2002 年 1 月
第 4 卷第 1 期

中国工程科学
Engineering Science

Jan. 2002
Vol.4 No.1

学术论文

数字电路可测性设计的一种故障定位方法

潘中良
（华南师范大学物理系，广州 510631）

[摘要] 在逻辑函数 Reed-Muller 模式的电路可测性设计方面，文章采用 AND 门阵列和 XOR 门树结构来设计电路，提出了一种设计方案，可实现任意逻辑函数的功能，而且所得电路具有通用测试集和完全可故障定位的特点。给出了进行故障定位的方法，并可把它应用于其他相关电路的可测性设计。

[关键词] 逻辑函数；数字电路；可测性设计；故障定位；单故障
[中图分类号] TN 79+1 [文献标识码] A [文章编号] 1009－1742（2002）01－0069－06

1 引言

电路系统在设计、制造以及运行过程中需要进行测试，以保证其设计要求或正常工作，它已成为集成电路设计与生产过程的一个重要组成部分。电路存在故障的测试通常称故障检测，若还要定位故障点，则称为故障定位或故障诊断[1]。故障定位的困难在于电路的多个节点故障是不可分的，或者虽可区分但测试矢量复杂[2,3]。为此，研究在进行组合电路的可测性设计时能使电路可测且故障可定位[4,5]。

在逻辑函数的电路可测性设计方面，Reddy 最先提出基于 Reed-Muller 模式的异或门级联的实现方法[1,2]。对有 n 个变量的逻辑函数，所设计的电路只需 $n+4$ 个测试矢量组成的测试集就可检测电路中所有的固定型故障。而且这种测试集是通用的，这点使得通过逻辑函数的 Reed-Muller 模式来进行电路的可测性设计，在理论上非常具有吸引力。例如，就逻辑函数 Reed-Muller 模式的不同类型，PPRM，FPRM 和 GRM 等研究了针对单故障和多故障的可测性设计[6]。目前对逻辑函数的可测性设计研究主要涉及故障的检测，而对故障的定位方面研究较少。为此笔者开展了有关故障定位的研究，其主要创新之处在于：提出了一种基于逻辑函数 Reed-Muller 模式的电路可测性设计方案，给出了进行故障定位的方法与详细实现步骤，同时该方法可应用于其他相关的可测性设计中。

2 电路设计

2.1 逻辑函数的表示形式

任何一个具有 n 个变量的组合逻辑函数 f 均可用如下的 Reed-Muller 模式表示：

$$f = A_0 \oplus A_1 x_1^* \oplus A_2 x_2^* \oplus \cdots \oplus A_n x_n^* \oplus A_{n+1} x_1^* x_2^* \oplus A_{n+2} x_1^* x_3^* \oplus \cdots \oplus A_t x_{n-1}^* x_n^* \oplus \cdots \oplus A_m x_1^* x_2^* \cdots x_n^*, \quad (1)$$

其中⊕表示异或运算，$x_i^* \in \{x_i, \bar{x}_i\}$，$i = 1,2,\cdots,n$，$A_j \in \{0,1\}$，$j = 0,1,2,\cdots,m$，$m = 2^n - 1$。

若在函数 f 的 Reed-Muller 表达式中，所有变量 x_i（$i = 1, 2, \cdots, n$）以原变量（即 x_i 的形式）出现，则称为 PPRM 形式；若在表达式中一些变量以原变量 x_i 出现，而另一些变量以求反变量 $\bar{x}_i$ 出现，则称为 FPRM 形式；若在表达式中有

[收稿日期] 2001－06－21；修回日期 2001－09－28
[基金项目] 国家自然科学基金资助项目（60006002）
[作者简介] 潘中良（1966－），男，重庆万县人，博士，华南师范大学物理系副研究员

一些原变量 x_i 和它的求反变量 $\bar{x}_i$ 同时出现，则称为函数的 GRM 形式，如 $f = x_1x_5 \oplus \bar{x}_1x_4x_6 \oplus x_2x_3$。GRM 比 FPRM 更具一般性，对同一逻辑函数用这三种形式中的 GRM 模式所需的项数（指在表达式中的积项）是最少的[6~8]，下面主要是针对这种模式，其结果也适用于 PPRM 和 FPRM。

