学校代号 10532　　学　　号 S1710W0830
分 类 号 TP391　　密　　级 普通

湖南大學
HUNAN UNIVERSITY

硕士学位论文

拆分压缩中基向量的生成算法研究

学位申请人姓名 夏　泽
培　养　单　位 信息科学与工程学院
导师姓名及职称 邝继顺 教授 文吉刚 高工
学　科　专　业 计算机科学与技术
研　究　方　向 SoC测试与设计
论 文 提 交 日 期 2020年5月15日

学校代号: 10532
学　　号: S1710W0830
密　　级: 普通

湖南大学硕士学位论文

拆分压缩中基向量的生成算法研究

学位申请人姓名: 夏　泽
导师姓名及职称: 邝继顺 教授 文吉刚 高工
培　养　单　位: 信息科学与工程学院
专　业　名　称: 计算机科学与技术
论文提交日期: 2020年5月15日
论文答辩日期: 2020年5月27日
答辩委员会主席: 徐成 教授

Research on Generation Algorithm of Base Vector in Split Compression

by

Ze Xia

B.E. (Hunan Agricultural University) 2017

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of engineering

in

Computer Science and Technology

in the

Graduate school

of

Hunan University

Supervisor

Professor Xinguo Lu

May, 2020

湖 南 大 学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名:　　签字日期:　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湖南大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

1、保密□，在____年解密后适用于本授权书

2、不保密□。

(请在以上相应方框内打"√")

作者签名:　　签字日期:　　年　　月　　日

导师签名:　　签字日期:　　年　　月　　日

摘　要

在20世纪90年代，一个较复杂的芯片包含几十万至几百万个晶体管，而现在一个芯片可以包含上百亿个晶体管。芯片日益增加的集成度和复杂度直接提高了生产故障芯片的概率。为了确保被测芯片的故障覆盖率达标，需要大量的测试数据进行检测。庞大的测试数据增加了硬件代价和测试应用时间，通过压缩测试数据能大大降低被测时间并节约硬件存储开销。本文在拆分压缩技术的基础上，对如何生成基向量展开了研究，主要做了以下三个方面的工作：

（1）提出了一种预填充的策略对测试集进行处理。此方法旨在消除测试集中的无关位，预先对测试集进行填充，填充的模式分为直接填充和策略填充两种，对于一个不包含无关位的测试集，本文以向量间距离最大最原则选取所需基向量。实验结果表明，使用预填充的方式，RL-Huff编码的压缩率可达74.32%，相比与对测试集进行直接编码，压缩率提高了11.75%，相比于哈达码矩阵变换，压缩率提高了2.47%。

（2）提出了一种基于kmeans++聚类算法结合测试集生成基向量的方法。该方法首先以距离最大为原则挑选出测试集初始聚类中心，然后计算测试集中的每一个列向量与聚类中心之间的欧几里得距离，以距离最小为原则对列向量重新划分聚类并计算出新的聚类中心，如此反复迭代，当聚类基本不再发生变化时，最终确定的聚类中心向量即为本文所求基向量。实验结果表明，使用聚类算法结合拆分压缩，RL-Huff编码的平均压缩率可达76.30%，与对测试集直接编码压缩相比平均压缩率提高了13.73%，与比哈达码变换相比，平均压缩率提高了4.45%。本人使用此方法对大电路进行了测试，在FDR编码编码方式下，此方法比对测试集直接压缩所获取的平均压缩率高6.06%。

（3）将kmeans++聚类算法结合位翻转算法进一步提高压缩率。此方法的主要思想是：先对原测试集进行预填充并利用kmeans++聚类算法生成当前测试集的主分量集，然后对主分量集进行故障模拟检测出部分故障。对于原测试集而言，可以将主分量集能检测出故障对应的确定位转化成为无关位，生成新测试集。最后对新测试集使用向量分解的方式进行压缩。结果表明使用位反转算法结合kmeans++算法可以将平均压缩率在（2）的基础上提高7%。

关键词：拆分压缩；基向量；主分量集；kmeans++算法；测试数据压缩

Abstract

In the 1990s, a more complex chip contained hundreds of thousands to millions of transistors, but now a chip can contain tens of billions of transistors. The increasing integration and complexity of the chip directly increases the number of circuit failures. To ensure that the fault coverage of the chip under test meets the standard, the test vector needs to be multiplied. The huge test data increases the hardware cost and test application time. Compressing the test data can greatly reduce the test time and save hardware storage overhead. Based on the split compression technology, this paper conducts research on how to generate basis vectors, and mainly does the following three aspects:

(1)Propose a pre-filled strategy to process the test set. This method aims to eliminate the irrelevant bits in the test set. The test set is filled in advance. The filling mode is divided into two types: direct filling and strategy filling. For a test set that does not contain irrelevant bits, this paper uses the maximum distance between vectors to select The required basis vector. The experimental results show that using the pre-filling method, the compression rate of RL-Huff encoding can reach 74.32%. Compared with direct encoding of the test set, the compression rate is increased by 11.75%. Compared with the Hadamard matrix transformation, the compression rate is increased. It was 2.47%.

(2)This paper proposes a method based on kmeans++ clustering algorithm combined with test set to generate basis vectors. This method first selects the initial clustering center of the test set based on the maximum distance, then calculates the Euclidean distance between each column vector in the test set and the clustering center, and re-divides the column vectors based on the minimum distance Class and calculate a new clustering center. After repeated iterations, when the clustering basically does not change, the final clustering center vector is the base vector obtained in this paper. The experimental results show that using clustering algorithm combined with split compression, the average compression rate of RL-Huff encoding can reach 76.30%, which is 13.73% higher than the direct compression compression of the test set, compared with the Bihad code , The average compression rate increased by 4.45%. I used this method to test large circuits. Under the FDR coding method, this method has an average compression rate that is 6.06% higher than that obtained by directly compressing the test set.

(3)Combining the kmeans++ clustering algorithm with the bit flip algorithm further improves the compression rate. The main idea of this method is: pre-fill the original test set

and use the kmeans++ clustering algorithm to generate the principal component set of the current test set, and then perform fault simulation on the principal component set to detect some faults. For the original test set, the determined bits corresponding to the faults that can be detected by the main component set can be converted into irrelevant bits to generate a new test set. Finally, the new test set is compressed using vector decomposition. The results show that using bit inversion algorithm combined with kmeans++ algorithm can increase the average compression rate by 7% on the basis of (2).

Key Words: Split compression; basis vector; principal component set; kmeans++ algorithm; test data compression

目　　录

插图索引

附表索引

第1章　绪论

1960年，仙童公司制造出第一块可以实际使用的单片集成电路。随着电子技术的继续发展，超大规模集成电路应运而生。1967年出现了大规模集成电路，集成度迅速提高；1977年超大规模集成电路面世，一个硅晶片中已经可以集成15万个以上的晶体管；1988年，16M DRAM问世，1平方厘米大小的硅片上集成有3500万个晶体管，标志着进入超大规模集成电路（VLSI）阶段；1997年，300MHz奔腾Ⅱ问世,采用0.25μm工艺，奔腾系列芯片的推出让计算机的发展如虎添翼，发展速度让人惊叹，至此，超大规模集成电路的发展又到了一个新的高度。2009年，intel酷睿i系列全新推出，创纪录采用了领先的32纳米工艺，并且下一代22纳米工艺正在研发。集成电路的集成度从小规模到大规模、再到超大规模的迅速发展，关键就在于集成电路的布图设计水平的迅速提高，集成电路的布图设计由此而日益复杂而精密。这些技术的发展，使得集成电路的发展进入了一个新的发展的里程碑。我国集成电路产业诞生于20世纪60年代,经历了以下几个发展阶段[1]:1965-1978年:以计算机和军工配套为目标，以逻辑电路为主要产品，初步建立集成电路工业基础及相关设备、仪器、材料的配套条件。1978-1990年:主要引进美国二手设备,改善集成电路装备水平,在“治散治乱”的同时，以消费类整机作为配套重点，较好地解决了彩电集成电路的国产化。1990-2000年:以908工程、909工程为重点，以CAD为突破口,抓好科技攻关和北方科研开发基地的建设，为信息产业服务,集成电路行业取得新的发展。2000-2005年:“十一五”期间，我国集成电路产业进入发展最快的历史阶段。芯片设计能力达到0.18μm,主流产品为0.35 0.8μm[2],芯片制造工艺水平达到12in 0.13pm,光刻机、离子注人机等关键设备取得重要突破。[3] 2006-2020年，国家信息化发展战略等重要文件中均将集成电路作为优先发展的重点领域。集成电路产业是电子信息产业的核心基础和战略性产业，建立较为完善的产业体系是实施信息产业强国战略的必然选择。以上种种迹象均表明：测试已成为迫切需要解决的问题，特别是进入深亚微米以及超高集成度发展阶段以来，通过集成各种IP核，系统级芯片SoC (System-on-Chip) 的功能更加强大，但也带来了一系列设计和测试的问题。例如，来自计算机、RF器件、消费电子产品和因特网基础设施市场的需求，迫使集成电路厂家必须提供完整的方案，同时解决测试系统在性能和测试效率方面的问题和挑战。

1.1 研究背景及研究意义

自1958年TI公司的JackS.Kilby和1959年仙童公司的Robert Noyce发明集成电路和硅平面集成电路以来，50年间，微电子和集成电路技术可谓发展神速，如同摩尔规律(Moore Law)所描述与预期的那样，按存储器算,集成度每18个月翻一番;就微处理器而言,集成度每两年翻一番;当前集成电路的集成度已从发明时的12个元件(2个晶体管、2个电容和8个电阻)发展到今天的数十亿个元件。集成电路功能日新月异，而成本迅速降低，微处理器上晶体管的价格每年平均下降约26%。通俗的来说，人们只要买得起报纸，就消费得起集成电路。正因为如此，集成电路已广泛渗透到国民经济、国家安全建设和人民生活的各个领域,其应用的深度和广度远远超过了其他技术，是当代信息社会发展的基石。它已如同细胞组成人体一样,成为现代工农业、国防装备和家庭耐用消费品不可分割的组成部分。

集成电路的生产，或多或少会存在故障随着时代的发展，芯片集成度以及复杂度的增加，故障出现的概率也随之增大。要保证生产出的芯片无缺陷，对我们而言是一个巨大的挑战。这其中不仅仅涉及到测试技术、测试装置，还涉及到电路和系统的设计、模拟和验证、制造等多个过程，我们将从几下几点来总结测试的复杂性和难点。

(1)速度、功能和性能更高的电路与系统要求与之匹配的自动动测试设备ATE(AutomaticeTest Equipment)，导致测试设备投资成本提高,测试成本随之提高。测试成为VLSI设计，测试和制造环节中费用和难度最大的一个环节。按照TRS(Internatinal Technology Roadmap Semiconductors)的研究[4],到2014年晶体管的测试成本要大于其制造成本。测试成本增加的因素主要归于两个:测试设备投资的提高和器件平均测试时间的增大。

(2)电路与系统的速度、性能和复杂程度的日益提高，导致测试数据量随之剧增，测试时间越来越长[5]，因而测试成本随之剧增。为了适应测试技术发展的需求，生产ATE的各公司不断推出性能更高的测试设备，例如，惠瑞捷(Verigy)公司推出Agilent93000系列测试仪[6],泰瑞达(Teradyne)推出Tiger系列测试仪[7],二者的每个测试引脚均配置处理器，可按需要灵活设置测试激励信号，以适应SoC测试的需要，但芯片的I/O数目有限，自动测试设备的通道量、吞吐能力和速度也有限，使得测试难度和复杂程度大大加剧[8,9]。测试时间成为SoC设计要考虑的重要因素[10]。(3)电路与系统的I/O、速度和测试时间的增加，测试功耗、ATE带宽等也成为重要影响因素，对器件的可靠性和测试质量提出更高的要求。VLSI测试功耗主要由两部分组成，一部分是内部功耗，一部分是I/O功耗。据研究，电路测试时内部功耗是正常功耗的2 4倍，分析和降低这部分测试功耗成为测试的一个研究热点[11,12]。即使对于低功耗设计的IC.要解决的问题典型的I/O功耗也占到总功耗

的50% 左右[13]，因此降低ATE测试数据线的转换次数也成为要解决的问题[14–16]。

(4)新产品竞争激烈程度的加剧及其存活周期的缩短，产品的上市时间(Time-1o-Marke)相对于开发周期变得越来越短，测试对产品的上市时间、随着技术的快速发展和市场竞争的加剧，产品市场寿命开发周期的影响将会越来越短，测试对产品的上市时间、开发周期的影响将会越来越大，测试开发时间已成为测试经济学研究的重要内容[17]。

综上所述，技术和经济的因素导致传统的模拟、验证和测试方法难以全面验证设计与产品制造的正确性，因此在设计和测试方面就应该有新的思想方法，设计出容易测试的电路。新的设计思想是在设计一开始就考虑测试问题，在设计前端就解决棘手的测试问题，即可测性设计DFT。可测性设计可以有效地解决或减轻复杂的测试问题，典型的DFT包括扫描/边界扫描设计和内建自测试。采用扫描/边界扫描结构，可通过少量的IO进行测试施加和测试响应分析，突出问题是扫描电路的附加面积、扫描深度、测试时间和测试功耗，而且因抗随机图形故障导致测试图形相当长。另外，伪随机测试中常用的是固定故障模型，对于CMOS深亚微米技术中的缺陷，还需要延迟、桥接、恒定开路等故障模型。基于扫描路径的测试和内建自测试面临的问题还有:测试时不断变化的位码使得测试功耗大大增加，既影响测试质量，又对电路的寿命有影响[18]。总而言之，测试研究的目的就是力图在预期的测试质量前提下，以尽可能低的成本对产品进行测试。

1.2 集成电路测试

1.2.1 测试

测试的目的是检查电路设计和制造的正确与否。为此，需建立一套规范的术语和检查分析方法，这也是电路测试研究的内容之一。

测试电路的一般过程是，先建立描述电路“好”或“坏”的模型，然后设计出能检验电路“好”或“坏”的测试数据，再把设计好的数据加在被检验的电路上；观察被检验电路的输出结果；最后分析与理想的结果是否一致。

被测试的电路称为被测电路CUT(Circuit Under Test)；对被测电路产生测试数据的方法和过程叫做测试生成(Test Generation)；产生的测试数据则叫做测试图形(Test Pattern,TP)；把测试图形施加到CUT的过程叫做测试施加(Test Application)；测试图形施加后被测电路的输出成为测试响应(Test Response)；检查电路实际的测试响应与理想的测试响应是否一致的过程叫做测试分析(Test Response Analysis)。那么，电路的测试过程用专业术语表达就是:先电路建模，然后测试图形生成，再测试施加;接着测试响应分析；最后得出CUT测试通过与否。如图1.1所示表达了电路测试分析的整个过程。

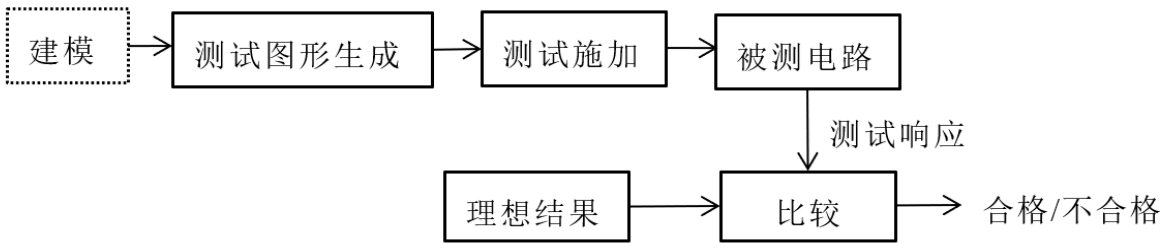


图 1.1 电路测试分析过程

1.2.2 生产测试

生产测试是对所有制造的芯片进行故障测试和随机缺陷测试，以确定制造的产品是否存在缺陷，也称为产品测试。生产测试必须考虑故障覆盖率、测试时间、测试设备的能力等因素来制定测试方案，以合理的成本测试芯片上每一个元器件及其连接。生产测试可分为裸片测试(或探针测试)和封装后测试。

(1)裸片测试(或探针测试):未封装前进行探针测试，测试比较简单，也可以避免不必要的封装费用。典型的探针测试结构如图1.3 (a) 所示，探针与晶片的接触如图1.3 (b)所示，测试时探针尖与环氧接触，避免划伤芯片。探针测试要解决的一个关键问题是如何放置探针，使其更好地贴近I/O引脚，随着I/O引脚间间距的日趋缩小，这个难度越来越大。

(2)封装后测试: IC封装后的测试主要有接触测试(Contact test)、老化测试(Burn-Intest)、电参数测试(Electrical parametric test)和功能测试(Functional test)。①接触测试:目的是找出与装配相关的错误，确定测试仪的引脚与芯片的I/O引脚正确连接。②老化测试: IC在其寿命期间可能失效，失效率可通过浴盆(bathtub) 曲线表示。新产品开始使用的前20周内，那些产品测试未检测出来的缺陷会导致许多芯片失效:随后的10 20 年内，产品失效率会保持在一个相对稳定的水平， 直至由于产品的超极限使用失效率呈指数规律增加。造成器件老化失效的主要因素是过多的热耗，以及电路内部噪声等。

1.2.3 功能测试

功能测试有多种分类方法。测试涉及到测试生成、测试施加和测试分析几个过程，因此测试也可按这些过程来分类。

按测试生成的方法，测试可分为穷举测试(exhaustive test)、伪穷举测试(pseudo exhaustivetest)、伪随机测试(pseudorandom test):和确定性测试(deterministic test)。

按测试施加的方式，测试可分为片外测试和片上测试。

按照测试图形施加的时间，测试可分为离线测试(of-line test)和在线测试(on-line test)。

在相关文献[19]中，总结了各类测试方法的特征以及相关术语。

(1)穷举测试。如果测试图形包含了原始输入所有可能的排列组合，这类测试图形称为穷举的测试集，把穷举的测试集施加到被测电路的方法就称为穷举测试。穷举测试优点在测试图形生成容易并有100%的故障覆盖率，但这样的方法只对小规模的纯组合电路有效。例如，对于一个原始输入为20的电路来讲，用1MHz的测试仪器来测试，需1s的时间[20]。对于时序电路来讲，穷举法就不适合，因为测试图形的时序对被测电路有非常大的影响。

(2)伪穷举测试。解决穷举测试时序问题的方法是伪穷举测试[21]，此方法用的也是穷举测试集，但测试矢量施加时时序上具有随机性。

(3)伪随机测试。伪随机的特征是测试图形的每位字都是随机的，是按照数学上的本原多项式采用线性反馈移位寄存器LFSR(Liner Feedback Shit Register)来产生的，这种方式产生的测试矢量所需成本最小。伪随机测试方法用故障模拟器来排列测试序列和计算测试覆盖率，固定故障的故障覆盖率可达85%以上。许多商用ATPG工具先用随机测试产生测试图形，然后用迭代法处理难测故障。纯随机测试中，一个测试图形可能出现不只一次，但伪随机测试中不会出现这样问题。伪随机测试也可用软件来生成。伪随机测试已成为VLSI嵌入式测试的主要方法。例如，内建自测试BIST和存储器测试中，一般都采用伪随机测试。

(4)确定性测试。确定性测试是基于故障的测试，是对特定的故障类型生成测试图形，一般确定性生成是按照算法，如D算法[22]、PODEM 算法[23]等，来完成测试生成。它是一个Np.complele 问题，需要迭代法来加速生成过程。确定性算法的优点是生成的测试图形非常短，难点是测试生成方法非常复杂，测试生成时间非常长。

(5)测试施加。把测试图形加到到被测电路的过程就是测试施加。测试图形施加到被测电路上时，他能用来检测故障，这是测试图形的价值所在。故障诊断时，测试图形可用来分析缺陷，电路正常工作时，测试图形可用于调试、修复子系统、可靠性分析。测试可对划片前的晶片施加，产品测试时有ATE和内建自测试施试方式，模拟时子EDA环境下施加测试图形。对于产品测试来说，测试施加成本是影响测试成本的主要因素。

