

# 深亚微米集成电路可测性设计及其综合的研究

## 摘 要

随着深亚微米集成电路时代的来临, 数字电路的可测性越来越显现出他的重要性, 而且正在作为一个不断发展的领域独立出来。本文对数字集成电路的可测性设计作了深入研究。

本文首先介绍了组合逻辑电路的测试方法, 一般是基于故障模拟的寻找测试码的方法, 这里介绍了 D 算法, 以及由其发展而来的 PODEM 算法和 FAN 算法。非冗余的组合逻辑电路总是能用此法找到给定故障的测试码的。

而对于含有记忆元件的时序电路, 由于电路的输出不仅跟当前输入有关, 而且还跟过去的输入也有关, 按照一般的故障模拟方法测试将是一件非常耗时而繁重的工作。我们介绍了基于扫描的测试方法, 扫描链的选择和建立是一项比较关键的技术, 直接关系着测试性能和电路的系统性能。我们提出了一些有效的扫描路径设计技术, 并且分析了它们的特点和一些设计中要权衡考虑的因素。基于扫描的内建自测试技术是目前发展比较迅速的可测性设计方法, 我们从原理和结构上对此作了全面分析, 最后提出了一种矩阵扫描的 BIST 结构方法, 可供测试及故障诊断用。

传统的基于电路输出电压的逻辑观察的测试技术在深亚微米集成技术条件下不足以满足越来越多的集成电路产品的低故障漏检率要求。于是, 寻找电压测试技术的对偶方法来捕获从逻辑观察法中漏检的故障变得非常具有现实意义。我们介绍了一种电流测试法, 即  $I_{DDQ}$  测试法。本文对这种方法的原理深入分析, 对可测故障和测试方案作了详细阐述。在 CMOS 电路物理参数不断缩小的情况下, 电路的静态损耗电流也增加, 这对  $I_{DDQ}$  测试方法是一大挑战。本文给出了一些  $I_{DDQ}$  电流控制方法, 以及灵活运用  $I_{DDQ}$  测试方法的技巧。

可测性设计的重要性与日俱增, 将其纳入自动化设计流程是迫切所需。本文对逻辑综合作了介绍, 分析了目前常用的综合系统中用的数据结构 BDD, 具体研究了

由 BDD 映射逻辑门电路的技术，并提出了一种按照相关性原则来选取变量顺序的优化算法。

通过在电路综合中将测试策略考虑进去的方式，可以影响着逻辑综合过程的行为，可将由于测试而带来的额外花费降低，这就是可测性综合。我们研究了将可测性设计纳入综合系统的具体方法，并论述了几种可测性设计方案自动综合实现的技术。

由于电路的网表信息最终是在逻辑综合后确定下来的，所以逻辑综合过程的电路信息对电路的测试是非常重要的。我们对 KFDD 电路的可测性作了定性分析，能以很小的代价获得对可测性的控制，保证最终实现的电路完全可测，或达到规定的故障覆盖率。

关键字 可测性设计，扫描，内建自测试，静态电流测试，二叉决策图

# Research on the Design-For-Testability of Deep Submicron Integrated Circuits and Its Synthesis

## Abstract

With the coming of Deep Submicron Integration era, the testability of digital circuits is becoming more important and stepping out as a developing domain. This paper focuses on Design-for-Testability of digital circuits.

The paper begins with the testing methods of combinational circuits. Finding test vectors based on fault simulation features these methods. D algorithm and the derived algorithms such as PODEM and FAN are introduced. We can always gain the test vectors for a given fault in non-redundant combinational circuit by these algorithms.

As to the sequential circuit with memory elements, the output of the circuit is determined not only by the current inputs, but also the past inputs. So the testing of such circuits is very time consuming and difficult under common fault simulation methods. We introduce a new testing method based on scan, in which the selection and setup of the scan chains is critical and influences the testability and the performance of the circuit. Some efficient techniques for designing scan paths are presented. There are the analyses of these techniques and some trade-off factors taken into consideration in the design. Built-in Self-Test based on scan develops rapidly recently. We have paid more attention on the principles and the structures of BIST methods. Finally, a BIST structure using matrix scan technique is given, which can be used for testing and fault diagnosis.

Traditional voltage testing techniques based on the logic observation of outputs can no longer meet the low fault escape requirements in more and more IC products in the situation of deep submicron integration. So it is very wise to find the complementary testing techniques detecting fault escaping logic testing. We brought up a kind of current testing technique,  $I_{DDQ}$  testing. The principle of  $I_{DDQ}$  testing, the testable fault and testing strategies are explored deeply. With the scaling of CMOS parameters, the quiescent current consumption is increasing, which challenges  $I_{DDQ}$  testing. We have developed some  $I_{DDQ}$  current control methods and flexible application techniques of  $I_{DDQ}$  testing.

The importance of design-for-testability is increasing, and it is obliged necessary to embed design-for-test into automated design flow. This paper introduces logic synthesis and analyses the data structure BDD commonly used in synthesis systems. We studied the mapping technology from BDD to logic gates and proposed an optimizing technique for BDD by ordering the BDD variables in the principle of variable relevance.

Taking testing techniques into consideration in the circuit synthesis process will affect the behavior of logic synthesis, which can minimize the overhead due to the testability. This is synthesis-for-testability. We studied the actual methods for embedding design-for-testability into synthesis system, and discussed the realizing approach of automated synthesis for some design-for-testability techniques.

For the reason that the netlist of the overall circuit is determined only after logic synthesis, the information of the circuit from the logic synthesis is very helpful to test. We analyzed the testability of KFDD circuit theoretically, and the cost for testability control is low. Full testable circuit can be guaranteed or specific fault coverage can be reachable in the final.

**Key Words** Design-For-Testability, Scan, Built-In Self-Test,  $I_{DDQ}$  Test, Binary Decision Diagram

# 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，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：

罗春桥

日期：2002年2月4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。

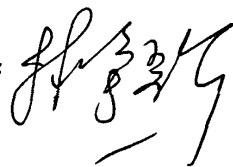
保密 ，在 5 年解密后适用本授权书。

本学位论文属于  
不保密 。

(请在以上方框内打“√”)

学位论文作者签名：罗春桥

指导教师签名：



日期：2002年2月4日

日期：2002年2月5日

## 第一章 绪论

随着集成电路规模的飞速增长，电路的逻辑设计和电路逻辑的测试越来越显现出他的重要性，而且正在作为一个不断发展的领域独立出来[1]。关于电路的逻辑诊断和测试方面的文献早在五十年代就已经有了，那时的测试主要是检测逻辑电路产品是否正常工作，也作为一种手段来找出生产过程中的弊端。进入六十年代，在当时技术条件下，逻辑电路实现了大规模化，也就是大规模集成电路（Large Scale Integratged,LSI）。由于电路的尺寸和复杂性都大大增加了，这使得测试工作变得更加困难，那时已意识到测试将成为集成电路大规模生产发展中的一个关键性的环节，这时国际上也出现了一些专门针对测试问题进行研究的机构和一些国际会议。到七十年代，就有大量有关测试方面的研究论文不断发表讨论。对于组合电路的测试，也产生了许多优秀的算法，应用的技术主要有布尔差分法，单路径敏化法，D算法，PODEM算法，FAN算法[2]。近几年，测试码模式的生成是一个主要的议题。由于时序电路的测试码模式生成很困难，可测性设计的概念就被提出来，也就是在电路开始进行设计阶段就要考虑被设计电路将来的测试问题，因为一个不可被测试的电路是一个没有使用价值的电路，这样，一些新的方法就又提出来了，如扫描设计，内建自测试设计等等，这些方法现已广泛地应用在许多逻辑电路的设计中，只是具体的实现和应用还有待进一步加强和完善，以及如何和现有的设计流程和技术完好地融合起来，使得电路的测试和测试设计变得更加方便，而对原电路的性能和花费影响最小，将是一个前沿性的课题。

在集成电路工艺进入深亚微米时代的今天，电路的可测性对我们提出了越来越多的要求，特别有些故障是我们传统的基于电压的逻辑观测法所不能解决的。这样驱动着可测性设计工程师们寻找其它的解决途径。基于CMOS电路静态电流的测量的测试方法被提出，并被看好为当今解决测试难题的又一良策，保证电子元件的质量和系统性能。

面对日益复杂的电路系统，完全靠手工的方法来保证电路的可测性将变得不容易，而且也不好保证测试质量。所以，将可测性的考虑提早地在电路设计的高层抽

象级别上进行，运用电路综合系统强大的计算能力来完成可测性设计的需要是一种迫切之需。

我们之所以要提倡可测性设计，也就是要将由工艺等原因引起的故障在电路的较底层暴露出来。因为对同样的一个故障，在较低的层面发现所花的代价要远远低于它在以后的高层面上发现所需的代价。试想一个有故障的芯片如果嵌入到一个系统中，必然导致整个系统的失效，将浪费大量的人力和物力。

### 1.1 可测性设计的提出

随着集成电路生产工艺的不断提高，集成电路的设计水平已进入到深亚微米时代。由于特征尺寸的不断缩小，单硅片上的集成度相应提高，100万门级的ASIC芯片已司空见惯。然而新的挑战依然摆在设计者面前，那就是可测性。

可测性早已被提出，但限于早期电路规模，它并不是设计过程中的主要问题，有很多算法和解决方案都能很好地完成任务。但在当今超大规模时代，不仅单晶片上电路逻辑功能超乎复杂化，时序逻辑引入的记忆单元使测试的实施更加困难，而且工艺水平提高所带来的故障模型多样化和集成度变大导致的故障几率的增加都将使得测试成本上升。这时为了保证设计的正确性以及方便设计的调整和工艺上的检查改进，可测性设计就变得尤为重要。很难想象一个100万门级的ASIC芯片不经恰当的测试就大规模生产而投入市场上。只有经过测试证明功能正确的芯片才能送到用户手中或用于自己更大的系统中，否则，一切等于白做。

为了尽快将新的产品投入市场，缩短设计周期是非常必要的。进入深亚微米集成电路时代后，ASIC芯片设计周期中有很大一部分时间是用于芯片的调试测试上的。如果事先没有很好地考虑它的可测性，一旦设计功能上出问题，将会花更多的时间和精力来调试或者根本无法解决，或者难以发现一些隐藏的问题，等到在用户手中出现就为时已晚。特别是当大规模生产某一款芯片时，由于总的测试时间是与单芯片的测试时间成线性关系，所以降低单芯片很小的测试时间，都将在总体上缩短不少测试时间，也就降低了成本，提高了产品在市场上的竞争力。一个好的可测性设计就可以实现这个目的，关键是可测性设计在产品的开发阶段能提供给设计者



强大的发现错误并改正的能力，并最终开发出自己想要的产品，而优秀的可测性设计方案能更快更好更准地将问题暴露于早期阶段，更及时地更正逻辑设计上的纰缪，调整时序上的混乱，改进工艺上的缺陷。

总之，可测性设计就是要为自始至终地保证设计的正确性而采取的一系列方便找错纠错措施，提高故障排除能力。

## 1.2 可测性设计的目标

由于电路规模越来越大，结构越来越复杂，而芯片面积和管脚又限制了测试手段的实施，使得大量的故障在传统的方法下变得不可测。在这种情况下，人们开始考虑在电路逻辑设计开始的同时，也安排测试的设计，使得电路的测试变得容易些，特别是要使测试码容易找到，使对整个电路进行有效测试大大简化，并能对结果妥善处理。这就是可测性设计(Design for Testability)所要求的。

可测性设计作为我们的设计目标，他应考虑以下几个问题：

- 故障覆盖率要高
- 测试数据要少
- 测试数据的生成时间要短
- 系统硬件花费要小
- 对电路逻辑功能影响小

## 1.3 可测性设计面临的问题

当逻辑电路的网络越来越大时，自动生成测试码的能力就变得越来越困难。当固定故障很多时，对测试码的错误模拟时间花费也将很大。有数据表明，计算机运行时间近似正比于逻辑门数的3次方[3]，因此，逻辑门数的少许增加，就会导致运行时间的快速增长，甚至使电路变得不可测，这就要求我们采用一些新的设计方法，配合测试的完成。

深亚微米集成电路(Deep Submicron Integrated Circuit, DSM IC)的可测性设计所面临的问题很多，但这许多的问题都可归结到下面的两个基本问题的花费上：

- 测试码生成
- 测试验证（通过故障模拟实现）

测试码的生成就是要产生发现故障所需要的电路激励，如果能快速方便地找到所需的测试码，当然测试的花费就相应小些。测试验证就是用提供的测试码对电路进行故障模拟以检测故障是否存在，如果能找到一种方法只需要少量的测试码就可快速诊断所有故障，或是达到预定的故障覆盖率，花费也会更小些。

可测性设计的花费主要基于两点，测试用时间和电路硬件开销，还有一点要考虑的就是可测性设计所带来的对系统功能的影响。

目前所作的可测性设计的改进研究，不外就是在可接受故障覆盖率下减小这两者的花费，并保证系统的性能。

虽然目前有强大的 EDA 工具为电路设计者服务，但电路规模飞速的增长以及器件特征尺寸的不断缩小，设计水平终究不及工艺水平提高的快。如何在日益缩小的芯片上为更为复杂的时序逻辑电路定制一个更为完备的可测性方案依然是许多设计工程师和 EDA 工具提供商亟待解决的问题。当特征物理量跨越一个阶梯之后，新的物理特征将会从次要的地位变成主导。首先是晶体管的物理特征随着特征尺寸的缩小，它带来的寄生参数的变化影响着电路的行为，使电路的特性变得对其他物理量更为敏感，也就出现了一些新的特征（如低功耗，高静态电流，高速）；其次就是互连线的延迟在整个电路系统中的影响将变得不可忽略，特别当电路规模庞大起来，进行逻辑功能块的划分，使得某些信号的连线过长而延时增加，如果不认真考虑他们的影响，建立准确的模型，可能本来正确的时序关系也会因为工艺水平的提高而变得不正确了。还有就是高速电路的出现使得电容电感等寄生参数日益发挥出了他们的作用，也影响着电路的性能，出现了串线(crosstalk)等问题。应该说在今天深亚微米集成电路设计中碰到的问题比以前多了很多，不仅会碰到，而且很多问题是一个人，一个工具，一次无法解决的，这时必须借助于电路本身的结构来进行设计问题或故障的排除，甚至利用计算技术将可测性设计融入综合系统中自动完成。

## 1.4 发展

可测性设计由来已久，只不过早期电路规模很小，逻辑上也较为简单，工艺上比不上现在这么复杂，所以有很多经典的方法足以应付。而且早期主要是解决组合逻辑的故障，所以有一些较著名的算法可实现对整个组合逻辑网络的遍历而取得测试码，从而检测测试码激励下的输出响应，如 D 算法、PODEM 算法和 FAN 算法，他们都采用通路敏化的技术在整个组合逻辑电路网络内进行故障驱动，直至能在输出端口观测到结果。已有很多测试码自动生成的 ATPG 工具可以应用，大大方便了电路的测试过程。

随着时序电路应用越来越广泛，而且时序结构也变得越来越复杂，针对时序电路的测试方法也得到了长足的发展。

首先是扫描通路设计方法，它将电路中的记忆元件用扫描触发器替换后互连起来，形成一条深入电路内部的扫描链路。这样就将整个时序电路转变成一个完全组合逻辑电路，通过扫描链对这个电路的组合部分进行控制和观测，从而改善了时序电路的测试问题。测试码的生成也可直接运用成熟的 ATPG 工具自动获取，有效地利用现有的测试方法。

由于电路规模和复杂性快速增长，简单的扫描方式已不适应新的要求，这种情况下诞生了基于扫描逻辑的内建自测试。它将测试码的生成，扫描链路控制和测试结果分析集成在一起，置于芯片内部。由于它可以直接利用芯片本身的电路特点量身定制，代价可以不是很高，但测试却很方便，而且可以提供系统级的测试接口。逻辑内建自测试是一种深亚微米集成电路中比较好的解决方案。

静态电流测试法 ( $I_{DDQ}$ ) [4] 是最近比较热的一种测试方法，它是与传统的基于逻辑观测的电压测试方法相对应的一种测试思想。正是由于集成电路工业的工艺进步，使得传统的电压测试法无法胜任，测试工程师们才想到了它的对偶方法—静态电流法。事实上，静态电流测试法确能检测到一些电压方法无法测试的故障。

最后是可实现性设计的自动化，与其他 EDA 工具一起成为整个电路设计过程中不可或缺的一环。由于可测性设计牵涉到具体的电路逻辑结构，如果将可测性设计在电路结构形成的逻辑综合阶段同时产生出来，可以得益于逻辑综合阶段形成的许多电路信息和数据交换；反过来，由于可测性设计的需要造成电路结构和器件上

的调整，也可直接影响综合行为。这样将两者融合在一起，直接利用双方的内部数据格式，能更加有效地实现可测性和性能的优化，设计者也可站在更高层次的抽象角度上管理整个电路系统的行为。

### 1.5 数字电路测试中几个基本定义

下面给出几个基本定义[2]，后面提到就不再另作介绍。

**逻辑门 (gate)**：能够实现一些简单逻辑运算的基本组合逻辑单元。如与非门(NAND)，或门(OR)，非门(NOT)等。

**触发器 (Flip-Flop)**：时序逻辑元件，它的输出不仅与当前输入有关，还与先前的输入序列有关。如 D 触发器。

**故障(Fault)**：数字电路与系统中由于制造过程，或操作条件，或老化等原因引起的元件物理失效或错误。

**逻辑故障 (Logic Fault)**：由于元件物理错误造成，可以改变数字逻辑元件的逻辑行为的故障。

**故障模型 (Fault Model)**：应用得最为广泛的逻辑故障模型是固定故障模型，即用 stuck-at-1(S-A-1)和 stuck-at-0(S-A-0)模拟电路中逻辑门间可能出现的故障，这时的故障点被认为是逻辑门的输入或输出被固定在逻辑电平 1 或 0。桥接故障也是比较常见的故障模型，电路中两节点或多个节点间电阻性地连接起来，但各故障节点的电平并不固定，而会随桥接阻值的变化而变动。另外还有一些其他的故障模型，如开路故障，延迟故障等等。本文着重研究的前两种故障模型的测试方法。

**测试码 (Test Vector)**：为了检测某一故障而在输入端给电路加的激励，以便从输出端观测的结果中确定电路有无故障。也叫测试向量。

**故障覆盖率 (Fault Coverage)**：由于不可能穷举电路的所有输入组合来进行故障测试，一般只是选用其中的一部分形成一个测试向量集。这个测试向量集所能检测到的故障数与总的故障数之百分比，称为这个测试向量集的故障覆盖率。一般是尽量要求故障覆盖率高，但不能无限制地以其他方面为代价。这也是评判一种测试方法或算法好坏的标准。

单故障(Single Fault): 每次测试只针对电路中的一个故障, 而其它不管。这样可将处理的问题简化, 便于算法的实现。单故障的测试集最后也能用来检测一些多故障情况(甚至桥接故障等)。而实践表明, 单故障的故障覆盖率可达到 90%以上, 可以满足要求。

## 1.6 可测性分析

为了指导可测性设计, 通常可对电路做可测性分析, 也就是对电路网络中各假想故障点测试难易程度的分析, 一般用可观性和可控性两个量来描述[5]。

可控性: 描写从原始输入端对电路给定内部节点赋以逻辑值的难易程度。

可观性: 描写把电路内部节点的逻辑值传播到原始输出的难以程度。

可测性分析的方法很多, 也很简单。作为一种可测性设计的量化分析, 为可测性设计提供了一种数量上的依据, 当然是可控性和可观性都容易最好, 但往往是它们的折中。

## 1.7 小结

可测性设计是当今集成电路发展所需, 是保证产品质量的关键, 也是复杂数字系统逻辑设计的一个有力补充。面对不同的电路系统, 有不同的测试方法。对于电路可测性的重要性, 至此, 我们已有了一定的认识。可测性要考虑的问题, 主要在于要能通过外部的激励, 能将电路内部的故障在观测点检测到。如何实现这一基本思想, 是本文的主要研究方向。由于可测性的考虑, 多多少少会影响到电路的正常逻辑功能, 如何运用合理的技术来既实现可测性需要, 又保证系统性能和成本, 也是设计者必须考虑的问题。

除了对可测性设计的产生与发展作了系统概述外, 本节给出了数字电路可测性设计中的一些基本概念。

## 第二章 组合逻辑电路的测试

组合电路是由一组逻辑门组成的一个布尔网络，它的内部各节点的逻辑状态仅由初始输入唯一决定。对于组合电路的测试通常采用故障模拟的办法，比如在一个特定的初始输入向量的激励下，通过观测输出结果就可判断这个电路是否存在故障。一般组合电路的故障模型选为单故障固定模型，这时只要针对不同的固定故障位置，找出能将这个故障信息传递到初始输出端的输入激励测试码就可以了。本章将介绍各种组合逻辑电路的测试码生成技术[2, 5-7]。

## 2.1 布尔差分法

设组合逻辑电路的输出可表示为函数  $f(x)$  的形式

$$f(x) = f(x_1, \dots, x_n)$$

$x_1, \dots, x_n$  为  $n$  个输入变量，设第  $i$  个输入变量  $x_i$  上有一个  $s$ -a-0 故障，记为  $X_{i, s-a-0}$ ，则函数变为

$$f_i(x) = f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$

记作  $f_i(0)$ ，同理  $f_i(x) = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$  表示输入变量  $x_i$  上有一个  $s$ -a-1 故障，记作  $f_i(1)$ 。检测  $X_{i, s-a-0}$  的条件为  $f(x) \oplus f_i(x) = 1$ 。即故障电路和正常电路输出逻辑相反。

由 shannon 定理知

$$f(x) = \bar{x}_i f_i(0) \oplus x_i f_i(1)$$

而  $f_i(x) = f_i(0)$

于是检测  $X_{i, s-a-0}$  的条件为

$$\bar{x}_i f_i(0) \oplus x_i f_i(1) \oplus f_i(0) = 1$$

$$\bar{x}_i f_i(0) \oplus x_i f_i(1) \oplus f_i(0) = x_i f_i(1) \oplus (\bar{x}_i \oplus 1) f_i(0) = x_i f_i(1) \oplus x_i f_i(0) =$$

$$x_i (f_i(1) \oplus f_i(0))$$

所以检测条件为  $x_i (f_i(1) \oplus f_i(0)) = 1$

其中

$$f_i(1) \oplus f_i(0) = f(x_1, \dots, x_i, \dots, x_n) \oplus f(x_1, \dots, \bar{x}_i, \dots, x_n)$$

称为  $f(x)$  对  $x_i$  的布尔差分, 记为  $\frac{\partial f}{\partial x_i}$ , 于是, 检测  $x_i$  上的 s-a-0 故障的条件为

$$x_i \frac{\partial f}{\partial x_i} = 1。同理可得检测  $x_i$  上的 s-a-1 故障的条件为  $\bar{x}_i \frac{\partial f}{\partial x_i} = 1。$$$

例 1. 组合电路如图 2.1 所示, 输出函数  $f(x) = f(x_1, x_2, x_3) = x_1 x_2 + x_3$ 。现要检测故障  $x_1$  s-a-0。

由  $x_1$  s-a-0 故障检测条件知,

$$x_1 \frac{\partial f}{\partial x_1} = x_1 [(x_2 + x_3) \oplus x_3] = x_1 x_2 \bar{x}_3 = 1$$

所以  $(x_1, x_2, x_3) = (1, 1, 0)$ 。即给图 2.1 所示电路加输入向量  $(x_1, x_2, x_3) = (1, 1, 0)$  可由  $f$  的结果检测是否有  $x_1$  s-a-0 故障。

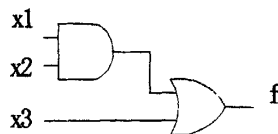


图 2.1 布尔差分法测试的组合电路  
Fig2.1 Combinational circuit tested by Boolean difference method

## 2.2 D 算法

下面的 D 算法也是要解决给定逻辑电路中某一根连线上的 s-a-0 或 s-a-1 故障的检测, 它在求解测试码的过程中不需要将电路换成逻辑函数, 而是直接从逻辑图上进行变换, 这样就可直接利用电路的门级网表信息来执行 D 算法而得到测试码。

### 2.2.1 基本概念

用符号  $D$  和  $\bar{D}$  表示故障信息:  $D$  无故障时为 1, 有故障时为 0;  $\bar{D}$  无故障时为 0, 有故障时为 1。

通路敏化：将故障信息传播到原始输出的过程，分为单通路敏化和多通路敏化。

逻辑门的原始立方 PC (Primary Cube)：它是真值表的简化表示方法，反映出逻辑门的逻辑关系。例如对应的输入与非门，其真值表如表 2.1，简化后如表 2.2，那么得到的原始立方 PC 如表 2.3。

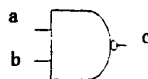


表2.1 二输入与非门真值表

a	b	c
0	0	1
0	1	0
1	1	0
1	0	1

表2.2 简化真值表

a	b	c
1	1	0
x	0	1
0	x	1

表2.3 原始立方表

1	1	0
x	0	1
0	x	1

逻辑门的原始 D 立方 PDC (Primary D Cube)：逻辑门的输入和输出都有 D 或  $\bar{D}$  的一种立方，反映正常元件传输 D 或  $\bar{D}$  的一种对应关系。还是以二输入与非门为例，它的原始 D 立方如表 2.4 所示。



表2.4 原始D立方

a	b	c
1	D	$\bar{D}$
x	0	1
0	x	1

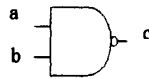
这时的变量取值集合为  $B_d = \{0, D, \bar{D}, 1\}$ ,

其中

$$0=(0, 0), D=(1, 0), \bar{D}=(0, 1), 1=(1, 1),$$

数序中第一个值为正常情况下的取值，第二个值为故障情况下的取值。

逻辑门的原始故障 D 立方 PDCF (Primary Fault D-cube)：反映故障元件的输入输出之间的对应关系，给元件（逻辑门）一组输入，使其输出为 D 或  $\bar{D}$ ，它的含义是对门的输出端有逻辑故障的一个测试。



a	b	c	正常	1	1	0
1	1	$\bar{D}$	故障	1	1	1



D 交(D intersection): 说明两个以上的门相互连接的情况, 两个集合的交集, 是求它们之间的公共部分。

对于图 2.2 所示电路, 它有一个二输入与非门和一个二输入或非门串接而成。与非门和或非门的真值表如表 2.5 和表 2.6 所示。



图 2.2 门电路  
Fig2.2 Gate-level circuit

表2.5 电路中与非门的真值表

a	b	c	d	e
1	1	0	x	x
x	0	1	x	x
0	x	1	x	x

表2.6 电路中或非门的真值表

a	b	c	d	e
x	x	0	0	1
x	x	1	x	0
x	x	x	1	0

扩展与非门和或非门的原始故障 D 立方可分别得到表 2.7 和表 2.8, 然后将这两个表求交, 即得到这两个门在电路中的 D 交。

表 2.7 与非门的扩展 PDCF

$$\begin{matrix} a & b & c & d & e \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{matrix}$$

表 2.8 或非门的扩展 PDCF

$$\begin{matrix} a & b & c & d & e \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{matrix}$$

求交结果为 (1 1 0 0 1)

一般计算为

$$\begin{matrix} 1 & 1 & 0 & x & x \\ x & x & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 \end{matrix}$$

求 D 交的数学定义:

设有两个 D 立方,  $a = (a_1, \dots, a_n)$ ,  $b = (b_1, \dots, b_n)$ ,

$\forall i, a_i, b_i \in \{x, 0, 1, D, \bar{D}\}$ ,  $a \cap b = \{a_1 \cap b_1, a_2 \cap b_2, \dots, a_n \cap b_n\}$

其中

$$x \cap a_i = a_i, a_i \cap x = a_i$$

当  $a_i \neq x, b_i \neq x$ , 则

$$a_i \cap b_i = \begin{cases} a_i & a_i \neq b_i \\ \phi & \text{其他} \end{cases}$$

只要有一个  $a_i \cap b_i = \phi$ , 则  $a \cap b = \phi$ 。

### 2.2.2 D 算法中常用的几个名词

**D 驱赶 (D drive)** 在故障点找出一个原始故障 D 立方 (PDCF) 后将 D 逐步传输到电路原始输出端的运算。

**D 链 (D chain)** 在 D 驱赶过程中出现的一个包含 D (或  $\bar{D}$ ) 在内的立方。

**D 前沿 (D frontier)** 在 D 驱赶过程中所遇到的输入为 D 或  $\bar{D}$ , 而这时输出尚未确定的电路算元的集合。

### 2.2.3 D 算法的构成

D 算法一般由三部分构成, 即:

**准备:** 故障敏化, 将故障点变成 D 或  $\bar{D}$ 。目的是使故障所在的门在故障存在和不存在时输出信号值不同, 即置 D 或  $\bar{D}$ 。

**传播:** 通路敏化, 将 D 或  $\bar{D}$  从故障门驱赶到原始输出端。

**线确定:** 根据 PDC 表把前面未确定的逻辑值确定下来, 是通过从故障所在门向原始输入回溯完成的, 并在回溯过程中对门的输入端赋逻辑值, 使门的输出端具有所需逻辑值。

### 2.2.4 D 算法的基本步骤

1. 在故障点置 D 或  $\bar{D}$ , 选择该点的一个原始 D 立方 PDCF 作为初始化 (tc, test-cube)。
2. 计算 D 前沿。

3. D 驱赶。选择被选点适当的 D 立方与现有的测试立方  $t_c$  进行 D 交，如果交的结果为非空，就得到了新的  $t_c$ ；如果为空，选择另一个原始 D 立方重复前面的工作。
4. 相容性保证(蕴涵)。每当进行一次 D 驱赶后，由于通路中某些未定的值变成了确定的值 '1' 或 '0'，如果在相关联的单元中，原始立方与此发生不一致，说明最近一次 D 驱赶不对，应予以抛弃。然后进行回溯：(1) 换一个 D 立方，如已穷尽，则(2) 换一个 D 前沿，如果也穷尽，则(3) 敏化多通路(有  $2^n-1$  种选择)
5. 一旦 D 驱赶到原始输出端，D 算法即告结束。所有原始输入值即为求得的测试码。

只要故障是可检测到的，用 D 算法总可把它的测试码找出来。但时间上是有问题的，一旦电路复杂起来，规模大起来，寻找测试码的时间是难以忍受的。这是一个 NP 问题，有改进的方法就是使用加权算法。

### 2.3 PODEM 算法

PODEM 算法与 D 算法很接近，但还包括了重新修改不正确决定的过程，它包括以下 4 个步骤：

1. 从某个目标对象为节点置值，将其他节点置为 "x"，该过程称为目标设置。
2. 从该目标回溯到输入管腿，并对输入管腿置值，该过程称为回溯。
3. 由输入管腿的值对电路进行仿真，观察敏感路径能否传到输出管腿上。如果不能，则对 2 得到的输入信号取非，重新进行仿真。
4. 修改目标，重复以上的过程，直到 D 演算值传到输出管腿为止。

