# 第三章 用预填充策略处理测试集

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](https://zh.wikipedia.org/wiki/%E6%95%B8%E5%AD%B8" \o "数学)转换后[映射](https://zh.wikipedia.org/wiki/%E6%98%A0%E5%B0%84" \o "映射)至另一[值域](https://zh.wikipedia.org/wiki/%E5%80%BC%E5%9F%9F" \o "值域)后再进行编码处理。常用于[音频](https://zh.wikipedia.org/wiki/%E9%9F%B3%E9%A2%91" \o "音频)信号编码和[图像](https://zh.wikipedia.org/wiki/%E5%9B%BE%E5%83%8F" \o "图像)或[视频](https://zh.wikipedia.org/wiki/%E8%A7%86%E9%A2%91" \o "视频)信号编码。变换编码经常与[量化](https://zh.wikipedia.org/wiki/%E9%87%8F%E5%8C%96_(%E4%BF%A1%E5%8F%B7%E5%A4%84%E7%90%86)" \o "量化 (信号处理))一起使用，进行[有损数据压缩](https://zh.wikipedia.org/wiki/%E6%9C%89%E6%8D%9F%E6%95%B0%E6%8D%AE%E5%8E%8B%E7%BC%A9" \o "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。



图 3.1 测试立方和位流

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。

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]。

[1 -1 1 1 1 0 -1]×=[2 -2 0 0 0 0 -2 5]

图3.2

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：

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3.1VectorDecomposition(T) |
| /\*  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  L = [ L, –L]  for i from 1 to M  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  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)  Return ProminentComponentSet, 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 = 

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 = 

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 = 

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。，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

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使用，其中n代表当前测试集的行数，为什么选择这个公式，因为需要将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 行、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 结果分析

为了验证预填充方法的有效性，我们对 ISCAS’89[57]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，那么对于X列的基向量，两两异或最终可得列向量的集合Y，主分量就是从集合Y中中选取的，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3.2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 | (136,1464) | 255 | 66.09 | 78.05 |
| 平均 |  |  | 64.66 | 78.62 |

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、3.3、3.4、3.5、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、3.3、3.4、3.5、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[48] | I-EFDR[59] | LHBE[60] | CCPRL[32] | 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。