软硬件测试中预确定距离测试

朱经纷 徐拾义
(上海大学计算机工程与科学学院 上海 200072)

摘 要 在随机测试的基础上提出了VLSI电路测试中的一个新概念,即预确定距离测试。随机测试广泛应用于软硬件测试中已经有多年了。众所周知,随机测试中每个测试码都是随机选取的而不管它是否与先前生成的测试码重复。尽管由于测试码选取的随机性使得随机测试并不是十分有效,但是对它作了一些实质性修改从而大大提高了它的测试效率。在预确定距离测试中,总是选择总距离最大的测试码来进行测试,以便使得该测试码所检测到的故障与先前的测试码所检测到的故障尽可能地不同。还详细介绍了构造一个预确定距离测试序列的生成算法,并将其应用到软件测试中。最后,从基准电路上获得的实验结果以及从理论上的分析也表明这种新方法的有效性。

关键词 随机测试,海明距离,笛卡儿距离,生成矩阵,预确定距离
中图法分类号 TP306 **文献标识号** A

Pre-determined Distance Testing for Hardware and Software Testing

ZHU Jing-fen XU Shi-yi
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)

Abstract This paper, based on the random testing, introduced a new concept of so called Pre-determined Distance Testing (PDDT) for VLSI circuits. Random testing has been employed for years in both software and hardware testing. It is well known that in random testing each test requires to be selected randomly regardless of the tests previously generated. Although random testing could be inefficient for its random selection of test patterns, some essential modifications have been adopted in PDDT to improve the testing efficiency. In this new method, the total distance among all test patterns is chosen maximal so that the set of faults detected by one test pattern is as different as possible from that of faults detected by the tests previously applied. Procedure of constructing a pre-determined distance test sequence (PDDTS) and its application in software testing were described in detail. Experimental results from Benchmark as well as theoretical analysis were also given to evaluate the performances of our new approach.

Keywords Hamming distance, Cartesian distance, Generation matrix, Pre-determined distance

1 引言

随着软硬件系统规模的快速增长,实际工作中对软硬件进行穷尽测试已经是不可能的了[1,2]。一些确定性的测试方法对于确保检测到大型被测系统中的所有故障已经显得力不从心了。因此,在许多实际测试的过程中经常采用降低故障覆盖率(大约为80%~90%)的方法来作为最终测试目标。由于硬件测试中不可能达到100%的故障覆盖率,而软件测试中不可能达到100%的分支覆盖率,这就使得硬件采取随机测试方法而软件采用黑盒测试方法,这是一种更合适的处理上述问题的方法。实际上,软件系统的黑盒测试也可以被称为随机测试。本文中我们将提出一些软硬件测试中均适用的随机测试新方法。

一般而言,测试的生成问题需要对电路(硬件)或程序(软件)中的结构化信息进行处理。随机测试不需要考虑专门的测试生成问题。然而,随机测试最大的不足之处在于为一组指定故障集随机生成的测试码个数要比一种固定方法所生成的测试码个数大得多(通常可能达到10倍或10倍以上[3])。在相同条件下,与固定生成方法相比,这种随机测试技术可能并不十分有效。于是,便产生了如何提高随机测试的效率这样一个问题,即在电路或程序中如何使用尽可能少的测试码和测试时间来测试出尽可能多的故障。

如前所述,随机测试中每一个测试码都是随机选择的,而不管是否与先前应用的测试码重复。这就会产生两个对随机测试效率有极大影响的重要问题[4,5],我们必须仔细考察并提出相关解决方案。

(1)测试过程中同一测试码可能会出现多次,这将增加测试码总数。

(2)虽然随机测试选取的两个或多个测试码不同,但是它们可能测试到类似的甚至是相同的故障。这类似于浪费了宝贵的测试时间和可用资源,从而增加了测试成本,降低了测试效率。

到稿日期:2008-06-24 本文受国家自然科学基金资助项目(60473033)资助。
朱经纷(1983-),男,硕士生,研究方向为软件测试、容错计算,E-mail:zjf1983@163.com;徐拾义(1941-),男,教授,博士生导师,研究方向为容错计算、可信计算、VLSI设计与测试、软件测试。