下例给出了一个用 PODEM 算法完成故障检测的过程。

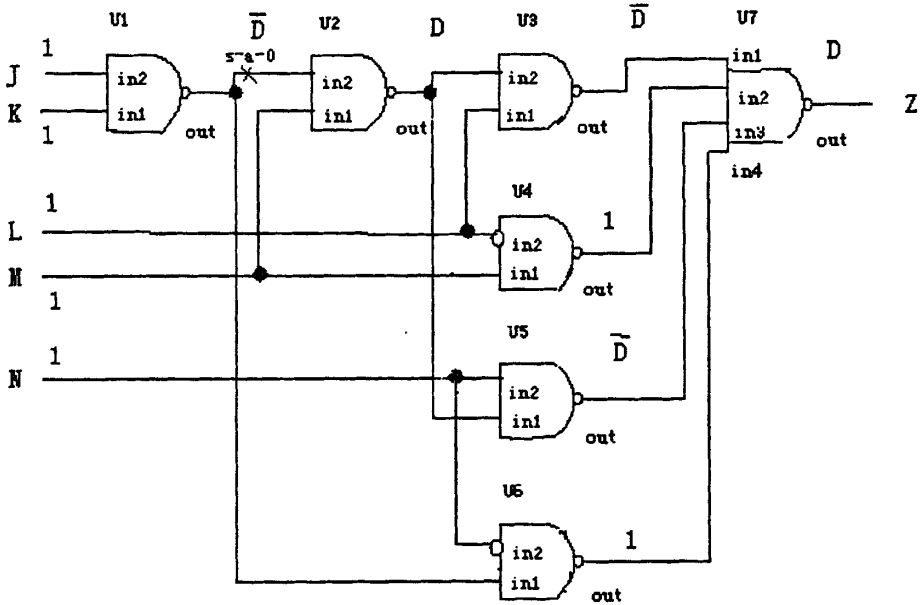


图 2.3 一个用 PODEM 算法完成故障检测的例子  
Fig.2.3 An example for detecting fault with PODEM algorithm

叠代步骤	目标	回溯	蕴涵	D 传递单元
1	U2.in2 = 0	J=1		
2	U2.in2 = 0	K=1	U6.out=1	
3	U2.in2 = 0	M=1	U2.out=D	U3, U5
4	U5.in1=1	N=1	U5.out= $\bar{D}$	U3, U5
5a	U7.in1=1	L=0	U7.out=1	U3, U5
6a	Retry	L=1	U7.out=D	completed

例：1. 以  $U2.in2 = S-A-1$  作为故障目标，设  $U2.in2 = \bar{D}$ ，由该  $\bar{D}$  向输入管脚回溯，置  $J=1, K=x$ 。

2. 不改变故障目标，这次回溯到 K，使  $K=1$ ，即  $U1.out=0$ ，因此， $U6.out=1$ 。
3. 为了将故障向输出处传递，置  $U2.in1=1$ ，即  $M=1$ ，使得  $U2.out=D$ 。
4. 为了将  $U2.out=D$  向输出传递，置  $U5.in2=1$ ，即  $N=1$ ，使  $U5.out=\bar{D}$ 。

5. 为使  $\bar{D}$  传过 U7, 就须 U7.in2=1。这样就置 U4.in2=1, 即 L=1, 使 U4.out=1。于是, U7.out=D, 这样就将 D 故障传递到了输出管脚上。

以上分析的 5 个过程中, 第一阶段(1, 2)为激发阶段, 对输入管脚设置一组固定值。第二阶段(3, 4, 5)为故障传递阶段, 为达到这一目标, 就要设置故障与输出管脚路径上经过的逻辑门, 将逻辑门与  $\bar{D}$  无关的项置为使能信号, 使  $\bar{D}$  得以传递, 最终的目标就是使  $\bar{D}$  能传到 U7.out 输出管脚上。

PODEM 算法是 D 算法的扩展, 而 FAN 算法是在 PODEM 算法的基础上进一步优化得来的, 可以减少检测时间。下节我们将对 FAN 算法作进一步认识。

## 2.4 FAN 算法

PODEM 算法使用回溯技术, 运用多路敏化法将 D 算法加以改进后, 使得对于一个可测故障总是可以用 PODEM 算法找到测试码的。而对于一般稍微复杂的组合逻辑电路, 寻找它的测试码是一件非常费时的工作, 最坏情况下, 将与电路大小成指数比例关系。为了加速测试码的生成, 针对 PODEM 算法的缺点, 如回溯重试可能会很多。这里提出的一种改进方法称之为 FAN 算法, 它更适应于大规模组合逻辑电路。它主要在以下方面有所改进:

- 1> 减少回溯次数
- 2> 缩短回溯之间的处理时间

### 2.4.1 定义

这里在对 D 算法和 PODEM 算法的基础上再增加几个定义, 以便理解 FAN 算法的实质。

约束线: 如果信号线可以从扇出点到达, 那么称该信号线为约束线。

自由线: 非约束线。

头线: 与某些约束线相邻的自由线。

多路回溯: 几条路径同时回溯, 它比单路径回溯更有效。

在多路回溯中，多个初始目标构成初始目标集，在回溯过程中出现要处理的目标叫当前目标，它们构成当前目标集。当回溯到头线或扇出点时，头线处的目标集合称为头线目标集合；扇出点的目标集合称为扇出点的目标集合。对同一扇出点可能会提出不同的目标值。对目标用一个数序  $(S, n_0(s), n_1(s))$  来描述。

### S 目标线

$n_0(s)$  要求目标线  $S$  的逻辑值为 0 的次数

$n_1(s)$  要求目标线  $S$  的逻辑值为 1 的次数

由当前目标确定下一目标的规则：

假设当前目标为  $y$ ， $x$  及  $x_i$  为下一目标，分别对应逻辑门的输出和输入。在扇出点， $x$  为扇出线， $x_i$  为各扇出分支。

1> 与门，设输入  $x$  最容易控制置 0，则

$$n_0(x) = n_0(y) \quad n_1(x) = n_1(y) \quad \text{而对于其他输入 } x_i,$$

$$n_0(x_i) = 0 \quad n_1(x_i) = n_1(y)$$

2> 或门，设输入  $x$  最容易控制置 1，则

$$n_0(x) = n_0(y) \quad n_1(x) = n_1(y) \quad \text{而对于其他输入 } x_i,$$

$$n_0(x_i) = n_0(y) \quad n_1(x_i) = 0$$

3> 与非门，设输入  $x$  最容易控制置 0，则

$$n_0(x) = n_1(y) \quad n_1(x) = n_0(y) \quad \text{而对于其他输入 } x_i,$$

$$n_0(x_i) = 0 \quad n_1(x_i) = n_0(y)$$

4> 或非门，设输入  $x$  最容易控制置 1，则

$$n_0(x) = n_1(y) \quad n_1(x) = n_0(y) \quad \text{而对于其他输入 } x_i,$$

$$n_0(x_i) = n_1(y) \quad n_1(x_i) = 0$$

5> 非门， $n_0(x) = n_1(y) \quad n_1(x) = n_0(y)$

6> 扇出点，扇出线要求置 0 或 1 的次数是各分支要求置 0 或 1 的次数之和。

$$n_0(x) = \sum_{i=1}^k n_0(x_i), \quad n_1(x) = \sum_{i=1}^k n_1(x_i)$$

对于基本门，从扇出向输入回溯，不改变要求置 0 或置 1 的次数，只有在扇出点才有对要求置 0 或置 1 次数的求和的运算。只有当前目标为空时，才能对各扇出点统计置 0 或置 1 的次数。

## 2.4.2 算法加速

PODEM 算法在回溯操作中采用的遗传算法如下：

- 1> 如果当前目标电平能通过设置当前门的任意一个输入到控制态（如，与 / 与非门的控制态为 0，或 / 或非门的控制态是 1）而获得，那么就将最容易设置的输入信号设为他的控制态。
- 2> 如果当前目标电平能通过设置当前门的所有输入信号到非控制态（如，与 / 与非门的控制态为 1，或 / 或非门的控制态是 0）而获得，那么选择最难设置的输入信号设置到非控制态。这是为了将不能设置成选定输入的信号早点找出，避免浪费时间在在该门的其他输入信号的设置上。

针对 PODEM 算法的不足，FAN 算法在下列方面进行算法的改进，能加速故障传递和测试码生成：

- 1> 作为一个 D 前沿要进行 D 驱动，选择最靠近原始输出的门。  
在遗传算法中，这可以运用可观测性来衡量。为了减少回溯次数，重要的是要尽早地将不存在的解决途径找出。在分支和界定算法中，当发现当前节点无解是，就立刻回溯而避免后续搜索，PODEM 算法缺少着方面的考虑。
- 2> 每一步通过唯一和蕴涵尽可能决定多的信号的值。运用蕴涵操作，通过电路结构同时前向和后向跟踪信号的取值。
- 3> 由唯一的故障蕴涵和决定来指定故障信号 D 或  $\bar{D}$ ，仅仅指定那些唯一确定的信号的值。
- 4> 当 D 前沿包含有单个门时，运用唯一敏化法。
- 5> 在头线停止回溯，推迟头线的线确定至最后。
- 6> 采用多路回溯，它比单路径回溯更有效。

### 2.4.3 FAN 算法的主要特点:

1. 每次回溯时，回溯到头线或扇出点为止，而不是回溯到原始输入点。头线可以由原始输入线确定，不会出现矛盾的情况。扇出点是矛盾的焦点，沿着不同的扇出分支到扇出点时，如果对扇出点的值要求不同，会出现矛盾的情况，以便决定是否继续。而仅根据一条扇出点分支的要求继续向后回溯，会增加回溯的次数。
2. 每当对网线赋一次值就进行向前向后的蕴涵操作，以达到减少回溯和提前暴露冲突的目的。

## 2.5 小结

这一章主要介绍了对于组合逻辑电路门级故障的测试方法。由于固定故障是一种比较常见的故障形式，其它很多故障也可由它来模拟，所以这里介绍的测试方法都是基于固定故障模型的考虑。一般由固定故障模型得到的测试码对其它故障类型也是有效的。

组合逻辑电路的测试关键在测试码的生成，本章重点介绍了 D 算法、PODEM 算法和 FAN 算法，这些算法都基于 D 算法的思想，即对设定的故障目标进行故障传递运算，直至在原始输出端能观测到为止，这个过程中确定下来的输入值即为所求的测试码。



### 第三章 时序逻辑电路的测试

随着集成电路规模的飞速增长，设计已迈向深亚微米时代，电路的测试和故障诊断已变得尤为突出和重要。这时测试和诊断无疑能为工艺的改进，设计的优化提供非常重要的信息，为下一步制造的成品率提高提供有力的保障。由于电路规模日趋复杂，时序电路的广泛存在，使得测试诊断在外部实施起来显得很艰难。

时序逻辑电路的输出和内部节点的逻辑状态不仅与当时的输入激励有关，而且还与过去输入向量有关，这主要是由于电路中存在有记忆元件。所以时序逻辑电路的测试要比组合逻辑电路复杂，因为内部记忆元件的状态是不好控制的，造成电路输出结果难以判断。本章针对时序逻辑电路的特点，介绍了时序电路测试中使用的方法和一些电路结构，以及它们应用于测试的原理。

#### 3.1 时序电路迭代组合电路测试法

##### 3.1.1 时序逻辑电路的 Hutmán 模型

如图 3.1 所示电路的输入为  $x_1, x_2, \dots, x_n$ ，输出为  $Z_1, Z_2, \dots, Z_m$ 。在每个反馈回路上都有相应状态变量  $Y_i (i = 1, \dots, p)$ ，由  $x_1, x_2, \dots, x_n, Y_1, Y_2, \dots, Y_p$  产生新的响应  $Z_1, Z_2, \dots, Z_m, y_1, y_2, \dots, y_p$ 。

$$Z_i(t) = f_i(x_1(t), x_2(t), \dots, x_n(t), Y_1(t), Y_2(t), \dots, Y_p(t))$$

其中

$$Y_j(t) = y_j(t + \Delta)$$

$$y_j(t) = g_j(x_1(t), x_2(t), \dots, x_n(t), Y_1(t), Y_2(t), \dots, Y_p(t))$$

$$1 \leq i \leq m, 1 \leq j \leq p$$

$\Delta$  为时序延时单位。

结果向量

$$Z(t) = \{z_1(t), z_2(t), \dots, z_m(t)\}$$

状态向量

$$S(t) = \{Y_1(t), Y_2(t), \dots, Y_p(t)\} = \{y_1(t + \Delta), y_2(t + \Delta), \dots, y_p(t + \Delta)\}$$

所有带有反馈的时序电路都可抽象为此模型。

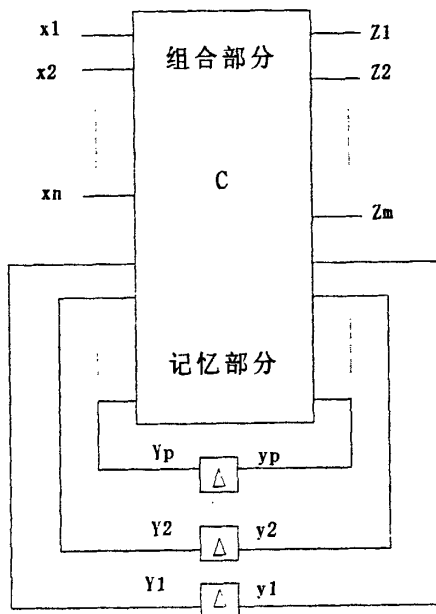


图 3.1 时序电路模型  
Fig3.1 Model of sequent circuit

### 3.1.2 时序电路迭代组合模型

如果要运用组合电路的测试码生成法对图 3.1 所示的电路模型进行分析，首先我们必须将电路中的时序反馈回路断开，形成一个纯粹的组合电路，然后再在每个时钟周期内对这个无反馈的组合电路分别进行故障传递运算，每次运算都要考虑状态变量  $S(t)$  的变化[5]。假设在  $r$  个时钟周期内可将故障状态传递到输出端，并确定相应的输入测试码，整个过程的实现可见图 3.2。

图 3.2 是将一个时序电路展开的示意图，它其实是将时间域上的测试问题转化为空间域上的问题，并把单故障变为多故障来考虑。将上面各分立而相关的电路在相应的引脚处前后连接起来，则形成一个相同电路单元的  $r$  次重复。对这个  $r$  次重复的大电路运用组合逻辑电路的故障分析法可解决它的测试问题。当然，当时序电路本身很复杂庞大的时候，这样的处理方法所带来的繁重性是不言而喻的。

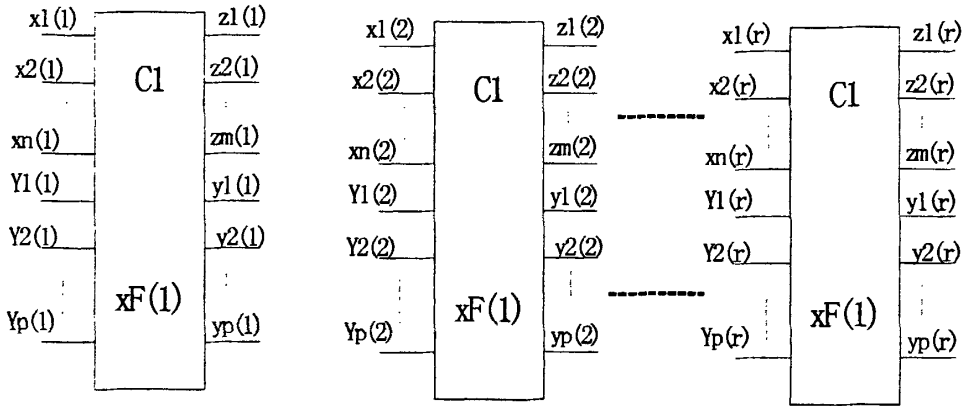


图 3.2 时序电路的迭代组合电路形式  
Fig3.2 The iteration of sequent circuit with combinational circuit

### 3.1.3 探索性算法

- 1> 把原来的一个输入向量序列  $\overline{X(1)}$ ,  $\overline{X(2)}$ , ...,  $\overline{X(r)}$  作用于时序电路, 变成该输入向量序列中的每一个分量, 分别作用于迭代组合电路模型中对应的帧。
- 2> 反馈线已被断开, 在运算上作为组合电路来处理。D 算法可用于该模型, 但概念上仍是时序电路, 在理论和实际操作上有区别。因此, 它不是严格的解析方法, 而是一种探索方法。
- 3> 目的是使最终的输出中包含故障信息 D, 同时使响应应当与内部状态无关。序列的长度就是根据使响应与初始状态无关这一要求而确定的。设序列的长度为 r, 则测试序列为:

$$T = \begin{pmatrix} x_1(1) & x_1(2) & \cdots & x_1(r) \\ x_2(1) & x_2(2) & \cdots & x_2(r) \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ x_n(1) & x_n(2) & \cdots & x_n(r) \end{pmatrix} = T^1 T^2 \dots T^r$$

$\overline{X(1)} \quad \overline{X(2)} \quad \cdots \quad \overline{X(r)}$

$T^i$  是测试码  $\overline{X(i)}$ ,  $1 \leq i \leq r$ 。

### 3.1.4 算法的实施

- 1 从最后一帧做起, 要使  $z_1(r), z_2(r), \dots, z_m(r)$  中包含 D 或  $\overline{D}$ , 也就是把故障点  $F(r)$  设置的 D 驱赶到  $z_1(r), z_2(r), \dots, z_m(r)$  中任意一个, 求出所需的初始输入 (PI(r))  $x_1(r), x_2(r), \dots, x_n(r)$  及伪输入 (SI(r))  $Y_1(r), Y_2(r), \dots, Y_p(r)$ 。
- 2 再处理倒数第二帧, 即第  $r-1$  帧。这一帧的伪输出  $SO(r-1) = SI(r)$ , 即  $Y_1(r), Y_2(r), \dots, Y_p(r)$ , 已经确定。根据 D 算法中求蕴涵的方法, 推出  $x_1(r-1), x_2(r-1), \dots, x_n(r-1)$ , 即 PI(r-1), 和  $Y_1(r-1), Y_2(r-1), \dots, Y_p(r-1)$ , 即 SI(r-1), 其中有些值为 x。
- 3 依次类推, 直到第一级。SI 的所有分量都是 x 时, 则确定为第一级。然后列出所有的 PI(i),  $1 \leq i \leq r$ , 即得到测试序列。

## 3.2 扫描测试

扫描测试简单定义为利用可扫描的移位寄存器结构执行测试功能。扫描测试是一种结构化的方法学, 它能标准化, 也可重复使用, 容易实现自动化 (在插入和测试向量生成两方面)。扫描结构允许数据状态用扫描移位寄存器置于芯片内部, 也允许芯片内部的数据状态用同样的扫描移位寄存器来观测。扫描结构方便了算法软件工具验证测试设计的正确性和生成所需的测试码来验证余下的电路。多路扫描移位寄存器, 或称多条扫描链, 能帮助优化测试器所需的测试向量的深度。总之, 扫描测试方法学加强了取得高质量检测的能力, 降低了测试成本 (测试码向量的大小), 减少了测试器的测试时间——也就减少了芯片成为合格产品所花的时间。

一般的功能测试只是仅仅将功能的、操作的、行为的测试码施加到被测电路的测试, 是用来验证电路行为的正确性与否, 因此, 它们没有针对确定结构的测试向量有效。

例如，图 3.3 给出的是一个具有组合电路输入和时序深度的电路。象这样的混合电路测试起来既要组合输入向量测试码，又要一些时序时钟才行。要完全测试这个电路需要  $2^6$  个组合向量和  $2^4$  个时序组合。因此，这个电路需要  $2^{4+6}$  个向量来获得 100% 的故障测试。如果再添加一些引脚或增加多一点时序深度，测试难度将成指数地增加 ( $2^{4+6}$  很快上升为  $2^{5+7}$ )。

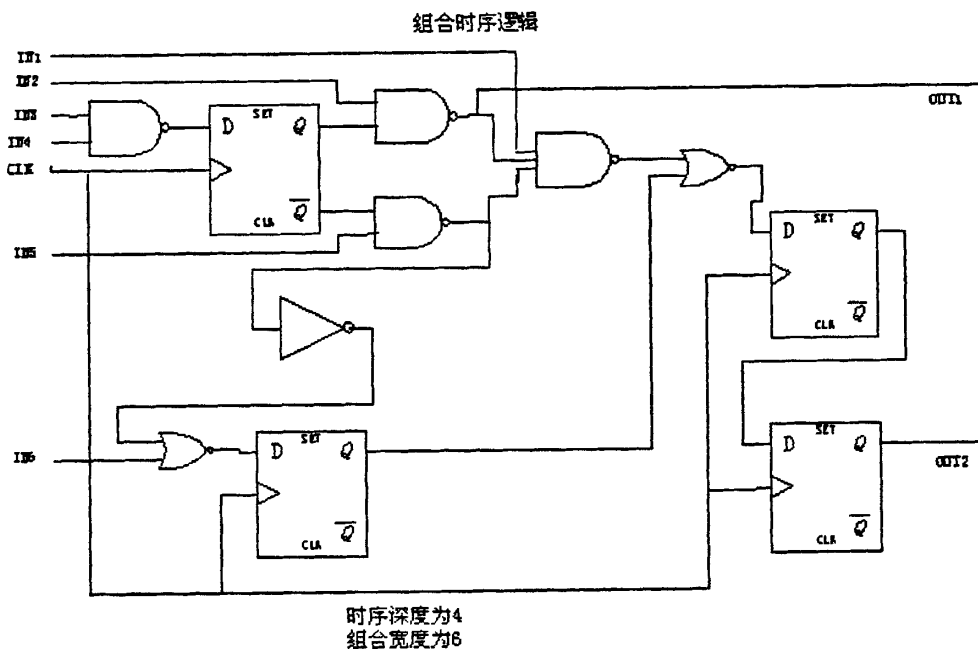


图 3.3 一个实际的时序电路  
Fig3.3 A general sequent circuit

### 3.2.1 扫描通路设计

由于时序电路的复杂化，传统的组合逻辑电路的测试方法是难以满足如今深亚微米集成电路的测试要求的，相反，利用时序电路本身的特性，将整个电路系统划分为不同的功能模块，分别对待时序元件和组合元件，可以为测试码的生成和测试过程的实施带来新的途径。在[8-11]中介绍了一些具体的扫描触发器选取方法。

扫描电路是针对很多的 ATPG 工具的优化措施，它将时序设计描述转变为仅仅是一个组合电路的描述。扫描使内部节点能直接访问，将每个触发器转变为至少一

对的测试点。测试点控制是在扫描触发器的  $Q$  输出端实现（如果支持的话也可是  $\bar{Q}$ ），能让电路控制值直接传递（通过执行扫描移位加载操作完成）。测试观测点在每个触发器的 D 输入端，能让电路状态值直接被观测到（通过执行扫描移位下载操作完成）。

图 3.4 是一个简化的扫描电路，ATPG 工具认为这个电路中的时序元件不存在，仅仅只是控制和观测点存在—正常的输入输出和所有其它测试点扫描可访问。

这个扫描电路由扫描插入过程生成。扫描插入过程将所有标准系统触发器用可扫描等价触发器替换，并将它们连成一条扫描链。当一个全扫描设计描述送给组合 ATPG 工具后，ATPG 工具将这个设计描述转变为内部格式表示这个扫描电路（仅仅是组合的）。

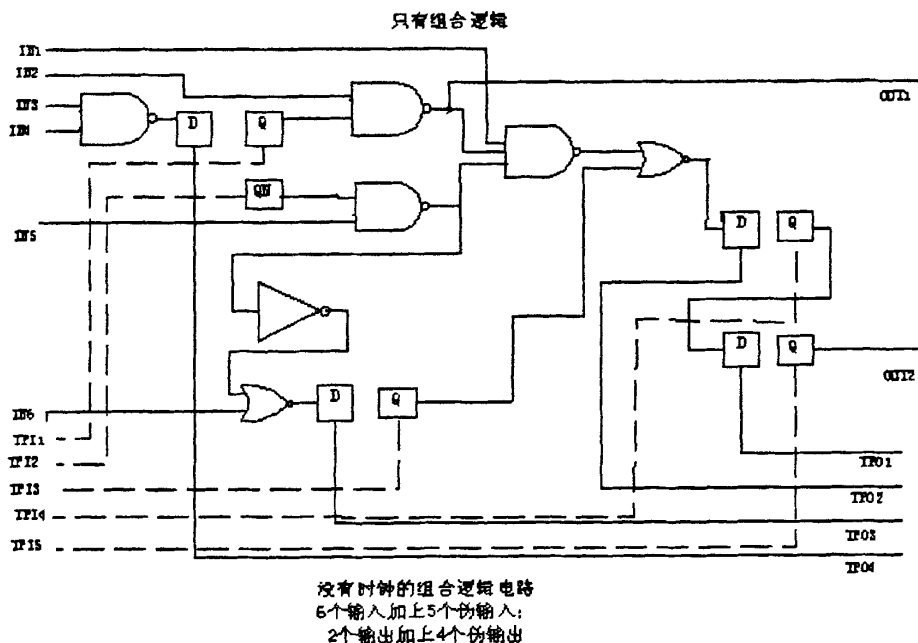


图 3.4 采用扫描后的时序电路  
Fig3.4 Sequent circuit with scan

扫描通路的可测性设计只需要增加少数几根外部连线(不超过 4 根)，便可以访问电路内部许多节点，改善电路的可控性和可观测性。扫描通路设计的思想是将电

路内部的记忆元件(如触发器)的结构稍加改进,使其能工作在正常状态和扫描测试状态下。在扫描测试状态时,这些经改进的记忆元件可互连成一条长链,形成一个移位寄存器,在电路的外部有一个扫描输入口和一个扫描输出口,通过这两个口,可以将测试码输入到电路内部需要测试的部分,预置这些记忆元件的状态,也可将测试码激励后的电路响应由这些被设置成观测点的触发器移位输出。

通过将电路内部的记忆元件的预置,就可将电路其余部分看作一个大的组合逻辑电路,从而可以利用组合逻辑电路的测试方法来有效地测试时序电路,这样就将时序电路测试的复杂性化解了。

如图 3.5,是一个可用于扫描通路的多路数据触发器(Multiplexed Data Flip-Flop)。在这个触发器中,由 T 来控制是  $d_0$  还是  $d_1$  在时钟触发下送给 Q。

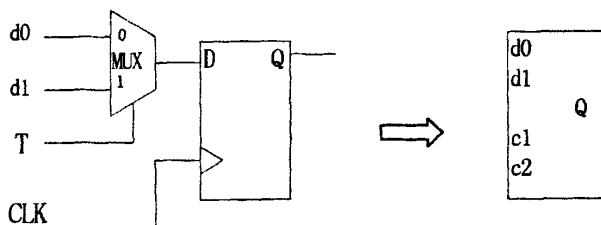


图 3.5 可扫描多路触发器  
Fig3.5 Scanable multiplexed data flip-flop

可以用这种触发器构成如图 3.6 所示的可测性设计的电路

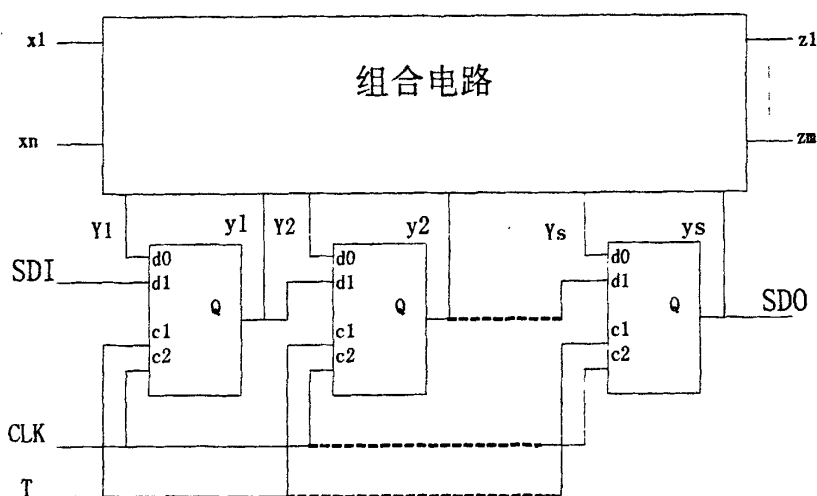


图 3.6 具有扫描通路的时序电路  
Fig3.6 Sequent circuit a scan path

SDI 扫描数据输入端口

SDO 扫描数据输出端口

正常运行时,  $T = 0$ ,  $Y_1, Y_2, \dots, Y_s$  作为触发器的输入,

测试模式下,  $T = 1$ ,  $D_i = Q_{i-1}$  ( $i=2,3,\dots,s$ ), 作为触发器的输入, 形成一个移位寄存器。

测试的步骤如下:

- 1> 置  $T = 1$  (扫描方式)
- 2> 将测试码  $y_i$  扫描到触发器中 ( $i=1,2,\dots,s$ )
- 3> 在输入端  $x_i$  上施加输入向量 ( $i=1,2,\dots,n$ )
- 4> 置  $T = 0$  (正常运行模式), 经充分长时间稳定下来后, 检查  $Z_k$  的值 ( $k=1,2,\dots,m$ )
- 5> 给一个时钟信号  $clk$
- 6> 置  $T = 1$ , 将  $y_i$  ( $i=1,2,\dots,s$ ) 扫描出, 检查  $y_i$ , 与正常值比较。

关于扫描触发器的结构, 可以有其他的多种变化, 但基本思想都是让他们能工作在正常和测试扫描两种状态下, 如双门触发器结构、电平敏化扫描设计、移位寄存器用触发器等。

### 3.2.2 扫描路径的选取

#### 1 全路径扫描

全路径扫描方式就是将电路中所有的触发器等记忆元件用具有扫描结构的相应触发器代替, 将它们串行连接起来, 在电路内部形成一条长链。当电路处于测试模式下时, 这条长链的功能相当于一个移位寄存器; 当电路处于正常工作模式时, 这条链上各个触发器按正常模式工作。正是因为全路径扫描将所有时序元件变得外部可观测, 所以, 将时序逻辑电路的测试问题完全转变成一个组合逻辑测试问题。能得到比较高的测试故障覆盖率, 不过也增加了门数和布线难度。

#### 2 部分扫描设计



部分扫描就是将电路中的部分记忆元件作为扫描链上的元件。因为根据实际情况来看，一方面要将电路中所有的时序元件连起来，并不是一件非常容易的事；而且即使采用全路径扫描，有时并不会提高电路的故障覆盖率，相反还会使测试速度变慢，使测试码的生成更加困难。鉴于此，发展了很多部分扫描路径选取的算法规则，在整个电路中搜索一部分的记忆元件作为扫描链上的备选元件。

部分扫描能带来更高的测试效率，而且相应的硬件花费也要小些，因此较全扫描更受欢迎。

### 3 多路扫描设计

在电路规模和时序结构日益增大和复杂的今天，扫描设计也会受到测试时间过长的挑战。为了减小扫描链的长度，但又不降低故障覆盖率，提出了多路扫描设计方法，它是在电路中同时建立多条扫描路径，可以减小扫描链的时序深度，缩短数据扫描入和扫描出的时间。由于各扫描链是并行的，它可使整个测试时间减低很多。另外它使各扫描链间的测试码有了重用的可能，减轻了测试码生成困难的问题，等效加快了测试码的生成速度。如下图所示，是一个多路扫描电路的结构示意图，整个电路被划分成不同的被测电路模块(Circuit Under Test, CUT)，由不同的扫描链(SC)输入测试码和记录测试响应，然后通过一个多输入特征分析器 MISR (Multiple input signature register)来检测测试结果，扫描输出。

