双逻辑实现逻辑函数，比单一逻辑实现有一定的优势[25]。因此在 AIG 基础上探测 XOR 结构，进行图形的变换，实现 AIG 的优化，也就是实现了逻辑函数的优化。又由于 AIG 图中的与节点，通过改变节点扇入扇出极性，在保证逻辑功能不变的前提下，可以变换 AIG 图中与节点的类型，因此也可以进行 AIG 的重构。

3.3 研究 AIG 中的逻辑结构

根据不同逻辑用 TBL 实现的不同，在 AIG 图中探测的由与门和反相器组成所得到的结构也不同，找到满足该逻辑的结构特征，实现逻辑的划分与压缩，得到不同的图结构，并进行比较。

3.3.1 可优化的 AIG 图

在进行基于 AIG 图逻辑探测之前，首先要得到一个化简的 AIG 图。因此本章借助逻辑综合优化工具 ABC，通过在终端输入优化命令可得到一个相对简化了的 AIG 图，之后再进行基于简化的 AIG 图的逻辑探测和压缩。在 ABC 中关于逻辑优化方法有很多，而 ABC 的优化主要是针对 AIG 图中的节点数以及图中的层级数，其中主要涉及三个命令:第一个命令是平衡 balance[29]，把 AIG 看成是一个未优化的平衡树，进行平衡优化，通过运用逻辑运算中的基本性质如结合律、交换律和分配律进行层级数的压缩，达到电路延时优化的目的；第二个命令是重写 rewrite，这个命令是通过哈希表实现的。把每个函数可能的结构预先储存在表中，其输入变量的上限设置为 4，当探测到某种逻辑函数，根据逻辑函数的 AIG 节点数判断是否替换表中更少节点数的 AIG 结构，达到节点数优化的目的。第三个命令是重构 refactor[30]，主要涉及到共享节点的优化，如果一个函数可以用图中已经存在的节点进行相应的计算，则直接用图中已经存在的节点进行连接和共享，而不需要重新构建新的子节点。同样是通过局部优化的方

法实现逻辑函数的优化。这三个命令集成在 ABC 包中的 resyn2 脚本中，因此优化一个电路通过脚本就可以实现将逻辑函数优化到相对简化的程度。

3.3.2 异或逻辑在 AIG 中的表示

异或运算 $f=a\bar{b}+\bar{a}b$，也可以用同或运算的反运算表示，即 $f=\bar{g}=\overline{ab+\bar{a}\bar{b}}$，所以异或门在 AIG 中的结构如图 3.1 所示：

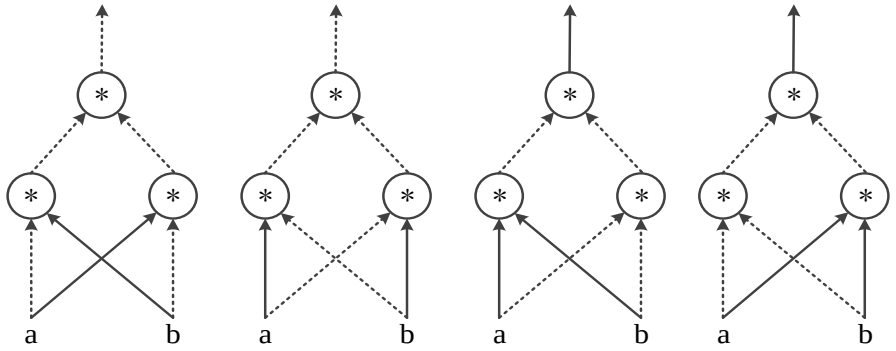


图 3.1 异或门的 4 种 AIG 表示形式: (a) $f=\bar{a}b+a\bar{b}$ (b) $f=a\bar{b}+\bar{a}b$ (c) $f=\bar{g}=\overline{ab+\bar{a}\bar{b}}$ (d) $f=\bar{g}=\overline{\bar{a}\bar{b}+ab}$

Fig.3.1 four AIG representations of XOR: (a) $f=\bar{a}b+a\bar{b}$ (b) $f=a\bar{b}+\bar{a}b$ (c) $f=\bar{g}=\overline{ab+\bar{a}\bar{b}}$ (d) $f=\bar{g}=\overline{\bar{a}\bar{b}+ab}$

从图 3.1 中可以看出一个异或门有四种 AIG 表示形式，前面两种是异或门的基本形式，区别在于乘积项相或的先后顺序不同，通过或运算的交换律得到两种不同形式的异或门结构。而后两种结构是由同或门的反得出，而又由于交换律，因此用同或门表示的异或门也有两种 AIG 结构。

图 3.2 为同或门的四种 AIG 表示形式，其中最顶层的节点称为根节点，根节点下面的两个扇入节点分别为根节点的左子节点和右子节点。

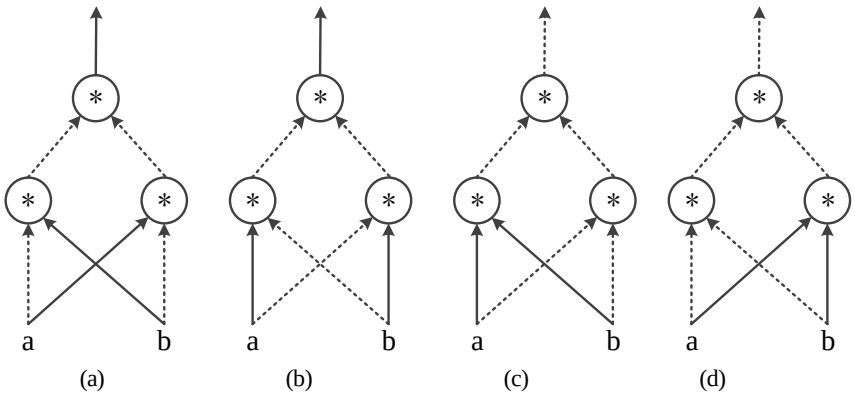


图 3.2 同或门的 4 种 AIG 表示形式: (a) $g=\bar{f}=\overline{\bar{a}b+a\bar{b}}$ (b) $g=\bar{f}=\overline{a\bar{b}+\bar{a}b}$ (c) $g=ab+\bar{a}\bar{b}$ (d) $g=\bar{a}\bar{b}+ab$

Fig. 3.2 four AIG representations of XNOR: (a) $g=\bar{f}=\overline{\bar{a}b+a\bar{b}}$ (b) $g=\bar{f}=\overline{a\bar{b}+\bar{a}b}$ (c) $g=ab+\bar{a}\bar{b}$ (d) $g=\bar{a}\bar{b}+ab$

搜索 AIG 中的异或结构，就是在搜索图 3.1 和图 3.2 中的八种结构，输入一个电路网表，先把这个网表经过 ABC 优化转化成 AIG 形式，保证每个节点的唯一性，通过遍历每个节点，判断是否符合异或结构，如果符合，就在它的根节点上做一个标记。

3.3.3 构建异或节点

由于上面只是简单的搜索出来异或结构并没有进行逻辑函数的优化，为了实现逻辑函数的优化，提出构建一个二输入的异或节点，用它来代替上述结构，并把它加入到 AIG 的哈希表中，保证该异或节点的唯一性，在构建异或节点的同时，考虑其两个扇入的特殊情况，下面是构建该异或节点的算法：

Step1：创建一个二输入的异或节点，两个扇入节点分别为 a，b。

Step1.1 判断如果第一个扇入节点 a 为常量 0：是，返回输入变量 b 节点；否，继续 Step1.2；

Step1.2 判断如果第二个扇入节点 b 为常量 0：是，返回输入变量 a 节点；否，继续 Step1.3；

Step1.3 判断如果第一个扇入节点 a 为常量 1：是，返回输入变量 b 的反变量；否，继续 Step1.4；

Step1.4 判断如果第一个扇入节点 b 为常量 1：是，返回输入变量 a 的反变量；否，继续 Step1.5；

Step1.5 判断如果第一个扇入节点 a 为第二个扇入节点 b：是，返回异或节点为常量 0；否，继续 Step1.6；

Step1.6 判断如果第一个扇入节点 a 为第二个扇入节点 b 的补：是，返回异或节点为常量 1；否，继续 Step2；

Step2：搜索 AIG 哈希表中是否含有扇入为 a 节点和 b 节点的根节点；

Step2.1 如果为空，是，创建一个以 a 节点和 b 节点为扇入的异或节点；否，继续 Step3；

Step3：返回这个新创建的异或门节点；

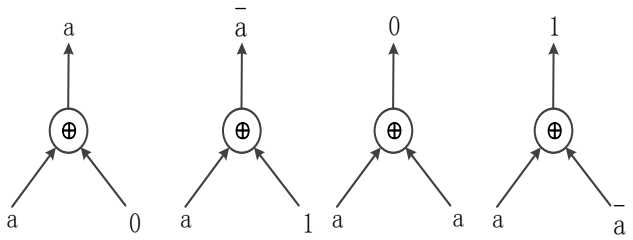


图 3.3 XOR 节点优化
Fig. 3.3 Optimized XOR

考虑新构建的异或门节点是否存在其扇入为 0 或 1 的常量节点，或者两个扇入节点相等和相反的情况，如图 3.3 所示。用构建的新的二输入异或门节点来代替 AIG 中的异或门结构，既节省了内存空间，又减少了电路的面积。

3.4 算法流程及演示例子

根据以上三个方面的讨论，提出的逻辑优化算法如下：

输入：电路的网表格式文件为 PLA 或 BLIF 格式。

输出：电路的网表格式文件为 BLIF 格式。

Step1. 将逻辑函数表示成 AIG 的形式，通过 ABC 的 resyn2 命令进行逻辑函数的化简，保证逻辑函数的 AIG 结构节点的唯一性，同时生成化简后的 AIG 形式。