在下一节，我们首先介绍一些基本概念，接着阐述测试码之间距离（包括海明距离和笛卡尔距离）与测试效率之间的关系。

2 背景和相关工作

本节中我们首先介绍一些相关工作。以下列出的定义1—定义6可以在文献[6,7]中找到类似的定义。

定义1 两个测试码 t_i 和 t_j 之间的海明距离 $HD(t_i,t_j)$ 定义为测试码间对应的不同二进制位个数。

定义2 一个测试码序列的总海明距离（THD）定义为测试码 t_i 与测试码序列中 t_i 之前的测试码之间海明距离的总和[8]，即

$$THD(t_i)=\sum_{j=1}^{i-1}HD(t_i,t_j)$$

定义3 测试集S的最大总海明距离（MTHD）定义为，选择测试集中每一个所有可能的测试码 t_i，使得 t_i 与之前已选择的测试码序列 $t_1,t_2,\cdots,t_{i-1}$ 之间的总海明距离达到最大值，即

$$MTHD(t_i)=\max_{t_i}\{THD(t_i)=\sum_{j=1}^{i-1}HD(t_i,t_j)\}$$

定义4 测试码 $A=(a_1,a_2,\cdots,a_j,\cdots,a_n)$ 和测试码 $B=(b_1,b_2,\cdots,b_j,\cdots,b_n)$ 之间的笛卡尔距离 $CD(A,B)$ 定义为

$$CD(A,B)=\sqrt{|a_1-b_1|+|a_2-b_2|+\cdots+|a_n-b_n|}$$

定义5 一个测试码序列的总笛卡尔距离（TCD）定义为测试码 t_i 与测试码序列中 t_i 之前的测试码之间笛卡尔距离的总和，即

$$TCD(t_i)=\sum_{j=1}^{i-1}CD(t_i,t_j)$$

定义6 测试集 S 的最大总笛卡尔距离（MTCD）定义为，选择测试集中每一个所有可能的测试码 t_i，使得 t_i 与之前已选择的测试码序列 $t_1,t_2,\cdots,t_{i-1}$ 之间的总笛卡尔距离达到最大值，即

$$MTCD(t_i)=\max_{t_i}\{TCD(t_i)=\sum_{j=1}^{i-1}CD(t_i,t_j)\}$$

下面通过一个例子来说明如何求最大总海明距离（MTHD）和最大总笛卡尔距离（MTCD）。

例1 考虑以下序列

$t_1=$ (00000)　$t_1=$ (00000)　$t_1=$ (00000)

$t_2=$ (11111)　$t_2=$ (11111)　$t_2=$ (11111)

$t_3=$ (00011)　$t_3=$ (00001)　$t_3=$ (11110)

(1)　(2)　(3)

在第1个序列中，$HD(t_1,t_2)=5$，$THD=\sum_{j=1}^{2}HD(t_3,t_j)=2+3=5$；而在第2个和第3个序列中，$THD(t_3)=1+4=5$。因此，这3个测试序列都有相同的总海明距离，并且达到了最大值，即 $MTHD(t_3)=5$。

然而，当增加第3个测试码时如果我们考虑总笛卡尔距离，那么不是所有这3个测试码序列的总笛卡尔距离都能够达到最大值。这是因为总笛卡尔距离依据选择测试码方式不同而变化很大。如第1个序列中，总笛卡尔距离 $TCD(t_3)=\sqrt{2}+\sqrt{3}=3.146$。在第2和第3个序列中，总笛卡尔距离 $TCD(t_3)=\sqrt{1}+\sqrt{4}=3$。因此，第1个序列已经达到了最大总笛卡尔距离，即 $MTCD(t_3)=3.146$，而另外两个测试码序列没有达到最大总笛卡尔距离。