不过多路扫描有时要求更多的外部输入输出端口。

图 3.7 是一个应用多路扫描技术来广播测试码给不同的被测模块的示意图。

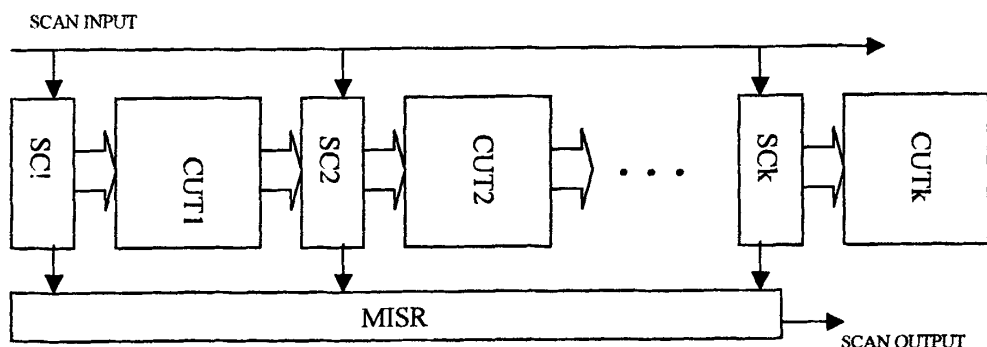


图 3.7 多路扫描电路  
Fig3.7 Multiple Scan circuit

#### 4 正交扫描

通常的扫描方式只是将电路中的记忆元件连接起来形成一条扫描链，例如象一个移位寄存器，一位数据一位数据顺序地流动。而正交扫描方式下，数据的流动方向与传统方式是正交的，它的数据是在跨寄存器间流动的[12]。它们之间的区别如图 3.8a 和图 3.8b 所示。

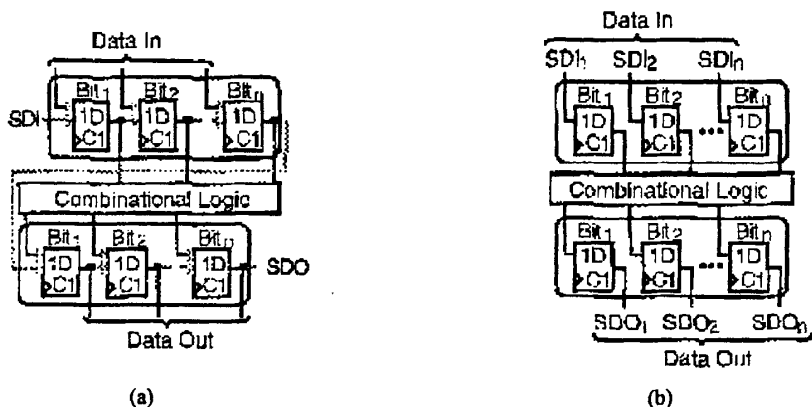


图 3.8 (a)传统扫描路径 (b)正交扫描路径  
Fig3.8 (a)Traditional scan path (b)Orthogonal scan path

由于传统扫描的数据流动和正常数据流动方向有别，在寄存器的每个触发器的数据输入端都加了一个复用器，将正常数据和扫描数据隔离。而正交扫描的数据流与正常功能下的数据流一致，所以少加了一些隔离复用逻辑电路。传统方法没有考虑电路具体的实现细节，或者说没有利用电路本身的功能。如果利用高层的数据通路逻辑信息，形成正交扫描通路，实现功能逻辑和扫描逻辑的共享，则能节省更多的硬件花费。

由于正交扫描同时有  $n$  条扫描输入（这里的  $n$  为数据通路的数据宽度，也就是寄存器的位宽，有点象多路扫描方式），相对于传统的扫描方式，则会更快些。它应用于测试的时间应该与数据通路上的寄存器数目成正比，而不是单个触发器的数目。

### 3.3 基于扫描的内建自测试

内建自测试(BIST, Built-in Self-Test)技术建立在扫描通路技术上，将测试码自动生成(Automate Test Pattern Generation, ATPG)功能和测试结果分析(一般采用特征分

析寄存器, Signature Analysis Register, SAR)功能集成在一起, 在电路内部形成一个专门供测试用的电路模块。这也是当今使用较广泛, 被看好为解决深亚微米集成电路比较有前途的一种测试方案[1, 13-20]。

它针对组合逻辑和时序逻辑, 存储器, 多路复用器等具有大量重复结构的单元。为了减少芯片用于外部测试的管脚数目, 内建自测试将测试所加的激励生成和响应结果分析都用嵌入模块实现, 只用一条线将检测结果送出片外。其测试码一般采用伪随机序列, 测试结果的检测则采用签名校对或特征向量分析技术[13, 21]。

### 3.3.1 伪随机序列发生器

有时, 扫描测试码不是来自一个测试器, 而是芯片上(或芯片外)某种计数器。一种普通的奇数阶(odd-ordered)计数器用于随机或确定的测试码向量的生成, 叫做线性反馈移位寄存器(LFSR, Linear Feedback Shift Register)。一个被配置来生成随机码流的LFSR叫做伪随机码生成器(PRPG, Pseudo-Random Pattern Generator)。如图3.10, 给扫描链路输入测试码的部件就是一个伪随机序列发生器。基于PRPG的测试向量可直接从LFSR的输出端输送到电路的扫描结构中, 或者经过一些去相关逻辑(decorrelation logic)后再应用(在输入不同扫描链的数据间加入一些逻辑上的间距, 以便所有的链不要接收完全一样的数据, 也称“随机性随机器”)。

### 3.3.2 LFSR 原理

LFSR利用了布尔逻辑特征—乘法可以由左移实现; 除法可以由右移实现。如果一个移位寄存器利用从不同位的反馈来右移, 那么这个操作是由2加上一个常数的出发。如果这个常数是一个质数, 则效果就相当于这个移位寄存器是在用一个质数除输入的数, 结果中总是有一个余数, 除非寄存器状态和下一个值相加恰好是这个质数。PRPG操作将在寄存器中生成所有可能的逻辑值, 除了全零状态。签名分析操作总是在寄存器中留下一个余数作为残留值。具有这些特征的多项式叫做质多项式

或本征多项式 (prime polynomial or primitive polynomial)，这个多项式指明要在寄存器什么地方加反馈点。

例如，图 3.10 中实现 PRPG 的 LFSR 的多项式  $x^3 + x^1 + x^0$  (也写作  $x^3 + x + 1$ ) 代表二进制位的位置  $2^3 + 2^1 + 2^0$ ，十进制  $8 + 2 + 1$ ，它是质数 11。如果这个 PRPG LFSR 用全 1 初始化，然后反复加时钟信号，结果将会是下列的 3 位序列值：

111 → 011 → 001 → 100 → 010 → 101 → 110 → 111

可以看到，它生成了  $2^n - 1 (2^3 - 1)$  或 7 个状态。

### 3.3.3 签名校对分析电路

同样地，有时输出验证的工具不是测试器，而是芯片上（或芯片外）器件，这样的器件被设计用来捕获输出响应，并将其转化为“通过或失败”显示，常见用于此类目的的器件是数据压缩线性反馈移位寄存器。

LFSRs 被配置来接收数据，并将数据流压缩成一个二进制签名时，就叫做签名分析器（或响应压缩器）。签名分析器能接收并行数据流或串行数据流。由多输入签名分析寄存器（MISR）处理的是多路并行数据流，由串行输入签名分析寄存器（SISR）处理的是串行数据流。如图 3.9 所示为一个多项式为  $P(x) = x^5 + x^4 + x^2 + 1$  的 SISR，它对应的二进制序列为“110101”。设在 seq\_in 端输入序列 B = “11110101”，则经 8 个时钟周期后，各 D 触发器上输出为 Q5Q4Q3Q2Q1 = “00101”——这就是 B 序列对应 P(x) 的签名。

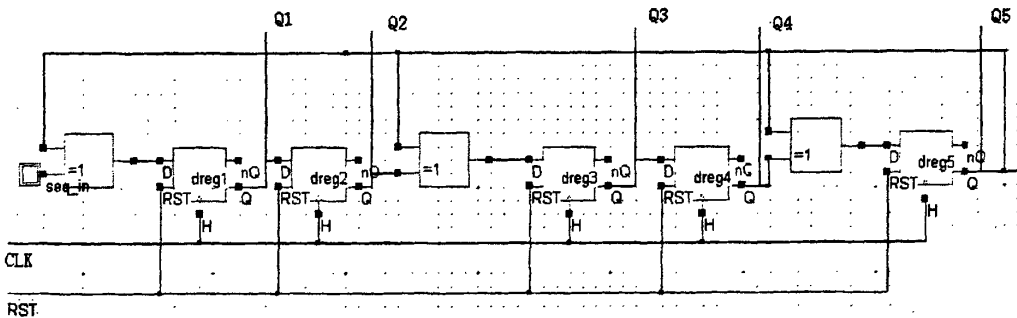


图 3.9 一个单输入特征分析寄存器的电路图  
Fig.3.9 A single input signature analysis register

### 3.3.4 逻辑内建自测试

用 LFSRs 执行逻辑测试时，通常称为内建自测试 (BIST, Built-in Self-Test)。当应用于一般的组合和时序逻辑时，这样的配置就叫做逻辑内建自测试或 LBIST。将 BIST 用作一种欺骗性的测试 (confidence test) 或方法，通常是为了减小指定生成测试向量的大小，而且不必依赖于测试器而实现 (例如，当芯片处于系统中时，测试工作也能进行，只要有稳定的时钟源，见图 3.10)。

这种测试通常只是一种欺骗性的测试，有以下几点原因。一是 PRPG LFSR 产生的序列值是有限的 (所有  $2^n - 1$  个值可以被产生，但并不是所有序列的排列)。如果设计中有抗随机测试码的逻辑，那么具有特定多项式的特定 LFSR 可能不会取得非常高的故障覆盖率 (例如，8 输入的与非门需要 7 个输入信号为逻辑电平 1，一个信号为逻辑电平 0 的测试码才可将逻辑电平为 0 的输入唯一地检测到，显然，这在随机码中不是那么容易或自然地发生的，一因此，这种类型的故障称为抗随机码故障)。为了解决这一问题，可用一种嵌入式的确定性向量生成器，但目前需要大量的分析数据，一般会带来显著的面积增加。

PRPG LFSR 基于本征多项式，可以生成  $2^n - 1$  种状态 (除了全 0 外的所有状态)。如果电路完全由组合逻辑构成，那么这些状态的完全解码就可检测到所有可测的固定故障。这儿有一个权衡考虑，当组合逻辑输入宽度不是太大时，32 位 LFSR 比较合理，400 位的 LFSR 就不好，因为 LFSR 越大，测试时间就越长 ( $2^{400} - 1$  要花很长时间)。

BIST 也能用来检测时序 (at-speed) 故障模型。很多的时序 (timing) 故障需要向量对转变 (transition) 来测试。PRPG LFSR 本质上是一个奇阶计数器的最小逻辑，计数顺序决定于多项式，序列顺序决定于 LFSR 中的初始值。然而，既然向量对是 LFSR 的函数，那也存在一类抗随机测试码的故障 (对组合逻辑深度为 1 或 2 或更大的组合逻辑电路都有抗随机测试码的故障)，尽管 BIST 逻辑和电路逻辑可以快速运行，AC (at-speed scan) 故障模型的故障覆盖率也是有限的。对驱动扫描链的 BIST 是如此，对作为 PRPG 和签名分析的 LFSRs 是扫描元件的 BIST 也是一样。

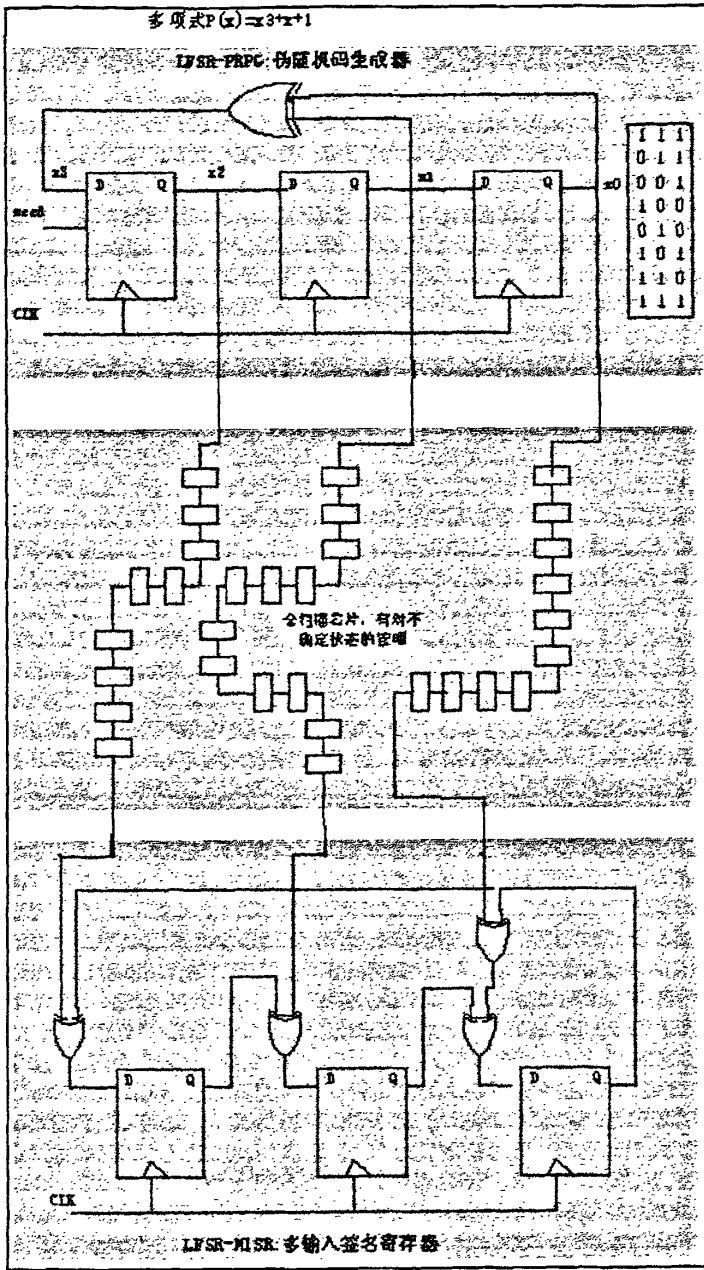


图 3.10 逻辑内建自测试电路  
Fig3.10 Logic built-in sel-test circuit

有另外一种 LBIST 存在——LFSRs 用作 PRPG, 签名分析应用于芯片而不用扫描, 或者说应用 BIST 时不用扫描。这类 BIST 就是将随机向量加在芯片的初始输入

端上，不严格限制只在芯片的内部，它们可以放在设计的任何结构层次级别上，只是设计涉及外部逻辑和面积的支持。

### 3.3.5 不确定态的处理

LBIST 另一个缺陷就是在签名寄存器上每个时钟内捕获的结果必须绝对具有确定性。如果在结果中有一个合理的 X 值（被允许的不确定状态），那么就有两个合理的签名；如果有两个 X 值，则有 4 个合理的签名；3 个 X 就是 8 个签名合理，如此类推，很快就会难以控制了。这意味着被测电路必须满足更多的 DFT 规则，而不仅仅限于全扫描约束。必须没有未被扫描的部分（存在未初始化状态），没有多环时序路径（透明锁存器，三态网络，长路径），签名分析器上数据的任何位置不存在不确定逻辑状态。

### 3.3.6 同名问题

另一个使用签名分析器所关心的问题是多个故障会产生自修复签名(self-repairing signature)。这种情况的发生就叫做同名。同名问题通常描述为两个故障产生一个正确的签名的概率。既然签名分析器可看作类计数器寄存器（签名寄存器的每个可能状态是一计数值），那么这种概率就是  $1/2^n$ 。

例如，如果一个 LFSR 是 4 位长，那么有 16 种可能的状态值。如果 LFSR 的输入连接到电路的 4 个输出口，那么它在每个时钟周期内将有一个计数值与输入数据有关。如果恰好加 16 个时钟周期，每个时钟周期对应一个不同的计数值，那么同名就是两个或多个错误映射到代表正确签名的计数值上的概率，这种情况下，概率为  $1/16$ ，通常表示为  $1/2^4$ 。

然而，同名行为不仅仅限于多故障，它对故障诊断也有蕴涵影响。如果加的时钟周期数多于 16，状态值中的某个状态就会重复出现。对于每个超过 16 的时钟周期，签名分析器将重复不同的状态值，因此，多种无故障行为和有故障行为可能被配置映射到相同的计数值（例如，一个完全正常电路需要 32 个时钟周期来测试完毕，可能会碰到所有的 15 个故障签名各两次，而通过签名表示两次）。输入给签名

分析器的数据也会将分析器引至不同的状态，可以通过给签名分析器输入数据流而使其重复地仅仅表示三四个状态。因此，如果最后签名的结果要可信，或其将用于其它形式的诊断，签名分析器的选择是很重要的。诊断过程通常由保持签名记录或签名字典作为一个查找表来完成（确定单故障生成确定签名）。

签名分析器的另一问题是“零输出”（Zero-ing Out）问题。这是由于重复 LFSR 的状态值，导致诊断信息丢失的另一种情况，这时的重复状态值是 0(或其它)，可看作是对 LFSR 记录的清除。例如，如果一个电路故障输出数据恰好是本征多项式(natural polynomial)值，那么，签名分析器将 0 输出一返回到它的初始值（注意：签名分析器 LFSR 可以有全零状态；PRPG LFSR 则不行）。如果签名分析器保持“零输出”，电路响应的记录就丢失了，签名结果变得很少。

同名、计数过多（overcount）和“零输出”的概率随 LFSR 长度增加而降低，因为它定义为  $1/2^n$ 。因此最小化这些问题的解决方案就是使签名分析器比响应字长些（宽些）—例如，给 16 位数据总线采用 20 位 LFSR，这样将计数值从  $2^{16}$  增至  $2^{20}$ ，提供了更多可能的签名值。

### 3.4 一种基于 BIST 的矩阵扫描测试方法

这里介绍一种基于 BIST 的故障诊断策略，它是通过对触发器阵列扫描，能将检测到有故障的 CUT(Circuit under test)，测试码和与之相应的响应同时找出，从而能应用传统的非 BIST 设计故障诊断方法来定位出故障门。它克服了传统基于 BIST 故障诊断方法中要么数据量太大[18, 22]，要么由于使用经过压缩处理的数据而带来的不确定性等缺点。并且电路结构简单可行，相应的算法也易于实现，是针对规模大而复杂的 VLSI(Very large scale integrated-circuit)的一种好的故障诊断策略。

#### 3.4.1 硬件环境

图 3.11 所示为多路扫描可测性设计建立的一个 BIST 硬件。由 ATPG 产生的测试码从  $n$  条链扫入，对  $n$  个 CUT 进行测试，然后将测试响应从  $n$  条链加载到一个多输入特征寄存器 MISR 中，获得特征码，然后扫描出来作为下线分析用。



如果这  $n$  条链等长, 且为  $m$ , 即每条链由  $m$  个触发器所组成, 被扫描到的触发器在概念模型上是一个  $n \times m$  阵列, 在作测试分析时一般只是利用了其列形成的扫描链路, 称列链路为主链路。如果将这个阵列中没有关联的行向触发器也链接起来, 那么在行向也可形成  $m$  条链路, 称之为从链路。对从链路的测试响应结果做特征分析, 也可达到测试的目的。为了故障诊断的实施, 对图 3.11 的电路结构作如下改进。在主从链路上分别设立一个多路单输入特征寄存器 MSISR(Multiple single-input signature register), 同时增加少量的控制电路, 让主从链路能同时对一测试序列的响应作特征分析, 然后将各条链路的特征同时扫描出来作为下线故障诊断的基本信息。这就是本文所要提出的新的硬件模型, 如图 3.12 所示。

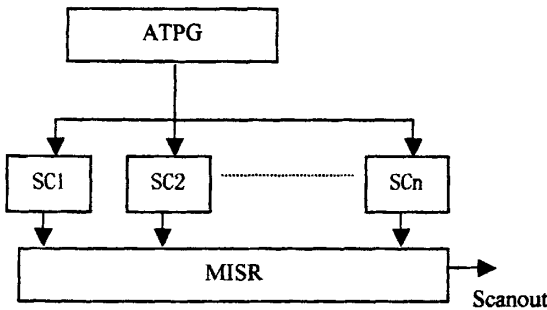


图 3.11 多路扫描 BIST 模型  
Fig3.11 Multi-scan BIST model

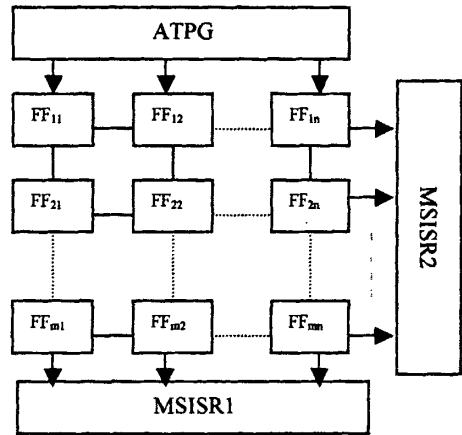


图 3.12 触发器矩阵扫描示意结构  
Fig3.12 Schematic structure of flip-flops matrix scan

### 3.4.2 数学模型

建立一个数学矩阵对应物理上的扫描触发器, 通过对数学矩阵的生成, 分析处理, 提取信息来判断触发器的故障捕获状态, 从而诊断故障。

记矩阵  $A_{m \times n}$  为触发器阵列,  $A_{m \times n}$  中的元素  $a_{ij}$  对应触发器阵列中具体的一个触发器  $FF_{ij}$ , 所在位置为从链路第  $i$  条, 主链路第  $j$  条上。 $a_{ij}$  的值代表  $FF_{ij}$  的状态,  $a_{ij}$  可以取 0 或 1。其中

$$a_{ij} = \begin{cases} 0 & FF_{ij} \text{ 没有捕获故障响应} \\ 1 & FF_{ij} \text{ 捕获故障响应} \end{cases}$$

设 MSISR1 在主链路上对  $n$  个测试向量的响应序列进行压缩后形成  $n$  个特征序列分别记为  $S_{d1}, S_{d2}, \dots, S_{dn}$ , 组成特征向量  $S_d = [S_{d1}, S_{d2}, \dots, S_{dn}]$ . 同样, 在从链路上 MSISR2 中形成特征序列分别记为  $S_{w1}, S_{w2}, \dots, S_{wn}$ , 组成特征向量  $S_w = [S_{w1}, S_{w2}, \dots, S_{wn}]$ .

为了便于分析, 这里定义一个操作  $E$ , 操作对象为特征序列, 操作结果为 0 或 1, 分别表示被操作序列没有或捕获到故障响应。

$$T_d = E(S_d) = [E(S_{d1}), E(S_{d2}), \dots, E(S_{dn})]$$

$$T_w = E(S_w) = [E(S_{w1}), E(S_{w2}), \dots, E(S_{wn})]$$

这样, 在操作  $E$  的作用下, 就将特征向量  $S_d, S_w$  变换成了在  $(0, 1)$  状态空间上分别标志各自扫描链路上有无故障响应被捕获到的信号向量。假定某一故障在触发器  $FF_{ij}$  上得到响应, 那么  $S_{dj}$  和  $S_{wi}$  将同时捕获到此故障状态的响应, 因此

$$T_{dj} = E(S_{dj}) = 1$$

$$T_{wi} = E(S_{wi}) = 1$$

而此时  $a_{ij} = 1$ , 刚好可由  $T_{dj}$  与  $T_{wi}$  的逻辑与来标志, 即  $a_{ij} = T_{wi} \cdot T_{dj}$ ,  $\cdot$  表示逻辑“与”。由于  $i, j$  具有任意性, 也就是触发器阵列上任意触发器  $FF$  捕获有故障响应, 都能在向量  $T_d$  和  $T_w$  上同时得到反映。由  $T_d$  和  $T_w$  按逻辑“与”可生成一矩阵。

$$T_w^T \cdot T_d = (T_{wi} \cdot T_{dj})_{n \times n}$$

如果将这一矩阵与触发器阵列对应起来看, 就会发现触发器阵列上每个触发器的故障响应都会在这个生成的数学矩阵上的相应的元素上用 1 得到标志。具体处理数据的算法描述如图 3.13 所示。

算法描述:

- w(1) 如果候选测试码不空, 加测试码, 收集特征, 生成  $S_d, S_w$ , 更新候选码状态; 否则转 4
- w(2) 检测  $S_d$ .
  - if  $|S_d|=0$ , 必要的话更新各链的 **fault free** 集合, 转 1
  - if  $|S_d|=1$ , 存储故障主链路号和  $S_w$  及其测试码, 转 1
  - if  $|S_d|>1$ , 记下各故障主电路号及其测试码和  $S_w$ , 更新 **multi-fault** 集合
- w(3) 处理 **multi-fault** 困难, 则转 1; 否则选取一个 **multi-fault**, 给部分主链加 **fault free** 测试码使  $|S_d|=1$ , 收集  $S_d, S_w$ , 转 2
- w(4) 如果 **multi-fault** 集合不空, 转 3; 否则结束。

图 3.13 矩阵扫描算法描述

### 3.4.3 性能分析和算法结构改进

#### 1 扫描矩阵的建立

多路扫描一方面是缩短扫描链路的长度, 从而缩短了扫描时间, 也有利于测试码的自动生成; 另一方面也是将整个大的电路划分成不同的小快, 这样既增强了电路的可控性和可观性, 也使各条链路上的测试码变得更容易兼容起来, 增强了测试的并行性。

对于某一具体电路, 由可测性设计要求的 BIST 假定要扫描  $a^2$  个触发器, 按照本文提出的触发器阵列形式形成扫描通路, 则有

$$n \times m = a^2$$

对于一组测试向量, 收集到的特征数为

$$S = n + m \geq 2\sqrt{n \times m} = 2a, \text{ 当 } n = m = a \text{ 时取等号。}$$

所以, 在相同的扫描触发器数目下, 当  $n = m = a$  时需要处理的数据量最少。

从定性的角度看, 一个方向上的链路长度增加, 必然导致另一方向上链路长度的缩短, 如果太短也不利于数据的压缩; 另一方面, 主链路加长, 每条链路捕捉到故障响应的机率就更大些, 也就是  $P(|S_d| > 1)$  增大, 同样, 主链路变短会使条数  $n$  增加, 也会使  $P(|S_d| > 1)$  增大, 所以理想情况下  $n = m = a$  为最优。不失一般性, 考虑到

其他因素，诸如电路规模、故障概率分布、故障覆盖率、触发器位置等等，只是尽量构成一个行列相差不大的矩阵即可。

## 2 扫描结构改进

由于在从链路方向要将各触发器的响应一一扫描压缩，在硬件和测试时间上都是一个不小的花费。特别在故障不是很多时，在实际应用中就不太合算了。

在图 3.12 所示结构中，可以将概念上的 MSISR2 去掉，而在 MSISR1 中再添加一路扫描压缩特征分析器，专门负责从链路的扫描压缩。同时，从链路上的触发器也不用真正在物理上相连了，而只需将主链路末端的触发器互连成一个循环移位寄存器。结构如图 3.13

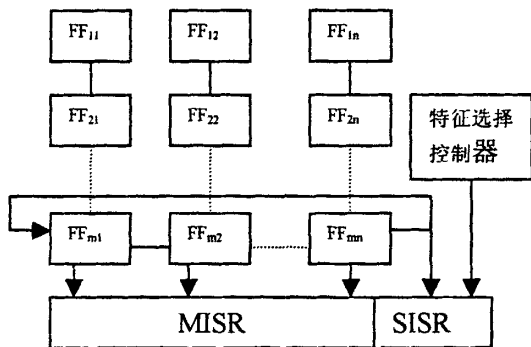


图 3.14 简单的主从链路扫描结构

Fig3.14 Simple structure of the main and second scanning path

只要在选择主链路时稍加考虑各链末端触发器的位置，就可很容易得到图 3.13 的结构。由于从链路扫描分析的硬件结构可做得非常紧凑，实际上它只是对一条被选定的末端触发器链的重复扫描，它的扫描速度可以设计得高些。每次在主链路扫描出各位之前，扫描一条从链，这由一个控制器将主频  $n$  倍频专供扫描从链路用。这样就实现了整个矩阵扫描的无间隙工作，甚至丝毫不增加原来仅做故障测试的时间花费。这种结构与传统的 BIST 结构相比，在硬件上并不需要太大的花费，基本上保持原来的风格；在线测试处理时间也没有增加，只是在有多故障出现时才会有针对性地加长测试时间。

### 3.4.4 数据处理及其性能分析

对于没有检测到故障的情况，我们的硬件结构和算法都不会带来额外的时间复杂性，它的时间花费就跟平常做测试所必需的时间花费一样。如果电路出现单链路故障，运用我们的算法可以迅速准确地定位，在线时间花费也跟平常做测试所必需的时间花费一样，而下线分析定位时间由于算法的简单性变得只需要存储少量的几个数据而已。只有在出现多链路故障时，才会在在线测试和数据采集上有一个循环迭代过程。而这个过程所造成的测试时间增加基本上正比于  $|S_d|$ 。从下面分析可知，这个比例系数是可以小于一般单链路故障的花费时间。

在出现多链路故障(multi-fault)同时被几条链路捕获时，由于  $|S_w| > 1$ ， $|S_d| > 1$ ，我们要先记下使  $S_{w_i}=1, S_{d_j}=1$  的主从链路号及其测试码，再加 fault free 测试码来屏蔽某些故障使  $|S_d|=1$  后，只需将先前  $S_{w_i}=1$  的从链路的特征记下作为下线分析，而其他的就可以不作处理，直到此次多故障捕获响应的触发器全部定位完毕。因为每条主链路都可能有故障响应，特别在多故障出现时，我们要给某些故障链路加 fault free 测试码，所以测试一开始就要为每条主链路收集 fault free 测试码，直到每条链路都有为止。收集 fault free 测试码可由  $S_{di}=0$  来判断。可以为每条链路多收集几个供需要时挑选用，这可方便 ATPG 来生成和利于诊断的实施。当测试码加完还有 multi-fault 时，一般由于总有 fault free 测试码的存在，最终是可以处理掉的。如果由于测试码选取的不完善造成有 multi-fault 无法处理，就会在 1,3,4 之间出现死循环。这时可在任何一处设立计数器判断这种状况作中断处理并报告错误信息。

由于制造工艺引起的失效往往会在电路的某一区域块中表现出来，加上逻辑时序的瓶颈也只是部分电路所造成，只要 CUT 具有适当的规模，如果多故障出现，它们在同一 CUT 的概率较大，也就是多故障更偏爱在同一 CUT 中。那么，相同条件下，单链路故障的机会就比多链路故障大。另一方面，即使有多个 CUT 存在故障，但对选定一组测试码，要同时有故障响应，也是一个小概率事件。这样，我们提出的算法和电路结构就显得更有效。

在对主从链路的特征分析后生成向量  $S_d$ ， $S_w$ ，可以不作生成矩阵处理，就用两个数组将他们分别存储即可，矩阵只是在概念上由它们生成的。最终我们得到的故

障信息就是：主链路号、测试码和向量  $S_w$ ，有了这些就可结合 CUT 的结构做最终的故障诊断了。

### 3.5 小结

这一章较为全面地介绍了当今时序电路的测试方法。其中扫描通路的思想是许多具体可测性设计技术的基础，也是时序电路内部的可控性和可观测性得到改善的原因。

扫描路径的建立是一个比较重要的步骤，直接关系到测试性能和电路的系统性能。有很多这方面的研究文献作出了大量的探讨，也提供了一些很好的方法[9,20]。在这一章把比较常见而有效的扫描路径设计技术系统地作了介绍，并且分析了它们的特点和一些设计中要权衡考虑的因素。