Step2. 搜索异或门结构。通过遍历 AIG 图中每个与门节点，找到满足 XOR/XNOR 的结构。如图 3.4 所示为搜索异或门的整个流程图。

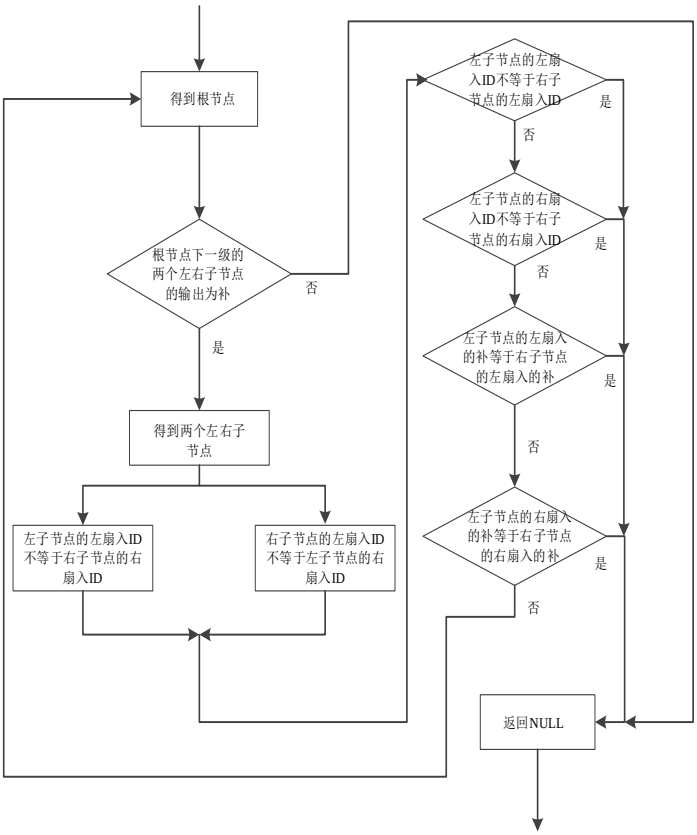


图 3.4 搜索异或门的流程图

Fig. 3.4 search for XOR flowchart

Step3. 判断是否满足 XOR/XNOR 的结构：不是，则跳到 step2 遍历下一个节点；是，则得到其左右子节点。

Step4. 判断两个左右子节点的扇出是否均为 1：是，则跳到 step5；否 跳到 step2 继续遍历下一个节点；

Step5. 构建一个新的异或门节点。用两个扇入构建新的二输入异或门节点。两个扇入节点分别为 a，b。如图 3.5 为构建异或节点的流程图。

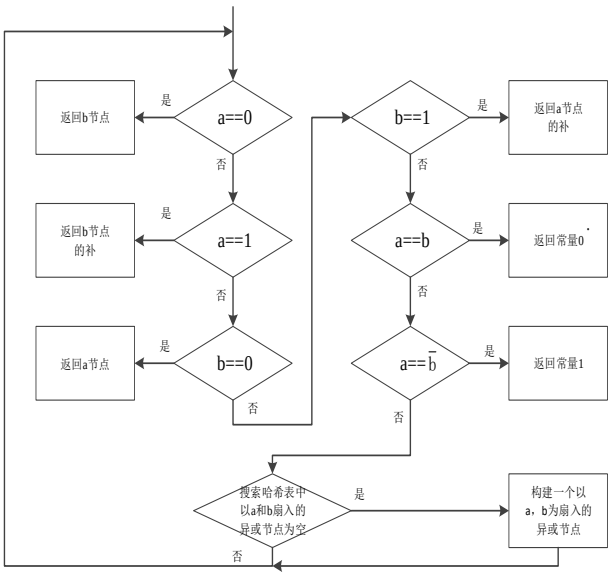


图 3.5 构建异或节点流程图

Fig. 3.5 construct XOR node flowchart

Step6. 生成优化后得 BLIF 网表格式，并输出该网表格式。程序结束。

以 benchmark 电路 xor5.blif 为例，图 3.6(a)为该电路的 AIG 形式。采用逻辑综合优化工具 ABC 进行布尔函数的分解，得到未优化的 AIG 图，通过使用结构哈希命令 resyn2，使得 xor5 电路由层级网表格式转换成每个与门节点均具有唯一扇入对的 AIG 形式，保证了图中每个与门节点扇入对的唯一性。

如图 3.6(a)所示，从逻辑函数的原始输出即整个图形的根节点开始按照从上往下的搜索方法，图中所有的与门节点均储存在哈希表中，并用数字表示，即每个与门节点的 ID，通过遍历每个与门节点，探测每个与门节点是否符合上述异或门和同或门的 8 种结构，进行结构匹配。从节点 18 开始遍历，探测到节点组(18，17，16)为一组的三个节点符合上述结构 g，即为同或门 $g = n9 * n15 + \overline{n9} * \overline{n15}$ 的结构，满足异或门的条件，因此构建一个二输入异或门节点来代替这样的结构。

如图 3.6(b)所示，又因为该结构是同或门，所以用一个二输入异或门节点的补来代替，如图 3.6(b)的根节点，这样就可以节省 2 个与门节点，对电路的面积和功耗都有一定的优化。之后继续探测下面的节点，同样搜索到节点组(9，7，8)满足图 3.1 和图 3.2 的结构(c)，即为异或门 $f = \bar{g} = \overline{ea + \bar{e}a}$ 的结构，所以直接可以替换成二输入异或门节点代替。节点组(15，13，14)和(12，11，10)分别满足上述图 3.1 和图 3.2 的(b)和(d)，即 $f = d * \overline{n12} + \bar{d} * n12$ 和 $f = \bar{g} = \overline{\bar{b}c + bc}$，直接转换成异或门代替，最终实现逻辑函数的优化，如图 3.6(b)所示。由于 xor5 电路比较特殊，它

是一个 5 变量的逻辑函数，可以完全由 XOR/NOT 运算实现。对于异或逻辑密集型电路，采用这种优化方法，更有利于 RML 和双逻辑的优化。

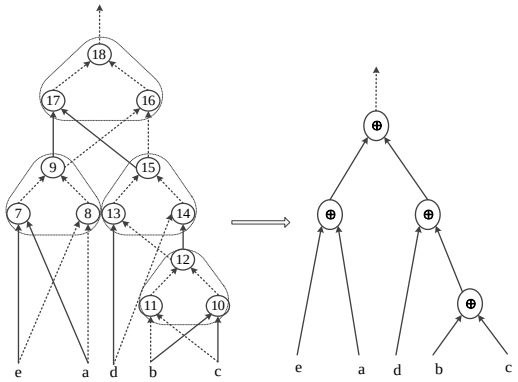


图 3.6 算法演示例子：(a) xor5 的 AIG 形式 (b) 优化后的 xor5 电路

Fig.3.6 xor5.blif example:(a) AIG of the xor5 (b) optimized xor5

3.5 实验结果及分析

本章的算法已用 C 语言实现，并在操作系统为 Ubuntu 14.04，CPU 时钟频率为 3.30GHz，内存为 8GB 的 PC 机上，对 MCNC 标准电路进行测试，实验结果如表 3.1 所示，其中“in”表示测试电路的输入，“out”表示测试电路的输出，“AND nodes”表示测试电路在逻辑综合工具 ABC 优化下生成的与门节点数，所提出的方法中“AND”表示测试电路在实现基于 AIG 的异或门探测算法后所生成的与门节点数，“XOR”表示测试电路在实现算法后所生成的异或门节点数，“total”表示测试电路在实现算法后所生成的总节点数，最后一列的“Xor rate”表示所生成的新图中异或门节点数占优化前 AIG 图中的与门节点数，即与门节点的转化率。与门节点的转化率越高，说明新生成的图中异或门占总节点数越多，反之亦然，从转化率也可以看出适于用 RML 表示的逻辑函数和适于 TBL 表示的逻辑函数。

表 3.1 异或门探测结果

Tab.3.1 The experiment results with search XOR

Circuits	in	out	ABC	Proposed			Xor rate
			AND nodes	AND	XOR	total	
xor5	5	1	12	0	4	4	0.333
t481	16	1	25	10	5	15	0.200
cm82a	5	3	16	10	2	12	0.125

parity	16	1	45	0	15	15	0.333
c499	41	32	387	75	104	179	0.268
c2670	233	140	565	421	49	470	0.086
dalu	75	16	1085	1021	24	1045	0.022
i10	257	224	1816	1554	99	1653	0.054
c6288	32	32	1870	1004	433	1437	0.231
c7552	207	108	1409	830	202	1032	0.143
term1	34	10	89	77	5	82	0.056
cm150a	21	1	46	46	0	46	0.000

从表 3.1 中可以看出，测试电路 parity、xor5 在优化后，生成的新图中与门节点数为零，说明这两个电路在优化前的 AIG 图中的所有与门节点全部转换为新生成的异或门节点，即它们适于用 RML 来实现。如电路 t481、c499、c6288 在优化后，多数与门节点也转化为了异或门节点，转换率相对比较高，即它们适合用 TBL 和 RML 同时实现。而电路 cm150a，在优化后新生成的图中异或门节点的数量为零，说明这个电路并不适合用 RML 来实现，用单一的 TBL 来实现会得到更好的优化效果。从实验结果来看，该方法相比逻辑综合工具 ABC，对于异或门密集型的电路有很大改善，对于异或门较少的电路，只能通过传统的 TBL 进行优化，但大部分电路还是适合用双逻辑来实现的。

以上测试的结果和文献[31]给出的结果保持一致性，相比于其他文献，本章的探测算法是基于 AIG 实现的，为后期逻辑函数工艺映射以及电路的验证方向提供了有效的表示方法，这也体现了该算法的优越性。

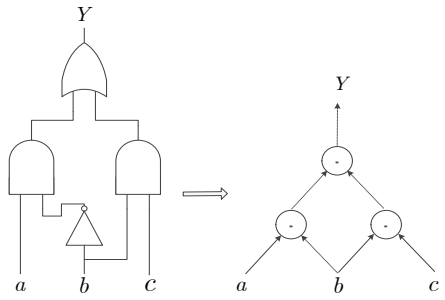
3.6 本章小结

本章从 AIG 的异或门结构出发，给出在 AIG 图上所有符合条件的异或门和同或门的结构，根据异或门在 AIG 中表示形式不同，提出了基于 AIG 的异或门结构探测方法。在探测的过程中，通过构建一种新的二输入异或门节点来代替 AIG 中的异或门结构，生成一种基于 AND/XOR/NOT 运算的双逻辑图形表示方法。这种图形结构具有表示简单、占用空间少、易于映射等优点，因此更有利于进行双逻辑综合与优化的研究。由于逻辑优化还包含后期与工艺相关的映射，而该双逻辑图是门级表示结构，更有利于后期的电路映射，为下一步基于双逻辑的图形优化工作奠定了基础。

4 基于双逻辑的面积优化技术

4.1 理论基础

通过图形可以用 AIG 表示任何逻辑函数。图 4.1(a)为逻辑函数 $Y=(a\bar{b})+(bc)$ 的电路图，图 4.1(b)为其 AIG 表示。



(a) (b)

图 4.1 $Y=(a\bar{b})+(bc)$ 的 AIG 表示：(a) 电路图 (b) AIG 表示

Fig. 4.1 The AIG of the function $Y=(a\bar{b})+(bc)$：(a) Circuit diagram (b) Corresponding AIG

通常根据图 4.1(b)中 AIG 节点的数目可以衡量该电路的大小，计算原始输入到原始输出的最长路径来衡量该电路的延时。图 4.1(b)中每个与门节点从下往上都有相应的层级数和相应的名称 ID，ID 用于节点间拓扑结构排序，节点的层级数从连接原始输入节点的非原始输入节点数算起，至连接原始输出节点的非原始输出节点数为止，因此图 4.1(b)中电路的大小为 3，层级数为 2。

AIG 最初用于解决逻辑函数的验证问题[32]，用其表示逻辑函数，具有结构简单、易于延展和计算等特点，一些基本的电路结构如或非图（OR-Inverter Graph，OIG），与非/非图（NAND-Inverter Graph，NIG）[33]均是基于 AIG 结构演变而来。由于 AIG 的这些特点，近年来研究者们探索将其用于逻辑综合和优化的研究中[34]。文献[16]提出了一种基于 AIG 逻辑综合方法，通过逻辑函数的 AIG 图形表示，利用图中结构共享在保证不增加延时的基础上达到减少电路面积的目的。而由于其前期逻辑优化方法应用在不大于 4 输入的电路上，优化的电路大小有限；文献[35]在其基础上拓展至 5 输入电路的应用，并和文献[16]结合起来运用，对于电路面积的优化得到更好的效果。在工艺映射过程中，通过引入割集对 AIG 进行结构划分，从输入节点到输出节点，对每个内部节点计算出 K-可行性割集，选择局部最优的割集进行工艺映射[36]，文献[37]对此进行了改进，在 AIG 图中不仅从输入节点到输出节点进行结构的划分，又从输出节点到输入节点重新映射，通过列举出 KL-可行性割集，进行逻辑单元的重映射，达到面积和延时优化的目的。

