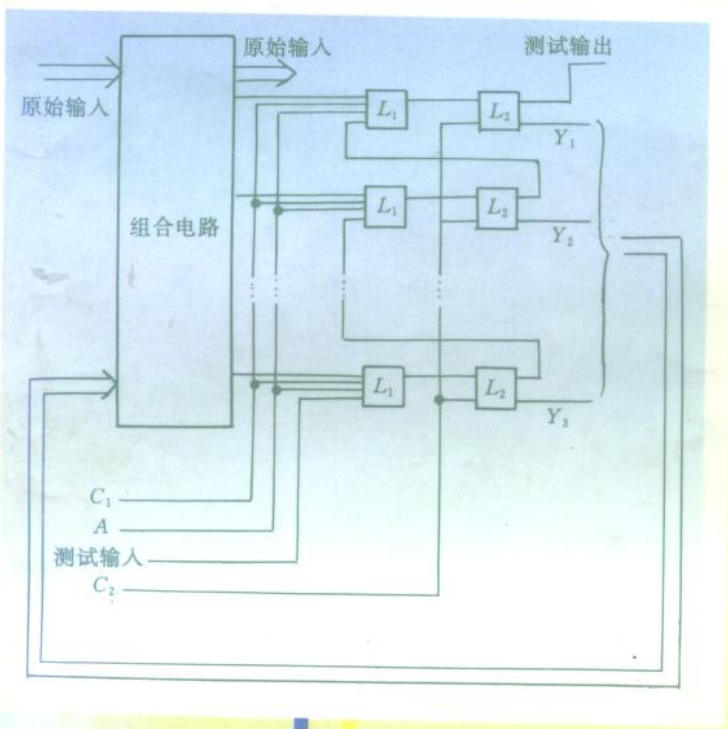
数字系统测试及可测试性设计

向东 编著

科学出版社



数字系统测试及可测试性设计

数字系统测试及可测试性设计

向 东 编著

科学出版社
1997

内 容 简 介

本书系统地介绍了数字系统测试及可测试性设计.该书是在作者近年的研究成果及国际上关注的最新研究热点的基础上撰写的.全书共分八章,内容包括:测试生成、可测试性设计、自测试、并行测试生成及并行测试的方法和算法.书中除了介绍一些深受人们喜爱的经典算法外,还给出了作者提出的20余个算法,这些算法大部分在工作站上用C语言实现,并被实践证明行之有效.

本书可作为高等学校计算机、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.

图书在版编目(CIP)数据

数字系统测试及可测试性设计/向东编著.—北京:科学出版社,1997

ISBN 7-03-005617-5

Ⅰ.数… Ⅱ.向… Ⅲ.①数字系统-测试②数字系统-测试-设计 Ⅳ.TP211

中国版本图书馆CIP数据核字(97)第02611号

科学出版社出版

北京东黄城根北街16号

邮政编码:100717

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1997年6月第 一 版 开本:787×1092 1/16

1997年6月第一次印刷 印张:19

印数:1—2 600 字数:437 000

定价:28.30元

前　言

计算机日益普遍的推广使用使人们对可靠性的要求越来越高.数字系统的可靠性很大程度上取决于测试的质量.利用测试手段可确定一个系统的生产与工作是否正确.高速发展的集成电路技术使得测试开销成为逻辑设计、生产代价的主要部分.虽然 *D*-算法、PODEM 算法、FAN 算法、SOCRATES 算法、TRAN 算法及递归学习算法在不同程度上将测试生成研究推向了一个新的阶段,但是这些方法只成功地解决了组合电路的测试生成问题.对于高度集成化的电路(特别是含时序逻辑较多的电路),采用现有的测试生成算法仍然不能达到令人满意的结果.通过采用一些技术来提高测试码生成效率已不可能从根本上解决问题.解决该问题的主要方法之一是进行可测试性设计.可测试设计是通过在电路中加入可测试逻辑来提高测试码生成效率.提高测试生成效率的另一有效途径是采用并行测试生成.并行测试生成是采用并行处理机来提高测试生成效率,每一个处理机只处理一个测试生成子任务.