2.2 电路结构

对有 n 个变量 x_i^*（$i=1, 2, \cdots, n$）的逻辑函数 f，设它的 GRM 中积项有 m 项。对每一积项在 f 中的位置做如下调整：按变量 $x_1, x_2, \cdots, x_n, \bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n$ 出现在积项中的次序，把变量个数较少的积项排在函数 f 表达式的左边。例如，对 $f=\bar{x}_3\oplus x_1\oplus\bar{x}_1\bar{x}_2\oplus x_1x_3x_4\oplus x_1x_2x_4$，调整后的形式为 $f = x_1 \oplus \bar{x}_3 \oplus \bar{x}_1\bar{x}_2 \oplus x_1x_2x_4 \oplus x_1x_3x_4$。设 f 调整后的积项为 $f_1(x), f_2(x), \cdots, f_m(x)$，即 $f = f_1(x) \oplus f_2(x) \oplus \cdots \oplus f_m(x)$，且当 $j < m$ 时积项 $f_j(x)$ 的变量数少于或等于积项 $f_{j+1}(x)$ 的变量数。用 AND_j 来实现 $f_j(x)$（$j=1,2, \cdots, m$），并把原始输入线 $x_1, x_2, \cdots, x_n$ 连接到相应的 AND 门。用多个异或门（XOR 门）把 x_i 的值求反，将 $\bar{x}_i$ 连到相应的 AND 门。在这些 AND 门中使 AND_1 位于最左边，AND_2 次之，AND_m 位于最右边，即 AND 门的排列是有序的。把 AND 门的输出接到具有 m 个输入的 XOR 门树（简称 XOR 树），XOR 树的输出即是逻辑函数 f 的输出。

GRM 电路的一般结构如图 1 所示。为了电路测试的需要，在其基础上增加了 4 个控制输入 C_0，C_1，C_2 和 C_3，以及 2 个与门 AND_{11} 和 AND_{12}，和 2 个或门 OR_{11} 和 OR_{12}，这 4 个门的输出节点分别为 O_1、O_2、O_3 和 O_4。AND_{11} 和 OR_{11} 的输入都是 C_1，C_2，C_3 和所有原始输入 x_i（$i=1, 2, \cdots, n$），AND_{12} 和 OR_{12} 的输入都是所有原始输入求反 $\bar{x}_i$。在图 1 和图 2 中这 4 个门的输入用粗线表示。当电路正常运行时把 C_0，C_1，C_2 和 C_3 的值赋 1。图 2 给出了实现逻辑函数 $h = x_1\oplus x_2x_3\oplus\bar{x}_2x_3x_4$ 的电路。