3 最大距离测试序列

本文的主要目的是使得随机测试的效率尽可能高。因而，我们必须处理好第1节中提到的传统随机测试的两个主要问题。通过进行大量的实验和仔细研究后，我们发现测试中一重要的有用的现象，即两个测试码的故障覆盖率与它们之间的距离有着十分密切的关系。根据最大似能原理和所做的实验，我们作一基本假设：两个测试码间的距离越大，那么它们能检测到的故障数就越多。反之，如果两个测试码之间的距离足够小，那么它们可能检测到几乎相同的故障。换句话说，这个基本假设表明距离大的测试码比距离小的测试码能够检测到更多的不同故障。此外，这个假设既适用于硬件测试也适用于软件测试[9]。如果测试时考虑测试码间的距离所花的成本要比穷尽测试少，那么这种方法就有可能比传统随机测试效率高。虽然，到目前为止这仅仅是本文所依赖的一个假设，但是通过大量实验和实际工作我们已经取了一些非常丰富的成果。接下来考察硬件测试中的一个例子。

例2 考虑一个具有快速进位的4-位全加器。该全加器有9条输入线（包括2个4-位数据输入线（A_4,A_3,A_2,A_1），（B_4,B_3,B_2,B_1）和来自低位的进位输入线 C_0）和5条输出线（包括1个4-位求和输出线（$\Sigma_4,\Sigma_3,\Sigma_2,\Sigma_1$）和向高位进位的输出线 C_5）。假设我们使用10个测试码 t_0-t_9 来测试电路中所有输入线上（$C_0,A_4,A_3,A_2,A_1,B_4,B_3,B_2,B_1$）上的单固定型故障，如图1所示。从图中可以看到，任意两个相邻测试码间的海明距离等于1，即 $HD(t_i,t_{i+1})=1(i=0,\cdots,8)$。这个测试集所能检测到的输入线上单固定型故障如表1所列，其中每个1(0)表示行中的测试码能够检测到相应列中输入线上的 s-a-1(s-a-0)故障。例如，测试码 $t_1=$(0000 00001)能够检测到输入线（$C_0,A_4,A_3,A_2,A_1,B_4,B_3,B_2$）上的 s-a-1 故障以及输入线（B_1）上的 s-a-0 故障。而测试码 $t_9=$(1111 11111)能够检测到所有输入线（$C_0,A_4,A_3,A_2,A_1,B_4,B_3,B_2,B_1$）上的 s-a-0 故障。

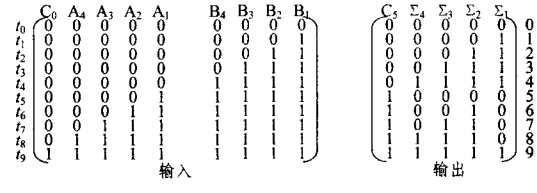

图1 全加器输入及相应输出

表1 测试集所能检测到的故障

	C_0	A_4	A_3	A_2	A_1	B_4	B_3	B_2	B_1
t_0	1	1	1	1	1	1	1	1	1
t_1	1	1	1	1	1	1	1	1	0
t_2	1	1	1	1	1	1	1	0	0
t_3	1	1	1	1	1	1	0	0	0
t_4	1	1	1	1	1	0	0	0	0
t_5	1	1	1	1	0	0	0	0	0
t_6	1	1	1	0	0	0	0	0	0
t_7	1	1	0	0	0	0	0	0	0
t_8	1	0	0	0	0	0	0	0	0
t_9	0	0	0	0	0	0	0	0	0

从表1中可以发现，$HD(t_0,t_1)=1$，测试码 t_0 和测试码 t_1 所分别检测到的故障除了在输入线 B_1 上不同外几乎都相同。其中 t_0 能检测到输入线 B_1 上的 s-a-1 故障而 t_1 能检测

输入线 B_1 上的 s-a-0 故障。然而测试码 t_0 和 t_9 所分别检测到的故障却完全不同，这可能是因为 t_0 与 t_9 之间的距离已达到最大值，即 $HD(t_0,t_9)=9$。同时，这反过来也说明了前面所作假设的合理性和可行性。