1.2.4 测试生成

测试生成就是产生测试图形的过程，有软件和硬件实现的方法。软件一般用EDA工具或用户自己开发的程序，按照算法对给定的电路和故障模型生成测试图形，也称ATPG,生成的测试图形效率高。本书主要研究的是确定性测试的ATPG,目前也有门级的ATPG工具。硬件一般采用内建自测试BIST电路生成伪随机测试图形。

测试生成过程的复杂程度及所需要的时间主要取决于所采用的测试方法。在线测试不需要测试生成;基于线性反馈移位寄存器的伪随机法测试生成工作也很少;确定性测试、设计验证测试和诊断程序开发则需要大量测试生成。

实际的电路具有成千上万的门或者线，Y寻找测试图形的过程相当烦琐和费时，数学上可证明此过程是NP-complete.在商用的ATPG工具中，就采用迭代法。迭代的目的是在不影响优化效果的情况下，高效、更可行地解决某些问题。算法与迭代是都用来求解测试生成的。

1.2.5 可测性设计

测试图形生成的过程相当冗长复杂，于是可测性设计方法就得以发展，其核心思想是在设计一开始就考虑测试设计，在设计阶段就解决棘手的测试问题。可测性设计于20世纪70年代中期才形成，最先是扫描路径的提出[24]，IBM在80286设计中就采用此技术[25]。

可测性设计可分为两种方法。一种是专项技术，它采用传统的方法对电路某些部分进行迭代设计，以提高可测性。例如，时序电路测试前的初始化、设置观察点和控制点、电路的分块等。另一种方法是系统化技术，是从设计一开始就建立测试结构，每个子电路都具有嵌入式测试的特征。例如，内部扫描、内建自测试、边界扫描等。

可测试目前还没有确切的定义，Bennetts 于1984年给出的定义是:大一个数字IC，如果对其测试图形的生成、施加和分析是在预定的成本和时间内达到预定的效果，则称这个IC是可测试的[26]。这个定义有些含糊，不同的设计者会有不同的解释。例如，关键词“成本”，也许IC制造家为了减少测试成本就会减少这方面的可靠性开支。

可测性设计要求在电路中引入嵌入式测试的结构，使得测试更容易、效率高，也就是指测试生成和完成得容易高效。测试也成为设计优化的因素。原先设计优化是在速度、面积和功率三个因素中优化，现在需考虑第四个因素，即可测性。

1.3 国内外研究现状

数据压缩的主要目的是为了减少测试集的数据量，下图1.2给出了测试压缩结构图，由图中可以看出其中包括三个部分，第一个部分是ATE（测试设备），里面存储了压缩激励，第二个部分是CUT（被测电路），第三个部分是压缩的响应。测试过程大致为先将压缩的测试激励经过解压器，还原成为测试向量，然后将测试向量施加在被测电路上，最后将产生的响应输出到测试设备，若实际响应与压缩响应一致，则认为此块芯片是合格的。测试数据压缩包括测试激励压缩与测试响应压缩，根据不同的情况将采取不同的压缩方法，测试激励压缩属于无损压缩，因为要保证压缩之后的测试向量能准确无误地被还原。

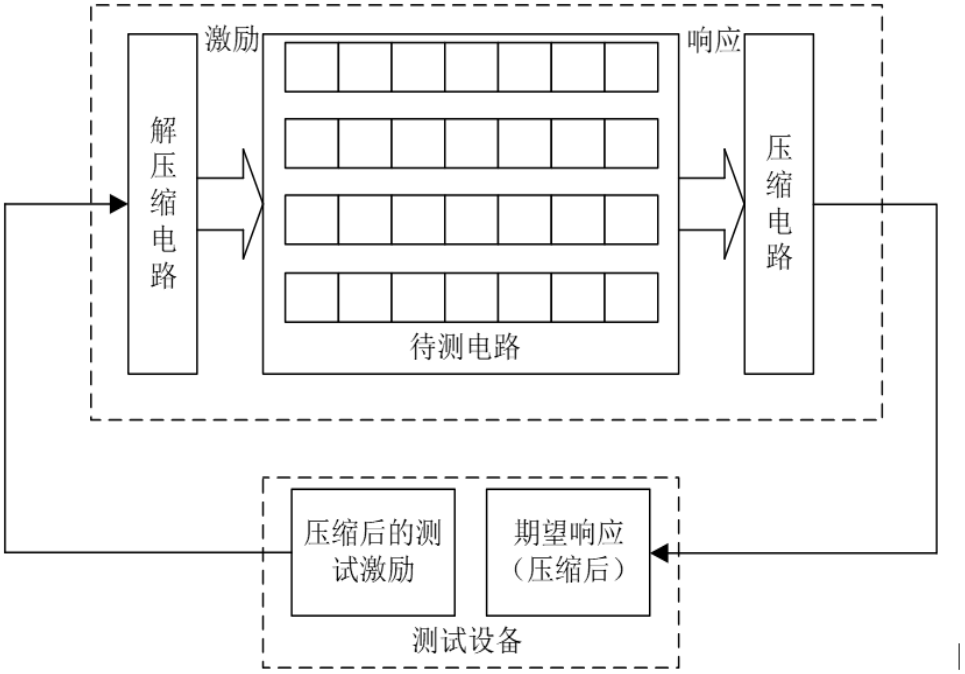


图 1.2 数字电路测试压缩结构图

测试激励压缩通常采用线性解压缩、编码压缩以及广播扫描压缩等方法。其中比较常用的编码有EFDR编码、FDR编码、Huffman编码以及Golomb编码。经过多年发展，近年提出了一些新的测试激励压缩方法，比如拆分压缩。在图像和音频信号的压缩中我们经常使用变换编码的方法，变换编码不是直接对原数据进行压缩，而是对图像或者声音进行采样、切成小块、变换到一个新的空间、量化，然后对量化值进行熵编码。即将一些能量较小的数据丢弃，将其转化成为易于压缩的数据，这类压缩方法属于有损压缩。

目前，测试激励压缩技术按测试数据存放的位置可以分为两类：内建自测试[27,28]和测试数据压缩。

(1)自建内测试[29]

内建自测试BIST就是在电路内部建立测试生成、施加、分析和测试控制结构。BIST方法可分为两类，一类是在线BIST,另一类是离线BIST。[30]在线BIST包括并发和非并发的方法，测试在电路的正常功能条件下进行，不把被测电路置于测试方式。采用并发在线BIST 方式时，测试与电路的正常操作同时进行，常用在编码和比较电路中。采用非并发在线BIST方式时，测试在电路的空闲状态进行，常用在故障诊断中，测试过程可以随时中断，电路的正常功能可以重新开始。

对于离线BIST,测试不在电路的正常功能条件下进行，可以应用在系统级、板级和芯片级测试，也可以用在制造、现场和操作级测试，但不能测试实时故障。离线BIST常采用片上测试图形生成器(Test-Pattern Generators, TPG)和输出响应分析器(Output Response Analyzers，OSA)。离线BIST又可分为功能性离线BIST和结构性离线BIST。功能性离线BIST 涉及的是基于被测电路功能描述的测试，常采用功能级或高级的故障模型，诊断软件常用这种测试方式。结构性离线BIST涉及的是基于被测电路结构描述的测试，通常测试生成和测试响应的压缩都采用一定形式的线性反馈移位寄存器[31,32]。

BIST所用的测试图形生成器(TPG)常见的形式有两种，一种是伪随机图形生成器(Pseudo random Pattern Generator,PRPG)它采用的是多输出线性反馈移位寄存器;另一种是移位寄存器图形生成器(Shift Register Pattern Generator,SRPG)，采用的是单输出的自动方式线性反馈移位寄存器。BIST[33]所用的输出响应分析器(OSA)常见的形式也有两种，一种是多输入特征分析寄存器，另一种是单输入特征分析寄存器，它们采用的都是线性反馈移位寄存器。

内建自测试电路一般包括测试生成电路（激励）、数据压缩电路、比较分析电路、理想结果存储电路（ROM）和测试控制电路，一般结构如下图1.3所示。

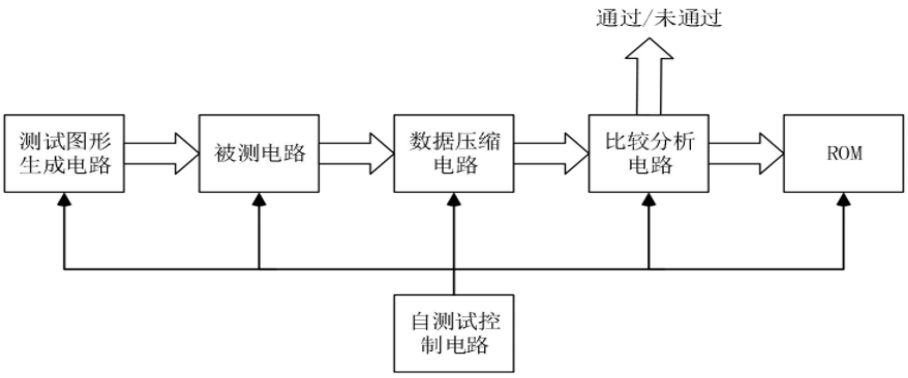


图 1.3 内建自测试一般结构

(2)测试数据压缩

把测试数据从芯片上全部或部分移出到离线的自动测试仪（ATE）上，利用

测试数据压缩技术降低ATE的存储成本和测试的时间成本。为了恢复原始测试数据，需要在芯片上设计解压电路，TRP框架如图1.4所示。

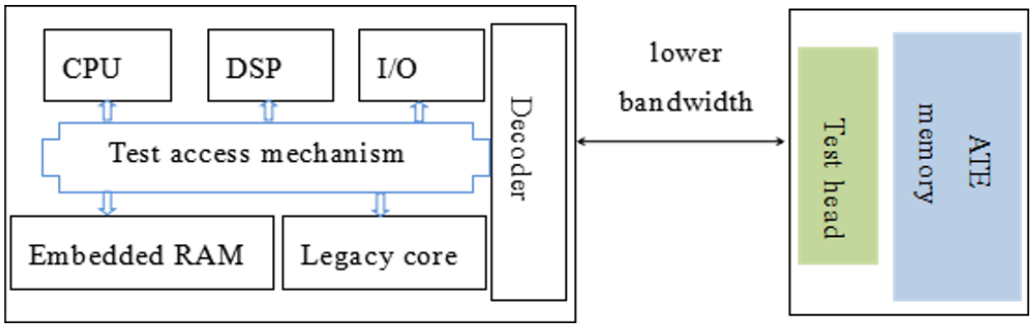


图 1.4 TRP框架图

绝大多数的编码压缩方案都是将测试数据压缩后全部存储在ATE上，测试时只要传送到芯片内部解码器中就能恢复出原始测试数据。基于测试集拆分的压缩方法把大部分压缩后的数据存储在ATE上，小部分拆分产生的控制信息存储在芯片内部，测试时由两部分共同还原出原始测试数据后再施加到扫描链中。

1.4 本文主要工作及组织结构

本文在拆压缩的基础上，主要针对基向量的生成方式进行了研究。由于主分量集是由基向量确定的，因此基向量选取的好坏将导致最终压缩率的高低。生成主分量集的方法有哈达码变换、K-L 变换、离散余弦变换等，本文提出两种新的基向量生成方法，一种是将测试集预填充后，以列向量间欧几里得距离最大为原则选取基向量，第二种是使用聚类算法生成实验所需的基向量。同时本文还将聚类算法与位翻转算法进行结合进一步提高压缩率。这三种方法针对压缩率都达到了较好效果。

本文共有五个章节，以下是各章节的内容概要：

第一章绪论，主要介绍研究背景及研究意义、集成电路测试的相关先修知识、国内外研究现状以及本文的主要工作。

第二章首先讲解测试的相关概念基础，接着介绍了随机测试技术以及常用的编码压缩技术，第一种是游程编码，第二种的字典编码，第三种是统计编码。虽然花了少量篇幅介绍了两种非编码压缩方法，在第二章的最后介绍了拆分压缩技术。

第三章充分阐述本文提出的使用预填充策略处理数据集的方法，包括填充规则、基向量的选取等步骤。

第四章详细介绍使用聚类算法提高压缩率的相关压缩方法，包括本文所使用聚类方法的介绍、基向量选取的规则、聚类算法的执行步骤等。

第五章仔细介绍kmeans++算法结合位翻转算法进一步提高压缩率的压缩方法。

最后结论，总结本文的主要工作，提出文中工作可以改进的方向并展望研究。

第2章　测试向量压缩

集成电路的发展日新月异，导致体积急剧减小的芯片上集成的晶体管数目却迅速增加。芯片复杂度的增加使得出现故障的概率增大，增加测试工作的困难。与此同时巨大的测试数据集不仅耗费测试功耗，更增长了测试时以及存储开销。数据压缩可以有效地解决上述问题。测试数据压缩分为激励压缩和响应压缩两种，结合本人研究方向，我着重介绍数据激励压缩。本章首先讲述电路可测性基础理论，然后就编码压缩方法以及非编码压缩方法以及拆分压缩做进一步阐述。

2.1　测试相关概念

2.1.1 故障检测的基本原理

电路测试中用故障来描述电路中的错误，而用测试图形来检测故障，现用单固定(SingleStuck-At, SSA)故障模型来阐述这一概念。单固定故障描述的是电路中一根线固定地接到逻辑值0或1这种现象，例如，这根线与地或VDD短路。

如下图2.1所示，在逻辑级可描述为输入a固定到逻辑值“0”，用s-a-0表示。输入a的s-a-0故障模型描述的是这样的失效机理: a与地短接时，不论何种逻辑信号加到与非门的输入端b.上，该门实现的功能都从z=ab变为z=1。

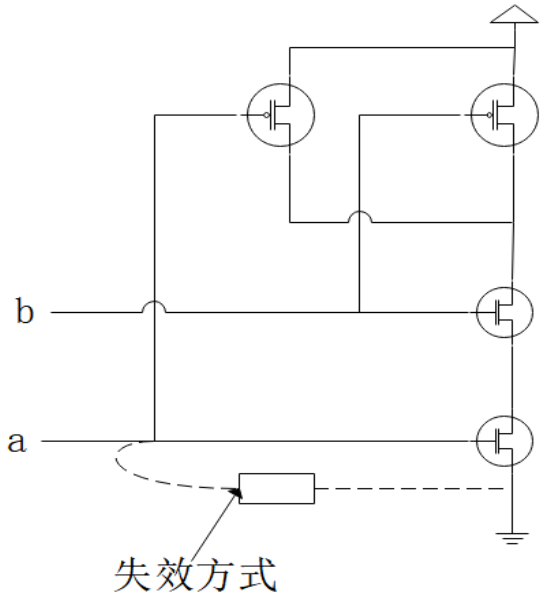


图 2.1 失效方式

下图2.2中的线a的s-a-0故障，可用a/0或a=0来简记:线a的s-a-1故障可用a/1或a=1来简记，类似地。线b的s-a-0 (s-a-1)故障用b/0或b=0 (b/1或者b=1)来简记，以此类推。我们把线a的s-a-0故障、a/0或a0这三种表达方法看做等同的表达方

法。

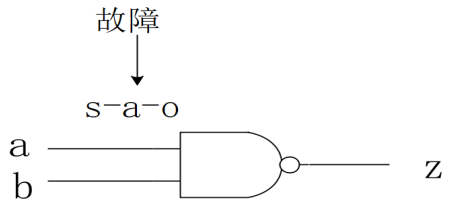


图 2.2 故障

故障检测就是对输入端施加信号，然后观察输出信号(输出响应)，再比较该输出响应和无故障时理想的输出响应，如果二者不同，则说检测到电路有故障。对于图2.1、2.2所示的电路，把输入、无故障时的输出响应FF和有故障的输出响应列表，得到表2.1。从此表可以看出，有故障时的输出响应并非总是与无故障时的输出响应不同，例如，故障b/1,只有当输入ab=10 时，无故障输出响应才与有故障时的响应不同，因此把ab=10称做故障b/1的测试图形。我们还注意到一一个故障可由多个测试图形检测到，例如，测试图形ab=00,01,10中的任意一个都可检测到故障z/0。另外一个情况是一个测试图形可用来检测多个故障，例如，测试图形ab=11可用来检测z/1、a/0 和b/0 故障，但这样的测试图形不能用来故障定位和故障诊断。

表 2.1 与非门的故障以及故障检测

输入	无故障时响应	有故障时响应				
ab		a/0	b/0	z/0	a/1	b/1
00	1	1	1	0	1	1
00	1	1	1	0	1	1
01	1	1	1	0	0	1
10	1	1	1	0	1	0
11	0	1	1	0	0	0

2.1.2 故障模型

要测试电路，先需建立故障模型。概括地讲，故障模型有两种:第一种为描述影响元器件间连接的故障模型，第二种为描述可能改变元器件真值表的故障模型。第种故障模型研究的前提是元器件无故障，只研究元器件间的互连问题，板级测试中主要研究这种故障模型，典型的模型有固定故障、一些开路故障和一些桥接故障。第二种故障模型与电路的制造工艺和版图结构有关，典型的有一些桥接故障和开路故障、恒定通故障和恒定开路故障等。集成电路测试中，不论是元器件内部还是它们之间的连接都必须检测。

我们根据故障的特点和在某种程度上的影响将其归类，称为故障模型（Fault

Model〉。常用的故障模型有：固定故障、桥接故障[34]、固定开路和固定短路故障[35,36]、延迟故障[37]。其中固定故障又可分为SSA型故障[38]和MSA型故障。

(1)SSA型故障

电路中某个门的根输入或输出线固定于逻辑1或0，这类缺陷用单固定故障模型描述，简称SSA故障,可用s-a-0故障描述，标识为h s-a-0、h/0或h=0。SSA故障即所谓的经典或标准故障模型，提出最早，研究和应用也最广泛。虽然SSA故障模型的合理性和代表性还有待理论论证，但可十分有效地检测出电路中存在的“错误”，SSA故障的这种有用性表现出以下属性:1、SSA故障模型表达了许多不同的失效方式。2、SSA故障模型是与工艺无关的故障模型。3、基于SSA故障的测试图形可以检测许多非经典性故障。4、SSA 故障模型的数目比其他类型的故障模型的数目要少，而且通过故障简化方法还可进一步减少此数目。

(2)MSA型故障

如果一根以上的线同时固定于逻辑1或0,这样的缺陷可用多重固定故障(Multiple Stuck-At fault)模型描述，简称MSA故障，“重”是指固定于同一逻辑值的线条的个数。随着器件对称性的降低和密度的增加，MSA故障出现的概率提高。电路中故障的数目随线条个数呈指数增加，N条线可能有2N个SSA故障，而N条线中m重线的可能数为:

$$C_N^m = \frac{N!}{(m!(N-m))} \tag{2.1}$$

则m重故障数为$2^m.C_N^m$，故N条线中可能存在的SSA故障的总个数为:

$$\sum_{m=1}^{N} 2^m C(N,m) = 3^N - 1 \tag{2.2}$$

影响MSA故障测试的主要因素是原始输入的个数和重聚的扇出点的个数。尽管可以用穷举或伪穷举测试来检测MSA故障，但是对VLSI不适用，且使用SSA故障的测试图形来检测MSA故障，得到的故障覆盖率偏高，这就会产生一个问题，测试及检测到会有多少MSA故障不能由SSA故障的测试机检测到? 答案取决于电路的结构而非电路的大小。就组合电路而言，一般有以下结论：

(1)对于冗余的两级电路，任意一个测试SSA故障的完全测试集可测试所有的MSA故障[39]。

(2)对于无扇出的电路，任意一个测试SSA故障的完全测试集可以测试所有的两重和三重故障[40]。

(3)对于内部无扇出的电路(即只有电路的原始输入才可能是根)，测试SSA故障的完全测试集可检测出至少98%的MSA故障，而且这样的测试集数目少

于6个[41]。

(4)对于无扇出的所有原始输入和电路C中所有扇出分支上的MSA故障，能够测试它们的完全测试集也能测试C中所有的MSA故障[42]。

(5)对于内部无扇出的电路C，任意一个测试SSA故障的完全测试集可以测试C中所有的MSA故障[43]。

2.1.3 故障模拟

故障模拟[44]是采用所谓的故障模拟器，对故障出现的设计模型施加测试集，进行模拟，然后分析有故障和无故障设计模型的响应，达到以下目的:

(1)测试给定的故障出现的条件;

(2)测试图形生成;

(3)衡量给定测试图形的效率;

(4)生成故障表。

故障模拟器需要的要素与其他模拟一样: 被测电路——设计模型，输入激励——测试图形，理想响应，待测故障——故障模型和故障列表，如图2.3所示。

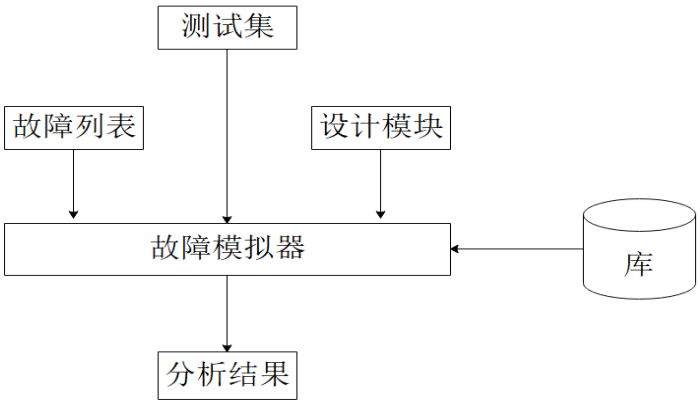
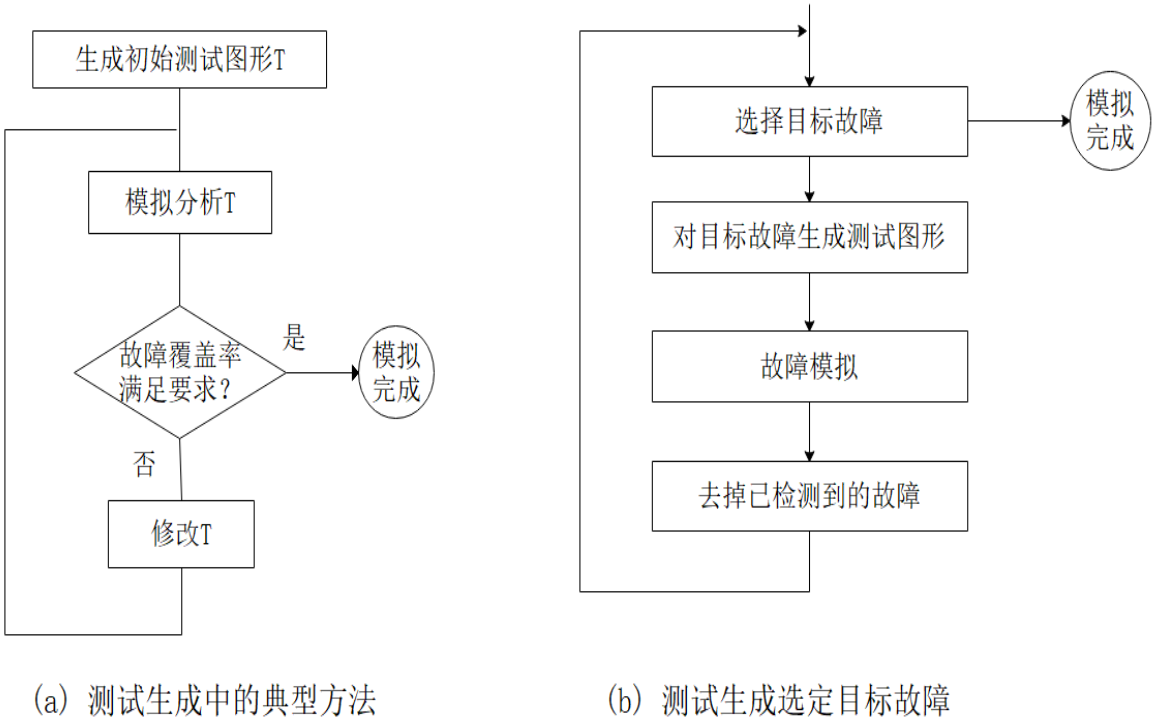


图 2.3 故障模拟要素

当给定故障测试生成时，故障模拟器从故障列表中选定一个故障，从测试集中选定输入激励加到设计模型上。如果模拟和测试分析表明有故障和无故障设计模型的响应不同，故障出现的条件也相符合，则选定的输入激励就是检测该故障的测试图形，保存测试图形并将该故障从故障列表中剔除，选择下一个故障，重新进行上述过程;否则换另一个输入激励，直到模拟和分析结果满意为止。如果选择测试集中所有的输入激励都不能测出某个故障，则称此故障为不可测，也从故障列表中剔除。重复这个过程直到故障列表变空为止。如图2.4所示为故障模拟器的流程图。



(a) 测试生成中的典型方法　　(b) 测试生成选定目标故障

图 2.4 故障模拟器的流程图

与逻辑模拟器一样，故障模拟器也分编译模拟器和事件驱动模拟器两类。故障模拟主要有3种方法:并行故障模拟(parallel fault simulation)、演绎故障模拟(deductive fault simulation)和并发故障模拟(concurrent fault simulation)。并行故障模拟常采用编译模拟器，而演绎故障模拟和并发故障模拟常采用事件驱动模拟器。

由于故障模拟时要分析有故障和无故障设计模型的响应，因此任何故障模拟都要包括一个无故障设计模型的模拟过程。

(1)并行故障模拟

并行故障模拟中[45]，故障注入与处理都是并行的。所谓的故障注入(fault insertion)是指选择要同时处理的故障子集，并给模拟程序构造一个识别要处理的故障子集的数据结构。模拟过程中，数据结构用于生成注入的故障的效应。

(2)故障演绎模拟

演绎故障模拟是Armstrong于1982年提出的一种故障模拟方法。演绎模拟只需一“遍”就可以找出所有的可测故障[46]。已证明，对少于500个门的电路，并行模拟比演绎模拟有效，对多于500个门的电路，演绎模拟更利于操作[47]。

(3)并发性故障模拟

并发性故障模拟方法综合了并行模拟和演绎模拟的优点[48–50]，比其他算法更快，而且使用动态内存分配。其模拟处理的方法如下所示:对设计模型施加输入激励后，如果故障可测,则故障在门电路的输出必可观，则在下一步模拟中按可观性进一步分析该故障的效应，否则舍去该故障，按可观性分析其他故障或故障效应。

2.2 随机测试技术

2.2.1 随机测试技术的概念

基于故障的确定性测试生成方法是指用专门的算法对给定的故障生成测试图形，优点是生成的测试图形长度短，但生成过程比较复杂，测试施加比较困难。由微处理器的测试软件算法或专用的片上测试电路可容易生成随机的或伪随机测试图形，通常还能达到比较高的故障覆盖率，因此在电路测试特别是内建自测试中得以广泛应用。随机测试和伪随机测试的缺点是一般要增加额外的硬件，此外有一些故障难以用随机方法检测得到，例如，原始输入、原始输出和扇出电路端出现的故障，此时需用确定方法来生成测试向量。随机测试和伪随机测试的关键问题是:要保证满足要求的故障覆盖率，测试图形的长度应该取多长:如何生成可靠又易于实现(不管软件实现还是硬件实现)的伪随机序列，另外还研究低功耗测试序列问题。

典型随机测试系统包括随机位序列产生电路、被测电路、参考电路或理想响应存储电路及输出响应比较器。随机测试的主要特征是把位独立的随机序列加到被测电路的输入.上，输出比较器把电路的实际响应与理想相应进行比较，若二者一致，则称被测电路无故障，否则称被测电路有故障，必要时进行相应的故障诊断。

2.2.2 故障检测率的估算

随机测试的一个关键问题是为了得到满足要求的故障覆盖率，测试图形的长度应该取多长，需引入故障检测率的概念。对于一固定型故障$f \equiv k(k \in \{0,1\})$，随机测试图形可检测得到的概率就是故障检测率，记为$p_d\{f \equiv k\}$, 该值是随机测试效率的度量，也是对特定故障检测的复杂程度的特征表达。

检测故障$f \equiv k$的概率应由两个并发事件的概率确定:故障在其源处的再现及故障传播到电路的一个输出端，分别对应由故障$f \equiv k$在其源处的再现概率$p_e\{f \equiv k\}$及故障在任一原始输出可观的概率$p_t\{f \equiv k\}$来估值。

2.2.3 测试图形长度的计算

数字电路测试中一个非常重要的参数是测试施加时间，一般而言，测试施加时间所路长度成正比。对于确定性测试方法，测试图形长度可由测试所有可能存在的故障所需的覆盖率来确定，满足覆盖率的测试长度一般不会超过200，当然，测试图形越长，故障覆盖越高。

对于随机测试，文献[51–53]研究了故障覆益率与随机测试图形长度的关系，对于一般的电路，典型函数关系为，故障覆盖率随测试图形长度的对数向100%方向

增加。对于一些特殊电路，如可编程电路，存在大量的抗随机测试图形故障(RandomPattern-Resistant fault, RPR故障)，故障覆盖率T与随机测试图形长度N可按下式近似表达：

$$T = 1 - e^{-T} = 1 - e^{(-\lg aN)} \times 100\% \tag{2.3}$$

其中对于每一给定的电路 λ 是常量。

2.3 编码压缩

编码压缩技术是指将某一种特定码字的编码用其他的码字表示，以减少实际码字的位数，从而达到“压缩”的效果，在超大规模集成电路测试过程中需要的测试集，也常常使用编码压缩的方法减少码字，对于不同的电路或者使用不同的压缩方法，达到的压缩率是不一样。由于测试集最终需要被误差错地还原，因此使用的是无损压缩。大致流程如下图2.5所示。

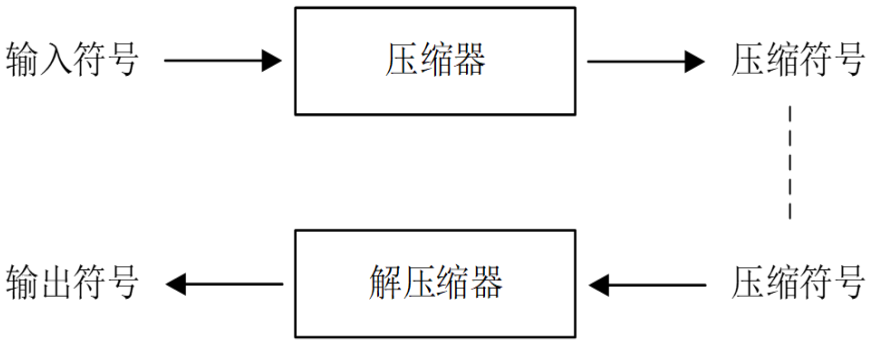


图 2.5 压缩通用流程图

2.3.1 游程编码

有很多测试数据压缩方法是基于游程编码的。游程是指连续相同的符号组序列[54]。Chandra提出了一系列游程编码方法，包括GoLomb编码[55]、FDR编码[56]、EFDR编码[57]、交替游程编码(ALT-FDR)[58]。FDR编码针对0游程，根据游程长度对码字进行分组，每组码字具有相同的前缀和不同的尾部，并且前缀和尾部的长度相等。在组Ai中，前缀的长度为i 位，每一组的前缀代表了这一组第一个编码的游程的长度。从组Ai到组Ai+1，前缀和尾部的长度分别增加了1位。长度为j的游程位于组Ai (i=[log2(j+3)]1)，组Ai 的大小为2i。表2.2列出了FDR编码前三组的

码表。例如测试向量T=000001 1 00000000001 0001，共有四个0游程，长度分别为5,0,10,3。FDR 编码后的序列为T(FDR)=1011 00 110100 1001，压缩了6位。

表 2.2 FDR编码表

组	游程长度	前缀	尾部	码字
A1	0	0	0	00
	1		1	01
A2	2	10	00	1000
	3		01	1001
	4		10	1010
	5		11	1011
A3	6	110	000	110000
	7		001	110001
	8		010	110010
	9		011	110011
	10		100	110100
	11		101	110101
	12		110	110110
	13		111	110111

EFDR编码作为典型的双游程编码，不仅可以使用较短的码字，对连续的0码字进行编码，对于连续的1码字也适用。因此当测试集的跳变数越少，使用EFDR编码所能达到的压缩率就越高。下表2.3给出了EFDR前三组编码。测试向量T=1111110 0000001 111111111111110 000001，共有四个游程，分别为1游程、0游程、0游程、1游程，长度分别为6,6,14,5。EFDR编码后的序列为T(EFDR)=11010 01010 1110111 01010，压缩了15位。

GoLomb编码是一种无损的数据压缩方法，由数学家Solomon W.Golomb在1960年代发明。Golomb编码只能对非负整数进行编码，符号表中的符号出现的概率符合几何分布(Geometric Distribution)时，使用Golomb编码可以取得最优效果，也就是说Golomb编码比较适合小的数字比大的数字出现概率比较高的编码。它使用较短的码长编码较小的数字，较长的码长编码较大的数字。表2.4列出了组长为4的Golomb码表。例如向量T=00000001000010000000010000001，0游程划分后序列为00000001 00001 0000000001 0000001，使用Golomb编码后序列为T(Golomb)=10111000110011010。原测试向量有30位，Golomb编码压缩后剩下17 位，压缩掉13 位。将测试立方中的不确定位填充成0，可以提高Golomb编码的压缩率。

表 2.3 EFDR编码表

组	游程长度	前缀	尾部	0油程	1油程
A1	1	0	0	000	100
	2		1	001	101
A2	3	10	00	01000	11000
	4		01	01001	11001
	5		10	01010	11010
	6		11	01011	11011
A3	7	110	000	0110000	1110000
	8		001	0110001	1110001
	9		010	0110010	1110010
	10		011	0110011	1110011
	11		100	0110100	1110100
	12		101	0110101	1110101
	13		110	0110110	1110110
	14		111	0110111	1110111

表 2.4 Golomb编码表

组	游程长度	前缀	尾部	码字
A1	0	0	00	000
	1		01	001
	2		10	010
	3		11	011
A2	4	10	00	1000
	5		01	1001
	6		10	1010
	7		11	1011
A3	8	110	00	11000
	9		01	11001
	10		10	11010
	11		11	11011

2.3.2 字典编码

字典编码就是一种随着数据流本身的特点动态地构建适合该数据流的编码与解码拆分压缩技术中的主分量生成方法研究表的压缩编码方法，在文本压缩、音频压缩中应用比较广泛。与统计编码相比，它既不使用统计的方法，也不使用变长码，而是选择许多字符串，利用字典的方法，对这些字符串进行编码压缩。字

典对字符串的保存可以是静态的，这称为非自适应LZ编码；也可以是动态的，这称为自适应LZ编码。

下图2.6是使用完全字典压缩方法的原理图，下图使用b条测试通道驱动n条扫描链，n往往大于b，那么每一条扫描链的长度会更短，通过这种方式会提高加载测试向量的效率，减少测试时间。此种方式也存在不足，完全字典，其中存储的数据量非常大，从而增大硬件开销，因此有些方法使用部分字典的方式，将不再字典中的数据进行纠错，该方式能控制字典的大小，在硬件代价与纠错成本之间进行权衡，从而达到想要的效果。

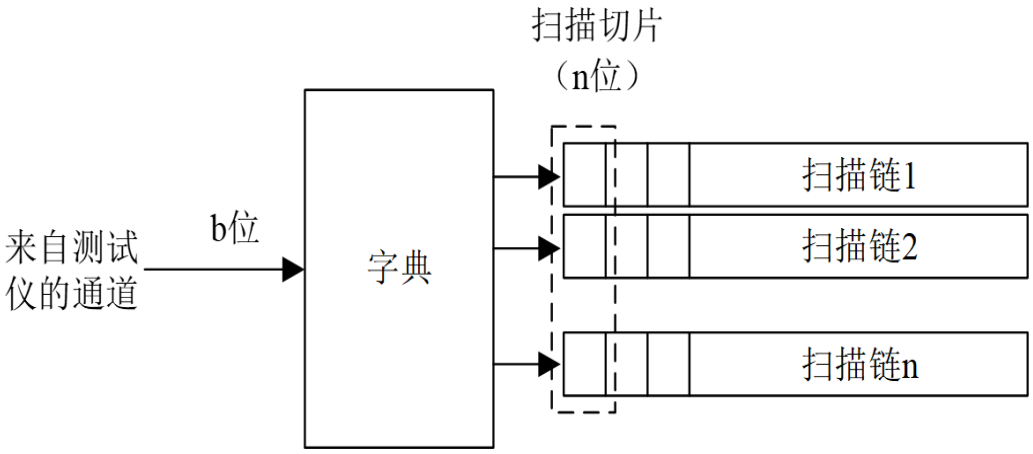


图 2.6 字典压缩方法

2.3.3 统计编码

统计编码是根据消息出现概率的分布特性而进行的压缩编码,它有别于预测编码和变换编码这种编码的宗旨在于,在消息和码字之间找到明确的一一对应关系,以便在恢复时能准确这种编码的宗旨在于,在消息和码字之间找到明确的一一对应关系,以便在恢复时能准确限制到可容忍的范围内。但不管什么途径,它们总是要使平均码长或码率压低到最低限度常用的编码有：Huffman编码[59]、Shannon－Famo编码、算术编码等。

Huffman编码是比较常用的统计编码方法之一，有些学者对其进一步改进，衍生出譬如最优选择哈夫曼编码(OP-SHC)[60]、多级哈夫曼编码[61,62]、变长到变长哈夫曼编码[63]等编码方式。Huffman编码的具体编码过程如下所示：(1)将信源符号按概率递减顺序排列。(2)把两个最小的概率加起来,作为新符号的概率；(3)重复步骤(1)、(2),直到概率和达到1为止；(4)在每次合并消息时，将被合并的消息赋以1和0或0和1；(5)寻找从每个信源符号到概率为1处的路径，记录下路径上的1和0；(6)对每个符号写出”1”、”0”序列（从码数的根到终节点）。

Huffman编码的有一些自生的特点，(1)哈夫曼方法构造出来的码不是唯一的,因为在给两个分支赋值时,可以是左支(或上支)为0,也可以是右支(或下支)为0,造成编码的不唯一。(2)哈夫曼编码对不同的信源的编码效率是不同的。当信源概率是2 的负幂时，哈夫曼编码的效率达到100%。当信源概率相等时，其编码效率最低。只有在概率分别恨不均匀时，哈夫曼编码才会产生显著的效果，信源分布均匀的情况一般不使用哈夫曼编码。(3)对信源进行哈夫曼编码之后会形成一个哈夫曼表，解码时必须参照这个哈夫曼表才能正确译码。(4)降低了编码的时间，改变了编码和解码的时间不对称性;(5)便于用硬件实现，编码和解码电路相对简单。这种方法适用于实时性要求较强的场合。虽然这种方法对某一个特定应用来说不一定最好，但从总体上说，只要哈夫曼编表基于大量概率统计，其编码效果是足够好的。

以下是我们的测试集数据，总共有80位，我们将其划分成为4位为一组的数据块，我们将每一个数据块出现的频率进行统计，如下表2.5所示，很明显表中1010、0000、1111三种数据块出现的频率最高。对这三组进行哈夫曼编码，编码的过程如下图2.7 所示。对于编码的快我们使用特殊码字表示，而为编码的快则需要在原有码字之前加上111，最后可得压缩率为38.8%

表 2.5 测试集的划分和各种块出现的频率)

测试集T	不同的块	频率
1010 0000 1010 1111	1010	9/20
1111 0000 1010 0001	0000	5/20
1010 0000 0010 1010	1111	3/20
0000 1010 1010 0000	0001	2/20
1010 1111 1010 0001	0010	1/20