全书共分八章.第一章介绍集成电路测试的现状与发展趋势.第二章介绍组合电路与时序电路的测试生成算法.组合电路的测试生成算法主要介绍了 PODEM 算法、FAN 算法、SOCRATES 算法、EST 算法及 TRAN 算法.时序电路的测试生成算法主要介绍了重复逻辑阵列模型、基于逻辑模拟的 CONTEST 算法及基于状态分解的 SEST 算法.第三章介绍随机测试与分析.第四章系统介绍可测试性分析.分别从非概率模型的可测度、概率模型的可测度、功能级的可测度及动态可测度来介绍可测试性分析.第五章介绍非结构化的可测试性设计,分别介绍测试点插入、自治测试及相应减少测试代价的可测试性设计方法.第六章介绍结构化的可测试性设计,即扫描设计.第七章主要介绍并行测试生成与并行故障模拟方法.并行测试生成主要介绍故障并行、搜索并行及故障并行的最优粒度;并行故障模拟主要介绍故障划分与电路划分.第八章主要介绍含 BIST 结构的测试调度和基于电路划分的并行测试.

目前国内出版的测试书籍大多内容已经陈旧,或者是只停留于传统方法的介绍,没有反映国际上的最新研究动向.与国内外同类书籍相比,本书有下列特点:

1. 内容新.本书是国内第一本系统介绍可测试性设计、可测试分析、并行测试及并行测试生成的书.书中除了介绍一些已被广泛接受的经典方法以外,还介绍了目前一些最新的研究专题.

2. 深入浅出.对所有涉及到的专题都是从不同的角度以浅显的方式进行细致的探讨.

3. 力求实用.书中提供了 40 余个算法,其中有些是深受人们喜爱的经典算法,有些是由作者提出并已在工作站上实现的行之有效的算法.

逻辑测试的发展趋势是:可测试性设计必须从测试生成的实际复杂性出发.可测试性综合是在电路设计完成以前将可测试性设计因素考虑进去以指导设计.这类方法的优点是:不需要在电路设计完成以后加入硬件逻辑来改进电路的可测试性.由于测试生成算法

已不能大幅度降低测试生成代价，所以采用并行处理机来产生测试码可有效地提高测试生成效率. 问题是如何将大规模的并行计算机系统有效地映射到测试码产生问题. 非固定型故障的测试生成与可测试性设计是测试研究的又一趋势. 非固定型故障可以覆盖许多固定型故障所不能检测到的缺陷. 这类故障有：转换故障、路径延迟故障及桥接故障等.

本书在撰写过程中受到了日本奈良科学技术大学测试界权威学者 Hideo Fujiwara 教授的热情帮助. Fujiwara 教授全面审查了本书的写作提纲，并提出了大量宝贵的意见. 在本书的写作过程中得到了中国科学院和中国工程院院士陈俊亮、北京工业大学梁业伟教授、北京航空航天大学金惠华教授、北京理工大学刘明业教授及中国科学院计算技术研究所沈理研究员等许多有益的帮助，国家自然科学基金委对本书的部分研究内容提供了资助，在此一并致谢.

作者谨以此书献给所有从事集成电路 CAD 及 CAT 研究的同行，特别是那些正在顽强学习与工作的学生. 由于时间仓促，书中难免有疏漏之处，敬请批评指正.

作　者

1996 年 11 月

目　录

第一章 引 论

随着电路的高度集成化，测试代价变得越来越难以接受. 测试代价主要分为两个方面：

1）测试生成代价；

2）测试码置入代价.

提高测试生成代价的主要方法有：

1）提出有效的测试生成算法；

2）并行测试生成与故障模拟；

3）可测试性设计.

目前最有影响且最有效的组合电路测试生成算法有 PODEM 算法，FAN 算法，SOCRATES 算法，EST 算法，基于传递闭包的 TRANS 算法以及 Kunz 等人采用的递归学习算法. 近年时序测试生成较有影响的算法有 BACK，ESSENTIAL，HITEC，FASTEST，GENTEST 及 CONTEST 等. 并行测试生成是采用并行计算机系统来处理测试生成问题. 最主要的并行测试生成策略有：模拟并行、启发式知识并行、搜索并行及故障并行. 可测试性设计主要分为特殊的方法与结构设计. 结构设计主要是指扫描设计. 扫描设计主要分为完全扫描与部分扫描. 降低测试码置入代价的方法主要有并行测试及可测试性设计.

本章将主要介绍逻辑电路测试及可测试性设计的主要研究结果，近年的研究热点及以后的研究趋势.