此外，在随机测试中，如果我们碰巧是按表 1 所示的顺序 t_0—t_9 来选择测试码，那么就只有当这 10 个测试码都选择完后才能彻底检测到输入线上的所有故障。但是，如果我们先选择测试码 t_0，然后直接就选择测试码 t_9，或者先选择测试码 t_9 接着选择测试码 t_0，则仅需要 2 个测试码就可以检测到输入线上所有可能的单固定型故障。基于这种思想，我们得出结论：随机测试的测试效率与测试码选择次序有着紧密的关系，即下一个测试码的选择是由之前已经选择好的测试码序列来确定的，并且新的测试码与之前确定好的测试码序列之间的距离应当是最大的。

我们把上述这种方法称为有序随机测试(ORT)。与传统随机测试不同，有序随机测试中每一个测试码的选择都显式地取决于目前已选择好的测试码序列。其主要思想是在选择测试码时考虑测试码之间的距离，使得每次选择测试码时尽可能与之前的测试码不同。以下定义将在后面用来处理这些问题。

定义 7 最大海明距离测试序列(MHDTS)是一个测试码序列，其中每一个测试码及对应的反码同时存在于该序列中，并且整个测试码序列的距离达到最大总海明距离(MTHD)。

如(0101,1110,0001,1010)是一个最大海明距离测试序列(MHDTS)，因为(0101,1010)和(1110,0001)是分别彼此互为反码的。但是(0101,1010,0111,0001)不是一个最大海明距离测试序列(MHDTS)。文献[7]中详细叙述了构造最大海明距离测试序列的算法。在一个有 $2k$ 个测试码的最大海明距离测试序列中，下标为偶数的测试码($t_0,t_2,\cdots,t_{2k-2}$)是随机选择的，而下标为奇数的测试码($t_1,t_3,\cdots,t_{2k-1}$)是通过对下标为偶数的测试码分别按位取反而得到的。现在假设一个具有 n 输入的函数，通过使用该算法生成了一个有 $2k$ 个测试码的测试集，那么该测试集的总海明距离为 $THD(t_{2k-1})=k^2n$[7]，该距离为最大总海明距离(MTHD)。因此，以这种算法生成的测试码序列是一个 MHDTS。然而在这个序列中仅仅考虑了海明距离，换句话说仅仅一半测试码可以达到最大距离。因而，仅考虑最大海明距离的测试码序列称为准最大距离测试序列(SMDTS)，因为测试序列中下标为偶数的测试码是随机选择的而没有使得它与之前的测试码保持最大距离。只有在测试码序列中同时考虑最大总海明距离(MTHD)和最大总笛卡尔距离(MTCD)，整个距离才能达到最大。换句话说，在构造一个测试码序列中当选择下标为偶数的测试码时应该考虑最大总笛卡儿距离，而当选择下标为奇数的测试码时应该考虑最大总海明距离(MTHD)，这样整个序列才会是一个总最大距离测试序列(TMDTS)。TMDTS 的正式定义如下。

定义 8 如果一个测试码序列在每次选择一个新的测试码后都包含一个总最大距离，那么该测试码序列就被称为总最大距离测试序列(TMDTS)。

接下来的算法描述了如何生成一个总最大距离测试序列。

算法 1 构造一个 TMDTS

步骤 1 对于每一 n 个输入变量，任意选择一个向量来作为第一个测试码 t_0。

步骤 2 为了从剩余的测试码中得到下标为奇数的新测试码，只需对之前的测试码按位取反即可。为了从剩余的测试码中得到下标为偶数的新测试码，选择与之前已经选择好的测试码序列能够达到最大总笛卡尔距离的测试码。然后把它加入到所选择的测试码序列中以形成一个新的测试码序列。