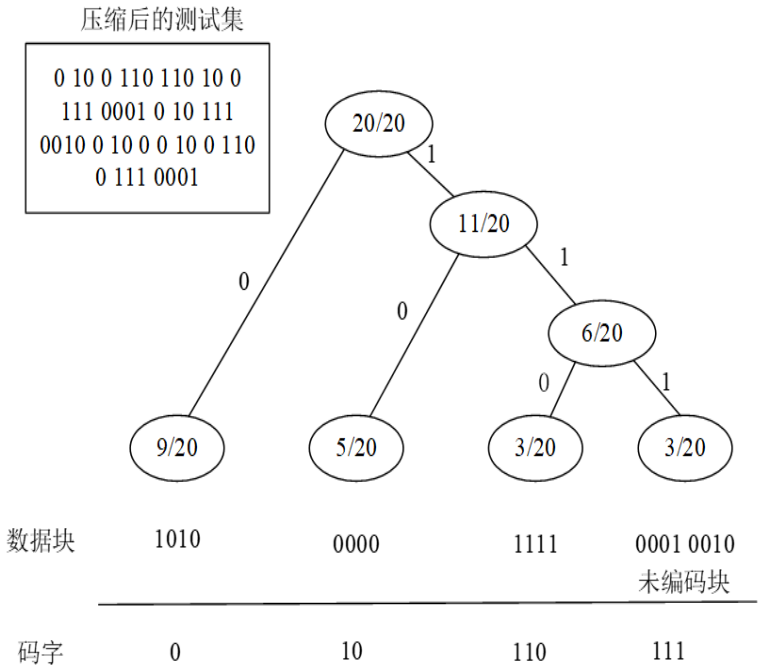


图 2.7 最优选择哈夫曼编码

除了上述本文介绍的几种编码外，常见的还有9C[64]、多层复制[65]、块合并[66]、BM-8C[67]、SVC[68]、选择扫描切片[69] 等编码。

2.4 非编码压缩

2.4.1 基于线性解压缩

线性解压缩器是一种包含了D触发器、RS触发器以及异或门的解压缩结构，下图是一种典型的时序线性解压缩器，由四部分组成，左边是代码测试仪有b个通道，经过线性反馈移位寄存器，到达了组合线性网络，他将LFSR中的数据扩展至右边的n条扫描链中，每一条扫描链中有m位数据。每当产生一个测试立方，需要输入b(q+m)位的外部数据。每当解压一个测试立方，LFSR便会被还原，重新初始化，等待时钟周期的到来继续将数据传送至扫描链。

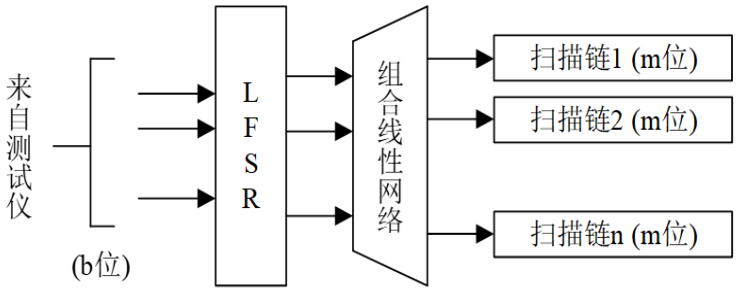


图 2.8 时序线性解压器

2.4.2 基于广播扫描

基于广播扫描的压缩方法本质上是一条测试通道驱动多条扫描链，也可以理解为把单个测试立方“广播”到多条扫描链上。Illinois扫描结构[70]是最先提出的广播扫描结构，如图2.9所示。

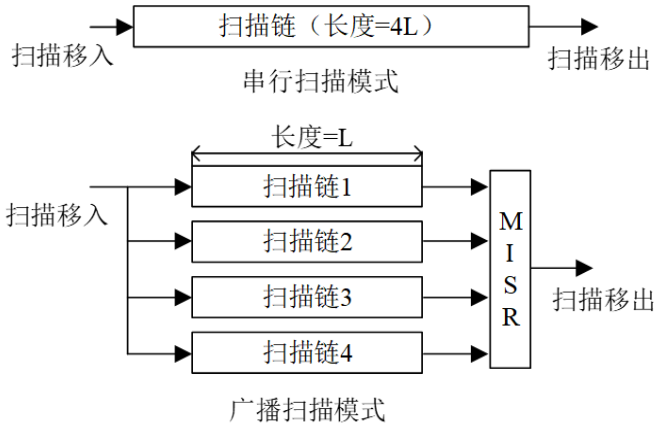


图 2.9 广播扫描结构

广播扫描模式较传统扫描模式不同点在于，他将扫描链分成若干片段的扫描链，上图的原始扫描链为4L，扫描链连接到同一输入，每一个子扫描链都为L，由于扫描链之间的值都相等，某些故障可能会遗漏。因此，需要与串行扫描模式相互配合来检测电路中存在的故障，从而提高故障覆盖率。

上文提及到，广播式扫描链因为其值相等会存在无法检测出某些故障的情

况，因此多输入扫描链的方式应运而生。其改进的措施为先将扫描链分为多组，每一组的输入端均不相同，并且保证组内相容从而保证每一组的故障覆盖率达到100%，还有一些学者，改进了扫描过程中重新配置广播扫描链的结构或者降低了测试过程中的功耗，均取得了不错的成果。

2.5 哈达码变换

拆分压缩技术是将测试集拆分为主分量集和残分量集，主分量集有很高的故障覆盖率，相对来说更加接近原测试集，残分量集则相反，但是残分量集更容易压缩。为了减少在ATE 上存储的数据量，提高压缩率，我们可以将主分量集存储在被测电路上，残分量集经过压缩后存储在ATE 上。拆分压缩的基本结构图如下图2.10所示。变换拆分这种方式之前被广泛应用于视频、音频等压缩场景，应用到测试集的压缩属于无损压缩，因此主分量集的选取和生成，对压缩率有着至关重要的影响，其次，由于必须确保硬件开销在一定范围内，就必须对被测电路上存储的主分量和最终能提升的压缩率之间做一个权衡。

2.5.1 拆分压缩

哈达码变换主要是基于拆分压缩技术，拆分压缩技术是将测试集拆分为主分量集和残分量集，主分量集有很高的故障覆盖率，相对来说更加接近原测试集，残分量集则相反，但是残分量集更容易压缩。为了减少在ATE 上存储的数据量，提高压缩率，我们可以将主分量集存储在被测电路上，残分量集经过压缩后存储在ATE 上。拆分压缩的基本结构图如下图2.10所示。变换拆分这种方式之前被广泛应用于视频、音频等压缩场景，应用到测试集的压缩属于无损压缩，因此主分量集的选取和生成，对压缩率有着至关重要的影响，其次，由于必须确保硬件开销在一定范围内，就必须对被测电路上存储的主分量和最终能提升的压缩率之间做一个权衡。

图 2.10 测试集拆分示意图

2.5.2 哈达码矩阵以及变换的基本流程

哈达码变换中使用的矩阵由-1和1组成，这种天然的特性非常适用于电路测试，1等价于1比特位，0等价于测试集中的-1。哈达码矩阵可以被递归定义：

$$H(k)=\begin{bmatrix} H(k-1) & H(k-1) \\ H(k-1) & -H(k-1) \end{bmatrix}, k=1,2,....,n, \tag{2.4}$$

其中H(0)=1,根据上述通式，当k=1以及k=2时有下列矩阵：

$$H(1)=\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.5}$$

$$H(2)=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{2.6}$$

哈达码矩阵的行列均是一个基本的向量，本文将其称作Walsh函数。以H(2)为例，它有4 个基本的向量分别是[1 1 -1 -1]，[1 -1 1 -1]，[1 -1 -1 1]和[1 1 1 1],。任何一个4 位的二进制序列都可以被这几个基本向量用线性组合的方式表示，比如向量[4 -4 -2 -4]可以被写为-1∗[2 2 2 2]+1∗[2 -2 2 -2] +1∗[2 2 -2 -2] +1∗[2 -2 -2 2]。基于此，电路的测试集均可以通过使用一组Walsh函数表示。此外，将哈达码矩阵取反也有相似的变换形式。

下面具体介绍一下如何使用哈达码矩阵进行拆分压缩的变换，首先根据原测试集的大小确定需要递归迭代的次数，然后对原测试集中的某一列用WHT 变换提取主分量，实质就是从哈达码矩阵中选取最合适的Walsh 函数来近似代替这一列。也就是说，通过将当前列向量与所有的Walsh 函数进行比较，选取差异最小的那一个Walsh 函数作为主分量。依次以原测试集中每一列为基准找出相对应的主分量构成主分量集，从上文分析可知，向量分解对单游程编码最有效，因为在变换中实验每次都是选取使得残分量中1 最少的Walsh 函数作为主分量。然而，对于双游程编码（ALT-FDR、EFDR、RL-Huff）、分块编码（OP-SHC）等方法而言，减少测试集中的1 并不是最优策略。

2.5.3 哈达码变换的优缺点

实验结果表明哈达码变换取得了较好的压缩收益，其主要优点有两个，第一哈达码矩阵通过递归调用之后生成全新矩阵，所涉及的硬件代价开销小。第二，通过递归最终生成的矩阵中，所有列向量两两之间均有差异，与原测试集中列向量匹配概率随之增大，最后得到残差集中0比特位也会更多。

哈达码矩阵中的列向量是随机的，是无特征的状态，有可能与原测试集中列向量差距较大，若哈达码矩阵中的列向量取自原测试集合，原则上来讲会取得更高的压缩增益，本文就是基于这一点展开的研究。

2.5.4 使用聚类思想构造主分量集合

哈达码矩阵中的列向量并非取自原测试集合，若对原测试集进行聚类，选取聚类中心作为基向量，生成主分量集合是否可以提高压缩增益。其中涉及到三个需要解决的问题，第一：原测试集合中存在较多的无关位，对于存在无关位的向量如何进行聚类。第二：原测试集的聚类中心无法确定，由于硬件存储开销，基向量的选取的个数需要符合一定的条件，不可能完全覆盖原测试集合，本实验中基向量选取的个数k=log2n其中n为原测试集的行数。第三：如何通过k列基向量生成大量的列向量，并产生具有更高压缩增益的主分量集合。下文将对三个问题逐一提供解决方案。

2.6 小结

本章详细阐述超大规模集成电路测试相关的概念、原理以及常用的压缩方法。首先本章对故障检测的基本原理、故障模型和故障模拟也进行了深一层的剖析。其次，重点介绍了随机测试技术。并使用较长的篇幅介绍了编码压缩，对FDR编码、EFDR编码、Huffman编码逐一举例分析其压缩原理，以及压缩特点。除了编码压缩，也提及了基于线性解压缩和基于广播扫描的非编码压缩方法，最后介绍了拆分压缩，该方法主要是将原测试集拆分为主分量集和残分量集，对残分量集压缩进行压缩进而达到较高的压缩率。

第3章　使用预填充策略处理数据集

本文主要是对测试向量进行压缩，测试向量是由一些0、1以及无关位所组成的向量，无关位可以任意填充为0或者1，对电路的测试并没有影响，只要确定位不变，就不会影响芯片的故障覆盖率，因此对无关位的填充至关重要，特别是对于编码压缩，填充的策略不同往往会对最终的压缩率产生较大的影响，本章主要就测试集如何填充进行相应的实验分析，通过合理地填充无关位，提高最终的压缩率。

3.1　变换拆分

变换拆分技术，就是将编码之数值经过一数学转换后映射至另一值域后再进行编码处理。常用于音频信号编码和图像或视频信号编码。变换编码经常与量化一起使用，进行有损数据压缩。本文所提及的变换拆分思想，是属于无损压缩。即在本实验中将原测试集拆分为主分量和残分量集，其中主分量集合存储在被测电路中，残分量并不是直接丢弃，而是将其压缩存储于自动测试仪，当需要将其施加在被测电路时，使用解压电路将残分量解压缩，然后与被测电路上的主分量进行异或，使其恢复成原测试集。

拆分压缩的结构图如下3.1所示，被测电路下方的数据我们称之为测试集，Vector1到Vectori是测试集对应的行称之为测试立方，而列向量我们称之为位流，通常对于测试集可以使用行变换或者列变换到达某种效果。在本文中默认是使用列变换，也可以叫做位流变换。

变换过程一般分为三个步骤：预处理、变换、构造残分量集。预处理阶段就是对位流的长度进行处理，可能需要对位流进行填充或者截断使其与基向量的长度保持一致。本文使用的是截断位流以匹配基向量。变换过程主要是通过线性反馈移位寄存器完成的，LFSR在原测试集中匹配到一个与原位流最相似的列向量（称之为主分量），然后与原为流进行异或生成残分量。构造参分量集便是将原测试与相应的主分量集进行异或即可。当需要对芯片施加测试时，先将存储在ATE中压缩的残差集取出，经过解压缩之后，与被测电路上的主分量进行异或，还原成原测试集。

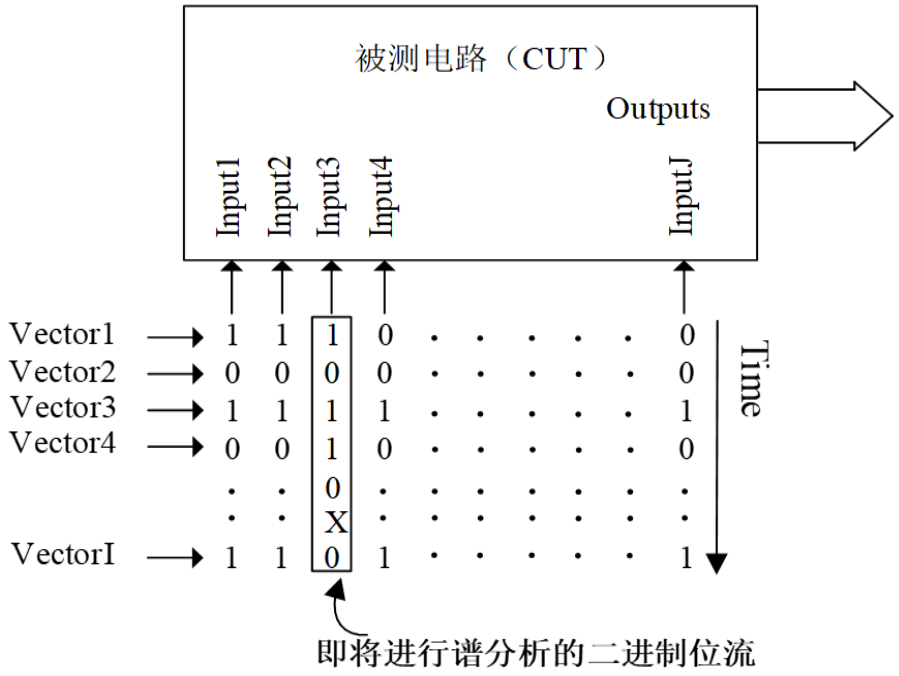


图 3.1 测试立方和位流

在图3.1中测试集有i个向量，对应了J个输入端，相当于每一个输入对应i位比特流。对于存在0、1和无关位的向量，如何等价为数学表达式来选取主分量是一个需要解决的问题，首先将0、1、和无关位分别用-1、1和0替代，使用这种方式之后，只需要简单地进行向量之间的乘法运算就能确定与其最相似的向量。例如对列向量（1, 0, 1, 1, 1, X, 0）进行谱分析，先要将列向量改写成（1, -1, 1, 1, -1, 0, -1）。然后通过线性反馈移位寄存器变换，得到一个系数向量，本例中为[2 -2 0 0 0 0 -2 5]。

系数向量中系数最大值对应的列向量就是所需的主分量，上图中为（1，-1，1，1，1，-1，-1），当系数向量中的最大值有多个时，可以任选一个对应的列向量作为所需的主分量，本例中选取的主分量当原测试集的无关位填充为0时，主分量和原测试及保持了一致，这是最好的情况，那么两者进行异或最终得到的残分量为（0，0，0，0，0，0，0），对于单游程编码，会产生极高的压缩率。若原测试集的列向量和线性反馈移位寄存器的矩阵大小不一致，则需要在位流尾部填

充无关位或者截断基向量即可。向量分解的伪代码如下算法3.1所示：

$$\begin{bmatrix} 1 & -1 & 1 & 1 & 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 0 & 0 & 0 & 0 & -2 & 5 \end{bmatrix} \tag{3.1}$$

实质上向量分解的目的是为了是参分量中0的个数更多，对于单游程编码而言目的就是要降低测试集中1的个数，对于无关位的填充，单游程编码会直接将其填充为0，拆分压缩技术对于双游程编码的提升效果相对较少，主要原因是影响压缩率的关键因素是跳变数，单纯的增加参分量中的0也不一定会提高残分量的压缩率。

算法 3.1 Kmeans

```
N : Number of vectors
M : Number of inputs
L : LFSR Matrix
T[1 : N, 1 : M] : Test set of dimensions N * M
Initialization(T) // 预处理测试集合
for i = 1 to M do
  p = L * T[1 : N, i] //测试集中每一列与位流相乘生成系数矩阵
  k = max(p) //在系数矩阵p中获取对应下表索引k
end for
Prominent = L[k] //生成主分量
Residual = T[1 : N, i] XOR Prominent //获取残分量集
ProminentComponentSet.add(Prominent)
ResidualComponentSet.add(Residual)
ReturnProminentComponentSet, ResidualComponentSet //返回所需数据
```

3.2 预填充测试集

3.2.1 直接填充与策略填充

对于不同的电路，如果采取编码压缩的方式，一旦编码方式得以确定，无关位具体是填充0或者1也随之确定，直接填充就是将不确定位直接填充成为0或者1，策略填充就是根据一定的策略进行填充，使得测试集在当前编码规则下的

压缩率更高。填充完毕之后，当前的测试集中就不存在无关位，对于一个完全确定的测试集，可以选取合适的聚类算法来选取基向量，最后生成主分量集。本章依旧使用拆分压缩，当无关位填充之后，采取随机选取基向量生成主分量集，进而计算出最终的压缩率。

在以往的研究过程中，并没有采取预填充的手段，对于主分量的选取，有的研究人员采用哈达码矩阵变化，有的采取最大相容类选取基向量最终确定主分量集合。哈达码矩阵最终的压缩效果不错但是由于哈达码矩阵是一种与原测试集无关的矩阵，因此所取得的压缩率还有提升的空间，而最大相容类一般使用KM算法或者贪婪算法和关系矩阵法相结合选取基向量，但是往往因为存在较多无关位，时间复杂度和算法复杂度都会很高，而且往往是第一个基向量能与很多列向量相容，继续往后选取，与其相容的会更少。基向量并不能涵盖所有的测试集列向量，通过基向量之间两两异或扩展出的其他列向量与原测试中向量越相似，压缩效果才会越好。

$$A=\begin{bmatrix} X & X & 1 & X & X & X & 0 & 0 & 0 \\ 0 & 0 & X & X & X & 1 & 1 & X & X \\ 1 & 0 & 0 & X & X & X & 1 & 1 & X \\ 1 & 0 & X & X & X & 1 & 1 & X & X \\ 1 & 1 & X & X & X & X & 1 & 0 & 1 \end{bmatrix} \tag{3.2}$$

上图给出原测试集，分别使用直接填充的方式与策略填充，获得最终所需的测试集。下图1是直接填充方式取得的测试集，采取的是0填充的方式，如果采取1填充，则将无关为全部填充为1即可。

$$A=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{3.3}$$

如若采取策略填充，则需要选择一个默认的编码格式，如果是单游程编码，

那么直接将无关位填充为0即可，如上图所示，如果是双游程编码，那么就需要使得原测试集的跳变数最少，以EFDR编码为例，填充之后的测试集如下所示：

$$A=\begin{bmatrix}1&1&1&0&0&0&0&0&0\\0&0&0&0&0&1&1&1&1\\1&0&0&1&1&1&1&1&1\\1&0&1&1&1&1&1&1&1\\1&1&1&1&1&1&1&0&1\end{bmatrix} \tag{3.4}$$

