逻辑测试和可测性设计

计算机技术编辑部

译 者 的 话

计算机在社会各领域的大量应用，要求计算机工作可靠。保证计算机的可靠性，很大程度上依赖于测试技术准确地判断机器中使用的集成逻辑电路，是否制造合格和具有正确的功能。然而随着LSI、VLSI电路密度急剧增长，使测试变得越来越困难。如何经济而有效地测试逻辑LSI、VLSI，已成为设计和应用LSI、VLSI的难题之一。

过去，系统设计较多地考虑把硬件的复杂性减至最低程度；而在电路设计完成之后才考虑测试问题。现在，由于硬件成本不断下降，测试费用不断上升，通过可测性设计，在电路芯片增加一些额外电路，以降低其测试的复杂性；在VLSI设计的同时考虑可测性设计，已引起人们极大的兴趣和关注。实践已证明可测性设计是解决测试逻辑VLSI的一条有效途径。

有关逻辑测试技术和可测性设计方面的专著不多，麻省理工学院出版的计算机系列丛书之一的“Logic Testing and Design for Testability”，较系统地阐述了逻辑测试技术和可测性设计的理论和实践。我们翻译出版这本资料，旨在给想要了解逻辑测试技术和可测性设计原理，并打算把这些技术应用于设计中的计算机系统设计，逻辑设计，电路设计和测试设计的工程技术、人员参考。本资料分两大部分。第一部分论述逻辑测试技术的基本原理、基本算法、测试生成故障模拟及测试的复杂性问题等。第二部分论述可测性设计方法，使测试应用和测试生成费用减至最低的设计技术，时序逻辑电路的扫描设计，压缩测试，内构测试及可自测系统等各种设计技术。

限于我们的知识水平和实践经验，尽管我们力求译校准确，但一定还会存在一些不妥的地方，希望读者给予批评指正。

本资料第一、二章由程昭，第三、四章由王旭东，第五、六、七章由陈开华，第八、九章由贾立宗译出。李经纬对全部资料作了校阅。原文内明显的错误，已作了订正。

逻辑测试和可测性设计

第一章　逻辑测试技术简介

1.1 逻辑电路……（1）
1.2 构造故障模型……（3）
1.3 测试问题……（5）
1.4 测试方案……（8）

第二章　测试生成

2.1 布尔差分……（11）
2.2 D 算法……（13）
2.3 PODEM 算法……（19）
2.4 FAN 算法……（22）
2.5 时序电路的测试生成……（29）

第三章　故障模拟

3.1 模拟方法论……（35）
3.2 并行故障模拟……（36）
3.3 演绎故障模拟……（37）
3.4 同时故障模拟……（41）
3.5 硬件模拟器……（44）

第四章　测试复杂性

4.1 NP—完全性……（46）
4.2 多项式时间类……（49）
4.3 故障条件下的闭合性……（52）

第五章　可测性设计入门

5.1 可测性……（56）
5.2 最低测试费用设计……（59）
5.3 组合逻辑与时序逻辑……（61）
5.4 特定设计和构造设计……（62）

第六章　最小测试操作费用设计

6.1 异或门嵌入法……（65）
6.2 最小可测试设计……（68）
6.3 双方式逻辑……（70）
6.4 固定参考值测试……（72）

第七章　最小测试生成费用设计

7.1 分割法和穷举测试……（74）
7.2 错症值可测设计……（78）
7.3 Reed-Muller 规范形式……（80）
7.4 可编程逻辑阵列……（82）

第八章　时序电路扫描设计

8.1 可移状态机……（89）
8.2 扫描设计方法……（92）
8.3 扫描设计的变型……（97）
8.4 不完全扫描设计和增强扫描设计……（99）

第九章　内部自测试设计

9.1 特征码分析……（102）
9.2 内部逻辑块观测器……（105）
9.3 具有扫描设计的自测试……（110）
9.4 自验证……（113）

第一章　逻辑测试技术简介

集成电路技术极大地促进了计算机的发展。电路密度已显著增加，器件成本随着其性能的改进而降低。在这些技术发展的同时，可靠性也变得日益重要。然而随着超大规模集成（VLSI）技术的出现，测试技术碰到了难以逾越的障碍而停滞不前。由于VLSI电路的门密度增长比电路引出端数目的增长要快得多，因而生成测试图形和进行故障模拟的能力越来越差。通过开发更快、更有效的测试图形生成算法或者采用增强可测性的设计技术，可以减少测试的困难。

1.1　逻辑电路

逻辑电路由相互连接的一些称为“门”的元件构成，这些门的输入值和输出响应仅用0和1两个值来表示。常用的一些门是与门（AND）、或门（OR）、非门（NOT）、与非门（NAND）、或非门（NOR）和异或门（EOR），它们的符号分别示于图1.1中。每一个门的输出，可以用输入的逻辑函数或布尔函数来表示。“逻辑”（logic）和“布尔”（Boolean）这两个术语常常用来表示相同的意思。一个逻辑函数可以用一个真值表、一个卡诺图、或一个立方集来表示。图1.2中，分别给出了这三种表示法的范例。布尔（逻辑）操作符“·”，“+”，和“—”分别对应于与，或和非操作。

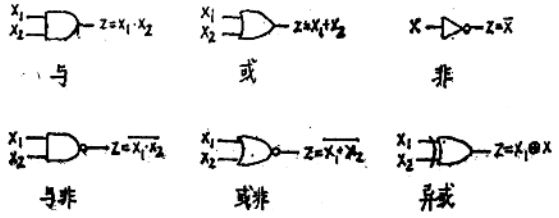


图1.1　逻辑门的符号

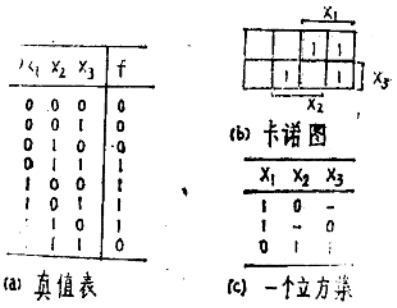


图1.2　逻辑函数的表示法

对于一个具有输入X_1和X_2的与门，当且仅当它的两个输入同时为1时，其输出才为1。这可以表示为$Z=X_1 \cdot X_2$。

对于一个具有输入X_1和X_2的或门，当且仅当它的任何一个输入为1时，它的输出Z便为1。这可以表示为$Z=X_1+X_2$。

对于一个具有输入X的反相门，当且仅当它的输入为0时，其输出Z为1，这可以表示为$Z=\overline{X}$。

对于一个具有输入X_1和X_2的与非门，当且仅当它的任何一个输入为0，其输出便为1。这可以表示为$Z=\overline{X_1 \cdot X_2}=\overline{X_1}+\overline{X_2}$

对于一个具有输入X_1和X_2的或非门，当且仅当它的两个输入均为0时，其输出Z才为1。这可以表示为：$Z=\overline{X_1+X_2}=\overline{X_1}\cdot\overline{X_2}$

对于一个具有输入X_1和X_2的异或门，当且仅当它的两个输入不相同时，其输出Z才为1。这可以表示为$Z=X_1\cdot\overline{X_2}+\overline{X_1}\cdot X_2=X_1\oplus X_2$。

