(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102640422 B
(45)授权公告日 2015.02.11

(21)申请号 201080037946.5
(22)申请日 2010.08.19
(30)优先权数据
61/235,285 2009.08.19 US
12/604,773 2009.10.23 US
61/257,146 2009.11.02 US
61/353,910 2010.06.11 US
12/859,161 2010.08.18 US
(85)PCT国际申请进入国家阶段日
2012.02.20
(86)PCT国际申请的申请数据
PCT/US2010/046027 2010.08.19
(87)PCT国际申请的公布数据
WO2011/022555 EN 2011.02.24
(73)专利权人 高通股份有限公司
地址 美国加利福尼亚州
(72)发明人 M·G·卢比 M·A·肖克罗拉西 L·C·明德
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 李小芳
(51)Int.Cl.
H03M 13/37(2006.01)
(56)对比文件
US 2007/0195894 A1,2007.08.23,说明书第0028段,0133-0162段、图1-16.
CN 1806392 A,2006.07.19,全文.
审查员 金霞

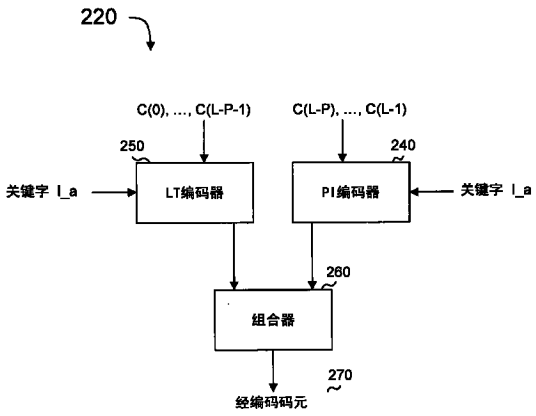
权利要求书3页 说明书81页 附图25页

(54)发明名称

为编码和解码过程采用使码元永久钝化的FEC码的方法和装置

(57)摘要

提供了对多个经编码码元的编码,其中根据从第一中间码元集合生成的第一码元与从第二中间码元集合生成的第二码元的组合生成经编码码元,每个集合具有至少一个不同的编码参数,其中这些中间码元是基于源码元集合生成的。还提供了解码数据的方法,其中从收到经编码码元集合解码一组中间码元,这些中间码元被组织成第一和第二码元集合以进行解码,其中第二集合中的中间码元被永久钝化以调度解码过程从经编码码元恢复中间码元,其中从解码出的中间码元集合恢复至少一些源码元。



CN 102640422 B

1. 一种经由一个或更多个能够输出电子信号的发射机电子地传送数据的方法，其中待传送的数据由有序源码元集合表示且所述数据是作为表示所述电子信号的至少一部分的经编码码元序列来传送的，所述方法包括：

以电子可读形式获得所述有序源码元集合；

从所述有序源码元集合生成一组中间码元，其中所述源码元是能从该组中间码元再生的；

指定中间码元集合以使得每个中间码元被指定为所述中间码元集合之一的成员且至少存在第一中间码元集合和第二中间码元集合，并且其中每个中间码元集合具有相关联的相异的编码参数且具有至少一个中间码元作为成员；以及

其中所述第一中间码元集合被指定为用于置信传播解码的码元且所述第二中间码元集合被指定为针对置信传播解码被钝化的码元；

生成多个经编码码元，其中经编码码元是从所述中间码元中的一个或更多个中间码元生成的，其中至少一个经编码码元是直接或间接从选自多个所述中间码元集合的多个中间码元生成的。

2. 如权利要求 1 所述的方法，其特征在于，每个经编码码元是根据从所述第一中间码元集合中的一个或更多个中间码元生成的第一码元与从所述第二中间码元集合中的一个或更多个中间码元生成的第二码元的组合生成的。

3. 如权利要求 2 所述的方法，其特征在于，所述相异的编码参数至少包括相异的度分布，以使得每个经编码码元是根据从具有第一度分布的所述第一中间码元集合中的一个或更多个中间码元生成的第一码元与从具有不同于所述第一度分布的第二度分布的所述第二中间码元集合中的一个或更多个中间码元生成的第二码元的组合生成的。