适应集成电路飞速发展的可测性需要，一种基于扫描的内建自测试方案应运而生，我们称之为 BIST。它是将测试过程中需要的测试码生成，扫描控制，测试响应分析集成在被测电路中的测试方式，可以让电路运行在正常和测试两种模式下。BIST 是一种非常有前途和价值的可测性设计思想，不仅为超大规模集成电路的测试提供了一种解决方案，而且还为如今出现的知识产权 (IP) 问题给出了解决途径，也为今后实现系统级的测试提供了接口标准的可能。

我们对 BIST 方法基于的原理和电路结构作了详细的分析和研究。最后我们提出了一种矩阵扫描的 BIST 故障诊断策略，它是建立在对触发器阵列的二维扫描的基础上。通过对特征的直接分析来正确获得捕获故障的链路上的触发器的状态信息，从而确定完整的故障响应序列，避免了通过特征的解码进行故障诊断所带来的麻烦和不确定性。这种方法所带来的硬件花费为一个 MSISR，一个控制选择器。MSISR 是由多个 SISR (single-input signature register) 并列集成实现的，由于很规则，连线简单，容易实现，占面积不会大。测试时间大部分用在收集特征上，如果碰到多故障，会有一个重复提取的过程，但改进算法自动识别数据的收集，会使这个过程花费的时间很少。

## 第四章 静态电流测试法

集成电路日益增长的复杂性和特征尺寸的不断缩小, 需求更加有效的测试策略来满足高的质量可靠性。传统的基于电路输出电压的逻辑观察的测试技术不足以满足越来越多的集成电路产品的低故障漏检率要求。于是, 寻找电压测试技术的对偶方法来捕获从逻辑观察法中漏检的故障变得非常具有现实意义[4]。通过观察集成电路芯片的静态电流耗损的测试方法是被看好为非常有前途, 也是当今最先进的测试方法, 一般称之为  $I_{DDQ}$  测试法。

### 4.1 概述

CMOS 制程工艺的进步使得在单硅晶片上集成更多器件成为可能。现在制造生产的 ICs 已达到了几年前根本无法想象的复杂性, 但其测试成本的增长明显快于芯片尺寸的增长。随着更多应用领域要求可靠的质量保证的发展趋势, 测试已成为设计深层考虑关心的问题, 未来的测试技术将面临的问题是如何在非常复杂的集成电路芯片上提供有效的测试手段来保证非常低的故障漏检率, 而且测试成本保持在较低的水平。

$I_{DDQ}$  测试是基于数字 CMOS 集成电路的静态电流消耗的观测, 如果同传统的电压逻辑观测法结合起来, 能够取得比使用单一方法较好的质量。事实上, [23]研究表明,  $I_{DDQ}$  测试能检测到不被任何其他方法检测到的唯一故障。正是因为有了这样的优点, 所以设计工程师们对  $I_{DDQ}$  测试作了很多深入的研究, 提出了一些很好的技巧和解决方案。

如果将  $I_{DDQ}$  测试与传统的电压测试方法结合起来, 互为补充, 将会收到更好的效果。现在发展的电流签名技术,  $I_{DDQ}$  差值法, 无疑为  $I_{DDQ}$  测试提供了更强的生命力。特别是与逻辑内建自测试相呼应,  $I_{DDQ}$  测试也可适应嵌入式, 这为今后的超大系统级电路提供了良好的测试解决途径。在一些涉及知识产权的电路领域, 这是很重要的特性, 甚至容易促成测试的标准化。

$I_{DDQ}$  测试在未来深亚微米 CMOS 工艺上的应用将会更加广泛。

## 4.2 无故障 $I_{DDQ}$ 电流

静态 CMOS 电路的无故障  $I_{DDQ}$  电流由漏电流引起，这些漏电流由不同的物理机理造成。nMOS 和 pMOS 晶体管中的静态漏电流如图 4.1 所示。根据这些电流所产生的机理可将其归类如下 5 种：

1. 反向偏置 pn 结漏电流  $I_D$
2. 亚阈值电流（弱反向） $I_{DS}$
3. 门栅在漏极到衬底之间感应的漏电流  $I_{GIDL}$
4. 体击穿电流  $I_{PT}$ ，由漏极流向源极，它是由于形成了对应双极晶体管造成的一漏极对应发射极，基体对应基极，源极对应集电极。
5. 穿过门栅和漏极之间的氧化层的沟道电流  $I_G$

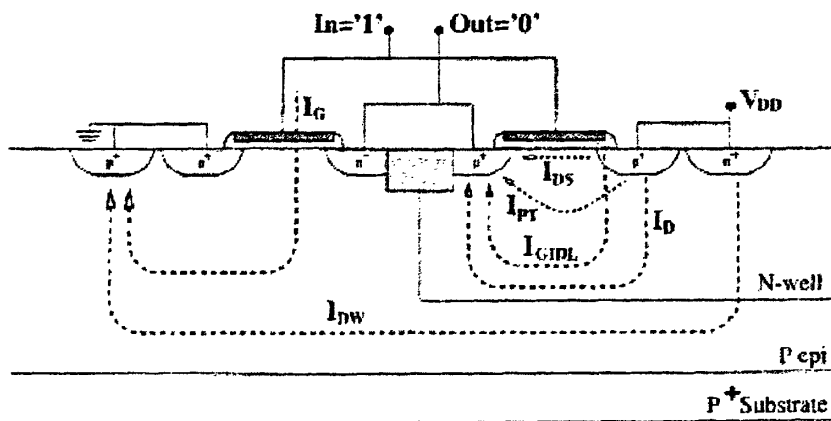


图 4.1 无故障的  $I_{DDQ}$  电流组成  
Fig4.1 Defect-free  $I_{DDQ}$  current components

在目前 CMOS 工艺水平下，弱响应亚阈值电流  $I_{DS}$  是  $I_{DDQ}$  的主要组成部分，在将来 0.1 $\mu\text{m}$  以下的深亚微米工艺水平下，由于沟道效应，其他组成部分将会变得重要起来，而且还会使整体的  $I_{DDQ}$  变大，这在后面会讨论。



### 4.3 $I_{DDQ}$ 测试基本原理

在低静态电流( $I_{DDQ}$ )的数字 CMOS 工艺下, 大部分常见的故障都会引起  $I_{DDQ}$  明显的增加, 这就是允许通过比较芯片静态电流与预先设定的阈值  $I_{DDQ}$  来区分有故障和无故障芯片的基本原理。如果被测试电路消耗的静态电流比阈值小, 那么该芯片就被接受为无故障的; 否则就判为有故障[24]。图 4.2 所示为一个芯片产品的静态电流直方图, 在这个图中, 有一个  $I_{DDQ}$  阈值, 处在这个阈值左边的表示是没有故障的芯片, 而处在右边的是有故障的芯片。这个图可定性地说明  $I_{DDQ}$  测试法的基本原理。给每种芯片设定正常  $I_{DDQ}$  阈值, 通过电路实际静态电流值与阈值的比较来判断电路有无故障。

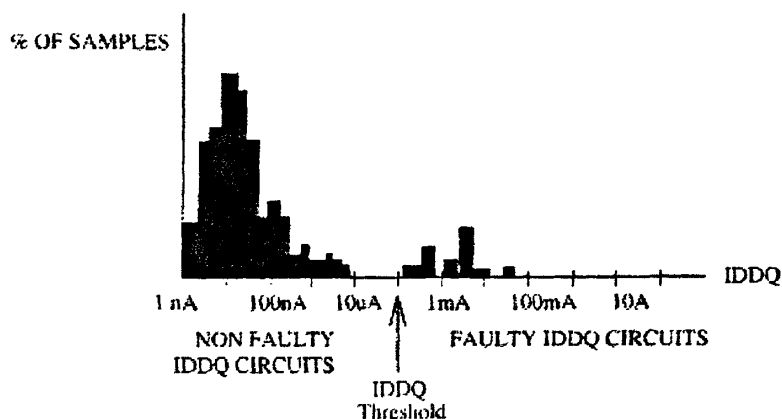


图 4.2  $I_{DDQ}$  电流分布柱状图  
Fig4.2  $I_{DDQ}$  current distribution histogram

$I_{DDQ}$  测试方法越来越为半导体工业界所接受, 在于它能检测到一些其他传统逻辑测试方法检测不到的故障的能力。同样地, 它也有测试成本需要考虑, 首先, 会使产量减少, 因为要想保证有故障的电路不被漏检, 一般  $I_{DDQ}$  阈值会选择较小些, 这样一些本来无故障的产品可能会被误检为不合格; 另外, 由于电流波形的建立较慢, 用于每个测试向量的测试时间也变长了, 而且传感电流还需要一些额外时间。这些缺点有时也可被只需要少量的高故障覆盖的  $I_{DDQ}$  测试向量补偿掉[25]; 不过, 也可应用一些技术来弥补不足, 如  $I_{DDQ}$  阈值的选择[26], 传感电路的合理设计[27]。

检测异常的静态电流消耗需要特殊的传感电路来执行所需的比较功能。目前有很多的解决方案，一般可分为片外传感器和片内传感器(built-in current sensor, BICS)。片外传感器的优点就是不占用芯片面积，不会造成延时增加，但其测试速度有限，而且其故障电流的分辨力也随芯片尺寸的增加而降低。总之，测试工程师必须认真考虑各种可能的选择来决定采用最好的  $I_{DDQ}$  测试策略。

#### 4.4 电路故障分类及其 $I_{DDQ}$ 可测性

CMOS 集成电路芯片中有很多的故障形式，根据它们对电路功能上造成的影响，可将其归为两类：

灾难性故障：在任何时候，只要将其表现出来，就会造成灾难性后果。

参数故障：它们一般只会使电路在性能上变得低劣。

由制程引入的这些超大规模集成电路芯片中的故障可能造成一些短路或断路，也就是桥接或者开路。桥接故障是指在电路中，两个或多个节点之间引入了不希望有的连接（线性或非线性的  $I/V$  特性）；开路是指电路中本应电气相连接的部分在材料层上电气不相连。尽管基于逻辑 / 电压的测试策略，如固定故障模型 (stuck-at model)，广泛用于描述由这些故障引起的故障行为，但在 1980 年，一项重要的实际故障模型研究执行下来，发现  $I_{DDQ}$  是最支持 CMOS ICs 的测试方法。

理论和实验分析表明，桥接故障，门栅氧化层短路（影响 MOS 结构中二氧化硅的短路）和连接线的断开是 CMOS 工艺中最多的故障，已有大量的研究阐述了他们的  $I_{DDQ}$  可测性。要想故障电路  $I_{DDQ}$  可测，唯一要求就是在加一个输入测试码后，其静态电流消耗异常之高（相对于无故障静态电流）且可观测。

##### 4.4.1 桥接故障

桥接故障是成熟的 CMOS 工艺中最常见的故障形式。一个桥接故障一般由不希望的两个或多个节点之间的电气连接构成，如图 4.3a。这种故障产生的影响无法由 stuck-at 模型来精确模拟，因为一个桥接节点并不是处在一成不变的逻辑电平上。根据不同的电阻值  $R_b$ ，桥接故障可用一条电阻性通路来模拟。如图 4.3c 所示，当桥接

电阻  $R_b$  变化时，节点 node A 和 node B 的电压是在不断变化的；由图 4.3b 也可看到， $I_{DDQ}$  电流是随着电阻  $R_b$  的增大而逐渐降低的。

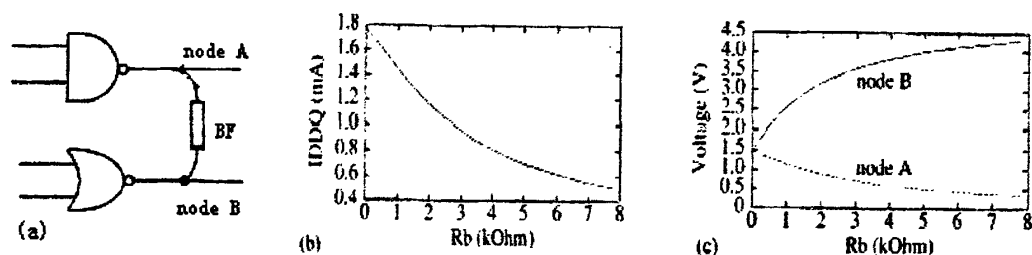


图 4.3 桥接故障及其消耗的  $I_{DDQ}$  电流和输出电压  
Fig4.3 Bridge defect and  $I_{DDQ}$  consumption and output voltage

[28]表明 3.0%的金属线间桥接电阻大于 1kohm，某些情况下上升至 20kohm。

桥接故障可能在 CMOS 电路内部产生异常高静态电流，图 4.3a 所示为在两个 CMOS 门的节点 node A 和 node B 处有一个桥接故障，为了产生一个高的  $I_{DDQ}$  电流值，必须满足条件：桥接点必须驱动到相反的逻辑电平。根据桥接电阻  $R_b$  的不同，桥上电流消耗也有些变化，见图 4.3b，node A=0，node B=1。对于  $I_{DDQ}$  环境下的测试技术，不用考虑故障对逻辑的影响，当然，这个输出逻辑在估计下级无故障电路的静态电消耗上是有用的。实际上，桥接故障对故障门的输出电压的影响可能使其呈现一中间状态值，见图 4.3c，而这个中间状态的电压在驱动下级无故障门电路时，可能就会造成很大的  $I_{DDQ}$  电流流过。这种情况下，故障的发现可以被下级无故障电路简化，如图 4.4 所示( $i_{DDQ}$ )。

如果组合逻辑 CMOS 电路中被桥接的节点不是逻辑无关的话，可能会在故障电路中形成一个反馈环。这样的反馈可能会引入记忆元件或环路震荡器，这依赖于反馈环上逻辑反向器的个数和驱动桥接节点的晶体管的大小。

在桥接故障影响时序电路的情况下，这些故障在特殊情况下可能会使  $I_{DDQ}$  测试无效[29, 30]。在这些故障电路中，桥接故障可能使记忆元件的记忆状态发生变化，从而掩蔽掉了这些桥接故障的存在，在 4.8 节中会有详细阐述。为了消除这一限制，[31]中提出了两个  $I_{DDQ}$  可测的触发器结构；在[32]中，Yamazaki 和 Miura 提出

了另外一种触发器结构，可使  $I_{DDQ}$  完全可测；[30]中提出了一种处理技巧，使触发器扫描链  $I_{DDQ}$  可测。另外，静态电流签名可改善这种可控性的限制[33]。

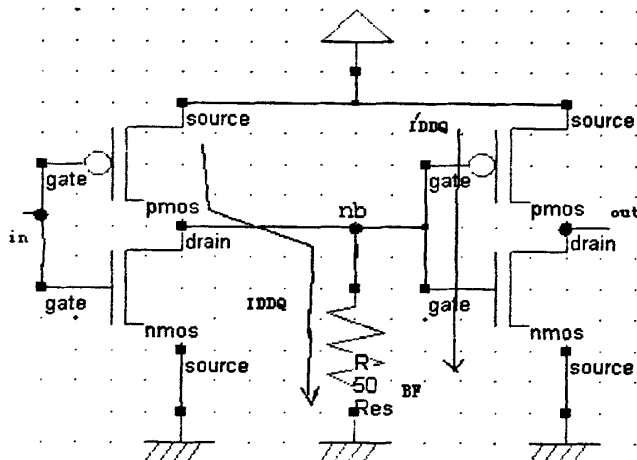


图 4.4 中间状态输入电压对无故障 CMOS 电路的  $I_{DDQ}$  电流的影响  
Fig4.4 Effect on  $I_{DDQ}$  of non-defective CMOS circuit due to intermediate input voltage

#### 4.4.2 其他 CMOS ICs 故障

另一种常见的故障形式是开路故障，由集成电路中任何相连接的介质的不希望的电气断开所造成。这种故障的阻抗通常很高，但有较大的耦合电容可能被引入。根据被开路故障所影响的节点的情况，故障电路的行为可能有所不同。如果晶体管的源或漏被断开，测量  $I_{DDQ}$  电流的通路可能就被锁住了；但是，如果开路影响的是 CMOS 晶体管的门栅（栅极是悬浮的），通过耦合电容的电容性信号的耦合作用，能产生一个可测的静态电流消耗来检测这些故障。综合考虑源漏节点和其他电路信号的影响，故障电路行为的特性在[34]中有描述。它表明了由于门电压值比阈值电压稍大后，故障电路如何表现出高静态电流消耗的。

#### 4.5 $I_{DDQ}$ 测试用电路传感器电路

为了执行  $I_{DDQ}$  测试，特殊的电路单元必须被安置在被检测电路中检测其静态电流，并将其与预定阈值比较，这些电路被称之为电流传感器。

如果这部分电路位于被测电路同一块芯片中，这个电流传感器就被称为内建电流传感器（BICS,built-in current sensor），相反，如果电流传感器位于被测电路外部，那么就称之为片外电流传感器了（off-chip current sensor）。

下面，我们首先描述一下不同类型的传感器，然后分析一些重要的参数来评价这些电路，最后概括一下设计这样的传感器的方法。

#### 4.5.1 电流传感器类型

如前面所讲，电流传感器的分类是依据其位置而定。片外传感器位于 CUT 外部，可以分开来制造；片内传感器(BICS)位于 CUT 集成电路内部，必须生产芯片同时制造出来。下面分析这两种传感器各自的优点和缺点。

除了根据位置，传感器也可由其功能或结构分类。

根据电流传感器对静态电流执行的功能，在传感器上产生的输出特性，可分为线性传感器和非线性传感器。如果一个电流传感器在传感电路的输出与感应的静态电流之间成线性关系，则称之为线性电流传感器，否则就称之为非线性电流传感器。

线性传感器还可进一步分为比例传感器—输出与静态电流成正比例关系；积分传感器—输出与静态电流的积分成正比例关系。

电流传感器也可依据其感应静态电流的构成元件分类，可命名为电阻传感器、镜像电流传感器、开关传感器和电容传感器。

电阻传感器：

    单一电阻传感器

    有一个 pn 结的电阻传感器

    带电阻和一个放大器的电阻传感器

    非线性电阻传感器

镜像电流传感器：

这种传感器用一个电流镜像电路来测量静态电流，静态电流通过镜像电路的一支，然后在镜像电路的另一支中反映出来。

开关电容传感器：

这一子类包括了所有的积分传感器。开关串联在电源和 CUT 之间，电容与 CUT 并联，提供引脚。当开关打开，静态电流使电容放电，过了积分时间后，留在电容上的电荷取决于  $I_{DDQ}$  电流的大小。

#### 4.5.2 评价 BICS 性能的参数

片内传感器的吸引力在于在同一块 CUT 芯片上集成传感器，使得由输引脚带来的电容减小，因而具有更高的运行速度，而且还使在线  $I_{DDQ}$  自测试成为可能。但同时片内 BICS 也在几个方面不利于被测电路。

- 使 CUT 性能降低
- 增加硅片面积

相反，片外  $I_{DDQ}$  测试传感器正成为商业测试设备的一种标准特征，而且正在努力使其标准化，不过，这可能只是一个难以达到的期望而已，因为片外传感器的缺点限制了它，如只能在 IC 制造过程中被测试，测试速度慢等。

基于上面的原因，设计者需要努力提高电流传感器的如下特性：速度、感应能力、低压降、尺寸小、最小化。需要设法减小电流传感器对硅片面积的影响(BICS)和对 CUT 性能的伤害。在规格说明和电流传感器的评估或设计之间权衡取舍，选择最佳的传感电路来执行  $I_{DDQ}$  测试任务。

#### 4.5.3 一个电流传感器实例

图 4.5 所示电路中虚线部分是一个比例电流传感器，它集成在电路内部 (Proportional BICS, PBICS)。它利用一个 CMOS 兼容的共基配置双极晶体管连接在旁边，可以让 CMOS 管中一部分的  $I_{DD}$  电流从被测电路中分流到一个电阻上，将其转变为一个可测电压  $V_{sens}$ ，然后将这个电压与外部参考电压  $V_{ref}$  比较，产生一个有无故障的信号  $f_{sens}$ 。

#### 4.6 $I_{DDQ}$ 测试的测试码自动生成

$I_{DDQ}$  测试克服了传统的逻辑测试或布尔测试方案面对实际故障模型多而面临的许多问题，特别是为了检测当前 CMOS 工艺中常见的故障形式—桥接故障(BF)，只需将桥接的两个故障点驱动到相反的逻辑电平就足够了。然而，两类关系到桥接故障的测试码自动生成的问题依然存在。首先，就象在逻辑测试中一样，实际电路中大量的可能故障迫使 ATPGs 要么设计出更精准的故障表示列表，要么先于测试码生成之前使用故障提取程序。然而，两种方法要想得到高的故障覆盖率，都非常耗内存和执行时间。另外，由于需要较长的时间来测量静态电流，尽可能使  $I_{DDQ}$  测试码集精简是有利的。

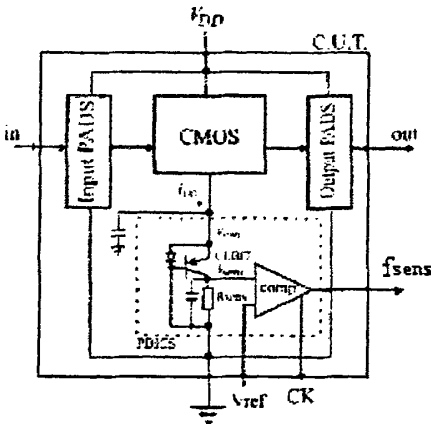


图 4.5 片上比例电流传感器框图  
Fig4.5 PBICS block diagram

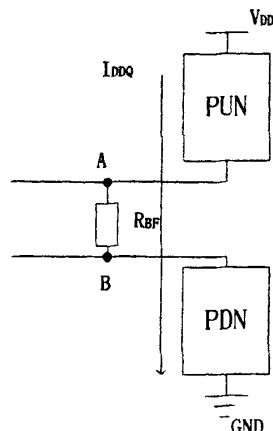


图 4.6 连接上拉和下拉 CMOS 电路的桥接故障  
Fig4.6 Bridge fault connecting pull-up and pull-down CMOS circuit

总之，这方面的研究面临的问题就是在执行时间、内存需求、故障覆盖率和测试码集大小上取得一个优化的平衡。为了展示解决这些富有挑战性的折衷处理的不同途径，下面将阐述面向 BFs 的最新  $I_{DDQ}$  ATPG

许多  $I_{DDQ}$  ATPG 方法是针对组合电路设计的，当然稍加修改也能应用于时序电路中。

将桥接故障进行分类考虑是一种首选策略，运用电路的布局信息提取实际故障情况，分别为门电路的外部桥接和内部桥接生成测试码。[35, 36]在此过程中还考虑泄露电流模型和伪固定故障模型。这样需要考虑的故障的数量少了很多，因为毕竟只有较少的节点对之间可能在物理上出现短接的可能。当然，它的不方便处就是要事先知道电路的布局信息，还需要时间和存储空间来提取可能的故障信息。

如果不在晶体管级电路上提取可能的故障信息，而在逻辑综合后的电路网表上产生可能的故障集，则需要的运算要简单得多，而且效果也不错。

至于测试向量的具体生成，也有很多不同的方式。[37]提供的方法是从逻辑测试用的测试向量中选择定量的子集来完成电流测试。[38, 39]运用随机的方法来产生测试码，不过只选择那些真正有用的测试码来提高故障覆盖率。根据电路本身的逻辑结构，也可确定性地运用具体算法来产生测试码，如利用后面介绍的逻辑综合数据结构信息，这个过程和为固定故障产生测试码没有两样，是针对特定的BFs进行的。如图4.6，要想检测这个BF，只需寻找合适的逻辑向量，将模块PUN打开，使故障节点A与 $V_{DD}$ 直接相连；模块PDN也打开，使故障节点B直接与GND相连。这样在 $V_{DD}$ 和GND之间就形成了一条电阻性通路，消耗的静态电流约为  $I_{DDQ} = V_{DD}/R_{BF}$

#### 4.7 低压 CMOS 电路的 $I_{DDQ}$ 测试

为了实现便携式计算和无线通讯系统的需要，现在很多电路要求功耗特别小，除了加强电路的设计技术，运用电源管理策略外，一般还降低电路工作的电压。按照比例降低 CMOS 电路的工作电压是一种有效的降低功耗的策略，因为功耗是与电压  $V_{DD}$  的平方成正比的。特别在集成电路进入深亚微米阶段，电路的工作电压随着器件特征尺寸的缩小，电路的正常工作电压也必须同时比例降低。然而为了维持电路的高性能，器件的阈值电压也必须同时比例缩小，这样造成的结果是电路的漏电流却极大地增加，这是由于在弱反向区，漏电流与阈值电压是成指数关系的。所以，低压 CMOS 电路的  $I_{DDQ}$  测试由此遭到挑战。



### 4.7.1 阈值电压降低对静态漏电流的影响

$I_{DDQ}$  测试基于的原理可简单地表述为无故障 CMOS 电路在稳定静态下流过电路的电流非常小，而在桥接故障下造成电路中有一条从  $V_{DD}$  到地的通路，使得这是的静态电流非常大，从而可以清楚地分辨出电路的故障状态。但在低压 CMOS 电路中的固有漏电流极大地增加了，这使得  $I_{DDQ}$  测试难以区分电路的故障状态。为了继续在深亚微米集成电路中运用  $I_{DDQ}$  测试技术，有很多解决方法被提出，如冷凝措施 [40]、应用衬底偏置技术[41]、漏电流控制技术[42]、电路划分等等。

由图 4.1 可知，漏电流主要有两个根源，反向偏置节漏电流  $I_D$  和亚阈值漏电流  $I_{DS}$ 。晶体管尺寸缩小，可大大减小节漏电流  $I_D$ 。然而这种缩小要求供电电压也比例缩小，最终造成器件阈值的比例缩小。而亚阈值漏电流与器件的阈值电压成指数关系，随着阈值电压的比例缩小而指数增加。最终使得亚阈值漏电流成为目前 CMOS 工艺下的主要漏电流。亚阈值漏电流  $I_{DS}$  跟阈值电压  $V_{th}$  的关系见图 4.7。

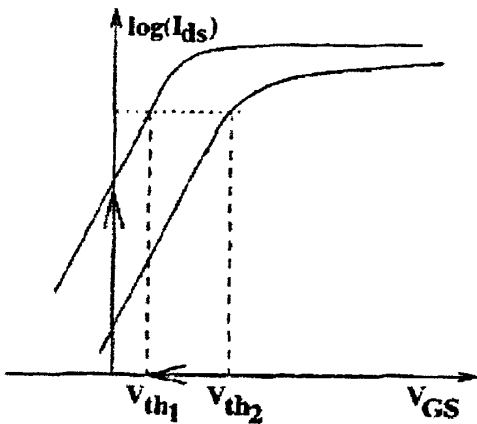


图 4.7 阈值电压降低导致漏电流的增加  
Fig4.7 Scaling  $V_{th}$  results in the increase in leakage current

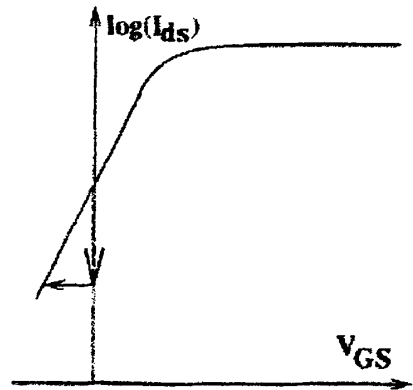


图 4.8 负的  $V_{GS}$  使漏电流降低  
Fig4.8 A negative  $V_{GS}$  leads to a great reduction in leakage

### 4.7.2 漏电流控制技术

#### 1. 门电路的输入向量控制

对于有多输入的门电路，在 CMOS 工艺下，不同的输入组合可能会引起不一样的静态电流流过。假设有一个二输入与非门，在输入为“1 1”时，两个 NMOS 管导通，流过的漏电流是上面两个 PMOS 管漏电流之和。当输入为“0 1”、“1 0”或“0 0”

时,至少有一个 NMOS 管是截止的,而在下拉网络中的截止管处于上端时,它的源电压  $V_S$  通过一个导通的 PMOS 管直接与  $V_{DD}$  相连,所以为正。在静态时,通过所有 NMOS 管的漏电流相等,所以只要考虑上端的截止管子就可以了。当  $V_S$  为正时,意味着  $V_{GS}$  为负,从图 4.8 的  $I_{DS}-V_{GS}$  特性曲线可知,这将大大减少漏电流。正的源电压同时说明存在体效应和  $V_{DS}$  的减小,他们都将增加阈值电压,从而降低漏电流。

对于每个门电路,它的静态电流都依赖于它的输入的组合,这使得这个电路的静态漏电流决定于它的初始输入。因此,给电路加一些合适的输入向量,可以显著地减小漏电流。最直接的办法就是枚举电路的输入组合,对于大电路这是不可取的。有一些随机搜索方法,可以找到最佳输入组合。

## 2. 双阈值电压设计

前面已提到,静态漏电流的增加主要是工艺条件下阈值电压的降低导致其指数上升。利用多阈值技术可以改变这种状态,最容易实行的就是双阈值策略。

在逻辑电路中,双阈值技术可以用来既减小系统漏电功耗,又降低系统的漏电流。在一些非关键路径上,采用较高的阈值电压,可以减小漏电流;在关键路径上,采用低阈值电压,维持系统的高性能。这样在不需额外电路的情况下,同时取得高性能和低功耗的目的

鉴于深亚微米集成电路的复杂性,并不是所有的非关键路径上的晶体管都可以采用高阈值电压的,否则,关键路径可能改变,从而造成整个电路的时延增加。为了在保证系统性能不受损害的情况下最好地节约漏电功耗,可以利用遗传算法来选择和分配高阈值电压。一是可以考虑电路中需要高阈值电压的部分,另外可以考虑不同的高阈值电压。双阈值电压可以通过通道植入和体偏置实现。

## 4.8 扫描链路的 $I_{DDQ}$ 测试

运用基于扫描链路的 DFT,或  $I_{DDQ}$ ,或两者都有,来测试时序电路并不能保证依赖于故障电阻和电路电平参数的桥接故障一定能检测出来。然而采用“透明”扫描链路技术,测试仪能使两种方法都有效地检测制造过程中的故障,包括扫描链路

中难于检测的短路故障。这里介绍的一种使扫描链路透明的策略，不管触发器有多少个，能使链路测试复杂性非常小。

基于扫描的逻辑测试在 70 年代后期和 80 年代改变了 IC 测试的复杂性。它们能将一个分布的时序逻辑转变为一个大而统一的移位寄存器，减少了测试的复杂性。正是由于这些技术，使复杂数字电路测试码生成成为可能。

同时，工程师们发现经典的基于电压的测试方法并不能保证数字 CMOS ICs 足够的质量水平。对质量的追求给数字 CMOS ICs 的测试流程带来了新的测试技术——静态电流测量法(QCM)，或称之为  $I_{DDQ}$ 。

扫描测试要能顺利进行，首先必须保证扫描路径行为的正确性。用电压测试方法的对偶方法来检测用于电压测试的扫描元件上的故障，应该能取到相得益彰的效果。本小节要阐述的内容是，如果将触发器电路“透明化”，从而令  $I_{DDQ}$  和电压测试方法能将时序电路中的绝大多数桥接故障检测出来。一个透明的时序电路是其内部输出直接进入其内部输入端而不受时钟的影响。透明的概念也可延伸到芯片级，在芯片级实现透明扫描的成本比局部实现还要小得多。