1.1 测试的重要性

我们希望尽可能早地发现产品的故障，因为要检测相同的故障在不同层次所需代价不同，层次越高代价越高. 通常认为要检测相同的故障在门级、芯片级(chip level)、板级(board level)、系统级(system level)及域级(field level)测试代价依次以 10 倍增长，而且随着电路输入管脚数及时钟频率的增加成指数增长. 通常认为测试生成时间是产品设计周期内最长的阶段，测试时间大约占据了整个产品设计与生产总时间的 40%. 产品投入市场的时间延后半年会导致产品利润降低 33%.

我们下面将介绍一种电路测试代价的评价模型. 设 C_e 为工程代价，包括产生测试码的代价等. 我们有

$$C_e = \frac{\text{周数} \times \text{每周的代价}}{\text{ASIC 单元数}} \tag{1.1}$$

若花费 16 周产生测试，每周所需代价为 4000 元，且总的 ASIC 单元数为 5000，C_e 可计算为

$$C_e = \frac{16 \times 4000}{5000} = 12.80 \text{ 元 /ASIC 单元} \tag{1.2}$$

若采用自动测试生成及可测试性设计只花费 3 周的时间来产生测试码，则

$$C_e = \frac{3 \times 4000}{5000} = 2.40 \text{ 元 /ASIC 单元} \tag{1.3}$$

设 C_t 为投入市场时间(time-to-market)的代价，产品封装时间延迟半年(27 周)则会使利润降低 33%，C_t 可由下式求出：

$$C_t = \frac{\text{系统价格} \times \text{利润极限} \times \text{延迟} \times 0.33}{27 \times K} \tag{1.4}$$

其中 K 为系统中 ASIC 单元的数目. 例如：系统的价格为 5000 元，由于系统中两个 ASIC 元件的测试产生，使得投入市场时间延迟了 12 周，因而导致了 15%的利润差值. 每一个 ASIC 单元的有效测试生成代价为

$$C_t = 5000 \times 0.15 \times 12 \times 0.33 = 55 \text{ 元} \tag{1.5}$$

假设由于可测试性设计使 ASIC 的面积开销由 A_0 增加为 A_1，则面积开销 C_a 为

$$C_a = C_0 \cdot \left[\frac{A_1}{A_0} Y_0^{1-\sqrt{\frac{A_1}{A_0}}} - 1 \right] \tag{1.6}$$

其中 Y_0 为晶片合格率(yield)，并且 C_0 为初始的面积开销代价.

例 1.1 假设一个有在线测试电路的 ASIC 价值 50 元. 当在电路中加入可测试性逻辑以后面积增加 10%，并假设晶片合格率为 $Y_0 = 50\%$. ASIC 的代价由 50 元增加为 53.50 元.

当系统性能由 S_0 降低为 S_1 时，系统价格由 price0 降为 price1，

$$\text{price1} = \text{price0}\left(\frac{S_1}{S_0}\right)^n \tag{1.7}$$

其中 n 为性能价格比的指数($1.2 \leqslant n \leqslant 2.0$).

$$C_s = \text{price0}\left(1 - \left(\frac{S_1}{S_0}\right)^n\right) = \frac{\text{ASIC 的价格}}{\text{已售产品的价格}} \tag{1.8}$$

例 1.2 若 ASIC 代价为 50 元，$n = 1.5$，已售产品的代价为 40%. 当在电路中插入可测试逻辑以后，系统性能降低 5%，则 ASIC 的代价增加 9 元.

综上所述，整个 ASIC 的测试代价 C_{total} 为

$$C_{\text{total}} = C_e + C_t + C_a + C_s$$

其中 C_e 为工程代价；C_t 为投入市场时间的代价；C_a 为面积开销的代价；C_s 为性能损耗代价.

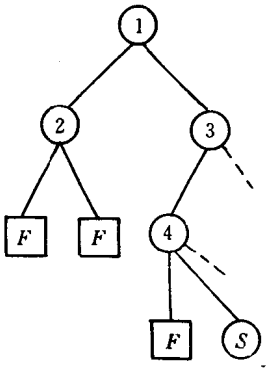


图 1.1 判定树

1.2 测试码生成的基本概念

1.2.1 判定树

几乎所有的测试生成算法都是依据判定树来求解的. 图 1.1 示出一个判定树结构.

图中每个节点代表该算法要解决的问题，每个节点有多个分枝；每个分枝代表该节点对应问题的一种选择. 图中的每一个方节点表示一个失败的终结点. 每一个圆节点

表示一致的蕴含，含 S 的圆节点表示找到了一个测试码.