4. 如权利要求 2 所述的方法，其特征在于，所述第一码元是使用向所述第一中间码元集合应用的链式反应编码过程生成的。

5. 如权利要求 2 所述的方法，其特征在于，所述第二码元是从所述第二中间码元集合随机选取的固定数目个码元的异或。

6. 如权利要求 2 所述的方法，其特征在于，所述第二码元是从所述第二中间码元集合随机选取的第一数目个码元的异或，并且其中所述第一数目取决于等于为了生成所述第一码元而从所述第一集合选取的码元数目的第二数目。

7. 如权利要求 2 所述的方法，其特征在于，所述组合是所述第一码元与所述第二码元的异或。

8. 如权利要求 1 所述的方法，其特征在于，所述中间码元包括所述有序源码元集合以及从所述有序源码元集合生成的冗余源码元集合。

9. 如权利要求8所述的方法，其特征在于，所述冗余码元中的至少一些是使用GF[2]操作生成的，且其他冗余码元是使用 GF[256] 操作生成的。

10. 如权利要求 1 所述的方法，其特征在于，所述中间码元是在编码期间使用解码过程从所述源码元生成的，其中所述解码过程基于所述中间码元与所述源码元之间的线性关系组。

11. 如权利要求10所述的方法，其特征在于，所述线性关系中的至少一些是GF[2]上的关系，且其他线性关系是 GF[256] 上的关系。

12. 如权利要求 1 所述的方法，其特征在于，能从给定的有序源码元集合生成的相异经编码码元的数目独立于该有序集合中的源码元数目。

13. 如权利要求 1 所述的方法，其特征在于，为了生成经编码码元而执行的码元操作的平均次数受独立于该有序集合中的码元数目的常数限制。

14. 如权利要求 1 所述的方法，其特征在于，所述第一码元集合在大于所述第二码元集合的数量级以上。

15. 一种从源接收数据的方法，其中所述数据是通过分组通信信道在目的地接收的，并且其中所述数据能由从表示从所述源发送给所述目的地的所述数据的有序源码元集合推导出的经编码码元集合表示，所述方法包括 ：

获得收到经编码码元集合 ；

从所述收到经编码码元集合解码一组中间码元 ；

将所述中间码元中的每一个与中间码元集合相关联，其中所述中间码元被关联到至少两个集合中，且其中第一集合包括被所述源指定为用于置信传播解码的码元且第二集合包括被所述源被指定为永久钝化码元集合以调度用于从所述收到经编码码元恢复所述中间码元的解码过程 ；以及

根据所述解码过程从所述中间码元集合恢复所述有序源码元集合的至少一些源码元。

16. 如权利要求 15 所述的方法，其特征在于，所述解码过程至少包括 ：第一解码阶段，其中取决于第二永久钝化码元集合以及为第一码元集合的子集的第三动态钝化码元集合生成简化经编码码元集合 ；以及第二解码阶段，其中所述简化经编码码元集合被用来解码所述第二永久钝化码元集合和所述第三动态钝化码元集合 ；以及第三解码阶段，其中解码出的第二永久钝化码元集合和第三动态钝化码元集合以及所述收到经编码码元集合被用来解码所述第一码元集合中的至少一些剩余中间码元。

17. 如权利要求 16 所述的方法，其特征在于，所述第一解码阶段使用置信传播解码结合钝化解码，和 / 或所述第二解码阶段使用高斯消元法。

18. 如权利要求 16 所述的方法，其特征在于，所述第三解码阶段使用回代、或者后向扫描继以前向扫描。

19. 如权利要求 16 所述的方法，其特征在于，考虑所述第三动态钝化码元集合中的码元数目少于源码元数目的 10％和 / 或少于所述第二永久钝化码元集合中的码元数目的 10％，所述解码过程对所述第三动态钝化码元集合进行操作。

20. 如权利要求 15 所述的方法，其特征在于，所述收到经编码码元作为 LDPC 码生成的码元或 Reed-Solomon 码生成的码元被操作。

21. 如权利要求 15 所述的方法，其特征在于，所述收到经编码码元集合的每个收到经编码码元就如是从所述第一码元集合中的一个或更多个码元生成的第一码元与从所述第二码元集合中的一个或更多个码元生成的第二码元的组合那样被操作。