事实上，逻辑函数也可由基于与、异或运算的所谓的 RML 来表示，同时也可由相应的逻辑门来实现。研究表明，不同的函数采用不同的逻辑实现，其优化结果各异。因此，提出采用 TBL 和 RML 的双逻辑优化的思想[25]。AIG 是逻辑函数基于 TBL 的门级图形表示，类似地，可以构建逻辑函数基于 RML 的门级图形表示。而要在逻辑函数的门级图形表示实现双逻辑优化，需要构建一种逻辑函数的双逻辑图形表示方法。AIG 为构建这种双逻辑门级图形表示并实现图形优化提供了基础。

以函数 $f(x_1,x_2,x_3)=x_1\overline{x_2}x_3+x_1x_2\overline{x_3}$ 为例，其 AIG 门级表示如图 4.2 所示。从图中计算每个节点的逻辑函数可以得出变量 x_2 和 x_3 所构成的 AIG 子逻辑网络满足 XOR 结构，而该结构在 AIG 图中需用三个与门节点和 5 个反相器，若用一个异或节点表示 XOR 结构，化简函数得到 $f(x_1,x_2,x_3)=x_1(x_2\oplus x_3)$ 比单一用 TBL 表示文字数更少，节点数也会减少，实现图压缩的目的，从而潜在的也减少电路延时。

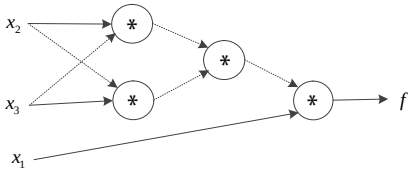


图 4.2 $f(x_1,x_2,x_3)=x_1\overline{x_2}x_3+x_1x_2\overline{x_3}$ 的 AIG 表示

Fig. 4.2 The AIG of the function $f(x_1,x_2,x_3)=x_1\overline{x_2}x_3+x_1x_2\overline{x_3}$

在 ABC 进行单元映射过程中，观察表明其生成的门级网表仍然存在可优化的结构。如图 4.3 所示。

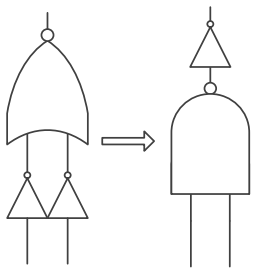


图 4.3 反相器最小化

Fig. 4.3 Minimize Inverters

通过优化映射后电路中的反相器可以进一步对电路面积优化，达到面积最小化的目的。

4.2 问题定义及算法思想

4.2.1 问题定义

所要解决的问题可描述为：给定一个 n 变量的逻辑函数 f，将其表示成只含有与门和异或门节点构成的 AXIG，实现由非门、与非门、或非门、异或门和同或门构成的门级电路结构，达到电路面积优化的目的。

4.2.2 算法思想

首先基于 AIG 进行逻辑优化，得到化简的 AIG；然后，进行 XOR 门的探测，根据图中 XOR 结构的类型不同，构建一种能同时表示双逻辑的图形结构 AXIG；其次，通过探测图形中最佳映射的门级结构进行逻辑映射；最后实现以晶体管数作为基准的电路面积优化。

4.3 基于双逻辑的门级图形表示

4.3.1 AXIG 的构建

为了在一个数据结构中能同时表示 TBL 和 RML，在 AIG 的基础上通过探测 XOR 结构，提出一种 AXIG 结构表示双逻辑函数。AXIG 是一种特殊的 DAG。在 AXIG 内部只包含二输入的与门节点和异或门节点，终端节点是没有输入边的原始输入节点和只有一个输入边没有输出边的原始输出节点。其中边连接两个节点，若节点 i 的扇出是节点 j 的扇入，则存在实线边(i，j)；若节点 i 的扇出取反后是节点 j 的扇入，则存在虚线边($\bar{i}$，j)；一般地，称连接原始输出的节点为根节点或者局部 AXIG 网络的输出节点为根节点。

以函数 $f(a,b,c)=(a+b)\oplus c$ 为例，通过对或运算应用 De Morgan 定理，实现由与门和异或门组成的门级图形，如图 4.4 所示。

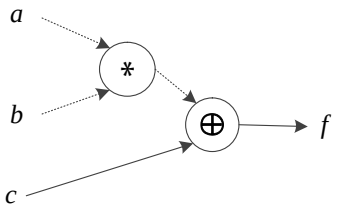


图 4.4 $f(a,b,c)=(a+b)\oplus c$ 的 AXIG 表示

Fig. 4.4 The AXIG of the function $f(a,b,c)=(a+b)\oplus c$

图 4.4 中“*”表示与门节点，“⊕”表示异或门节点，而由于 $a+b=\overline{\bar{a}\bar{b}}$，因此得到含有与门和异或门以及边所表示的反相器组成的 AXIG 图。其中虚线为

变量或逻辑函数取反的值，而实线表示变量或逻辑函数的值。由于图中含有与门、异或门以及反相器，因此任意逻辑函数均可由 AXIG 图实现。

4.3.2 逻辑函数的双逻辑门级表示

AXIG 的构建是在 AIG 的基础上完成的。根据异或门在 AIG 中表示的不同结构进行探测，并用异或门类型节点替代，达到减少图中节点数的目的，并实现逻辑函数基于双逻辑的图形表示。

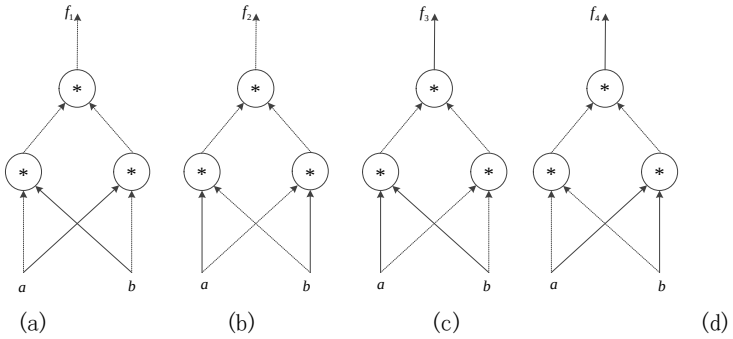
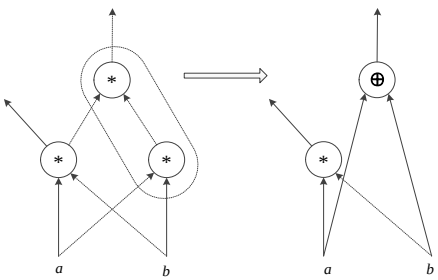


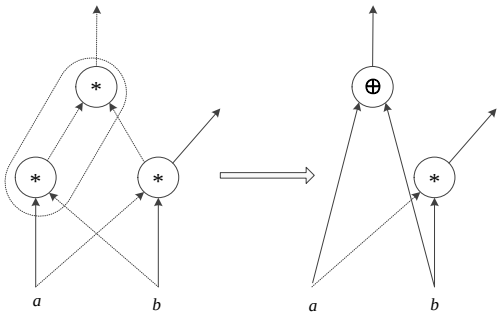
图 4.5 探测 XOR 结构: (a) $f_1 = \bar{a}b + a\bar{b}$ (b) $f_2 = a\bar{b} + \bar{a}b$
(c) $f_3 = \overline{ab + \bar{a}\bar{b}}$ (d) $f_4 = \overline{\bar{a}\bar{b} + ab}$

Fig. 4.5 Different XOR structure: (a) Function $f_1 = \bar{a}b + a\bar{b}$ (b) Function $f_2 = a\bar{b} + \bar{a}b$
(b)Function $f_3 = \overline{ab + \bar{a}\bar{b}}$ (d) Function $f_4 = \overline{\bar{a}\bar{b} + ab}$

图 4.5 是异或门在 AIG 中的四种不同表示结构，其中图 4.5(a)-(b)为 AIG 中异或门的基本结构，图 4.5(c)-(d)是通过 De Morgan 定理派生而来的异或结构。计算图中每种异或门结构的反相器可知，图 4.5(c)-(d)仅需 4 个反相器，而图 4.5(a)-(b)则需 5 个反相器，因此采用后两种结构表示异或门可实现反相器数量的减少达到 AIG 优化。在 AIG 图中探测异或结构，不仅仅存在图 4.5 所示的异或结构，而且还存在异或结构根节点下左右两个子节点扇出大于等于 2 的称之为共享节点结构。



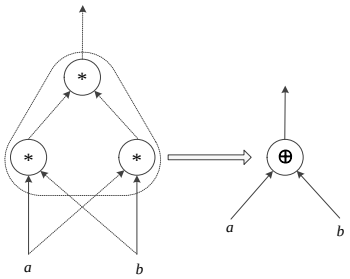
(a) 右子节点共享的 AIG (b) 生成的 AXIG 表示
(a) Right node shared (b) Derived AXIG



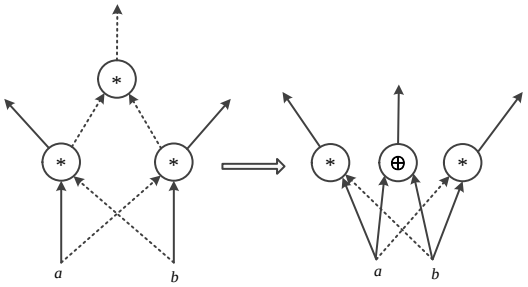
(b) 左子节点共享的 AIG (d) 化简的 AXIG 表示
(b) Left node shared (d) Optimized AXIG
图 4.6 单子节点共享型
Fig. 4.6 Single node shared

对于异或结构，根节点的共享不会影响探测异或结构，而其左右两个子节点的共享性为探测异或结构，构建异或门节点提供了判断条件。通过判断异或结构根节点下左右子节点的共享性来判断是否构建异或门节点，可分为以下三种情况：

Type I: 基本型，即左右子节点均不是共享节点，如图 4.7 所示。



(a) 无共享节点的 AIG (b) AXIG 中的异或节点
(a) AIG without shared nodes (b) XOR node in AXIG
图 4.7 图 4.5(b)的 AIG 及其 AXIG 表示
Fig. 4.7 The XOR structure of Fig. 4.5(b)



(a) 双子共享节点的 AIG (b) 优化的 AXIG
(a) AIG with shared double nodes (b) Reduced AXIG
图 4.8 双子节点共享型
Fig. 4.8 Shared double nodes

图 4.7 给出了以图 4.5(b)为例的异或结构优化，得到一个异或门节点，即将图 4.7(a)中 AIG 的三个与门节点合为一个异或门节点，实现图的压缩，逻辑函数的优化。

Type II：单子节点共享型，如图 4.6 所示。

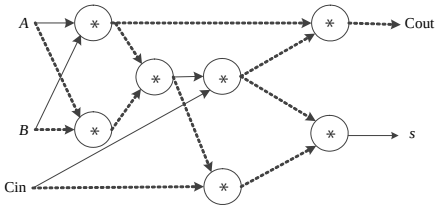
通过判断异或结构中根节点下左右子节点的扇出数目，如果左子节点为共享节点，如图 4.6(a)所示，则保留当前子节点，合并根节点与右子节点为异或节点，如图 4.6(a)虚线圈中两个与门节点转换为如图 4.6(b)所示的 AXIG 结构；同理，如果右子节点是共享节点如图 4.6(c)所示，则可合并虚线圈中根节点与左子节点为异或节点，构建 AXIG 如图 4.6(d)所示。

