利用整数存储无理数的测试数据编码压缩方法

詹文法[1,2], 梁华国[2], 程一飞[1], 吴海峰[1], 朱世娟[1]

[1] (安庆师范大学计算机与信息学院 安庆 246133)
[2] (合肥工业大学电子科学与应用物理学院 合肥 230009)
(zhanwf@aqtc.edu.cn)

摘 要: 针对集成电路测试过程中自动测试设备需要传输大量测试数据到被测芯片, 浪费了大量的测试数据传输时间, 不能降低芯片测试成本的情况, 提出一种整数存储无理数的测试数据编码压缩方法. 首先将测试数据按游程长度划分, 默认第 1 个游程长度为小数的个位, 其他游程长度依次为小数的小数位, 将测试数据转换成小数; 然后提出用二分查找无理数的方法, 将该小数转化成可以整数表示的无理数; 最后存储无理数对应的整数表示 m, l, k. 该方法采取传输测试数据规律而不是测试数据本身的方法, 理论上可以将整个测试集的存储转化成对单个或若干个无理数对应整数表示的存储. 对部分 ISCAS89 标准电路中规模较大的时序电路进行实验, 结果表明, 在同样实验环境下, 其压缩效果方面优于 Golomb 码、FDR 码、EFDR 码、MFVRCVB 码等成熟的编码方法.

关键词: 测试数据压缩; 无理数; 静态编码; 动态编码
中图法分类号: TP391.41

Test Data Compression Coding-based Scheme Storing Integers Represented for Irrational Numbers

Zhan Wenfa[1,2], Liang Huaguo[2], Cheng Yifei[1], Wu Haifeng[1], and Zhu Shijuan[1]

[1] (*School of Computer & Information, Anqing Normal University, Anqing* 246133)
[2] (*School of Electronic Science and Applied Physics, Hefei University of Technology, Hefei* 230009)

Abstract: A compression method for test data coding is presented, by storing irrational numbers in integer form against large-amount time waste of transmission of test data and high cost of chip testing due to large-amount test data transmitted to chips for test via automatic testing device in integrated circuit test in procedures as follows. Firstly, dividing test data according to run lengths and setting the first run length as the default digit before the decimal point, other lengths as decimal part respectively, before transforming test data into its decimal form. Secondly, transforming the decimal into an irrational number that can be expressed in integer form via binary search of irrational numbers. Thirdly, storing integers corresponding to their irrational form for expressing m, l and k. The method is theoretically available to transform the storage of whole test unit into that expressed in integer form corresponding to single or several irrational numbers because it transmits the rule of test data but not the test data itself. The experimental results on larger sequential circuits in ISCAS89 benchmark circuits show that, the test data can be compressed with higher efficiency than that compressed by commonly-applied coding such as Golomb, FDR, EFDR and MFVRCVB under the same experimental circumstances.

Key words: test data compression; irrational number; static encoding; dynamic encoding

收稿日期: 2015-10-28; 修回日期: 2016-04-01. 基金项目: 国家自然科学基金(61306046, 61540011); 安徽省学术技术带头人后备人选(GXBJZD2016075, 2015H053). 詹文法(1978—), 男, 博士, 教授, 硕士生导师, CCF 会员, 主要研究方向为测试数据压缩、可测性设计等; 梁华国(1959—), 男, 博士, 教授, 博士生导师, CCF 会员, 主要研究方向为测试数据压缩、嵌入式系统综合与测试、数字系统设计自动化等; 程一飞(1976—), 男, 硕士, 副教授, CCF 会员, 主要研究方向为测试数据压缩等; 吴海峰(1982—), 男, 硕士, 副教授, 主要研究方向为测试数据压缩等; 朱世娟(1983—), 女, 硕士, 讲师, 主要研究方向为测试数据压缩等.

随着集成电路的发展，如何处理越来越庞大的测试数据已成为集成电路测试的关键难题之一. 根据 ITRS 在 2010 年的报告数据，测试一个芯片，在 2009 年仅仅需要 85 个测试模式数，只需要压缩比为 80；而到 2019 年，对测试模式数的要求则需要达到 20 370 个，对压缩比的要求则需要达到 12 000. 仅仅 10 年，模式数增加到 240 倍，压缩比增加到 150 倍[1].

