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

工程硕士学位论文

拆分压缩中基向量的生成算法研究

学位申请人姓名 XXX
培 养 单 位 信息科学与工程学院
导师姓名及职称 XXX
学 科 专 业 计算机技术
研 究 方 向 SoC测试与设计
论 文 提 交 日 期 2020年4月24日

学校代号: 10532
学　　号: XXX
密　　级: 普通

湖南大学工程硕士学位论文

拆分压缩中基向量的生成算法研究

学位申请人姓名: XXX
导师姓名及职称: XXX
培　养　单　位: 信息科学与工程学院
专　业　名　称: 计算机技术
论文提交日期: 2020年4月24日
论文答辩日期: 2020年4月30日
答辩委员会主席: XXX

Research on Generation Algorithm of Base Vector in Split Compression

by

XXX

B.E. (Hunan Agricultural University) 2017

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of engineering

in

Computer Technology

in the

Graduate school

of

Hunan University

Supervisor

Professor XXX

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。为了确保芯片的故障覆盖率达标，需要大量的测试数据对其进行检测。庞大的测试数据增加了硬件代价和测试时间，通过压缩测试数据能大大降低被测时间并节约硬件成本。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。此方法通过预先对测试集进行填充消除所包含的无关位。填充方式可分为直接填充和策略填充两种，直接填充指是将原测试集中的无关位直接填充为码字0或者码字1，策略填充则是将无关位按照有助于提高特定编码压缩率的方式进行填充。对测试集填充后，本文以向量间距离最大为原则选取所需基向量。实验结果表明，通过采取预填充的压缩方法，RL-Huff 编码的压缩率可达74.32%，相比于对测试集直接进行编码，压缩率提高了11.75%，相比于哈达码变换，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的初始聚类中心，然后计算出测试集中每一个列向量与聚类中心之间的欧几里得距离，以距离最小为原则将列向量重新划分到不同的聚类，并计算出新的聚类中心，如此反复迭代，当聚类中的列向量基本不再发生变化时，所得的聚类中心即为本文所求基向量。实验结果表明，通过使用kmeans++ 聚类算法结合测试集生成基向量的方法可使RL-Huff 编码的平均压缩率提高至76.30%，与对测试集直接进行编码压缩相比，平均压缩率提高了13.73%，与哈达码变换相比，平均压缩率提高了4.45%。为了进一步验证方法的可靠性，本人对b15、b17、b20、b21 等大电路进行了相关实验，在FDR 编码方式下，此方法所获取的压缩率比对测试集直接进行压缩高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：首先对主分量集进行故障模拟检测出电路的部分故障X，假设原测试集的故障覆盖率为1，则原测试集只需检测出1-X这部分故障即可。为了避免相同的故障被反复检测，本文在不影响故障覆盖率的情况下将原测试中部分确定位翻转为无关位，然后将位翻转之后的测试集与主分量集进行异或获取残分量集，从而提升压缩率。实验结果表明通过使用kmeans++ 聚类算法结合位翻转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 chips directly increase the probability of producing failed chips. In order to ensure that the tested chip has 100% fault coverage, a large amount of test data is required to detect it. The huge test data increases the hardware cost and test application time. Compressing the test data can greatly reduce the test time and save the hardware storage overhead. In this paper, based on the split compression technology, we study how to generate basis vectors, and mainly do the following three aspects of work:

(1)A pre-populated strategy is proposed to process the test set. This method aims to eliminate extraneous bits in the test set and fill the test set in advance. The specific stuffing can be divided into direct stuffing and strategy stuffing. Direct stuffing means replacing the irrelevant bits in the original test set with codeword 0. Strategy stuffing is to fill the irrelevant bits in a way that contributes to a particular encoding compression. After filling the test set, this paper selects the required base vector based on the principle of the largest distance between vectors. The experimental results show that, using the pre-filling method, the compression rate of RL-Huff coding can reach 74.32%. Compared with the direct encoding of the test set, the compression rate is increased by 11.75%. 2.47%.

(2)(2)A method of generating basis vectors based on kmeans ++ clustering algorithm combined with test set is proposed. This method selects the initial clustering center of the test set based on the principle of maximum distance, then calculates the Euclidean distance between each column vector in the test set and the clustering center, and re-divides the column vector into the principle of minimum distance Different clusters and calculate new cluster centers. Iterate iteratively in this way. When the column vectors in the cluster basically no longer change, the resulting cluster center is the base vector sought in this paper. Experimental results show that the use of clustering algorithm combined with split compression method can increase the average compression rate of RL-Huff encoding to 76.30%. Compared with the direct compression of the test set, the average compression rate is increased by 13.73%, which is Compared with transcoding, the average compression rate is increased by 4.45%. In order to further verify the reliability of the method, I conducted related experiments on large circuits such as b15, b17, b20, and b21. Under the FDR coding method, the compression ratio obtained by this method is 6.06% higher than that of the

direct test set.

(3)The kmeans ++ clustering algorithm combined with the bit flip algorithm further improves the compression rate. The main idea of this method is: first, the main component set is used for fault simulation to detect part of the circuit's fault, then the original test set is also used for fault simulation, the main component set can detect the fault corresponding to the corresponding bit is converted into an irrelevant bit, and Generate a new test set. Finally, the new original test set is XORed with the main component set to obtain a new residual component set and improve the compression rate. Experimental results show that the use of bit reversal 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

随着科学技术的进步，超大规模集成电路的发展速度日新月异。芯片集成度以及复杂度的急剧上升增加了制造合格芯片的困难，同时也增大了芯片制造过程中出错的概率。因此，集成电路的测试显得尤为重要。

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，集成电路更新迭代速度惊人，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;当今芯片的集成度从早期的10多个元器件，发展至上十亿个晶体管。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。

我国集成电路的发展经历了以下几个阶段[1]:1978-1990年我国主要靠进口芯片来实现人们日常生活电子产品的国产化。1990-2000 年:以CAD制图、908工程、909工程为重点，促进集成电路行业的发展。2000-2005年:“十一五”期间，我国集成电路相关产业飞速发展，制造的芯片体积降低至0.18 μm[2]，并且在光刻机等领域取得优异成绩[3]。2006-2020年，国家将集成电路将作为重点领域开展针对性研究。种种迹象表明，集成电路已如同细胞组成人体一样，成为当今社会有序运行不可缺少的重要组成部分。

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，本文将从以下几点来总结测试的复杂性和难点。

(1)电路的性能越好功能越完善所需要的检测开销越高，根据TRS的研究表明[4]。近年来生产芯片的主要开销来源于测试，生产芯片的成本反而更低。而功能比较完善的芯片内部构造相对复杂，在芯片的设计过程中需要综合考虑各种问题，包括功耗、响应速度等等，因此测试成本相比于普通芯片而言会更高。