比较图 4.6 中 AIG 与 AXIG 的节点数可知，通过 AXIG 的构建可节省 1 个与门节点和 4 个反相器。

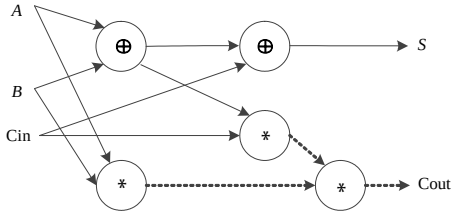
Type III：双子节点共享型，如图 4.8 所示。

图 4.8(a)表示异或结构左右子节点均为共享节点的情况，如果进行异或门节点的构建，如图 4.8(b)所示。与图 4.8(a)相比，虽然优化了 3 个反相器，但增加了门的扇入，并且直接用异或节点代替并不比单一的 TBL 表示更优化，因此图 4.8(a)这种异或结构不进行异或节点的构建。

对给定的 AIG 进行双逻辑优化，可实现电路面积的最小化。以全加器为例，图 4.9 为全加器电路的两种图形表示。从图中可以看出，较之于图 4.9(a)的 AIG 表示，图 4.9(b)的 AXIG 节点数减少 2 个，层级数也减少 1 层。



(a) 全加器电路 AIG 表示
(a) The AIG of the full adder circuit



(b) 全加器电路 AXIG 表示
(b) The AXIG of the full adder circuit

图 4.9 全加器的 AIG 与 AXIG 表示
Fig. 4.9 The AIG and AXIG of the full adder circuit

4.3.3 基于双逻辑的映射

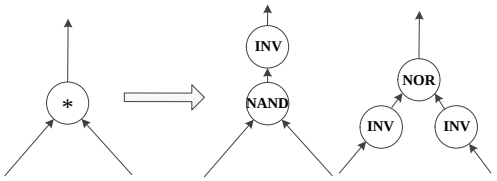
根据 AXIG 图中每个节点扇入扇出的极性不同以及异或运算非号存在如式 4.1 所示伸缩性质：

$$\overline{x_1 \oplus x_2 \oplus x_3} = \overline{x_1 \oplus x_2} \oplus x_3 = \overline{x_1} \oplus x_2 \oplus x_3 = x_1 \oplus \overline{x_2} \oplus x_3 = x_1 \oplus x_2 \oplus \overline{x_3} \qquad \text{（公式 4.1）}$$

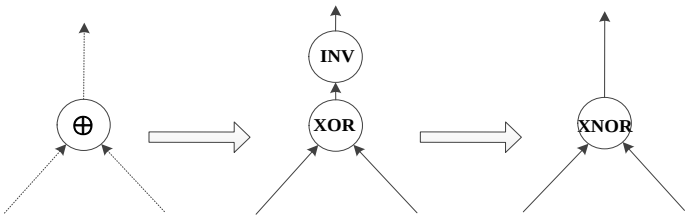
式中，x_1、x_2、x_3分别为逻辑函数中的三个输入变量。

在工艺映射过程中，结合标准单元库中门单元权值信息进行单元映射。由于不同的图结构可以根据标准单元映射成不同的门级结构，而为了实现面积优化的目的，因此对图中每种结构选择最优的映射结果进行图形匹配。按照 AXIG 图中每个节点的扇入/扇出极性不同，分为以下三种情形讨论：

Case I：门节点的扇入/扇出极性相同，如图 4.10 所示。



(a) 与门扇入/扇出极性一致
(a) AND node fanin/fanout with the same polarity

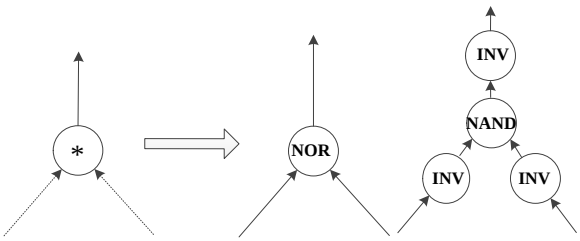


(b) 异或门扇入/扇出极性一致
(b) XOR node fanin/fanout with the same polarity
图 4.10 门节点扇入/扇出极性一致
Fig. 4.10 Node fanin and fanout with the same polarity

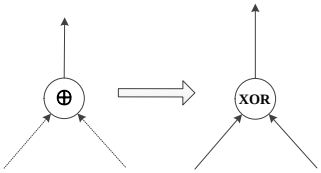
由图 4.10(a)可以看出，一个基本的与门节点可以映射成两种类型门级结构：与非门和反相器；或非门和反相器。据此，可根据单元库门级的权值信息计算出每种门级结构的晶体管数。如图 4.10(a)中由与非门和反相器组成的结构所需晶体管数为 6，而由或非门和反相器组成的与门晶体管数为 8。因此在 AXIG 中，由计算出的晶体管数可知图 4.10(a)中与门节点选择映射成与非门和反向器的结构更易于面积优化。而图 4.10(b)中异或门的扇入扇出极性均为反，则异或门节点根据 De Morgan 定理转换为一个异或门级联一个反相器的结构，而

这种结构又可以用同或门代替，因此图 4.10(b)可直接映射成一个同或门表示，而不需要反相器的存在，这样也可以通过减少反相器的数量达到优化面积的目的。

Case II：门节点扇入/扇出极性不同时，如图 4.11 所示。



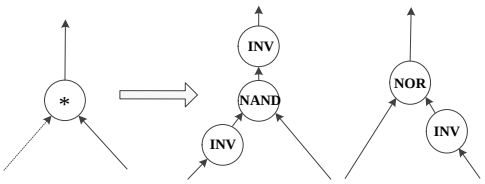
(a) 与门扇入为反变量扇出为原变量
(a) AND node with negative fanin and positive fanout



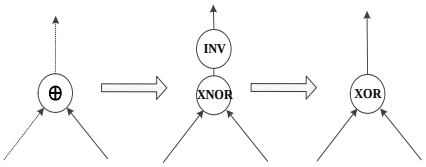
(b) 异或门扇入为反变量扇出为原变量
(b) XOR node with negative fanin and positive fanout
图 4.11 门节点扇入/扇出极性相反
Fig. 4.11 Fanin and fanout with opposite polarity

图 4.11 给出了当节点扇入和扇出极性不同时的映射。图 4.11(a)为 AXIG 图中与节点的派生节点，其两个扇入节点的极性均为反，而扇出节点极性为正。通过与工艺相关的映射，可得到图 4.11(a)右边的两种不同逻辑结构，从图中可以明显得出映射成或非门是最优的。图 4.11(b)是一个异或节点，通过 De Morgan 定理可以将其化简为一个异或门，因此选择映射成一个异或门更优。相反的，对于一个异或节点，其两个扇入均为正极性，扇出为反极性时，可以直接映射成同或门。

Case III：门节点一个扇入和扇出变量的极性相同，如图 4.12 所示。



(a) 与门节点一个扇入与扇出变量极性相同
(a) One of fanins and fanouts have the same polarity in AND node.



(b) 异或门节点一个扇入与扇出变量极性相同
(b) One of fanins and fanouts have the same polarity in XOR node
图 4.12 门节点的一个扇入与扇出变量同极性
Fig. 4.12 One of the fanins and fanouts have the same polarity in the gate

当节点其中一个扇入变量和扇出变量的极性保持一致时，如图 4.12(a)所示的与门节点选择映射成或非门与反相器结合的方式更适于面积优化；而图 4.12(b)的异或门节点根据式 1 反相器的增减选择映射成一个异或门的方式可以达到优化面积的目的。从上述的三种类型来看，对于异或门节点，不管异或门节点的扇入或扇出的极性如何变化，其最终总可以映射为一个异或门或者是同或门而不包含任何反相器。

4.4 算法流程及演示例子

根据以上讨论，提出面积优化算法如下：

Step1. 将逻辑函数表示成 AIG。

Step2. 找出 AIG 中符合异或结构的根节点(如图 4.5)，并标记根节点为异或结点。

Step3. 探测 AIG 中异或结构。遍历 AIG 中每个标记的异或节点。

Step3.1. 判断 XOR 结构是否满足 Type III：是，则将去掉 XOR 结构的异或节点标记；否，继续 Step3.2.

Step3.2. 判断 XOR 结构是否满足 Type II(a)左子节点扇出数大于等于 2，右子节点扇出数等于 1：是，合并根节点与右子节点；否，继续 Step 3.3.

Step3.3. 判断 XOR 结构是否满足 Type II(c)右子节点扇出数大于等于 2，左子节点扇出数等于 1：是，合并根节点与左子节点；否，继续 Step3.4.

Step3.4 XOR 结构满足 Type I 合并根节点与其左右子节点，构建异或类型节点. 返回 Step3，继续遍历下个已标记的异或节点，直到遍历完为止.

Step4. 与工艺相关的映射.遍历 AXIG 中每个节点。

Step4.1 判断该节点是否为异或节点：是，继续 Step4.2；否，跳到 Step4.1.1.

Step4.1.1 判断该节点是否满足映射过程中 Case II(a)：是，计算映射的门级权值，选择权值小的结构进行映射，遍历下一个节点；否，继续 Step4.1.2.

Step4.1.2 判断该节点是否满足映射过程中 Case III(a)：是，计算映射的门级权值，选择权值小的结构进行映射，遍历下一个节点；否，继续 Step4.1.3.

Step4.1.3 判断该节点是否满足映射过程中 Case I(a)：是，计算映射的门级权

值，选择权值小的结构进行映射，遍历下一个节点；否，跳到 Step4.

Step4.2 判断该异或节点是否满足映射过程中 Case I(b)和 Case II(b)：是，进行门级映射；否，继续 Step4.3.

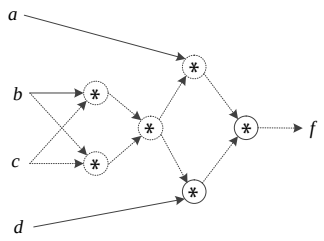
Step4.3 判断该异或节点是否满足映射过程中 Case III(b)：是，进行门级映射；否，返回到 Step4.

Step5. 生成优化后的门级 blif 格式.

Step6. 通过 ABC 的 cec 命令验证优化后的门级网表与输入网表功能是否相同：是，继续 Step7； 否，输出错误信息.

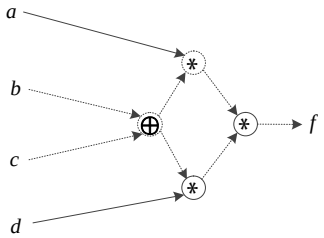
Step7. 输出门级网表，以及网表中每种门的个数，程序结束.