减少测试数据量一直是该领域的研究热点，主要分为测试集压缩方法[2]、内建自测试方法[3-4]和静态编码压缩方法[5-18]3 类. 1) 测试集压缩方法. 其特点是确保在不降低故障覆盖率的情况下有选择性尝试使用不同的敏化路径，通过算法调整测试立方体中无关位的位置，或者将相容的 2 个或多个测试向量合并成单一测试向量的方法来减少总的测试向量的个数，达到测试集最小化或最优化. 该方法的优势是所有工作由软件实现，不会增加额外的硬件成本. 但其缺点也非常明显，测试向量个数的减少造成了对非模型故障的覆盖率的降低；另一个缺点是压缩后测试集的测试数据量仍然非常庞大，很难一次性完全直接存储在 ATE 的存储器中，因此需要与其他方法结合使用. 2) 内建自测试方法. 在被测电路中新增一部分电路专门用于测试，能够完成测试模式生成、测试控制、测试调度和测试结果分析，这样可以不依赖外部的自动测试设备独立进行测试；能够通过减少昂贵的自动测试设备的成本费来达到节约测试成本的目的，还可以支持测试重用和全速测试. 其缺点在生成测试模式时产生了大量的对测试毫无贡献的测试模式，既增加了测试功耗，又浪费了测试时间；另外，有部分未设计内建自测试方法的 IP 核也限制其应用. 3) 静态编码压缩方法. 通过编码方法用一个较小的测试集 T_E 去编码测试集 T_D，将存储和传输时对 T_D 的操作变换成对 T_E 的操作. 即存储 T_E 在 ATE 的存储器中，测试时仅传输 T_E 而不是 T_D 到被测芯片，最终 T_E 到 T_D 的还原由被测芯片上的解压电路来完成. 静态编码方法采用的是无损压缩方法，还原后的测试集的确定位跟原始测试集的确定位完全一一对应，还原后的无关位都被程序填充成了特定的值，因此其只会增加而不会降低被测电路的故障覆盖率；该方法的另一优势是可以不需要提供被测电路的内部结构，能够很好地保护 IP 核的知识产权. 因此，该方法在集成电路测试领域中得到了广泛应用. 当前已有很多比较成熟的编码，如 Golomb 编码[5]，FDR 码[6]，VIHC 码[7]，交替连续码[8]，Variable-Tail 码[9]，混合游程码[10]，SVIC 码[11]，变游程码[12]，EFDR[13]，MFVLC[14]，MFVRCVB[15]，AFDR[16]和共游程码[18]等.

本文提出一种动态编码压缩方法，将游程长度出现的规律表示成形如 $\frac{\sqrt[k]{m}}{l}$ 的无理数(其中 m，l，k 全部是整数)，存储时只用存储 m，l，k 和原始测试数据长度 p 这 4 个整数，将整个测试集的存储变换成单个或若干个无理数对应的整数存储；另外，提出一种二分查找无理数的方法，可以快速查找无理数所对应的整数表示，减少了算法的复杂度.

1 二分查找无理数编码算法

1.1 问题引入

不失一般性，假设原始的测试集 T_D={1011XX-111011X01100X1X1X010}，共 26 位. 传统的编码方法是采用对该测试数据按编码规则进行代码字替换，然后存储代码字，ATE 的存储器中存储的是代码字本身，称为直接编码. 本文选择适当的方法将上述测试数据的无关位有选择性地填充，为叙述简单性，将本实例中所有无关位全部填充为 1，则上述测试集就被填充为 T_D={10111111011101100-11111010}. 如果采用 1 这种类型的游程编码方法，其对应的游程长度分别为 1, 7, 3, 2, 0, 5 和 1. 接下来，将游程长度转换成小数 t，其原则是默认第一个游程长度为小数的整数部分，其余依次为小数的小数部分，即得到 t=1.732051，此时完成了测试集到小数的转换. 推而广之，即可以按照一定的规则将整个测试集转换成单个或若干个小数来表示. 现在的问题是小数并不能直接存储在 ATE 中，必须要解决小数的存储问题. 无独有偶，将无理数 $\sqrt{3}$ 展开成小数，有 $k=\sqrt{3}\approx 1.732051$，其前 7 位对应的数字正好是 1, 7, 3, 2, 0, 5 和 1. 显然，小数 t=1.732051 不易于直接存储，但无理数 $k=\sqrt{3}$ 很易于存储，只用存储被开方数 3、开方次数 2 和原始长度 26 就可以了，这样就可以将 26 位原始测试数据的存储变换成 3，2 和 26 这 3 个整数的存储. 理想情况下可以将单个测试向量，甚至整个测试集变换为一个或若干个形如 $\frac{\sqrt[k]{m}}{l}$ 的无理数来存储，即将传统的对整个测试集的存储变换对一个或若干个确定的无理数对应整数表示的存储.

1.2 无理数表示方法

通过第 1.1 节分析可知，将测试集变换为无理数，

需要经过 2 个步骤: 1)将测试集转化为小数; 2)将小数转化为无理数.

1.2.1　测试集转化为小数的方法

由于测试数据本身只含有“0”, “1”和“X”(无关位)3 种数据, 一种简单的方法是统计游程长度, 例如, 对于第 1.1 节中的测试集 T_D= {1011XX111011-X01100X1X1X010}, 将无关位全部填充为 1, 其对应的游程长度分别为 1, 7, 3, 2, 0, 5 和 1, 将第一个游程长度作为小数的个位, 后面所有游程长度依次连接后作为小数部分, 从而构成小数 1.732 051, 这样就把对测试向量的存储转换成对小数的存储. 然而在实际的测试集中可能有的游程长度很长, 如对于测试集 T_D={1111XX11111X1110}, 其游程长度为 15, 超过了 9(十进制所能表示的最大长度), 这时用 1 位十进制就不能表示, 可以采用“切分游程”或“使用更大进制”的方法解决: 1) 将原始游程长度进行切分, 切分后的游程长度最大不超过 1 位数据最大值 9; 2) 使用大于十进制数的更大进制的表示方法, 如采用 n 进制数表示, 其中 n 不小于最大游程长度. 这 2 种方法也可以混合使用, 以达到最优效果.