#### 4.8.1 时序元件的 $I_{DDQ}$ 测试问题

图 4.9 表示了一个典型的 CMOS 工艺下的主从触发器的实现结构，采用单相时钟。当时钟为低电平时，传输门 TG1 和 TG4 处在导通状态，TG2 和 TG3 处在非导通状态，因此，此时触发器的主锁存器从数据输入端接收新的数据，而从锁存器保持原来的数据。在时钟向正极性转变时，主锁存器不再接收输入数据而将当前数据送给从锁存器，按这种方式，触发器实现了主从操作。

现在  $b_1$  与  $V_{DD}$  间有个桥接故障，可能是由门栅氧化引脚孔或其他材料错误引起的。在适当的稳态输入激励下，这些桥接错误会产生异常高的电流，在  $V_{DD}$  和  $V_{SS}$  间产生一个直流路径， $I_{DDQ}$  能检测到他们。然而，要确定有这样的错误存在，流过这个直流路径的电流必须比电路的各种漏电流要高些，亦即  $I_{DDQ}$  不能检测到不能在  $V_{DD}$  和  $V_{SS}$  间以任何形式的稳态输入激励产生大静态电流的错误，例如， $b_1$  与  $V_{DD}$  间的桥接故障的电阻很小，则不会产生高的静态电流，那么  $I_{DDQ}$  不能检测到它。

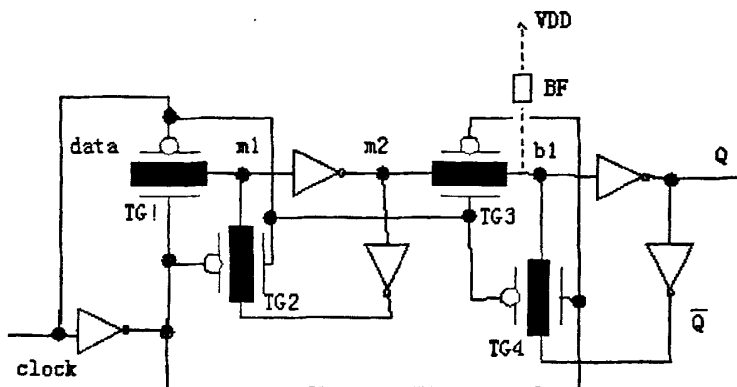


图 4.9 一种有桥接故障的典型 CMOS 触发器  
Fig4.9 A typical CMOS flip-flop with bridge-fault

对于图 4.9 的电路，时钟在正跳过程中，TG1 和 TG4 本来正处在导通状态，现在停止导通；TG2 和 TG3 本来正关闭着，开始导通。现在节点  $m_2$  驱动节点  $b_1$ ，在此之前一直由节点  $\bar{Q}$  通过 TG4 来驱动。因为 TG1 关闭，TG2 打开，节点  $m_2$  的输入节点  $m_1$  自己进入一个转变阶段，因此它的驱动能力有限。在无故障情况下，通过一对前后端相连反向器的正反馈，触发器能渡过这个过度阶段。然而现在，由于桥接故障，节点  $b_1$  被单独驱动到  $V_{DD}$ （或  $V_{SS}$ ）。对于一个低阻桥接故障，通过这个故障的电压驱动能力要比  $m_2$  强多了。结果，这个驱动将主锁存器内容重写。因此，稳态下无电流流过， $I_{DDQ}$  不能检测到这个故障。

用  $I_{DDQ}$  检测 CMOS 触发器中的桥接故障的问题在本质上与不能检测用传输门或非门实现的触发器上的桥接故障的原理是相同的，因为在 CMOS 工艺下，设计者惯用开关或传输门实现触发器，利用它们交替地开和关来保证触发器的主从操作。但开关门的双向传输特性使得  $I_{DDQ}$  不能检测这样的触发器中的桥接故障。

#### 4.8.2 触发器 $I_{DDQ}$ 可测的解决方案

如果故障节点间的逻辑冲突在稳态下存在且保持着， $I_{DDQ}$  就能检测出这个桥接故障。这种情况下，只要通过故障的驱动不改写触发器主锁存器的内容，这种冲突就继续存在着。可以通过如下途径取得：

1. 用适当的数据初始化主锁存器，保持时钟低电平。用外部控制信号 Test 打破主锁存器中的反馈环，最后，改变时钟至高电平，让 TG3 开始导通。
2. 在触发器数据输入端保持适当的数据值，外部控制信号 Test 和时钟同时使所有的 TGs 导通。

第一种方案要求在主锁存器反馈路径上增加 TG、反向器和控制信号 Test，而第二种方案使触发器透明，它是较好的选择，因为此时，触发器内部所有节点被稳定地驱动到高电平或低电平。

通过在电路结构上增加测试模块使触发器透明，以便在测试模式下，所有的 TGs 同时导通，这样保证输入的数据总是驱动 m1、m2 和 b1。

图 4.10 表示了这个概念和一种可能的实现方案，不过需要额外的测试逻辑电路。将时钟 *clock* 及其反相信号  $\overline{clock}$  通过测试逻辑块与 TG2 和 TG3 连接，测试控制信号控制触发器在正常和测试模式下的切换。这个方案需要一个或非门和一个反向器来实现。

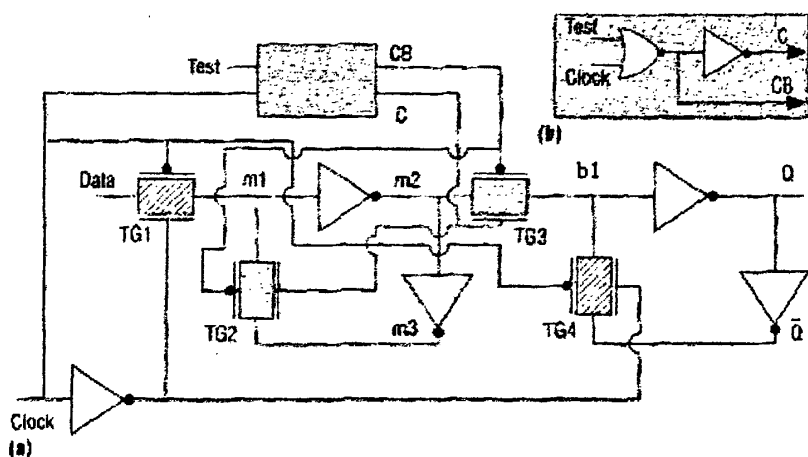


图 4.10 使触发器透明化的概念及其测试块的实现  
Fig4.10 Concept of transparent flip-flop and the implementation of test block

在外加测试逻辑块的控制下，当  $Test = 1$  时，电路处在测试模式下，信号 C 被固定在高电平而 CB 在低电平，不受时钟信号 *clock* 逻辑电平的影响。这样保证了在测试模式下，TG2 和 TG3 导通。如果时钟信号 *clock* 处在低电平，TG1 和 TG4 也导通，因此，在测试模式下，触发器是透明的。

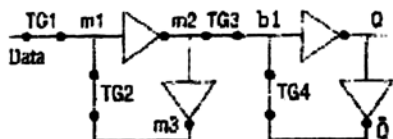


图 4.11 测试模式下触发器等效电路  
Fig4.11 Equal circuit of flip-flop under test mode

图 4.11 给出了触发器在测试模式下的等效电路。因为所有节点被固定地驱动，在测试模式下， $I_{DDQ}$  或电压测量法都能检测到触发器任何节点和  $V_{DD}$  (或  $V_{SS}$ ) 间的桥接故障。

### 4.8.3 透明扫描

尽管在小范围内改变触发器的控制电路结构，可以解决时序元件的  $I_{DDQ}$  测试难题，但芯片级的全局解决方案能有效控制成本，更有价值。一条扫描链，如果所有的触发器都是透明的，则称这条扫描链处在透明模式下。图 4.12 展示了用单相时钟使触发器透明的扫描链路透明概念，但是，这只是技术上的概念，与具体实现无关。

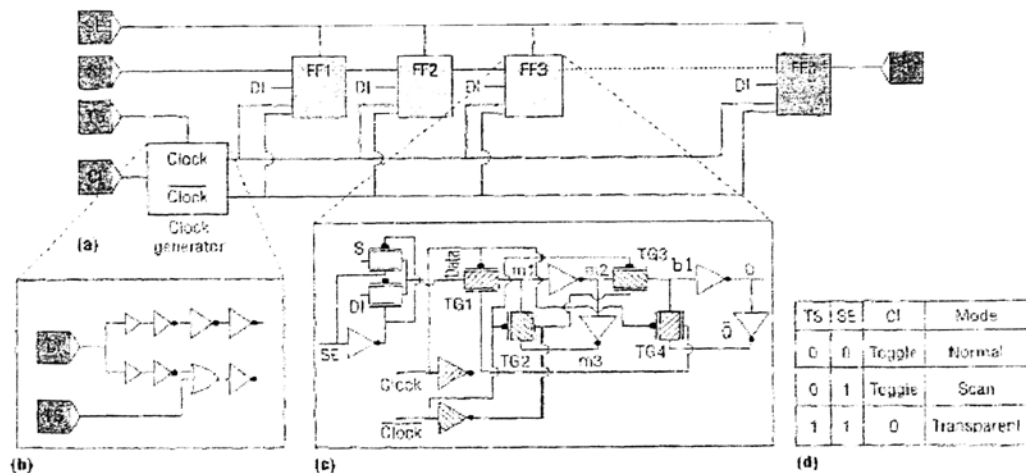


图 4.12 (a)透明扫描链 (b)对时钟生成器的外加控制 (c)改进扫描触发器 (d)触发器运行模式  
Fig4.12 (a)Transparent scan chain (b)Extra control in clock generator (c)Modified scan flip-flop (d)Modes of flip-flop operations

图 4.12a 给出了一条扫描链路，扫描输入(SI)，扫描输出(SO)和扫描使能(SE)信号，SE 选择扫描链路的扫描和正常工作模式。图 4.12a 也给出了一个时钟驱动器，带有时钟输入(CI)和透明扫描(TS)使能信号。图 4.12b 和 4.12c 给出了扩展时钟驱动器和扫描触发器的方块图。

#### 4.9 小结

$I_{DDQ}$  测试非常有效地提高了静态数字 CMOS ICs 的质量，减少了测试成本和复杂性。这项技术利用的是 CMOS 电路的一个重要特性：稳态下，数字 CMOS 电路消耗的电流非常小。因此，很高的稳态电流表明可能有故障或错误存在。错误在特征上是物理的(如短路)。

$I_{DDQ}$  测试需要一个好的传感电路来辅助完成测试任务，这给电路设计者提出了新的要求。由于用于  $I_{DDQ}$  测试的测试码也可根据电路的结构而自动生成，这使得  $I_{DDQ}$  测试更加便于实施，不过如何选择那些能使故障静态电流增加的测试码还值得研究。采用一些有针对性的控制技术，可以化解  $I_{DDQ}$  测试所面临的高静态电流问题，使得  $I_{DDQ}$  测试在深亚微米集成电路中仍不失为一种很好的选择。

尽管没有其他测试方法能达到  $I_{DDQ}$  所能达到的测试质量， $I_{DDQ}$  测试也不是万能的，例如， $I_{DDQ}$  测试不能检测大多的开路故障，而且它被认为是比电压测试方法要慢得多的一种方式。

## 第五章 电路逻辑综合技术

逻辑综合的过程是把初始门级电路描述或布尔逻辑函数转换为优化的基于某一目标工艺单元库的逻辑单元实现的电路描述[43]。在综合系统中，一般把逻辑过程分为两个阶段：与工艺无关的优化阶段和工艺映射阶段。根据所综合电路性质的不同，逻辑综合技术又分为组合逻辑电路逻辑综合技术和时序逻辑电路逻辑综合技术。这一章将简要介绍数字电路逻辑综合的实现过程、解决的问题和一些具体综合技术。

另外，逻辑综合中采用的数据结构也是综合算法得以顺利完成的关键，甚至直接影响着综合电路的好坏。能够让计算机系统更快更好地控制和操作的数据表达形式，无疑为今后深亚微米集成电路的庞大和复杂情况提供了有力的解决途径，也为研究电路的可测性打开方便之门。

由于逻辑综合是电路逻辑结构的最后决定者，它对电路的可测性起着至关重要的作用，所以非常有必要了解综合的过程和具体实现技术，一方面可帮助我们优化电路的可测性设计，另一方面也可用可测性设计来影响综合的算法和行为。

### 5.1 逻辑综合任务和技术

习惯上将数字电路分为组合逻辑电路和时序逻辑电路。一般的时序逻辑电路都是由组合逻辑部分和时序元件构成。对组合逻辑电路和时序逻辑电路分别有不同的综合优化方法，但组合逻辑电路的优化综合是根本。

逻辑综合最终是要将一个用原始输入表示的原始输出规范说明转化成一个在选定工艺库下优化的门级网表，这里的优化可以是针对不同的目标，如面积、速度、功耗等。通常的标准就是当每个库元件具有单独的固定花费时，实现的总体花费要最小。这个固定费用能反映出目标工艺下芯片上一个门所占面积，所有使用的门的费用的总和必须尽可能小。另一个更重要的衡量标准就是速度。

#### 5.1.1 组合逻辑电路逻辑综合技术

组合逻辑电路的逻辑综合所处理的对象一般为表示组合逻辑电路的多级逻辑函



数，而多级逻辑函数又可表达成为一个布尔网络。所谓布尔网络是指一个有向无环图（DAG-Direct Acyclic Graph），其中每个节点对应一个逻辑变量，与初始输入（primary input）变量和初始输出（primary output）变量对应的节点分别称为源（source）和汇（sink）节点。每一个非源节点表示一个逻辑函数（逻辑门），逻辑函数一般被表达为和积的形式（sum of products）或因式形式（factored form）。连接节点的有向边代表节点函数间的输入输出关系。

## 1. 与工艺无关的优化

所谓与工艺无关的优化（technology independent optimization）就是对布尔网络进行优化处理以得到一个优化的布尔网络，但在最终获得的优化布尔网络中，与节点相对应的逻辑门与目标工艺无关。在这一阶段的优化目标主要是面积和延迟，但由于布尔网络中的逻辑门是与工艺无关的，所以得到的面积和延迟只能是估算的。

工艺无关阶段的逻辑优化工作是对没有限定的库元件操作，可用任何单输出的逻辑门，电路成本由它的布尔代数规定的字的数目来决定，实际上是在一系列的代数和布尔操作下（extract, simplify, substitute, collapse, etc）将表示形式转变成为一个布尔网络，且具有最小数目的字，这个数目一般来说是与芯片面积非常紧密相关的。

一般把布尔网络中所有节点函数的面积复杂度（area complexity）之和作为布尔网络的面积测度（area measure），而一个节点的面积复杂度定义为节点函数因子形式中的字符（literal）数，所谓面积优化，就是获得一个面积测度最小的布尔网络。

实现面积优化的算法包括全局面积优化算法和局部面积优化算法。全局面积优化的目标就是同时最小化一组逻辑等式的面积复杂度，而局部面积优化的目标则是最小化布尔网络中单一节点或某一节点局部邻域的面积复杂度。

## 2. 工艺映射

在与库元件有关的逻辑综合阶段，一般是指拿库中的单元去匹配（matching）布尔网络中的元件，实现一个在面积和时延上都比较优化的网络，这一工作也通常称之为工艺映射（technology mapping），或叫库绑定（library binding）。

工艺映射是将布尔网络转换为与特定目标工艺有关的电路实现。与工艺映射有关的问题包括匹配和覆盖(covering)。简单地说,匹配的过程就是用单元库中某一逻辑单元实现布尔网络的某一局部,该逻辑单元就称为一个匹配,所有可行的匹配组成该布尔网络的匹配集。常用的匹配方法有布尔匹配(Boolean mapping)和基于图的匹配(graph-based mapping)。匹配集中可以实现布尔网络的一组匹配称为该布尔网络的一个覆盖。面积优化过程就是获得一个最小覆盖的过程。

在考虑电路的延迟时,每个门电路必须要有一个延迟模型。任何实际的延迟模型必须考虑门输出的电容负载。一个简单但常用的延迟模型表达式如下:

设延迟是从门  $g$  的一个输入端点  $i$  到其输出端点,则  $\tau_{i,g} = \alpha_{i,g} + \beta_{i,g} \times \gamma$

此处  $\gamma$  代表了与门输出负载有关的参数。

最近,功耗被认为是逻辑块的最重要特性,在优化过程中也必须同时考虑。

如果在工艺映射阶段不仅仅针对网络中单个的门元件,还将网络中局部相连的许多门电路作为对象,挑选库中对应的一个元件来匹配,那算法就会更加复杂,不过优化后的性能也会更佳。事实上,这也是许多高级综合系统中必须具备的能力。

### 5.1.2 时序逻辑电路的逻辑综合技术

传统的时序电路逻辑综合技术是把时序电路分解为纯粹的组合模块(combination block)和连接个组合模块的寄存器(register),然后应用组合电路逻辑综合的技术对个组合模块进行优化处理以获得优化的组合模块。最后将各优化的组合模块与寄存器重新连接起来组成一个优化的时序电路。这种综合技术已很难满足高性能 IC's 设计要求,目前的时序电路逻辑综合技术多把时序电路作为一个整体进行优化处理。

大多数时序电路逻辑综合系统可以接受几种不同的时序电路描述,最为常见的时序电路描述为门级网表(net-list of gates)和有限状态机(finite-state machine)。其中门级网表描述由相互连接的单输出组合逻辑门(single-output combinational gate)和锁存器(latch)构成,这里的锁存器仅仅是广义的延迟元件,只有在工艺映射阶段才将这

些延迟元件映射为单元库中实际的锁存器；而有限状态机多用状态转换图（STG-state transition graph）来表示。并且这两种电路描述是可以互相转换的。

## 1. STG 处理技术

- 1> 状态最小化(state minimization)。由于有不确定的状态转换(unspecified transition)和显式输出无关项(don't care)，所以在 STG 表示中存在某些自由度。如果两个状态对任意等效输入序列能够产生等效的输出序列，则称这两个状态是等效的。所谓状态最小化就是利用 STG 表示的自由度和等效状态构造一个具有更少状态的有限状态机。通常这个有限状态机能转换为一个更小的逻辑实现。
- 2> 状态分配(state assignment)。状态分配通过为每一个状态分配一个二进制编码而实现从一个 STG 到网表的映射。
- 3> STG 抽取(STG extraction)。STG 抽取是与状态分配相反的过程。给出一个逻辑级的实现可以从中抽取 STG 以便进行状态的最小化操作。

## 2. 门级网表处理技术

- 1> Retiming。Retiming 是一种通过跨越逻辑门移动寄存器，从而使时钟周期最小化或寄存器数最小化以满足对时钟周期的约束。
- 2> 工艺映射。在时序电路的工艺映射中，我们用一对端点代表来自锁存器的反馈，这样就确保了电路表示的无环特性。因此，可以非常容易地将组合逻辑的工艺映射扩展到时序电路中。

### 5.1.3 逻辑综合与可测性设计的关系

由于电路的结构信息最终是在逻辑综合后确定下来的，所以逻辑综合过程的电路信息对电路的测试是非常重要的。无论是组合的测试方法，还是时序的方法，都必须以综合后的电路物理信息为依据。例如 ATPG 工具可根据综合的电路网表信息来自动生成测试码。另外，有些设计到具体物理结构的方法，也必须在这步完成修改，例如要增加一些测试点来增强整个电路的可控性和可观测性。

可测性设计的考虑也影响着逻辑综合过程的行为。一是面积上的权衡，一般由

于可测性设计的需要，都会在电路中增加一些额外的辅助电路来提高故障覆盖率。还有就是有一些方便可测性设计的元件一般也具有较普通元件大的面积，在选用它们时就要考虑是否真的需要，或者值不值得。可测性设计的方法也很多，对于特定的电路可能某一方法特别适合，而且能使电路面积更小。另一方面是时延的考虑，特别是在扫描通路设计当中，不同的时序元件给系统的电路延时影响差别很大。在集成电路进入深亚微米时代，电路各部分的延时问题是一个非常敏感的问题，稍微的时序偏差可能导致整个系统性能的降低，这也是综合过程中必须严格考察验证的工作。实在不行，可能就要对电路的结构进行调整。特别在关键路径上的时序元件，应该谨慎考虑用作测试用途。如果综合系统能够利用电路本身的结构特征，将一些系统功能元件与测试用元件共享，避免重复应用带来电路复杂和路径加长，也能减小测试考虑带来的负面影响。

如果能在逻辑综合阶段将电路各方面性能优化，当然对可测性设计是有利的，因为电路本身简单后，可测性问题也随之容易。另一方面，综合后的电路信息保留完备，或者数据接口容易，也可加强可测性设计的自动化。下节我们将讨论有关逻辑综合过程中常用的一种数据结构，它在逻辑优化中起着重要的作用。

## 5.2 逻辑综合过程中采用的数据结构

二叉决策图(BDD)是一种广泛应用于计算机科学运算的数据结构。在数字系统设计中，它可用来表示一个布尔逻辑函数。运用 BDD 能够实现对布尔逻辑函数符号化的表示和操作，在[44]中有详细阐述。利用 BDD 也能进行逻辑验证和测试及逻辑综合[45]。

但 BDD 是一种对变量顺序十分敏感的图形表示，不同的变量顺序有时会产生大小有很大差别的图来，一旦问题复杂，如果变量顺序选择不当，将会造成过大的数据量而使运算速度变慢甚至崩溃。所以，能找到一个变量顺序使所操作的布尔方程的 BDD 节点数小是非常必要的。在[46-48]中介绍了利用对称性和下界法使 BDD 最小化的方法，由于建立在动态的全局搜索上，在时间和空间上还是有不小的开销。这里介绍一种新的方法，利用布尔方程本身变量的相关特性，快速地寻找出适当变

量顺序建立起 ROBDD。

BDD 这种表示的几个优点:

1. 许多经常碰到的函数有了一个合理的表示, 例如, 所有对称函数 (包括奇偶校验函数) 由图表示, 顶点数最多增长到变量数的平方。
2. 当处理一系列操作时, 基于我们的算法的程序的运行时间有所降低, 就是时间的复杂性被限制于图大小的乘积内。
3. 根据最简图来讲, 这种图表示是一种唯一形式。

不足之处:

1. 开始要选择一些系统的输入变量的顺序, 而且图的大小对这些变量的顺序非常敏感, 也就是不恰当的变量顺序可能会造成图的存储数据量非常大。
2. 寻找变量顺序使图表示最小是一个 NP 完全问题, 一般可凭经验解决, 用一些遗传算法来选择似乎有可能。
3. 更严重的是, 有些函数用布尔表达式或逻辑电路表示时大小很合理, 但用图来表示却太大而不切实际 (整数乘法器)

下面我们对这些问题详细阐述。

### 5.2.1 BDD 图数据结构的定义及其应用

二叉决策图(BDD)是一种用来表示布尔函数的有向无环图, 它在逻辑综合中起着重要的作用, 是许多算法实现所必须依赖的数据结构形式。它在形式验证和电路测试中的应用也非常广泛。

#### 1. 基本定义与符号

设有布尔函数  $f(x_1, x_2, \dots, x_n)$ , 布尔变量  $x_i$  定义在集合  $B = \{0, 1\}$  上。下面说明一些对这个布尔函数操作的定义和符号表示。

- 代值(restriction): 将函数某一变量  $x_i$  用常量  $b$  替代后的结果, 记作:  $f_{|x_i=b}$ , 则

$$f_{|x_i=b}(x_1, x_2, \dots, x_n) = f(x_1, x_2, \dots, x_{i-1}, b, x_{i+1}, \dots, x_n)$$

- Shannon 展开式:

$$f = x_i \cdot f_{|x_i=1} + \bar{x}_i \cdot f_{|x_i=0}$$

- 合成(composition): 将函数某一变量用函数  $g$  带入后的结果, 记作  $f|_{x_i=g}$ , 则

$$f|_{x_i=g}(x_1 \dots x_n) = f(x_1 \dots x_{i-1}, g(x_1, \dots, x_n), x_{i+1}, \dots, x_n)$$

- 相关(dependence): 影响函数值的变量集合, 记作  $I_f$ ,  $I_f = \{x_i | f|_{x_i=0} \neq f|_{x_i=1}\}$

当函数恒为 1 或 0 时,  $I_f$  为空(empty)

- 满足集(satisfying): 使函数值为 1 的变量序列集合, 记作  $S_f$ ,

$$S_f = \{(x_1 \dots x_n) | f(x_1 \dots x_n) = 1\}$$

定义 1: 函数图是一个有根的有向图, 它有两类顶点包含于顶点集  $V$  中, 非终端节点  $v$ , 它的属性是有两个子节点  $low(v)$  和  $high(v)$  及一个变量索引号  $index(v) \in \{1, 2, \dots, n\}$ ; 端节点  $v$ , 它的属性是有一个固定值  $value(v) \in \{0, 1\}$ 。

$index(v)$  表示该节点处将要代值的变量的索引号, 设为  $i$ , 则非终端节点  $v$  的两个子节点  $low(v)$  和  $high(v)$  所表示的函数分别为:

$$f_{low(v)} = f_v|_{x_i=0}$$

$$f_{high(v)} = f_v|_{x_i=1}$$

$value(v)$  表示节点  $v$  的函数值。

定义 2: 一个有根节点  $v$  的函数图  $G$  表示函数  $f_v$  的递归定义如下:

- 1> 如果  $v$  是一个端节点
  - a. 如果  $value(v) = 1$ , 则  $f_v = 1$
  - b. 如果  $value(v) = 0$ , 则  $f_v = 0$

2> 如果  $v$  是一个非端节点, 且  $index(v) = i$  则  $f_v$  为:

$$f_v(x_1, \dots, x_n) = \bar{x}_i \cdot f_{low(v)}(x_1, \dots, x_n) + x_i \cdot f_{high(v)}(x_1, \dots, x_n)$$

有了定义 1 和定义 2, 我们就可以为布尔函数生成它的函数图了。例如, 对于布尔函数  $f(x_1, x_2, x_3) = x_1 x_2 + x_3$ , 可以得到它的函数图如图 5.1 所示。这个图的每个非终端节点所表示的函数示于其旁, 它的构图变量顺序为  $\langle x_1, x_2, x_3 \rangle$ , 边上的数值为上面的节点在 Shannon

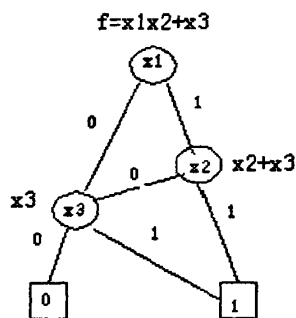


图 5.1 布尔函数  $f(x_1, x_2, x_3) = x_1 x_2 + x_3$  的 BDD 图  
Fig5.1 A BDD of boolean function  $f(x_1, x_2, x_3) = x_1 x_2 + x_3$

分解时该节点所对应变量的取值，边所指下面的节点即表示取该值后的函数。如果不作说明，本文中其它的函数图均表示左边的边取 0，右边的边取 1，而不再标注。我们一般称这个由 Shannon 分解得到的函数图为 BDD 图(Binary Decision Diagram)。

**定义 3:** 如果存在一个从函数图  $G$  的节点到函数图  $G'$  的节点的一一映射函数  $\sigma$  使得对于任意  $v$ ，如果  $\sigma(v) = v'$ ， $\Rightarrow \text{value}(v) = \text{value}(v')$  ( $v, v'$  都是端节点)或  $\text{index}(v) = \text{index}(v')$ ， $\sigma(\text{low}(v)) = \text{low}(v')$  和  $\sigma(\text{high}(v)) = \text{high}(v')$ ，则称函数图  $G$  和函数图  $G'$  是同构的。

**定义 4:** 对函数图  $G$  中任何节点  $v$ ，以  $v$  为根的子图定义为包括  $v$  和它的所有子孙的图。

**推论 1.** 如果图  $G$  和  $G'$  按映射  $\sigma$  同构，那么对于图  $G$  中任意节点  $v$ ，以  $v$  为根的子图同以  $\sigma(v)$  为根的子图也同构。

**定义 5:** 如果函数图  $G$  不含节点  $v$ ，使得  $\text{low}(v) = \text{high}(v)$ ，也不含不同节点  $v$  和  $v'$ ，而他们是同构的，则称图  $G$  是最简的。

**推论 2.** 对最简函数图中每个节点  $v$ ，以  $v$  为根的子图也是最简的。

**定理 1:** 对任意布尔函数  $f$ ，存在唯一最简函数图表示  $f$ ，任何其他函数图表示  $f$  包含更多的顶点

## 2. 化简(reduction)

这个过程是由端节点到根，一个唯一的整型标志号分配给每个唯一的子图根节点，也就是每个节点  $v$  被分配有一个标志号  $\text{id}(v)$ ，对于任何两个节点  $u$  和  $v$ ，当且仅当  $f_u = f_v$  时， $\text{id}(u) = \text{id}(v)$ 。如果将图这样标号，则这个算法构造的图的每个节点有唯一标号。

假如索引号大于  $i$  的节点已被标号，现在对索引号为  $i$  的节点标号，让节点  $v$  取  $id(v)$  等于已标号的某个节点的标号，仅当下面两种情况之一满足：

- 2>  $id(low(v)) = id(high(v))$ ，这时  $v$  为冗余，置  $id(v) = id(low(v))$ 。
- 3> 有某个已标号的节点  $u$ ， $index(u) = i$  且  $id(low(v)) = id(low(u))$ ， $id(high(v)) = id(high(u))$ ，那么  $u, v$  子图同构，置  $id(v) = id(u)$ 。

具体实施步骤如下：

- 1> 根据索引将节点收集到表中（由遍历过程做）
- 2> 处理这些表，由含有这些端节点的一个开始，直到含根节点的一个止，对每个节点创建一个关键字，对端节点采用值的形式，对非端节点采用数序( $lowid, highid$ )的形式
- 3> 如果一个节点的  $lowid = highid$ ，则置  $id(v) = lowid$ ，余下节点按它们的关键字排序。对排序表处理的过程是给所有具有相同关键字的节点分配一个给定的标号
- 4> 为每个唯一标号选择一个节点记录，并由标号索引的矩阵中的这个节点存储一个指针。这些被选的节点将最终形成最简图

例如如图 5.1 本来应该如图 5.2 所示，先由  $x_1$  节点的  $low(x_1)$  指向图中阴影部分，再由阴影部分在 Shannon 分解下指向  $x_3$  节点。只是由于阴影节点的  $low(x_2)$  和  $high(x_2)$  都指向  $x_3$  节点，所以它的  $lowid = highid$ ，则阴影节点的  $id$  与  $x_3$  节点  $id$  相同，是一个节点，所以可以直接由虚线所示连接  $x_3$  节点和  $x_1$  节点，即的简化了的图 5.1。

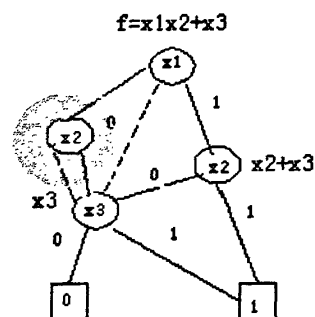


图 5.2 非简化的 BDD 图  
Fig5.2 A BDD without reduction

### 3. BDD 图的应用

前面已将 BDD 图的定义和一些特性交代清楚了，我们现在可以按照规则由布尔函数构造我们需要的 BDD 图。一般来说，在变量顺序给定的情况下，每个布尔函数的 BDD 图的形式是唯一的，不过它必须是一个最简的 BDD 图。有了 BDD 图，我们就可以用它来进行我们想要的操作了。