(2)随着测试数据量的增大，测试时间显著增加[5]。测试时间增加，往往会延长芯片上市时间，从而造成经济上损失。因此，相关领域的公司研发出了更强大的ATE，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]。虽然众多公司积极采取措施应对这一问题，但由于被测电路有限的IO通道以及ATE设备的局限性（数据处理能力、吞吐量），集成电路的测试依旧是一个厄待解决的难题[8,9]，因此测试时间也成为片上系统设计需要考虑的重要指标之一[10]。

(3)集成电路的测试功耗以及自动测试仪的带宽，也成为影响芯片测试的重要因素。电路在测试时，其功耗最高可达正常工作时的4倍。如何降低功耗[11,12]以及减少ATE测试数据间的转换次数是相关领域的研究热点[13,14]。

综上所述，基于传统的模拟、验证以及测试的方式难以判断芯片是否存在故障，因此在设计和测试方面应该加以改进[15,16]，设计出容易测试的电路。可测性设计DFT[17] 就是在设计前预先考虑棘手的测试问题，从而避免芯片开发完之后，需要耗费高额的测试成本检测芯片的可用性。可测性设计可以有效地缓解复杂的测试问题，内建自测试和边界扫描设计是两种经典的可测性设计方法。采用边界扫描结构，可通过少量的IO进行测试施加和测试响应分析，其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均会增加。基于内建自测试虽然可以取得不错的效果，但是面临的主要问题有：测试功耗随着测试集的增加而增加，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在保证芯片质量的同时以尽可能低的成本对其完成测试。

1.2 集成电路测试

1.2.1 测试

测试过程就是将测试激励输入到被测器件中，并对测试响应进行分析的过程。如下图所示1.1。在电路检测的过程中，首先需要对测试电路建模，描述出测试电路输出结果的好坏标准，然后产生测试数据，将有相应输入个数的测试位流置于被测电路（CUT）的测试端。最后以无故障电路产生的响应为标准，与测试向量的输出响应进行对比，以确定被测电路是否有故障，从而筛选出合格与不合格的芯片。

建模是属于电路测试过程中的第一步，至关重要。建模这个步骤一旦出现差错，后续的流程也会随之出错，常用的建模方法有基于节点信号特征的建模方法以及基于电路故障搜索建模方法等。本文主要是针对测试激励压缩开展的研究，实质上测试响应也需要进行压缩，对比实验结果的准确性，来判断芯片是否存在故障，测试响应压缩与测试激励压缩不尽相同，测试响应压缩可以丢弃掉某些值，并且也不要求压缩之后的测试激励无差错地还原，只要最终达到检测芯片的目的即可。

图 1.1 电路测试分析过程

1.2.2 生产测试和功能测试

生产测试又被称为产品测试，通过对芯片进行缺陷测试或者故障测试以验证其是否符合标准。生产测试方案的制订必须考虑AET性能、测试功耗以及测试时长等影响因子。

功能测试就是对产品的功能进行测试，属于黑盒测试，主要通过检验芯片功能是否符合预期来判断芯片的质量。按照测试过程分类，功能测试可以分为测试施加、测试生成和测试结果验证。按测试生成的方法分类，功能测试可分为穷举测试、伪穷举测试、伪随机测试等。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包括(1)穷举测试、(2)伪穷举测试[20]、(3) 伪随机测试[21]、(4)确定性测试、(5)测试施加[22,23]。

1.2.3 可测性设计

在芯片测试过程中，测试图形生成冗长且困难，于是可测性设计方法就得以发展，其主要是为了解决芯片测试过程所面临的诸多问题，比如测试时长、测试功耗、测试数据量等。因此，在设计芯片的初始阶段就需要考虑如何测试的问题。可测性设计出现于上个世纪70 年代，最先提出的是通过扫描路径[24]对电路进行设计，其中一个典型的应用就是IBM进行的80286研发[25]。

通常有两种方法实现可测性设计。一种方法是专项技术，另一种方法是系统化技术。对于可测性设计，著名学者Bennetts给出了一个定义，给定一个集成电路，通过耗费有限时间和成本完成对其测试，并达到预期的效果，则称此电路是可测的[26]。实质上这个定义有些含糊，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。其流程大致为：在扫描模式下，扫描链移入一个即将被施加到组合逻辑上的测试向量。在系统模式下，经过一个时钟周期，测试向量被施加到组合逻辑上，输出响应在时钟信号下进入触发器。然后扫描链在扫描模式下将组合逻辑的输出响应移出来，同时将下一个测试向量移进去，如此反复进行直至测试完毕。

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，然后将已压缩的激励送入到被测芯片上，最后将经过解压器解压缩的测试激励送入被测电路进行测试，捕获电路测试响应，若实际响应与压缩响应一致，则认为此块芯片是合格的。测试激励压缩与测试响应压缩是测试数据压缩的两个部分，根据不同的场景将采取不同的压缩方法，本文所研究的激励压缩要保证压缩之后的测试向量能准确无误地被还原，属于无损压缩。

图 1.2 数字电路测试压缩结构图

测试激励有多种压缩方式，比如编码压缩、线性扫描压缩等等。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，近年来提出了拆分压缩这种新型压缩方法。在图像和音频信号的压缩中人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，而是将数据以另外一种便于压缩的方式展现出来，通俗地说就是使用一种特殊的码字替换原码字，从而提高压缩率。视频压缩或者音频压缩可以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)内建自测试[29]

内建自测试是为了快速地对集成电路进行诊断与测试，为了实现这一功能，

一种合适的方式就是将测试规定为一种系统功能。

内建自测试由内建测试（built-in test ,BIT）和自测试（self-test）这两个部分组成。BIST 是电路（芯片、板或者整机）测试自身的能力。自测试往往不是由硬件实现的，而基于软件实现的，完全基于软件的方式满足集成电路系统级别的要求，会产生诸多缺点，比如降低测试诊断分辨率、延长开发周期与时间、消耗大量的开发成本等等。

内建自测试，是一种使用器件的部分电路来测试器件本身的设计技术，它可以测试功能、结构，但不能测试参数。现在，BIST 技术主要分两种：在线（On-line）[30]和离线（Off-line）。在线，进一步可分为并发在线（Concurrent on-line）和非并发在线（Noconcurrent on-line）；离线，进一步可分为功能离线（Functional off-line）和结构离线（Structural on-line）。

在线BIST，是在能够完成所需要求的情况下进行的。并发在线测试能按照所需要求完成，和其他操作同时进行，这种测试通常用编码技术或复制、比较技术以及线性反馈移位寄存器[31,32]的技术来完成；非并发在线，是指能按照所需要求来完成空闲状态，也叫做后台操作，这种测试通常用软件程序来完成。在线测试是指由其他能够按照要求完成的操作，因此，在线测试能检测实时错误。