以逻辑函数 $f = a(b \oplus c) + (b \mathrm{e} c)d$ 为例，说明整个算法的流程。图 4.13(a)为逻辑函数 f 的 AIG 表示，计算图中每个节点的逻辑函数，探测到图 4.13(a)中变量 b 和变量 c 符合图 4.5(c)的异或结构。由于变量 b 和变量 c 所构成的异或结构只有根节点为共享节点，其左右子节点均不是共享节点，而上文中提到根节点共享不影响构建异或节点，因此构建以变量 b 和变量 c 为扇入的异或结点，保留其他节点不变，生成逻辑函数 f 的 AXIG 表示，如图 4.13(b)所示。 在工艺映射过程中，以基本门和复合门组成的单元库为基准得到电路实现。根据图 4.13(b)中节点的 ID 排序，从原始输入节点遍历到原始输出节点，判断图中每个节点的扇入扇出极性，得到逻辑函数的电路实现如图 4.13(c)所示。逻辑函数 f 最终由 1 个异或门、3 个与非门和 1 个反相器实现，比用 ABC 实现减少了 2 个晶体管，实现了电路面积的优化。



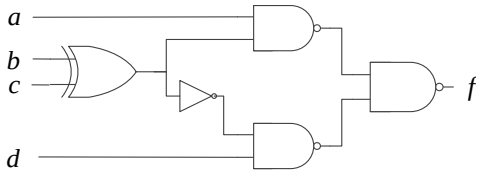
(a) $f = a(b \oplus c) + (b \mathrm{e} c)d$ 的 AIG 表示

(a) The AIG of the function $f = a(b \oplus c) + (b \mathrm{e} c)d$



(b) $f = a(b \oplus c) + (b \mathrm{e} c)d$ 的 AXIG 表示

(b) The AXIG of the function $f = a(b \oplus c) + (b \mathrm{e} c)d$



(c) $f = a(b \oplus c) + (b \odot c)d$ 的门级电路

(c) The circuit of the function $f = a(b \oplus c) + (b \odot c)d$

图 4.13 $f = a(b \oplus c) + (b \odot c)d$ 的优化流程

Fig. 4.13 The optimization process of the function $f = a(b \oplus c) + (b \odot c)d$

4.5 实验结果及分析

所提出的算法用 C 语言实现，并在操作系统为 Ubuntu 14.04，CPU 时钟频率为 3.30GHz，内存为 8GB 的 PC 机上应用于 MCNC 标准电路进行测试，所用的标准单元库是 MCNC.genlib。实验结果如表 4.1 所示，其中“in”表示测试电路的输入个数，“out”表示测试电路的输出个数，“nodes”分为四部分组成，其中“ABC”是测试电路在逻辑综合工具 ABC 优化下生成的与门节点数，“[38]”表示文献[38]算法的结果，是只含有与门和反相器生成的节点数。“[39]”表示文献[39]算法的结果，是不考虑 AIG 异或结构子节点共享情况下进行优化生成与门和异或门的结果，“AXIG”是本章构建 AXIG 图所生成的节点数。由表 4.1 可看出文献[39]和“AXIG”的节点数明显少于 ABC 和文献[38]的节点数，这是因为本章和文献[39]引入了 XOR 结构，证明了电路采用双逻辑实现更有利于提升电路的性能。而在表中最后三列，是计算本章节点数分别与 ABC、文献[38]以及文献[39]改进的百分比，从表中可知大部分电路所得百分比为负，说明本章构建的 AXIG 节点数相较于 ABC、文献[38]和文献[39]明显减少。而其中对于电路 C432 而言，经过 ABC 优化后电路中不存在异或结构，因此节点数没有改变，说明 C432 电路采用 TBL 表示更利于其优化。而对于电路 C499，在构建 AXIG 过程中由于不存在异或结构子节点共享类型，因此和文献[39]的结果持平，但是对于大部分存在 XOR 的电路，仍然具有一定数量的异或结构子节点共享类型进行图形优化，因此本章在减少节点数方面具有一定的优势。

表 4.1 节点数的比较

Tab. 4.1 Comparison of node number

测试电路	输入	输出	节点数				百分比		
			ABC	[38]	[39]	AXIG	ABC	[38]	[39]
C499	41	32	387	386	179	179	-53.75%	-53.63%	0.00%
C2670	233	140	565	534	493	470	-16.81%	-11.99%	-4.67%

C6288	32	32	1 870	1 870	1 870	1 437	-23.16%	-23.16%	-23.16%
C7552	207	108	1 409	1 377	1 118	1 032	-26.76%	-25.05%	-7.69%
C1355	41	32	387	390	182	179	-53.75%	-54.10%	-1.65%
C1908	33	25	361	354	231	227	-37.12%	-35.88%	-1.73%
C432	36	7	127	127	127	127	0.00%	0.00%	0.00%
C880	60	26	305	306	278	269	-11.80%	-12.09%	-3.24%
C3540	50	22	935	918	887	871	-6.84%	-5.12%	-1.80%
i10	257	224	1 816	1 808	1 654	1 653	-8.98%	-8.57%	-0.06%
平均值							-23.90%	-22.96%	-4.40%

考虑到图中节点具有多种类型，实现不同节点所需要的实际电路晶体管数也不一样，因此用节点数来评估显得不太客观。为了更加准确的评估电路中的面积，因此本章采用晶体管数作为衡量电路面积的基准。

表 4.2 给出了晶体管数的比较。在表 4.1 所生成的 AXIG 图基础上，进行单元库的映射。本章分别与逻辑综合工具 ABC、文献[38]和文献[39]进行电路晶体管数的比较，其中 ABC 和文献[39]采取同一个单元库进行映射，库中包含 NAND-2、NOR-2、XOR-2、XNOR-2、INV。而文献[38]中采用的单元库只包含了 NAND-2、NOR-2、INV。“AXIG”是本章的方法，用的单元库同 ABC 映射的单元库一致。最后一部分是所得出的晶体管数的比较，其中与 ABC 和文献[38]相比具有一定的优势，而和文献[39]相比只有少许改进，所以需要在映射过程中针对映射算法进一步的研究。而其中电路 C1908、C3540、i10 中含有异或结构子节点共享型比较少，并且这几种电路本身用 TBL 进行优化比转换成 RML 更具有优势，而本章算法主要针对 XOR 集中型电路效果更好。

表 4.2 晶体管数的比较

Tab. 4.2 Comparison of transistor count

测试电路	输入	输出	晶体管数				百分比		
			ABC	［38］	［39］	AXIG	ABC	［38］	［39］
C499	41	32	1 364	1 788	1 358	1 358	-0.44%	-24.05%	0.00%
C2670	233	140	2 496	2 414	2 258	2 247	-9.98%	-6.92%	-0.49%
C6288	32	32	8 484	8 454	8 454	8 420	-0.75%	-0.40%	-0.40%
C7552	207	108	6 054	6 406	5 612	5 566	-8.06%	-13.11%	-0.82%
C1355	41	32	1 368	1 802	1 368	1 368	0.00%	-24.08%	0.00%
C1908	33	25	1 424	1 628	1 350	1 400	-1.69%	-14.00%	3.70%
C432	36	7	638	596	594	570	-10.66%	-4.36%	-4.04%
C880	60	26	1 396	1 394	1 316	1 310	-6.16%	-6.03%	-0.46%

C3540	50	22	4 254	4 038	3 898	3 970	-6.68%	-1.68%	1.85%
i10	257	224	8 254	8 116	7 868	7 874	-4.60%	-2.98%	0.08%
平均值							-4.90%	-9.76%	-0.06%

4.6 本章小结

本章提出了一种逻辑函数基于双逻辑的图形优化算法，通过探测 AIG 图中的异或结构，找出图中所有满足条件的 XOR，构建新的数据结构 AXIG，在新构建的 AXIG 中通过局部逻辑变换进行与工艺相关的映射，实现电路以晶体管数为基准的面积优化。所提出的方法与现有的学术界逻辑综合工具 ABC 相比，节点数平均减少了 23.90%，电路的晶体管数减少了 4.90%。本章主要针对电路的面积进行研究，值得指出的是构建 AXIG 可降低 AIG 的逻辑深度，因此潜在的达到延时优化的目的。

5 基于 AXIG 的面积优化技术

基于新型数据结构 AXIG 的门级图形表示，以及其在后期基于标准单元库进行的逻辑函数映射技术，对于电路面积的最小化实现具有重要意义。针对上一章最后图形的映射问题，本章将继续研究。在前一章的研究过程中，后期的逻辑函数映射是通过 AXIG 图中映射每个与门节点为与非门或者或非门而实现的，在映射过程中，由于必须遍历图中每个与门节点和异或门节点，算法的时间复杂度会相应的增加。因此对于后期的映射技术，本章更进一步的研究了反相器最小化算法。由于时间关系，本章研究的内容有待改进和完善，在这里主要提出一种反相器最小化实现方法的思想，并且结合 AXIG 图进行更进一步的优化。

5.1 理论基础

5.1.1 基于 AIG 的映射方法

本章是基于标准单元库的映射方法。通常情况下基于标准单元库的映射分为以下三个步骤：1)逻辑图的构建，2）匹配，探测图中是否存在某个具体的逻辑结构可以由单元库中的逻辑单元实现。3）选择，选择最优的匹配方式进行电路的优化。而本章逻辑图的构建是基于 AIG 实现的，通过在 AIG 图中进行布尔匹配，在映射中选择合适的结构进行匹配的算法有动态规划的方法等，这样可以从全局出发保证在线性时间复杂度内得到一个优化的结果。Luca 在逻辑优化中提出了一种采用贪心算法解决逻辑映射过程中的函数结构划分和选择的部分[40]，如图 5.1 所示。

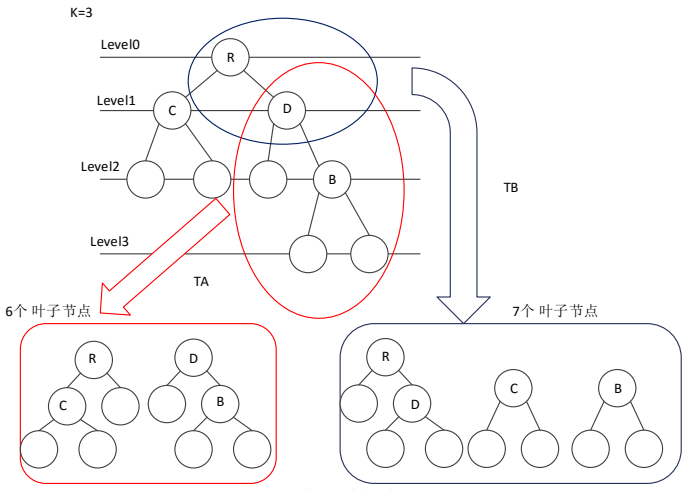


图 5.1 贪心算法分解

Fig. 5.1. Greedy tree decomposition

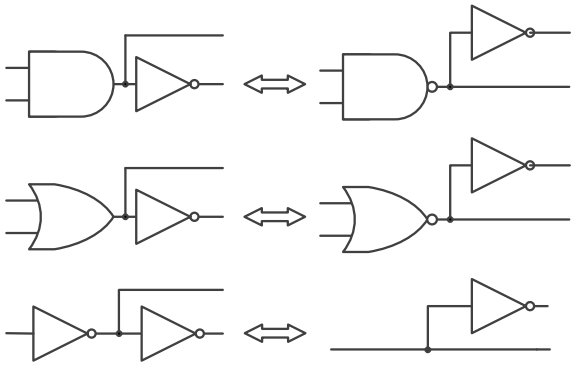
图 5.1 可以看作是逻辑函数 AIG 的表达式，通过贪心算法在逻辑映射过程中，选择最终所得到的叶子节点数最少的划分方法，实现逻辑函数的映射过程。图中节点 R 为该逻辑函数的根节点，并且设定每次切割所得到的叶子节点不超过 3 输入的切割集，即在划分一个 AIG 图时，切割子片段的输入个数最大不能超过 K 个，也叫做 K-可行性割集。图中设定的 K 为 3。从图中可以看出，有两种划分方法分别是 TA 和 TB。其中 TB 为任意划分图中的结构，则最终的结果如 TB 所指向的方框所示，得到的叶子节点数为 7。而采用贪心算法切割 AIG 子图所得结果如图 TA 所示的 6 个叶子节点数。因此使用 TA 划分的算法能得到更好的结果。