逻辑电路可分为组合（Combinational）电路和时序（Sequential）电路两大类。一个组合（逻辑）电路，由一组不带反馈回路的，相互连接的门组成。（所谓反馈回路，指从某个门G的输出到其一个输入的直接通路。）在

某一给定时刻，组合电路输出值仅仅取决于现时的输入值，因此，它的每一个输出，能由它的输入变量的函数来规定。在图1.3中示出一个组合逻辑电路的方框图，其中X_1……，X_n是输入，而Z_1……Z_m是输出

图 1.3组合逻辑电路框图

时序（逻辑）电路中包含了反馈回路。在某给定时刻，时序电路的输出值不仅依赖于现时的输入，也依赖于先前所施加的输入。先前输入的历史情况，用电路的状态（State）来概括。图1.4示出时序电路的方框图。

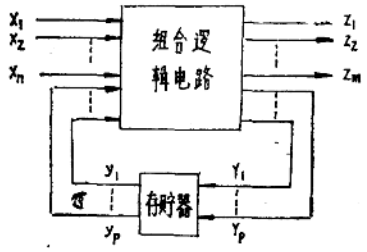


图 1.4时序电路方框图

一个时序电路由两部分组成：一个组合电路和一个存贮电路。图1.4中的电路具有n个原输入X_1，……X_n；m个原输出Z_1，……，Z_m；P个反馈输入y_1，……，y_p；P个反馈输出Y_1，Y_p。而Y_1，……Y_p是存贮电路的输入，y_1……y_p是存贮电路的输出。这个电路的现时状态由变量y_1，……y_p表示，而它的下一个状态由Y_1，……，Y_p确定。

时序电路的数学模型被称为时序机（Sequential machine）或有限状态机（finite state machine）。

一个时序机M由下列各项来表征：

有限状态集S，

有限输入符号集I，

有限输出符号集O，

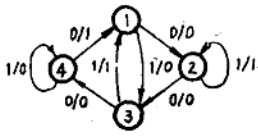
SXI的N变换为S（称为下一状态函数），和SXI的Z变换为O（称为输出函数）。

时序机M用五元组（S，I，O，N，Z）来表示。

一个时序机可以方便地用状态表（State table‘也叫流程表flow table）的列表方式来表示，或者用状态图（state diagram）的图形方式来表示。正如图1.5（a）中所示，状态表列出现时状态作为行标题，输入符号（也叫作输入值或输入状态）作为列标题。在S_i行和I_j列的表目表示下一状态N（S_i，I_j）及输出，Z（S_i，I_j）。在图1.5（a）中示出的时序机，给出四个状态（标为1，2，3，4）及二进制的输入和输出。图1.5（b）则给出与图1.5（a）中的状态表相对应的状态图。状态图是一个指向图，图中的结点对应时序机的状态，图中的弧线表示状态的转变。每一根弧线标有与状态转变相对应的，用斜线分开的输入值和输出值。

	0	1
2	2，0	3，0
3	3，0	2，1
4	4，0	1，1
1	1，1	4，0

(a)状态表



(b)状态图

图 1.5 时序机的表示法

根据电路的动作是否受控于离散的时刻，将时序电路分为同步（synchronous）时序电路和异步（asynchronous）时序电路。同步时序电路的工作，由被称为时钟脉冲（clock pulse），或简称为时钟（clock）的同步脉冲信号控制。时钟一般加到电路的信息存贮部分。图1.6示出同步时序电路的方框图。在同步时序电路中，所用的一系列的双稳态存贮元件称为钟控触发器（clocked flip-flops即FF）最流行的存贮元件如图1.7所示的D触发器（D的意思是Delay延迟），T触发器（T的意思是Trigger翻转触发器），SR触发器（SR意思是set--Reset置位--复位）和JK触发器。图1.8示出一个SR触发器的具体实现。其他的触发

器可类似地由与非门或者或非门交叉耦合而构成。

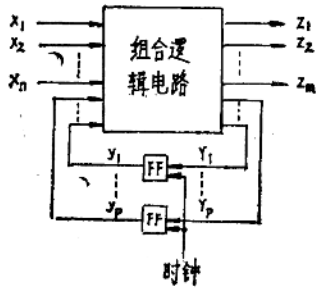


图 1.6 同步时序逻辑电路方框图

异步时序电路的动作是不同步的，它不受时钟控制。如图1.9中所示，每根反馈线假定具有一个有限的、正的纯延迟。这样的异步时序电路要能正确工作，要求以下条件：

·由于存在延时，组合电路可能会产生一个瞬时错误或尖峰信号，称为险象（hazard）。

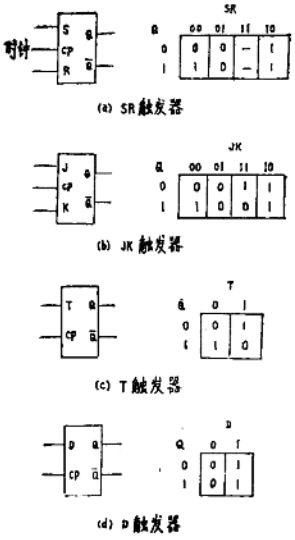


图 1.7 触发器的表示法

如果这样的一个错误，加到非时钟控制的触发器或锁存器的输入端，会导致永久性的错误状态。因此，该电路的组合逻辑部分，应设计成无险象的电路。

·要限制输入只有当全部存贮元件都处于

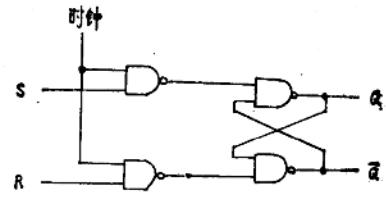


图 1.8 SR触发器的实现

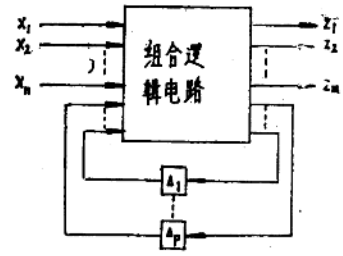


图 1.9 异步时序逻辑电路框图

稳定状态（即对所有i值，$y_i=Y_i$）时，状态才可以改变。这叫作基本操作方式（fundamental mode operation）。

·在电路状态转换期间，有两个或两个以上的状态变量发生变化，这种情况叫作竞态（race）现象。如果电路的正确动作依赖于竞态的结束，这种竞态被称为临界竞态。为了保证电路的工作不受瞬态变化的影响，应适当地进行状态分配以避免临界竞态。