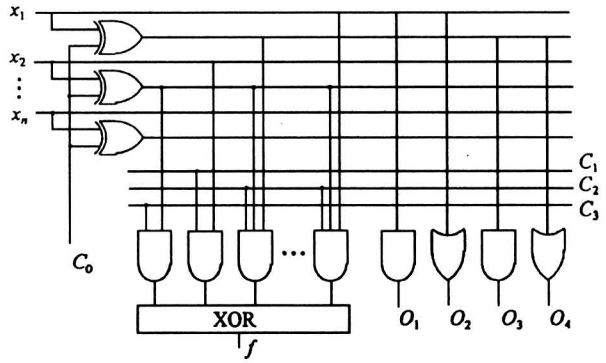

图 1 GRM 的电路可测性设计

Fig.1 The testable realization of GRM

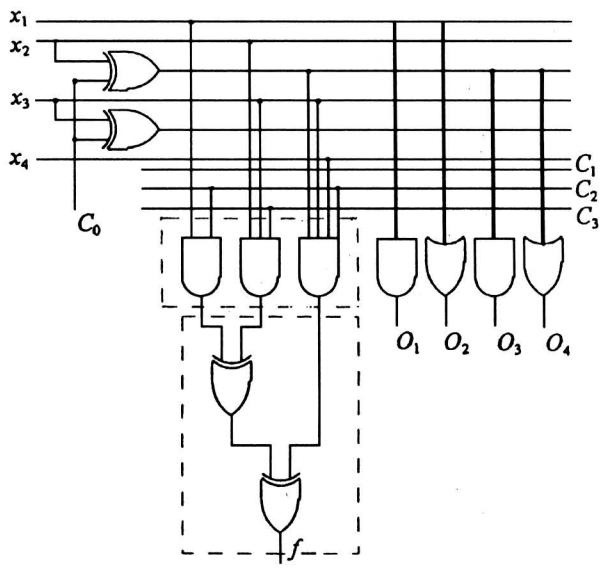

图 2 逻辑函数 h 的电路实现

Fig.2 The circuit realization of function h

控制输入 C_1，C_2 和 C_3 的连接方法如下：定义矢量 $\boldsymbol{V}_1$ = (0 0 1)，$\boldsymbol{V}_2$ = (0 1 0)，$\boldsymbol{V}_3$ = (0 1 1)，并为 XOR 树的原始输出节点赋这三个矢量中的一个，把其余的两个矢量分别赋给该 XOR 门的两个输入节点，以此类推，直至 XOR 树的每一个原始输入节点都赋了 $\boldsymbol{V}_1$，$\boldsymbol{V}_2$，$\boldsymbol{V}_3$ 中的一个矢量，参见图 3a。图 3a 和图 3b 分别是 8 个输入的 XOR 树和 XOR 门级联的例子。对 AND 阵列的每一个 AND 门，若它的输出节点（即相应 XOR 树的输入节点）被分配的矢量为 $\boldsymbol{V}_i$（$i=1, 2, 3$），则把控制输入 C_i（$i=1, 2, 3$）连至该 AND 门作为门的输入节点。

图 3 可见，变量数相同时，XOR 树电路比 XOR 门级联有更小的延迟。下面在讨论 GRM 电路时，把它分为：输入与控制、AND 阵列、XOR 树等三部分，在图 2 中已用虚线标出。具有 n 个输入变量的逻辑函数 f，若只考虑电路中节点的固定型故障（s－a－0 或 s－a－1，通常电路中的一

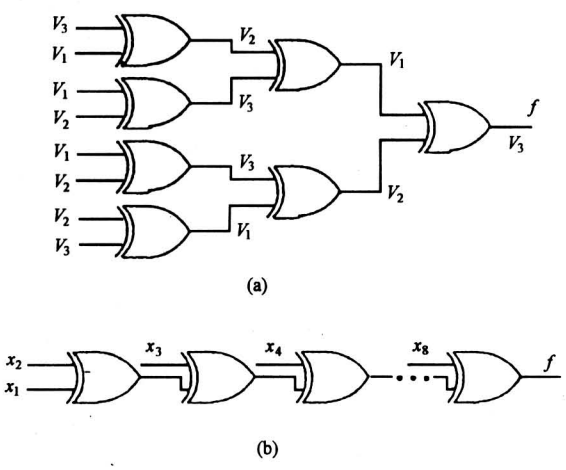

图 3　XOR 门树与 XOR 门级联

Fig.3　XOR gates tree and XOR gates cascade

些短路故障属于 s－a－0，开路故障属于 s－a－1），可以得到检测 GRM 电路的通用测试集。定义 $n+4$ 维矢量的各分量形式为 $\boldsymbol{R}=(C_0\ C_1\ C_2\ C_3\ x_1\ x_2\cdots x_n)$；$\boldsymbol{a}_1=(0\ 0\ 0\ 0\ 0\ 0\cdots 0)$，$\boldsymbol{a}_2=(0\ 1\ 1\ 1\ 1\ 1\cdots 1)$，$\boldsymbol{a}_3=(1\ 0\ 0\ 0\ 0\ 0\cdots 0)$；矢量 $\boldsymbol{b}_i(i=1,2,\cdots,n)$ 的分量为 $C_0=0$，$C_1=C_2=C_3=1$，$x_i=0$，其他所有分量 $x_j=1(j\neq i)$；$\boldsymbol{d}_1=(0\ 0\ 1\ 1\ 1\ 1\cdots 1)$，$\boldsymbol{d}_2=(0\ 1\ 0\ 1\ 1\ 1\cdots 1)$。其中 $\boldsymbol{T}_1=\{\boldsymbol{a}_1,\ \boldsymbol{a}_2,\ \boldsymbol{a}_3\}$ 可以检测 GRM 电路输入与控制部分的故障；$\boldsymbol{T}_2=\{\boldsymbol{a}_2,\ \boldsymbol{b}_1,\ \boldsymbol{b}_2,\ \cdots,\ \boldsymbol{b}_n\}$ 能检测 AND 阵列的故障；$\boldsymbol{T}_3=\{\boldsymbol{a}_1,\ \boldsymbol{d}_1,\ \boldsymbol{d}_2\}$ 可以检测 XOR 树的故障。用如上 $n+5$ 个矢量组成的测试集是电路的通用测试集 $\boldsymbol{T}=\boldsymbol{T}_1\cup\boldsymbol{T}_2\cup\boldsymbol{T}_3$。