设有两个定义在相同变量集  $B^n = \{x_1, x_2, \dots, x_n\}$  上的布尔函数  $f_1, f_2$ 。现在  $f_1$  和  $f_2$  之间要进行布尔操作  $\langle op \rangle$ ，则

$$[f_1 \langle op \rangle f_2](x_1, x_2, \dots, x_n) = f_1(x_1, x_2, \dots, x_n) \langle op \rangle f_2(x_1, x_2, \dots, x_n)$$

分别将  $f_1, f_2$  Shannon 分解，则

$$f_1 \langle op \rangle f_2 = (x_i f_1|_{x_i=1} + \bar{x}_i f_1|_{x_i=0}) \langle op \rangle (x_i f_2|_{x_i=1} + \bar{x}_i f_2|_{x_i=0}) = x_i (f_1|_{x_i=1} \langle op \rangle f_2|_{x_i=1}) + \bar{x}_i (f_1|_{x_i=0} \langle op \rangle f_2|_{x_i=0})$$

由这个表达式可知，在进行两个布尔函数操作时，只要将以这两个布尔函数为根节点的 BDD 图递归地在其子图上应用相应的操作就可以得到结果的 BDD 图表示。如果我们应用化简操作将结果 BDD 图化为最简形式，它所对应的布尔函数应该是唯一的。至此，我们就将布尔函数的运算转化为 BDD 图的运算。而 BDD 图又具有一些特殊的性质，在表示数据时可以具有更加简洁的形式，再者，它的运算形式更加简单。这也是逻辑综合和测试中采用这种数据结构的原因。

设  $f_1, f_2$  是以  $v_1, v_2$  为根的 BDD 图表示的函数，用 BDD 图进行它们之间的布尔操作时，有下列情况要考虑：

- 1>  $v_1, v_2$  为端节点
- 2>  $v_1, v_2$  中至少有一个为端节点

在这两种情况下，如果  $\text{index}(v_1) = \text{index}(v_2) = i$ ，则创建节点  $u$ ， $\text{index}(u) = i$ ，则递归地应用布尔操作于  $\text{low}(v_1)$ 、 $\text{low}(v_2)$ 、 $\text{high}(v_1)$  和  $\text{high}(v_2)$  上，产生的子图分别作为以  $u$  为根的  $\text{low}(u)$  和  $\text{high}(u)$ 。

- 3>  $\text{index}(v_1) = i$ ，但  $v_2$  是端节点或  $\text{index}(v_2) > i$

这时  $v_2$  与  $x_i$  不相关，即  $f_2|_{x_i=0} = f_2|_{x_i=1} = f_2$ 。创建节点  $u$ ， $\text{index}(u) = i$ ，递归地应用布尔操作于  $\text{low}(v_1)$ 、 $v_2$ 、 $\text{high}(v_1)$  和  $v_2$  上，产生的子图分别作为  $u$  的  $\text{low}(u)$  和  $\text{high}(u)$ ，反之亦然。

一般这样产生的 BDD 图通常不是最简的，所以须运用简化算法后再返回结果，以便释放空间和方便下一步操作，如比较、合成等。简化后的 BDD 图也很容易得到布尔函数的满足集（就是遍历每个变量到达终端节点 1 的所有路径的集合）。

### 5.2.2 KFDD 图数据结构介绍

KFDD 图其实是 BDD 图更为一般的形式，它在构图过程中不仅可采用 Shannon 分解方式，还可采用 Davio 分解方式来生成图结构，只不过在生成的图数据结构中必须注明每个变量的分解方式。但它的形式较灵活，有些用 BDD 图表示数据量很大的函数，用 KFDD 图表示起来很简洁。

#### 1. 记号与定义

设映射  $f: B^n \rightarrow B$  是定义在变量集  $X_n = \{x_1, \dots, x_n\}$  上的布尔函数，DDs(Decision Diagrams)是其基于图的表示形式，它的每个非终端节点被标以变量  $x_i$ ，由这个节点代表的函数被分解为两个子函数，而且，如果没有明确申明，下面的图是有序的。例如，DD 的所有路径上的变量以相同的次序出现。DD 图的生成是下面三种分解方法的递归应用。

$$f = \bar{x}_i f_i^0 + x_i f_i^1 \quad \text{Shannon (S)}.$$

$$f = f_i^0 \oplus x_i f_i^2 \quad \text{positive Davio (pD)}.$$

$$f = f_i^1 \oplus \bar{x}_i f_i^2 \quad \text{negative Davio (nD)}.$$

这里的  $f_i^0$  和  $f_i^1$  分别表示  $x_i = 0$  和  $x_i = 1$  时  $f$  的余因子， $f_i^2$  定义为

$$f_i^2 = f_i^0 \oplus f_i^1.$$

$\oplus$  是 EXOR 运算符。

递归地运用这些分解方法，直至函数值为 0 或 1 时停止，此为 DD 图的终端节点。在构造 DD 图时，如果节点  $v$  以变量  $x_i$  按 S 或 pD(nD)方式分解，则其左子节点为  $f_i^0$  ( $f_i^1$ )；如果是按 S(pD 或 nD)方式分解，则其右子节点为  $f_i^1$  ( $f_i^2$ )。

如果只执行 Shannon 分解，则得到的是二叉决策图(BDD)；如果只执行 Davio 分解，则是一个功能决策图 (FDD, Functional DD) [49]；而 KFDD(Kronecker Functional DD)是所有的分解方式都使用后得到的 DD 图。对于每个变量  $x_i$  都有相应的分解类型，它们组成一个分解类型表 (DTL, decomposition type list)  $d$ ， $d = (d_1, \dots, d_n)$ 。此处  $d_i \in \{S, pD, nD\}$ ，也就是每个变量  $x_i$  有一个固定的分解类型  $d_i$  可选。

如果引入互补边 (CEs), KFDD 的大小可进一步缩小, 一个节点就可用来表示一个函数的同时也代表这个函数的补。

## 2. 一个 KFDD 图的例子

图 5.3 是一个 DTL  $d=(S, pD, nD, S)$  的 KFDD 图表示形式, 它代表的布尔函数为

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \bar{x}_3 \oplus x_1 \bar{x}_2 \bar{x}_3 \oplus \bar{x}_2 x_3 x_4 \oplus x_1 x_2$$

例如标注  $x_4$  的节点是一个 S 类型节点, 有一条 CE 边指向节点 1, 所以这个节点代表的函数为  $\bar{x}_4 \cdot 1 + x_4 \cdot 0 = \bar{x}_4$ 。而阴影节点是 nD 类型节点, 它的两条边都指向  $x_4$  节点, 所以它代表的函数为  $\bar{x}_4 \oplus \bar{x}_3 \cdot \bar{x}_4 = \bar{x}_4 \cdot (1 \oplus \bar{x}_3) = \bar{x}_4 \cdot x_3$ 。

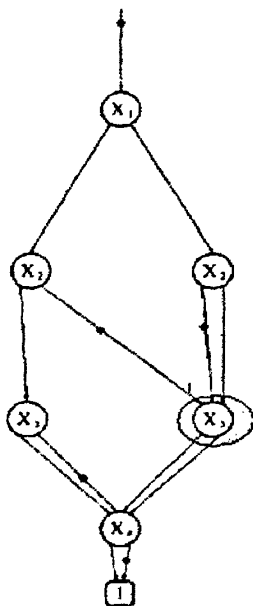


图 5.3 一个分解类型  $d=(S, pD, nD, S)$  的 KFDD 图  
Fig5.3 KFDD with DTL  $d=(S, pD, nD, S)$

KFDD 图的运算操作与 BDD 图类似, 不过具有更多的灵活性和变化形式, Rolf Drechsler 对此作了详尽的阐述[50]。

### 5.3 BDD 优化方法

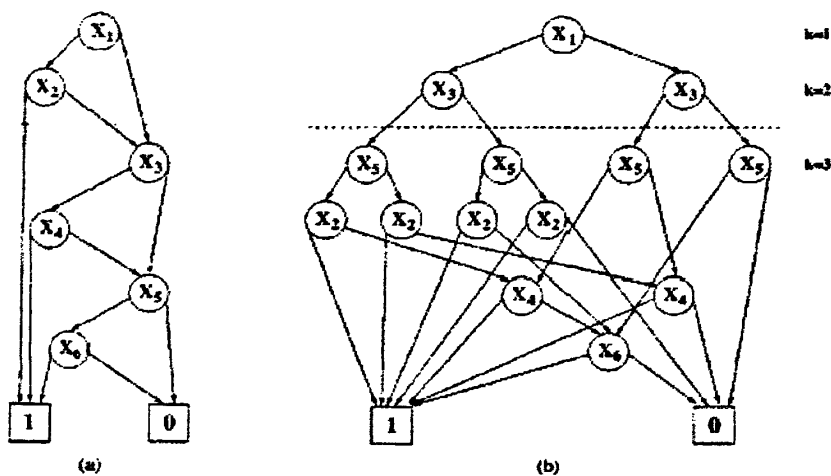
BDD 图数据结构是较常用的一种图形式，本节以此为例探讨它的优化方法。实际应用中的 BDD 图的变量顺序是不能任意改动的，对 BDD 图的运算操作都是按照一个特定变量顺序进行的。我们称这样的 BDD 图是有序的，如果它经过化简剔除冗余节点，则称之为 ROBDD(Reduced Ordered Binary Decision Diagram)。由于 BDD 是一种对变量顺序非常敏感的数据结构形式，不恰当的变量顺序可能会造成 BDD 太大而无法操作，或运算效率低下。如何选择变量顺序，使 BDD 最小化或可接受是一件非常重要而棘手的问题。

#### 5.3.1 快速下界法

一种用得比较多的最小化 BDD 的技术是下界限定法。穷举所有的变量顺序而选择最佳的结果，其工作量是难以想象的。而利用遗传算法的优势，在选择变量顺序过程中，通过下界的设定，尝试各种不同的变量顺序，是一种较理想的选择。一些非常低劣的变量顺序会因为 BDD 的快速增大而超过下界而很快舍弃，避免无用工作。如果再辅助一些已知条件，有些较次的变量顺序也会在早期发现而停止进一步的尝试。另外，在某些情况要想找到最优解决方案可能花费会更大，得不偿失，这时可通过设置下界找到一个可以接受的解就行了。

例如图 5.4a 是函数  $f = x_1 x_2 + x_3 x_4 + x_5 x_6$  在变量顺序  $\langle x_1, x_2, x_3, x_4, x_5, x_6 \rangle$  下的 ROBDD 图，如果以其非终端节点数衡量 BDD 大小，则它的大小为 6。如果要考察在变量顺序  $\langle x_1, x_3, x_5, x_2, x_4, x_6 \rangle$  下函数的 BDD 大小，就可将  $\text{low\_bound} = 6$  作为下界值。图 5.4b 是在这一变量顺序下的实际 ROBDD 图。其实我们在运用下界技术时，当图 5.4b 进行到深度  $k=2$  时就可舍弃这一变量顺序了。因为当  $k=2$  时， $\text{cost}_{\text{BDD}}|_{k=2} = 3$ ，此时还有 4 个变量未进行分解，则

$$\text{cost}_{\text{BDD}} \geq \text{cost}_{\text{BDD}}|_{k=2} + 4 = 7 \geq \text{low\_bound}$$



5.4 函数  $f = x_1x_2 + x_3x_4 + x_5x_6$  的 BDD 图  
 Fig5.4 BDDs for function  $f = x_1x_2 + x_3x_4 + x_5x_6$

### 5.3.2 变量相关法

本小节为 ROBDD(reduced ordered binary decision diagram)提供了一种新的寻找变量顺序算法。直接从被处理的布尔方程入手，根据其内在的变量的相关性在不同变量顺序下对 BDD 大小的直接影响的规律，可以预先直接由给定的变量顺序估算 BDD 的节点数。在全局上对变量相关性的权衡，可以快速找到一个合适的变量顺序而建立所需的 ROBDD。

#### 1 基本概念

由于 BDD 的大小非常依赖于变量的顺序，下面的工作就是要找一个变量顺序  $O$  使 BDD 最小。

如果变量顺序  $O < \dots x_i x_j \dots >$  是在变量集  $X_n$  上的一个序列，则表示  $x_i$  在  $x_j$  之前。对于  $f$  的一个变量顺序  $O$  如果第  $k$  个变量为  $x_i$ ，则计为  $x_i = O(k)$ ，而把方程  $f$  在变量顺序  $O$  下的 BDD 节点数计为  $nodes(f, O)$ 。在第  $k$  层次的节点数计为  $nodes_k(f, O)$ 。设节点  $v$  是 BDD 图第  $k$  层的节点，则  $low(v)$  表示在节点  $v$  处  $O(k)=0$  的结果， $high(v)$  表示在节点  $v$  处  $O(k)=1$  的结果。本文后面提到的 BDD 图，如果不作说明，均按节点  $v$  左边的边指向  $high(v)$ ，右边的边指向  $low(v)$  建立，节点个数不包括终端节点 1 和 0。

**规则1.** 在选取 BDD 的变量顺序时, 尽量让相关紧密的变量相连, 即相关变量应尽量在  $O < x_1, x_2, \dots, x_n >$  中相连, 这样有利于 BDD 的规模较小。

**规则2.** 相关度大的变量应在不违反规则 1 的条件下尽量处在变量顺序的前面; 或局部违反规则 1 而处在变量顺序前面, 但能使 BDD 节点数更少。

**定义**

**变量相关性:** 指布尔方程  $f(x_1, x_2, \dots, x_n)$  中变量  $x_1, x_2, \dots, x_n$  之间的关联性。方程中如果一个变量与另一变量有逻辑“与”的关系, 则这两个变量是相关的。如,  $f = x_1x_2 + x_3x_4$ , 则  $x_1$  与  $x_2$  相关,  $x_3$  与  $x_4$  相关,  $x_1$  与  $x_3$  不相关。如果逻辑表达式是以和积的形式表述每一个积项所包含的变量都相关, 并形成一相关集。变量  $x_i$  的相关集记  $R(x_i) \ i=1,2,\dots,n$ 。上例中  $R(x_1)=\{x_2\}$ ,  $R(x_3)=\{x_4\}$ 。

**相关度:** 一个变量的相关度为该变量在其布尔方程最简和积项中出现的次数。

**相关性破坏度:** 对任一选取的 BDD 变量顺序, 违反规则 1 的程度描述。

**相关性权衡:** 假如一个变量在多个相关集中出现, 在选取该变量的后续变量时, 应该对整体相关性及相关性破坏度进行权衡, 尽量让满足规则 1 和规则 2 的变量成为后续变量。

**2 相关性对 BDD 大小的影响**

设有布尔方程

$$f = x_1x_2x_3 + x_4x_5 + x_6 \tag{5.1}$$

选取两种变量顺序  $O_1 < x_1x_2x_3x_4x_5x_6 >$ ,  $O_2 < x_1x_2x_4x_3x_5x_6 >$ , 由  $O_1, O_2$  建立的 BDD 图分别如图 5.5 和图 5.6 所示。

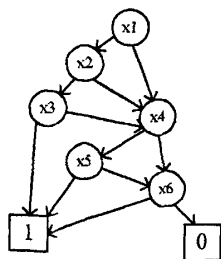


图 5.5 布尔方程 5.1 的 BDD1  
Fig 5.5 BDD1 for function 5.1

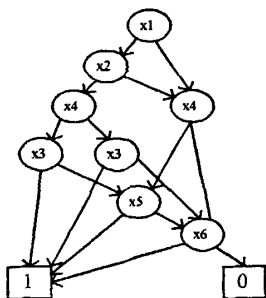


图 5.6 布尔方程 5.1 的 BDD2  
Fig 5.6 BDD2 for function 5.1

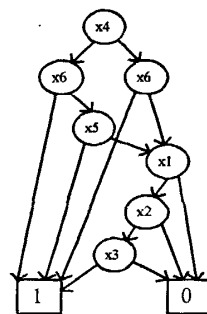


图 5.7 布尔方程 5.1 的 BDD3  
Fig 5.7 BDD3 for function 5.1

BDD2的节点比BDD1的节点多2个，从图中可看出是在变量 $x_3$ 和 $x_4$ 上分别多了一个节点。考查 $O_1$ 和 $O_2$ 可发现， $O_1$ 中变量的顺序完全按照相关性原则选取，而 $O_2$ 的变量顺序却与方程中变量所蕴含的相关性有冲突，如 $x_4$ 就把 $x_3$ 与 $x_2$ 隔开， $x_3$ 把 $x_4$ 与 $x_5$ 隔开，这就是为什么BDD2中会多出 $x_3$ 和 $x_4$ 两节点的原因。

再选一种变量顺序 $O_3 < x_4 x_6 x_5 x_1 x_2 x_3 >$ 得到BDD3如图5.7所示。BDD3有9个节点，比BDD1多1个，比BDD2少1个，按照相关性原则可知， $O_3$ 中变量 $x_6$ 的位置不好，它将相关紧密的 $x_4$ 与 $x_5$ 分开了，但相对于 $O_2$ 而言，它所造成的相关性破坏程度还是轻的。对方程(5.1)，任何其他变量顺序所建立的BDD的节点数都不会少于BDD1的，因 $O_1$ 的变量顺序完全满足方程(5.1)所蕴含的变量相关性与原则，因此它所表示的BDD是最小的。

对于布尔方程

$$f = x_1 x_2 + x_1 x_3 + x_4 x_5 \quad (5.2)$$

由于 $x_1$ 在不同的项 $x_1 x_2$ 和 $x_1 x_3$ 中都出现，它是一个多相关量，相关度为2，在考虑变量顺序时它应该优先。现有变量顺序 $O_1 < x_1 x_2 x_3 x_4 x_5 >$ ， $O_2 < x_1 x_2 x_4 x_5 x_3 >$ 它们对应的BDD图分别如图5.8和图5.9所示

变形方程5.2得

$$f = x_1 (x_2 + x_3) + x_4 x_5$$

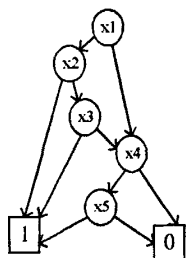
$x_2 + x_3$ 作为整体与 $x_1$ 相关，所以在考虑相关性时，应将 $x_2, x_3$ 同时考虑进去，而由对称性知 $x_2, x_3$ 的位置可互换，按照规则1和规则2， $O_1$ 是较好的选择。

下面讨论有多个布尔方程函数的情况。为简单起见，在此只以有两个方程为例，也就是要同时建立两个BDD。

设

$$\begin{cases} f_1 = x_1 x_2 + x_3 x_4 x_5 \\ f_2 = x_1 x_3 x_4 + x_2 x_5 \end{cases} \quad (5.3)$$

在变量顺序 $O_1 < x_1 x_2 x_3 x_4 x_5 >$ ， $O_2 < x_3 x_4 x_1 x_2 x_5 >$ ， $O_3 < x_1 x_3 x_4 x_5 x_2 >$ 下，布尔方程组5.3的BDD图分别如图5.10、图5.11、图5.12所示。



5.8 布尔方程 5.2 的 BDD4  
5.8 BDD4 for function 5.2

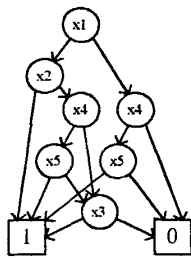
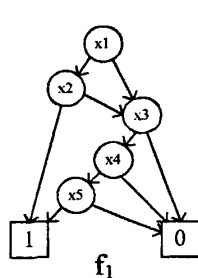
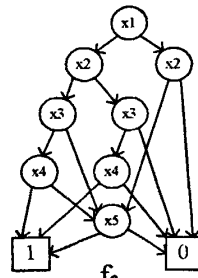


图 5.9 布尔方程 5.2 的 BDD5  
Fig5.9 BDD4 for function 5.2

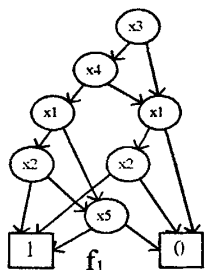


$f_1$

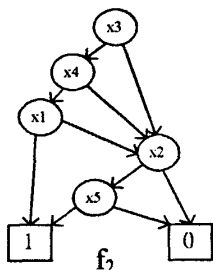


$f_2$

图 5.10 方程组 3 在  $O_1 < x_1 x_2 x_3 x_4 x_5 >$  下的 BDD  
Fig5.10 BDD for equations 5.3 with  $O_1 < x_1 x_2 x_3 x_4 x_5 >$

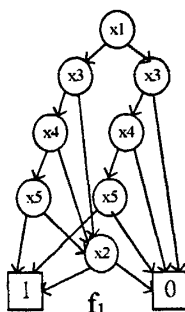


$f_1$

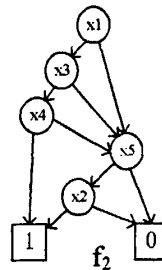


$f_2$

图 5.11 方程组 3 在  $O_2 < x_3 x_4 x_1 x_2 x_5 >$  下的 BDD  
Fig5.11 BDD for equations 5.3 with  $O_2 < x_3 x_4 x_1 x_2 x_5 >$



$f_1$



$f_2$

图 5.12 方程组 3 在  $O_3 < x_1 x_3 x_4 x_5 x_2 >$  下的 BDD  
Fig5.12 BDD for equations 5.3 with  $O_3 < x_1 x_3 x_4 x_5 x_2 >$

可以发现，任何一种变量顺序  $O$  都不能同时满足  $f_1$  和  $f_2$  的变量相关性均不被破坏。要同时为  $f_1, f_2$  建立 BDD，为了使 BDDs 总的节点数小，则必须找到一种折衷的变量顺序  $O$ ，这时利用变量相关性原则及其在被破坏情况下对 BDD 大小的影响，可以很快找出合适的变量顺序。

上例中 BDDs 的节点数不会少于 12。

通过上面的图例，可以清晰地发现破坏变量相关性后给 BDD 带来的影响。一般地，设有布尔方程

$$f = g(X_n) + h(Y_m)$$

$X_n, Y_m$  为映射在  $\{0, 1\}$  上的变量

集合， $X_n \cap Y_m = \phi$

设  $x_i, x_j \in X_n$ ，且  $x_i, x_j$  相关，若变量顺序  $O_f = \langle \dots x_i x_j \dots O_h \rangle$ ，则对应局部 BDD 图如图 5.13。图中曲线包围节点  $V_h$  的部分表示在变量顺序  $O_h$  下以  $V_h$  为根节点的  $h(Y_m)$  的



BDD。如果将变量顺序  $O_f$  改为  $\langle \dots x_i O_h x_j \dots \rangle$ ，即用局部变量顺序  $O_h$  来破坏  $x_i, x_j$  的相关性，则所建立的局部 BDD 图如图 5.14 所示，这时由变量集  $Y_m$  引起的节点数为： $nodes(Y_m) = nodes(Vh(x_j)) + nodes(Vh) = 2 * nodes(Vh) = 2 * nodes(h, O_h)$

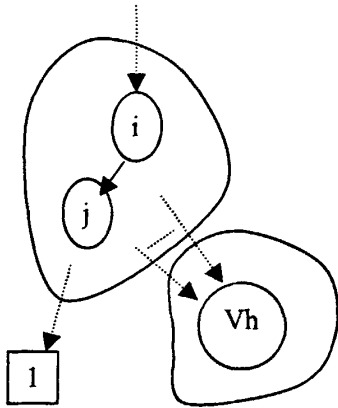


图 5.13 保证相关变量  $x_i, x_j$  相连的局部 BDD 图  
Fig 5.13 Local BDD with adjacence of relevant variables  $x_i, x_j$

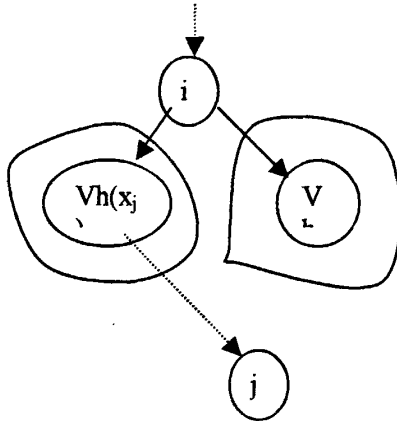


图 5.14 破坏相关变量  $x_i, x_j$  相连的局部 BDD 图  
Fig 5.14 Local BDD with conflict of relevant variables  $x_i, x_j$

如果同时  $O_h$  还破坏其它变量相关性时，随着破坏的深度和层次，所引入的节点数将以 2 的指数倍增长。如果变量  $x_i$  在  $O$  中不违反相关性原则，则它在方程的 BDD 图中贡献节点数为 1，即

$$nodes_k(f, O) = 1 \quad \text{当 } O(k) = x_i, \text{ 且 } x_i \text{ 不违反相关性原则.}$$

如对方程 5.1，在其  $O_2$  下

$$nodes(f, O_2) = \sum nodes_k(f, O_2) = 1 + 1 + 2 * 1 + 2 * 1 + 1 + 1 = 8$$

在  $O_3$  下

$$nodes(f, O_3) = \sum nodes_k(f, O_3) = 1 + 2 * 1 + 1 + 1 + 1 + 1 = 7$$

如果对方程 5.1 在变量顺序  $O_4 < x_1 x_2 x_4 x_6 x_5 x_3 >$  下建立 BDD，则由于  $x_6$  首先违反了  $x_4$  与  $x_5$  的相关性，又因为  $\langle x_4 x_6 x_5 \rangle$  违反了  $x_2$  与  $x_3$  的相关性，所以

$$nodes(f, O_4) = \sum nodes_k(f, O_4) = 1 + 1 + 2 * 1 + 2 * 2 * 1 + 2 * 1 + 1 = 11$$

对于方程组 5.3，按照此算法得到的 BDD 节点数也一一与实际相吻合。

### 3 ROBDD 的建立

- 顺序选择前的工作

读入逻辑方程  $f(x_1, \dots, x_n)$ , 建立变量表  $X_n(x_1, \dots, x_n)$ . 根据方程  $f$  的逻辑关系建立变量相关表  $table(x_1, \dots, x_n)$ ,  $table$  由  $n$  个子索引表  $R(x_i)$  组成, 索引关键字为变量  $x_i$ .  $R(x_i)$  表示与  $x_i$  相关的变量的集合. 如果  $x_i$  在  $f$  中出现在不同的项中,  $R(x_i)$  将是一个多维表, 其维数由其关键字的相关度决定。

$Table((x_1, \dots, x_n) = (R(x_1), \dots, R(x_n))$

建立好变量相关表后, 就可以为每个变量预先统计其相关度。

- BDD 节点估算

对一给定的变量顺序  $O$  估算 BDD 的大小。

预置  $nodes_k(f, O) = 1$

for  $i=1$  to  $m$

```
{
    w(1)取  $O(i)$ , 查找相关性表
    w(2)取最近变量  $O(j)$ , 且  $O(i), O(j)$  相关,  $i < j$ 
        计算  $a1=j-i$ 
        对任意正整数  $k$ ,  $i < k < j$ , 若  $k$  存在,
         $nodes_k(f, O) = 2 * nodes_k(f, O)$ ;
}
```

w(3)求和  $\sum nodes_k(f, O)$ ;

- 伪码描述

下面给出由布尔方程直接快速寻找变量顺序来建立 BDD 的代码描述。

`find_optimal_ordering` 是主循环体,

`find_optimal_ordering_aid` 是辅助处理部分.