双游程编码需要在保证条变数最少的情况下，还必须要权衡连续出现0或者1的个数，比如条变数均为2，总共有18位的0、1和X有两种填充方式，一种是9个连续确定位为0，9个连续确定位为1，一种是12个连续确定位为0，6个连续确定位为1，那么应该选取后面的方式，因为连续的确定位越多获得的压缩效果会越好。

3.2.2 选取基向量

在本文中基向量的选取是至关重要的，主分量集是由基向量相互异或所得出的，基向量如果取自原测试集，基向量就和原测试集中一部分向量会保持一致，增加残差集中0的个数，本人在下面的章节中使用聚类算法选取基向量，旨在使选出的基向量能尽可能的与原测试中的向量相似，同时也对聚类算法进行了完善，使其基向量两两异或之后能产生更多的向量，以匹配原测试中已有的列向量，从而进一步提高压缩率。

本章提取取基向量的策略分三步进行，第一步随机从原测试集中选取一个列向量当作初始基向量a。第一步将向量a与原测试集的每一列向量分别计算两两之间的欧几里得距离dis。$dis=\sqrt{((x_1-a_1)^2+(x_2-a_2)^2+....+(x_i-a_i)^2)}$，其中x为某一个基向量，a为原测试集中的某一个列向量。下标即为对应的位，取其中最大的dis对应的列向量作为新的基向量。3、目前已经选取了两个基向量，接下来dis的计算方式为：

$$dis=\min a(\sqrt{(x_1-a_1)^2+(x_2-a_2)^2+....+(x_i-a_i)^2},....,\sqrt{(y_1-a_1)^2+(y_2-a_2)^2+....+(y_i-a_i)^2}) \tag{3.5}$$

当有多列基向量时，dis为当前列向量与众多基向量两两之间求的欧几里得距离的最小值。如此往复，最终计算出本电路所需的所有基向量值。对于电路不同

的规模，选取基向量的个数也是不一样的，在本章中基向量选取的个数X计算的方式为$X \geq ceil(\log_2^n a)$，其中n代表当前测试集的行数，由于需要将X列基向量存储在被测电路中，因此X不能取过大，取的过大会浪费存储空间。取X个列向量最终需要生成的是2的X次方个列向量，主分量集就是在这些列向量中选取的。上文提及的哈达码矩阵，在拆分压缩中取得了较好的压缩率，n 阶哈达码矩阵有2^n行、2^n列，其中n的值就是上述x的值，为了保证自变量一致，本文也是按照此规则选取基向量的个数。

3.3 实验结果与分析

为了验证预填充方法的有效性，我们对ISCAS’89[71]中大部分电路进行了实验，包括S5378、S9234、S13207、S15850、S38417、S38584等电路，本章将挑选其中的电路做具体描述，同时计算出在FDR、EFDR、Huffman等编码方式下对应的压缩率，并与其他方式所得压缩率做一个对比。

上文提及到本文选取基向量的个数为$X \geq ceil(\log_2^n a)$，那么对于X列的基向量，两两异或最终可得$2^x - 1$列向量的集合Y，主分量集即从原测试集中选出每一列，然后将其与集合Y中的列向量一一对比,选取出最相似的一列，构造出一个集合。实验结果如下表3.1所示，第一列为电路名称，第二列为当前电路的行数和列数，第三列是变换矩阵的个数，也就是x个列向量最终产生向量数的总和，第四列和第五列代表残差集所能达到的最小压缩率和最大压缩率。

表 3.1 随机矩阵与测试集变换结果

电路名	规模	变换矩阵个数	最小压缩率	最大压缩率
s5378	(111,214)	127	59.64%	70.78%
s9234	(159,247)	255	58.89%	70.12%
s13207	(236,700)	255	72.17%	93.74%
s15850	(126,611)	127	68.14%	83.30%
s38417	(99,1664)	127	63.03%	75.74%
s38584	(126,611)	127	68.14%	83.30%
平均			66.09%	78.05%

为了验证本实验是否对多种编码方式均适用，本文选取了FDR、EFDR、ALT-FDR以及Golomb等多种编码方式对变换拆分之后的残差集进行压缩，同时与使用哈达码变换达到的压缩率进行对比。上文提及本实验使用了直接填充与策略填充两种方式，直接填充1的效果相对较策略填充和全0填充效果差，策略填充和直接填充0作比较，直接填充0的效果甚至更好，主要原因猜想如下，第一：拆分压缩的目的是使得残差集中的0更多，如果直接将无关位填充位0，可以增加残差集中0的个数。第二：在选取基向量的过程中本文使用基向量之间两两距离最大

的原则选取，更增加了基向量异或之后所产生向量的多样性，从而获得了更多与原测试集中相似的向量，从而提高了压缩率。

下表3.2-3.6中，第一列代表电路名，第二列表示对电路直接使用编码压缩能取得的压缩率，第三列表示对原电路使用哈达码矩阵拆分压缩之后达到的压缩率，第四列为使用最大相容类提取主分量集所获取的压缩率，第五列为使用本方法所能达到的压缩率。以压缩率的提升为原则，本实验直接将无关位填充为0。

表3.2为FDR编码下各电路的压缩率，为了方便观察，本人将当前电路下取得的最高压缩加粗了。电路s13207在不会使用拆分压缩的方法下也能取得80%的压缩率，证明当前电路的无关位较多并且本方法在当前电路下可获得最高的压缩率，对于s38584电路本方法的压缩率略低于其他两种方法，其他电路所取得压缩率均高于其他两种方法，因此平均压缩率依然是最高的。

表 3.2 FDR编码压缩率(%)

电路	直接编码	哈达码变换	最大相容	本方法
s5378	47.98	67.51	67.86	68.43
s9234	43.61	66.19	65.71	66.39
s13207	81.31	89.65	89.92	91.69
s15850	66.21	80.66	80.73	81.88
s38417	43.27	72.44	72.84	73.04
s38584	60.93	75.99	75.34	75.09
平均	57.22	75.41	75.35	76.08

以下是EFDR和VIHC的压缩率，可以看出本方法比其他两种压缩方法，依然有提高压缩率，也可以从侧面说明，本方法不仅对单游程编码有效，对其他方式的编码也是有效果的。

表 3.3 EFDR编码压缩率(%)

电路	直接编码	哈达码变换	最大相容	本方法
s5378	53.67	64.50	64.52	65.60
s9234	48.66	62.74	62.62	62.71
s13207	82.49	88.89	88.03	90.95
s15850	68.66	78.67	78.83	78.88
s38417	62.02	71.63	71.76	71.71
s38584	64.28	73.45	72.37	74.69
平均	63.30	73.30	73.36	74.09

表 3.4 VIHC编码压缩率(%)

电路	直接编码	哈达码变换	最大相容	本方法
s5378	51.75	69.63	69.66	70.78
s9234	47.23	69.58	69.53	70.12
s13207	83.55	92.20	91.91	93.74
s15850	67.97	82.96	82.98	83.30
s38417	53.39	74.79	74.83	75.74
s38584	62.30	78.11	78.80	79.50
平均	61.03	77.89	77.83	78.86

表 3.5 RL-Huff编码压缩率(%)

电路	直接编码	哈达码变换	最大相容	本方法
s5378	52.58	64.02	64.14	66.73
s9234	47.26	60.33	60.16	63.16
s13207	82.49	88.71	88.81	92.23
s15850	67.35	77.33	77.99	79.47
s38417	63.32	69.66	69.64	71.43
s38584	62.40	71.05	72.28	72.91
平均	62.57	71.85	72.17	74.32

表 3.6 ALT-FDR编码压缩率(%)

电路	直接编码	哈达码变换	最大相容	本方法
s5378	49.95	61.62	62.02	63.54
s9234	44.96	58.31	58.16	58.89
s13207	80.23	86.52	86.61	90.16
s15850	65.83	75.76	75.89	76.78
s38417	60.55	68.40	67.51	68.13
s38584	61.13	69.70	71.83	72.09
平均	60.58	70.06	70.24	71.60

AFER和RL-Huff编码均是双游程编码，由上图可以看出RL-Huff编码和ALT-FDR编码比哈达码变换所取得的压缩率分别平均高出2.47%和1.54%，比最大相容所取得的压缩率高2.15%和1.36%。

从表3.2-3.6中的结果可知，本方法与直接编码、哈达码变换以及最大相容类选取主分量三种方法对比，取得的平均压缩率均比较高。

表3.7的数据展示了本方法与近几年的其他压缩方法做对比，本方法的压缩率为使用FDR编码所达到，其中的第一列为电路名，第二到第六列是近年提出的相关压缩方法。SVC是使用的是对称编码。L-EFDR属于双游程编码，对EFDR编

码进行了改进，提高了压缩率。LHBE 方法先对确定位进行划分，在进行长度折半编码，降低了编码的游程数量。CCCPRL也是一种基于相容数据快计数的游程编码方法。从实验结果可以看出，本方法的平均方法高达76.28%，与其他压缩方法相比较均有提高。结果表明对无关位预填充，同时在选取基向量时，按照距离跨度最大的方式选取可以获取较高的压缩增益。从表中可得不管何种压缩方式，s13207所取得的压缩率均是最高的，由此推断s13207电路中无关位会比较多，或者确定位较多的位流中有大段值相同的序列。通过观察其测试集，也验证了这一推论。

表 3.7 本方法与其他压缩方法压缩率比较(%)

电路名	SVC[72]	I-EFDR[73]	LHBE[74]	CCPRL[75]	本方法
s5378	51.80	55.10	53.10	61.08	68.43
s9234	50.94	52.73	52.33	62.95	66.39
s13207	83.77	83.82	83.87	90.06	91.69
s15850	69.98	71.05	70.78	76.32	80.88
s38417	63.30	64.57	64.10	64.61	73.04
s38584	66.26	66.70	66.60	75.38	77.06
平均	64.34	68.01	65.13	71.73	76.28

3.4 小结

为了降低电路的存储开销，提高数据压缩率，本章在变换拆分压缩技术的基础上提出预填充的方法预处理测试集，此方法有直接填充于策略填充两种模式，对于拆分压缩直接将无关位填充位0效果最佳。其次，在基向量的选取过程中并不是一味随机选取，而是随机选取第一列，然后根据欧几里得距离依次确定其他的基向量，基向量两两之间差距的增大导致异或之后产生的列向量更加丰富，从而提高了压缩率。实验结果表明，使用本方法在很大程度上提高了压缩率，大大减少了测试时间和测试数据的存储成本。

第4章　使用聚类算法提高压缩率

聚类算法是机器学习中涉及对数据进行分组的一种算法。在给定的数据集中，我们可以通过聚类算法将其分成一些不同的组。在理论上，相同的组的数据之间有相同的属性或者是特征，不同组数据之间的属性或者特征相差就会比较大。聚类算法是一种非监督学习算法，并且作为一种常用的数据分析算法在很多领域上得到应用。基于此，本文将聚类算法与数据压缩相结合，找到了一种提高压缩率的方法。第三章使用预填充测试集结合距离优先法则选取了基向量，最终获得了较好的压缩增益，但是基向量的选取始终是随机的，如果使用聚类算法将原测试集的列向量进行划分，以聚类中心作为基向量生成主分量集，在一定程度上能增大与原测试集的相似程度，提高残差集的压缩率。

4.1　聚类算法

4.1.1 DBSCAN聚类算法

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)，它是一种基于高密度连通区域的、基于密度的聚类算法，能够将具有足够高密度的区域划分为簇，并在具有噪声的数据中发现任意形状的簇，即要求在聚类空间中的一定区域内所包含对象(点或其他空间对象)的数目不小于某一给定阈值。过滤低密度区域，发现稠密度样本点。同一类别的样本，他们之间是紧密相连的，即在该类别任意样本周围不远处一定有同类别的样本存在。下面介绍DBSCAN算法的基本定义以及思路。假设当前有假设样本集是$D = (x_1, x_2, ...x_m)$

(1)-邻域：对于$x_j \in D$，其-邻域包含样本集D中与x_j的距离不大于的子样本集。

(2)核心对象：对于任一样本$x_j \in D$，如果其-邻域对应的子样本集个数至少包含MinPts个样本，则x_j是核心对象。

(3)密度直达：如果x_i位于x_j的-邻域中，且x_j是核心对象，则称x_i由x_j密度直达。

(4)密度可达：对于x_i和x_j,如果存在样本序列$p_1, p_2, ..., p_T$满足$p_1 = x_i$，$p_T = x_j$,且，由p_t密度直达，则称x_j由x_i密度可达。也就是说，密度可达满足传递性。此时序列中的传递样本$p_1, p_2, ..., p_T - 1$均为核心对象，因为只有核心对象才能使其他样本密度直达。

(5)密度相连：对于x_i和x_j，如果存在核心对象样本x_k，使x_i和x_j均由x_k密度可达，则称x_i和x_j密度相连。

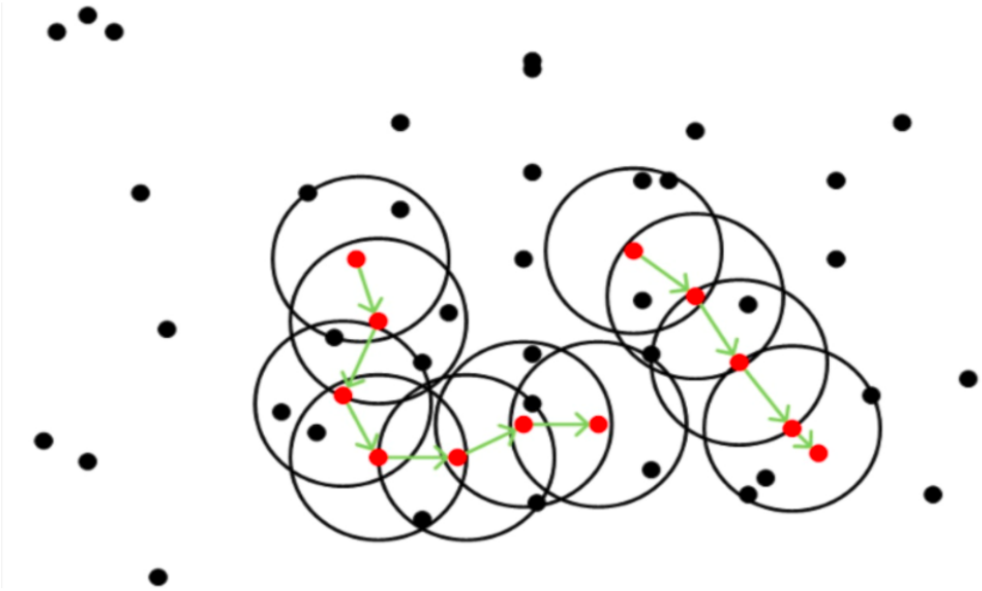


图 4.1 DBSCAN图示

从上图4.1可以很容易看出理解上述定义，图中MinPts=5，红色的点都是核心对象，因为其-邻域至少有5个样本。黑色的样本是非核心对象。所有核心对象密度直达的样本在以红色核心对象为中心的超球体内，如果不在超球体内，则不能密度直达。图中用绿色箭头连起来的核心对象组成了密度可达的样本序列。在这些密度可达的样本序列的-邻域内所有的样本相互都是密度相连的。

由密度可达关系导出的最大密度相连的样本集合，即为我们最终聚类的一个类别，或者说一个簇。这个DBSCAN的簇里面可以有一个或者多个核心对象。如果只有一个核心对象，则簇里其他的非核心对象样本都在这个核心对象的-邻域里;如果有多个核心对象，则簇里的任意一个核心对象的-邻域中一定有一个其他的核心对象，否则这两个核心对象无法密度可达。这些核心对象的-邻域里所有的样本的集合组成的一个DBSCAN聚类簇。

在此算法中有两个关键点，一个是某一样本的邻域距离阈值（即领域半径）一个是MinPts 某一样本的距离为的邻域中样本个数的阈值（即最少点个数）。将其与数据压缩相结合，即为列向量之间的欧几里得距离。原测试集中如果有大于等于MinPts个列向量与当前列向量之间的欧几里得距离小于，就可以确定当前列向量为一个中心点。根据电路的大小，需要选取的基向量数目为k，那么可以设计合理的和MinPts确定聚类中心即可。

4.1.2 kmeans聚类算法

Kmeans算法又叫做k-平均算法（英文：k-means clustering）源于信号处理中的一种向量量化方法，现在则更多地作为一种聚类分析方法流行于数据挖掘领域。k-平均聚类的目的是：把n个点划分到k个聚类中，使得每个点都属于离他最近的均值（此即聚类中心）对应的聚类，以之作为聚类的标准。

Means 是发现给定数据集的K 个簇的聚类算法, 之所以称之为K-均值是因为它可以发现K 个不同的簇, 且每个簇的中心采用簇中所含值的均值计算而成.簇个数K 是用户指定的, 每一个簇通过其质心（centroid）, 即簇中所有点的中心来描述。聚类与分类算法的最大区别在于, 分类的目标类别已知, 而聚类的目标类别是未知的。Kmeans中的专业术语如下所示：

簇：所有数据的点集合，簇中的对象是相似的。

质心：簇中所有点的中心(计算所有点的均值而来)。

SSE：Sum of Sqared Error（误差平方和）, 它被用来评估模型的好坏，SSE 值越小，表示越接近它们的质心. 聚类效果越好。由于对误差取了平方，因此更加注重那些远离中心的点（一般为边界点或离群点）。

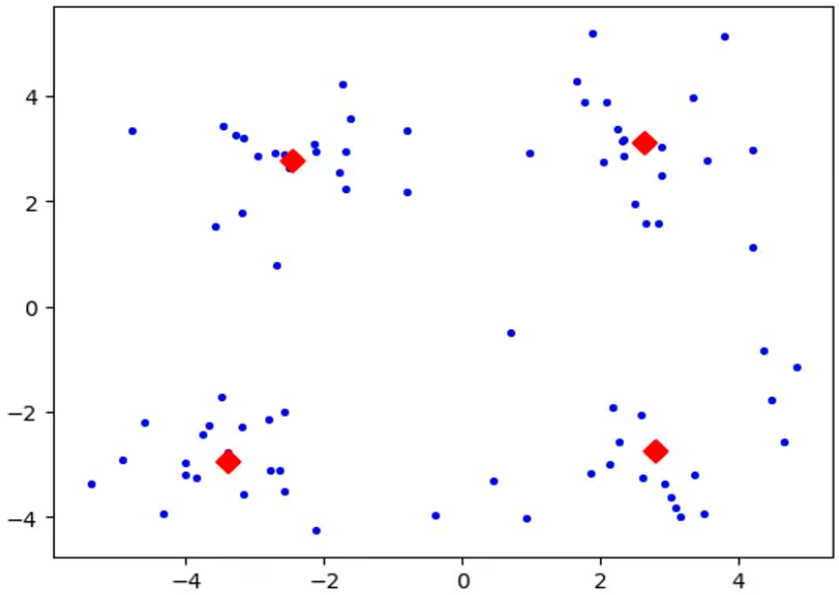


图 4.2 聚类分布状态图

从上图4.2中的数据分布可以看出，样本点大致分为四簇。所以设置初始聚类中心k=4，如果对数据不是很了解，可以先设置一个k，进行聚类之后，再根据结果的聚类效果来调整聚类中心的数量。上图可以看成二维坐标，其中每一个点都有相对应的x坐标和y坐标，通过欧几里得距离公式计算出SSE。Kmeans算法的计算流程如下所示：