3　GRM 电路的故障定位

若 GRM 电路发生了一个故障（s－a－0 或 s－a－1），可依次对电路的三部分即输入与控制、XOR 树、AND 阵列进行故障定位。

3.1　输入与控制部分的故障定位

首先对节点 C_0 的故障做定位。令矢量 $\boldsymbol{S}_1=(1\ 0\ 0\ 0\cdots 0)$，$\boldsymbol{S}_2=(0\ 0\ 0\ 0\cdots 0)$，$\boldsymbol{S}_1$ 和 $\boldsymbol{S}_2$ 中各分量的含义与矢量 $\boldsymbol{R}$ 相同。对电路的输入节点施加矢量 $\boldsymbol{S}_1$，若节点 $O_4=0$，则节点 C_0 发生了 s－a－0 故障；其他的故障情况，即 C_0 发生 s－a－1 故障、C_1，C_2，C_3 和原始输入节点 x_i 及其求反 $\bar{x}_i$ 中的一个节点发生 s－a－0 或 s－a－1 故障时都会使 $O_4=1$。对电路的输入节点施加矢量 $\boldsymbol{S}_2$，若节点 $O_3=1$，则节点 C_0 发生了 s－a－1 故障；其他故障情况下即 C_0 发生 s－a－0 故障、C_1，C_2，C_3 和原始输入节点 x_i 及其求反 $\bar{x}_i$ 中的一个节点发生 s－a－0 或 s－a－1 故障时都会使 $O_3=0$。

使控制输入 $C_0=0$，此时电路的原始输入 x_i 与其求反 $\bar{x}_i$ 的值相同。对电路的输入节点施加 $n+3$ 个矢量（分量的含义同矢量 $\boldsymbol{R}$）：$\boldsymbol{r}_{c1}$，$\boldsymbol{r}_{c2}$，$\boldsymbol{r}_{c3}$，$\boldsymbol{r}_1$，$\boldsymbol{r}_2$，…，$\boldsymbol{r}_n$。每个矢量中只有一个分量为 1，其他分量全为 0。其中 $\boldsymbol{r}_{c1}$ 的分量 C_1 为 1，$\boldsymbol{r}_{c2}$ 的分量 C_2 为 1，$\boldsymbol{r}_{c3}$ 的分量 $\boldsymbol{C}_3$ 为 1，$\boldsymbol{r}_i$（$i=1,2,\cdots,n$）的分量 x_i 为 1。把节点 O_2 作为观测点，用矢量 $\boldsymbol{r}_{c1}$ 对发生在节点 C_1 的 s－a－0 故障进行定位，此时矢量 $\boldsymbol{r}_{c1}$ 会使 $O_2=0$；而其他情况，如节点 C_1 无故障、C_1 有 s－a－1 故障、其它节点（C_0 除外）发生故障（s－a－0 或 s－a－1）时，矢量 $\boldsymbol{r}_{c1}$ 使 $O_2=1$。同理用矢量 $\boldsymbol{r}_{c2}$ 和 $\boldsymbol{r}_{c3}$ 分别对节点 C_2 和 C_3 的 s－a－0 故障定位；用矢量 $\boldsymbol{r}_j$（$1\leqslant j\leqslant n$），以节点 O_2 作为观测点对发生在原始输入节点 x_j 的 s－a－0 故障进行定位，且以节点 O_4 作为观测点对发生在节点 $\bar{x}_j$ 的 s－a－0 故障定位。

对电路的输入节点施加 $n+3$ 个矢量：$\boldsymbol{t}_{c1}$，$\boldsymbol{t}_{c2}$，$\boldsymbol{t}_{c3}$，$\boldsymbol{t}_1$，$\boldsymbol{t}_2$，…，$\boldsymbol{t}_n$，各矢量的分量含义同矢量 $\boldsymbol{R}$。每一矢量有 2 个分量的值为 0，其他分量都为 1。其中分量 C_0 都为 0，矢量 $\boldsymbol{t}_{c1}$ 的分量 $C_1=0$，$\boldsymbol{t}_{c2}$ 的分量 $C_2=0$，$\boldsymbol{t}_{c3}$ 的分量 $C_3=0$，$\boldsymbol{t}_i$（$i=1,2,\cdots,n$）的分量 x_i 为 0。把节点 O_1 作为观测点，分别用矢量 $\boldsymbol{t}_{c1}$，$\boldsymbol{t}_{c2}$ 和 $\boldsymbol{t}_{c3}$ 对发生在节点 C_1，C_2 和 C_3 的 s－a－1 故障定位，例如，节点 C_1 有 s－a－1 故障时，矢量 $\boldsymbol{t}_{c1}$ 使 $O_1=1$，而当节点 C_1 有 s－a－0 故障及其他节点（C_0 除外）发生故障时，矢量 $\boldsymbol{t}_{c1}$ 使 $O_1=0$。同理用矢量 $\boldsymbol{t}_j$（$1\leqslant j\leqslant n$），以节点 O_1 作为观测点对原始输入节点 x_j 的 s－a－1 故障定位，且以节点 O_3 作为观测点对发生在节点 $\bar{x}_j$ 的 s－a－1 故障定位。