离线BIST[33]是指不能够完成所需要求的情况下进行的，通常使用输出响应分析器和测试向量生成器来完成。功能离线是根据被测器件有什么作用来测试的；结构离线是根据被测器件结构的类型来测试的。由于离线测试没有按照所需要求来完成操作，因此离线测试不能检测实时错误[34]。

BIST 电路一般是由测试图形生成激励、被测电路、数据压缩的电路、进行比较分析的电路、存储理想结果的电路（ROM）和进行自测试控制电路组成，一般结构如图1.3所示。

图 1.3 内建自测试一般结构

推崇大力发展内建自测试主要分为四个目的，第一：为了解决芯片复杂度的问题。当芯片复杂度大大提升时，简单地对测试难集进行拆分是不可能的，并且

使用传统化技术解决相对复杂的测试问题相当困难。BIST提供了层次化的解决方法，通过自顶向下的方式对芯片进行测试，同时为了更好地检测一块芯片，判断芯片是否存在质量问题，BIST提供了对加载原件的有效测试，减少了系统级别测试的压力。第二：解决了芯片质量低下的问题。评价芯片质量是一件复杂的任务，首先发生故障的种类不仅取决于系统、芯片，芯片的制造工艺也是不可忽略的因素。通常情况下会将芯片质量定义一个考量标准，比如将芯片的拒收率控制在十万分之1，或者将测试成本控制在一定的范围内。第三：解决了BIST 经济效益的问题。是否采用BIST方案对芯片进行设计，在生产芯片的初始阶段就是一个需要考量的问题。如果是从芯片的层面考虑，使用BIST方案不会节约较多的测试成本，但是BIST往往作用于芯片生产流程的整个生命周期，因此可以带来极为可观的收益。第四：为了解决测试应用问题。近年来，印制板测试主要基于在线测试，在线测试无法首先系统级诊断，只在板从系统拆解之后才是有效的。其次，BIST 为了解决相关测试问题提出了相关优秀的解决方法，比如，在内建自测试时，能够对整机进行测试包括芯片、零件等而避免了昂贵的外设代价，其次对于硬件板和芯片的离线测试，可以使用在系统级别测试。

BIST 的测试向量生成技术可以分为三类：穷举/伪穷举测试法；伪随机测试法；确定性测试法。内建自测是当前广泛应用的可测试性设计方法。其主要思想是测试的向量是由自身获取，不依赖于外部输入[35]。

(2)测试数据压缩

在测试数据压缩方面，国外的起步较早，在1998 年，Jas 和Touba 提出一种基于码字0编码的编码方案[39]，其以较短的固定码字替换较长的0码字串。在2001年，Lorse和Sam等人提出来一种基于Golomb 代码的新的解压缩架构，这种全新的测试压缩方法在编码片上系统中运用广泛，通过使用单个自动测试仪I/O通道驱动片上系统中的多个核，此种解压缩架构又被称为交织解压缩架构[40]。2013年，相关学着基于典型测试序列中0码字的运行分布，提出来一种可变长度的编码方式，并将其称为FDR编码[41]，FDR编码是一种用于对连续0游程编码的方法，若测试集中的1码字较多，会影响压缩率。Maleh 和Abaji 在FDR编码的基础上加以完善，提出了EFDR[42]的编码方式，此种编码不仅可以对0游程编码，也适用于对1游程编码。随后Usha S. Mehta 和Kankar S为了在游程编码的基础上进一步提高压缩率，提出了一种名为Hamming Distance Based Reordering 和Columnwise Bit Stuffing with Difference Vector （HDR-CBS-DV）的新方案[43]。同时Dasgupta 伊利诺伊大学相关研究人员提出了一种伊利诺伊扫描结构[44]。文献[45]提出了一种新的LFSR 重播技术，该技术利用电路内部网络的响应作为改变LFSR 状态的控制信号；文献[46]提出了一种新颖的压缩方法和低成本的解压缩架构，它将基于符号和基于线性的技术的优势结合在一个统一的多核SoC 解决方案中；为了减少测试响

应数据，文献[47]提出了一种新的测试响应压缩方案，称为选择性测试响应集；

国内在数据压缩领域起步较晚，但是也有相关研究成果被提出。文献[48]提出了一种测试数据压缩方法，它结合了基于字典的压缩和基于位掩码的压缩的优点，是一种基于非固定的索引，它使用较短的索引来表示具有较高出现频率的切片，而使用较长的索引来表示具有较低出现频率的切片，然后采用有效的位掩码和基于最大度的Clique 分区算法来创建尽可能多的兼容切片，文中还提出了一种新的可变前缀双游程代码来压缩不兼容的切片来减少测试数据量；文献[49]提出了一种考虑模式信息维度的基于游程的压缩方法；文献[50] 提出了一种选择性模式压缩方案，可以在基于扫描的测试期间，有最小的测试功率和测试数据量；文献[51]提出了一种二维（空间/时间）压缩技术，可以减少测试数据量并测试知识产权内核扫描测试的应用时间。

通常而言测试数据压缩框架如图1.4所示，ATE上存储了所有的测试数据，购买自动测试仪价格昂贵，因此往往会将测试数据进行压缩节约成本[36–38]，因此在原有的基础上必须设计解压缩电路，将测试数据还原，其框架图流程图如下图1.4所示。

图 1.4 TRP框架图

1.4 本文主要工作及组织结构

本文在拆分压缩的基础上，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，本文在拆分压缩的基础上,提出来两种基向量生成算法，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。

文本总共分为五个章节阐述自己的工作：

第一章是绪论部分，首先介绍了集成电路发展的背景以及电路测试的相关先修知识，其次就国内外研究现状进行分析最后总结本文的组织结构。

第二章主要就常用的压缩方法做了详细地介绍。第一部分讲解了编码压缩技术，包括游程编码、字典编码以及统计编码。第二部分就基线性解压缩和广播扫描两种非编码压缩方法做出了相应的分析与阐述，最后介绍了拆分压缩技术和哈达码变换，并对哈达码变换的优缺点进行了分析。

第三章重点介绍了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。

第四章详细介绍了相关聚类算法以及如何将聚类算法应用于测试电路压缩的具体步骤。同时本章进行了大量的实验用于验证此方法的有效性，并通过动态选取基向量数，较好地反映出压缩率与基向量个数之间的线性关系。

第五章介绍了位翻转算法的相关概念与计算流程，并将kmeans++算法结合位翻转算法进一步提高压缩率。

最后总结全文，并对该研究工作中的问题进行了展望。

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，测试芯片需要的巨大测试数据集不仅提高了测试功耗，更增加了测试时间以及存储开销。数据压缩可以有效地解决上述问题。本章首先讲述电路可测性基础理论，然后就编码压缩方法、非编码压缩方法以及拆分压缩做进一步阐述。

2.1　测试压缩原理

测试压缩的主要目的在于减少测试数据量、降低测试时间以及节约最终的芯片制造成本。测试数据压缩可以采用多种方式，其目的均是为了降低原测试集中的比特位。当前比较常用的素具压缩算法有基于编码的数据压缩算法，基于线性扫描的数据压缩算法以及基于拆分测试集的数据压缩算法。