1.2.2 隐含枚举

一个完全的算法总能找到一个可测故障的测试码，因为这类算法总是隐含地枚举完所有可能的取值. 这类算法不同于穷举的方法，需要穷举所有可能的取值. 使用隐含枚举可将搜索引向满足所有取值限制的向量. 在测试生成过程中，随着这些取值限制的增加，能够满足这些限制的向量越来越少.

1.2.3 回溯

前面我们已经提到，求解一个故障的测试码是通过搜索一个判定树来完成的. 要确认一种取值或者说传播故障信息，可能有多种取值选择. 我们往往会遇到一种冲突的信号取值，这样必须改变相应的不正确取值，并保留该取值以前的状态.

1.2.4 D-前沿

D-前沿由所有这样的门组成：门的输出为任取值，该门至少有一个输入为 D 或者 $\overline{D}$. 在测试生成过程中，当 D-前沿变空时，必须回溯.

1.2.5 双向蕴含

在测试生成过程中，FAN 算法进行向前和向后蕴含，尽可能多地确定可以唯一确定的信号线. 如图 1.2(a)所示，要在 L 处产生故障 $L/1$ 的故障信息，必须将 L 置为 0，则要求将 J，K，E 置为 1. FAN 算法通过向前和向后蕴含确定尽可能多的唯一确定的信号线. 当要将 J 定为 1 值，必须将 A，B 都置为 1；要将 K 置为 1，此时只需将 C 置为 0. 这样不需要回溯就可以在 L 处产生一个故障信息 D. 不采用唯一蕴含时，可能首先产生一个信号目标 $K=1$，要将 K 置为 1 可能将 B 置为 0；然而只有将 B 置为 1 时，才能将 J 置为 1. 这样在 B 处产生冲突的信号要求，则必须回溯.

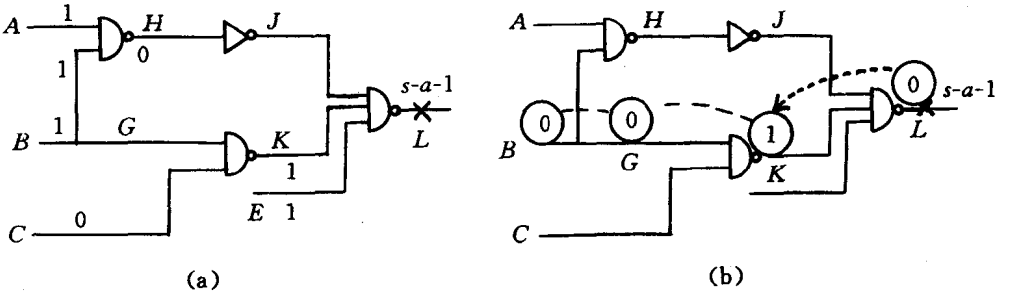


图 1.2 双向蕴含

1.2.6 唯一敏化

唯一敏化主要考虑以下情形：1)唯一的 D-前沿，2)由 D-前沿到原始输出的所有路径都必须经过的门. 我们称信号线 y 静态地控制 x，即 $y\in \mathrm{dom}(x)$，仅当所有由 x 到原始输

出的有向路径必须经过 y. $\mathrm{dom}(x)$即是所有由 x 到原始输出的路径都必须经过的信号线集合.

在得到以上定义后,我们可以得到以下规则:

(1) 设 x 为当前唯一的 D-前沿,并且 $\mathrm{dom}(x)=\{y_1,y_2,\cdots,y_n\}$为这个门对应的输出. 这些门为 $G=\{g_1,g_2,\cdots,g_n\}$. 对于所有的门 $g\in G$,将 g 所有由 x 不可达的输入赋以非控制值.

如图 1.3 所示,只有唯一的 D-前沿 a. a 到原始输出的所有路径都经过 g,由 a 不存在到 d 的路径,则要将 a 的故障信息传播至 g,d 必须置为 1.

(2) 设 x 为当前唯一的 D-前沿,x 扇出到门 $g_1,g_2,\cdots,g_n$,并且这些门的非控制输入均为 $v\in\{0,1\}$,则将所有 $y\neq x$ 置为 v 值,y 同时扇入到 $g_1,g_2,\cdots,g_n$.