3.2　XOR 树的故障定位

由于 GRM 电路的 XOR 树的输入为 AND 门阵列的 AND 门输出，增加了对 XOR 树进行故障定位的难度。下面用 AB 表示在 XOR 树中从节点 A 至节点 B 的路径，包含 A 和 B 两个端点（节点）。由于 XOR 树是无扇出电路，从一个节点 A 至另一节点 B 的路径是唯一的。设每一个与门 AND_j（$j=1,2,\cdots,m$）的输出节点为 Y_j，电路的原始输出（主输出）节点为 Y，因此 Y_1Y，Y_2Y，…，Y_mY 都是 XOR 树中的路径。为 XOR 树的故障定

位设置两张表 Tab.1 和 Tab.2，Tab.1 用于存放可能是有故障的节点，Tab.2 用于存放无故障节点。

3.2.1 XOR 树 s－a－0 故障的定位 对 XOR 树的 s－a－0 故障定位采用算法 1。

算法 1

Step 1 使控制输入 $C_0=0$，$C_1=C_2=C_3=1$；置 Tab.1 和 Tab.2 为空。

Step 2 首先对门 AND_1，由于它的输入节点是最少的，因此可以选择电路的一个输入矢量 $\boldsymbol{X}_1$，$\boldsymbol{X}_1$ 使 AND_1 的输出 Y_1 的值为 1，而使其他与门 AND_j（$j>1$）的输出 Y_j 的值为 0。这里，选取的 $\boldsymbol{X}_1$ 是把与 AND_1 的输入对应的电路原始输入节点的值取为 1，而把原始输入的其它节点取值为 0。例如，对有 6 个原始输入 $x_1\cdots x_6$ 的 GRM 电路，若 AND_1 的输入为 x_1，x_3 和 x_4，则把其输入值取 1，而把 x_2，x_5 和 x_6 的值取 0。

对电路施加矢量 $\boldsymbol{X}_1$，若电路的输出 $Y=0$，则在 XOR 树中从节点 Y_1 至 Y 路径上有一个节点发生 s－a－0 故障，将路径 Y_1Y 中所有节点写入 Tab.1；若电路的输出 $Y=1$，则节点 Y_1 至 Y 路径上所有节点无故障，将 Y_1Y 路径上所有节点写入 Tab.2。

Step 3 对 AND 阵列的其他与门 AND_j（$j\geqslant 2$）可进行类似于门 AND_1 的处理，但此时当选择一个输入矢量 $\boldsymbol{X}_j$ 使门 AND_j 的输出 $Y_j=1$ 时，可能会使多个（设为 k 个）其他与门 AND_i（$i<j$）的输出 $Y_i=1$。下面针对路径 Y_jY 进行讨论。

在对路径 Y_jY 进行操作时，若在对 Y_i（$i=1,2,\cdots,j-1$）进行的操作中路径都无故障，此时把路径 Y_jY 与路径 $Y_{j-1}Y$ 的交点记为 C，对应 C 的 XOR 门输入节点记为 A 和 B，如图 4（a）所示。设输入矢量 $\boldsymbol{X}_j$ 会使 k 个门 AND_i（$i<j$）的输出 $Y_i=1$。当 $k=0$ 或偶数时，对被测电路施加矢量 X_j，若电路输出 $Y=0$，则路径 Y_jB 有故障；若 $Y=1$ 时，则路径 Y_jB 无故障。当 k 为奇数时则相反，若 $Y=1$，则 Y_jB 有故障；若 $Y=0$ 时，则 Y_jB 无故障。若确定了路径 Y_jB 有故障，则转 Step 4，若确定路径 Y_jB 无故障，则对路径 $Y_{j+1}Y$ 继续进行此步的故障定位操作。