测试数据压缩有别于视频压缩，视频压缩属于有损压缩，可以丢弃残差集。当丢弃部分数据之后，并不会有损人们的视觉或者听觉效果。测试数据压缩属于无损压缩，压缩之后的数据需要被无差错地还原。测试集由0、1以及*X*位（无关位）组成，*X*位的产生和自动生成算法有关，测试集中具有较多的无关位，部分电路测试集甚至可达80%以上。在测试压缩中，*X*根据特定的场景被任意地填充为0 或1，压缩率往往随着的无关位的增加而提高。本文主要针对测试激励进行相关压缩算法的研究，常见的测试激励压缩方法有广播扫描、编码压缩以及线性解压缩。在测试响应压缩领域也有很多压缩方法，常用的有奇偶校验、特征分析等压缩方法。其中通过线性反馈移位寄存器（LFSR）来对特征进行分析是十分普遍的现象[52,53]。

2.2　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。编码压缩大体可分为三类：第一类是信息源的统计特性分为变换编码、子带编码等。第二类是根据人眼视觉特性，采取小波变换、基于方向滤波的图像编码等。第三类是根据传递景物特征编码。采用不同的编码方式对相同的数据集编码方法，所达到的压缩率是不一样。压缩通用流程如下图2.1 所示。

图 2.1 压缩通用流程图

2.2.1 游程编码

游程编码，又称运行长度编码，指由一连串字符（或信号采样值）组成的数字编码。该编码属于无损压缩编码，其广泛应用于数据压缩领域，同时游程编码[54]也是作为二值图的重要编码方法之一。目前在数据压缩算法中较为常用的游程编码有Golomb编码[55]、FDR编码[56]、EFDR编码[57]以及交替游程编码(ALT-FDR)[58]。根据游程编码进行分类，有单游程编码和双游程编码两种方式。其中单游程编码中比较常见的有FDR编码，FDR编码的基本规则是，只对游程中0的长串进行编码，根据0串的长度会生成唯一的码字，当0串越长所对应的码字相对于原串而言就越短，从而压缩率也越高，对于游程中出现的1比特位，我们使用使用码字00来进行编码，因此当测试集中1比特位较多时，不建议使用FDR编码，EFDR的效果更佳。下表2.1为FDR的前几组编码。

EFDR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码字也同样适用。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2.2为EFDR前一部分编码规则。下面举例说明EFDR的的压缩流程。测试向量T = 11111100000001111111111111110，根据EFDR码表可将其划分为三个游程，其长度分别为6,6,14。使用EFDR编码压缩后测试向量变为11010 01010 1110111，减少了12位。

表 2.1 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

表 2.2 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

2.2.2 字典编码

字典编码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，在文本压缩、音频压缩中应用比较广泛。相比于统计编码，它的实现既不需要用到

统计的方法，也不需要用到变长码的方式，而是从数据流中选择多个字符串，利用字典的方式，对其进行编码压缩。字典编码根据是否可以通过静态的方式对数据流中的字符串进行保存分为非自适应LZ编码和自适应LZ编码。

下图2.2是使用完全字典压缩方法的原理图，图中*b*代表测试通道的数量。*n*代表扫描链的数量，*n*的值往往大于*b*，*n*与*b*的比值越大，每一条扫描链的长度就越。使用此方法能提高加载测试向量的效率，减少测试时间。其不足之处在于，若使用完全字典的方式，需要存储巨大的数据量，增大硬件开销。因此部分学者使用部分字典的方式，对不在字典中的数据进行纠错以达到控制字典的大小，通过在硬件代价与纠错成本之间进行权衡，从而达到一定的压缩率。

图 2.2 字典压缩方法

2.2.3 统计编码

统计编码是一种在测试数据压缩中广泛运用的压缩方法。其本质就是通过某种方式将原数据集拆分为一个个数据块并进行数学统计，将重复率高的数据块以短码字编码，从而达到数据压缩的目的。假定原数据集合为*X*，先按照一定的规则对*X* 进行分块或者分类，统计出相同块的数量，然后进行编码。被大家广为熟知的有霍夫曼编码[59]。霍夫曼编码通过构造霍夫曼树，将原数据集合拆分成为等长的信源符号。然后统计出相同的信源符号的个数并排序，将重复次数多的部分用短码字描述，重复次数少的部分用长码字描述，从而减少原测试集中的比特位，压缩测试数据。

部分学者对Huffman编码进行改进，衍生出譬如最优选择哈夫曼(OP-SHC)[60]以及多级哈夫曼[61–63]等编码方式。下面将举例说明如何使用霍夫曼编码，下图中框中为本例使用的数据集，总共有80位。对于初始数据集的处理，首先将其划分成为4 位为一组的比特块，然后将每一个比特块出现的频率进行统计，如下表2.3 所示，可以发现表中1010、0000、1111三种数据块出现的频率最

高。最后对出现次数最多比特块进行霍夫曼编码，如下图2.3所示。对于未编码的块则需要在原有码字之前加上111，通过使用霍夫曼编码可将原数据集中的数据压缩38.8%。

表 2.3 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 2.3 最优选择哈夫曼编码

除了上文介绍的几种编码外，9C[64]、多层复制[65]、块合并[66]、BM-8C[67]、SVC[68]、选择扫描切片[69] 等编码在数据压缩领域也较为常用。

2.3 非编码压缩

2.3.1 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，如下图2.4所示，其由四部分组成，左边是拥有b个通道的代码测试仪，经过LFSR到达了组合线性网络，然后将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，因此需要$b(q+m)$ 个比特位的数据才能产生一个测试

集。当对测试集解压缩时，线性反馈移位寄存器会将压缩数据还原为原测试集合并进行初始化，等待时钟周期的到来再将数据传送至扫描链。

图 2.4 时序线性解压器

2.3.2 基于广播扫描

“广播扫描”从字面上的意思来理解，就是使用广播的方式将数据施加多条扫描链，可以类比于IO多路复用的方式，即使用一个线程监听多个socket，在本实验中即驱动多个扫描链。其中Illinois扫描结构[70]比较具有代表性，其结构如图2.5所示。

图 2.5 广播扫描结构

广播扫描模式较传统扫描模式不同点在于，广播扫描模式将扫描链进行拆分并将扫描链连接到同一输入。上图的原始扫描链为4L，每一个子扫描链都为L，这可能会导致故障遗漏，因此，需要结合串行扫描模式才能排查出电路中存在的故障，提高故障覆盖率。广播扫描模式由于扫描链之间会存在相等值，将导致某些故障无法检测，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但组内数据相容，最终确保每一组故障覆盖率达标。其次采取重新配置广播扫描链的结构或者降低测试过程所需的功耗也