1.2 构造故障模型

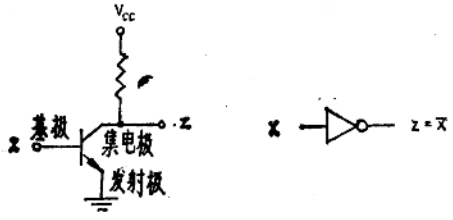
逻辑门由晶体管实现时，晶体管分为双极型晶体管和金属氧化物半导体场效应晶体管（MOSFET，或简称MOS）。以双极型晶体管为基础的逻辑电路系列有晶体管——晶体管逻辑（TTL），射极耦合逻辑（ECL）等等。以MOSFET为基础的某些逻辑电路系列是P—沟道MOSFET（P—MOS）逻辑，n—沟道MOSFET（n—MOS）逻辑和互补MOSFET（CMOS）逻辑。尽管TTL和ECL对高速应用很重要，但由于其功耗大所产生的热量多，以及门的尺寸大，因而它们的集成度受到限制。相反，MOS逻辑电路系列是非常适合LSI和VLSI的，因为它们比双极型逻辑电路系列可获得

更高的集成度。现今大多数LSI和VLSI电路是用MOS实现的。然而MOS工艺应用于LSI、VLSI电路日益增多，已引起新的测试方面的问题。对于这点在本章后面的部分会有介绍。

一个电路的故障，就是一个或多个元件的物理缺陷。故障可分为逻辑故障和参数故障。所谓逻辑故障（logical fault）就是导致一个电路单元或输入信号的逻辑函数变成某些其他逻辑函数的故障；而参数故障（Parametrie fault）改变了电路参数值的大小，引起诸如电路速度、电流、电压等参数的变化。

与时间有关的电路工作出错，主要是由于电路的延迟造成的。这种涉及像低速门等电路延迟的故障称为延迟故障（delay fault）。一般来说，延迟故障只影响电路的定时操作，它可能引起险象或临界竞态。

在某些时间间隔内出现，而在另一些时间间隔内不出现的故障是间歇故障（intermittent fault）。那些总是存在的，不是在测试期间发生、消失或改变其特性的故障，称为永久性故障（Permanent fault）或实在故障（Solit fault）。尽管许多间歇故障最终会变成实在故障，但间歇故障的早期发现对电路的可靠工作是非常重要的。但是没有可靠的能检测它们存在的方法，因为当进行测试时，这种故障可能没有表现出来。在本书中，我们将主要研究实在的逻辑故障。



(a) 用双极型晶体管实现　　(b) 逻辑符号

图 1.10　反相器

图1.10中给出一个n—P—n晶体管实现的反相器。当其输入X是高电压时，输出Z是低电压；当X是低电压时，Z是高电压。该晶体管的集电极开路或基极开路将导致输出Z为恒定的高电压，即固定在逻辑1状态（S—a—1）。另一方面，集电极和发射极之间的短路，将导致Z为恒定低电压，即固定在逻辑0状态（S—a—0）。这些故障叫作固定故障（stuck—at fault）。

两根线被短路的故障称为桥接故障（bridging fault）。利用故障模型技术将确定桥接故障具有什么样的影响。通常，当带有高低不同电平的两根线发生桥接时，不是其中高电平就是其中低电平将起支配作用。如果是两根输出线被短路，而且是低电平起支配作用；则这种情况可用一个二线的与门来代替，如图1.11中所示，其效果与TTL门中常使用的线—与逻辑相同。（图1.12中示出在TTL门中使用的线—与逻辑）。如果是高电平起支配作用，则这两根线可用一个二线的或门来代替。ECL门就具有将彼此的输出简单地连在一起而实现输出"或"逻辑的特性。因此，在用ECL门实现的逻辑电路中，桥接故障将使假装的信号成为线"或"的结果。

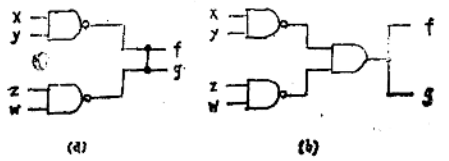


图　1.11　与门型的桥接故障

对于大多数实际问题，可以成功地用固定故障或桥接故障模仿逻辑故障。但是，并不是所有的故障都可以用这些类型的故障来模仿的。下面我们通过一个实例来说明这个问题。图1.13示出用P—MOS和n-MOS场效应晶体管构成的二输入CMOS与非门。如果两个输入X_1和X_2均为高电平，则其输出Z为低电平。在图1.13(b)中，标出了四种可能的开路故障（编号：①～④）。首先标号为1的故障，其故障原因是开路，或者说输入X_1与上推P沟道晶体管脱开。在这种故障情况下，当输入X_1为低电平，X_2

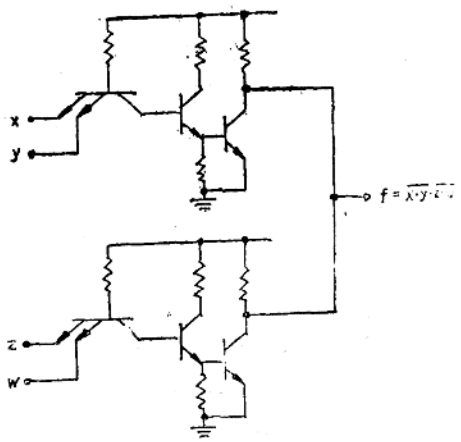


图 1.12 在门中使用的线—与逻辑

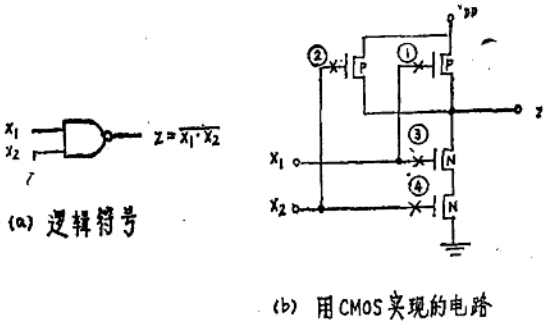


图 1.13 CMOS两输入与非门

为高电平时，输出Z变成不希望的高阻状态，而保持前面的输出逻辑值。然而在多长的一段时间内，保持该状态，由结点的漏电流来确定。(表1.1是这个两输入CMOS与非门在无故障情况和三种故障情况下的真值表。)其结果，这些开路故障使组合电路变成了时序电路。因此，它们不能用经典的故障(例如，固定故障)来模拟。

表 1.1 CMOS与非门的真值表

X_1	X_2	Z (正常电路)	Z (在①处开路)	Z (在②处开路)	Z (在③或④开路)
0	0	1	1	1	1
0	1	1	前面的状态	1	1
1	0	1	1	前面的状态	1
1	1	0	0	0	前面的状态

在可编程序逻辑阵列（PLA）中的交叉点故障，也不能用经典的故障来模拟。图1.14

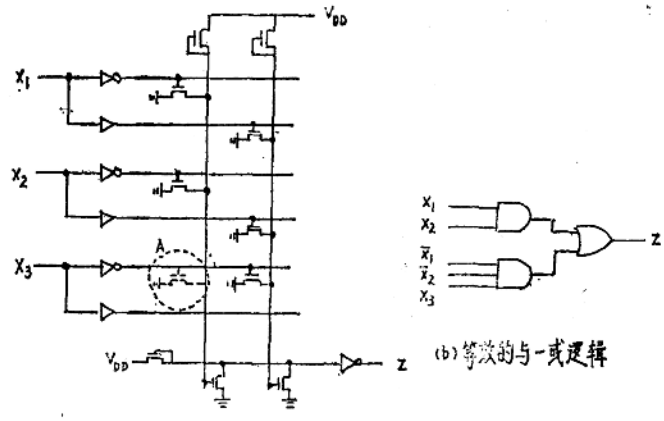


图 1.14 可编程序逻辑阵列

(a) 示出用MOS技术实现的PLA。通常，PLA实现两级与—或逻辑电路。图1.14 (b) 示出与图1.14(a)中PLA等效的两极与—或逻辑图。在PLA阵列的每个交叉结点上，都固有一个器件（二极管或三极管），即使不使用，它也将存在。通过对每个器件的连接编程，来获得所希望的逻辑功能。在一个PLA中，多了或缺少了器件，都会引起交叉点故障。虽然其中的大部分可用固定故障来模拟，但仍然有一些交叉点故障不能用固定故障来模拟。