22. 如权利要求 21 所述的方法，其特征在于，每个收到经编码码元就如所述组合是所述第一码元与所述第二码元的异或那样被操作。

23. 如权利要求 21 所述的方法，其特征在于，每个收到经编码码元就如所述第二码元是从所述第二集合随机选取的固定数目个码元的异或那样被操作。

24. 如权利要求 21 所述的方法，其特征在于，每个收到经编码码元就如所述第二码元

是从所述第二集合随机选取的第一数目个码元的异或那样被操作，其中码元的所述第一数目取决于为了生成所述第一码元而从所述第一集合选取的码元的第二数目。

25. 如权利要求 21 所述的方法，其特征在于，所述解码过程就像所述第一码元是基于链式反应码从所述第一码元集合选取的那样进行操作。

26. 如权利要求 15 所述的方法，其特征在于，所述解码过程就像所述第二永久钝化码元集合的大小与源码元数目的平方根成比例那样进行操作。

27. 如权利要求 15 所述的方法，其特征在于，所述解码过程就像所述中间码元包括所述有序源码元集合以及从所述有序源码元集合生成的冗余码元集合那样进行操作。

28. 如权利要求 27 所述的方法，其特征在于，所述解码过程就像所述冗余码元中的至少一些是使用 GF[2] 操作生成的且其他冗余码元是使用 GF[256] 操作生成的那样进行操作。

29. 如权利要求 15 所述的方法，其特征在于，所述解码过程就像所述中间码元包括所述有序源码元集合那样进行操作。

30. 如权利要求 15 所述的方法，其特征在于，所述解码过程就像所述中间码元是使用解码过程基于所述中间码元与所述源码元之间的线性关系组从所述源码元生成的码元那样进行操作。

31. 如权利要求 30 所述的方法，其特征在于，所述解码过程就像所述线性关系中的至少一些是 GF[2] 上的关系且其他线性关系是 GF[256] 上的关系那样进行操作。

32. 如权利要求 15 所述的方法，其特征在于，所述解码过程就像能接收到的不同的可能经编码码元的数目独立于所述有序集合中的源码元数目那样进行操作。

33. 如权利要求 15 所述的方法，其特征在于，为了从所述收到经编码码元集合解码所述源码元集合要执行的码元操作的平均次数受常数乘以源码元数目限制，其中所述常数独立于所述源码元数目。

34. 如权利要求 15 所述的方法，其特征在于，所述解码过程就像所述第一码元集合中的码元数目在大于所述第二永久钝化码元集合中的码元数目的数量级以上那样进行操作。

35. 如权利要求 15 所述的方法，其特征在于，所述解码过程进行操作以使得从 N ＝ K+A 个经编码码元的集合恢复所有 K 个源码元的集合对于某个 K、N 和 A 至少具有 1-(0.01)^(A+1) 的下限的成功概率，其中 A ＝ 0、1 或 2，其中所述下限独立于源码元数目。

为编码和解码过程采用使码元永久钝化的 FEC 码的方法和装置

[0001] 交叉引用

[0002] 本申请是于 2009 年 10 月 23 日提交、发明人名为 M. Amin Shokrollahi 等人且题为“Method and Apparatus Employing FEC Codes with Permanent Inactivation of Symbols for Encoding and Decoding Processes(为编码和解码过程采用使码元永久钝化的 FEC 码的方法和装置)”的美国专利申请号 12/604,773 的部分延续,且进一步要求发明人皆为 M. Amin Shokrollahi 等人且皆题为“Method and Apparatus Employing FEC Codes with Permanent Inactivation of Symbols for Encoding and Decoding Processes(为编码和解码过程采用使码元永久钝化的 FEC 码的方法和装置)”的以下临时申请的优先权:于 2010 年 6 月 11 日提交的美国临时专利申请号 61/353,910、于 2009 年 11 月 2 日提交的美国临时专利申请号 61/257,146、以及于 2009 年 8 月 19 日提交的美国临时专利申请号 61/235,285。以上引用的每篇临时和非临时申请藉此通用地通过援引纳入于此。

[0003] 以下参考的全部内容通用地通过援引纳入于此:

[0004] 1) 授予 Michael G. Luby 的题为“Information Additive Code Generator and Decoder for Communication Systems(用于通信系统的信息加性码生成器和解码器)”的美国专利号 6,307,487(下文称为“Luby I”);

[0005] 2) 授予 Michael G. Luby 的题为“Information Additive Group Code Generator and Decoder for Communication Systems(用于通信系统的信息加性群码生成器和解码器)”的美国专利号 6,320,520(下文称为“Luby II”);

[0006] 3) 授予 M. Amin Shokrollahi 的题为“Multi-Stage Code Generator and Decoder for Communication Systems(用于通信系统的多级码生成器和解码器)”的美国专利号 7,068,729(下文称为“Shokrollahi I”);