如图 1.3 所示,目前只有一个 D-前沿 h,则要将 h 点的故障效应传播到输出必须将 j 置为 1. 图中信号线 i 只是 k 的输入,则不能将 i 置为 1. 一条有向路径 $p=(x_1,x_2,\cdots,x_n)$,$x_1\neq x_n$,称为潜在的传播路径,仅当 x_1 的取值为 D 或 $\overline{D}$,而 $x_2,\cdots,x_n$ 为未确定的取值.

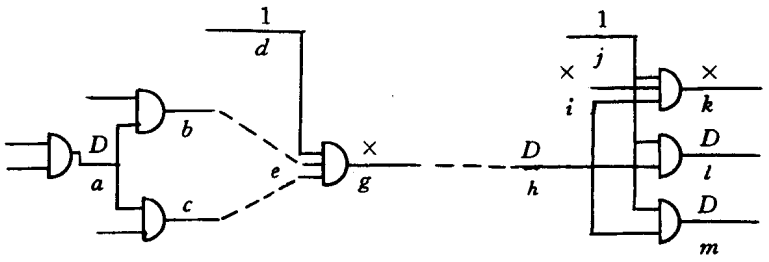


图 1.3 唯一敏化

信号线 g 称为信号线 x 动态的控制线,即 $g\in\mathrm{dyn-dom}(x)$,仅当所有由 x 到原始输出的潜在传播路径都经过 y.

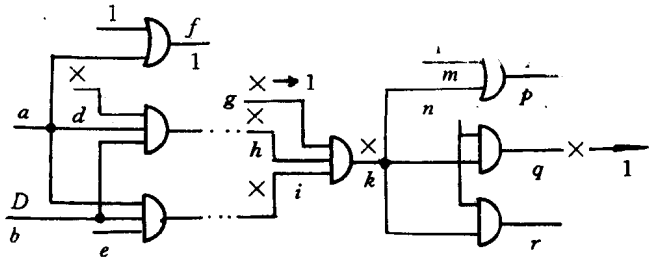


图 1.4 动态的唯一敏化

如图 1.4 所示,$(a,\cdots,h,k,q)$,$(a,\cdots,i,k,q)$为潜在的传播路径;而$(a,f,\cdots)$和$(a,\cdots,h,k,p)$则不是,$\mathrm{dom}(a)=\varnothing$,$\mathrm{dyn-dom}(a)=\{k\}$. 类似于上一节,我们可以得到以下规则:

(1) 若 D-前沿有信号线 $x_1,\cdots,x_n$. 设 $y=\{y_1,\cdots,y_m\}$为 $x_1,x_2,\cdots,x_n$ 的共有的动态控制信号线,即

$$Y=\mathrm{dyn-dom}(x_1)\cap\cdots\cap\mathrm{dyn-dom}(x_n) \tag{1.9}$$

Y 为门 $G=\{g_1,\cdots,g_m\}$ 的输出集. 对所有 $g\in G$,g 的所有 D-前沿的存在潜在传播路径的输入赋以 g 门的非控制值,这些输入由故障点不可达. 如图 1.4 所示,k 不为 $\mathrm{dom}(a)$ 的元素,但由于 $af\cdots$ 不为潜在传播路径,所以 $k\in \mathrm{dyn}-\mathrm{dom}(a)$. g 置为输出为 k 门的非控制值.

(2) 若 $Y=\mathrm{dyn}-\mathrm{dom}(x_1)\cap\cdots\cap\mathrm{dyn}-\mathrm{dom}(x_n)$,$x_1,\cdots,x_n$ 为 D-前沿,$y\in Y$. 若 y 扇出到 $g_1,g_2,\cdots,g_k$ 门,其中由任意 $g_i\in\{g_1,g_2,\cdots,g_k\}$ 至少存在一条潜在的传播路径,并且 $g_1,g_2,\cdots,g_k$ 门的非控制输入值均为 $v\in\{0,1\}$,则对所有 $z\neq y$,z 扇出到所有 $g_1,g_2,\cdots,g_k$ 门,则将 $z\leftarrow v$. 如图 1.4 所示,由于 p,m 已赋值 1,则 $kp\cdots$ 不为潜在的传播路径. 要将 k 的故障信息传播原始输出,n 必须被置为 1.