可用来提高压缩率。

2.4 哈达码变换

2.4.1 基本原理

哈达码变换是利用哈达码矩阵作为变换矩阵，将原有数据表示为新数据的一种变换方式。自然界有很多的信号，例如图像、声音、视频等经过变换编码后进行高倍率的压缩。哈达码变换使用通式$Y = HX$表示，其中X和Y均是一列向量，H为哈达码矩阵。哈达码变换已广泛应用于视频和音频编码中，由于其硬件代价低实现简单的特点，也被人们应用于测试数据压缩。与多媒体数据的压缩的不同之处是，激励压缩属于无损压缩。

2.4.2 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，矩阵中1等价于测试集中的1比特位，-1等价于测试集中的0 比特位。哈达码矩阵可以被递归定义：

$$H(k) = \begin{bmatrix} H(k-1) & H(k-1) \\ H(k-1) & -H(k-1) \end{bmatrix}, k = 1, 2, ..., n, \tag{2.1}$$

其中H(0)=1,根据上述通式，当$k = 1$以及$k = 2$时有下列矩阵：

$$H(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.2}$$

$$H(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{2.3}$$

哈达码矩阵的行列均是一个基本向量，本文将其称作Walsh函数。以H(2)为例，它有4 个基本向量，通过向量间两两相互组合，就可以变换出任何一个具有4个比特位的列向量。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。

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（EFDR、RL-Huff、ALT-FDR）、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。

2.4.3 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。

2.4.4 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。

2.5 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。首先本章对测试压缩原理进行了简要的描述。其次，重点介绍了编码压缩，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，也提及了基于非编码的压缩方法。由于本文的研究主要是基于拆分压缩，并且是与哈达码变换进行对比，因此详细介绍了相关知识。其中包括哈达码矩阵的构造、哈达码变换的基本流程以及哈达码变换存在的优势以及不足，最后提出了使用聚类方法构造主分量集合这一思想。

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，对电路的故障覆盖率并没有影响。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，本章主要对测试集如何填充进行了相应的实验，通过合理地填充无关位，来提高最终的压缩率。

3.1　变换拆分

变换拆分技术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，残分量经过压缩之后存储于自动测试仪中。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。

拆分压缩的结构图如下3.1所示，被测电路下方的数据集合被称之为测试集，Vector1到Vectori是测试集对应的行，也称之为测试立方，相应的列向量被称之为位流。通常对于测试集可以使用行变换或者列变换到达数据压缩的效果。在本文中默认是使用列变换，也可以叫做位流变换。

变换拆分总共有三个步骤：分别为预处理、变换以及构造残分量集。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。变换过程主要是通过线性反馈移位寄存器（LFSR）完成，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），然后与原位流进行异或生成残分量。对测试集中的每一列以上文中提及的方式进行计算便可构造出残分量集。

在图3.1中测试集有i个向量，对应了J个输入端，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，原测试集被转化成为一个只包含0、-1和1确定位的矩阵，至此只需简单地进行向量之间的乘法运算就能确定列向量所对应的主分量。

图 3.1 测试立方和位流

向量分解的伪代码如下算法3.1所示，向量分解是为了提高残差集中0比特位的个数，对于单游程编码而言直接将无关位填充为0即可，拆分压缩技术对于双游程编码的压缩率提升较少，因为影响双游程编码压缩率的关键因素是跳变数，单纯地增加残分量集中的0也不一定会提高最终的压缩率。

算法 3.1 向量分解

```
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
```

如矩阵3.1所示，将列向量（1, 0, 1, 1, 1, X, 0）谱分析后，向量转换为（1, -1, 1, 1, -1, 0, -1）。然后通过线性反馈移位寄存器变换，得到[2 -2 0 0 0 0 -2 5] 这个系数向量。系数向量中系数最大值对应的列向量就是所需的主分量，对应于上图的（1，-1，1，1，1，-1，-1），当系数向量中的最大值有多个时，可以任选其中的任一列作为所求的主分量。在本例中如果将原测试集的无关位填充为0，原列向量

与主分量两两异或后最终所得的残差为（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长度和线性反馈移位寄存器的矩阵行数不一致，则需要在位流尾部填充无关位或者截断基向量即可。

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

3.2 预填充测试集

3.2.1 直接填充与策略填充

如果采用编码压缩的方式对测试集进行压缩，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码字0或者码字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，采取距离最大原则选取基向量生成主分量集。

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变换，有的采取最大相容类变换。通过哈达码变换可以取得较好的压缩增益，但是由于变换产生的列向量与原测试集无关，其获取的压缩率还有提升的空间。而采用最大相容类方式获取的压缩率并不高，由于使用KM算法或者贪婪算法结合关系矩阵法选取基向量，往往因为存在较多无关位，导致算法的时间复杂度过高。采用最大相容类方式确定的第一个基向量能与很多列向量相容，但继续往后选取，与其相容的会越来越少。基向量只能涵盖少数几列测试列向量，只有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。

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

通过一个例子来说明预填充方式的相关做法。上图给出原测试集，分别使用直接填充与策略填充的方式去除测试集中的无关位。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，本例采用FDR编码，直接将无关位填充为0即可，如上所示。如果采用双游程编码，要以原测试集的跳变数最少为原则对无关位进行填充，以EFDR编码为例，填充之后的测试集如下所示：

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

游程编码不仅要保证跳变数最少，还需要权衡连续出现0或者1的个数，比如跳变数均为2，向量11XXXXXXXXXXXXX000总共有18位,此向量有多种填充方式，一种是将无关位全部填充0，一种是将无关位填充为1，本例应该采取全0方式填充，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

基向量的选取方式至关重要，主分量集是由基向量集合确定的。若基向量集取自原测试集，由基向量集所产生的主分量集与原测试集中的向量相似度会更高，从而增加残差集中0的个数。在本文的第四章中使用聚类算法选取基向量，旨在使选出的基向量尽可能与原测试中的向量相似。为了更好地匹配原测试中已有的列向量，本文在原有的聚类算法上也做了相应的优化。

本章提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。第二步计算向量a与原测试集的每一列向量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。第三步，根据当前确定的基向量求出全部初始基向量，后续步骤dis的计算方式为：

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

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。对于不同的电路，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。X个列向量之间两两异或将产生$2^X - 1$个列向量，主分量集就是由些列向量确定的。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，本文按照哈达码矩阵变换的规则选取基向量个数。

3.3 实验结果与分析

为了体现预填充方法的可行性，本文对ISCAS’89[71]中大部分电路进行了验证，包括S5378、S9234、S13207、S15850、S38417、S38584 等电路，本章将挑选其中部分电路做具体分析，计算出其在FDR、EFDR、Huffman 等编码方式下对应的压缩率，并与其他压缩方法进行对比。