假定图1.14 (a)中所示交叉点A处有一个额外的器件。当不存在此故障时，其输出实现的逻辑函数是：$Z = X_1X_2 + \overline{X}_1\overline{X}_2X_3$。

如果在A处有一个晶体管，则第一项积X_1X_2将变成$X_1X_2X_3$，因此，有故障的输出函数将是

$Z = X_1X_2X_3 + \overline{X}_1\overline{X}_2X_3$

然而，在图1.14 (b) 所示等效的与—或逻辑电路中的任何固定故障，都不可能产生这样的函数。

1.3 测试问题

为了保证系统的正常运行，当出现故障时，我们必须有能力发现它，并能将其定位或隔离到一个确定的，容易更换的元件。前面的过程叫做故障检测 (fault detection) 后面的

过程叫做故障定位（fault location），故障隔离（fault isolation），或故障诊断（fault Ldiagnosis）。这些任务通过测试来完成。测试是一个检测和定位或者定位故障的过程。测试可类分为故障检验测试和故障诊断测试。故障检验测试只辨别某个电路有故障还是无故障；如果有故障，它给不出有关故障识别的任何情况。故障诊断测试则提供故障位置、故障类型及其他的信息。测试所提供的信息量称为该测试的诊断分辨力（diagnosis resolution）；故障检验测试就是一种诊断分辨力为0的故障诊断测试。

对逻辑电路的测试，是施加一序列输入测试图形，当故障存在时，则产生错误的输出响应，然后将其响应与正确的（期待的）响应相比较。在测试中使用的这种输入图形叫做测试图形（test pattern）。通常，对一个逻辑电路的测试包括许多测试图形，把它们称之为一个测试集（test set）或一个测试序列（test sequence）。测试序列意味着一系列测试图形，如果测试图形必须按照一定的顺序施加，则使用这个术语。有时把测试图形与输出响应一起称做测试数据（test data）。

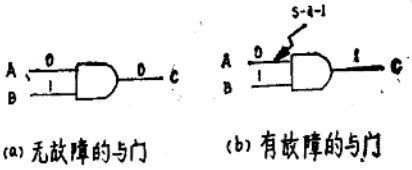


图 1.15 测试固定故障

图1.15（a)示出一个无故障的与门，图1.15(b）示出有故障的与门，其中输入端A固定1（s-a-1)。加给无故障与门的输入图形，产生0值输出，与此相反，有故障与门的输出值为1，这是由于在A处的固定1故障产生错误的响应。在这里，故障门和无故障门之间存在着明显的差别，因此，图1.15（a)所示的测试图形01是对输入A S-a-1故障的一个测试图形。

如果在一个电路中仅存在一个故障，则称为单故障（single fault）；如果同时存在两个或多个故障，该故障集合称为多故障（multiple fault）。对于一个具有k条线的电路，最多可能有2k个固定的单故障。对于多故障，可能的故障数显著地增加到3^K-1个，因为任何线可能是无故障、固定0故障或固定1故障。具有100根线的电路，大约可包含5×10^{47}个故障。故障数如此之多，以致根本无法设想，因而对多故障的测试是不实际的。然而，对大多数测试方法来说，已经证明单一固定故障模型提供了最好的实践基础。例如，已知在任何一个无内部扇出的组合电路中，每个完整的单故障检验测试集，对由六个或少于六个故障构成的多故障，至少可复盖其全部故障的98%。（Agarwal和Fugn 1981）。

由于几个不同的故障常常会引起一个电路出现完全相同的不正常工作的情况，将这些等效故障（luqivalent fault），或称为不可区分的故障（indistinguishable fault）组成

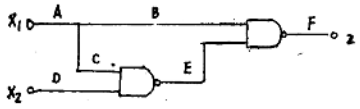


图 1.16

一个等效类型是适宜的。在图1.16所示电路中，有六条信号线（或称网线），因此，最多可有12个单一固定故障。表1.2列出这个电路的真值表，其中各列给出全部可能的单固定故障，各行给出全部输入图形。例如，线A的固定0故障表示为A/0。从表1.2中可以看出，由A/0、B/0、E/0、D/1和F/1等故障所引起的电路不正常工作的状态是完全相同的，因此，把它们分组在一个故障等效类型中。同样，故障C/0、D/0和E/1被分组在一起，如表1.3中所示。如果一个故障的出现没有引起电路不正常工作，这种故障叫作冗余（redundant）故障。换句话说，带有冗余故障的电路输出函数，与该电路无故障的输出函数是完全

表 1.2 全部单固定故障

X_1 X_2	Z	A/0	B/0	C/0	D/0	E/0	F/0	A/1	B/1	C/1	D/1	E/1	F/1
0 0	1	1	1	1	1	1	0	0	0	1	1	1	1
0 1	1	1	1	1	1	1	0	1	0	1	1	1	1
1 0	0	1	1	0	0	1	0	0	0	0	1	0	1
1 1	1	1	1	0	0	1	0	1	1	1	1	0	1

表 1.3 等效故障类型

f_0	Z（无故障），C/1
f_1	A/0, B/0, E/0, D/1, F/1
f_2	C/0, D/0, E/1
f_3	F/0
f_4	A/1
f_5	B/1

表 1.4 典型故障的故障表

X_1	X_2	f_1	f_2	f_3	f_4	f_5
0	0			X	X	X
0	1			X		X
1	0	X				
1	1		X	X		

相同的。在图1.16所示电路中，故障C/1是冗余故障，如表1.3所示。

在故障检测和定位中，由于任何两个等效故障不能从输入—输出的工作状态上来区分，因此，只需要考虑那些从每个等效故障类型中选出的典型故障（representative fault）。表1.4就是表1.3中的五个典型故障的故障表，它清楚地指出哪个测试图形将能检测出哪些故障。决定一个故障检验测试集，简单说来就是这样一件事：确定若干行的集合，其中每行表示一个测试图形，使表示一个故障类型的每一列可以在这些行的一行中有一个“X”。在此例中，我们得到的故障检验测试集是{00,10,11}。通过确定若干行的集合，使得对应这些行没有成对的列图形是相同的，而得到故障诊断测试集。从表1.4中我们得到{00, 01, 10, 11}作为诊断测试集。