步骤 3 重复步骤 2 直到所有 2^n 个测试码都已经选过一次了，如图 2 所示。

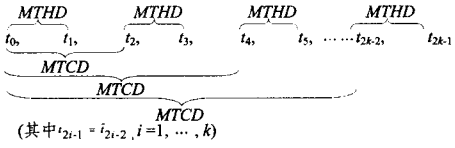

图 2 算法 1 图示

4 构造预确定距离测试序列

在上一节算法 1 中，我们知道每一个新选择的测试码应该使得整个测试码序列的总距离为最大值。同时，也看到 MTHD 仅需对之前测试码按位取反就可以很容易地获得。然而，要真正获得一个与之前的测试码序列有最大总笛卡尔距离(MTCD)的新测试码却有点困难。

理论上，直接通过分别计算和比较所有剩余测试码与已经确定好的测试码序列之间的距离值，我们每次总是可以选择能产生最大总笛卡尔距离(MTCD)的测试码。令人遗憾的是，这个算法的时间复杂度达到 $O(4^n)$，该复杂度对即使是小规模的系统也无法忍受，这种算法的本质是遍历每一种可能情况。事实上，这种遍历所有可能剩余的测试码直到找到一个合适的测试码是一个可满足问题[10]。为了解决这个问题，在实际应用中需要采取一些折中的方法。本文所采取的方法是利用从一个给定的生成多项式或生成矩阵生成的所谓线性递归序列。虽然这种方法可能不能保证生成一个理论上是 MTCD 的测试码序列，但是这种方法总是能使所选择的测试码的 TCD 尽可能地大。尤其对那些输入线很多(50 或更多)的系统来说，这种折中的方法几乎能够完成算法 1 所描述的功能。

定义 9 如果一个测试序列每次在选择一个新的测试码时都包含一个 MTHD 和一个预先确定的距离，那么这个测试序列就被称为预确定距离测试序列(PDDTS)。

同样，本文也给出了如何构造一个 PDDTS 的算法。事实上，仅需对算法 1 中步骤 2 作一些修改就可以形成新的算法 2，从而有助于计算距离，极大地减小测试序列生成的时间复杂性。

算法 2 构造一个预确定距离测试序列(PDDTS)

步骤 1 同算法 1 中步骤 1 相同。

步骤 2 从一个给定的生成矩阵选择一个新的向量为序列中第 $2i$ 个测试码 t_{2i}($i=1,2,\cdots,k$)，使得 $TCD(t_{2i})$ 和预先确定的距离一样大。然后对 t_{2i} 取反得到测试码 t_{2i+1}($i=0,1,2,\cdots,k$)，即 $t_{2i+1}=\bar{t}_{2i}$。将两者均加入到测试码序列中。

步骤 3 同算法 1 中的步骤 3 相同。

另外，生成矩阵的实现可以很容易地由线性反馈移位寄存器等目前比较成熟的方法来完成。我们来看一个构造 PDDTS 的例子，首先考察一个(15,7)递归序列的生成多项式：

$$g(x)=1+x^4+x^6+x^7+x^8$$

我们可以很容易地得到相应的生成矩阵为：

$$G(x)=\begin{pmatrix}100000010001011\\010000011001110\\001000001100111\\000100010111000\\000010001011100\\000001000101110\\000000100010111\end{pmatrix}$$

容易看出，由 $G(x)$ 中横向量的不同线性组合所生成的所有测试码与其它测试码之间有一个确定的最小海明距离。而且，在本例中任意两个测试码 t_i,t_j 之间的海明距离有如下关系：$HD(t_i,t_j)\geqslant 5$。因此，任意两个测试码之间的笛卡尔距离必定 $CD(t_i,t_j)\geqslant\sqrt{5}$。换句话说，依据所选择的何种生成矩阵，我们可以预设最小笛卡尔距离。

这样，我们可以构造出一个测试序列，其中每对互反的测试码之间的总海明距离 $THD=15$，剩余的测试码对之间的总笛卡尔距离 $TCD\geqslant\sqrt{5}$，如图 3 所示。