1.2.7 全局蕴含

Schulz 等人提出的 SOCRATES 算法采用了以下全局蕴含:如图 1.5(a)所示,若 F 要求取 1 值,无论采用哪一种取值方式都要求 B 取 1 值,则可以得出这样的规则:$(F=1)\Rightarrow(B=1)$;相反,我们可以得到 $(B\neq1)\Rightarrow(F\neq1)$. 这些规则都是在测试生成的预处理过程中得到的,我们称为静态学习过程. 在测试生成过程中,电路中的某些线取确定值时,也存在相应的全局蕴含. 如图 1.5(b)所示,已知 A 点已取确定值 1 时,将 F 反向蕴含,可得 $B=0$,则此时可得规则:$(F=0)\Rightarrow(B=0)$. 类似可得 $(B\neq0)\Rightarrow(F\neq0)$.

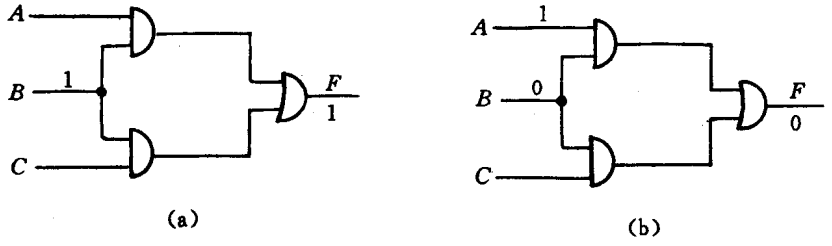


图 1.5 全局蕴含

1.2.8 搜索状态

在测试码生成过程中,当部分原始输入取确定值时,将这些值向前蕴含,得到一个新的电路分解(decomposition). 此时,可将那些与电路取×值相邻,且取确定值的线看成是电路的一个割集. 每一种原始输入的部分取值对应一个唯一的割集,并对应于搜索树中一个唯一的节点. 任一这类节点称为一个搜索状态.

1.2.9 *E*-前沿

类似于前面定义的 D-前沿,对应于原始输入的一种部分赋值,满足以下条件的线 L 为 E-前沿的元素:L 取确定值,并且由 L 存在一条到原始输出的×-路径. 这里的×-路径是指该路径上的任一线都为任取值. 如图 1.6(a)所示电路及其原始输入赋值,E-前沿 $E_1=\{(5,D),(7,1)\}$;图 1.6(b)所示电路的 E-前沿 $E_2=\{(5,D)\}$. 设 E-前沿 E_1,E_2 分别对应于搜索树中的节点 N_1,N_2. 若 $E_1=E_2$,则称 N_1,N_2 对应的搜索状态等价;若 $E_1\subseteq E_2$,且

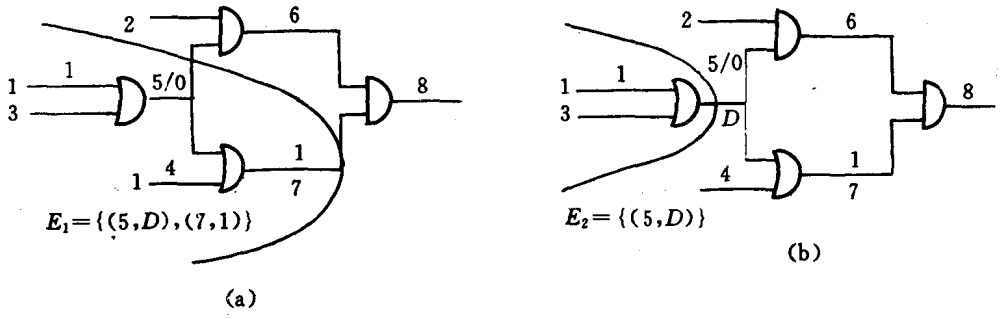


图 1.6 搜索状态

E_2 中的 D-前沿信号 $D_2 \subseteq E_1$，则称 E_1 控制 E_2. 如图 1.6(a)，1.6(b)中所示 E-前沿分别为 E_1 与 E_2，并有 E_2 控制 E_1.

1.3 故障模型

1.3.1 固定型故障

故障模型就是将物理缺陷的影响模型化为逻辑函数的逻辑及时延等方面的特性. 目前用得最多的故障模型是单固定型故障，即是任何时候电路中只有一条信号线固定为 0(或 1)值. 无论电路输入取什么值时该线取值不变. 一个故障将电路变成另一新电路. 设对于输出 Z_l 而言，输入向量 t，无故障和故障的函数分别为 $Z_l(t)$ 和 $Z_l'(t)$，$Z_l(t) \neq Z_l'(t)$ 当且仅当 t 为该故障的测试.