本文选取基向量的个数为$X \geq ceil(\log_2^n a)$ ，对于X列的基向量，两两异或最终

可得$2^x - 1$个列向量（集合Y）。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并构造出一个新的集合，即主分量集。实验结果如下表3.1 所示，首列为电路名称，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进行对比。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，因此为了最大化压缩率本文直接将无关位填充为0。

对于直接将无关位填充为0可以取得最佳的压缩率，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，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，提高了压缩率。

下表3.2-3.6为本方法与哈达码变换以及最大相容类的压缩率对比情况，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，电路s13207在不使用拆分压缩的方法下能取得80%的压缩率，可以说明当前电路所包含的无关位较多。从表中数据可得，直接编码所获取的压缩率普遍较低，使用哈达码变换、最大相容类以及本章方法均可较为明显地提高压缩率。除了s38584 电路，使用预填充的方法所取得的压缩率均高于其他方法，因此取得了最高的平均压缩率。

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编码和VIHC编码所对应的压缩率，采用本方法所达到的平均压缩率比哈达码变换高0.79%，比最大相容类高0.73%，因此本方法不仅对单游程编码有效，也同样适用于其他编码方式。

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，由上图可以得RL-Huff编码和ALT-FDR编码比哈达码变换所取得的压缩率平均高出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三种方法对比，均取得了较高的压缩率。

表3.7中的数据为本方法与其他压缩方法在不同电路下获取的压缩增益，本方法的压缩率为使用FDR编码所达，SVC 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，再进行编码，降低了编码的比特数量。由实验结果可知，本方法的平均压缩率高达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。从表中数据分析可知，不管使用何种压缩方法，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。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，本章基于变换压缩的研究方法提出了一种预填充方法，此方法预先对测试集进行处理，去除原测试集中的无关位。此方法有直接填充与策略填充两种模式，对于拆分压缩直接将无关位填充为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，使用本方法可以大大提高压缩率并减少了测试集的存储代价，节约测试时间成本。

第4章　使用聚类算法提高压缩率

聚类算法是器学习中的一类经典算法，常用于数据分类与数据特征提取，聚类即物以类聚，通过定义了一个分类标准，对原数据集进行划分。机器学习中涉及的算法众多，总体可分为基于监督学习与非监督学习两类，聚类算法初始并无任何标记，也无特征可以提取，对于不同的数据集，其采取的距离计算方式也是不一样的，因此被归于无监督学习一类。无监督学习的算法需要自动找到这些无标记数据集所具有的特征。

本文将聚类算法与数据压缩相结合，通过对原测试集进行分析，提取其中的特征，找到了一种提高压缩率的方法。在本文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，但是基向量的选取是毫无规律的，具有较大的随机性，如果使用聚类算法，对原测试集进行聚类，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，理论上来说可进一步提高残差集的压缩率。

4.1　kmeans聚类算法

Kmeans算法又叫做k-平均算法（英文：k-means clustering），是集简单和经典于一身的以距离作为考量标准的聚类算法。kmeans算法将距离相近的点归结于同一类,一般而言可以使用欧几里得距离、曼哈顿距离或者海明距离等作为“距离”的考量标准。kmeans算法是一种比较经典的聚类算法，其实现简单聚类效果较好，但是初始聚类中心的选取对结果的影响较大，往往需要通过一些特定的优化手段或者限定条件使其获得稳定而可靠的实验结果。

不同的聚类算法之间的聚类原理不同，约束条件也不一样，最终的聚类结果也不一样。当选取某个聚类算法时需要结合自身的使用场景，具体问题具体分析。有些聚类算法需要事先给定聚类个数，无法简单地通过迭代直接确定数据集的特征自动划分聚类数，比如kmeans 和kmeans++ 算法。有些聚类算法需要实现给定”密度”，比如dbscan聚类算法，虽然此方法可对数据集自动聚类但在算法运行的开始需要给定密度的具体值。其对n个样本点属于同一聚类的定义为，若样本中心在以d为半径的区域类有n个样本点，则这些样本可以划分为同样一类。因此聚类算法的运用是十分灵活的，如何更好的结合自身往往需要对不同聚类算法进行大量的实验，分析其中的原理。分类算法也可以用于数据的划分，但是不适合本实验，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。下文将着重介绍kmeans聚类算法，首先简要了解kmeans算法的相

关术语

簇：所有数据的点集合，簇中的对象是相似的。

质心：通过计算所有点的均值得到簇的质心。

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 4.1 聚类分布状态图

从上图4.1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)从原数据集合中随机确定k个点作为聚类中心。

(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)重复步骤3至步骤5，直到集合之间质心间的距离差低于某个阈值算法迭代结束。

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，本文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的均值向量其表达式为：

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

为了进一步理解kmeans算法，本章以二维坐标系为例，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。

因此有学者提出了kmeans++算法，其改进主要针对kmeans算法会随机选取初始质心。kmeans++在质心选取之前还进行了几个步骤，第一步从原数据集合中任意挑选出一个数据作为初始聚类中心。第二步，计算出集合中其他数据与初始聚类中心之间的距离，并将其相加，然后取随机值r，若r等于某一个数据与聚类中心之间的距离则将当前数据设置为新的聚类中心。第三步，重复上述步骤，直到确定所有的聚类中心。本文所使用的的kmeans++算法在第二步与传统做法有所不同，本人的做法为，若原数据集中有x个数据点，求出将x个数据点与聚类中心之间的欧式距离，选取其中最远的距离对应的数据作为第二个聚类中心，依次迭代获取最终的初始聚类中心。

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

本人使用了多种聚类算法进行相关实验，包括基于密度的dbscan聚类算法、基于机器学习的马尔科夫模型以及基于均值漂移相关的聚类算法。基于密度聚类的算法过于依赖密度的大小，密度又取决于邻域范围的设定以及邻域中列向量个

数的设定。实验结果表明kmeans++ 算法能较好地结合原测试集，提高压缩率。dbscan聚类算法虽然可以自动确定聚类个数，但是需要设置ϵ-邻域以及$MinPts$。本人经过多次尝试发现对于不同的电路，其ϵ-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++聚类算法计算出本文所需要的基向量。

在初始基向量的选取过程中，需要进行测试集预填充，由于本人采取拆分压缩的压缩方法，将原测试集中的无关位填充为0即可。本实验所需要的基向量即为将原测试集聚类之后所产生的聚类中心，下面是选取的具体步骤：为了方便表述，将聚类中心使用基向量来表述。

(1)将原测试集进行预填充，并从中任意挑选出一个列向量作为初基向量，由于是随机选取本实验直接使用第一列作为初始基向量。

(2)依次求出原测试集中其他列向量与，已挑选基向量之间的欧式距离，取距离最大的一列作为第二个选取的基向量。

(3)在原测试集中选取其他列向量作为下一个基向量，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)重复上述步骤直至确定k初始基向量，其中k的数量由我们原测试集的数据规模决定。