(1)首先确定一个k值，即我们希望将数据集经过聚类得到k个集合。一般而言k值并不是事先所知晓的需要尝试多次。

(2)从数据集中随机选择k个数据点作为质心。

(3)对数据集中每一个点，计算其与每一个质心的距离（如欧式距离），离哪个质心近，就划分到那个质心所属的集合。

(4)把所有数据归好集合后，一共有k个集合。然后重新计算每个集合的质心。

(5)如果新计算出来的质心和原来的质心之间的距离小于某一个设置的阈值

（表示重新计算的质心的位置变化不大，趋于稳定，或者说收敛），便可以认为聚类已经达到期望的结果，算法终止。

(6)如果新质心和原质心距离变化很大，需要迭代3 5步骤。

通过上述的步骤基本上可以将一组数据划分成为k个聚类，如下我们将其转化为数学表达式，假设簇划分为$(C_1, C_2, C_3, ..., C_N)$,则本实验的目标是最小化平方误差E：

$$E = \sum_{i=1}^{n} \sum_{x \in C_i}^{n} \|x - u_i\|_2^2 \tag{4.1}$$

其中u_i为C_i的均值向量，有时也称为质心，表达式为：

$$u_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \tag{4.2}$$

为了进一步理解kmeans算法，下面以二维坐标系为例，讲述如何划分聚类。在二维坐标系中有6个点分别为p1(0,0)，p2(1,2)，p3(3，1)，p4(8，8),P5(9，10)，p6(10，7)，点与点之间的距离直接使用欧几里得距离公式计算，如果点并不是二维的坐标轴上，而是在三维、四维甚至n维，其距离计算公式为对应位两两相减求平方和之后开更根号。由于数据量较小，通过观察可得本数据可以划分为两个聚类，为了演示效果直接令k=2。初始时随机选两个点p1和p2，计算出剩余点与其距离如下表4.1所示：

表 4.1 测试集的划分和各种块出现的频率

坐标点	P1	P2
P3	3.16	2.24
P4	11.3	9.22
P5	13.5	11.3
P6	12.3	10.3

按照距离为原则划分两组中的数据为：组A中只有p1，组B中为其他点。第二步分别计算A组和B组的质心，A组质心为p1=(0,0),B组质心为pB=(6.2，5.6)，质心的计算公式为簇中各个向量相加取平均值即可。第三步，以质心为原点计算每一个点到质心的距离再次分组距离分布如下表4.2所示：

表 4.2 测试集的划分和各种块出现的频率)

坐标点	P1	P2
P2	2.24	6.3246
P3	3.16	5.6036
P4	11.3	3
P5	13.5	5.2154
P6	12.2	4.0497

第二次分组结果为组A：p1、p2、p3。组B：p4、p5、p6。再次计算质心轮询上面的步骤直至组A和组B中的点不发生变化，迭代结束。根据上述实验思路给出具体的算法流程伪代码如下所示：

算法 4.1 Kmeans

输入: 类簇个数K，迭代终止阈值ϕ，最大迭代次数MAX_COUNT
输出: 聚类结果
for $k = 0$ to K **do**
  $center_k = rand()\%COUNT$ //初始化K个类簇的聚类中心
end for
for $k = 0$ to MAX_COUNT **do**
  for $t = 0$ to T **do**
    $For\ every\ x_i$ //所有数据对象
    $Dis(x_i, center_k)$ //将x_i归于某一个聚簇中
  end for
  for $k = 0$ to K **do**
    $center_k = avg(cluster(k))$ // 将每一个聚簇的平均值作为新的聚类中心
  end for
  $Diff = new(cluster) - old(cluster)$
  if $Diff <= \phi$ **then**
    $return\ res$ // 如果新旧聚簇无明显变化则终止循环
  end if
end for
$return\ res$

Kmeans算法其优势是简单且易于实现，收敛速度快，对于结果密集型且区别明显的聚类簇分类效果好，但是其缺陷也非常明显，首先我们需要手动确定聚类数K，若原数据集并不能直接分为k个聚类有2k或者数据集较为分散没有明显特征，kmeans算法效果就会较差。其次kmeans对初始质心的选取依赖较大，不同的随机种子对结果影响非常明显。

基于此有学者提出了kmeans++算法，其改进主要针对Kmeans算法初始质心的随机选取。kmeans++在随机选取初始质心之前还进行了几个步骤（默认需要选取k个初始质心）。第一：从数据集中随机选取一个初始质心。第二：对于每个点，都计算其和最近的一个质心的距离D(x)并保存在一个数组里，然后把这些距离加起来得到Sum(D(x))。第三：取一个随机值，用权重的方式来取计算下一个质心。算法的实现为，先取一个能落在Sum(D(x))中的随机值Random，然后用Random -=

D(x)，直到其¡=0，此时的点就是下一个质心。第四：重复第(2)和第(3)步直到所有的K个种子点都被选出来。接下来按照kmeans算法的步骤执行即可。

4.2 聚类算法结合测试集选取基向量

4.2.1 聚类算法的选择

在实验的过程中本人尝试了多种聚类算法，包括基于马尔可夫模型的聚类算法、均值漂移聚类算法、凝聚层次聚类等，最终成功与实验相结合的只有上文提及的DBSCAN聚类算法和kmeans聚类算法。在本实验中，聚类对象为一个个向量，一个向量至少有100多位，距离的计算使用多维欧几里得距离。DBSCAN聚类算法虽然可以自动确定聚类个数，但是需要设置-邻域以及MinPts。本人经过多次尝试发现对于不同的电路，其-邻域以及MinPts均不相同，无法给出统一的标准，而且这两个值需要不断尝试才能获取较好的压缩增益，如果设置不当与随机选取无异。

Kmeans算法主要的缺陷有两点，第一无法确定数据集需要聚类的数目K，第二，只能随机选取初始质心。争对这两个缺陷本文给出了解决方案，首先本文根据电路的行来确定需要聚类的数目K，这样做有两点好处，第一对于不同的电路都适用，具有通用性。第二，会保证最终的聚类数不会过多，本实验最终要选择聚类中心充当基向量，若聚类数过多会导致基向量过多，增加硬件开销。针对第二个缺陷，本文结合kmeans++的思想，使用一种极大化质心(基向量)距离的选取方式，保证了初始质心(基向量)之间的距离足够远。通过上述两种解决方案，本实验使用kmeans++算法对已填充的原测试进行聚类。下文将介绍详细过程。

4.2.2 基向量的选取

基向量选取完成后，测试集相对应的主分量集也随之确定。本章主要根据kmeans++算法给出基向量相应的计算方法。

在初始基向量的选取过程中，需要进行测试集预填充，是第三章完成的工作。由于本人采取的压缩方法是拆分压缩，将原测试集中的无关位填充为0即可。下面是选取的具体步骤：

(1)从我们生成的“填充测试集”随机选择一个列向量作为第一个聚类中心。

(2)对于数据集中的每一个列向量x，计算它与最近聚类中心(指已选择的聚类中心)的距离D(x)。

(3)在原测试集中选择一个新的列向量作为下一个聚类中心，将其类比为数学公式为$D_x = \max a(D_1, D_2, ..., D_N)$，其中$D_1$表示测试集中第一个列向量与基向量之间的最小距离。同理D_N表示测试集中第N个列向量与基向量之间的最小距离，这

个步骤主要保证基向量之间的距离最大。

(4)重复2和3直到k个聚类中心被选择出来，其中k的数量由我们原测试集的数据规模决定。

(5)选取出来k个初始的聚类中心之后，计算每个列向量到聚类中心的距离，这里采用欧式距离计算。

(6)每个列向量均能计算出k个距离，我们将其划分在最小距离所对应的聚类中心的类中。

(7)对应类簇中所有数据对象的均值，即为更新后该类簇中心。

(8)判断中心点是否满足迭代条件，如果不满足则返回第二步重复计算。迭代条件可以设定为迭代之后类族的中心点不发生变化或者直接初始化迭代的次数。

算法 4.2 *PrincipalComponentSetGenerationl* (*T*)

输入: 初始K个基向量
输出: 主分量集
N : *Number of vectors*
M : *Number of columns*
$J[1:N, 1ceiling(log2N)]$: *Base vector matrix*
$T[1:N, 1:M]$: *Test set set size*
$Z = Xor(J)$ //将基向量两两间进行异或得到矩阵Z
$Z = [Z, -Z]$ //将矩阵Z取反与原来的Z组成新矩阵
for $i = 1$ to M **do**
  $d = Hamming(Z, T[1:N, i])$ //计算矩阵Z与原测试集中第i列的汉明距离
  $t = min(d)$ //求出汉明距离d中绝对值最小所对应的索引k
  $Principal = Z[t]$ //提取主分量
  PrincipalComponentSetadd(*Principal*)
end for
ReturnPrincipalComponentSet //返回主分量集

4.2.3 主分量集的获取

得到K个基向量后，将其依次进行相互异或，可得到一个列向量数为$2^n - 1$的矩阵，然后将这个矩阵取反且与原矩阵拼成一个新的矩阵，计算测试集中的列与新矩阵中的每一列的汉明距离，选取汉明距离最小时所对应新矩阵中的列作为主分量，构成主分量集，得到主分量集后，将它与原测试集进行异或，生成残分量集，最后对残分量集进一步压缩。主分量生成的具体过程如算法4.2所示。

例如测试集A的基向量为β1(1,0,1,0,1)、β2(1,1,0,0,1)、β3(0,1,1,1,1)组成，通

过基向量两两异或可生成矩阵Z，基向量为三列，对应的Z为7列。

$$Z = (\beta_1,\beta_2,\beta_4,\beta_1 \bigoplus \beta_2,\beta_1 \bigoplus \beta_4,\beta_2 \bigoplus \beta_4,\beta_1 \bigoplus \beta_2 \bigoplus \beta_4) = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

然后将Z取反与原矩阵拼接成为全新矩阵，取反即为0和1位置互换。

$$Z = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{4.4}$$

最终计算出测试集A中的每一列与矩阵Z_new每一列的汉明距离dis，将dis所对应中的列向量作为当前测试集中相应列的主分量。例如测试集中第二列β2(1,1,0,0,1) 与矩阵Z_new中第二列(1,1,0,0,1)的汉明距离为0，则原测试集第二列的主分量为(1,1,0,0,1)。以此类推，测试集每一列使用此运算将得到主分量集合。最终将主分量集合原测试集进行异或，计算出相应残差集，并运用编码压缩得到压缩率。

4.3 实验结果与分析

本实验使用kmeans++算法需要预先设定聚类数K，这在一定程度上影响了实验准确性。针对这一问题。实验结果分为两部分，第一部分根据电路大小选取基向量个数，并取得压缩率，同时与其他拆分压缩所取得的压缩率进行对比。第二部分，由于基向量需要存储代价，基向量的选取不能无限制的增加，本人在一定范围内动态增加或者减少基向量个数，观察压缩率的变化情况，大致绘制出基向量-压缩率之间的趋势变化图。

4.3.1 根据电路大小确定基向量数

根据电路大小选取基向量，其选取方式选取为$k = ceil(\log_2^N a)$其中N为测试集的行数。为了验证聚类算法的有效性，我们对ISCAS’89中大部分电路进行了实验，包括S5378、S9234、S13207、S15850、S38417、S38584等电路，本章将挑选其中的电路做具体描述，并与其他方式所得压缩率做一个对比。

实验结果如下表4.3-4.7所示，本文选取了FDR、EFDR、ALT-FDR、RL-Huff、VIHC以及Golomb六种编码方式对变换拆分之后的残差集进行压缩，同时与使用哈达码变换达、直接预填充方式获取的压缩率进行对比。

第一列为电路名称，第二列为测试集直接压缩之后的结果，第三列是对测试集哈达码变换拆分之后的压缩率，第四列为先对测试集预填充拆分压缩之后的压缩率，第五列是由kmeans++聚类算法结合拆分压缩所的压缩率。

表4.3是FDR 编码在各种情况下的压缩率比较，FDR 编码是一种单游程编码方式，其码表中对连续0子串进行编码，遇到1比特位直接使用00进行编码，当0串越长所对应的码字相对于原串越短，压缩率越高。由表中的数据可知，采用本章所提出方法计算的压缩率比测试集直接编码的平均高20%，比对原测试集进行哈达码变换相比，平均压缩率高百分之2.4，比预填充压压缩率平均高1.72%。

表 4.3 FDR编码压缩率(%)

电路	直接编码	哈达码变换	预填充	本方法
s5378	47.98	67.51	68.43	70.76
s9234	43.61	66.19	66.39	69.59
s13207	81.31	89.65	91.69	92.29
s15850	66.21	80.66	81.88	81.75
s38417	43.27	72.44	73.04	75.35
s38584	60.93	75.99	75.09	77.03
平均	57.22	75.41	76.08	77.80

表4.4是EFDR 编码在不同情况下压缩率的比较，EFDR 编码是一种双游程编码方式，属于对FDR 编码的一种扩展，既可以对0 游程进行编码，也可以对1进行编码，对测试集中0和1的数目并没有具体的要求，只需要保证跳变数最少即可，如果测试集中0和1 相继出现，压缩效果就会比较差，由表可知，采用本章所提出方法的压缩率比对测试集直接编码的压缩率平均高12.37%，比哈达码变换的压缩率平均高2.37%，比在预填充方法的压缩率平均高1.58%。

表 4.4 EFDR编码压缩率(%)

电路	直接编码	哈达码变换	预填充	本方法
s5378	53.67	64.50	65.60	67.75
s9234	48.66	62.74	62.71	66.14
s13207	82.49	88.89	90.95	91.60
s15850	68.66	78.67	78.88	79.84
s38417	62.02	71.63	71.71	74.06
s38584	64.28	73.45	74.69	74.60
平均	63.30	73.30	74.09	75.67

表4.5-4.6分别是是ALT-FDR 编码和RL-Huff 编码在各种情况下的压缩率比较。由表可知，采用本章所提出方法ALT-FDR 编码和RL-Huff 编码的压缩率分别比对测试集直接编码的压缩率平均高12.42%和13.73%，比哈达码变换的压缩率分别平均高3.24%和4.45%，比预填充策所达到的平均压缩率分别高1.7%和1.98%。虽然ALT-FDR 编码和RL-Huff 编码都是属于双游程编码，根据跳变数最少来进行编码，但是较其他压缩方法获得的相对压缩增益是最高的。

表 4.5 ALT-FDR编码压缩率(%)

电路	直接编码	哈达码变换	预填充	本方法
s5378	49.95	61.62	63.54	65.17
s9234	44.96	58.31	58.89	62.72
s13207	80.23	86.52	90.16	90.88
s15850	65.83	75.76	76.78	77.82
s38417	60.55	68.40	68.13	71.23
s38584	61.13	69.70	72.09	71.97
平均	60.58	70.06	71.60	73.30

表 4.6 RL-Huff编码压缩率(%)

电路	直接编码	哈达码变换	预填充	本方法
s5378	52.58	64.02	66.73	68.59
s9234	47.26	60.33	63.16	67.07
s13207	82.49	88.71	92.23	92.82
s15850	67.35	77.33	79.47	80.57
s38417	63.32	69.66	71.43	74.02
s38584	62.40	71.05	72.91	74.74
平均	62.57	71.85	74.32	76.30

表4.7是VIHC 编码在不同情况下压缩率的比较，VIHC 编码是一种单游程编码方式，采用“1”最少的原则进行拆分。由下表可知，采用本章所提出方法的压

缩率比对测试集直接编码的压缩率平均高19.25%，比哈达码变换的压缩率平均高2.39%，比预填充获取的平均压缩率高2.02%，用VIHC 编码也使压缩率平均达到了1.42%。

表 4.7 VIHC编码压缩率(%)

电路	直接编码	哈达码变换	预填充	本方法
s5378	51.75	69.63	70.78	72.81
s9234	47.23	69.58	70.12	73.25
s13207	83.55	92.20	93.74	94.20
s15850	67.97	82.96	83.30	84.28
s38417	53.39	74.79	75.74	77.86
s38584	62.30	78.11	79.50	79.25
平均	61.03	77.89	78.86	80.28

从上表中的结果可知，本方法与直接编码、哈达码变换、从预填充这三种方法相比，平均压缩率都要高。

为了验证本方法确实有效，本文除了对ISCAS’89中大部分电路进行了实验，还对b15、b17、b20、b21等大电路进行了相关实验。大电路中虽然拥有较多的无关位，但是电路所包含的比特位比较多有些甚至上百万位，上文实验中提及的电路大部分都是10万位以内。实验结果如下表4.8-4.9所示，第一列为电路名称，第二列为当前电路的行数和列数，第三列是对测试集直接编码的压缩率，第四列为本方法所达到的压缩率。

表4.8为各个大电路在FDR编码下的的压缩率，可以看出大电路中本身就存在较多的无关位，直接压缩也可取得较高放入压缩增益，但是使用本方法比直接编码的压缩率平均提高近6%

表 4.8 FDR编码压缩率(%)

电路	电路规模	直接编码	本方法
b15	1630*483	83.26	91.64
b17	2555*1452	89.93	94.10
b20	5510*522	84.15	90.54
b21	5496*522	84.13	90.63
b22	3369*767	85.10	89.94
平均		85.31	91.37

表4.9为各个大电路在EFDR、VIHC、ALT-FDR和RL-Huff下的压缩率，由表中数据可得当前压缩方法确实有利于提高压缩率，即使电路规模相对较大，也能获得较好的压缩增益。

表 4.9 大电路在直接编码和kmeans++算法下压缩率

电路	电路规模	直接压缩				本方法			
		EFDR	VIHC	AFDR	RL-Huff	EFDR	VIHC	AFDR	RL-Huff
b15	1630*483	85.99	86.72	85.12	87.01	91.08	92.91	90.61	92.04
b17	2555*1452	91.70	91.88	91.10	92.30	93.81	94.90	93.56	94.46
b20	5510*522	86.56	86.44	85.81	87.50	89.92	91.96	89.51	91.07
b21	5496*522	86.74	86.45	86.04	87.69	89.98	92.09	89.55	91.16
b22	3369*767	86.77	86.90	86.00	87.30	89.35	90.99	88.80	90.20
平均		87.55	87.68	86.81	88.36	90.83	92.57	90.41	91.79

4.3.2 动态选取基向量数

使用kmeans++算法并不能自动确定电路的聚类数，需要事先给定。虽然上文通过电路行数选取基向量获得了较好的压缩增益，并且在同等条件下比其他压缩方法取得的压缩率也更高，但并不能证明按照电路行数来选取聚类数会获得最高的压缩增益。基于此，本人在研究过程中，动态地选取了基向量数，进一步计算出确定基向量数与取得最终压缩率之间的关系。

本实验依然对ISCAS’89中大部分电路进行了实验，包括S5378、S9234、S13207、S15850、S38417、S38584电路，下表是当前算法在FDR编码方式下，压缩率的变化情况，第一列为电路名，第二列为原始压缩率，第三列到第七列为选取7到11个聚类数电路所能达到的压缩率。

表 4.10 FDR编码压缩率(%)

电路名	直接编码	7列	8列	9列	10列	11列
s5378	47.98	70.76	72.96	73.57	74.82	76.3
s9234	43.61	68.86	69.54	69.59	70.54	72.25
s13207	81.31	91.28	92.29	92.76	93.45	94.23
s15850	66.21	81.75	82.67	82.69	84.24	84.88
s38417	43.27	75.35	77.54	76.03	76.9	77.36
s38584	60.93	76.09	77.03	78.05	79.05	80.15
平均	57.22	77.35	78.67	78.78	79.83	80.86

根据上表4.10我们绘制出如下的折线图4.3，从图中可以看出随着基向量数的增加，压缩率整体呈上升趋势，整体来看当基向量数从7 列增加至8列时压缩率提高了1.32%，之后随基向量增加压缩增益的增加幅度稍微有所降低。

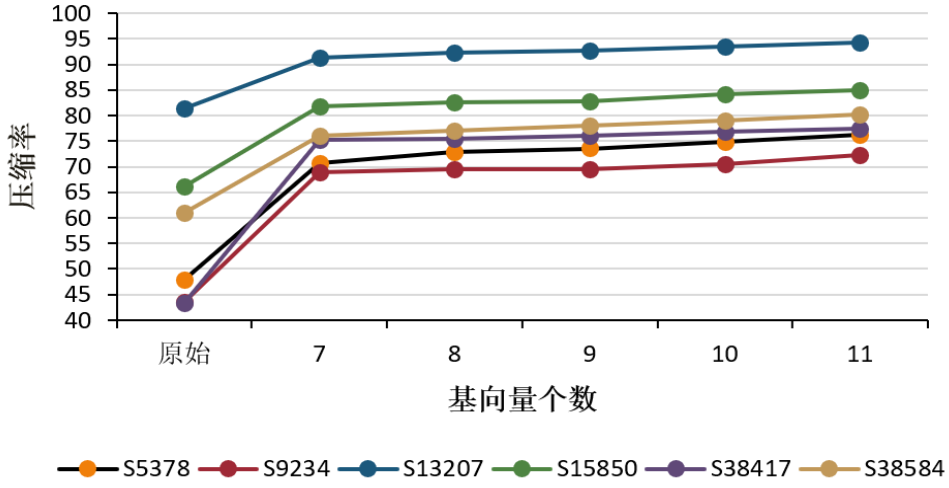


图 4.3 FDR编码方式折线图

下表4.11、图4.4是当前算法在VIHC编码方式下，压缩率的变化情况。

表 4.11 VIHC编码压缩率(%)