设有两故障 f,g,Z 为电路输出的函数. 一个测试 t 可区分 f,g 仅当 $Z_f(t) \neq Z_g(t)$，则 t,g 称为是可区分的. 若 $Z_f(t) \neq Z_g(t)$，则 f 与 g 称为等价故障，其中 t 为任一输入向量. 等价故障是不可区分的.

如图 1.7(a)所示，$a/0,b/0,c/1$ 是等价故障，图 1.7(b)中 $x/1,y/1,z/0$ 则是等价故障. 采用等价故障可降低故障集合的规模. 图 1.7(a)中故障 $a/1$ 的测试为 01；01 也为故障 $c/0$ 的测试. 对于两故障 f,g，若 f 的任意测试均可检测 g，则称 g 控制故障 f. 图 1.7(a)中，$c/0$ 控制 $a/1,b/1$；图 1.7(b)中，$z/1$ 控制 $x/0,y/0$. 生成测试码的过程中，只需找到 f 的测试就能检测 g. 故障精简将所有的等价故障集只用一个故障来表示，并且若 g 支配 f，则只保留 f.

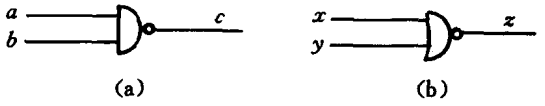


图 1.7 单固定型故障

图 1.8 所示电路为一个 2-输入的多路转换器，采用故障精简以后故障集为 $\{A/1,B/1,C/0,E/1,F/1,G/0,H/0,I/0\}$. 必须注意的是：扇出源与扇出分枝处的故障是不等价的. 如测试 $(A,B,C)=(1,0,0)$ 可检测 $B/1$，而不能检测 $E/1$.

一个组合电路的校验点定义为原始输入与扇出分枝的集合. 在非冗余的电路中，检测

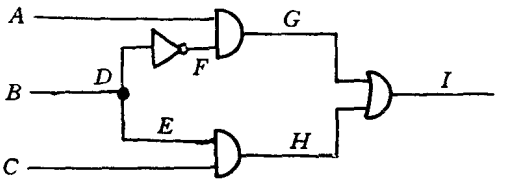


图 1.8 故障精简

所有校验点处单固定型故障的测试集，必然检测电路中所有的单故障. 图 1.8 所示电路的校验点为$\{A,B,C,D,E\}$.

当电路中同时有多个信号线固定为相应逻辑值的故障模型称为多固定型故障. 覆盖电路中所有单固定型故障的测试集必然可覆盖电路中较大比例的多故障.

故障检测分故障激活与故障效应传播两个过程. 对于故障A/i而言，故障激活就是在A处产生一个值$\bar{i}$. 故障激活以后，将所得的故障效应(D或$\bar{D}$)传播到原始输出的过程称为故障效应传播($i\in\{0,1\}$). 图 1.9(a),(b)分别描述激活与传播故障$G/1$.

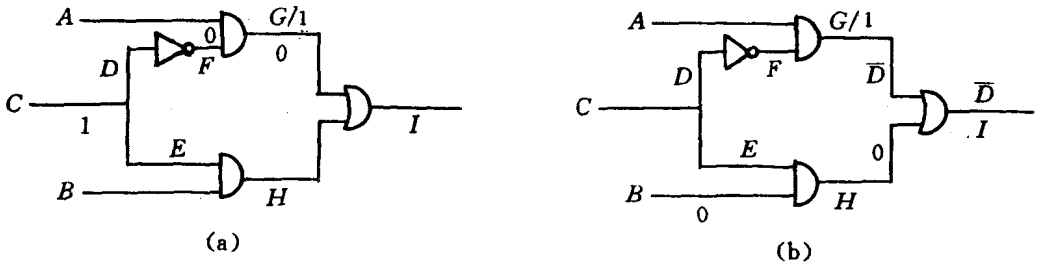


图 1.9 冗余故障识别

常常存在下述情况：电路中某一故障是不可激活的或者是故障效应不能传播到原始输出或者是两者都不能实现，则该故障为冗余故障. 在出现冗余故障时，一些本来可检测的故障可能变得不可检测. 测试生成很大一部分时间都是花在冗余故障的测试产生上. 一个有效的测试生成算法常常在测试产生以前识别冗余故障.