(5)计算出原测试集中每一列与初始基向量之间的距离。

(6)每个列向量均能计算出k个距离，k个距离就代表k聚类，以距离最小为原则将其划分到相应的聚类中。

(7)确定k个聚类之后，将每个聚类中的列向量相加之后求平均值，获取新的

聚类中心，即新的基向量。

(8)判断新选取的基向量计算的阈值是否小于等于事先给定的阈值，一旦满足迭代终止条件则，当前获取的基向量为本文所求，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，将其两两之间进行异或，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，选取汉明距离最小所对应的列作为主分量，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β_1(1,0,1,0,1)、β_2(1,1,0,0,1)、β_4(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2(1, 1, 0, 0, 1) 与矩阵Z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)。依此类推，直到计算出原测试集每一列所对应的主分量，构造出主分量集。最终将主分量集合原测试集进行异或，计算出相应残差集，获取压缩率。

4.3 实验结果与分析

本实验中选择的kmeans++算法，需要预先设定聚类中心的个数k，聚类数k定义得过少或者过多对结果均会产生一定的影响。针对此问题，本文分两个步骤进行了相关实验。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，本文对S5378、S9234、S13207、S15850、S38417、S38584等电路进行了实验，下文将对其中部分电路做具体描述，同时与其他压缩方式所得的压缩增益做对比。

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、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，第三列是对哈达码变换拆分之后的压缩率，第四列为对测试集预填充所获取的压缩率，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，采用本章方法计算的压缩率比对测试集直接编码平均高20%，比哈达码变换平均高2.4%，比预填充方法平均高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展示了EFDR 编码在不同压缩方法下所获取压缩增益，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，采用本章描述的方法所获取的压缩率比对测试集直接编码获取的压缩率平均高12.37%，比哈达码变换获取的压缩率平均高2.37%，比在预填充方法下获取的压缩率平均高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展示了ALT-FDR 编码和RL-Huff 编码在不同压缩方法下所获取压缩增益。由表可知，采用本章所提出的方法，ALT-FDR 编码和RL-Huff 编码所获取的压缩率比对测试集直接编码平均高12.42%和13.73%，比哈达码变换获取的压缩率平均高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，需要根据跳变数最少来进行编码，但是较其他压缩方法使用本方法获取的压缩增益依然是最高的。

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展示了VIHC 编码在不同压缩方法下所获取压缩增益，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，采用本章所提出算法获取的压缩率比对测试集直接编码获取的压缩率平均高19.25%，比哈达码变换获取的压缩率平均高2.39%，比预填充获取的平均压缩率高2.02%。

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，本方法对比于直接编码、哈达码变换、和预填充这三种方法，所获取的平均压缩率是最高的。

为了验证本方法确实有效，本文除了验证ISCAS’89中所包含的电路，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，第三列是直接编码获取的压缩率，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的压缩增益，但是使用本方法比直接编码所获取的压缩率平均提高了近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

下表4.9为各个大电路在EFDR、VIHC、ALT-FDR和RL-Huff编码下的压缩率，由表中数据可得，本章提及的压缩算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，并且在同等条件下相比于其他压缩方法也能取得更高的压缩率，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获取压缩率之间的关系。

通过对S5378、S9234、S13207、S15850、S38417、S38584等电路进行了实验，下表4.10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个聚类数残差集所能达到的压缩率。

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中的数据大致绘制出如下的折线图4.2，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 4.2 FDR编码方式折线图

下表4.11、图4.3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 4.3 VIHC编码方式折线图

下表4.12-4.13以及图4.4、4.5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	52.58	68.59	71.26	71.79	73.59	74.91
s9234	47.26	66.05	67.07	67.05	68.17	70.18
s13207	82.94	91.81	92.82	93.23	93.79	94.62
s15850	67.35	80.57	81.73	81.77	83.65	84.34
s38417	63.32	74.02	74.28	74.71	75.57	76.17
s38584	62.40	73.70	74.74	76.13	77.42	78.78
平均	62.64	75.79	76.98	77.45	78.70	79.83

图 4.4 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 4.5 AFDR编码方式折线图

从上述的折线图可知，随着选取基向量个数的增加，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的相关实验，同时也使用当前压缩方法对大电路测试集进行了压缩。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

测试立方压缩旨在通过提高压缩率，降低测试时间、节约压缩成本。测试立方由一系列0、1以及无关位X组成，其中X既可以被填充为0也可以被填充为1，而不影响原测试集的故障覆盖率，因此包含较多无关位的测试集往往会获得较高的压缩率。本章将使用位翻转算法在不影响故障覆盖率的前提下，通过增加原测试集的无关位提高压缩率。

5.1　相关概念

5.1.1 故障检测冗余度

故障检测冗余度通俗的来说就是同一个故障在测试过程中被多次检测。比如，数据冗余是指某个数据在特定情况下反复出现，那么故障检测冗余度类也是类似的。测试集由ATPG产生，初始测试集中包含大量数据，其中存在众多的无关位。由于可以将部分无关位会被转化成为确定位，测试集的规模会大大减小，但测试立方整体的故障覆盖率保持不变。

5.1.2 位翻转

本文主使用拆分压缩的方式提高压缩率，拆分压缩会将原测试立方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。假设主分量的故障覆盖率为A，测试立方的总故障覆盖率为C，剩余故障覆盖率为$C-A$。通过故障模拟之后原测试立方只要能检测出剩余故障即可，为了避免相同故障被反复检测可以将原测试立方中特定的确定位翻转为X，在总故障覆盖率不变的情况下，提高压缩率。这个过程被称之为位翻转。

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。

下图5.1为位翻转应结合本实验的压缩过程：图中可以看出原测试立方的故障覆盖率达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%，剩余故障为30%。在保证故障覆盖率的条件下，将原测试集部分确定位翻转为无关位并获得新测试集，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]中提及了多轮位翻转算法，但是由于每一轮翻转，均会产生代价，并且硬件代价与实现相对比较困难，本人化繁为简直接使用一轮压缩，然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

若C代表原测试集合的故障覆盖率，对原测试集进行位填充之后，拆分为主分量集和残差集，A代表主分量集合所能达到的故障覆盖率，假设原测试集合的故障覆盖率B，只需要满足$B > C - A$即可，下文将基于这种思路写出位翻转算法的伪代码。

算法5.1为翻转测试集的基本过程，首先将原测试立方中的确定位一一翻转，并进行故障模拟，如果不影响故障覆盖率则直接翻转，反之还原，若故障模拟的时间为t，翻转一位的时间为s，假设原测试集合中由k个确定位，那么算法的时间复杂度为kts，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。

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

5.3 实验结果与分析

为了说明kmeans++算法结合位翻转算法确实有效，本文对S5378、S9234、S13207、S15850等电路进行了实验。本章将挑选其中部分电路做具体描述，本文选取选取了FDR、EFDR、ALT-FDR编码方式对变换拆分之后的残差集进行压缩，同时与直接使用位翻转算法所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翻转所达到的压缩率，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