Step 4 若发现一条路径有故障，则设此故障路径为 Y_tY。通过选取 XOR 树中的一些路径 Y_jY（$j>t$），并对电路施加多个矢量 $\boldsymbol{X}_j$，即可确定故障节点。首先考虑到与故障路径 Y_tY 相交的路径 Y_jY，见图 4b，施加矢量 $\boldsymbol{X}_j$，当 $k=0$ 或偶数时，若 $Y=0$，则 CY 故障（Y_tA 无故障），若 $Y=1$，则 Y_tA 故障（CY 无故障）；当 k 为奇数时，若 $Y=1$，则 CY 故障，若 $Y=0$，则 Y_tA 故障。其具体步骤见图 5。

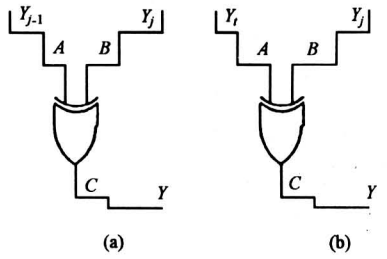

图 4 路径相交

Fig.4 The intersect of two paths

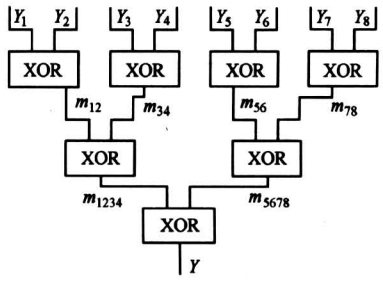

图 5 一个 XOR 树结构

Fig.5 An example of XOR gates tree

当已确定故障路径为 Y_1Y 时，对电路施加矢量 $\boldsymbol{X}_2$，若电路输出 $Y=0$ 则 $m_{12}Y$ 故障，节点 Y_1 无故障，若电路输出 $Y=1$ 则节点 Y_1 故障，$m_{12}Y$ 无故障。若是 $m_{12}Y$ 故障，则再用路径 Y_3Y 的矢量 $\boldsymbol{X}_3$ 判定是节点 m_{12} 还是路径 $m_{1234}Y$ 故障；若 $m_{1234}Y$ 有故障，则用路径 Y_5Y 的 $\boldsymbol{X}_5$ 判定是节点 m_{1234} 还是节点 Y 有故障。因此当路径 Y_1Y 有故障时，只需用 Y_jY 中的一些路径就可确定故障节点，例如图 5 中的 Y_2Y，Y_3Y 和 Y_5Y。若确定路径 Y_1Y 无故障，则故障路径在其他的子树上，例如当路径 Y_5Y 有故障时，由于 Y_1Y 无故障，因此节点 Y 无故障，是路径 Y_5m_{5678} 有故障，此时可对以节点 m_{5678} 为输出的子树进行处理，依次选取路径 Y_6m_{5678}、Y_7m_{5678} 来进行。

在算法进行过程中当发现故障路径时，将该路径中不在 Tab.2 中的节点写入表 Tab.1；当发现无故障路径时将其写入表 Tab.2，并在表 Tab.1 除去

此路径的节点。确定某一路径有故障，则使用 Setp 4 以缩小故障范围，当表 Tab.1 中只有一个节点时，该节点即为故障节点，算法结束。

3.2.2　对 XOR 树 s－a－1 故障的定位　首先确定 XOR 树是否有 s－a－1 故障。设零矢量 θ 为：控制输入 $C_0=C_1=C_2=C_3=0$，电路的所有原始输入节点取值 0。在零矢量 θ 作用下，若电路的原始输出 $Y=0$，则 XOR 树无 s－a－1 故障；若 $Y=1$，则 XOR 树有 s－a－1 故障。XOR 树的 s－a－1 故障定位与 s－a－0 故障定位类似，主要是在算法 1 中把 $Y=0$ 改为 $Y=1$，而把 $Y=1$ 改为 $Y=0$。每一个与门 AND_j 仍用前面的方法选取矢量 $\boldsymbol{X}_j$。例如相对应于 Setp 2，在矢量 $\boldsymbol{X}_1$ 的作用下，若 $Y=1$，则路径 Y_1Y 有 s－a－1 故障；若 $Y=0$，则路径 Y_1Y 无故障，因为故障 s－a－1 只可能发生在路径 Y_1Y 或其他路径 Y_iY $(2\leqslant i\leqslant m)$。