需要指出的是, 在特殊情况下, 如果测试数据的游程长度非常长, 超过了 n 进制所能表示的最大游程长度 l_{max}, 这时需要将该长游程切分成若干段编码. 原则是将该长游程连续切分成若干个长度等于 l_{max} 的游程和 1 个小于 l_{max} 的游程来分别进行编码, 这样划分的理由是为了解压方便, 解压时可以通过判断当前游程长度与 l_{max} 的值是否相等来确定后续游程与当前游程在编码前是否是同一个游程. 如对于游程为 000000000 000000000 0001 按 0 类型游程编码, 超过了十进制所能表示的最大游程长度 9, 需要将其分割成 000000000, 000000000 和 0001 这 3 个部分, 即需要编码的游程长度分别为 9, 9 和 3; 解压时根据当前的游程长度是否为 9 来判断后续游程与当前游程是否属于同一游程, 从而可以确定在当前游程解压结束时是否需要输出 1(如果是对 0 类型游程编码)或 0(如果是对 1 类型游程编码). 特殊情况下, 游程长度正好为 9, 即为 0000000001, 这时需要切分为 000000000 和 1 共 2 段, 即其长度分别为 9 和 0.

1.2.2　小数转化为无理数对应整数表示的方法

为了叙述方便, 不失一般性, 记小数为 x, 无理数为 $\frac{\sqrt[k]{m}}{l}$(其中 m, l, k 全部是整数). 上述问题实际上就是找适当的整数 m, l, k, 使 $\frac{\sqrt[k]{m}}{l}$ 展开成小数时, 其前 p(p 为小数 x 的所有位数)项正好等于 x. 压缩时只用存储整数 m, l, k 和 p, 这 4 个整数的存储可以用定长码, 如经典的游程编码, 也可以用变长码, 如 Golomb 码或 FDR 码等, 本文采用偶数位标记编码, 详见第 1.3 节.

二分查找无理数编码算法的难点是如何计算出确定小数对应的无理数的整数表示. 一种可行的策略是不直接计算无理数的值, 通过二分查找法逐次逼近查找到需要的无理数, 其流程图如图 1 所示.

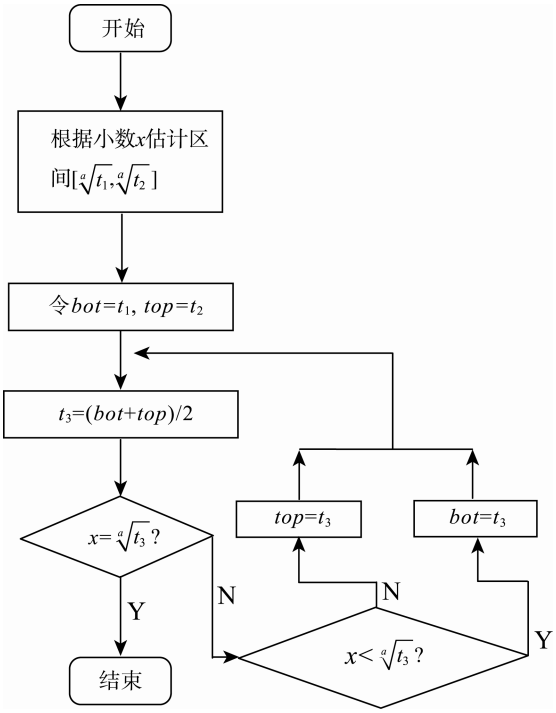

图 1　二分查找无理数编码算法流程图

首先根据上述确定的小数 x 估计该小数对应的无理数所在的区间 $\left[\sqrt[q]{t_1},\sqrt[q]{t_2}\right]$(其中 a 为开方次数, 为整数; 为方便计算, t_1, t_2 的初始值为整数), 使 $x\in\left[\sqrt[q]{t_1},\sqrt[q]{t_2}\right]$; 再在[$t_1$, t_2]上取 t_3, 使 $t_3=\frac{t_1+t_2}{2}$, 并判断 $\sqrt[q]{t_3}$ 在适当精度的四舍五入情况下与 x 是否相等, 否则, 判断 x 落在 $\left[\sqrt[q]{t_1},\sqrt[q]{t_3}\right]$ 还是 $\left[\sqrt[q]{t_3},\sqrt[q]{t_2}\right]$; 此过程一直继续, 直到 $\sqrt[q]{t_3}$ 在适当精度的四舍五入情况下与 x 相等. 其中, t_3 是形如 $\frac{t_3'}{2^n}$ 的分数(其中 t_3' 和 n 都是整数), $\sqrt[q]{\frac{t_3}{2^n}}=\frac{\sqrt[q]{t_3}}{\sqrt[q]{2^n}}$, 通过分母有理化

即可转换成 $\frac{\sqrt[k]{m}}{l}$ 的形式.

例如, 对于小数 x=1.732, 首先根据 1.7<x=1.732<1.8 计算得到 2<1.7^2=2.89 和 1.8^2=3. 24<4, 取 t_1=2, t_2=4, 得到 x 所在无理数区间为 $\left[\sqrt{2},\sqrt{4}\right]$, 取 $\sqrt[a]{t_3}=\sqrt[a]{\frac{t_1+t_2}{2}}=\sqrt{3}$, 比较 $\sqrt{3}$ 与 x 的大小, 此时有 $\sqrt{3}$ 的前 4 位正好与小数 x 相等, $\sqrt{3}$ 即可所求无理数, 对应的有 m=3, k=2, l=1. p 为原始测试数据长度, 17 位, p=17.