而在映射过程中，选择单元库也是逻辑映射进一步优化的关键。我们从前一章得到一种新的数据结构 AXIG，在这种新的数据结构中包含两种不同类型的节点分别与节点和异或节点。而在具体的门实现过程中对于简单门(AND、OR、NAND、NOR)来说可知一个二输入与门采用 CMOS 技术需要 6 个晶体管实现，而一个二输入与非门相比于一个二输入与门少了一个反相器，因此需要 4 个晶体管实现。同样一个二输入或门需要 6 个晶体管实现，而一个二输入或非门需要 4 个晶体管实现。因此在确定具体单元库中的逻辑单元时，相比于二输入的与门和或门，我们选择二输入的与非门和或非门组成单元库中的简单门，又因为在 AXIG 中包含有异或节点，异或节点也可由二输入的与非门和或非门组合排列实现，而为了尽可能的减少电路中的晶体管数，我们引入二输入的异或门和二输入的同或门作为组成要映射的单元库中的复合门。因此我们所要映射的标准单元库已经构成，即 INV,XOR-2,XNOR-2,NAND-2,NOR-2。

5.1.2 极性图映射方法

本章通过特定的单元库实现逻辑函数的映射。在已实现的 AXIG 基础上，完成节点类型的转换。从单元库中可以看出，AXIG 中所有的与节点，不管每个与节点的扇入扇出极性如何，都将映射成由与非门或者或非门分别与反相器组合的形式，因此本章采用了 Alok 提出的极性图方法进行逻辑函数的单元库映射[17]。以下是关于门节点极性转换的理论基础。

所谓极性图，就是根据门扇入扇出添加反相器构成的一种图形结构。实际上改变门的扇入扇出极性，通过在门的扇入扇出边添加一个非门，就能改变门的类型，实现门之间的转换。如下图 5.2 所示，为简单门改变扇出极性。

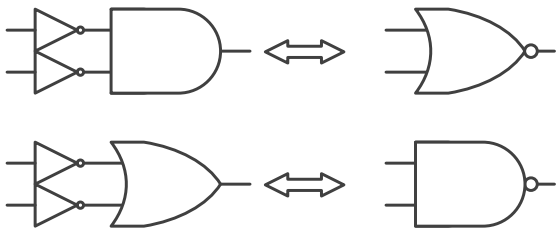


(a) (b)

图 5.2 改变门类型：(a) 插入反相器 (b) 变换门类型

Fig. 5.2 switch simple gate：(a)change fanout (b) change gate

从图 5.2(a)中可以看出，一个与门有两个扇出，其中一个扇出连接一个非门，若要实现与门和与非门之间的转换，可以在保持逻辑功能不变的情况下，非门从与门的一个扇出移到了另一个扇出上面，同时改变与门的类型，如图 5.2(b)所示。同样的一个或门和一个非门，也可以看成一个或非门的实现。这是通过改变门的扇出极性，导致门类型转换。而改变门扇入的极性，门的类型也会受到相应的变化，如下图 5.3 所示，为改变简单门的扇入极性。



(a) (b)

图 5.3 门间转换: (a) 简单门改变扇入极性 (b) 改变门类型

Fig. 5.3 switch gate: (a) change fanin (b) change gate type

从图 5.3(a)中可以看出，一个逻辑与门通过在其两个扇入上分别添加非门，改变与门两个扇入的极性，实现门类型的转化，如图 5.3(b)所示。实际上这一转换运用了德摩根定理中的 $\bar{x}*\bar{y}=\overline{x+y}$ 公式。而改变一个或门的两个扇入极性，通过德摩根定理 $\bar{x}+\bar{y}=\overline{x*y}$ 可得到一个与非门。实际上用或非门和与非门分别代替图中的与门和两个反相器以及或门和两个反相器的过程，实现了逻辑映射过程的更进一步优化。

运用这种改变门的扇入和扇出极性的方法，进行电路的实际化简，如下图 5.4 所示。

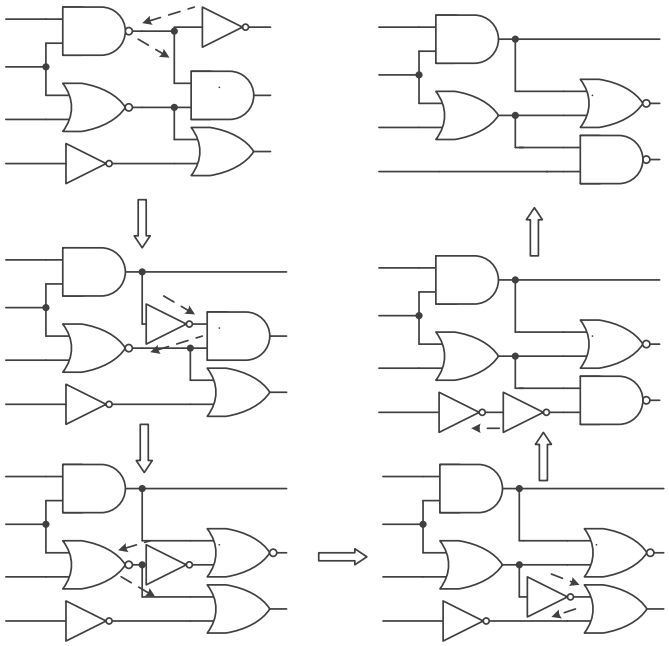


图 5.4 减少反相器
Fig. 5.4 minimized inverters

从图 5.4 中可以看出，从最初的 NAND 的扇出开始，将 NAND 连接的 NOT 推进 NAND 中，因此 NAND 也相应变成了 AND，而此时在保证整个电路逻辑功能不变的情况下，在 AND 的另一扇出添加一个 NOT 来保持逻辑功能的平衡。类似的接着将这个 NOT 和它的扇出 AND 结合，又由于要改变 AND 的双扇入极性，因此默认添加一个反相器来抵消或非门扇入的反，接着继续 NOT 的结合，通过将 NOT 反复推到逻辑门的扇入或者扇出，最终实现了电路中不存在反相器的电路。将初始电路和最终生成的电路对比，由于最终电路中不包含反相器，因此相比于初始电路晶体管数减少了四个。但并不是所有的电路都能将反相器通过这种方式进行化简，如下图 5.5 所示。

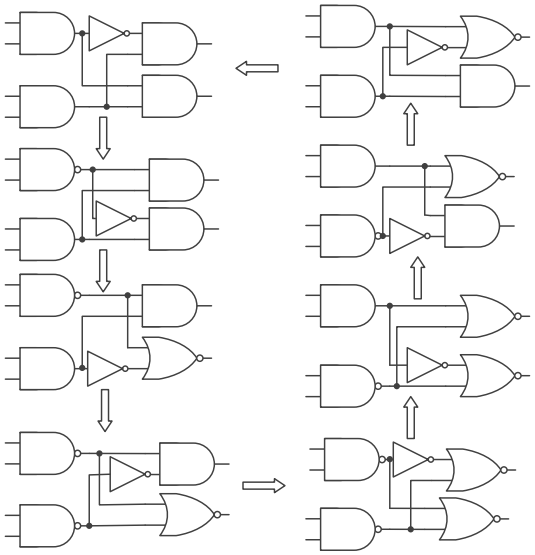
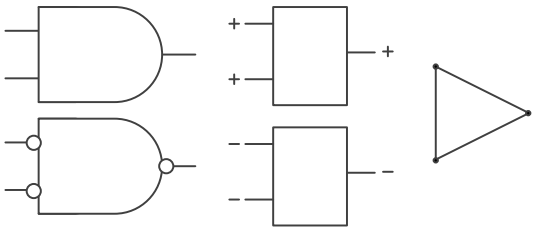


图 5.5 未化简的电路
Fig. 5.5 not minimized inverters

图 5.5 中不管怎么推进图中的反相器，或者和反相器扇出的门电路结合，或者和反相器扇入的门结合，都不能去掉图中的非门。从门级图中化简反相器并不直观，因此 Alok 在多级网络中提出了减少反相器的方法是基于 AND 和 OR 的极性转化实现的。



(a) (b) (c)
图 5.6 极性图：(a)与门和或门 (b)极性表示 (c)基于与门极性图
Fig. 5.6 polarity graph：(a)AND and OR (b) polarity representation (c)based AND polarity graph

图 5.6 是 AND 和 OR 的极性图表示过程，其中如图 5.6(a)所示，以 AND 为基础门，那么 OR 门可以看作是由 AND 的扇入和扇出同时添加反相器得到。图 5.6(b)更形象的表示门的扇入扇出极性，由于与门是基础门，那么认为与门的扇入和扇出极性均为正，用符号‘+’表示。而或门的极性要以与门的极性为准，若和与门的极性相同，也用符号‘+’表示，若和与门极性相反，用‘-’表示，如图 5.6(b)所示。或门的扇入扇出极性和与门相反，因此是三个负号表示。图 5.6(c)为门的极性图。

我们将极性相同的两点用实线连接，极性相反的点用虚线连接，因此构成了一个基本的极性图。从图 5.6(c)中可以看出这个极性图由三条边构成。

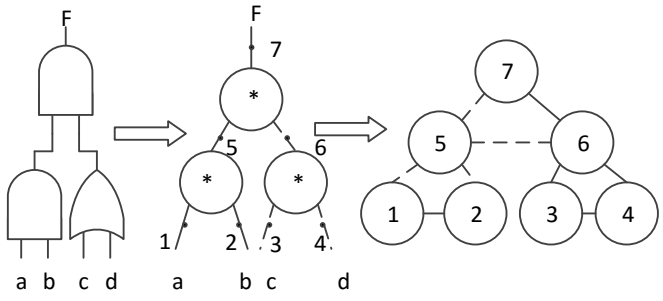


图 5.7 极性图流程：(a)逻辑函数 $F=(a*b)*(c+d)$ (b)标记极性节点 (c)生成极性图

Fig. 5.7 polarity flow：(a) function $F=(a*b)*(c+d)$ (b) marked node (c) derived polarity

将电路中的逻辑门依据单元库中选择好的基本门进行节点的映射，生成极性图，如图 5.7(c)所示，为逻辑函数 $F=(a*b)*(c+d)$ 的极性图。一个逻辑函数经过 ABC 优化生成如图 5.7(b)所示的 AIG 图，在 AIG 图中标记每个与门的扇入和扇出节点，依据选择好的基础门的极性为标准，针对图中每个与门节点的扇入扇出极性生成如图 5.7(c)所示的极性图。

当极性图生成之后，进行极性图的二着色过程。所谓二着色即标记图中每个节点的极性，在着色过程中会发现有节点既可以是正极性还可以是负极性如图 5.8 所示的节点 3，这时通常认为着色发生冲突，意味着在实际电路中可以通过添加反相器来表示该节点的极性。着色过程如下所示：

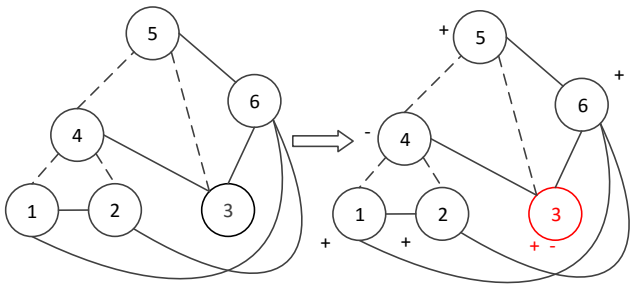
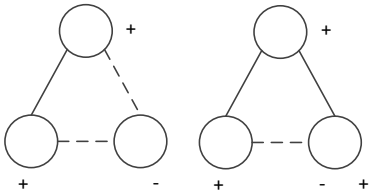


图 5.8 着色过程

Fig. 5.8 coloring process

在着色过程中，会出现节点的极性有正有反的现象。在实际的观察中发现当一个环中存在的虚线个数为奇数时，就会存在双极性的节点，如图 5.9(b)所示，而图 5.9(a)的虚线个数是偶数，所以每个节点的极性都是唯一的。当出现双极性