从上表可以看出本方法能极大地提升残差集的压缩率，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

位翻转方法旨在通过将确定位翻转为无关位来提高压缩率。由于主分量集自身可以检测出部分故障，原测试集只需检测剩余故障即可。通过故障模拟，原测试将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新的残差集，此时的残差集包含的0比特位大大增加，能进一步提高压缩率。使用kmeans++算法集合位翻转的压缩方法，能使压缩率在第四章的基础上提高7%。

结论与展望

随着科学技术的进步，超大规模集成电路的发展速度日新月异。芯片集成度以及复杂度的急剧上升增加了制造合格芯片的困难。为了确保芯片的故障覆盖率达标，需要大量的测试数据对其进行检测。庞大的测试数据增加了硬件代价以及测试时间，一种卓有成效的方法就是对测试数据进行压缩。近年来有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，进一部提高压缩率。主分量的选取直接影响最终的压缩率，本文在拆分压缩的基础上，对如何生成基向量展开了研究。

第一章绪论，介绍了集成电路发展的背景以及电路测试的相关先修知识，同时就国内外研究现状进行分析，最后总结本文的组织结构。第二章主要就常用的压缩方法做了详细地介绍。第一部分讲解了编码压缩技术，包括游程编码、字典编码以及统计编码。第二部分就基于线性解压缩和基于广播扫描两种非编码压缩方法做出了相应的分析与阐述，最后介绍了拆分压缩技术和哈达码变换，并对哈达码变换的优缺点进行了分析。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。

本文针对测试集自身特性，在拆分压缩技术中，针对基向量的生成算法提出了两个创新点主要有：

（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%。通过和大多数压缩方法进行对比表明，对于大多数基准电路，使用本方法均可以获得较好的效果。

（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。该方法通过步骤（1）消除原测试集中的无关位，然后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，随后取当前聚类列向量每一位的均值作为聚类中心向量，并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，然后对主分量集进行故障模拟检测出电路的部分故障，然后对原测试集也进行故障模拟，将主分量集能检测出故障对应的确定位转化成为无关位，并生成新的测试集。最后将新的原测试集与主分量集异或，得到新的残分量集，提升压缩率。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相比于步骤（2）提高了7%。

针对在拆分压缩技术中的基向量的生成算法研究，文中提出的三种方法虽然在取得了一定的成果，但仍然有不足和和厄待改进之处。现总结如下：

(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还有很大的空间，可能存在某种聚类算法可以直接确定原测试集聚类个数并且硬件代价也相对较小。

(3)本文第五章通过将kmeans++聚类算法结合位翻转算法来提高压缩率，此方法的主要思想是：对主分量集进行故障模拟检测出电路的部分故障，然将原测

试集部分确定位转化为无关位，从而提升压缩率。虽说针对所有基准电路，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，因此对翻转算法进一步优化增加可翻转的确定位，是一个可以研究的突破点，比如根据主分量集合的确定位情况设计相应的翻转算法，从而提高残分量的压缩率。

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] 何山. BIST原理和设计方法. 中国造船工程学会，2008,9（1）：65–67

[35] 陈小保,邢座程,李少青. 星载处理器流水线的BIST 设计方法. 中国计算机学会,2007:302–305

[36] Chandra A, Chakrabarty K. Test Resource Partitioning for SOCs. IEEE Design & Test of Computers, 2001, 18(5):80–91

[37] Chakrabarty K, Iyengar V, Chandra A. Test Resource Partitioning. est Resource Partitioning for System-on-a-Chip. Springer US, 2002, 619–622

[38] Wurtenberger A, Rosinger P, Al-Hashimi B M, et al. Cost model driven test resource partitioning for socs. Electronics Letters, 2006, 42(16):915–916.

[39] A. Jas and N.A. Touba. ”Test Vector Compression via Cyclical Scan Chains and Its工程硕士学位论文Application to Testing Core-Based Designs,” Proc. Int’l Test Conf. (ITC 98), IEEE CS Press, 1998, pp. 458–464

[40] Anshuman,Chandra Krishnendu,Chakrabarty. ”Efficient test data compression and decompression for system-on-a-chip,” using internal scan chains and golomb codingMunich,2001:145–149..

[41] 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.

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

[43] Usha S. Mehta, Kankar S. Dasgupta2, Niranjan M. Hamming Distance Based Reordering and Columnwise Bit Stuffing with Difference Vector: Devashrayee.A Better Scheme for Test Data Compression with Run Length Based Codes.India,2110:33–38

[44] Barber M, Loranger M. Test Resource Partitioning. IEEE Design & Test of Computers．2000,17(3):126–132

[45] WEI LI,XUAN YANG,ZIBIN DAI. Research on design of a reconfigurable parallel structure targeted at LFSR[C]. 2011:59–63

[46] Ruan X, Katti R S. Data-Independent Pattern Run-Length Compression for Testing Embedded Cores in SoCs. IEEE Transactions on Computers, 2007, 56(4):545–556.

[47] Armstrong D B. A Deductive Method for Simulating Faults in Logic Circuits Computers IEEE Transactions on, 1972, C-21 (5):464–471

[48] 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. IEEE,2002:449–452 vol.2

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

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

[51] Omaña, M, Rossi, D, Fuzzi F, et al. Novel approach to reduce power droop during scan-based logic BIST. IEEE, 2013:1–6

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

[53] Gonciari PT,A1 Hashimi B,Nicolici N．Improving compression ratio,area overhead and test application time for system-on-a-chip test data compression decompression．Design,Automation & Test in Europe Conference,2002,604–611

[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，感谢陪我一起走过这段岁月的人儿，愿你们身体健康事事顺心，也同样希望你们的生活中，因为有我的存在可以带给你们一些欢乐与帮助。

首先，由衷地感谢我的导师xxx对我的悉心指导和多方关怀！老师对我的帮助不仅仅是学业上的指导，更是设计到了生活中的方方面面，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xxx，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xxx老师、xxx老师、xxx博士、xxx博士给我的指导与帮助。感谢师弟xxx，遇到问题时和你探讨总能激发灵感。感谢幽默室友xxx，因为你让我的生活变得十分有趣。感谢大佬室友xxx，是你让我了解到自己离优秀的距离还相当遥远。感谢好朋友xxx，每次和你交流我都能获得灵感。还要特别感谢xxx同学对我的帮助。

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] Fifth author. Intrusion Detection for In-vehicle Network by Using Single GAN in Connected Vehicles[J]. Journal of Circuits, Systems, and Computers, .(has been accepted).

[2] second author. 基于测试集主成分的变换–拆分法提高编码压缩率.电子测量与仪器学报.

[3] 软件著作权：xxx. PRE聚源集成电路测试向量压缩软件. 登记号：2020SR0199298.