1.3 无理数对应整数表示的存储方法

通过第 1.2 节的分析, 可以将单一测试向量, 甚至整个测试集变换成单个或若干个形如 $\frac{\sqrt[k]{m}}{l}$ 的无理数. 现在的问题是如何存储无理数, 该问题可以转化为整数 m, l, k 的存储问题. 考虑到测试向量或测试集的长度, 还需要一个整数 p 来表示长度, 即每个无理数需要存储 4 个整数. 这 4 个整数的存储可以用传统的编码方法, 如定长码中的经典的游程编码, 或用变长码中的 Golomb 码或 FDR 码等. 不失一般性, 本文对这 4 个整数的编码全部采用偶数位标记编码[19]. 相比传统的变长码, 如 Golomb 码、FDR 码等, 偶数位标记编码的优点是奇数位完全表示代码字的长度信息, 偶数位表示代码字是否结束信息, 这样可以分割开不同的代码字, 使每个代码字能够自包含. 考虑到实际应用中整数 m, l, k 和 p 均大于 0, 将文献[19]中偶数位标记编码稍加修改, 使游程长度从 1 开始编码, 码表如表 1 所示. 可以看出, 偶数位标记编码具有如下特点: 1) 每个代码字的奇数位和偶数位数即长度相等. 如游程长度为 13 的代码字奇数位为 110, 偶数位为 001, 长度均为 3 位. 2) 偶数位表示代码字是否结束, 为 0 表示代码字继续, 为 1 表示本代码字结束; 奇数位表示游程长度信息, 在奇数位前添加 1 位数据 1, 这时对应的十进制数值与实际表示游程长度信息相差 1 所对应的二进制数值. 如游程长度为 13 的代码字为 101001, 偶数位为 001, 解压电路根据偶数位最后 1 位数据判断该代码字结束; 奇位数为 110, 将其前面增加 1 位数据 1, 变成 1110, 该数值对应的十进制长度信息为 14, 比其代表的长度信息 13 多 1. 3) 每组之间, 如从 A_i 组过渡到 A_{i+1} 组时, 代码字的长度增加 2 位, 奇数位和偶数位都分别增加 1 位. 4) A_i 组包含元素的个数为 2^i. 5) 码值游程长度 t 和所在组 A_i 的对应关系为, $i=\left\lceil \mathrm{lb}(t+2)-1\right\rceil$. 需要指出的是, 对于 2)中的 2 条性质, 游程长度 t 代表的十进制数值相比 1 与奇数位 a 组合成二进制数的数值之间差 1, 即码值的大小可以直接由奇数位来表示, 奇数位组成的二进制数的数值比实际期望表示的数值大 1, 用偶数位是否为 1 来表示代码字是否结束. 该 2 条性质可用于解压电路的设计, 详见第 2.1 节.

表 1 偶数位标记编码方法码表

t	组数	奇数位(a)	偶数位(b)	代码字
1	A_1	0	1	01
2		1	1	11
3	A_2	00	01	00 01
4		01	01	00 11
5		10	01	10 01
6		11	01	10 11
7	A_3	000	001	00 00 01
8		001	001	00 00 11
⋮		⋮	⋮	⋮
13		110	001	10 10 01
14		111	001	10 10 11
⋮				

一个测试集进行无理数编码的实例如图 2 所示.

原始测试集 001 0000001 0000000001 0X1 000XX1 XXX00XX01

无关位填充后的测试集: 001 0000001 0000000001 001 000001 000000001 (38 位)

对应小数为 2.692582

对应的无理数为 $\frac{\sqrt{29}}{2}$

存储内容为 2, 29, 2, 38

编码为 11 10101001 11 0000101011(22位)

图 2 测试集进行无理数编码实例

2 解压电路

2.1 硬件解压

解压电路的难点是将整数进行开方运算, 一种方法是借助于CPU中集成的浮点处理单元x87 FPU, 即利用SoC自身的CPU模块实现开方运算. x87 FPU具有自己的指令系统, 包括常用的指令类型: 浮点传送指令、浮点算术运算指令、浮点超越函数指令、浮点比较指令和 FPU 控制指令[20]. 浮点超越函数指令可以实现指数运算(F2XM1 指令)和对数运算(FYL2X 指令). 可以将 $\sqrt[k]{m}$ 转换成 $2^{(\mathrm{lb}\,m)/k}$, 即可使用指数运算指令和对数运算指令计算出结果. 另一种方法本文是自行设计解压电路, 其结构类似于浮点处理单元 x87 FPU 中指数运算单元和对数运算单元. 由于实际的SoC都包含有CPU, 因此最

可行的方法是采用第一种方法.

本文方法的解压电路结构如图 3 所示，其中包含一个有限状态机、一个特殊的 t+1 位计数器、一个 T 触发器和一些组合逻辑，同时利用了 SoC 中的 CPU 模块. 解压电路嵌入到芯片中，具有规模小和结构简单的特点.