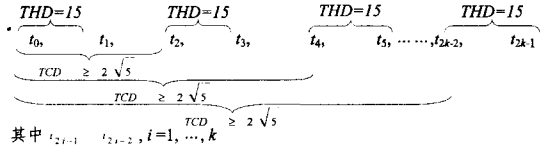

图 3　预确定距离测试序列

其中 $t_{2i-2}(i=1,\cdots,k)$ 是由生成矩阵 $G(x)$ 的向量线性组合而生成的测试码。

值得注意的是预先确定的距离与测试序列长度有着密切的关系。这是因为距离越大，所能生成的可用测试码个数越少。如在上述 $G(x)$ 中，首先确定(最小)海明距离为大于或等于 5，那么我们最多有 $2^7=128$ 个测试码可用，其中每对可能的测试码之间的笛卡尔距离大于或等于$\sqrt{5}$。再加上取反后得到的测试码，我们可应用的测试码总数最多为 256 个。然而，如果我们将海明距离增大到 7，那么新的生成矩阵中可用的测试码个数为 $2^5=32$ 个，这比我们所需要的测试码个数要少得多，很可能完全不能满足随机测试的需求，这也就是预确定距离的确定应当根据实际情况和实际工作的需要而定的主要原因。正常情况下，预确定距离不应选择太大，否则测试码序列总长度会缩短得很快。

我们使用 SMDTS 和 PDDTS 在基准电路上做了大量实验。实验结果如表 2 所列，同时表中也给出了新方法和传统方法之间的对比。从实验结果来看，PDDTS 的测试效率要比传统方法的测试效率高得多。

表 2 给出了使用不同方法(传统方法、SMDTS 和 PDDTS)在大部分基准电路上要达到 82%以上的故障覆盖率所需的测试码个数。我们对每种电路都进行了 10 次实验，表中的结果为 10 次实验的平均值。表中第 1 列表示基准电路名称以及随机测试选取的测试码总数，其它各列分别表示在相应方法下要达到 82%的故障覆盖率所需的测试码数以及运行这些测试码需要的时间。

表 2　实验结果

基准电路/测试序列长度	达到 82%的故障覆盖率所需的测试码数/运行时间		
	传统方法	SMDTS	PDDTS
	(长度/时间)	(长度/时间)	(长度/时间)
C880/500	364 / 2.8	190 / 2.7	154 / 2.1
C1355/500	427 / 2.6	198 / 2.6	169 / 2.4
C1908/800	504 / 4.4	326 / 3.7	287 / 3.2
C2670/800	593 / 4.5	382 /3.8	324 / 3.3
C3540/800	632 / 4.5	421 / 3.6	346 / 2.9
C5315/1500	912 / 5.2	623 / 4.7	529 / 3.8
C6288/2000	1334 / 7.5	805 / 8.6	637 / 5.4
C7552/2000	1805 / 15.1	1043 / 11.3	845 / 7.9

5　软件测试中的 PDDTS

通常在软件测试中由于黑盒测试简单而便捷，使得它比白盒测试更受人们的青睐。如前所述，测试数据选择是软件测试(尤其是黑盒测试)中的一个重要组成部分[11]。前面阐述的 PDDTS 思想同样可以应用于软件的黑盒测试中。我们可以将输入空间划分为多个子域，使得软件与同一子域的输入的响应方式类似。

一般而言，一个程序的输入可以是数值、字符串、数据结构以及它们的组合。为了方便起见，本文仅讨论数值型输入。同样，我们可以对任意两个输入定义一个距离并且构造一个 PDDTS 来进行软件测试。假定 x 是一个实数型变量，其取值范围为$[x_a,x_b]$。现在我们将 x 的值映射为一个 3-位二进制数(b_1,b_2,b_3)，其中$(b_i(\{0,1\},i=1,2,3)$。这样 x 的取值被分为 6 个子区域，每个子区域都对应着一个 3-位二进制编码，如图 4 和表 3 所示。

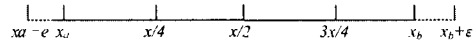

图 4　变量 x 划分为 6 个子域

根据表 3 所列的映射规则，程序中每一个变量都被编码成一个 3-位二进制数。这样整个程序的输入值就可以看作为二进制向量，类似于硬件中的输入码一样。根据最大距离和预确定距离的思想，我们专门为输入值设计距离，以便它们之间距离尽可能大。例如我们将变量 x 的最小值 x_a 和最大值 x_b 分别映射成(000)和(111)，这样它们之间的海明距离达到最大值。