[0007] 4) 授予 M. Amin Shokrollahi 的题为“Systems and Processes for Decoding a Chain Reaction Code Through Inactivation(用于通过钝化来解码链式反应码的系统和过程)”的美国专利号 6,856,263(下文称为“Shokrollahi II”);

[0008] 5) 授予 M. Amin Shokrollahi 的题为“Systematic Encoding and Decoding of Chain Reaction Codes(链式反应码的系统编码和解码)”的美国专利号 6,909,383(下文称为“Shokrollahi III”);

[0009] 6) 授予 Michael G. Luby 和 M. Amin Shokrollahi 且题为“In-Place Transformations with Applications to Encoding and Decoding Various Classes of Codes(应用于编码和解码各类码的原地变换)”的美国专利公开号 2006/0280254(下文称为“Luby III”);

[0010] 7) 发明人名为 M. Amin Shokrollahi 且题为“Multiple Field Based Code Generator and Decoder for Communications Systems(用于通信系统的基于多域的码生成器和解码器)”的美国专利公开号 2007/0195894(下文称为“Shokrollahi IV”)。

发明领域

[0011] 本发明涉及在通信系统中编码和解码数据，尤其涉及以高效率方式编码和解码数据以计及所传达数据中的差错和间隙的通信系统。

[0012] 发明背景

[0013] 用于通过通信信道在发送方与接收方之间传输文件的技术是许多文献的主题。较佳地，接收方希望以某种确定性程度接收由发送方在信道上传送的数据的确切副本。在信道不具有理想保真度的情况下（这囊括了几乎所有在物理上可实现的系统），所关心的是如何处理传输中丢失和混淆的数据。丢失的数据（擦除）通常比讹误的数据（差错）更容易处理，因为接收方不能总是辨别出何时讹误的数据是接收出错的数据。已开发出许多纠错码来纠正擦除和 / 或差错。典型地，所使用的特定码是基于关于数据正藉以传送的信道的失真以及正被传送的数据的特性的某些信息来选取的。例如，在知晓信道具有长失真期的场合，阵发差错码对于此应用可能是最合适的。在预期仅有短且不频繁的差错的场合，简单的奇偶校验码可能是最佳的。

[0014] 如本文中所使用的，“源数据”是指在一个或更多个发送方处可用以及使用接收机通过从有或无差错和 / 或擦除等的所传送序列进行恢复来获得的数据。如本文中所使用的，“经编码数据”是指被传达且能被用来恢复或获得源数据的数据。在简单情形中，经编码数据是源数据的副本，但若接收到的经编码数据（例如，由于差错和 / 或擦除）不同于所传送的经编码数据，则在该简单情形中，在缺少关于源数据的附加数据的情况下，源数据可能无法完全恢复。传输可通过空间或时间。在更复杂的情形中，经编码数据是基于源数据在变换中生成的，并从一个或更多个发送方传送给接收方。若发现源数据将成为经编码数据的一部分，则编码被称为“系统的”。在系统编码的简单示例中，关于源数据的冗余信息被追加到源数据末尾以形成经编码数据。

[0015] 如本文中所使用的，“输入数据”是指 FEC（前向纠错）编码器装置或 FEC 编码器模块、组件、步骤等（“FEC 编码器”）的输入端出现的数据，而“输出数据”是指 FEC 编码器的输出端出现的数据。相应地，将期望输出数据出现在 FEC 解码器的输入端，且将期望该 FEC 解码器基于它处理的该输出数据而输出该输入数据或其对应数据。在一些情形中，输入数据是或者包括源数据，且在一些情形中，输出数据是或者包括经编码数据。在其他情形中，发送方设备或发送方程序代码可包括一个以上 FEC 编码器，即源数据在一系列多个 FEC 编码器中被变换成经编码数据。类似地，在接收方，可存在一个以上应用于从收到经编码数据生成源数据的 FEC 解码器。

[0016] 数据可被视为被划分成码元。编码器是从源码元或输入码元的序列生成经编码码元或输出码元的计算机系统、设备、电子电路或诸如此类，且解码器是从接收到或恢复出的经编码码元或输出码元恢复源码元或输入码元的序列的配对物。编码器和解码器在时间和 / 或空间上被信道分开，且任何接收到的经编码码元可能与相应的所传送经编码码元不完全相同，并且它们可能不按与它们被传送的顺序完全相同的顺序被接收。码元的“大小”可以用比特来度量，无论该码元实际上是否被断成比特流，其中当码元是选自有 2^M 个码元的字母表时，码元具有 M 比特的大小。在本文中的许多示例中，码元是以字节来度量的，且码可能位于有 256 种可能性的域上（有 256 种可能的 8 比特模式），但是应理解，可以使用不同的数据度量单位，且以各种方式来度量数据是公知的。