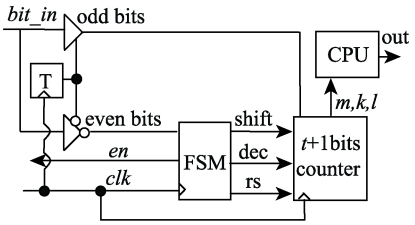

图 3　解压电路结构

这个特殊的 t+1 位计数器特别之处在于：设定计数器最低位为 1，当数据被移入计数器需要被解码时，该位和其他数据一起移到计数器的高位. 此功能可以使用一些组合电路实现，且不会明显增加硬件开销. $t+1$ 位计数器中，t 的值是由压缩结果中被开方数和开方次数的最大值 $L_{\max}$ 决定，$t=\left\lceil \mathrm{lb}\left(L_{\max}+2\right)\right\rceil-1$.

解压电路的工作过程如下：

Step1. 有限状态机(finite state machine, FSM)发出控制信号使能信号 *en* 为 1，由 *bit_in* 输入 ATE 的数据通过一个 T 型触发器被分割成奇数位和偶数位两部分. t+1 位计数器最低位预先初始化为 1，其他位初始化为 0，奇数位数据直接移入 t+1 位计数器，偶数位数据移入 FSM.

Step2. FSM 重复读取编码数据直到偶数位的值等于 1；同时，把奇数位的编码数据移入 t+1 位计数器.

Step3. 等待特殊的 t+1 位计数器完全接收到当前代码字的奇数位数据后，将 t+1 位计数器进行一次自减运算后再把 t+1 位计数器的数据(m)移入 CPU.

Step4. 重复 Step2, Step3，把得到的 t+1 位计数器的数据(k, l 和 p)移入 CPU.

Step5. CPU 利用浮点处理单元 x87 FPU 进行开方运算，计算 $\frac{\sqrt[k]{m}}{l}$，输出其二进制形式的前 p 位.

2.2　软件解压

软件解压方法的流程图如图 4 所示，基本原理是先由与自动测试设备相连的控制计算机估算出无理数对应高精度小数的大概范围，再通过多次二分迭代的方法逐渐逼近要找的高精度小数，可以边解压边运行测试，这样既能避免复杂的开方运算，又能避免开方运算的等待时间. 需要指出的是，本文方法中传送到自动测试设备的不是原始的测试数据，而是运算过后游程长度的编码.

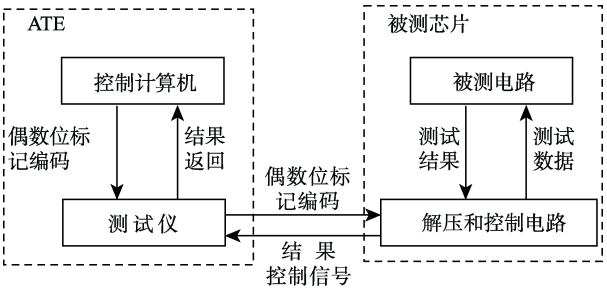

图 4　软件解压方法流程图

不失一般性，以十进制为例，假设原始的测试集为 001 00XX001 XX000XXXX1 XX1 000001 XX-X00X0X1 001，可以转化为无理数 $\frac{\sqrt{29}}{2}$，即对应的 m=29, l=2, k=2, p=41. 解压步骤如下：

Step1. 由控制计算机估计出单个无理数所对应的单精度或双精度小数，无理数对应的小数为 $b_0.b_1b_2\cdots b_{n-1}b_n$，其中 $b_0,b_1,b_2,\cdots,b_{n-1},b_n$ 为对应的游程长度，记 $t_1=b_0.b_1b_2\cdots b_{n-1}\left(b_n-1\right)$，$t_2=b_0.b_1b_2\cdots\ b_{n-1}\left(b_n+1\right)$. 将整数部分 b_0 和前 n–1 位小数部分 b_{n-1} 对应的游程长度依次转换成游程编码，本文采用偶数位标记编码. 再将偶数位标记编码的代码字依次由测试仪的通道输入解压和控制电路，控制计算机记下传输到 ATE 通道的游程数量记为 u. 由解压和控制电路将偶数位标记编码还原成原始测试数据传输到被测电路(circuit-under-test, CUT). 假设由控制计算机算出 $\frac{\sqrt{29}}{2}=2.692$，控制计算机将 2, 6, 9 对应的游程长度转换成传统的游程编码，如偶数位标记编码后依次为 11，1011，001001；再依次由测试仪通道输入解压和控制电路，由如图 3 所示的偶数位标记编码解压电路将其转化成 001 000001 0000000001，并传输到 CUT. 传输的游程数量 u=3，另有 $t_1=2.691$，$t_2=2.693$，这些数据保存在控制计算机中.

Step2. 记 $\frac{\sqrt[k]{m}}{l}=t$，对该式变换后有 $m=(tl)^k$，容易证明 $t_1\leqslant t\leqslant t_2$，即有 $(t_1l)^k\leqslant(tl)^k=m\leqslant(t_2l)^k$. 令 $t'=\frac{t_1+t_2}{2}$，判断 $(t'l)^k$ 与 m 的大小情况. 若 $(t'l)^k<m$，执行下一步；若 $(t'l)^k>m$，转 Step4；若 $(t'l)^k=m$，转 Step5.

Step3. 比较 t_1 与 t'，找出相同数据的位数，即为已经解压的游程数量. 将从 u 开始到 t_1 与 t' 相同的数据位数的游程长度转换成传统的游程编码，如偶数位标记编码后，通过测试仪通道输入到解压和控制电路，由解压电路转换成原始测试向量，再传输到 CUT，并更新 u 为 t_1 与 t' 相同的数据位数. 令 $t_1=t'$，重复 Step2.