表 3　变量映射规则

二进制编码	对应的 x 值	说明
000	x_a	
001	$(x_a,x/4]$中随机选取	
010	$(x/4,x/2]$中随机选取	
011	$(x_a-\varepsilon,x_a)$中随机选取	非法值
100	$(x_b,x_b+\varepsilon)$中随机选取	非法值
101	$(x/2,3x/4]$中随机选取	
110	$(3x/4,x_b]$中随机选取	
111	x_b	

此外，我们还考虑了非法输入值$(x_a-\varepsilon,x_a)$和$(x_b,x_b+\varepsilon)$，以便能够对程序的健壮性进行测试。一旦确定了映射规则就可以自动地生成测试码。下面通过一个例子来说明在软件测试中如何实施该思想。假设被测程序 P 具有 5 个输入变量 A,B,C,D 和 E(均为实数型变量)，那么输入码形式应该为 $A_1A_2A_3B_1B_2B_3C_1C_2C_3D_1D_2D_3E_1E_2E_3$，其中 $A_i,B_i,C_i\in\{0,1\}$。

根据第 4 节描述的 PDDTS 的生成矩阵，我们可以按以下方式来首先选择 6 个测试码。所有下标为偶数的测试码 t_0，t_2 和 t_4 是从生成矩阵 $G(x)$ 中选择的，下标为奇数的测试码 t_1，t_3 和 t_5 是由 t_0，t_2 和 t_4 取反得到的。其余的测试码生成与前 6 个测试码的生成类似。接下来我们根据表 3 中的规则将这些 PDDTS 中的二进制输入码转换成实数来作为被测程序的测试用例。

6 PDDTS 测试效率的数学分析

通常，在随机测试时如果已知一些条件，为了能够覆盖到指定的故障集我们需要粗略估算出所需的测试码总数[12]。文献[13]中给出了一种有效的方式，并且给出了所需测试码总数 T 的上界为

$$T=\left\lceil \frac{\ln(1-\sqrt[n]{C})}{\ln(1-d_{\min})} \right\rceil \tag{1}$$

其中 C 是由测试者确定的可信度，n 是被测电路中的故障总数，$d_{\min}$ 是由一个随机测试码所最难检测到故障的概率。

事实上，当给定一个被测电路时式(1)中的 3 个参数有两个参数是常数，而另一个参数 $d_{\min}$ 是可以通过各种方式来改进的。设计 PDDTS 的目的就是用来使得 $d_{\min}$ 尽可能高，从而直接减少所需的测试码个数。从上述非正式的描述中可以看出，对式(1)中的分子部分，由于 n 对任何给定的电路来说是一个常数，而 C 是在测试前必须预先给出的常数。因此，式(1)的分子部分仍然不变。而分母部分 $\ln(1-d_{\min})$ 的绝对值即使在 $d_{\min}$ 增加一点点也会快速增长。因为 $(1-d_{\min})$ 的值属于 $(0,1)$ 而 $\ln(1-d_{\min})$ 的值属于 $(-\infty,0)$。因而即使 $d_{\min}$ 的值增加一点点，相应对数值也会增加很多，式(1)的值就会降低很多，自然所需的测试码总数就会大大减少。上述讨论可由图 5 所示的对数曲线来描述。

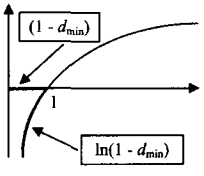

图 5 两者之间的关系

图 5 中粗线表明如果 $d_{\min}$ 稍微增加一点，$\ln(1-d_{\min})$ 的绝对值就会增加很多，即大大减少所需的测试码个数。下面通过一个具体的例子来详细阐明上述思想。