逻辑电路的测试分两个主要阶段：生成被测电路的测试图形，即测试生成(generation阶段)和对电路施加测试图形，即测试应用（test application）阶段。测试图形的生成是很重的，然而对大型（如LSI和VLSI）电路，这也是非常困难的。所以，在这个领域中，过去二十年的大部分努力都投入研究和开发有效而经济的测试生成方法。

测试（测试图形集或测试图形序列）的质量主要依赖于故障复盖率以及测试的规模或时间长短。测试的故障复盖率(fault coverage，或称为测试复盖率test coverage)，是指在被测电路中，能检测出或被定位的故障百分比。对于一个给定的测试的故障复盖率，通过称之为故障模拟(fault simulation)的过程来确定。在故障模拟中，每个给定的测试图形被加到一个无故障电路和每个有给定故障的电路，模拟每个电路的行为特性，分析每个电路的响应，从而找出这个测试图形检测出的故障。故障模拟也用来产生故障字典(fault dictionaries)，在故障字典中收集了识别有故障的元件或组成部分所需要的信息。

概 要

测试生成的过程包括构造故障模型并简化，测试图形生成，故障模拟，故障复盖率估计，以及故障字典的产生。第一个步骤包括为被测电路开发一个故障字典（即构造出所假设的故障模型）和利用故障等效关系简化故障的数目。通常，采用“单固定故障”模型，故障字典依据逻辑电路描述，通过以列表的形式排列

每个门的可区分的故障，直接生成。下一步，为测试在故障字典中所列出的故障集生成测试图形。然后，对照故障字典中的故障电路模拟测试图形，并根据故障模拟结果（包括已检测出故障表和未检测出故障表）计算出故障复盖率。如果这个故障复盖率不符合要求，则对未测试到的故障重复进行测试图形生成和故障模拟过程，直到达到满意的结果。最后，逐项登记检测及定位故障所需的充分的信息以完成故障字典。如图1.17所示。

通常，上述测试生成过程以应用软件汇集的形式自动实现，既切实可行又经济，效果较好。特别是测试图形生成和故障模拟，对有效地以低的计算费用得出高的故障复盖率将相互

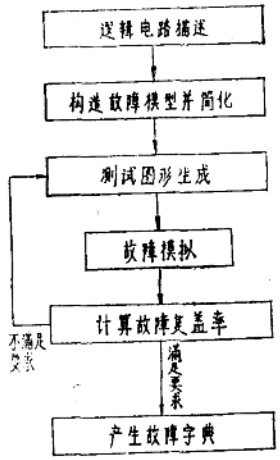


图 1.17 测试生成的过程

发生影响。LSI和VLSI电路的应用，要求更加有效的自动测试生成（automatic test generation）系统。

1.4 测试方案

逻辑测试分不同的阶段进行，包括芯片，电路板和系统在工厂的测试，以及在维护和修理期间电路板和系统的现场测试。一般来说，首先进行故障检测，如果已确定有故障存在，则采用故障诊断，隔离有故障的结点或元件以便修理。在LSI和VLSI芯片测试中，没有必要定位出故障的门，因为如果它的任何一个输出值是错误的话，整个芯片必定废掉。然而，在电路板和系统的测试中，需要进行诊断，而且在工厂测试和现场维修中，系统级的诊断是非常重要的。当一个故障已在系统级定位后，便更换元件或单独的模块，直到维修好为止。

按所采用的测试数据的生成和处理技术，测试方法可分为两类：连机测试和脱机测试。连机测试（on-line testing）在系统运行期间，与正常的计算机操作同时进行。由正常计算所产生的数据模式作为测试图形，某个故障造成的错误，可由内部设置的监测电路检测出。由于连机测试时，测试和正常计算可以同时进行，因此，这种测试也称为并发测试（concurrent testing），而且对间歇故障的检测很有效。脱机测试（off-line testing）是系统处于脱机状态时进行的测试。在脱机测试中，通常要提供预先生成的专用测试图形，系统正常计算产生的数据模式不能做为测试图形使用。

测试方法又可分为外部测试和内部测试。在外部测试（external testing）中，测试设备对被测系统来说是外部的，即通过外部的测试仪器对被测系统施加测试图形，并鉴别其响应。在内部测试（built-in testing）中，测试设备建立在被测系统内部。正如下面所叙述的那样，连机测试利用内部的监测电路来检测由某些间歇的或实在的故障所引起的错误，因而它属于内部测试；而脱机测试则包含外部测试和内部测试两种方案。

连机测试方案通常用冗余技术（redundancy techniques）实现，冗余技术包括信息冗余（information redundancy）和硬件冗余（hardware redundancy）两种类型。信息冗余方法包括诸如奇偶校验、循环冗余校验及错误校正编码等流行的编码方案；硬件冗余技术包括门级的自校验电路、模块级的双重模块，以及系统级的双工计算机。

最广泛使用的线性编码，对单位故障检测是奇校验和偶校验编码。对多位故障检测是海明编码。在这些错误检测编码中，为检测错误只简单地附加一位或几位冗余位。例如，考虑一个具有n个输入和m个输出的组合电路，如果在正常操作中只能出现$k<2^m$个输出值，则2^m-k个不允许出现的值中任何一个的出现，都将表示该电路内的故障引起的一个错误。那些应该出现的输出值叫作代码字(code words)，而那些不允许的输出值叫作非代码字(noncode words)。错误检测码一般用来设计能自动检测电路中错误的硬件。（如图1.18中所示）。

自检电路（self-checking circuits）是仅通过观察其输出，就可以判定该电路内出现故障的一种电路。自检电路的一个重要类型是全自检电路（totally self-checking circuits），其定义如下：如果一个电路对一个故障集F是故障安全的，则对于F中的任何故障及任何允许的输入编码，其输出是非代码字或正确的代码字，而决不会是不正确的代码字。换句话说，故障电路的输出不能是一个代码字，同时，它与电路的正确输出是不同的。因此，只要输出是一个代码字，就可安全地假定

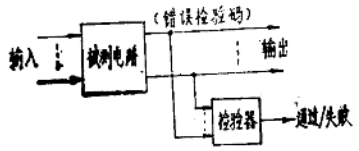


图 1.18 采用信息冗余的连机测试

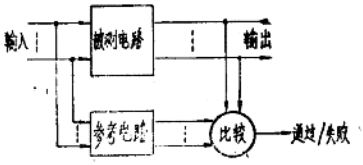


图 1.19 采用硬件冗余的连机测试

它是正确的。另一方面，如果一个电路对一个故障集F是自测试（self—testing）的，则对F中的任何故障f，都存在一个可检测F的允许的代码输入（也就是得到的输出是一个非代码字），换句话说，该输入集对指定的故障集中的每一个故障至少包含一个测试。全自检电路对所考虑的所有故障既是故障安全的，又是自测试的。因此，全自检电路能保证该电路中的任何故障不会引起输出错而无需故障检测。

对采用硬件冗余的连机测试的最有效的方法之一，是双份冗余（dual redundancy），就是把被测试的资源简单地双份设置。图1.19示出内部测试的这种双份设置方案。虽然完全复置方案是硬件冗余用于故障检测的最高限度，因而比其他的硬件冗余方法需要更多的硬件，但是这种方案能够容易地在计算机系统的任何部分，以及在计算机层次结构内的任何层级上被采用。