Step4. 比较 t_2 与 t'，找出相同数据的位数，即为已经解压的游程数量. 将从 u 开始到 t_2 与 t' 相同的数据位数的游程长度转换成传统的游程编码，如偶数位标记

编码后，通过测试仪通道输入到解压和控制电路，由解压电路转换成原始测试向量，再传输到 CUT，并更新 u 为 t_2 与 t' 相同的数据位数．令 $t_2=t'$，重复 Step2.

Step5．将从 u 开始到 p 结束的游程长度转换成传统的游程编码，如偶数位标记编码后，通过测试仪通道输入到解压和控制电路，由解压电路转换成原始测试向量，再传输到 CUT.

Step6．将测试结果与期望的理论值进行比较，如果结果一致，则被测芯片测试的通过；否则，被测芯片的测试不通过.

上述例子中，由控制计算机算出 $\frac{\sqrt{29}}{2}$=2.692，控制计算机将 2，6，9 对应的游程长度转换成传统的游程编码，如偶数位标记编码后依次为 11，1011，001001；通过测试仪通道输入到解压和控制电路，由解压电路转换成原始测试向量，再传输到 CUT 由 ATE 通道传输该编码给被测芯片．首先有 $t'=\frac{t_1+t_2}{2}=\frac{2.691+2.693}{2}=2.692$，计算 $(t'l)^k=(2.692\times 2)^2=28.98746<m=29$；$t_1$ 和 t' 相同的位数为 4 位，本次由 ATE 通道传输新增第 4 位数据 2 按偶数位标记编码为 11 再传输到被测芯片解压和控制电路，由解压将其还原成 001 传输到 CUT，令 $t_1=t'=2.692$，重复 Step2，有 $t'=\frac{t_1+t_2}{2}=\frac{2.692+2.693}{2}=2.6925$，计算 $(t'l)^k=(2.6925\times 2)^2=28.99823<m=29$；$t_1$ 和 t' 相同的位数为 4 位，该 4 位已经全部由 ATE 通道传输给被测芯片过了，不再传输数据到被测芯片，令 $t_1=t'=2.6925$，重复 Step2，有 $\frac{2.6925+2.693}{2}=2.69275$，计算 $t'=\frac{t_1+t_2}{2}=(t'l)^k=(2.69275\times 2)^2=29.00361>m=29$；$t_2$ 和 t' 相同的位数仍然为 4 位，该 4 位已经全部由 ATE 通道传输给被测芯片了，仍然不再传输数据到被测芯片，令 $t_2=t'=2.69275$，重复 Step2，有 $t'=\frac{t_1+t_2}{2}=\frac{2.6925+2.69275}{2}=2.692625$，计算 $(t'l)^k=(2.692625\times 2)^2=29.00092>m=29$；此过程一直进行到 t' 的前 7 位分别 2，6，9，2，5，8 和 2，即有 $t'\approx 2.692582$，测试过程结束.

3 整体综合过程

理论上，一个测试集可以压缩为一个无理数．但在实际应用时，考虑到时间复杂度和空间复杂度，一般将一个测试集压缩为一组无理数．整体综合过程采取分段查找无理数的方法，逐步增大段长可以达到局部最优化效果.

Step1．采用自动测试模式生成工具 ATPG，生成确定的完全测试集 T，记其测试向量个数为 N.

Step2．选取从第 1 位开始的连续前若干位为确定位个数最多的某一测试向量，随机将其他所有测试向量级联，即将一个向量的尾部接另一个向量的首部，记为 S，其长度记为 w.

Step3．从前往后按 0 类型游程统计游程长度，直到无关位结束．若无关位与前一游程可以组成同一游程，舍弃最后一游程长度，只记录前若干游程长度；否则，记录前所有游程长度，并将游程长度信息转化为小数来表示．转化原则如下：将每段长为 q 位游程长度转化为一个小数 x，该段第一个游程长度定义为小数的个位，其他全部作为小数的小数部分，令 q=10.

Step4．查找无理数 $\frac{\sqrt[k]{m}}{l}$．实际上就是找适当的整数 m，l，k，使 $\frac{\sqrt[k]{m}}{l}$ 展开成小数时其前 p(p 为小数 x 的所有位数)项正好等于 x，压缩时只用存储整数 m，l，k 和 p．令 k=2，查找形如 $\frac{\sqrt[k]{m}}{l}$ 的无理数，取小数 x 的整数部分和小数部分前 t 位，构成新的小数，记为 y，计算 y^k 并仅保留其整数部分，记为 a，即 $a=\lfloor y^k \rfloor$；计算 $\left(y+(0.1)^t\right)^k$ 的值，向上取整数，记为 b，即 $b=\left\lceil \left(y+(0.1)^t\right)^k \right\rceil$，则有 $x\in\left(\sqrt{a},\sqrt{b}\right)$.

Step5．二分无理数区间．逐次逼近 x，取 $c=\frac{a+b}{2}$，计算 $\sqrt[k]{c}$ 并与 x 比较，判断其前 q 位能否与 x 一一对应．如果一一对应，执行下一步；否则，若 $\sqrt[k]{c}<x$，则令 a=c，重复本步骤；若 $\sqrt[k]{c}>x$，则令 b=c，重复本步骤，直至 $\sqrt[k]{c}$ 前 q 位能与 x 一一对应，执行下一步.