3.3　AND 阵列的故障定位

在进行故障定位时，依次确定门 AND_1，AND_2，…，AND_m 是否有故障，然后对故障节点进行定位。

3.3.1　AND 阵列 s－a－0 故障的定位　对 AND 门阵列的每一个 AND 门，由于每个输入节点的 s－a－0 故障和门输出节点的 s－a－0 故障等效，因此只需把故障定位到 AND 门。

算法 2

Step 1　使控制输入 $C_0=0$；置 $j=1$。

Step 2　选取电路的一个输入矢量 $\boldsymbol{Z}_j$，使 AND_j 的输出 $\boldsymbol{Z}_j$ 的值为 1。这里选取 $\boldsymbol{Z}_j$ 时是把对应 AND_j 输入节点的控制输入 C_i $(1\leqslant i\leqslant 3)$ 和电路原始输入的值全取 1，其他输入节点值取 0（与算法 1 中选取矢量 $\boldsymbol{X}_j$ 类似）。对被测电路施加矢量 $\boldsymbol{Z}_j$，记电路的输出为 $f(\boldsymbol{Z}_j)$，并计算电路无故障时的输出值 $g(\boldsymbol{Z}_j)$。若 $g(\boldsymbol{Z}_j)\neq f(\boldsymbol{Z}_j)$ 则 AND_j 有故障，算法结束；若 $f(\boldsymbol{Z}_j)=g(\boldsymbol{Z}_j)$ 则 AND_j 无故障，执行 Step 3。

Setp 3　置 $j\leftarrow j+1$，若 $j<m+1$，转 Step2；若 $j=m+1$，输出所有 AND 门无 s－a－0 故障的相关信息，算法结束。

3.3.2　对 AND 阵列 s－a－1 故障的定位　AND 阵列的 s－a－1 故障主要有 AND 门的输入节点故障和输出节点故障，把 AND 门的输出节点故障作为 XOR 树的输入节点故障处理，这里仅考虑 AND 门输入节点的 s－a－1 故障。

算法 3

Step 1　使控制输入 $C_0=0$；置 $j=1$。

Step 2　对电路 AND 阵列的门 AND_j，依次确定其输入节点 u_i（$i=1$，2，…，K_j，K_j 为输入节点数）和输出节点 Y_j 的 s－a－1 故障，置 $i=1$。

Step 3　对门 AND_j 的输入节点 u_i，生成矢量 $\boldsymbol{U}_i$，使该节点 u_i 值为 0，而使门 AND_j 的其他输入节点 u_j 值为 1，对 AND_j 未使用的其他控制输入 C_i $(1\leqslant i\leqslant 3)$ 值和原始输入值为 0。施加矢量 $\boldsymbol{U}_i$，记电路的输出为 $f(\boldsymbol{U}_i)$，并计算电路无故障时的输出值 $g(\boldsymbol{U}_i)$。若 $f(\boldsymbol{U}_i)=g(\boldsymbol{U}_i)$，则 AND_j 的节点 u_i 无故障，执行 Setp 4。若 $f(\boldsymbol{U}_i)\neq g(\boldsymbol{U}_i)$，则 AND_j 的输入节点 u_i 或输出节点 Y_j 有故障。由于在算法 1 中已对节点 Y_j 的故障情况做了处理。若节点 Y_j 有故障，u_i 就无 s－a－1 故障，执行 Step 4；若节点 Y_j 无故障，u_i 就有 s－a－1 故障，输出相关的故障信息，算法结束。

Step 4　置 $i\leftarrow i+1$，若 $i\leqslant K_j$，转 Step 3；若 $i>K_j$，执行 Step 5。

Step 5　置 $j\leftarrow j+1$，若 $j\leqslant m$，转 Step 2；若 $j>m$，则所有 AND 门的输入节点无 s－a－1 故障。输出相关信息，算法结束。

算法 2 和算法 3 说明：

1）算法 2 中的矢量 $\boldsymbol{Z}_j$ 和算法 3 中的矢量 $\boldsymbol{U}_i$ 有相同的性质，都使所有门 AND_s（$s>j$）的输出 $Y_s=0$，以及对 AND 阵列中的与门按 AND_1，AND_2，…，AND_m 的次序操作，保证了算法的有效性。

2）算法 3 中对无故障电路输出 $g(\boldsymbol{U}_i)$ 的计算，可根据逻辑函数 GRM 模式中每一积项的表达形式，模拟计算在矢量 $\boldsymbol{U}_i$ 下每个门 AND_k（$1\leqslant k\leqslant j$）无故障时的输出值 Y_k。若 Y_k 中等于 1 的个数为偶数或 0，则电路输出 $Y=0$，为奇数时 $Y=1$。算法 2 中 $g(\boldsymbol{Z}_j)$ 的计算与此类似。