在外部测试方法中，被测电路(circuit under test—CUT)是用自动测试设备(automatic test equipment—ATE)进行测试。自动测试时一个测试图形序列加给被测电路，并将被测电路的响应与参考值（正确的响应）相比较。图1.20示出通用的测试方案，其测试模式和参考值用软件为主的方法或者用硬件为主的

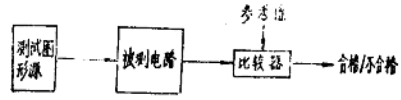


图 1.120 通用的测试方案。

方法产生。在前者情况下，测试图形和正确结果预先生成并贮存在存贮器中。测试图形可由设计师及测试工程师人工地生成，也可用测试生成程序自动地生成。而后者情况却与此相反，每次测试被测电路时，测试图形和参考值由硬件实现的算法产生。在这种以硬件为基础的方案中，测试图形通常是随机地或伪随机地产生，并同时加给被测电路和称之为标准样品电路（gold circuit）的已知合格电路，并将被测电路与标准样品电路的输出响应加以比较。这种测试方法（如图1.21所示）叫做随

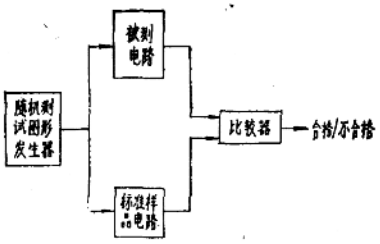


图 1.21 随机性测试

机性测试（random testing）；与此相对照，前面的以软件为基础的测试方法，叫做确定性测试（deterministic testing）。确定性测试

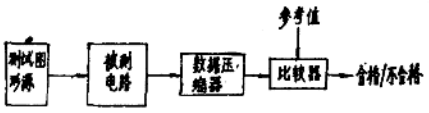


图 1.22

需要昂贵的或费时的测试生成及存贮巨大数量的测试数据的存贮器，而随机性测试避免了生成或存贮测试数据所需的费用和时间。但是随机性测试也有它的缺点。需要标准样品电路可能有些麻烦，标准样品电路的可靠性没有保证；进行测试时，被测电路与样品电路之间的同步以及随机测试图形的故障复盖率也可能出现问题。

在上述几种测试方案中，都要将所有的输出响应与参考值进行比较，通常，这个数据量是相当大的。分析和存贮大量响应数据的困难，采用一种称之为压缩测试（compact testing）的方法可以避免。在这种测试中，不是比较所有的响应数据，而是将压缩了的响应数据加以比较，图1.22中示出压缩测试的通用方案。这个方案模型允许测试图形生成及数据压缩的方法有很大的差异。总的说来压缩测试法归类为确定性测试或随机性测试，与所用的生成测试图形的技术有关系。数据压缩器，可用诸如计数器，线性反馈移位寄存器之类简单的电路来实现。由于压缩测试需要很少的测试设备，因此，它很适用于内部测试。压缩测试广泛使用的一方法是特征码分析（signature-

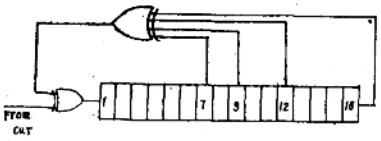


图 1.23 一个十六位线性反馈移位寄存器

analysis），它通过一个十六位的线性反馈移位寄存器来压缩输出响应，该寄存器的内容叫作“特征”（signature）。图1.23是一个十六位线性反馈移位寄存器的示意图。

第二章 测试生成

在这一章内，我们讨论测试生成问题，并考查几种生成测试的计算算法。前面四节，只限于讨论组合电路的测试生成；在最后一节，研究时序电路的测试生成问题。

多年来，已提出了许多种测试生成的算法，其中大多数是理论意义大于其实际意义，因而，并没有在实际中使用，只有少数几种算法得到了实际应用。应用最广的算法是D—算法（Roth1966）。在某种意义上讲，这是一种完整的算法，只要存在对任何逻辑故障的测试，D—算法将可以生成这种测试。但是必须指出，对于具有许多异或门的电路的测试生成，D—算法的效果却很差。为了改善D—算法的这一缺陷，提出了另一种测试生成算法——面向通路判定（Path Oriented DEcision Making—PODEM）算法，称为PODEM算法。（Goel 1981 a）。最近又提出一种面向扇出的测试生成算法——FAN算法（Fujiwara和Shi-

mono 1983 a, b)，这种测试生成算法比D—算法和PODEM 算法更加有效。下面将分别详细叙述这三种方法。

一般说来，对时序电路的测试生成比组合电路的测试生成更加困难，这主要是由于时序电路缺乏可控制性和可观测性。然而，时序电路的可控制性和可观测性，亦即它们的可测性，可以通过某些通称为“可测性设计”的技术，提高到组合电路的水平，这些技术使时序电路的测试可简化为类同组合电路的测试。因此，如果假设被测的时序电路总是采用这些设计技术实现的话，我们仅研究对组合电路行之有效的测试生成算法就足够了。基于这个理由，本章着重研究组合电路的测试生成算法，对时序电路的测试生成算法，只进行简略地讨论。

2.1 布尔差分

组合逻辑电路的每一个输出，实现一个逻辑（布尔）函数。令F（x_1, x_2……, x_n）是布尔输入变量x_1, x_2, ……, x_n的逻辑函数。F（X）对于输入x_i的布尔差分(Booleau difference) 定义为：

$F(X_1\cdots\cdots, X_i\cdots\cdots, X_n)\oplus(X_1, \cdots\cdots \overline{X_i}, \cdots\cdots, X_n)$

以$\frac{dF(X)}{dxi}$来表示。

它也可以表示为：

$$\frac{dF(X)}{dxi}=F_i(0)\oplus F_i(1)\ ,$$

其中，$F_i(0)=F(x_1, \cdots\cdots, X_{i-1}, 0, X_{i+1}, \cdots\cdots X_n)$,

$F_i(1)=F(X_1, \cdots\cdots, X_{i-1}, 1, X_{i+1}, \cdots\cdots, X_n)$。

通常，一个逻辑函数是由某些利用诸如求补、求积、求和等逻辑操作的分函数组成。因此，利用分函数的布尔差分来表示这些操作的布尔差分的公式，在布尔差分计算中是很有用的。根据布尔差分的定义，可以推导出下列公式：

$$\frac{d\overline{F(X)}}{dx_i}=\frac{dF(X)}{dx_i} \qquad (1)$$

$$\frac{dF(X)}{dx_i}=\frac{dF(X)}{d\overline{x_i}} \qquad (2)$$

$$\frac{d}{dx_i}\left(\frac{dF(X)}{dx_i}\right)=\frac{d}{dx_i}\left(\frac{dF(X)}{dx_i}\right) \qquad (3)$$

$$\frac{d[F(X)G(X)]}{dx_i}=F(X)\frac{dG(X)}{dx_i}\oplus G(G)\frac{dF(X)}{dx_i}\oplus\frac{dF(X)}{dx_i}\frac{dG(X)}{dx_i} \qquad (4)$$

$$\frac{d[F(X)+G(X)]}{dx_i}=\overline{F(X)}\frac{dG(X)}{dx_i}\oplus\overline{G(X)}\frac{dF(X)}{dx_i}\oplus\frac{dF(X)}{dx_i}\frac{dG(X)}{dx_i} \qquad (5)$$

$$\frac{d[F(X)\oplus G(X)]}{dx_i}=\frac{dF(X)}{dx_i}\oplus\frac{dG(X)}{dx_i} \qquad (6)$$