`find_optimal_ordering(f(x1, ..., xn))`

```
{
     $O = \phi$ ;
    Node(f, O) = 1;
     $V = \{ x_1, \dots, x_n \}$ ;
    Initialize (table(x1, ..., xn));
     $x_i = \text{select}(\text{table});$  //从 table 中选一关联度较大的变量  $x_i$  作为 BDD 的根节点;
    temp =  $x_i$ ;
    for(k=1; k ≤ n; k++)
    {
         $O(k) = \text{temp}$ ;
         $x_j = \text{traverse}(\text{table}, O(k));$  //查表 table, 找 temp 的相关变量作为后续变量, 优先考虑相关度
```

```

//大的变量.如果没有相关的后续变量,则在剩下的变量中选取一
//个相关度较大的变量,设找到后续变量为  $x_j$ .

temp =  $x_j$ ;
nodes(f,O)= nodes (f, O) + nodes $_k$  (f, O);
Table = delete( table ,  $x_i$ );
}}
find_optimal_ordering_aid( f( $x_1, \dots, x_n$ ),  $O_{pre}$ )
{
     $O = O_{pre}$ ;
    for( $k=1$ ;  $k \leq n$ ;  $k++$ )
    {
        if ( flag( $O(k)$ ) // $O(k)$ 的后续变量有多种选择
            set  $O(k+1)=$ another option; //在每个后续变量按相关性原则有多种选择可能的变量处,
            //尝试不同于  $O_{pre}$  的选择。
            modify( $O$ ); //修改变量顺序
            if (nodes(f,  $O_{pre}$ ) > nodes(f,  $O$ ))
                return  $O$ ;
            else
                return  $O_{pre}$ ;
    }
}

```

#### 4. 结论

利用被处理方程自身固有的变量信息，按照相关性原则来选取变量顺序，实现 BDD 的最小化，是一种方便而有效的方法。它较以前的变量动态交换法具有很多优点，首先可以体现在它的直接性上，另外就是节点预估计，对于明显将引入太多节点的变量顺序事先就可排除，避免了建立 ROBDD 的盲目性和重复性。对于复杂的多方程和多变量情况，本文提出的算法也是简单有效的，如果能结合当前的上下界定法将会更加显示出它的优越性。

#### 5.4 BDD 在逻辑综合中应用的例子

直接对一个大的电路表示来修改调整是一件非常困难的工作，而将其符号化地表示出来，再利用计算机强大的计算能力在相关规则条件下优化处理，会取到很好的效果。BDD 图就是应用得非常多的一种符号表示形式，而且它易于操作，很多优化的算法都能顺利应用。

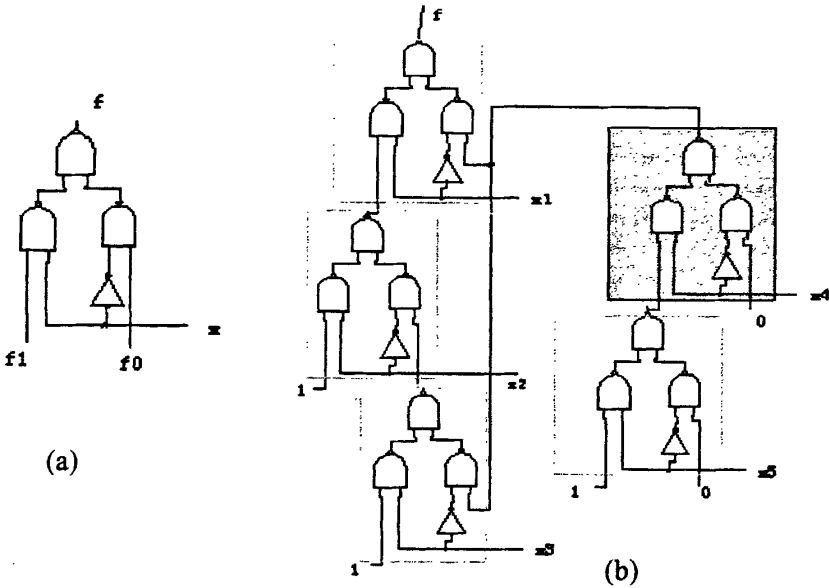


图 5.15 (a)一个 BDD 节点单元 (b)用(a)映射图 5.8 的 BDD 图得到的电路  
 Fig5.15 (a)A BDD node cell (b)Circuit derived from BDD of figure 5.8 by mapping cell (a)

由一个 BDD 图是可以直接映射成一个门电路实现的逻辑网络，所以在工艺无关映射阶段的许多优化算法可直接对 BDD 图操作，例如得到的 BDD 图越小，那么映射的电路实现的面积就会越小，图的深度越小，那么相应的时延也会越小，也可由 BDD 图定性地了解整个电路的关键路径所在，在后面的优化步骤中可作为参考信息。如果用图 5.15(a)所示的由二输入与非门和反向器组成的单元电路将 BDD 图映射成门级电路，那么对于逻辑函数  $f(x_1, x_2, x_3, x_4, x_5) = x_1 x_2 + x_1 x_3 + x_4 x_5$ ，它的一个 BDD 图如图 5.8 所示，可以得到电路图如图 5.15(b)所示。

如果考虑到 BDD 图中一些特殊的节点，例如节点的左子图是终端节点 1 或右子图是终端节点 0，则该节点的匹配模式可进一步简化。如上图阴影部分的电路对应 BDD 图中的  $x_4$  节点，由于它的右子图是 0，所以这个节点的函数图可用图 5.16(a)表示。

如果将这个模式图和其他特殊节点的简化模式匹配图一起作为映射操作的基本单元，那么上面的 BDD 图可初步映射成图 5.16(b)所示的电路。

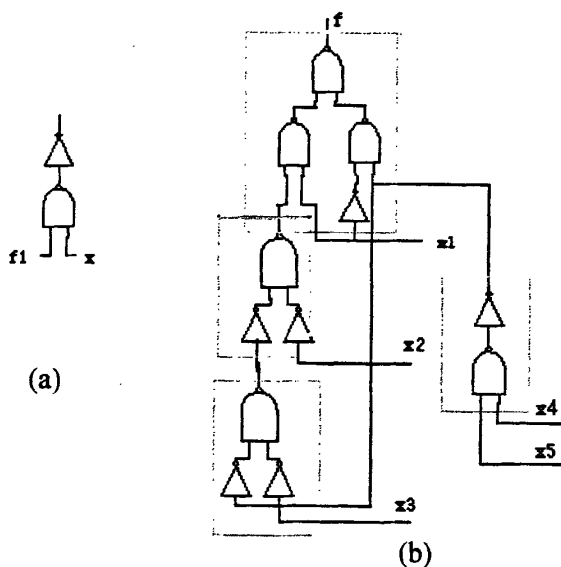


图 5.16 (a)一个简化的 BDD 节点单元 (b)重新映射图 5.8 的 BDD 图得到的电路  
 Fig5.16 (a)A reduced BDD node cell (b)Circuit derived from BDD of figure 5.8 by remapping cell (a)

## 5.5 小结

本章介绍了逻辑综合中的一些基本问题和采用的方法。由于逻辑综合是整个 EDA 系统中一个承前启后的重要环节，它对电路的影响非常大，我们对它的了解有助于我们对于电路可测性的评估和改进。

逻辑综合中所使用的数据结构直接关系到综合的进展和效果，特别电路规模超过百万门级后，综合过程中的数据量是大得惊人的，如果没有有效的控制技术，一则浪费时间，二则综合程序会崩溃。对于 BDD 的使用，好处是很明显的，其实它是一种更为一般的数据结构 KFDD 的特殊形式，只不过它的应用更为广泛。BDD 的一个缺点就是它的大小与变量顺序十分相关，一些简单的函数关系在不恰当的变量顺序下可能会产生非常大的 BDD 图。我们提出了一些 BDD 变量顺序选择的算法，能使这一情况得到改善。

## 第六章 可测性逻辑综合

### 6.1 可测性逻辑综合的引入

逻辑综合优化过程中，往往是对电路的时延和面积作优化为主要的优化目标，会利用不同的电路结构来实现同一种功能[43]。一方面，可测性的问题考虑较少；另一方面，前一步中可测的电路可能在经过优化打散或整合之后，变得不可测了，特别是在电路规模达到百万门级以后，逻辑关系变得异常复杂，要想保证设计的电路每部分都是可测的，并不是一件容易的事。

传统的在逻辑综合后对门级网表结构的改进或增加测试点来达到可测的目的的方法，随着电路规模的增大并不能适应新的要求。一是这一步很难顺利完成；另外，逻辑优化后提高的电路系统性能也将会被抹杀掉。

如果在电路设计的开始就对电路的可测性考虑得过于详细，一方面丧失了高层设计的灵活性，使早期的开发设计工具也变得异常繁重起来；另一方面，也不适应超大规模集成电路时代所要求的自动化特性。目前，已有很多优良的可测性设计方案可供设计者利用，而将它们各自的特点联系起来，取长补短，适应不同电路要求，再配以适当的算法，结合传统的电路综合目标，就可将电路可测性问题的处理自动化。

可测电路的设计通常分两步走：首先，电路必须满足特定的功能的要求，然后在考虑可测性方面的问题。通过在电路综合中将测试策略考虑进去的方式，可将由于测试而带来的额外花费降低[51-53]，也就是本章所要阐述的可测性综合。

### 6.2 基于工艺级的可测性逻辑综合

工艺映射涉及到用一个工艺库中的原始元件实现一个逻辑网络的过程。组合逻辑网络被分解成逻辑树，这棵树的每个节点代表了一个基函数(二输入与非门或反向器)，边代表了一个逻辑网线。这棵树称之为主体树。工艺库中门电路也同样用基函数表示，生成模式树。由上一章的介绍可知，工艺映射由下面两个问题组成：

- 1> 匹配问题：从主体树选择一部分电路的模式，然后从工艺库中的模式树中找到与之相匹配的。
- 2> 覆盖问题：从众多的匹配中选择一个覆盖整个网络而面积、时延、功耗或它们的组合最小的方案。

### 6.2.1 可测条件下的面积速度优化

在加入可测性考虑的情况下，逻辑综合中涉及工艺映射的处理过程要复杂些。基于规则的技术和算法，能有效地将一个逻辑网络绑定到一个工艺库，它对普遍的元件是非常有用的，如对称性组合逻辑电路（AND, OR）、寄存器、加法器、减法器。代表存储元件的输入数据的树节点处在模式树的根部（叫  $P_d$ ），表示其输出的节点则处在模式树的叶子上（叫  $P_o$ ）。要想得到成本最小的覆盖，3种可能性需要考虑：

- 1> 工艺库中有扫描元件，包括了匹配模式  $P_d$  的模式，这个扫描元件则是一个有效的匹配。
- 2> 工艺库没有所要的扫描元件，这时模式  $P_d$  必须完全由组合逻辑实现，输出到扫描元件。
- 3> 条件 1 满足，但采用原始的存储元件，用上面 2 同样的步骤，能获得更低成本的覆盖。

这个概念在图 6 中已展示出来，图 6a 中，AND 门输出给 Mux-SRL（Mux-Scanable Register Logic，通常用于边界扫描设计）。每个元件在工艺库中可能有一种匹配，许多情况下，一种更有效的实现方法可能是用一个基本的 SRL 和一个用组合逻辑优化的 MUX 逻辑，如图 6b 所示，可以在面积和速度上具有灵活的选择。

大多数的工艺库提供具有两相存储输出（ $Q$  和  $\bar{Q}$ ）的存储元件，通过调整匹配图的极性（如用双反向器），模式  $P_o$  的输出可能找到一个更低花费的匹配。数据输出可能分布在  $Q$  和  $\bar{Q}$  间，扫描输出可由其中任何一个得到。有些工艺库还提供专门的扫描输出引脚，或许能使替代的代价更低。

对于扫描元件，一般都会具有比正常元件更大的时延，所以为了保证系统性能，在满足一定的可测性条件下，可以不将有些存储元件纳入扫描行列，特别是有些关键路径上的元件更是需要如此考虑，甚至可以以牺牲面积为代价。

一般在工艺库中对于同种逻辑功能的模块有不同的实现形式，它们各有优缺点。特别是为了可测性的考虑，许多的时序模块都加上了便于测试用的组合控制逻辑。它们的代价一般较正常的要大，在选择替换时要慎重考虑。

与工艺相关的模块替换可测综合方法是一种简单而有效的可测性设计综合方法，它是建立在扫描通路技术的基础上，利用一些增强可测性的单元来替换掉电路中某些元件，从而实现基于全局的扫描测试技术，它也包括在可测性薄弱的地方直接插入辅助测试元件的方法。这种方法一般要求对电路有一个较清晰的结构层次的掌握，在电路的高层综合中已形成基本框架，只不过在工艺映射阶段可以再综合形成一个代价最小的实现。

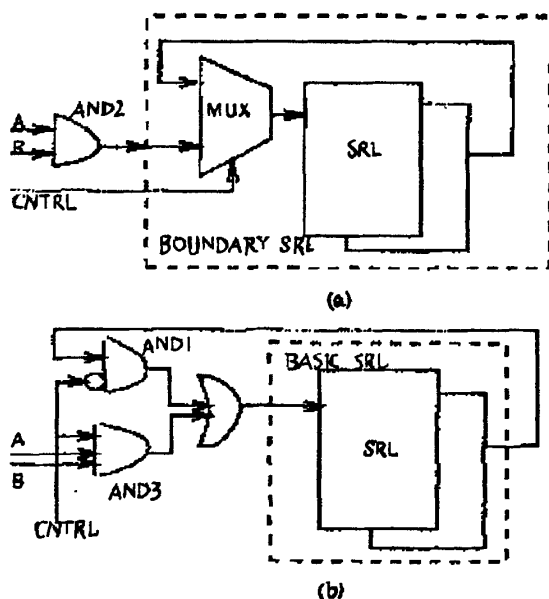


图 6.1 (a)工艺无关的与门输入 MUX-SRL (b)优化工艺映射后的 MUX-SRL

Figure 6.1 (a) AND gate-feeding technology independent MUX-SRL

(b) Optimized and technology mapped MUX-SRL



## 6.2.2 基于底层实现的 $I_{DDQ}$ 测试考虑

上面一小节提到的是一些基于工艺实现的门电路或模块参数模型的可测条件下的优化综合。这里将工艺实现的底层晶体管级信息也考虑进来，因为它在做  $I_{DDQ}$  测试时是非常有用的。

由于对低功耗电路的强烈需求，加快了低压 CMOS ICs 的发展，也使电路规模和复杂性与日俱增，这对电路的测试是一个挑战，也是对  $I_{DDQ}$  测试的考验——正常静态下电路消耗的静态电流增大到可以和故障静态电流相比拟。这在第四章有详细的说明，同时也介绍了一些具体的控制技术。在逻辑综合阶段，我们至少可以有两件事情做，来提高电路的  $I_{DDQ}$  可测性。

一是提早运用  $I_{DDQ}$  测试规则和技术，对电路的底层实现进行约束，以确保  $I_{DDQ}$  测试在一定条件下能完成。如结构布局的安排、工艺的选择等。

另一件事情就是对底层结构信息的提取，以保证  $I_{DDQ}$  测试时的正常有利的驱动的生成。这样可以为  $I_{DDQ}$  测试取得更加有效的测试向量，提高故障覆盖率。

## 6.3 结构级可测性综合

### 6.3.1 增加测试点

比较传统的方法是在逻辑综合过程的最后，在整个电路的基本结构确定的情况下，由可测性分析，找出测试困难的地方，然后通过增加测试观测点来局部修改网络结构。通过增加一些测试点，可提高对整个电路的可控性和可观测性，一方面生成测试码容易些，缓解了 ATPG 的压力，有利于提高测试速度和故障覆盖率；另一方面它占用了有限的输入输出资源，当电路规模很大时，可能需要增加的测试引脚很多，给正常系统输入输出造成障碍。由于增加的测试点也需要连线输入输出，可能会给布线造成一定的难度，特别在布线资源紧张的情况下，值得考虑是否有必要。

例如，在图 6.2 所示电路模块中，给 Module1 的 degate 信号会使与 Module 2 相连的信号线 1 和信号线 2 上的值变得不可控，那么，可以在这儿加两个测试点，使它能控制性地驱动 Module 2，实现测试激励的目的。再者，如果 Module 1 的

degate 信号取相反的值，那么 Module 1 上的这两个输出线的值可直接在外部观测到。

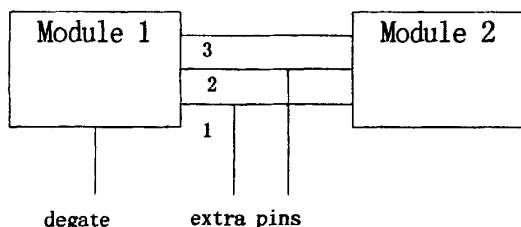


图 6.2 一个增加外部测试点的例子  
Fig6.2 An example of adding extra test points

在使用通路扫描的情况下，可以将内部模块间增加的测试点通过复用器或其他控制电路与扫描触发器连接，这时就不必为过多的内部测试点而发愁了，因为只需要共用扫描输入输出接口就行了。

### 6.3.2 扫描及扫描链选择的综合

扫描链路被广泛地应用于时序电路设计中以提高其可测性。扫描链路提供了对内部节点直接访问的接口，大大提高了对电路的可控性和可观测性，因而不必再为时序电路专门来生成测试码，也就省去了一件非常困难耗时的工作，因为通过扫描路径的设置，能有效地将对时序电路的测试转变为对一个组合电路的测试。

有许多的扫描技术被应用，它们各有优缺点。扫描设计总的说有如下一些负面影响要考虑：

- 1> 增加额外的电路使双稳电路可扫描，从而增加了整个电路的面积。
- 2> 扫描链路中的双稳电路的传播延时会增加，可能会使电路性能降低。
- 3> 需要增加测试用引脚。
- 4> 内部互连线的增加，造成面积增加，性能降低。
- 5> 测试码串行化，使测试时间增加。

这里提出一种扫描综合方法，称为“受益扫描”综合法，它在逻辑综合过程中选择扫描路径，通过共享功能块和测试逻辑电路使扫描链路所引起的面积增加和性能降低等负面影响降至最小。采用受益扫描方式选择的扫描链综合的电路都具有面积小、布线较容易(相对传统的 Muxed 触发器扫描路径)的优点。

## 1 概述

有很多不同的方案减小扫描设计所带来的额外花费，如精心地考虑扫描链路元件的选择能减少互连线或测试时间；放弃对整个电路采用全部扫描设计来取得完全可控和可观测，而用部分扫描设计方法能降低硬件开销，它是通过仅使系统的双稳态电路的一部分可扫描来实现的；无花费扫描试图通过组合逻辑来选取原始输入向量建立扫描通路，这些方法通常在电路被设计以后再解决减少测试开销的问题。

还有一些工作是在电路的综合过程中考虑插入扫描链路的。这里介绍的方法称之为受益扫描[53]，是将电路综合和扫描插入融为一体，了解电路的功能有利于扫描链路元件的选择，因为这个方法是让能共用的功能逻辑和测试逻辑尽可能共享，以达到减少扫描链路花费的目的。扫描链可认为是用 Muxed 的触发器实现的，允许测试逻辑与功能逻辑共享。

## 2 扫描元件关系的分类

既然每个触发器的输入方程是触发器输出和初始输入的函数，那么，同步电路中的触发器与电路中其它每个触发器必存在一定关系。这样的关系可能不重要，例如，触发器的输入不是另外一个触发器输出的函数。或者这种关系可能更为复杂，也可能简单到是否一个触发器的输入函数包括有另一个触发器的输出。这种关系也能基于更为复杂的特性，例如，输入函数是一个变量的正逻辑（或负逻辑）。

这些关系中，有些或所有允许测试逻辑与功能逻辑共享，这些就是“受益关系”；其它关系不允许逻辑共享，MUX 必须插入，使触发器可扫描，这些就是“非受益关系”。为了实现受益扫描，触发器间的关系根据 Shannon 展开定理确定。设  $FF_i$  和  $FF_j$  是扫描链上的两个相邻触发器， $FF_i$  的输出  $Q_i$  在电路正常工作下通过组合逻辑电路传给  $FF_j$  的输入  $D_j$ ，则  $D_j$  可由  $Q_i$  按 Shannon 展开如下：

$$D_j = Q_i f_{Q_i=1} + \bar{Q}_i f_{Q_i=0}$$

$f$  是输出给  $D_j$  的逻辑函数，根据这个表达式，可以将  $FF_j$  相对于  $FF_i$  归成几类，示于表 6.1。标记为 B 的关系为“受益关系”，constant 表示其取值为 0 或 1，T 为扫描测试选择信号，为 1 时为扫描模式，0 为正常状态。s, a 为任意函数。

例如 case 2 的实际电路可用图 6.3 左边表示, 前后两个触发器满足受益关系, 在选为扫描链时, 不需要将 FF<sub>j</sub> 用扫描专用触发器替换, 也不要另外再加大量的辅助电路, 仅一个与门(AND gate)就可使 FF<sub>j</sub> 可扫描了, 修改后的电路如图 6.3 右边的电路。

我们可以由综合系统自动完成扫描链上触发器之间关系的分类, 根据实际情况可节省测试的硬件开销, 在总体上达到减小面积的目的。

表6.1 触发器关系分类

*B* 表示两各处发器间存在受益关系

Constant 是 '0' 或 '1'.

$Q_i^*$  表示触发器输出可以反相

类型	$f_{out}$	$f_{in}$	方程式	新的扫描函数	开销
Case 0	h	h	$D_j = b$	$D_j = T'h + T'Q_i$	MUX
<i>B</i> Case 1	constant	constant	$D_j = Q_i^*$	$D_j = Q_i^*$	none
<i>B</i> Case 2	constant not constant	not constant constant	$D_j = Q_i^* + f_{out}$ $D_j = Q_i^* f_{out}$	$D_j = Q_i^* + T'f_{out}$ $D_j = Q_i^* (T + f_{out})$	AND OR
<i>B</i> Case 3	$f(Q_i)$	not constant	$D_j = Q_i^* \oplus f_{out}$	$D_j = Q_i^* \oplus T'f_{out}$	AND(OR)
Case 4	not constant	not constant	$D_j = Q_i f_{out} + Q_i' f_{out}$	$D_j = T'h + T'Q_i$	MUX
<i>B</i> Case 4S	not constant (s+a)	not constant (s'a)	$D_j = s Q_i^* + s' a$	$D_j = (s + T) Q_i^* + (s + T)' a$	OR

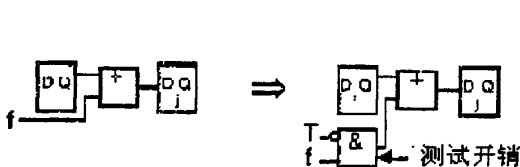


图 6.3 Case 2 的触发器例子  
Fig6.3 Example of case 2 flip-flop

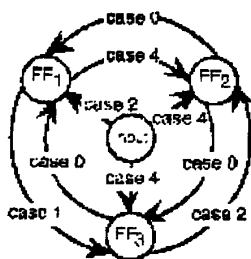


图 6.4 触发器关系图  
Fig6.4 Relationship graph of flip-flops

### 3 扫描链路选择

由于测试用的扫描链路可选择不同的时序元件形成, 而具体如何选择是一个可以综合的问题。根据上面的受益关系的分析, 我们可以为一个电路内部的所有触发

器之间的相互关系建立一个受益关系图，在这个图中，所有相邻的触发器之间可以按照数据流向有一条有向弧线，而弧线的权值可定义为在它们之间建立扫描链时的硬件开销，图 6.4 所示为一个触发器关系简图。

有了这个带权值的受益关系图，我们就可以利用自动综合系统来为我们找到一条代价最小的扫描链路。这个问题其实就是最小路径的问题，有很多很好的算法可以胜任这一任务。

### 6.3.3 时序电路内建自测试的逻辑综合

BIST 作为一种针对超大规模集成电路器件不断增长的测试复杂性的解决方案，近几年赢得了广泛的应用。电路的复杂性源于 VLSI 技术中设计规则尺寸的缩小，集成度的显著提高和逻辑复杂性的加大。另外一个重要因素就是行为模型综合系统的进步—运用自动化的设计综合技术，将设计从设计过程中提升到一个更高级别的抽象层面上，使之对电路底层的门级了解不再十分熟悉。另一方面，这些设计过程也显著地减少了实现系统功能所需的设计周期。手工为一个给定的系统设计一个有效的 BIST 方案实现完全的 BIST 功能（测试码生成，响应压缩，系统输入隔离和 BIST 序列的正确控制）会破坏由自动设计综合技术带来的优点，反而增加设计周期和给系统或 BIST 带来另外的不正确操作的风险。因此，BIST 的自动化技术刻不容缓，必须跟上自动设计综合能力步伐。尽管象随机存取器（RAMs）这样规则结构的电路实现 BIST 的技术得到了发展，目前已实现自动化，但由于缺乏针对一般时序逻辑 VLSI 器件和电路包的自动 BIST 技术，导致还没有一个完全的 BIST 自动实现系统。

#### 1 BIST 技术的结构和操作

运用这里的方法实现的 BIST 基本结构示于图 6.5。基本原理就是选择性地将系统中的记忆元件用特殊的 BIST 触发单元替换，然后将这些触发器单元互连形成一条 BIST 触发器的电路链。链中的签名分析寄存器(SAR)在系统作诊断时使 BIST 的使用非常有效。SAR 的位置安排可简化器件和系统测试的接口，不过，它可以放

在 BIST 触发器链中的任何位置。选择器(Multiplexers)可用来隔离系统数据和 BIST 电路，以便系统诊断过程中 BIST 执行的结果可在下次重用。除了隔离外，选择器还可将 BIST 链产生的测试码加在由系统输入驱动的正常电路上。最后需要一个控制电路，控制 BIST 序列和保持 SAR 中的结果，直到它被测试口读出。

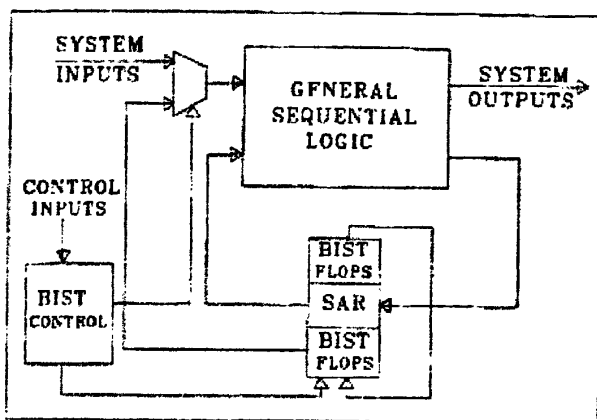


图 6.5 一种 BIST 结构  
Fig6.5 A kind of BIST structure

用于系统选择替换的 BIST 触发器、D 触发器示于图 6.6。这个 BIST 触发器用了一个异或门和每个输入由一个非门驱动。用两个 BIST 控制信号 “B0”, “B1”，可以实现 4 种运行模式（见表 6.2），包括初始化 (Initialization)、移位模式 (Shift Mode)、系统模式 (System Mode) 和 BIST 模式 (BIST Mode)，有点类似于基本内建逻辑块观测器 (BILBO) 单元的应用。

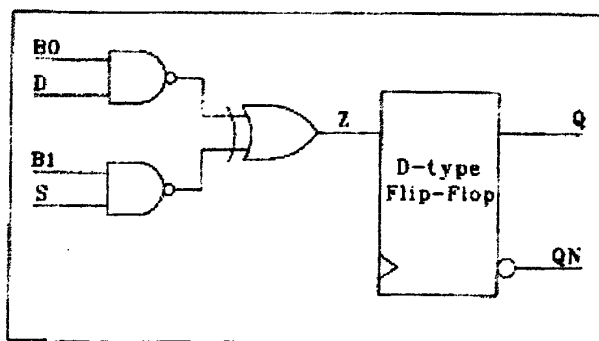


图 6.6 BIST 触发器  
Fig 6.6 BIST flip-flop

初始化模式是在 BIST 序列的开始设置触发器的输出到逻辑电平‘0’，初始化 BIST 链及整个被测的时序逻辑电路。移位模式是配置 BIST 链成为一个部分扫描链，用来获得增加故障覆盖率和 / 或测试正交度。系统模式使触发器的行为正常化为 D 触发器，用来执行系统功能。最后，BIST 模式对系统输入数据和 BIST 链中前级触发器的输出数据执行异或功能。

表6.2 BIST触发器运行模式

B0	B1	INPUT TO FLIP-FLOP	MODE
0	0	Logic Zero	Initialization
0	1	Value at Input S	Shift Mode
1	0	Value at Input D	System Mode
1	1	D exclusive-or S	BIST Mode

运行时，BIST 控制器在 BIST 控制序列的激活下将 BIST 触发器和 SAR 通过 B0 和 B1 在几个时钟周期后置为初始化模式。首先，将 BIST 触发器自己初始化（只需一个时钟周期），然后初始化被测电路中所有时序逻辑（被测电路中每级存储元件需要一个时钟周期）。这样，电路就进入 BIST 模式，BIST 链中的初始值被加到被测时序逻辑电路上。BIST 链输入端的结果值用链中初始化的值压缩，并通过 BIST 链中一个存储元件（异或函数）进行移位。这个值代表了第一次测试码施加后的签名结果，并作为下一个测试码加在被测时序电路上。每个运行时钟周期产生的签名用于测试电路下一次的输入，这个过程得以连续进行，此时，代表了整个时序逻辑电路故障信息的每一位传输到 SAR，进行正常的的数据压缩和签名分析，经过 BIST 模式下规定的时钟周期数后（由 BIST 控制器设计决定），SAR 不再进行进一步的数据压缩，直到其内容被测试器或系统诊断装置读完为止，这个 BIST 序列作废。

## 2 BIST 方法的其他方面

选择性地将系统中记忆元件用 BIST 触发器替换，利于减少由 BIST 带来的逻辑电路的额外花费。选择性地替换记忆元件的基本思想可见于图 6.7。在最简单情况下，触发器到触发器进行数据传送，只需要很少的测试码来测试这些触发器和它们

的互连。所以，由链中第一个 BIST 触发器产生的测试码将足以测试其后的触发器，这一思想可扩展到由小逻辑锥（如两输入与非门）驱动的记忆元件。测试与非门和它相连的触发器的测试码很好产生，而非门也不会太大地破坏测试码，所以，由与非门的触发器驱动电路可被很好地测试。输入数少的逻辑锥，如与非门，容易控制，只需将一输入置 1，相当于前后两个触发器直接相连。因此，可以不将被只有少数输入的逻辑锥驱动的记忆元件替换掉，由于 BIST 嵌入造成的额外逻辑花费就会最小化而不影响故障覆盖率。

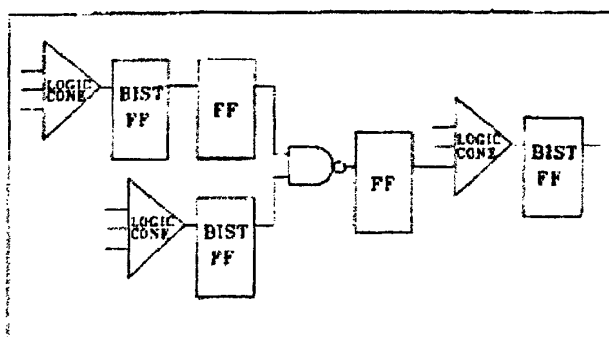


图 6.7 选择性地替换成 BIST 触发器  
Fig 6.7 Selective replacement for BIST flip-flop

另外，选择性替换有机会阻止将处于关键路径上的记忆元件替换掉，避免了额外的延迟的引入。

### 3 BIST 方法的自动实现

第一步 决定是否用 BIST 触发器替换记忆元件（面积考虑）

这可由软件自动实现，但设计者能选择逻辑锥的输入数目，用来决定选择性替代的所有记忆元件，由大于这个规定输入数目的逻辑锥驱动的记忆元件将被选择为用 BIST 触发器替换。既然选择性替换利于减少额外逻辑开销，指定一个较大的输入数将导致较少的触发器被替换，然而，指定的输入数太大，减少了 BIST 触发器的数目会达不到所希望的故障覆盖率。另外，也可能使 BIST 序列由于不能正确初始化整个时序逻辑电路而不具备获得自生成结果的能力。因此，推荐将有三或更多输入的逻辑锥驱动的记忆元件选择性替换。



### 第二步 剔除关键路径上的替换（时序考虑）

由软件系统选择的用 BIST 触发器替换的记忆元件可列表于一个文本文件中，以便设计者能查看并编辑。这时，设计者能将处于关键路径上的触发器从这个列表中删除，以防 BIST 触发器的应用阻碍了系统性能。同样，设计者也可根据需要把没有被选中的记忆元件加到这个替换列表中。这个文本文件也包含了决定哪个系统输入被隔离的信息。

### 第三步 不让相连的触发器同时为 BIST 触发器（逻辑考虑）

设计者在对选择性替换触发器列表文件修改后，可由软件系统来检查寄存器的相连情况，并排除这种情况的发生并使之最小化。BIST 方法中出现的一个简单的寄存器相连情况如图 6.8 所示。触发器连接在 BIST 链的相同序列上，在 BIST 模式下，XOR 门的输出将总是逻辑‘0’，然而，从第一个 BIST 触发器出来的数据包含有 BIST 链中先前检测的有关故障信息，这个故障信息将会由于寄存器的相连而丢掉，从而使 BIST 序列的故障覆盖率降低。一般，这种情况在正常的选择性记忆元件替代中很少出现，但在一些精细情况下，如计数器、复杂移位寄存器，寄存器的相连就会出现。

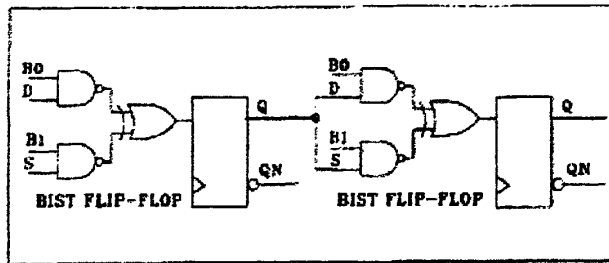


图 6.8 触发器相连的情况  
Fig 6.8 Example for flip-flop adjacency

由于 BIST 链中每位数据包含有故障信息，寄存器相连造成的单个一位数据的丢失可能导致故障屏蔽。当第一个触发器作为驱动第二个触发器的逻辑锥输入时，不要将这两个连续的触发器连接在一条链中。

#### 第四步 BIST 链的形成

将记忆元件中用 BIST 触发器替换的触发器相连, 形成 BIST 链, 增加输入隔离复用器。对 VLSI 实现, 可修改规范标准单元器件互连的网表; 对 PLD, 输入给 PLD 系统划分程序的文件修改为反映适当 BIST 电路的实现, 则 PLD 综合过程照旧。

### 6.4 逻辑门级电路可测性综合

当代 VLSI 电路增长的复杂性只有通过先进的 CAD 系统来控制管理, 这也是允许有效地操作布尔函数的一个重要关键特征。

最近几年, 决策图 DDs 作为一种布尔函数表达和操作的数据结构得到了越来越多的注意。它是 VLSI CAD 系统中最先进的数据结构, 他们在不同的应用中, 如验证、综合和测试, 得到了广泛的应用[44, 50, 54-58], 同时, 他们也集成到了商业工具中。

对于它们的数据结构形式和操作在第五章已介绍。下面就有关测试方面的问题对 DDs 进行研究。有些测试算法成功地选用 DDs 作为数据结构完成故障检测、同步化和内建自测试任务。以下主要集中在对 DD 表示派生的电路的可测试综合上。由不同类型的 DD 和它的结构特性, 可产生各种类型的 DD 电路, 针对静态和动态故障模型, 分析了他们的可测性特性。总之, 完整测试集和所有出现的冗余计算在基于 DD 操作算法下, 显得非常容易而有效, 可以得到具有高度可测性的电路。