节点时，需要用一个反相器进行替代。而我们为了实现电路面积最小化，就要尽可能的减少反相器的出现，也就意味着找出图中尽可能少的虚线奇数环，找到的虚线奇数环越少所添加的反相器个数也会减少，从而实现电路面积优化的目地。



(a) (b)

图 5.9 虚线环的个数：(a)虚线偶数环 (b)虚线奇数环

Fig. 5.9 the number of negative line: (a) even cycle (b) odd cycle

本章将上述映射算法运用到新的数据结构中，由于新的数据结构 AXIG 中不仅仅有与节点，还有异或节点的存在，那么异或节点在极性图中如何表示；以及本章的单元库中只包含 NAND、NOR 门，因此我们选择这两种逻辑门作为基本门进行节点类型的转换；在搜索图中虚线奇数环时，结合异或门和同或门的性质进行反相器的化简，最终得到一个简化的电路。

5.2 问题的定义和基本思路

5.2.1 问题定义

本章所要解决的问题描述如下：给定一个 n 变量电路的 AXIG 形式，以选择的 NAND、NOR 逻辑门为基础，将 AXIG 图中所有的节点依据 NAND、NOR 两种基本门，生成一个极性图。在极性图中通过搜索图中相应的奇数环，实现反相器最小化的目的。

5.2.2 基本思路

所要解决的问题的思路如下所示：给定一个 n 变量的电路，针对最终所要映射所得的两种简单逻辑门 NAND、NOR，将 AXIG 图转换成基于 NAND 和 NOR 的极性图，之后通过搜索图中的所有奇数环，选择奇数环中出现次数最多的那个节点并删除，之后确定图中反相器的数量，达到所实现的电路面积最小化的目的。

5.3 基于 AXIG 的面积优化方法

5.3.1 基于 NAND 和 NOR 的极性图

本章是基于 NAND 和 NOR 实现的单元映射，AXIG 图中所有的与门都要用 NAND 或者 NOR 与反相器结合的方式实现。图 5.10 是以 NAND 和 NOR 为基本门时的极性图。从图中可以看出以 NAND 为基础，NOR 虽然扇出和扇入的极性与 NAND 完全相反，但最终生成的极性图和 NAND 保持一致，极性图中三边均是实线组成。

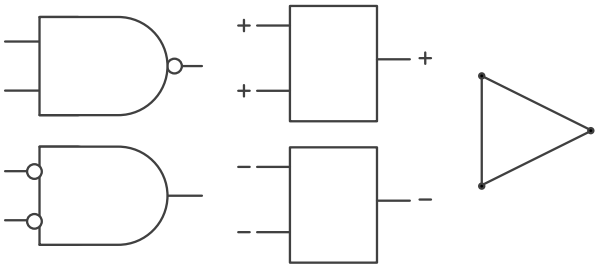


图 5.10 NAND 和 NOR 极性图

Fig.5.10 NAND and NOR polarity graph

在 AXIG 图中每个与节点之间扇入和扇出的极性不都是一样的，而在搜索图中每个与节点画极性图时，统一将与节点的扇出做为正输出，若图中与节点的扇出连接一个反相器，则将这个反相器认为是级联下一个与门节点的扇入。如图 5.11 是所有 AXIG 图中可能出现的与门节点不同扇入极性所生成的极性图。如第一个与门，以与非门为基础进行极性的判断，与门的两个扇入和与非门极性相同，因此标记为‘+’，而与门的扇出和与非门的扇出极性不同，因此标价为‘-’，而极性图中两个扇入极性相同，则用实线相连；而两个扇入的极性均和扇出的极性相反，因此用虚线相连。

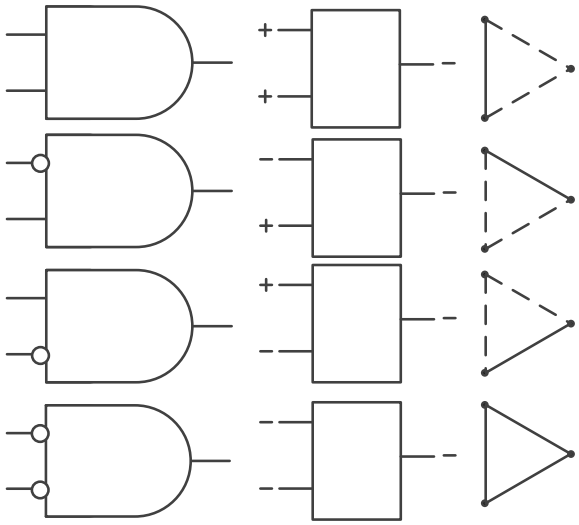


图 5.11 派生节点的极性图

Fig. 5.11 other nodes polarity graph

以上是本文提出的以 NAND 和 NOR 为基础所生成的极性图。而对于 XOR 和 XNOR 节点只需要标记出它们的扇入和扇出就可以画出完整的极性图。

5.3.2 搜索图算法的改进

本章的另一改进是搜索算法的改进。文献[17]提出了两种二着色方法，当在图中搜索到一个奇数环时，就要删除环中的某个节点，使其破坏掉图中的奇数环，这样也就不会有双极性的节点出现，当图中所有的节点都可以独立的着一种颜色时，表明该图已经标记完，找到图中已删除的节点个数，就可以知道最终插入反相器的个数。

本章的目地就是尽可能找出最少破坏奇数环的节点。通过观察发现，当找到一个奇数环时，应记下这个奇数环所涉及到的所有节点，接着继续寻找下一个奇数环。不管是用图搜索中的 DFS 或者 BFS 搜索方法，这两种方法只是搜索路径不一样，但最终都可以找出图中的所有奇数环，并标记所有奇数环所涉及到的节点数。通过记录节点数可知，若某个节点数越多，说明这个节点占用越多的奇数环，又或者说这个节点被几个奇数环所共享。因此删除这个被几个奇数环所共享的节点，这样可以同时破坏掉几个奇数环，而不是任意的去删除某个节点。之后继续重复搜索剩下图中的奇数环，寻找尽可能多的共享节点，实现着色过程。这样做的目的可以大大减少图中反相器的数量，通过添加少量反相器，就可以实现电路的映射过程。

5.4 算法流程及演示例子

根据以上讨论，提出基于 AXIG 的逻辑映射算法如下所示：

输入：前一章所生成的中间电路的 AXIG 表示。

输出：优化后电路的门级表示。

Step1. 计算中间电路网表中的原始输入、原始输出以及网络内部的节点数。

Step2. 为该电路创建储存网络的邻接矩阵，分配内存空间，并且为每个节点初始化为 0。

Step3. 生成极性图。遍历 AXIG 网络中的每个节点。

Step4. 遍历极性图中的奇数环。并记录奇数环中的节点 ID

Step5. 找到最大数的节点 ID，并标计该节点不存在，并将图中所有与该节点相连的线置 0.

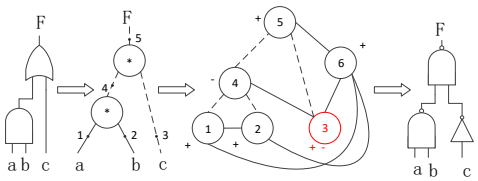
Step6. 继续着色，若存在节点着色不唯一：是，跳到 Step4；否，继续 Step7

Step7. 完成着色，输入删除节点的个数。

Step8. 将完成着色的图转化为由 NAND 和 NOR 组成的电路图。

Step9. 返回生成的门级网表。

以逻辑函数 $F=ab+c$ 为例，如图 5.12 所示为逻辑函数映射成只含有与非门和或非门还有反相器组成的电路。图 5.12(a)是最初的输入电路由一个与门和一个或门组成，若用 CMOS 实现，则所需的晶体管数为 12 个。图 5.12(b)为该逻辑函数的 AIG 表示形式，图中标记出了每个节点的扇入和扇出极性的顺序。图 5.12(c)是生成的极性图，并进行了一种着色方式。通过运用所提出的算法，可以得出图中的着色方法可以实行最终电路的面积最小化。图中节点 3 的极性为双极型，因此在具体的电路实现过程中要添加反相器。最终的电路实现如图 5.12(d)所示，由两个与非门和一个反相器组成，这种实现方法所需的晶体管数为 10 个，比初始电路少了两个晶体管，因此可以实现电路面积的优化。



(a) (b) (c) (d)

图 5.12 算法流程例子: (a) $F=ab+c$ (b)函数 AIG 表示 (c)极性图 (d) 生成的电路图

Fig. 5.12 the example of algorithm：(a) function $F=ab+c$

(b) AIG representation(c) derived polarity graph (d) derived circuit

5.5 本章小结

本章基于 AXIG 图采用简单门 NAND 和 NOR 以及复合门 XOR 和 XNOR 实现电路的映射。在映射过程中采用极性图的二着色方法减少图中反相器的数量，并将该方法运用到 MCNC[41]电路中，最终实现电路面积的优化。在映射过程中，只是为了达到面积最小化的目的，并没有考虑到实际映射过程中门的扇出和反相器的扇出都是有限定的，因此还需要在后期加入反相器树进行更进一步的实现反相器的最小化。

6 总结与展望

6.1 工作总结

本文的主要工作是围绕逻辑函数的双逻辑图形表示以及映射过程中如何实现反相器最小化这两个方面进行的。对于逻辑函数的图形表示，基于 AIG 构建 XOR 节点，生成一种基于双逻辑实现的图形结构 AXIG。在构建的过程中，主要以图形中的节点数为衡量电路面积的标准，而在后期与工艺相关的映射中，采用简单门和复合门组成的单元库实现图中节点的映射，生成极性图，实现电路中反相器最小化技术。

本论文的主要工作包含以下三个方面：

(1)基于 AIG 的探测

本文的逻辑函数是基于双逻辑实现的。而双逻辑的实现第一步需要逻辑函数的探测。本文借助图形数据结构 AIG 进行逻辑函数的探测，在所生成的 AIG 图中探测满足异或结构的特征，找到所有满足异或结构的节点，进行节点的压缩。最终压缩后的结果与 AIG 相比，节点数明显减少。

(2)基于 AIG 构建 AXIG

针对逻辑函数的双逻辑表示，提出了一种逻辑函数基于双逻辑的图形优化算法，通过探测 AIG 图中的异或结构，找出图中所有满足条件的 XOR，构建新的数据结构 AXIG，在新构建的 AXIG 中通过局部逻辑变换进行与工艺相关的映射，实现电路以晶体管数为基准的面积优化。所提出的方法与现有的学术界逻辑综合工具 ABC 相比，节点数明显减少，晶体管数也相应的减少。并且构建的 AXIG 可降低 AIG 的逻辑深度，因此潜在的达到延时优化的目的。

(3)基于 AXIG 实现面积的优化

本章将已有的映射算法运用到新的数据结构 AXIG 中，本章的单元库中只包含 NAND、NOR 门，因此选择这两种逻辑门作为基本门进行节点类型的转换；以这两个逻辑门为基准生成极性图，再搜索极性图中虚线奇数环，有选择的删除奇数环中的共享节点，达到优化反相器的目的，实现电路面积的优化。

6.2 研究工作的局限性及工作展望

本文对基于 AIG 的双逻辑面积优化技术进行了研究，借助逻辑综合工具 ABC 的基础上得到了较好的结果。但本文中的方法还存在需要改进的空间。研究表明，基于 AIG 的双逻辑面积优化仍需要进一步的研究：

（1） 本文中采用 AIG 进行逻辑函数的优化，还需要更进一步的研究逻辑函数的表示结构。在 AIG 图中所有节点局限于二输入的与门，即使进行节

点扇入扇出的极性变换，也只是将二输入的与门更换成二输入的与非门、或非门和或门。而对于更多输入门的实现存在局限性。如果在 AIG 图中加入三输入节点类型，或者用三输入节点表示任意逻辑函数是目前逻辑函数表示的一个趋势。

（2） 文中构建 AXIG 是基于 AIG 实现的，而所得到的 AXIG 图受限于最初生成的 AIG 图。若最初 AIG 图并不存在 XOR 结构，则探测的 XOR 也会受到限制，从而得到的 AXIG 也并不能很好的探测出所有的 XOR 门。并且在 AXIG 图中，并没有针对 AXIG 进行与异或运算的优化，因此需要改进。

（3） 基于双逻辑的门级映射过程，在与工艺相关的映射过程中，生成的逻辑图为了实现反相器最小化，并没有考虑实际映射中节点的扇出最大数，这一点在后期需要进一步研究如何限制门级扇出数和反相器的最大扇出数。

总的来说，本文的工作有待进一步的研究，但同时也为逻辑综合所生成的中间电路优化的研究工作奠定了基础，对于图形的压缩以及逻辑函数的优化以及新型的数据结构的研究具有一定影响。

参考文献

[1] XIANG WANG, LU YING, YI ZHANG, et al. Power optimization in logic synthesis for mixed polarity reed-muller logic circuits [J]. The Computer Journal, 2015, 58(6): 1306-1313.