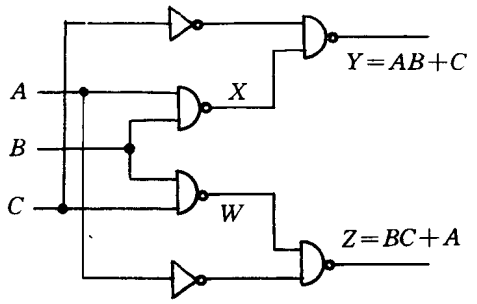
1.3.2 其余故障模型

除了固定型故障以外，近年来用得最多的故障模型还有I_{ddq}故障、延迟故障及转换故障(transition fault). 以下将简单地介绍这些故障模型.

I_{ddq}测试适合于静态 CMOS 电路. 通常情况下静止电流值I_{dd}很低，当出现物理缺陷时，电流值很高. 电流测量是在所有的开关瞬变都平衡以后完成. I_{ddq}测试测量电路许多状态的电流，当检测到某一电流值很高时，则不必往下处理.

图 1.10(a)中X,W产生一个“与”桥接故障. 一旦在X,W点处测量到一个高电流状态，则检测到该故障. I_{ddq}测试还可以处理门氧化短路故障等. I_{ddq}测试只需要故障激活，不必在原始输出观察故障信息.

I_{ddq}测试的主要优点有：可覆盖大部分的桥接故障和一些开路故障. 测试产生很容易. 缺点是电路必须设计为具有较低的I_{ddq}，测试置入速度较慢，并且电流阈值必须凭经验来确定.



(a)

A	B	C	X	W	Y	Z
0	1	1	1/0	0	1	1
1	1	0	0	1/0	1	1

(b)

图 1.10 I_{ddq}测试

另外一种故障模型为路径延迟故障. 当一路径传播延迟超过规定的最坏延迟时间,则称为产生一个路径延迟故障. 该故障模型可考虑分散的延迟,并且可覆盖转换故障及门延迟故障.

电路的一条路径 $P=\{g_0,e_0,\cdots,g_n,e_n,g_{n+1}\}$,其中 g_0 是原始输入,g_{n+1}是原始输出,g_i $(1\leq i\leq n)$是门,e_i 将 g_i 的输出连接到 g_{i+1}的输入. 当路径上存在一个延迟故障时,g_0 输入一个跳变,g_{n+1}应输出一跳变,但跳变传播时间超过给定的时间界限 T_c. 可以将一条路径的延迟认为是该路径上所有门延迟的总和. 由于采用单固定型故障不能覆盖很多物理缺陷,路径延迟故障较好地解决了该问题. 一个电路的路径数随门数的增长成指数增长. 如 C880 有 60 个输入,26 个输出,383 个门,其路径总数为 8642 条;C3540 有 50 个输入,140 个输出,1669 个门,路径总数为 28676671. 可见,当电路规模较大时,要检测电路中所有路径延迟故障几乎是不可能的. 因此对电路结构作转换以减少电路路径数目是一个很受重视的研究方向. 已有的标准电路中,C6288 的延迟故障可测试性最差. 该电路含大约 1.98×10^{19}个路径. 最新的研究结果表明,要得到 C6288 电路 100%的测试效率,需要占用 Sparc 2 工作站 50 余天的 CPU 时间.

转换故障分为 0→1 转换(slow-to-rise)和 1→0 的转换(slow-to-fall). 一个两向量序列(v_1,v_2)为信号线 k 的 0→1 转换故障的测试仅当 v_1 将 k 置为 0,且 v_2 可检测 $k/0$(k:s-a-0)故障. 类似地,一个 2-向量序列(v_1,v_2)称为 k 的 1→0 转换故障的测试,仅当 v_1 将 k 置为 1,且 v_2 可检测 $k/1$ 故障.

该故障模型是一种理想的抽象故障模型,可以与延迟的上升和下降时间联系起来. 该故障模型可覆盖 CMOS 电路的断开(stuck-open)故障及逻辑门的延迟故障,只需作少量修改就可与固定型故障模型相通. 在测试过程中,可以实现时序电路的快速测试码置入. 这种故障模型的缺陷是不能覆盖所有的延迟故障.

1.4 测试生成

1.4.1 组合电路测试生成

D-算法是最早的完全测试生成算法. 该算法用逻辑值$\{0,1,D,\overline{D},\times\}$来描述故障电路. 故障效应可沿着多个路径传播. D-算法在故障效应驱赶及信号确认过程中几乎不采