Step6．去除游程长度信息前 q 位，取剩余游程长度信息的前 q 位转化为小数 x，重复 Step4，Step5，直至游程长度信息为空.

Step7．取 $q=q+1$，重复 Step4~Step6，直至压缩率不再增长，此时的压缩结果即为对应 k 值得最优结果.

Step8．取 k=3,4,…,10000，重复 Step4~Step7，共得到 9 999 个无理数．比较这些无理数的编码结果，取压缩率最大的一组无理数，压缩时只用存储无理数中的整数 m，l，k 和 p.

4 实验结果及分析

本节通过实验来验证本文方案的压缩效果，并与当前成熟的方法进行比较．为了增加实验的

可比性，使用与文献[5-8,13-15]同样的实验环境，采用通用的 Mintest 测试生成工具产生的测试集，分别对 ISCAS89 标准电路中 6 个规模较大的时序电路进行实验. 本文的实验结果是在取十六进制的基础上实现的，主要考虑以下两点：1) 十六进制比十进制可以表示的游程长度更长；2) 十六进制比 n(n>16)进制在运算和存储方面都有很大优势，解压时的代价比较小，容易实现.

ISCAS89 标准电路的实验结果如表 2 所示，可以看出，本文方案对 ISCAS89 标准电路的压缩率在 52.85%~86.10%变化，其平均压缩率可以达到 65.42%.

表 2　本文方案的压缩效果

电路名称	原始测试集/位	编码后数据/位	压缩率/%
s5378	23 754	10 439	56.05
s9234	39 273	17 534	55.35
s13207	165 200	22 966	86.10
s15850	76 986	22 727	70.48
s38417	164 736	77 674	52.85
s38584	199 104	56 384	71.68
平均			65.42

为了进一步验证本文方案的效果，将其与国内外成熟方案[5-8,13-15]在同等实验环境下比较，在 ISCAS89 标准电路的实验结果如表 3 所示. 其中，第 3 列为 Golomb 码压缩结果，第 4 列为 FDR 码压缩结果，第 5 列为 EFDR 码压缩结果，第 6 列为交替连续码压缩结果，第 7 列为 MFVLC 码压缩结果，第 8 列为 MFVRCVB 码压缩结果，第 9 列为 VIHC 码压缩结果. 从表 3 可以看出，本文方案比其他方案压缩效果要好. 对于 ISCAS89 标准电路的 6 个时序电路的压缩效率的平均值，本文方案比 Golomb 码高 15.54%，比 FDR 码高 10.09%，比 EFDR 码高 7.49%，比交替连续码高 7%，比 MFVLC 码高 4.99%，比 MFVRCVB 码高 1.81%，比 VIHC 码高 4.41%. 与其他方案相比，本文方案在 6 个时序电路中有 4 个电路压缩效果最好，在 s38584 中压缩效果仅次于 MFVLC 的压缩效果，这些数据进一步说明了本文方案的稳定性.

表 3　8 种方案压缩效果比较　　%

电路名称	本文方案	文献[5]	文献[6]	文献[13]	文献[8]	文献[14]	文献[15]	文献[7]
s5378	**56.05**	40.70	48.19	50.81	45.12	52.15	53.29	51.78
s9234	**55.35**	43.34	44.88	45.89	42.79	45.82	53.80	47.25
s13207	**86.10**	74.48	78.67	79.38	80.43	81.58	83.95	83.51
s15850	**70.48**	41.77	52.87	56.29	65.13	67.70	67.41	67.94
s38417	52.85	44.12	54.53	52.35	56.52	43.06	**58.15**	53.36
s38584	71.68	54.86	52.85	62.91	60.57	**72.29**	65.06	62.28
平均	**65.42**	49.88	55.33	57.93	58.42	60.43	63.61	61.01

5　结　语

传统的编码方法都是直接存储代码字，压缩效果必然不会超过其理论极限熵的值，而集成电路的发展又需要非常高的压缩效果. 本文提出一种动态编码压缩方法，其不直接用代码字来存储游程长度，而是将游程长度出现的规律表示成形如 $\frac{\sqrt[k]{m}}{l}$ 的无理数，存储时只用存储 m, l, k 和原始测试数据长度 p 这 4 个整数；可以将对整个测试集的存储转换成了对单个或若干个无理数对应的整数存储.

本文方案同样具有编码方案的优点，即能够独立于被测电路而不需要了解被测电路的内部结构，能够很好地保护知识产权.

参考文献(References):

[1] Test and test equipment. International technology roadmap for semiconductors[OL]. [2015-10-28]. http://www.itrs.net/Links/2010ITRS/Home2010. htm

[2] El-Maleh A H, Khursheed S S, Sait S M. Efficient static compaction techniques for sequential circuits based on reverse-order restoration and test relaxation[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2006, 25 (11): 2556-2564

[3] Liang Huaguo, Liu Jun, Jiang Cuiyun, *et al*. Constraint input reduction BIST scheme for multiple scan chains[J]. Journal of Computer Aided Design & Computer Graphics, 2007, 19(3): 371-375(in Chinese)
(梁华国, 刘　军, 蒋翠云, 等. 约束输入精简的多扫描链 BIST 方案[J]. 计算机辅助设计与图形学学报, 2007, 19(3): 371-375)