3）在算法 2 和算法 3 中，对控制输入 C_1，C_2 和 C_3 的扇出分支节点，可作为 AND 阵列中 AND 门的输入节点进行故障定位。

在具体进行故障定位时，可首先对被测电路施加测试集 T，若发现电路有故障（本文假定只发生一个故障 s－a－0 或 s－a－1），再做故障定位。在做定位时，若用电路每部分的测试集 T_1，T_2 和 T_3 能将故障确定于输入与控制、AND 阵列、XOR

树中的一个部分，则只需对相应的部分进行，否则首先对电路的输入与控制部分进行故障定位；然后对 XOR 树及 AND 阵列部分做故障定位。

3.4 实验结果

已将如上故障定位方法用 C++ 语言在 586 计算机（Pentium4，1.5 GHZ）上实现，实验时电路结构采用电路描述语言（CDL）描述；然后生成检测电路故障的测试集 T；对电路依次设置每一节点的 s－a－0 或 s－a－1 故障，并由算法来进行定位。选取了 15 个逻辑布尔函数，变量最多的一个函数有 19 个变量，25 项积项，图 2 所示电路是实验时选取积项数最少的一个。结果表明：故障定位准确率均为 100%；对 GRM 电路节点的 s－a－0 或 s－a－1 故障进行定位所需的平均时间＜2 min；随着电路规模的增大，故障定位时间会有所增加。

4 结论

在数字电路的可测性设计中，研究逻辑函数电路可测性的实现具有实用价值，逻辑函数的 GRM 形式可以使用 AND 阵列和 XOR 门树结构来实现，用本文方法所得的电路是完全可故障定位的。此外，**a.** 本文采用增加三个控制输入 C_1，C_2 和 C_3 的方法，在于减少 GRM 电路的测试集规模，否则电路测试集规模会较大[6]。关于不使用控制输入并能得到规模较小的测试集的设计方法正在研究中。**b.** 对控制输入 C_0 的故障定位，在文中仅讨论了扇出源的情形，但对其区分扇出源与扇出分支节点的故障，也是可进行定位的。**c.** XOR 门级联是 XOR 树的特殊情形，因此在电路设计时可以用 XOR 门级联代替 XOR 树，并不需要使用控制输入 C_1，C_2 和 C_3，可直接应用文中方法对故障做定位，但整个电路的延迟比使用 XOR 树结构时要大一些。今后将在如何用较少的元件来实现电路的可测性、更有效的测试方法等方面做进一步研究。

参考文献

[1] 杨士元.数字系统的故障诊断与可靠性设计 [M].北京:清华大学出版社,2000

[2] 陈光禹,张世箕.数据域测试及仪器 [M].北京:电子工业出版社,1994

[3] 潘中良.数字电路测试的神经网络方法的研究与实现 [D].成都:电子科技大学,1997

[4] 陈光禹,潘中良.可测性设计技术 [M].北京:电子工业出版社,1997

[5] Mike W, Bennett B, Ley A. Boundary scan: the internet of test [J]. IEEE Design & Test of Computer, 1999, (July): 34～43

[6] Sasao T. Easily testable realization for generalized Reed-Muller expressions [J]. IEEE Trans Computer, 1997, 46(6):709～716

[7] Drechsler R, Becker B, Drechsler N. Genetic algorithm for minimization of fixed polarity Reed-Muller expressions [J]. IEE Proc Computer and Digital Techniques, 2000,147(5): 349～354

[8] Sasao T, Debnath D. Generalized Reed-Muller expressions: complexity and an exact minimization algorithm [J]. IEICE Trans Fundamental, 1996, E78(12):2123～2130

A Fault Location Approach for the Testable Realization of Logic Functions

Pan Zhongliang

(*Department of Physics, South China Normal University, Guangzhou* 510631, *China*)

[**Abstract**] An approach of design for testability(DFT) for logic functions is presented in the paper, which employs AND gates and XOR gates tree to realize the generalized Reed-Muller expression of arbitrary logic functions. The major features of the approach are: 1) The circuits adopting the DFT techniques in the paper are totally fault locatable. 2) The circuits have universal test sets for fault detection, the cardinality of the test sets is $(n+5)$, where n is equal to the number of input variables in the logic function. A fault location method for the circuits is presented, which can identify all fault equivalence classes in the AND gates, and the faults in XOR gate tree in the circuits.

[**Key words**] logic functions; Reed-Muller expressions; design for testability; single stuck at fault; faults location