电路名	直接编码	7列	8列	9列	10列	11列
s5378	51.75	72.81	75.18	76.02	77.68	78.97
s9234	47.23	72.35	73.25	73.4	74.33	75.98
s13207	83.55	93.42	94.2	94.55	95.01	95.71
s15850	67.97	84.82	85.28	85.42	86.8	87.47
s38417	53.39	77.86	78.17	78.64	79.23	79.73
s38584	62.30	78.42	79.25	80.51	81.62	82.72
平均	61.03	79.95	80.89	81.42	82.445	83.43

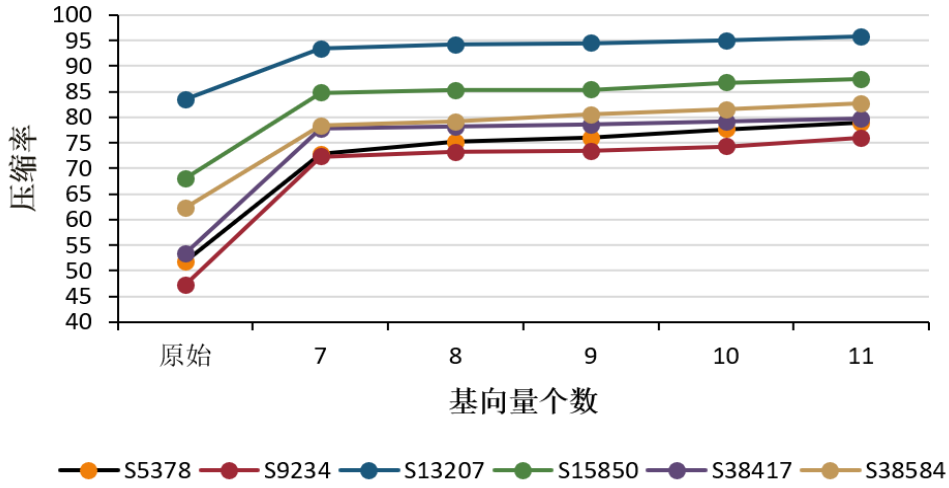


图 4.4 VIHC编码方式折线图

下表4.12-4.13以及图4.5、4.6分别为当前算法在RL_Huff、AFDR编码下压缩率的变化情况。

表 4.12 RL_Huff编码压缩率(%)

电路名	直接编码	7列	8列	9列	10列	11列
s5378	68.59	71.26	71.79	73.59	74.91	68.59
s9234	66.05	67.07	67.05	68.17	70.18	66.05
s13207	91.81	92.82	93.23	93.79	94.62	91.81
s15850	80.57	81.73	81.77	83.65	84.34	80.57
s38417	74.02	74.28	74.71	75.57	76.17	74.02
s38584	73.7	74.74	76.13	77.42	78.78	73.7
平均	75.79	76.98	77.45	78.70	79.83	75.79

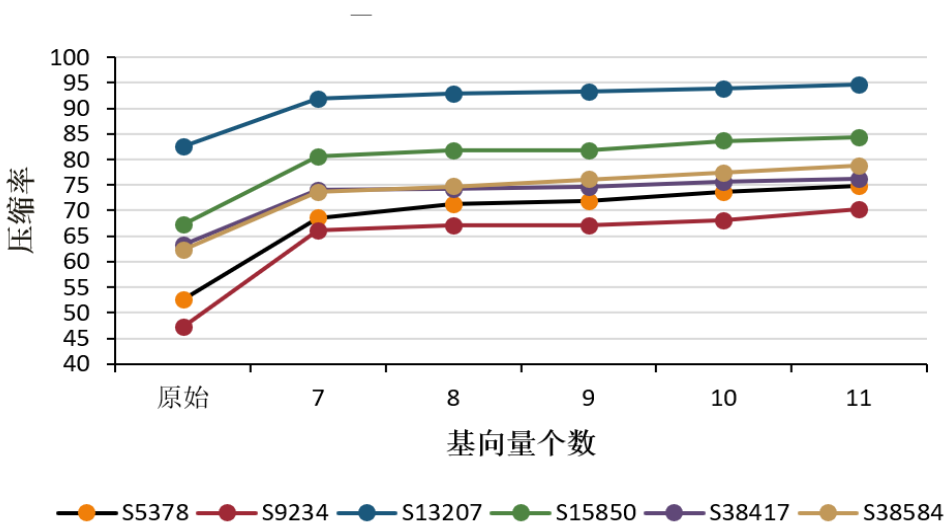


图 4.5 RL_Huff编码方式折线图

表 4.13 AFDR编码压缩率(%)

电路名	直接编码	7列	8列	9列	10列	11列
s5378	49.95	65.17	67.66	68.72	69.85	71.53
s9234	45.14	61.78	62.72	63.06	63.91	66.11
s13207	80.12	89.66	90.88	91.45	92.24	93.18
s15850	65.64	77.82	78.96	78.99	81.01	81.66
s38417	60.52	71.23	71.38	71.97	72.77	73.34
s38584	61.09	70.82	71.97	73.29	74.57	75.93
平均	60.41	72.75	73.93	74.58	75.73	76.96

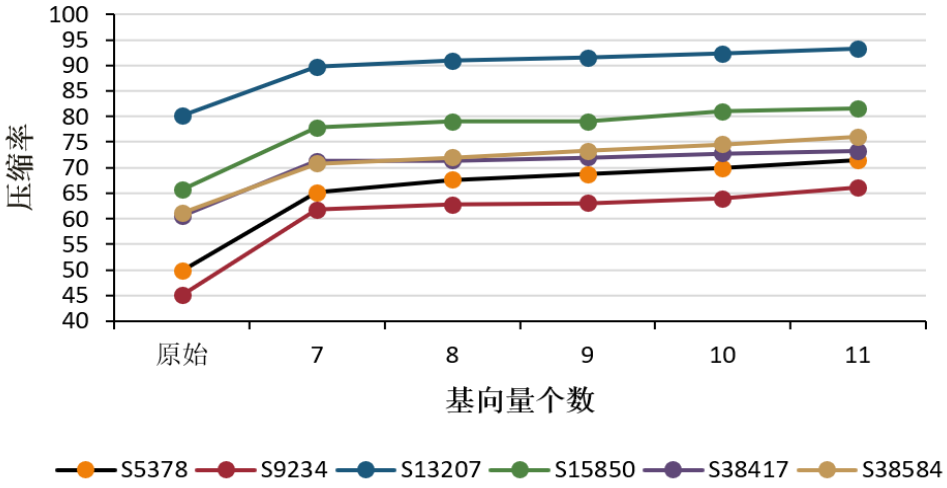


图 4.6 AFDR编码方式折线图

从上述的图表可以看出，随着基向量个数的增加，压缩率呈上升趋势，仔细观察可以发现，当基向量增加一个，压缩率提升百分之1左右。总而言之当前压缩方法无论是根据电路大小确定基向量，还是动态选取基向量均能取得不错的压缩效果。

4.4 小结

本章提出了一种使用聚类算法结合原测试生成主分量的数据压缩方法，该方法先通过预填充方法，去除原测试集中的无关位，然后根据电路大小确定基向量数。由于初始基向量的个数会影响最终的压缩率，并且无法确定初始聚类数，本人进行了动态选取基向量数相关实验，同时也使用当前压缩方法对大电路进行了测试。实验结果表明，使用kmeans++聚类算法集合原测试的压缩方法能大大提高压缩率。

第5章　使用kmeasn++算法结合位翻转算法进一步提高压缩率

测试数据压缩能减少数据存储开销，降低测试时间。测试集由一系列0、1以及无关位X组成，X、可以被任意填充为0或者1而不影响故障覆盖率，因此往往包含较多无关位的测试集会获得较高的压缩增益。本章将基于这种思想对原测试集进行修改从而提高压缩率。

5.1　相关概念

5.1.1 故障检测冗余度

故障检测冗余度是指单个故障在测试过程中被重复检测。比如，数据冗余是指在一个集合中相同数据重复出现，那么故障检测冗余度类也是类似的。测试集由ATPG产生，一开始产生的测试集会存在大量的数据，这些数据中有很多无关位，通过位流动态紧缩之后测试集的规模会大大减小，减小的测试集故障覆盖率并不会改变，因为很多无关位会被转化成为确定位，如此，电路中会存在多个故障被重复检测。

5.1.2 位翻转

本文主要使用拆分压缩的方式提高压缩率，拆分压缩会将原测试集拆分成为主分量集和残分量集，如果主分量集和原测试集高度相似便会提高残分量集的压缩率。假设使用主分量集代替原测试对电路进行故障模拟，就会检测出一部分故障，因此在使用原测试集对电路进行故障模拟时便会产生大量的故障冗余度。既然一部分故障已经被主分量集合检测，那么原测试集只需要检测剩余故障即可，假设原测试的总故障覆盖率为C，主分量的故障覆盖率为A，剩余故障覆盖率为C-A。原测试集只需要达到C-A的故障覆盖率即可，因此可以将原测试及中部分确定位翻转位无关位，在总故障覆盖率不变的情况下，提高压缩率。这种确定位转变为无关位的过程称之为位翻转。

5.1.3 位翻转应用于压缩

原测试集中包含的确定位会影响最终的压缩率，如果可以将部分确定位翻转成为无关位，将无关位按照有利于压缩的方向填充便能提高压缩率。为了降低硬件开销以及实验复杂度本人使用的是一轮位翻转，具体过程分为下述几个步骤：1、结合第三章与第四章的算法将原测试拆分成为主分量集和残差集。2、将主分

量集进行故障模拟，记录能检测的故障数A，在不影响故障覆盖率的前提下将原测试集中的某些确定位翻转成为无关位X，生成新的测试集。3、将新的测试中的无关位进行填充然后与原主分量集进行异或，在本实验中无关位的填充方式主要根据主分量集对应的位进行填充。

位翻转应用于压缩过程如图5.1所示：假设原测试集的故障覆盖率(FC)为100%，拆分压缩之后的主分量集能取得的故障覆盖率可达到70%，那么原测试集只需要检测出剩余的30%的故障即可，在保证故障覆盖率的条件下，对原测试集的确定位进行翻转得到新测试集，根据主分量的确定位对原测试集的无关位进行填充，最后将主分量集与已填充的测试集进行异或得到最终需要进行压缩的新残差集。

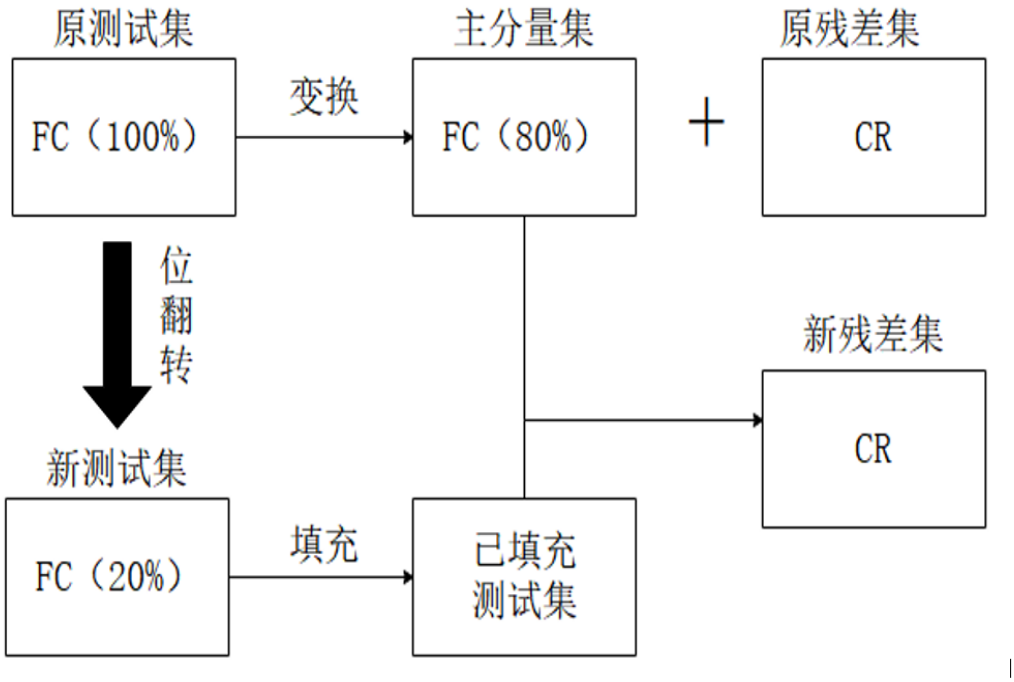


图 5.1 FDR编码方式折线图

在论文“多次随机变换拆分测试激励压缩方法研究”[76]中提及了多轮为翻转算法，但是由于每一轮翻转，均会产生代价，基本上在进行第二轮反转时提升的压缩率就会大大低于代价，本人化繁为简直接使用一轮压缩之后根据主分量集对原测试集进行二次填充，在保证压缩率的前提下节约了硬件开销以及存储代价。

5.2 翻转算法

假设原测试集经过故障模拟后能达到的故障覆盖率为C，对原测试进行位填充之后，拆分为主分量集和残差集，对主分量集进行故障模拟之后能达到的故障覆盖率为B，在对原测试及进行位翻转之后得到了新测试集，假设新测试集能达

到的故障覆盖率为A，那么A+B¿=C,下文将基于这种思路写出位翻转算法的伪代码。

算法 5.1 位翻转基本过程

```
CoverageC //原测试集的故障覆盖率
CoverageA //拆分原测试集后主分量集的故障覆盖率
CoverageC ← FaultCoverage (T)
CoverageB ← FaultCoverage (L)
for bit in T do
  if bit = 1 or 0 then
    bit ← X
  end if
  if Coverage < CoverageA − CoverageB then
    bit ← 1 or 0
  end if
end for
Return T
```

算法5.1为翻转测试集的基本过程，首先依次将测试集中的每个确定位变为无关位X，如果影响故障覆盖率，则将无关位X还原为翻转前的确定位。假如测试集总共含有k个确定位，模拟一个向量需要时间为t，则该算法运行的总时间为kt。理论上而言该算法可以通过较小的代价取得较高的压缩率。

5.3 实验结果与分析

为了验证预填充方法的有效性，我们对ISCAS’89[71]中大部分电路进行了实验，包括S5378、S9234、S13207、S15850等电路，本章将挑选其中的电路做具体描述，为了验证本实验是否对多种编码方式均适用，本文选取了FDR、EFDR、ALT-FDR编码方式对变换拆分之后的残差集进行压缩，同时与单纯使用位翻转算法达到的压缩率进行对比。

实验结果如下所示，表5.1-5.3分别表示各方法在FDR、EFDR、ALT-FDR编码下缩能达到的压缩率，其中第一列为电路名称，第二列表示对测试集直接编码所能取得的压缩率，第三列表示使用原测试集集合kmeans++算法所能达到的压缩率，第四列表示对测试集直接反转再编码所达到的压缩率，第五列为使用本方法所达到的压缩率。

表 5.1 FDR编码压缩率(%)

电路	直接编码	Kmeans++聚类	直接翻转	本方法
s5378	47.98	70.76	78.06	79.69
s9234	43.61	69.59	74.01	78.06
s13207	81.31	92.29	91.93	94.93
s15850	66.21	81.75	86.04	86.27
s38417	43.21	75.35	78.71	80.31
平均	56.46	77.95	81.75	83.85

表 5.2 EFDR编码压缩率(%)

电路	直接编码	Kmenas++聚类	直接翻转	本方法
s5378	53.67	67.75	76.16	77.68
s9234	48.66	66.14	71.28	75.85
s13207	82.49	91.60	91.53	94.52
s15850	68.66	79.84	84.24	84.99
s38417	62.02	74.06	78.24	79.13
平均	63.0	75.88	80.29	82.43

表 5.3 ALT-FDR编码压缩率(%)

电路	直接编码	Kmenas++聚类	直接翻转	本方法
s5378	49.95	65.17	72.17	76.26
s9234	45.14	62.72	67.61	73.50
s13207	80.12	90.88	88.93	94.09
s15850	65.64	77.82	81.64	83.85
s38417	60.52	71.23	75.32	76.93
平均	60.27	73.56	77.13	80.93

从上表可以看出使用本方法使测试集的压缩率得到了极大的提升，比对电路直接编码所达到的平均压缩率高22.49%，比对原测试集直接翻转所取得的平均压缩率高2.68%。

5.4 小结

位翻转方法在不改变原测试集故障覆盖率的条件下，通过将确定位翻转为无关位来提高压缩率。拆分压缩会将原测试集拆分为主分量集与残差集，主分量本就可以检测一部分故障，因此原测试只需要检测主分量集未检测到的故障即可。当原测试集的某些确定位翻转为无关位后，得到了一个新测试集，新测试集填充无关位之后与主分量异或得到了残差集，此时的残差集包含的0比特位大大增加，

从而提高了压缩率。上表中也很好地反映了这一结果，相比于第四章我们所获取的压缩率，本章将压缩率在其基础上提高了7%。

结论与展望

随着集成电路的集成度和复杂度不断提高，导致集成电路的测试工作面临诸多挑战。芯片日益增加的集成度和复杂度直接提高了电路产生故障数的概率。为了确保被测芯片的故障覆盖率达标，测试向量需要成倍的增加。庞大的测试数据增加了硬件代价和测试应用时间，一种卓有成效的方法就是对测试数据进行压缩，通过压缩测试数据不仅能大大降低被测时间更可以节约硬件存储开销。近年来有关学者提出了一种拆分压缩技术，此技术将原测试集拆分成为主分量集（在本文中由基向量生成）以及残分量集，主分量的选取直接影响最终的压缩率，本文在拆分压缩技术的基础上，对如何生成基向量展开了研究。

第一章绪论，主要介绍研究背景及研究意义、集成电路测试的相关先修知识、国内外研究现状、本文的主要工作以及文章的组织架构。第二章首先讲解测试的相关概念基础，接着介绍了随机测试技术以及常用的编码压缩技术，第一种是游程编码，第二种的字典编码，第三种是统计编码。随后花了少量篇幅介绍了两种非编码压缩方法，最后介绍了拆分压缩技术，基本思想是将原测试集，映射到另一个更易于压缩的表示空间，来达到提高压缩率的目的。本文就是在拆分压缩技术的基础上通过挑选出合适的基向量进行数据压缩。本文基于测试集拆分压缩技术提出了三点创新性工作，主要有：

(1)采用预填充的策略对测试集进行处理。此方法首先对测试数据进行预填充，填充的方式有直接填充和策略填充两种，填充的目的是使测试集在当前编码规则下的压缩率更高，填充完毕后的测试集不存在无关位，然后以向量间距离最大最原则选取需要的基向量。由于本文使用的是拆分压缩技术，当原测试及中包含的码字0较多会有助于压缩率的提升，建议直接将无关位填充为码字0。实验结果表明，使用预填充的方式，RL-Huff编码的压缩率可达74.32%，相比与对测试集进行直接编码，压缩率提高了11.75%，并且硬件开销也是可接受的。通过对ISCAS’89部分基准大电路的实验结果表明，本方法取得了较理想的效果。