[0017] Luby I 描述了使用诸如链式反应码之类的代码以计算高效、存储器高效和带宽高效的方式解决纠错。由链式反应编码器产生的经编码码元的一种属性在于接收方只要已接收到足够的经编码码元就能够恢复出原始文件。具体而言，为了以高概率恢复原始的 K 个源码元，接收方需要大约 K+A 个经编码码元。

[0018] 给定情形的“绝对接收开销”由值 A 表示，而“相对接收开销”可演算为比率 A/K。绝对接收开销是除了信息论最小数据量以外还需要接收多少额外数据的度量，且它可取决于解码器的可靠性并且可作为源码元数目 K 的函数而变化。类似地，相对接收开销 A/K 是除了信息论最小数据量以外还需要接收多少额外数据相对于正被恢复的源数据的大小的度量，且也可取决于解码器的可靠性并且可作为源码元数目 K 的函数而变化。

[0019] 链式反应码对于基于分组的网络上的通信极其有用。然而，它们有时可能是计算相当密集的。若在使用链式反应或其他无码率 (rateless) 码进行编码的动态编码器之前使用静态编码器来编码源码元，则解码器或许能够更频繁或更容易地进行解码。例如，此类解码器在 Shokrollahi I 中示出。在其中示出的示例中，源码元是静态编码器的输入码元，静态编码器产生的输出码元作为动态编码器的输入码元，动态编码器产生的输出码元是经编码码元，其中动态编码器是能以相对于输入码元数目并非为固定比率的数量生成数个输出码元的无码率编码器。静态编码器可包括一个以上固定码率编码器。例如，静态编码器可包括汉明 (Hamming) 编码器、低密度奇偶校验 (“LDPC”) 编码器、高密度奇偶校验 (“HDPC”) 编码器和 / 或诸如此类。

[0020] 链式反应码具有以下属性 ：当在解码器处从收到码元恢复出一些码元时，这些码元或许能被用来恢复附加码元，附加码元又可被用来恢复更多码元。较佳地，解码器处码元求解的链式反应可持续进行，以使得在收到码元池被用尽之前恢复出所有合意码元。较佳地，执行链式反应编码和解码过程的计算复杂度较低。

[0021] 解码器处的恢复过程可涉及确定接收到哪些码元，创建可将原始输入码元映射到接收到的那些经编码码元的矩阵，随后将该矩阵求逆并执行该逆矩阵与收到经编码码元矢量的矩阵乘法。在典型的系统中，此举的蛮力实现会耗费过多计算努力和存储器需求。当然，对于特定的收到经编码码元集合，可能无法恢复出所有原始输入码元，即使在有可能恢复出所有原始输入码元的场合，要计算出结果可能也是计算昂贵的。

[0022] ShokrollahiII 描述了被称为“钝化 (inactivation)”的办法，其中解码在两个步骤中发生。在第一步骤中，解码器观察它具有哪些可用的收到经编码码元、矩阵可能看似如何，并至少大致确定在给定这些收到经编码码元的情况下将允许链式反应过程完成的解码步骤序列。在第二步骤中，解码器根据所确定的解码步骤序列来运行链式反应解码。这可以按存储器高效的方式 (即，比存储器效率较低的过程要求较少存储器存储进行操作的方式) 进行。

[0023] 在钝化办法中，第一解码步骤涉及操纵矩阵或其等效物以确定能解出的某个数目的输入码元并且当该确定中断时，将输入码元之一指定为“钝化码元”并通过假定该钝化码元其实被解出了来继续该确定过程，随后在结束时，使用高斯消元法或其他某种方法对远小于原始解码矩阵的矩阵求逆来求解这些钝化码元。使用该确定，可对收到经编码码元执行链式反应序列以得到恢复出的输入码元，恢复出的输入码元可以是所有原始输入码元或合适的原始输入码元集合。

[0024] 对于对解码器强加了严格约束的一些应用，诸如在解码器为具有有限存储器和计算能力的低功率设备的场合，或者诸如当对可允许的绝对或相对接收开销有严格约束时，可关于以上描述的钝化办法指出改善的方法。