例 3 设有一电路 N，其中共有 $n=500$ 个独立故障，且 $d_{\min}=0.005$，假定预先确定的可信度为 $C=0.95$，那么由传统随机测试所需的测试码总数应为

$$T=\left\lceil \frac{\ln(1-\sqrt[n]{C})}{\ln(1-d_{\min})} \right\rceil=\left\lceil \frac{\ln(1-\sqrt[500]{0.95})}{\ln(1-0.005)} \right\rceil=\lceil 1818.4 \rceil=1819$$

而在 PDDTS 中，最难检测到的故障的最小概率值 $d_{\min}$ 有所增加。假定 $d_{\min}$ 从 0.005 增加到 0.007，则所需的测试码总数应为

$$T=\left\lceil \frac{\ln(1-\sqrt[n]{C})}{\ln(1-d_{\min})} \right\rceil=\left\lceil \frac{\ln(1-\sqrt[500]{0.95})}{\ln(1-0.007)} \right\rceil=\lceil 1307.5 \rceil=1308$$

也就是说测试码总数从 1819 下降为 1308，节省了 500 多个测试码但同时达到相同的可信度。值得注意的是，$d_{\min}$ 增加一点点时所需的测试码总数却可大大减少。

结束语 随机测试广泛应用于各种软硬件测试中。然而，随机测试由于其测试码选择的随机性而使得在实际测试过程中随机测试的测试效率并不是十分理想。本文分析了影响随机测试测试效率的因素，提出了有序随机测试(ORT)这一思想，并通过实验和理论分析证明了有序随机测试的有效性。虽然从理论上来说，我们应当选择一个最大距离测试序列来进行测试，但由于生成一个最大距离测试序列的时间复杂性太大，使得其反而会降低随机测试的测试效率。因此本文采取一种折中的方法，即构造预确定距离测试序列来降低其时间复杂性，这样既保证了测试效率又降低了时间复杂性。此外，本文还将这一思想应用于软件测试中，但软件测试中的实验证明还有待进一步研究和完善。

参 考 文 献

[1] Beizer B. Software Testing Techniques [M]. Van Nostrand Reinhold, 1990

[2] Meyers G. The Art of Software Testing [M]. New Jersey: John Wiley & Sons, Inc., 1979

[3] SethS, Agrawal V, Farhat H. A Statistical Theory of Digital Circuit Testability [J]. IEEE Transactions on Computers, 1990, 39(4): 582-586

[4] Wagner K D, Chin C K, McCluskey E J. Pseudorandom Testing [J]. IEEE Transactions on Computers, 1987, 36(3): 332-343

[5] Malaiya Y K, Yang S. The Coverage Problem for Random Testing [C]// Proceedings of the 13th International Test Conference. 1984: 237-242

[6] Xu Shiyi. Random-like Testing of Very Large Scale Integration Circuits [J]. Journal of Shanghai University, 1998, 2(4): 279-183

[7] Xu Shiyi. Maximum Distance Testing [C]// Proceeding of the 11th IEEE ATS'02. 2002: 15-20

[8] Malaiya Y K. Antirandom Testing: Getting the Most Out of Black-box Testing [C]// Proceeding of the 6th International Symposium on Software Reliability Engineering. 1995: 86-95

[9] Malaiya Y K, von Mayrhauser A, Srimani P K. An Examination of Fault Exposure Ratio [J]. IEEE Transactions on Software Engineering, 1993, 19(11): 1087-1094

[10] Abramovici M, Breuer M A, Friedman A D. Digital Systems Testing and Testable Design [M]. New York: Computer Science Press, 1990

[11] Cohen D M, Dalal S R, Kajla A, et al. The Automatic Efficient Test Generator (AETG) System [C]// Proceedings of ISSRE. 1994: 303-309

[12] Majumdar A, Vrudhula S B K. Fault Coverage and Test Length Estimation for Random Pattern Testing [J]. IEEE Transactions on Computers, 1995, 44(2): 234-247

[13] Hamlet R, Taylor R. Partition Testing Does Not Inspire Confidence [C]// Proceedings of the 2nd Workshop on Software Testing, Verification and Analysis. 1988: 206-215