[4] Xiang D, Chen M J, Fujiwara H. Using weighted scan enable signals to improve test effectiveness of scan-based BIST[J]. IEEE Transactions on Computers, 2007, 56(12): 1619-1628

[5] Chandra A, Chakrabarty K. System-on-a-chip test-data compression and decompression architectures based on Golomb codes[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2001, 20(3): 355-368

[6] Chandra A, Chakrabarty K. Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes[J]. IEEE Transactions on Computers, 2003, 52 (8): 1076-1088

[7] Gonciari P T, Al-Hashimi B M, Nicolici N. Variable-length input huffman coding for system-on-a-chip test[J]. IEEE Transactions on Computer Aided Design of Integrated Circuits and

Systems, 2003, 22(6): 783-796

[8] Liang Huaguo, Jiang Cuiyun. Efficient test data compression and decompression based on alternation and run length codes[J]. Chinese Journal of Computers, 2004, 27(4): 548-554 (in Chinese)
(梁华国, 蒋翠云. 基于交替与连续长度码的有效测试数据压缩和解压[J]. 计算机学报, 2004, 27(4): 548-554)

[9] Han Yinhe, Li Xiaowei, Xu Yongjun, *et al*. Test resource partitioning using Variable-Tail code[J]. Acta Electronica Sinica, 2004, 32(8): 1346-1350(in Chinese)
(韩银和, 李晓维, 徐勇军, 等. 应用 Variable-Tail 编码压缩的测试资源划分方法[J]. 电子学报, 2004, 32(8): 1346-1350)

[10] Fang Jianping, Hao Yue, Liu Hongxia, *et al*. A hybrid run-length coding for SoC test data compression[J]. Acta Electronica Sinica, 2005, 33 (11): 1973-1977(in Chinese)
(方建平, 郝 跃, 刘红侠, 等. 应用混合游程编码的 SOC 测试数据压缩方法[J]. 电子学报, 2005, 33 (11): 1973-1977)

[11] Hu Bing, Chen Guangju, Xie Yongle. System-on-a-chip test data compression based on Huffman shift codes[J]. Chinese Journal of Scientific Instrument, 2005, 26(11): 1114-1118(in Chinese)
(胡 兵, 陈光禩, 谢永乐. 基于变移霍夫曼编码的 SOC 测试数据压缩[J]. 仪器仪表学报, 2005, 26(11): 1114-1118)

[12] Yu Yang, Peng Xiyuan, Zhang Yigang. Test data compression method for multiple scan chains based on repeated sub-vectors[J]. Chinese Journal of Scientific Instrument, 2009, 30(2): 356-361(in Chinese)
(俞 洋, 彭喜元, 张毅刚. 基于重复子向量的测试数据压缩算法[J]. 仪器仪表学报, 2009, 30(2): 356-361)

[13] El-Maleh A H. Test data compression for system-on-a-chip using extended frequency-directed run-length code[J]. IET Computers & Digital Techniques, 2008, 2(3): 155-163

[14] Zhan Wenfa, Liang Huaguo, Shi Feng, *et al*. A test data compression scheme based on mixed fixed and variable length coding[J]. Chinese Journal of Computers, 2008, 31(10): 1826-1834 (in Chinese)
(詹文法, 梁华国, 时 峰, 等. 混合定变长码的测试数据压缩方案[J]. 计算机学报, 2008, 31(10): 1826-1834)

[15] Zhan Wenfa, Liang Huaguo, Shi Feng, *et al*. A test data compression scheme based on mixed fixed and variable run-length coding in virtual block[J]. Acta Electronica Sinica, 2009, 37(8): 1837-1841(in Chinese)
(詹文法, 梁华国, 时 峰, 等. 一种混合定变长虚拟块游程编码的测试数据压缩方案[J]. 电子学报, 2009, 37(8): 1837-1841)

[16] Kuang Jishun, Zhou Yingbo, Cai Shuo. Adaptive EFDR coding method for test data compression[J]. Journal of Electronics & Information Technology, 2015, 37(10): 2529-2535(in Chinese)
(邝继顺, 周颖波, 蔡 烁. 一种用于测试数据压缩的自适应 EFDR 编码方法[J]. 电子与信息学报, 2015, 37(10): 2529-2535)

[17] Zhou Bin, Ye Yizheng, Li Zhaolin. BIST scheme based on two-dimensional test data compression[J]. Journal of Computer-Aided Design & Computer Graphics. 2009, 21(4): 481-486+492 (in Chinese)
(周 彬, 叶以正, 李兆麟. 基于二维测试数据压缩的 BIST 方案[J]. 计算机辅助设计与图形学学报, 2009, 21(4): 481-486+492)

[18] Zhan W F, El-Maleh A. A new scheme of test data compression based on equal-run-length coding(ERLC)[J]. Integration the VLSI Journal, 2012, 45 (1): 91-98

[19] Zhan W F, Liang H G, Jiang C Y, *et al*. A scheme of test data compression based on coding of even bits marking and selective output inversion[J]. Computers & Electrical Engineering, 2010, 36(5): 969-977

[20] Qian Xiaojie. 32-bits assembly language programming[M]. Beijing: China Machine Press, 2011: 218-226(in Chinese)
(钱晓捷. 32 位汇编语言程序设计[M]. 北京: 机械工业出版社, 2011: 218-226)