[0025] 另外，用于在受最小子码元大小约束的情况下将文件或大数据块划分成尽可能少的源块、且随后在受最大子块大小约束的情况下将源块拆分成尽可能少的子块的方法可能是有用的。

[0026] 发明简要概述

[0027] 根据依照本发明各方面的编码器的一个实施例，发送方的编码器在通信信道上从一个或更多个发送方向一个或更多个接收方传送有序源码元集合，其中该编码器生成包括从源码元生成的多个经编码码元的待发送数据。在第一步骤，使用可逆的方法从源码元生成中间码元，即，还存在用于从中间码元生成源码元的逆方法。在另一步骤，将中间码元划分成第一中间码元集合和第二中间码元集合，其中第一中间码元集合中有至少一个中间码元且第二中间码元集合中有至少一个中间码元，并且从来自这两个集合中每个集合的至少一个中间码元生成至少一个经编码码元。在一些变型中，存在两个以上集合。

[0028] 在一些实施例中，生成第一和第二临时码元集合的值，其中第一临时码元集合的值取决于第一中间码元集合的值且第二临时码元集合的值取决于第二中间码元集合的值。经编码码元的值是从第一和第二临时码元集合生成的。

[0029] 在一些变型中，可生成的经编码码元的数目独立于源码元数目。

[0030] 还提供了解码器实施例。根据依照本发明各方面的解码器的一个实施例中，接收机的解码器接收从中间码元生成的经编码码元，其中中间码元是使用可逆的方法从源码元生成的，即，还存在用于从中间码元生成源码元的逆方法，并且其中至少一个中间码元被指定为永久钝化码元且存在不在永久钝化码元当中的至少另一个中间码元。解码器从收到经编码码元解码中间码元集合，且解码器计及至少一个永久钝化码元，并使用该逆方法从经解码中间码元集合生成源码元。

[0031] 在解码时，调度解码步骤，且搁置对永久钝化码元的调度。永久钝化码元可使用新颖的或常规的方法来求解，并随后被用来求解其他中间码元。求解永久钝化码元（以及若使用了的其他运行中钝化）的一种办法可能是通过应用高斯消元法来求解钝化码元。剩余中间码元中的一些基于恢复出的永久钝化码元和收到经编码码元的值来恢复。

[0032] 在解码方法的一些变型中，永久钝化码元包括来自编码实施例的第二中间码元集合。在解码方法的一些变型中，永久钝化码元包括中间码元的子集，其中相应的编码方法不是多级链式反应码。此类编码方法可包括用于该中间码元的子集的 Tornado 码、Reed-Solomon 码、链式反应码 (Luby I 中描述的示例）或诸如此类中的一个或更多个。

[0033] 中间码元被用于编码和解码，其中针对合意的性能特性集（诸如可解码性）指示用于从源码元生成中间码元的方法以及相应的逆方法。在一些实施例中，中间码元包括源码元。在一些实施例中，中间码元包括源码元连同从源码元生成的冗余码元，其中冗余码元可以是链式反应码元、LDPC 码元、HDPC 码元、或其他类型的冗余码元。替换地，中间码元可基于码元之间的规定关系，例如中间码元与源码元之间的关系、以及中间码元之间的附加 LDPC 和 HDPC 关系，其中解码方法被用于基于该规定关系从源码元生成中间码元。

[0034] 这些方法和系统可由电子电路或由执行编程指令且具有用于实现编码和 / 或解

码的恰适指令程序代码的处理设备实现。

[0035] 通过本发明可达成众多益处。例如，在具体实施例中，编码数据以在信道上传输的计算花销得以减少。在另一个具体实施例中，解码此类数据的计算花销得以减少。在另一个具体实施例中，绝对和相对接收开销减少相当多。取决于实施例，可达成这些益处中的一个或更多个。贯穿本说明更详细且在以下更具体地提供这些及其他益处。