DDs 中最普遍的类型是二叉决策图 BDDs。许多情况下, 他们在表达的压缩性和操作的有效性上能提供一个可接受的折中方案。BDDs 由基于 AND/OR 的布尔分解生成, 因而称之为 Shannon 分解, 最坏情况下, 综合操作与输入 BDD 大小成多项式关系。

然而, 有时 BDDs 还不足以解决所有问题, 如有些相关的布尔函数不能由 BDDs 有效地表达出来。因此, 在过去几年内, 有几种 BDD 扩展形式被提出来, 其

中就有基于 AND/EXOR 布尔分解的 DDs, 如 FDDs(Functional decision diagrams)和 KFDD[49]。FDDs 仅仅只是基于 AND/EXOR 布尔分解, 也叫 Davio 分解, KFDD 是一种 BDD 和 FDD 的统一形式, 融合了两种表达类型的优点。

#### 6.4.1 测试码自动生成和故障模拟

经典的 ATPG 算法是面向路径的, 意味着从故障位置开始, 确定信号值, 允许初始化和故障传播。当然, 由于 ATPG 是 NP 完全问题, 要想找一个普适的方案很困难, 因此, 它的搜索是由遗传方法来引导的。作为面向路径算法的一种备选方案, 基于 BDD 的 ATPG 方法(采用不同故障模型)得到了一些发展[59]。它的基本原理是可控性计算和符号化仿真的传播函数, 易于理解, 只要 BDDs 还小, 运行时间也很小。另外, 基于 BDD 的 ATPG 算法通常不仅仅提供单一一个故障测试码的计算, 而是一个给定故障所有的测试码。扩展这一概念是非常必要的, 因为在实际应用中常会碰到 BDD 很大的情况, 造成 ATPG 无法完成或非常耗时, 将基于 BDD 的方法和面向路径的方法结合起来就有了解决的可能性, 朝这个方向的有希望的进展可参考[60, 61]。

#### 6.4.2 可测性综合

近来基于 DD 的综合方法赢得了大量的兴趣, 所有的这些方法的一个共同点就是将 DD 翻译成电路, 用实现相应的分解模型的子电路替代 DD 中每个节点, 期望 DD 的结构特性和有效的 DD 操作算法能成功地用来优化和确定电路的特性。在考虑其可测性能及其与 DDs 的关系之前, 我们给出了几个例子证明这个概念的合理性。

将 KFDD(代替 BDD)映射成电路[56, 57], 除了标准逻辑综合工具中使用的 AND 和 OR 门外, 可能在综合过程中引入 EXOR 门。在[62, 63]中可发现其他基于 DD 用 EXOR 门的多级综合方法。总之, 基于 EXOR 的综合最近重新获得青睐, 主要是由于新的技术使这种实现更切实际, 例如, FPGA 技术使之使用多功能逻辑块成为可能, 而不再区分使用的逻辑的类型, 这要求把 EXORs 更多地集成到主流综合中。两级逻辑 AND/EXOR 实现的布尔函数通常较 AND/OR 实现的更为简凑, 复杂性降

低，利于测试码生成。由 KFDD 派生的多级电路，也叫 KFDD 电路，由 AND/EXOR 和 AND/OR 两种分解方式的组合组成。由于他们不是限定在仅仅只有一种分解类型上，他们能提供比 BDD 或 FDD 电路更为简凑的表达形式，这已在理论上证明，他们之间是存在指数级的差距的。而且 KFDD 是最为一般的 DD 类型。

### 6.4.3 KFDD 电路

KFDD 作为 BDD 更一般的形式，它也同样具有 BDD 图带来的许多优点。在电路综合中，我们可以利用 KFDD 数据特性，实现电路的可测性综合。

我们考虑的电路是定义在由初始输入(PI)和输出(PO)二输入与门 AND、或门 OR 和反向器 NOT 组成的标准库(STD)上。另外，我们也考虑两个其他的库。EXORLIB 库，由 STD 中标准单元加二输入异或门 EXOR 组成；KFDDLIB 库，包含实现 S 节点，pD 节点和 nD 节点分解的三种单元。例如，S 节点是一个(2, 1)复用器，带有 2 个数据输入端和一个控制输入口。KFDDLIB 库还包含有退化单元，它们是由 S, pD, nD 节点在只要有一个函数是 0 或 1 或节点的两个子函数相等或相反的情况下得出的，相应的这些 KFDD 节点称为退化节点。例如，nD 节点的退化单元的完整列表表示于表 6.3 (S 和 pD 节点的退化单元列表也可同样获得)。

表 6.3 nD 节点的退化单元

Subfunctions	Output behavior
$f_1^1 = 1, f_1^2 = 1$	$f = x_1$
$f_1^1 = 0, f_1^2 = 1$	$f = x_2$
$f_1^1 = 0$	$f = \overline{f_1^2} \cdot x_1$
$f_1^1 = 1$	$f = \overline{f_1^2} + x_1$
$f_1^2 = 1$	$f = f_1^1 \oplus x_1$
$f_1^1 = f_1^2$	$f = f_1^1 \cdot x_1$
$f_1^1 = \overline{f_1^2}$	$f = f_1^1 + x_1$

通过按拓扑顺序遍历相应的图，用库中相应的单元取代各个节点，很容易从 KFDD 图得到 KFDD 电路。按照 KFDD 的定义，布尔函数

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 \bar{x}_4 \oplus \bar{x}_1 \bar{x}_3 x_4 \oplus \bar{x}_1 x_2 (x_3 + x_4) \oplus x_1 x_2 \bar{x}_3 \bar{x}_4 \text{ 按分解方式 } d = (S, pD, nD, S)$$

的 KFDD 图如图 6.9a 所示，图 6.9b 的 KFDD 电路就是由图 6.9a 的 KFDD 图生成的，这种映射也可一般化到多输出函数的 KFDD 上。

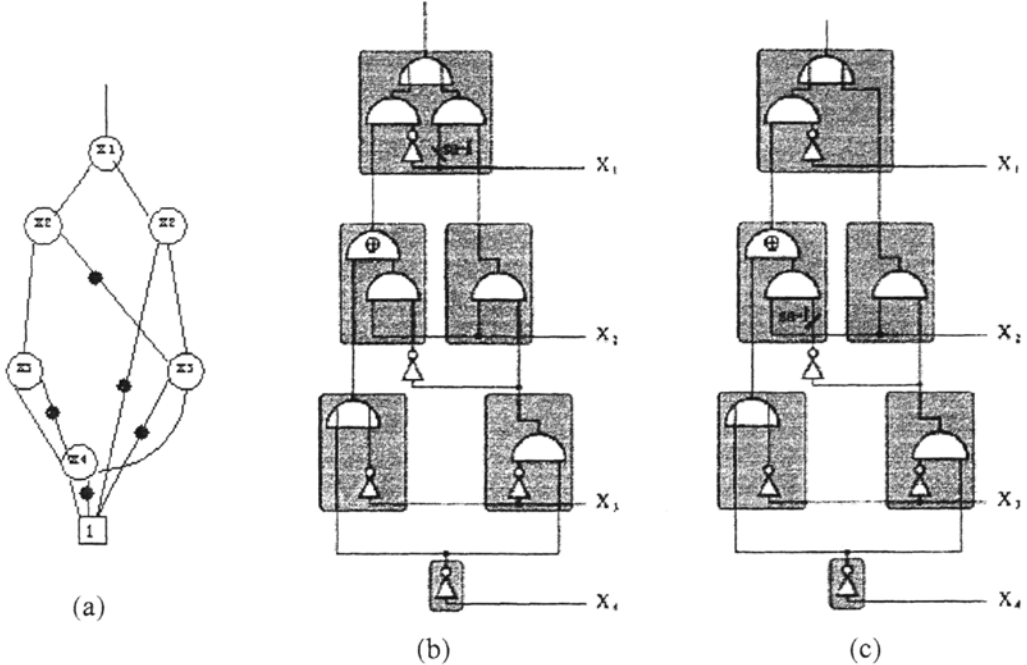


图 6.9 KFDD 图及其电路  
Fig 6.9 KFDD and its circuit

#### 6.4.4 KFDD 电路的可测性分析

这里针对的故障模型是单元故障模型 (cellular fault model, CFM) 和固定故障模型 (stuck-at fault model, SAFM)，其中 CFM 指故障刚好改变给定组合电路中一个节点的逻辑行为，而且改变后的逻辑行为仍是组合性的。对于固定工艺库上的电路，如果它是 CFM 完全可测的话，它也是 SAFM 完全可测的。如果电路节点  $v$  不存在 FM 形式的冗余故障，则称节点  $v$  完全 FM 可测。一个电路中所有节点完全 FM 可测，则该电路完全 FM 可测。一般的电路都是非冗余的，或至少是知道冗余位置的。

引理 1 设  $C$  是一个 KFDD 电路，则从电路上一个  $S$  节点 ( $pD$  节点或  $nD$  节点) 的输出端到  $PO$  的传播路径的建立所花时间与  $C$  大小成线性关系。

假设故障处于  $v$  点，按照深度优先搜索(depth first search, DFS)方式寻找通往初始输出点 POs 所到达的节点，并以 DFS 做标记。然后从一个可以到达的输出点 PO 开始向故障点回溯。如果碰到 S 节点或退化节点，通过适当设置控制变量的值，很容易让传播通过这个节点。如果到达非退化的 Davio 节点  $w$ ，则按如下处理：如果节点  $w$  在下级的左路祖先节点被标记为 DFS，选择  $w$  的左路输入数据，当且仅当  $w$  是 pD 节点 (nD 节点) 时，置  $w$  节点变量为 0(1)，这样可以阻塞  $w$  的右路输入数据。如果  $w$  在下级的左路祖先节点没有被标记为 DFS，选择  $w$  的右路输入数据，当且仅当  $w$  是 pD 节点 (nD 节点) 时，置  $w$  节点变量为 1(0) (注意，异或门 (EXOR) 在其一个输入为常值时，总是能将另一个输入传播出去的。另外，变量一旦被置位，将在后面不再会碰到，因为 KFDD 是有序的) 这样，我们就可以得到一个能将故障传播到初始输出端的测试码了。而这个计算过程的时间复杂性就是深度优先遍历 KFDD 的复杂性，就是 KFDD 的节点数。所以这个过程的时间花费与 C 的大小成线性关系。

作为 KFDD 的特例，BDD 具有更为优良的可测性质。如果电路是由 BDD 图生成的，假设某个节点中有故障，如节点  $v$  的  $\text{low}(v)$  (或  $\text{high}(v)$ ) 中有故障信息，根据 BDD 图的生成规律，可以很容易地设置节点  $v$  对应的变量  $x_i$  的值而让  $\text{low}(v)$  (或  $\text{high}(v)$ ) 中的故障传递到节点  $v$  中来；如果是节点  $v$  中存在有故障，我们知道  $\text{low}(v)$  和  $\text{high}(v)$  取值可以是任意组合，所以也一定能将节点  $v$  中的故障信息传递到节点  $v$  的上级节点中。因此，由最简 BDD 图生成的电路是完全可测的。

BDD 与测试码关系也容易掌握。我们可以将变量值  $x_1, \dots, x_n$  的集合看作图中一条从根节点开始的路径。在某处顶点  $v, \text{index}(v) = i$ ，如果  $x_i = 0$  则路径沿  $v$  的  $\text{low}$  子图延伸， $x_i = 1$  则路径沿  $v$  的  $\text{high}$  子图延伸。这样，一个具体测试码其实就是 BDD 图上的一条路径，而最终的 BDD 图反映了电路的结构，我们可以直接从 BDD 上得到具体故障的测试码，也就是对路径的选择。

要想对一般的 KFDD 电路进行可测性分析，首先必须清楚 KFDD 电路中每个节点的可控性，也就是每个 KFDD 节点可输入值的集合。这里定义了一个节点的可控性类别 CCs (controllability classes)，它是按照 KFDD 节点的两个输入组合可能的取值情况分类，具有相同输入组合的所有节点同属一个 CC。

引理 2 设 C 是一个 KFDD 电路。

- 1 C 中一个 KFDD 节点的输入数据是 0,1 可控的。在退化节点上, 可以是所有输入组合。相应初始输入 PI 指定值的计算所花时间与 PI 个数成线性关系。
- 2 非退化节点属于下列可控制类型的一种。

Controllability class	Applicable values at data inputs			
1	00	01	10	11
2		01	10	11
3	00	01	10	
4	00		10	11
5	00	01		11

定理 1 设  $C_{KFDD}$  是建立在库 KFDDLIB 上的 KFDD 电路。那么下面的成立:  
 $C_{KFDD}$  的冗余和 CFM 完全测试集的计算复杂性是  $O(|C_{KFDD}||KFDD_{max}|)$ 。

其中  $KFDD_{max}$  是计算 KFDD 图 G 中节点的 CCs 过程中出现的最大的 KFDD, 它的大小可与 G 的大小比较, 一般最多是  $|G|$  的平方。

定理 2 设  $C_{KFDD}$ ,  $C_{EXOR}$ ,  $C_{STD}$  是同一个 KFDD 图分别在库 KFDDLIB, EXORLIB, STD 上的 KFDD 电路, 则下列命题成立:

- 1  $C_{KFDD}$  和  $C_{EXOR}$  是 SAFM 完全可测, 每个测试码的计算时间是线性的。
- 2  $C_{STD}$  SAFM 完全可测, 当且仅当下列条件成立:
  - ① CC3 不包括 pD 或 nD 节点
  - ② CC4 为空
  - ③ CC5 不含 S 节点

在图 6.9b 的电路中, 由计算可知由变量  $x_1$  控制的 S 节点的数据输入不可能是 01, 所以该节点属于 CC4 可控性类型, 这时, 在右边与门输入控制线上的 stuck-at-1 故障就是冗余的, 一般根据 CCs 的划分, 总可以将 KFDD 电路的这种冗余找出并移走。不过移走这类冗余, 有时会引入新的冗余, 这是由于改变了 KFDD 电路的传播

特性的缘故。如果将图 6.9b 电路中的 stuck-at-1 冗余消除后得到图 6.9c 的电路，在新的地方产生了 stuck-at-1 冗余——在故障点初始化故障后，故障信息无法穿过根节点的或门。

一般 KFDD 电路即使在不使用冗余消除技术的情况下，也可在很短时间内得到节点可测性类型 CCs 和达到很高的固定故障覆盖率。如果根据 KFDD 电路自身的特性运用针对性的冗余消除技术，会达到更好的效果。

## 6.5 小结

深亚微米条件下的电路系统的复杂性要求我们必须借助于计算机辅助工具来完成将电路的设计，而可测性设计是当今集成电路设计中不可或缺的一个重要环节，所以，将它纳入自动化设计流程是一件非常有实际意义的事。首先它能保证在设计较高抽象层次上开始考虑电路的可测性问题而不必拘泥于许多细节性的工作，这对复杂大系统的设计是非常必要，可以减轻可测性设计带来的不堪重负而更多地专注于系统性能和功能的设计。其次，可测性设计集成在自动综合系统中，能让可测性的约束自始至终地跟踪整个设计流程，能有力地确保最终设计出来的电路是可测的。而且利用计算机强大的计算处理能力，在规则约束下，可以有很多的优化算法能帮助设计者在合理的时间内寻找最佳或满意的可测性设计方案。特别在对芯片面积和速度非常敏感的设计中，在逻辑综合阶段直接对可测性的考虑和综合优化，能起到更好的效果。如果利用传统综合方法的特点及底层数据信息，可以更好地检测模拟电路的故障，改善电路的可测试性问题。

本章论述了各种可测性设计方案自动综合实现的技术，并结合逻辑综合中的面积和时延的考虑，可以得到一个可测性自动优化系统。另外，对于门级电路本身的可测性问题在逻辑综合阶段可根据 KFDD 电路的特性作具体分析，而且这种电路是容易利用底层信息控制其可测性的，相应的测试码由 KFDD 图也很好获得。



## 第七章 总结

### 7.1 本文研究内容及其创新

本文详细阐述了深亚微米集成电路的测试问题。首先分别对组合逻辑电路和时序逻辑电路的测试方法作了深入的介绍和研究,并在此基础上提出了一种可用于复杂大规模集成电路测试和故障诊断的矩阵扫描 **BIST** 电路结构。

由于基于电压观测的逻辑方法还是无法满足集成电路工艺水平提高后的故障测试要求,一种基于 CMOS 电路静态电流的检测的测试方法应运而生,它能解决许多电压法难以解决的测试问题。而随着电路要求的提高和 CMOS 工艺进步,  $I_{DDQ}$  测试法也由于静态电流的增加而受到挑战。另外,有些具体的电路结构对  $I_{DDQ}$  测试有抵抗性。本文根据具体的电路结构提出了一些灵活运用  $I_{DDQ}$  测试的技巧。

为了把可测性设计带入综合系统中,本文简要介绍了逻辑综合的过程及方法。我们提出了一种逻辑综合中常用的数据结构 **BDD** 的优化方法,他能找到最优的变量顺序使 **BDDs** 更小。

本文最后研究了可测性设计方法的自动综合实现途径,意在将可测性设计化为综合过程中的约束条件来影响原来的综合行为。再加上常规的综合目标的考虑,就可初步实现一个自动化的可测性设计综合系统了。此外,我们分析了用于逻辑综合的数据结构的特性,可以帮助我们对电路可测性的控制,实现底层可测性综合。

### 7.2 今后研究方向

可测性设计在深亚微米集成电路中的作用将越来越重要,而且随着工艺水平的进步和逻辑复杂性的提高必须不断发展。基于扫描的 **BIST** 技术被证明是一种解决复杂逻辑测试和系统测试的有效方案,而对电路高质量的需要要求它有更高的故障覆盖率以及合理的结构,并为今后系统级的测试提供标准接口。

$I_{DDQ}$  测试法由于依赖于静态电流的观测,发展优良的电流传感电路会有利于  $I_{DDQ}$  测试的效果。如何将低压下的正常静态电流有别于故障静态电流,是一个很值

得研究的方向。

集成电路随着规模的增大而逐渐走向设计的自动化，可测性设计也应该随着深亚微米时代的来临而迈向自动设计流程。这样可以更快更好地帮我们实现超大规模集成电路的可测性设计，实现全局的优化。

## 参考文献

- [1] Crouch Alfred L. Design For Test For Digital IC's and Embedded Core Systems. NJ, Prentice Hall PTR, 1999, pp.93-174.
- [2] Wang Francis C. Digital Circuit Testing A Guide to DFT and Other Techniques. CA, 1991, pp.1-35.
- [3] Williams T.W, Parker K.P. Design for testability— a survey. IEEE Proc. 1983, Vol.71(1), pp.383-398.
- [4] Ferré A, Isern E, Rius J, etc.  $I_{DDQ}$  Testing: state of the art and future trends. INTEGRATION, the VLSI journal, 1998, vol.26, pp.167-196.
- [5] 薛宏熙, 边计年, 苏明. 数字系统设计自动化. 北京, 清华大学出版社, 1996, pp.303-349.
- [6] Hideo Fujiwara, Takeshi Shimono. On the Acceleration of Test Generation Algorithms. IEEE Transactions on Computers, 1983, vol.C-32(12), pp.1137-1144.
- [7] 刘丽华, 辛德禄, 李本俊. 专用集成电路设计方法. 北京, 北京邮电大学出版社, 2000, pp.1-230.
- [8] Cheng K.T, Agrawal V.D. A partial scan method for sequential circuits with feedback. IEEE Transactions on Computer-Aided Design, 1990, vol.39(4), pp.544-548.
- [9] Kiefer Gundolf, Wunderlich Hans-Joachim. Deterministic BIST with partial scan. Journal of Electronic Testing: Theory and Applications, 2000, vol.16, pp.169-177.
- [10] Lee K J, Chen J J, Huang C H. Broadcasting test patterns to multiple circuit. IEEE Transactions on Computer-Aided Design, 1999, vol.18(12), pp.1793-1802.
- [11] Pradhan, D.K, Saxena A. novel scheme to reduce test application time in circuit with full scan. IEEE Transactions on Computer-Aided Design, 1995, vol.14(12), pp.1577-1586.
- [12] Norwood Robert B, McCluskey Edward J. High-Level Synthesis for Orthogonal Scan. In VLSI Test Symposium, Monterey CA, 1997, pp.370-375.
- [13] David René. Random Testing of Digital Circuits Theory and Applications. NY, Marcel Dekker Inc, 1998, pp.1-334.
- [14] Nadeau-Dostie B, Burek D, Hassan A. Scan Bist: A multifrequency scan-based BIST method. IEEE Design&Test on Computer, 1994, vol.11(1), pp.7-16.
- [15] Rajski J, Tyszer J. fault diagnosis in scan based BIST. In Proceedings International Test Conference, Washington, DC, 1997, PP.894-902.
- [16] Aitken R C, Agarwal V K. A diagnosis method using pseudo-random vectors without intermediate signatures. In Proceeding of IEEE ICCAD, Santa Clara, California, 1989, pp.574-577.
- [17] Waicukauskis J A et al. Failure diagnosis of structured VLSI. IEEE Design&Test on Computer, 1989, vol.6(3), pp.49-60.
- [18] Wu Y, Adham S M I. Scan-Based BIST fault diagnosis. IEEE Transaction on Computer-Aided Design, 1999, vol.18(2), pp.203-211.
- [19] Mathew Ben, Saab Daniel G. Combining Multiple DFT Schemes With Test Generation. IEEE Transaction on Computer-Aided Design, 1999, vol.18(6), pp.685-695.
- [20] Wunderlich Hans-Joachim. BIST for systems-on-a-chip. INTEGRATION, the VLSI journal, 1998, vol.26(1-2), pp.55-78.
- [21] 黄建文. 微电子电路设计原理及应用. 北京, 中国铁道出版社, 1999, pp.1-150.
- [22] Chen J C and Womack B F. A study of fault signatures for diagnostics. In Proceeding of ISCAS, New Orleans, Louisiana, 1990, pp.2701-2704.
- [23] Maxwell P, Aitken R, Kollitz K, etc.  $I_{DDQ}$  and AC scan: the war against unmodeled defects. In Int. Test Conf., Washington DC, 1996, pp. 250-258.
- [24] Levi M.W. CMOS is most testable. In Int. Test Conf., 1981, pp.217-220.
- [25] Isern E, Figueras J.  $I_{DDQ}$  test and diagnosis of CMOS circuits. IEEE Design&Test of Computer, 1995, vol.12(4), pp.60-67.
- [26] Athan S P, Landis D L, Al-Arian S A. A Novel Built-in Current Sensor for  $I_{DDQ}$  Testing of Deep Submicron CMOS ICs. In 14<sup>th</sup> VLSI Test Symposium, Princeton NJ, 1996, pp.118-123.

- [27] Gyvez J P, Wetering E V. Average Leakage Current Estimation of CMOS Logic Circuits. In 19<sup>th</sup> IEEE Proceeding on VLSI Test Symposium, Marina CA, 2001, pp.375-379.
- [28] Guez R R, Bruls E, Figueras J. Bridging defects resistance in the metal layer of a CMOS process. Journal of Electronic Testing: Theory and Application, 1996, vol.8, pp.35-46.
- [29] Lee K J, Breuer M A. Design and test rules for CMOS circuits to facilitate  $I_{DDQ}$  testing of bridging faults. IEEE Transactions on Computer-Aided Design, 1992, vol.11 (5), pp. 659-670.
- [30] Sachdev M. Testing Defects in Scan Chains. IEEE Design & Test of Computer, 1995, winter, pp.45-51.
- [31] Sachdev M.  $I_{DDQ}$  and voltage testable CMOS Flip-flop configuration. In Int. Test Conf., Washington DC, 1995, pp.534-543.
- [32] Yamazaki H, Miura Y.  $I_{DDQ}$  testability of flip-flop structures. In Int. Workshop on  $I_{DDQ}$  Testing, Washington DC, 1996, pp.29-33.
- [33] Guez R R, Figueras J. Bridges in sequential CMOS circuits: current-voltage signature. In VLSI Test Symposium, Monterey, CA, 1997, pp.68-73.
- [34] Champac V H, Rubio J A, Figueras J. Electrical model of the floating gate defect in CMOS ICs: implications on  $I_{DDQ}$  testing. IEEE Transactions on Computer-Aided Design, 1994, vol.13 (3), pp.359-369.
- [35] Ferguson F J, Larrabee T. Test pattern generation for realistic bridge faults in CMOS ICs. In Int. Test Conf., 1991, pp.492-499.
- [36] Song P, Lo J C. Testing the realistic bridging faults in CMOS circuits. In Int. Workshop on  $I_{DDQ}$  Testing, Washington DC, 1996, pp.84-88.
- [37] Mao W, Gulati R K, Goel D K, etc. QUIETEST: a quiescent current testing methodology for detecting leakage faults. In Int. Conf. on Computer Aided Design, Santa Clara, CA, 1990, pp.280-283.
- [38] Chakravarty S, Thadikaran P. Simulation and generation of  $I_{DDQ}$  tests for bridging faults in combinational circuits. In VLSI Test Symp., Atlantic, NJ, 1993, pp.25-32.
- [39] Debaney W H. Coverage of node shorts using internal access and equivalence classes. Journal of VLSI Design, 1993, vol.1 (1), pp.71-86.
- [40] Szekeley V, Rencz M, Torok S, etc. Cooling as a possible way to extend the usability of I testing. *Electronic Letter*, 1997, vol.33, pp.2117-2118.
- [41] Sachdev M. Separate I testing for signal and bias paths in CMOS IC's for defect diagnosis. Journal of Electronic Testing: Theory and Application, 1996, vol.8, pp. 203-214.
- [42] Chen Zhanping, Wei Liqiong, Kaushik Roy. On Effective  $I_{DDQ}$  Testing of Low-Voltage CMOS Circuits Using Leakage Control Techniques. IEEE Transactions on Very Large Scale Integration Systems, 2001, vol.9(5), pp. 718-725.
- [43] Jongeneel Dirkjan, Otten Ralph H.J.M. Technology mapping for area and speed. INTEGRATION, the VLSI journal, 2000, vol.29, pp.45-66.
- [44] Bryant R E. Graph-Based Algorithms for Bloolean Function Manipulation. IEEE Transaction on Computer-Aided Design, 1986, vol.C-35(8), pp.677-691.
- [45] Malik S, Wang A R, Brayton R K, et al. Logic verification using binary decision diagrams in a logic synthesis environment. IEEE Int. Conf. on Computer-Aided Design, Santa Clara CA, 1988, pp.6-9.
- [46] Scholl C, Möller D, Molitor P. BDD Minimization Using Symmetries. IEEE Transaction on Computer-Aided Design, 1999, vol.18(2), pp.81-100.
- [47] Jankovic D, Günther W, Drechsler R. Lower Bound Sifting for MDDs. In 30<sup>th</sup> IEEE Int. Symp. on Multiple-Valued Logic, Los Alamitos, CA, 2000, pp.193-198.
- [48] Drechsler R, Drechsler N, Günther W. Fast Exact Minimization of BDD's. IEEE Transaction on Computer-Aided Design, 2000,19(3), pp.384-389.
- [49] Kechschull U, Schubert E, Rosenstiel W. Multilevel logic synthesis based on functional decision diagrams. In Proc. European Conf. on Design Automation, Brussels, Belgium, 1992, pp. 43-47.

- [50] Drechsler Rolf, Becker Bernd. Ordered Kronecker Function Decision Diagrams——A Data Structure for Representation and Manipulation of Boolean Functions. *IEEE Transactions on Computer-aided Design* , 1998, vol.17(10), pp.965-973.
- [51] Chickermane Virek, Zarrinch kamran. Addressing Early Design for Test Synthesis in a Production Environment. In *Proc. Int. Test Conf.*, Washington, DC, 1997, pp.246-255.
- [52] Stroud C E. An Automated BIST for General Sequential Logic Synthesis. In *25<sup>th</sup> ACM/IEEE Design Automation Conference*, Anaheim, CA, 1988, pp.3-8.
- [53] Norwood R B, McCluskey E J. Synthesis-for-Scan and Scan Chain Ordering. In *14<sup>th</sup> VLSI Test Symposium*, Princeton, NJ, 1996, pp.87-92.
- [54] Bryant R E. Symbolic Boolean manipulation with ordered binary decision diagrams. In *ACM Comput. Surveys* 24 ,1992, pp.293–318.
- [55] Becker B, Drechsler R. Decision diagrams in synthesis – algorithms, applications and extensions. In *Proc. VLSI Design Conf.*, Hyderabad, India, 1997, pp. 46–50.
- [56] Hengster Harry, Drechsler Rolf, Eckrich Stefan. AND/EXOR-Based Synthesis of Testable KFDD-Circuits with Small Depth. In *IEEE, Proceedings of ATS'*, Hsinchu, Taiwan, 1996, pp.148-154 .
- [57] Becker Bernd, Drechsler Rolf. Synthesis for Testability: Circuits Derived from Ordered Kronecker Functional Decision Diagrams. In *European Design and Test Conference*, Paris, France, 1995 , pp.592.
- [58] Becker Bernd. Testing with decision diagrams. *INTEGRATION, the VLSI journal*, 1998, vol.26, pp.5-20.
- [59] Drechsler R, BiTeS: a BDD based test pattern generator for strong robust path delay faults. In *Proc. European Design Automation Conf.*, 1994, pp.322–327.
- [60] Cho H, Jeong S, Somenzi, F C. Pixley, Synchronizing sequences and symbolic traversal techniques in test generation. *Journal of Electronic Testing : Theory and Application*, 1993, vol. 4 , pp.19–31.
- [61] Corno F, Prinetto P, Rebaudengo M, etc. Comparing topological, symbolic and GA-based ATPGs: an experimental approach. In *Proc. Int. Test Conf.*, Washington, DC, 1996, pp.39–47.
- [62] Sasao T, Hamachi H, Wada S, etc. Multi-level logic synthesis based on pseudo-Kronecker decision diagrams and local transformation. *IFIP WG 10.5 Workshop on Applications of the Reed–Muller Expansion in Circuit Design*, 1995, pp. 152–160.
- [63] Tsai C C, Marek-Sadowska M. Logic synthesis for testability. In *Proc. Great Lakes Symp. VLSI*, Ames, IA, 1996, pp. 118–121.

## 致 谢

感谢我敬爱的导师林争辉教授在我学业上的悉心指导，工作和生活上的关怀。先生的严谨治学态度和不断进取的精神激励着我，使我坚持不懈地进行研究，成为我人生历程中的宝贵财富。在本文的写作过程中，从论文选题立意到开题写作，直至定稿，先生都花费了大量心血。先生对我论文发表情况非常关注，并提供了许多有价值的修改意见，使我得以顺利完成研究任务。林老师的教导恩情毕生难忘。

同时也感谢两位授课老师，李世煜老师和秦建业老师，他们传授的专业知识也是本文能够完成的基础。

另外，我还要感谢在学习和工作上给予了我帮助的师友们，他们在研究中给我提供了许多敏锐独到的意见和建议，对于他们的帮助我将铭记在心。

我还要对我的同窗好友们致以深切的谢意，是他们使我的生活变得丰富多彩，也策动着我不断进取，感谢热情帮助过我和资助过我的朋友们，我将永远不会忘记。

同时感谢从事办公室，机房管理的老师对我研究工作给予的方便和支持。

## 攻读学位期间发表论文情况

1. 罗春桥, 林争辉 基于 BIST 矩阵扫描的一种 VLSI 故障诊断策略 上海交通大学学报
2. 罗春桥, 林争辉 利用变量相关性快速建立 ROBDD 上海交通大学学报

注:

第一篇已录用, 第二篇在审稿中。