例2.1　考虑图2.1中的电路，其输出函数是$F(X)=x_1x_2+\overline{x_2}x_3$。根据布尔差分定义可得到：

$$\frac{dF}{dx_1}=F_1(0)\oplus F_1(1)$$

$$=\overline{x_2}x_3\oplus(x_2+\overline{x_2}x_3)$$

$$=\overline{x_2}x_3\,\overline{(x_2+\overline{x_2}x_3)}+\overline{x_2}x_3\,\overline{(x_2+\overline{x_2}x_3)} \quad \text{（注）}$$

$$=x_2$$

利用布尔代数：

$$\frac{dF}{dx_1}=\frac{d(G_1+G_2)}{dx_1}\qquad [G_1=x_1x_2,\ G_2=\overline{x_2}x_3]$$

$$=\overline{G_1}\frac{dG_2}{dx_1}\oplus\overline{G_2}\frac{dG_1}{dx_1}\oplus\frac{dG_1}{dx_1}\frac{dG_2}{dx_1}$$

[根据公式5]

注：原文为：$\overline{x_2x_3}(x_2+\overline{x_2}x_3)\oplus\overline{x_2}x_3\,\overline{(x_2+\overline{x_2}x_3)}$ 其中的半加符号“⊕”应为“+”。

$$=\overline{G_2}\frac{dG_1}{dx_1} \qquad [因为\frac{dG_2}{dx_1}=0]$$

$$=(x_2+\overline{x_3})\quad(x_1\frac{dx_2}{dx_1}\oplus x_2\frac{dx_1}{dx_1}\oplus \frac{dx_1}{dx_1}\frac{dx_2}{dx_1}) \qquad (根据公式4)$$

$$=(x_2+\overline{x_3})\,x_2 \qquad [因为\frac{dx_1}{dx_1}=1,\ \frac{dx_2}{dx_1}=0]$$

$$=x_2。$$

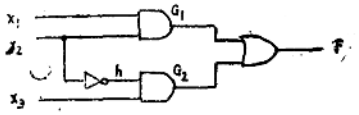


图 2.1

假设F（$x_1\cdots\cdots x_n$）是组合电路的输出函数，令输入x_i为s-a-0的故障是α。该故障电路实现的函数为：

$$F\alpha(x_1x_2\cdots\cdots x_n)=F(x_1\cdots\cdots x_{i-1},\ 0,\ x_{i+1},\cdots\cdots x_n)$$

$$=F_i(0)$$

检测故障α的测试图形是满足$F(X)\oplus F\alpha(X)=1$的输入组合或输入向量X。利用某些运算，得到：

$$F(X)\oplus F\alpha(X)=(\overline{x_i}F_i(0)\oplus x_iF_i(1))\oplus F_i(0)$$

$$=\overline{x_i}F_i(0)\oplus x_iF_i(1)\oplus(\overline{x_i}+x_i)F_i(0)$$

$$=\overline{x_i}F_i(0)\oplus x_i\overline{F_i}(0)\oplus x_iF_i(1)\oplus x_iF_i(0)$$

$$=x_i(F_i(1)\oplus F_i(1))$$

$$=x_i\frac{dF(X)}{dx_i}$$

因此，检测x_i的s-a-o故障的全部测试集是

$$\left\{X;\ x_i\frac{dF(X)}{dx_i}\right\}$$

用布尔表达式$x_i\dfrac{dF}{dx_i}$定义。

这个表达式的意思是$x_i=1$，且$dF/dx_i=1$。由于$x_i=1$，则x_i在故障输入端施加相反的值，因数dF/dx_i保证这个错误信号影响F的值。同样，检测x_i的s-a-1故障的全部测试集，用布尔表达式$\overline{x_i}\dfrac{dF}{dx_i}$定义。

例2.2　考虑图2.1中的电路。其输出函数为$F(X)=x_1x_2+\overline{x_2}x_3$检测$x_1$的s-a-o故障的测试集计算如下：

$$\frac{dF}{dx_1}=F_1(0)\oplus F_1(1)$$

$$=\overline{x_2}x_3\oplus(x_2+\overline{x_2}x_3)$$

$$=x_2。$$

检测这个故障的测试集用布尔表达式

$$x_1\frac{dF(X)}{dx_1}=x_1x_2\text{ 确定。}$$

这个表达式意味着x_1的s-a-o故障引起一个错误的信号x_1；而且，仅当$x_1x_2=1$时，即$x_1=1$且$x_2=1$时，这个错误信号影响输出值F。

通常，故障不仅能够出现在外部或原输入端，而且也会出现在电路内部的信号线。布尔差分也可用来推导内部信号线上的固定

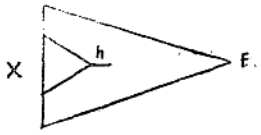


图 2.2　内部信号线

故障的测试。对于图2.2中的电路，令F（X）是电路的输出函数，h是内部信号线。我们可以将h表示为X的函数h（X）；还可以把h视为一个输入，将输入函数F表示为X和h的函数。令这个函数是F^*，即

$$F^*(Xh)=F(X)$$

为了检测h的s-a-o故障，必须传播相反的值1到h，并将正常信号值和错误信号值的差别传播到电路的输出。这样，检测h的s-a-o故障的全部测试集用布尔表达式

$$h(X)\frac{dF^*(X,h)}{dh}\text{ 确定。}$$

类似地可以得到，检测h的s-a-1故障的全部测试集用布尔表达式

$$\overline{h(X)}\frac{dF^*(X,h)}{dh}$$ 确定。

例2.3 考虑图2.1中的电路。假设门G_2的输入h有s-a-1故障，电路输出函数 $F(X)=x_1x_2+\overline{x_2}x_3$，表示为$x_1$，$x_2$，$x_3$和h的函数为：

$$F^*(X,h)=x_1x_2+hx_3$$

其中$h(X)=\overline{x_2}$。从而得到

$$\frac{dF^*(X,h)}{dh}=x_1x_2\oplus(x_1x_2+x_3)$$

$$=(\overline{x_1}+\overline{x_2})x_3。$$

检测h s-a-1 故障的全部测试集，由下面的布尔表达式

$$\overline{h(X)}\frac{dF^*(X,h)}{dh}=x_2(\overline{x_1}+\overline{x_2})x_3$$

$$=\overline{x_1}x_2x_3$$ 确定.

