我们使用的是列变换，你懂为什么吗，首先，你从之前的测试向量中每一次都是读取一行，然后再将每一行重新组装成为一列一列的向量，那么我们计算的是每一列之间的距离，也就是说，一列一列都是一个个的位流，我们是将一行一行的数据送入到被测电路。所以说这个是不一样的，你要将自己的想法调整一下，数据如何输入，如何变换。

取系数向量中所有系数绝对值最大的那个系数相对应的基向量作为主分量，当有多个最大的系数绝对值相等时，则随意让一个相对应的基向量函数作为主分量。因此可以选用（1, 0, 1, 1, 1, 0, 0）作为原位流（1, 0, 1, 1, 0, X, 0）的主分量。假设将该原位流中的X取值为0时，这时主分量与原位流有6位值是一样的，此时残分量为（0, 0, 0, 0, 1, 0, 0），只有1个非0位。在变换时，如果原位流与LFSR矩阵大小不匹配，需要在原位流末端添X补齐，也可以截断基向量的部分使基向量长度与原位流长度一致。假设实验需要不止一个主分量，此时可以按照系数向量中系数绝对值从大到小排列的原则来多选几个基向量来表示原位流。向量分解的具体过程见算法3.2。注意，我们对LFSR矩阵进行了取反。

|  |
| --- |
| 算法3.2 VectorDecomposition(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 ) //将测试集T中的0换成-1，X换成0，并补齐大小  L = [ L, –L] //将原LFSR矩阵与取反后的LFSR矩阵拼成一个新矩阵  for i from 1 to M  p = L\*T[1:N, i] //将新矩阵与测试集中的第i列相乘得到系数矩阵p  k = max( p ) //得到系数矩阵p中绝对值最大的数的索引k  Prominent = L[k] //提取主分量  Residual = T[1:N, i] XOR Prominent //原位流与主分量异或得到残分量  ProminentComponentSet.add(Prominent)  ResidualComponentSet.add(Residual)  Return ProminentComponentSet, ResidualComponentSet //返回主分量集，残分量集 |

减少残分量集中1的个数是向量分解的全部目的。对于单游程编码而言，减少1的数目就相当于减少了游程的个数，从而直接减少了编码后的测试数据量。