[2] LUNYAO WANG, YINSHUI XIA, XIEXIONG CHEN. Logic synthesis and optimization based on dual logic [J]. Journal of Computer-Aided Design and Computer Graphics, 2012, 24(7): 961-967.

[3] YIHANG CHEN, YANG CHEN, JUINNDAR HUANG. ROBDD-based area minimization synthesis for reconfigurable single-electron transistor arrays[C]// VLSI Design, Automation and Test (VLSI-DAT), 2015 International Symposium on IEEE, 2015: 1-4.

[4] NOUREDDINE M A, ZARAKET F A. Model checking software with first order logic specifications using AIG solvers [J]. Trans. on Software Engineering, 2016, 42(8): 741-763.

[5] WESTE N H E, HARRIS D M. CMOS VLSI design: a circuits and systems perspective[M]. Pearson Education India, 2005.

[6] MARQUES F S, ROSA L S, RIBAS R P, et al. DAG based library-free technology mapping[C]// ACM Great Lakes Symposium on Vlsi 2007, Stresa, Lago Maggiore, Italy, March. 2007:293-298.

[7] Brayton R, Mishchenko A. ABC: An academic industrial-strength verification tool[C]//International Conference on Computer Aided Verification. Springer Berlin Heidelberg, 2010: 24-40.

[8] 何其贵. 数字电子技术基础[M]. 北京航空航天大学出版社, 2005.

[9] BRAYTON R K, RUDELL R, SANGIOVANNI-VINCENTELLI A, et al. MIS: A multiple-level logic optimization system[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1987, 6(6): 1062-1081.

[10] SENTOVICH E M, SINGH K J, LAVAGNO L, et al. SIS: A System for Sequential Circuit Synthesis[J]. Electronics Research Laboratory Memo. No. ERL/UCB M92/41, 1998

[11] Berkeley Logic Synthesis and Verification Group, “Abc: A System for Sequential Synthesis and Verification,” release 20130425. [OL]. Available: http://www.eecs.berkeley.edu/alanmi/abc/

[12] BRAYTON R K, HACHTEL G D, SANGIOVANNI-VINCENTELLI A, et al. VIS: A system for verification and synthesis[C]//International conference on computer aided verification. Springer Berlin Heidelberg, 1996: 428-432.

[13] BRYANT R E. Graph-Based Algorithms for Boolean Function Manipulation[J]. IEEE Transactions on Computers, 1986, 100(8):677-691.

[14] FABIO SOMENZI. CUDD: CU decision diagram package-release 2.4.0[CP/OL]. University of Colorado at Boulder, 2009,

[15] CHAI D, JIANG J H, JIANG Y, et al. MVSIS 2.0 programmer’s manual[J]. UC Berkeley (May 2003), 2003.

[16] FEI SUN, YINSHUI XIA. BDD based detection algorithm for XOR-type logic[C]//Proceedings of the 2008 11th IEEE International Conference on Communication Technology, . Piscataway: IEEE Press, 2008: 351-354.

[17] MISHCHENKO A, CHATTERJEE S, BRAYTON R. DAG-aware AIG rewriting a fresh look at combinational logic synthesis[C] Proceedings of the 43rd ACM Design Automation Conference, Piscataway: IEEE press, 2006: 532-535

[18] JAIN A, BRYANT R E. Inverter minimization in multi-level logic networks[C]//Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers., 1993 IEEE/ACM International Conference on. IEEE, 1993: 462-465.

[19] 罗嵘，刘伟，罗洪，刘勇等. 现代逻辑设计[M]. 电子工业出版社， 2006

[20] 欧阳星明. 数字逻辑第二版[M]. 华中科技大学出版社， 2005

[21] HACHTEL G D, SOMENZI F. Multi-Level Logic Synthesis[J]. Logic Synthesis and Verification Algorithms, 1996: 409-453.

[22] BIERE, A. AIGER Format.2007 [OL] fmv.jku.at/aiger/.

[23] Berkeley U C. Berkeley logic interchange format (BLIF)[J]. Oct Tools Distribution, 1992, 2: 197-247.

[24] 叶锡恩， 毛科益，夏银水. 一种新的用于探测 Pure Reed-Muller 逻辑的算法[J]. 浙江大学学报(理学版)， 2007， 34(3): 299-303

[25] 夏银水, 毛科益, 叶锡恩. 逻辑函数适于双逻辑实现的探测算法[J]. 计算机辅助设计与图形学学报, 2007, 19(12):1522-1527.
[26] 王伦耀, 夏银水, 陈偕雄. 基于多数覆盖的二级 MPRM 函数逻辑优化[J]. 电子与信息学报, 2012, 34(4):986-991.
[27] 叶锡恩, 毛科益, 夏银水. 基于乘积项的双逻辑实现探测算法[J]. 电子学报, 2009, 37(05):961-965.
[28] 王伦耀, 夏银水, 陈偕雄等. 基于不相交乘积项的逻辑探测和拆分算法[J]. 电子学报, 2012, 40(10):2091-2096.
[29] CORTADELLA J. Timing-driven logic bi-decomposition[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2003, 22(6): 675-685.
[30] BRAYTON R K. The decomposition and factorization of Boolean expressions[C]//Proc. Int. Symp. Circ. Sys.(ISCAS 82) Rome. 1982.
[31] 王伦耀, 夏银水, 陈偕雄. 逻辑函数的双逻辑综合与优化[J]. 计算机辅助设计与图形学学报, 2012, 24(7):961-967.
[32] KUEHLMANN A, PARUTHI V, KROHM F, et al. Robust Boolean reasoning for equivalence checking and functional property verification[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2006, 21(12):1377-1394.
[33] BALASUBRAMANIAN P, EDWARDS D A. Synthesis of Power and Delay Optimized NIG structures[C] Canadian Conference on Electrical and Computer Engineering. 2007:239-242.
[34] MISHCHENKO A, BRAYTON R. Scalable Logic Synthesis using a Simple Circuit Structure[C] Intl Workshop on Logic Synthesis. 2006.
[35] LI N, DUBROVA E. AIG rewriting using 5-input cuts[C] 2012 IEEE 30th International Conference on Computer Design (ICCD). IEEE, 2011:429-430.
[36] WU J L, ZHU W X. On a local search algorithm for the capacitated max-k-cut problem [J]. Kuwait Journal of Science, 2014, 41(3): 129-138.
[37] MACHADO L, MARTINS M G A, CALLEGARO V, et al. Iterative remapping respecting timing constraints[C]// 2013 IEEE Computer Society Annual Symposium on VLSI(ISVLSI). Natal, Brazil, 2013: 236-241.
[38] MATOS J M, RITT M, RIBAS R, et al. Deriving reduced transistor count circuits from AIGs[C]// Proc. of the 27th Symposium on Integrated Circuits and Systems Design, Aracaju, Brazil, 2014: 1-7.
[39] MATOS J M, BACKES M H, RITT M, et al. Mapping circuits with simple cells from Xor-And-Inverter graphs[C]// Proc. of the 24th International Workshop on Logic and Synthesis, Mountain View, CA, USA, 2015: 1-4.
[40] AMAR L, GAILLARDON P E, DE MICHELI G. MIXSyn: An efficient logic synthesis methodology for mixed XOR-AND/OR domi-nated circuits[C]//Proceedings of 18th Asia and South Pacific Design Automation Conference (ASP-DAC). Piscataway: IEEE Press, 2013: 133-138.
[41] ALBRECHT C. International Workshop for Logic Synthesis 2005 benchmarks [EB/OL].[2005-06].http://www.iwls.org/iwls2005/benchmarks.html.

在学研究成果

一、 参与的项目

“基于双逻辑的低功耗 IP 核设计基础理论与关键技术”，国家自然科学基金，重点项目(61131001)，2012.01~2016.12。

二、 发表的论文

1.赵思思，夏银水，马雪娇，吴世雄. 逻辑函数基于 AXIG 的实现. 无线通信技术。(第一作者)

致　谢

时间飞逝，转瞬就是 3 年的研究生生涯。还记得初来时什么都不懂的样子，而实验室就像是个大家庭，无论是老师，还是师兄师们都会给予各种帮助。转眼间我也变成师姐了。而实验室依旧是键盘敲击声，或者时而进行的热烈讨论，仿佛还是一点没变。

回想起刚来时，对于研究生学习的各种茫然、焦虑。而夏老师总是鼓励和引导我，找来相关的书籍资料让我们阅读学习。在初定研究方向时，又是夏老师指导我们如何开展文献阅读和开展研究。在我研究上遇到困难时，夏老师总是循循善诱，为我指引方向，或者和我探讨解决方案，提供更多思路。夏老师要求我们每星期去找他汇报研究进度，并讨论遇到的难题，一方面督促我们的研究进度，另一方面防止我们因为一些小问题而卡在死胡同里。夏老师对工作非常认真负责，我们的论文他都会修改很多遍，连标点都不放过。他对事物认真的态度，深深影响了我们。在此，向夏老师们致以诚挚的敬意和衷心的感谢。

同时，真诚地感谢实验室室的王伦耀教授，王健、钱利波老师以及师兄储著飞，在科研方法、科研论文的撰写和修改上，他们都给予了我大量的帮助。生活中，他们的平易近人和热情让我觉得很温暖。

感谢在研究过程中给予我们很多指导和帮助的师兄师姐们：寇彦宏、屈凤霞、厉琼莹、张骏立、汪纪波等。以及感谢 14 级电路信号班的同窗好友们，三年的时光，我们欢笑在一起，奋斗在一起，互相帮助，共同交流，解决了多个技术性难题，也克服了人生的坎坷，使我的研究生生活愈加充实丰富。

还要特别感谢我的父母亲，是你们一直激励着我，给我前进的动力，我的每一点进步，都与你们密切相关，是你们默默的付出让我毫无顾虑的前行。

最后，感谢百忙之中评阅本论文的专家教授和参与答辩会的诸位老师，感谢你们提出专业的指导和意见，使我继续进步。