因此，当且仅当$x_1=0$，$x_2=x_3=1$时，可检测出

h s-a-1故障。

例2.4 考虑图2.3中的电路。其输出函数为$F(X)=\overline{x_1}+x_1x_2=\overline{x_1}+x_2$，计算检测h s—a—1故障的全部测试集。此处h是门G的输入，如图 2.3 中所示。F 可以表示为X和 h 的函数 F^*：

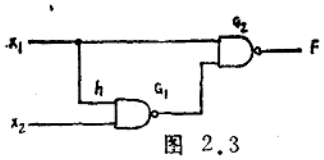


图 2.3

$F^*(X,h)=\overline{x_1}+hx_2$其中，$h(X)=x_1$则

$$\frac{dF^*(X,h)}{dh}=\overline{x_1}\oplus(\overline{x_1}+x_2)$$

$$=x_1x_2$$

检测h s—a—1故障的整个测试集，由下列布尔表达式确定：

$$\overline{h(X)}\frac{dF^*(X,h)}{dh}=\overline{x_1}\cdot x_1x_2=0$$

这就是说，对h s-a-1故障不存在测试图形，即这个故障是冗余故障。

在逻辑电路输入的两个故障对其输出的影响，可以通过定义双布尔差分

为$\frac{dF(X)}{d(x_ix_j)}=F(x_1\cdots,x_i,\cdots x_j\cdots x_n)\oplus F(x_1,\cdots\overline{x_i},\cdots\overline{x_j},\cdots x_n)$ 来分析。

因此，对多固定故障的测试生成，可采用多布尔差分的方法（Ku和 Messon1975）来概述。

利用布尔差分的测试生成方法是由sellers等人提出的（1968a）。布尔差分法具有代数学的特点，通过巧妙地处理电路方程以生成测试。还有一些其它的代数学的测试生成方法，其中包括 poage 的命题法（propositional method）（1963）；Armstrong的等效范式（equivalent normal form）（1966）；Bossen和Hong的因果方程（cause-effect equation-（1971）；clegg的SPOOF方法；以及 Kinoshita 等人提出的结构描述函数（structure description function）（1980）。

这些方法都建立无障碍电路的方程，并巧妙处理这些方程以生成测试。然而，一般说，巧妙地处理代数方程是件困难的任务。对于大型电路，要推导出一给定故障的测试，可能需要进行大量的代数运算。此外，这些对给定故障生成全部测试的代数运算方法有一个缺点，就是它们需要大量的存贮空间和费很长的时间，因此，用于大型电路可能是不现实的。

2.2 D算法

有一些利用拓扑的门级电路描述来取代布尔方程运算的测试生成方法，这些方法都涉及到利用回溯技巧的通路敏化和信号传播。

一维通路敏化（one-dimensional path sensitization）或单通路敏化（single-path sensitization）是最简单的测试生成方法之一。其基本思想是从故障位置，比如说门 G0，选择一条通路，顺序通过门G_1，…G_n而达到电路的一个输出。首先，指定G0的输入，以便在故障位置产生一个与故障值相反的值（对s-a-1

是0，对s-a-0是1）；然后，为了将故障信号沿着G_1，…G_n这条通路传播到电路的原输出端而确定通路上门G_1，…G_n的输入。上述技术例示A图2.4中。对于门G_i的输入端，除了在这条通路上的输入，对与门（AND）/与非门（NAND）其余全部设置成1；对或门（OR）

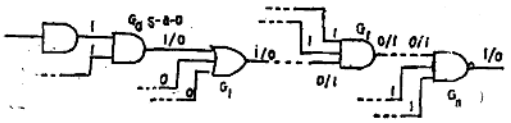


图 2.4 单通路敏化

/或非门（NOR）全部设置成0。在图2.4中，符号0/1表示一个信号在正常的或好的电路中值为0，在故障电路中值为1；符号1/0则表示一个信号在正常电路中值为1，在故障电路中值为0。这个过程称为这种方法的正向跟踪（forward trace）阶段或错误传播（error propagatioe）阶段，这条通路叫作已敏化了的通路（Sensitized path）。最后，我们必须求出实现所有门的必需输入值的原输入图形，这是通过从G_0，…G_n的输入返回跟踪到电路的原输入端而求得。这个过程叫作这个方法的回溯（backwarcl-trace）阶段，或线合理性证明（line-justification）阶段。

例2.5 对于图2.5中的电路，为检测x_1 s-a-0故障，x_1必须为1，而且错误信号必须沿着某一通路传播到电路的输出。我们可以选择通过G_1，G_5和G_7的通路，或选择G_1，G_4，G_6和G_7的通路，将错误信号传播到输出Z。要传过G_1，x_2必须为1；要传过G_5和G_7，要求G_3的输出为1，以及G_6的输出为0。这就意味着G_2的输出必须是0，也就是要求$x_3=0$或$x_4=0$。因此，我们有两种测试：$x_1x_2\bar{x}_3$和$x_1x_2\bar{x}_4$。利用这两种测试，沿着$x_1G_1G_5G_7Z$通路可传播正常的与故障信号之间的差别，因而这条通路是敏化了的通路。

单通路敏化法有一个致命的缺点。通常，电路中可能存在一些可测试的故障，如果一次仅允许敏化一条通路，则可能对这些故障无法

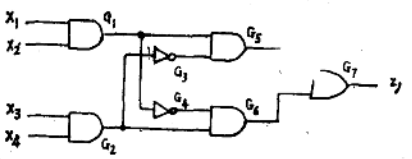


图 2.5

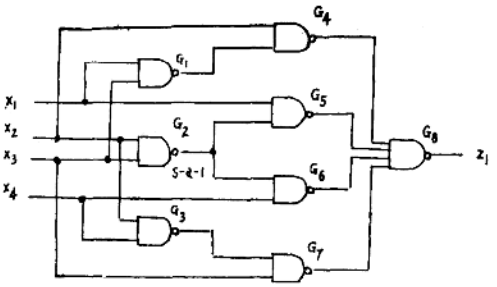


图 2.6

生成测试。Schneider（1967）曾给出一个简单的例子。该电路如图2.6所示，成为问题的故障是G_2的输出s-a-1。为了在G_2的输出产生0值，要求输入$x_2=x_3=1$。从G_2到Z_1的任何单一通路都必须经过G_5或G_6。如果只选择通过G_5和G_8的单一通路加以敏化，则要求$x_1=1$以及$G_4=G_6=G_7=1$。而$G_6=1$要求$x_4=0$。但是$x_2=x_3=1$和$x_4=0$意味着$G_7=0$，这是一个矛盾。由于该电路对称，因此沿另一条G_6G_8通路敏化也必定不成功。这个矛盾似乎意味着这个故障是不可检测的，然而输入模式$x_1=x_2=x_3=x_4=1$能测试上述成问题的故障。图2.7示出了对这种输入模式的正常电路及故障电路的工作状态。在这个图中我们要注意，该故障的影

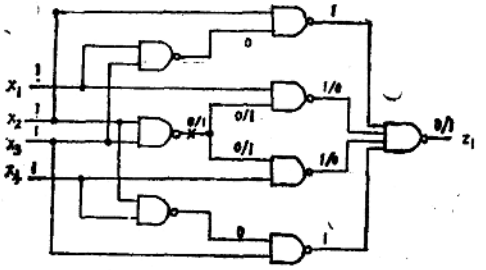


图 2.7 多通路敏化

响已同时沿着两条通路传播。单通路敏化法之