(2)提出了一种基于kmeans++聚类算法结合测试集生成基向量的方法。该方法的基本思想是对已填充的测试向量进行聚类，即相似的列向量我们将其归为一类，然后取当前聚类列向量每一位的均值作为聚类中心向量。然后以每一个聚类中心向量为基准，将测试集中的每一个列向量与以获得的聚类中心向量进行对比，若欧几里得距离最小则归为一类，如此反复迭代，当聚类基本不再发生变化时，最终确定的聚类中心向量即为我们所求的基向量。实验结果表明，通过使用kmeans++算法生成的基向量来进行压缩，RL-Huff编码的平均压缩率可

达76.30%，与对测试集进行直接编码压缩相比平均压缩率提高了13.73%，与哈达码相比，压缩率提高了4.45%。同时本人使用此方法对大电路进行了测试，在FDR编码编码方式下，比对测试集直接压缩所获取的缩率高6.06%。由于基向量的选取的个数会直接影响最终的压缩率，为了更好的反映两者的对应关系，本人通过选取不同个数的基向量，计算出相应的压缩率，并绘画出其相应的折线图。

(3)将kmeans++聚类算法结合单轮位翻转算法进一步提高压缩率。此方法的主要思想是，先对原测试集进行预填充，利用kmeans++聚类算法找出所需的基向量并生成主分量集，然后使用主分量集进行故障模拟检测出一部分故障，原测试集只需检测出剩余故障即可，对于原测试集而言，可以将主分量能检测出故障相应确定位转化成为无关位，生成新测试集。最后对新测试集使用向量分解的方式进行压缩，本人基于多轮位翻转算法提出一种保留原主分量集合的单轮位翻转算法，使得在进行故障模拟时，所需要的硬件代价更小。结果表明使用位翻转算法结合kmeans++算法可以将平均压缩率在(2)的基础上提高7%。

本文所研究的三个基于拆分压缩中基向量的生成方法虽然在研究上取得了一定的成功，但仍然有缺陷和待改进的地方，总结如下：

(1)本文在第三章使用采用预填充的策略对测试集进行处理，并且其取得了较好的效果，但是主要因为是因为基向量选取方式所导致的，在预填充的测试集中，如果选举基向量的策略为随机选取，而不是使用向量间距离最大原则为依据选取会对压缩率产生较大的影响，实验表明使用随机选取方式所达到的压缩率，与使用哈达码变换所达到的压缩率十分接近。虽然本文使用了据间距最大原则，但是第一列基向量的选取依旧是随机的，初始基向量的选取对实验结果的影响较为明显，因此预填充之后，对于基向量的选择还可以继续优化，减少随机选取带来的误差。

(2)本文第四章提出了一种基于kmeans++聚类算法结合测试集生成基向量的方法。该方法的基本思想是对已填充的测试向量进行聚类，以每一个聚类中心向量为基准，将测试集中的每一个列向量与以获得的聚类中心向量进行对比，若欧几里得距离最小则归为一类，如此反复迭代，当聚类基本不再发生变化时，基向量获取结束。由实验结果可知此方法能获取较高的压缩增益，但也有缺陷，第一kmeans++算法是一个以距离为基准的算法，简单易用，其缺点是无法确定聚类数目，需要事先设定，其次在聚类时初始基向量的选取也是随机的对实验结果也会存在一定的影响。综上所述，在聚类算法的选择上可能还有比kmeans++更合适的算法，能解决上述问题，进一步提高压缩率。

(3)本文第五章将kmeans++聚类算法结合位翻转算法进一步提高压缩率，此方法基本思想是对在不影响原测试集故障覆盖率的前提下，将测试集中部分确定

位翻转为无关位。虽说针对ISCAS’99基准电路，此方法提升了较高的压缩率，但是本文中提及的翻转算法只翻转了一轮，并且翻转时使用的是贪婪算法，可能无法到达全局最优的效果，因此优化翻转算法进一步增加可翻转的确定位，比如可以根据对比测试集检测的故障集合以及相似主分量集检测的故障集合来设计翻转算法。

参考文献

[1] International Technology Roadmap for Semiconductors 2005 Edition Executive Summary, http://www.itrs.net/Links/2005ITRS/ExecSum2007.pdf.

[2] 我国集成电路产业发展分析. http://www.chinairn.com/doc/70270/229278.html.

[3] 完善集成电路产业链,增强核心产业自主性集成电路“十一五”专项规划解读. http://www.china.com.cn/policy/txt/2008-01/10/content9509005.htm.

[4] Semiconductor Industry Association (SIA) . Test Equipment, International Technology Roadmap for Semiconductors (ITRS) 2006 Update [R]. 2006, http://public.itrs.net/

[5] Sehgal A, Chakrabarty K. Optimization of Dual-Speed TAM Architectures for Efficient Modular Testing of SoCs [J]. IEEE Trans. On Computers, 2007, 56(1):120–133

[6] Sheng s, Hsiao MS. Success-driven learning in ATPG for preimage computation [J]. IEEE Design and Test of Computers, 2004, 21(6):504–512

[7] Verigy Agilent 93000 Flexible Parallel Test Solution for Power Estimation [J]. http://www.verigy.com/content/dav/verigy/Internet/Products/V93000 2006.

[8] Hashempour H, Lombardi F. Application of Arithmetic Coding to Compression of VLSI Test Data [J]. IEEE Trans. On Compt, 2005, 54(9):1166–1178

[9] Saiki T, Ichihara H, Inoue T. A Reconfgrale Embedded Decompressor for Test Compression [C]. Los Alamitos, California, USA: Proceedings of the Third IEEE International Workshop on DELTA'06, 2006

[10] Hashempour H, Lombardi F.Moussa, M. Diaz Nava. “Analyzing the Cost of the Design for Reuse”, ¡Reuse Techniquesfor VLSI Design¿. Kluwer Academic Publishers, 1999

[11] Burch R, Najm F, Yang P, et al. A Monte Carlo Approach for Power Estimation [J]. IEEE д 19 Trans. On VLSI Systems, 1993, 1(3):63–71

[12] Hsiao MS, Rudnick EM, Patel JH. Effects of Delay Models on Peak Power Estimation of VLSI Sequential Circuits [C]. Los Alamitos, California, USA: Proc. Int'l Conf. Computer-Aided Design (ICCAD), 1997:45–51.

[13] Yoon M. Sequence-Switch Coding for Low-Power Data Transmission [J]. On ,39a1 VLSI Systems, 2004, 12(12):1048–1051.

[14] Zhang H, George V, Rabaey JM. Low swing on-chip signaling techniques: effectiveness and robustness [J]. IEEE Trans. On VLSI Systems, 2000, 8(6):264–272.

[15] Shin Y, Chae SI, Choi K. Partial a bus-invert coding for a power optimization of application-specific systems [J]. IEEE Trans. VLSI Systems, 2001, 9(4):377–383.

[16] Lin RB, Tsai CM. Weight-based bus invert coding for low power applications [C]. Los Alamitos, California, USA: In Proc. ASP-DAC VLSI Design, 2002:21–125.

[17] Mousa M, NavaD. Reuse Tchiues for VLSI Design IM. Kluwer Academic Publishers, 1999, 86–90.

[18] Pateras S. A Comparison of Structural Test Approaches CI.Los Alamitos, Californi-a, USA: Electronics Manufacturing Technology Symposium, IEEE/CPMT/SEMI 29thInternational, 2004:206–221.

[19] Miron Abramovici, Melvin A. Breuer, Arthur D. Friedman, ¡Digital Systems Testing and Testable Design¿. 北京:清华大学出版社, 2004, 1(3):63–71.

[20] Burch R, Najm F, Yang P, et al. Fujiwara, H "Logic Tsing and Design for Testability," MIT Pres. Cambridge MA.1986. 66–87

[21] MeCluskey, E. J. and s. Brzxrs-Nesbat "Design for Autonomous Test" IEE Trans. Compute, Vol. 1981, C-30, No. 11, 866–875.

[22] Roth J P. Diagnosis of Automata Failures: A Calculus and a Method. IBM Journal of Research and Development, 1966, 10(4):278–291.

[23] Goel P. An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits. IEEE Transactions on Computers, 1981, 30(3):215–222.

[24] Williams, T. W. and J. B. Angel, "Enhancing Testability of Large Scale Integrated Circuit via Test Points and Additional Logic," IEEE Trans. Comput, 1973, Vol. C-22, No.1, 46–60.

[25] Eichelberger, E. B., and Williams, T. w. A Logic Design Structure for LSI Testability," Proc. Design Automation Conf, 1977, 462–468.

[26] Bennetts, R. G. "Design of Testable Logic Circuits," Addison-Wesley, Reading, MA, 1984, P.164.

[27] Agrawal V D ,Kime C R,Saluja K. Tutorial on Built-In-Self-Test,Partl Principles. Journal of DesignandTest of Computers. 1993, 10(1): 73–82

[28] Agrawal V D,Kime C R,Saluja K. A Tutorial on Built —In —Self-Test,Part2 Application. Journal of DesignandTest of Computers.1993, 10(21):69–77

[29] Agrawal V D,Kime C R,Saluja KArmstrong D B. A Deductive Method for Simulating Faults in Logic Circuits. Computers IEEE Transactions on, 1972, C-21 (5):464-471

[30] El-Maleh A H, Al-Abaji R H. Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression. In: Proc of International Conference on Electronics, Circuits and Systems. IEEE, vol.2 2002:449–452

[31] Horowitz,Hill. The Art of Electronics,2nd Edition. 1999: 665–667

[32] Koenemann B. LFSR-coded test patterns for scan design. In：Proc of European Test Conference, 1991, 237–242

[33] 王伟征, 邝继顺, 尤志强, 等. 一种基于轮流扫描捕获的低功耗低费用BIST 方法. 计算机研究与发展, 2012, 49(4):864–872

[34] Pomeranz I, Reddy S M. On static compaction of test sequences for synchronous sequential circuits[C]. Design Automation Conference Proceedings 1996, DBLP, 1996:215–220

[35] Fan X, Moore W, Hora C, et al. Stuck-Open Fault Diagnosis with Stuck-At Model. In: Proc. of Test Symposium, 2005, European, IEEE Xplore, 2005:182–187

[36] Abd-El-Barr M, Xu Y, Mccrosky C. Transistor stuck-open fault detection in multi-level CMOS circuits.In: Proc. of VLSI, 1999. Proceedings. Ninth Great Lakes Symposium on. IEEE Xplore, 1999:388–391.

[37] Smith G L. Model for Delay Faults Based upon Paths. In: Proc. of International Test Conference 1985, Philadelphia, Pa, Usa, November. DBLP, 1985:342-351

[38] 雷绍充, 邵志标, 梁峰. VLSI 测试方法学和可测性设计. 北京: 电子工业出版社, 2005, 19–27.

[39] I. Kohavi and Z. Kohavi. ”Detection of Multiple Faults in Combinational Logic Networks,” IEEE Trans. On Computers, 1972, June, Vol. C-21, No.6, 556–668

[40] J. P. Hayes. ”A NAND Model for Fault Diagnosis in Combinational Logic Networks,” IEEE Trans. On Computers, December, 1971, Vol. C-20, No.12, 1496–1506.

[41] V. K. Agarwal and A. S. Fung. ”Multiple Fault Testing of Large Circuits by Single Fault Test Sets,” IEEE Trans. On Computers, November, 1981, Vol. c-30, No.11, 855–865.

[42] D. R. Schertz and G. Metze. ”A New Representation of Faults in Combinational Digital Circuits,” IEEE Trans. On Computers, August, 1972, Vol. C-21, No.8, 858–866.

[43] D. C. Bossen and S. J.Hong. ”Cause-Effect Analysis for Multiple Fault Detection in Combination Networks,” IEEE Trans. On Computers, November, 1971, Vol. C-20, No. 12, 1252-1257.

[44] Waicukauski J A, Eichelberger E B, Forlenza D O, et al. Fault simulation for structured VLSI. Vlsi Systems Design, 1985, 52–57

[45] V.S. lyenger and D. T. Tang. ”On Simulation Faults in Parallel,” Digest of Papers 18 Intel Sym. On Fault-Tolerant Computing, 110–115

[46] D. B.Armstrong. ”A deductive method for simulation logic faults in logic circuit,” IEEE Trans. Comput., 1972, Vol. C-21, No.5, 464–471.

[47] H. Y.Chang, et al. ”Comparison of parallel and deductive fault simulation methods,” IEEE Trans. Comput, 1974, Vol. C-23, No.11, 193–200.

[48] E. G. Ulrich and T. Baker. ”The concurrent simulation f activity in digital networks,” Compute, 1974, Vol. 7, No.2, 39–44.

[49] M.Abramovici,et al, M. A.Breuer and K. Kumar. ”Concurrent fault simulation and functional level modeling,” Proc. 14th Design Automation Conference, 1977, 128–137

[50] W. A.Rogers, J. F.Guzolek and J. Abraham. IEEE Trans. Comput-Aided Des, 1987, Vol. CAD-6, No.9, 848–862.

[51] Agrawal VD, Dally WG. A Hardware Logic Simulation Systems . IEEE Trans. Comput,1978, C27(11):1054–1055

[52] Parker KP, McCluskey EJ. Probalistic Treatment of General Combinational Networks [J]. IEEE Trans. Comput., 1975, C24(6):668–670

[53] Eichelberger EB, Lindbloom e. Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test d [J]. IBM Journal of Research .and : Development, 1983,27(33):265-272.

[54] Jas A, Touba N A. Test Vector Decompression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs. Proc.int.test Conf. 1998:458–464.

[55] Chandra A, Member S, Chakrabarty K, et al. System-on-a-chip test-data compression and decompression architectures based on Golomb codes. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2006, 20(3):355–368.

[56] Chandra A, Chakrabarty K. Test Data Compression and Test Resource Partitioning for System-on-a-Chip Using Frequency-Directed Run-Length (FDR) Codes. IEEE Transactions on Computers, 2003, 52(8):1076–1088.

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

[58] Chandra A, Chakrabarty K. A unified approach to reduce SOC test data volume, scan power and testing time. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2003, 22(3):352–363.

[59] Jas A, Ghosh-dastidar J, Ng M, et al. An efficient test vector compression scheme using selective Huffman coding. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2003, 22(6):797–806.

[60] Kavousianos X, Kalligeros E, Nikolos D. Optimal Selective Huffman Coding for Test-Data Compression. IEEE Transactions on Computers, 2007, 56(8):1146–1152

[61] Kavousianos, Xrysovalantis, Emmanouil Kalligeros, and Dimitris Nikolos. Multilevel Huffman coding: an efficienttest-data compression method for IP cores.Computer-Aided Design of Integrated Circuits and Systems. IEEE Transactions on,2007,26(6): 1070-1083

[62] Kavousianos X, Kalligeros E, Nikolos D. Multilevel-Huffman test-data compression for IP cores with multiple scan chains. Very Large Scale Integration Systems IEEE Transactions on, 2008, 16(7):926–931

[63] Kavousianos X, Kalligeros E, Nikolos D. Test Data Compression Based on Variable-to-Variable Huffman Encoding With Codeword Reusability. Computer-Aided Design of Integrated Circuits and Systems. IEEE Transactions on, 2008, 27(7):1333–1338

[64] Tehranipoor M, Nourani M, Chakrabarty K. Nine-coded compression technique for testing embedded cores in socs. Very Large Scale Integration Systems IEEE Transactions on, 2005, 13(6):719–731

[65] Lin S, Chung-Len Lee, Chen J, et al. A multilayer data copy test data compression scheme for reducing shifting-in power for multiple scan design. Very Large Scale Integration Systems IEEE Transactions on, 2007, 15(7):767–776

[66] El-Maleh A H. Efficient test compression technique based on block merging. Iet Computers and Digital Techniques, 2008, 2(5):327–335

[67] Wu Tie-Bin, Liu Heng-Zhu, Liu Peng-Xia. Efficient Test Compression Technique for SoC Based on Block Merging and Eight Coding. Journal of Electron Test, 2013, 29:849–-859

[68] Sinanoglu O. Scan Architecture With Align-Encode. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2008, 27(12):2303–2316.

[69] Wang Z, Chakrabarty K. Test data compression using selective encoding of scan slices. IEEE Transactions on Very Large Scale Integration Systems, 2008, 16(11):1429–1440.

[70] Hamzaoglu I, Patel J H. Reducing Test Application Time for Full Scan Embedded Cores. International Symposium on Fault-tolerant Computing. 1999:260

[71] Brglez F, Bryan D, Kozminski K. Combinational Profiles Of Sequential Benchmark Circuits. Circuits and Systems IEEE International Symposium on, 1989, 3:1929–1934

[72] Sinanoglu O. Scan Architecture With Align-Encode. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2008, 27(12):2303–2316.

[73] 梁华国, 蒋翠云, 罗强. 应用对称编码的测试数据压缩解压方法. 计算机研究与发展, 2011, 48(12):2391–2399

[74] 程一飞, 詹文法. 长度折半的测试资源编码方法. 电子测量与仪器学报,2016, 30(3):480–486

[75] Yuan Haiying, Mei Jiaping, Song Hongying. Test Data Compression for System-on-a-Chip using Count Compatible Pattern Run-Length Coding. Journal of Electron Test, 2014, 30:237–242

[76] 秦光睿. 多次随机变换拆分测试激励压缩方法研究[D]. 2018

致　谢

如果说人生是一首优美的乐曲,那么湖大的日子则是其中一个美妙的音符。三年前我成功考上了心心念念的湖大，一切都是那么新鲜，由于我是个跨专业的考生，在研究生期间我加倍努力，打实专业基础，在此期间由于结识了一批优秀的同学，让我对自己的人生有了规划与追求。这一路走来，有欢声笑语，也有挫败失落。三年的研究生时光即将结束，要感谢的人太多，在此衷心向所有支持和帮助我的老师们、同学们致以最诚挚的谢意。

首先，由衷地感谢我的导师邝继顺老师！老师对我的帮助太多，从生活到学习方方面面都会涉及，老师是一个很开明的人，对于学生想做的事情只要对人生、学业有益，均会全力支持。老师是个很睿智的人，对于生活中遇到的困难，均会加以开导，每次与老师谈话完毕总是豁然开朗。同时老师是一个很严谨的人，对待工作一丝不苟，以学生学业为重，为了提升教学质量，经常想各种方法来激起学生的学习积极性与主动性。三年前，考研初试刚刚结束，我就联系了老师，知道我是跨考生后老师并没有嫌弃我基础薄弱，反而悉心指导我该看什么书籍，该朝哪方面努力，在此由衷感谢导师这三年对我的帮助与关照，您的悉心教导足矣让我受用终身。

感谢我的校外导师文吉刚，感谢您在实践期间对我的帮助与悉心指导，每次遇到不会解决的问题时，都会很耐心地给我解答。特别是择业期间，你给我提供了很多宝贵意见，结合我自身实际情况给我分析了各个企业的优劣，最终使我获得了心仪的工作。祝愿文吉刚老师在今后的每一天身体健康、事事顺心。

感谢实验室的老师以及各位同学，感谢尤志强老师、凌纯清老师、张明和博士、谭谨慧博士、李莹师姐、刘鹏程师兄、杨家啟师兄在学习与生活上对我的指导与帮助。感谢师弟陈泽军，在实验室活动时，每次有你的参加，都会十分精彩有趣。感谢逗逼室友杨远达，因为你让我的生活变得十分有趣。感谢大佬室友彭浩，是你让我了解到自己离优秀的距离还相当遥远。特别感谢何振茜同学，对我的学业、生活帮助非常大。

尤其要感谢我的母亲，谢谢你一直以来的支持与鼓励，教我明辨是非，教我如何做一个大写的人。感谢我的父亲，感谢您为家庭的默默付出，任劳任怨，因为您我才有机会顺利完成学业，拥有更精彩的人生。在以后的工作与生活中，我会带着你们的期望继续前行，愿你们身体健康，事事顺心。

最后，感谢参与盲审以及答辩的各位专家和老师，谢谢你们提出的宝贵意见。

附录A　发表论文和参加科研情况说明

[1] 社会达的四区论文

[2] 软件著作权：夏泽，邝继顺. PRE聚源集成电路测试向量压缩软件. 登记号：2020SR0199298.