[0036] 对本文中公开的本发明的本质和优点的进一步理解可参照本说明的其余部分和附图来实现。

[0037] 附图简述

[0038] 图 1 是使用包括永久钝化的多级编码连同其他特征和要素的通信系统的框图。

[0039] 图 2 是在本文中的其他各图中使用的变量、阵列及诸如此类的表。

[0040] 图 3 是图 1 中所示的编码器的具体实施例的框图。

[0041] 图 4 是更详细地示出图 3 的动态编码器的框图。

[0042] 图 5 是解说永久钝化 (PI) 编码过程的流程图。

[0043] 图 6 是解说动态编码过程的流程图。

[0044] 图 7 是演算用于码元演算的权重的操作的流程图。

[0045] 图 8 解说了可存储在存储器中、可用于基于查找值来确定码元的度的表。

[0046] 图 9 示出了编码或解码过程中使用的矩阵。

[0047] 图 10 示出关于具体最小多项式表示图 9 中所示的矩阵的各部分的方程。

[0048] 图 11 是解说用于建立在编码或解码中使用的阵列的过程的流程图。

[0049] 图 12 解说了将由解码器使用表示解码器已知的 R 个静态码元或方程的子矩阵 SE 求解以从表示收到经编码码元的阵列 D() 恢复表示恢复出的源码元的阵列 C() 的方程组的矩阵表示。

[0050] 图 13 解说了使用 OTF 钝化从图 12 的矩阵的行 / 列置换得到的矩阵。

[0051] 图 14 是描述用于生成图 12 中的矩阵的过程的框图。

[0052] 图 15 解说了将由解码器使用子矩阵 SE 和与永久钝化码元相对应的子矩阵求解以从表示收到经编码码元的阵列 D() 恢复表示恢复出的源码元的阵列 C() 的方程组的矩阵表示。

[0053] 图 16 是解说用于生成如可在图 12 的矩阵或图 15 的矩阵中使用的 LT 子矩阵的过程的流程图。

[0054] 图 17 是解说用于生成如可在图 15 的矩阵中使用的 PI 子矩阵的过程的流程图。

[0055] 图 18 是矩阵生成器的框图。

[0056] 图 19 是解说用于生成 SE 子矩阵的过程的流程图。

[0057] 图 20 是解说用于生成 PI 子矩阵的过程的流程图。

[0058] 图 21 是解说用于在解码器中求解恢复出的码元的过程的流程图。

[0059] 图 22 解说将由解码器求解以从表示收到经编码码元的阵列 D() 恢复表示恢复出的源码元的阵列 C() 的方程组在置换之后的矩阵表示。

[0060] 图 23 解说将由解码器求解且与图 26 中所示的矩阵相对应的方程组的矩阵表示。

[0061] 图 24 解说可用作解码过程的一部分的矩阵表示。

[0062] 图 25 解说可用作解码过程的另一部分的矩阵表示。

[0063] 图 26 解说在部分求解之后将由解码器求解的方程组的矩阵表示。

[0064] 图 27 是解说用于在解码器中求解恢复出的码元的另一过程的流程图。

[0065] 图 28 解说将由解码器求解的方程组的矩阵表示。

[0066] 图 29 解说将由解码器求解的方程组的矩阵表示。

[0067] 图 30 解说示例编码系统，其可实现为硬件模块、软件模块、或存储在程序存储中并由处理器执行的可能作为并非如该图中所示地分开的综合代码单元的程序代码的部分。

[0068] 图 31 解说示例解码系统，其可实现为硬件模块、软件模块、或存储在程序存储中并由处理器执行的可能作为并非如该图中所示地分开的综合代码单元的程序代码的部分。

[0069] 所附的附录 A 是用于编码器 / 解码器系统、纠错方案、以及针对数据对象的可靠递送的应用的具体实施例的代码规范，有时带有所使用的本发明的细节，其还包括系统编码器 / 解码器如何可用于对象递送传输的规范。应理解，附录 A 中描述的具体实施例并不限制本发明的示例且本发明的一些方面可使用附录 A 的教示而其他方面可能不使用附录 A 的教示。还应理解，附录 A 中的限定性语句关于具体实施例的要求可能是限定性的，且此类限定性语句可能涉及或可能不涉及所要求保护的发明，且因此权利要求语言不受此类限定性语句限定。

[0070] 具体实施例的详细描述

[0071] 用于实现本文中引述的编码器和解码器的各部分的细节由 Luby I、Luby II、Shokrollahi I、Shokrollahi II、Shokrollahi III、Luby III、和 Shokrollahi IV 提供且出于简明起见不在此全部重复。它们的全部公开内容通用地通过援引纳入于此，且应理解，除非另行指出，否则其中的实现不是本发明所必要的，且还可以使用许多其他变型、修改或替换。

[0072] 如本文中所描述的多级编码在多级内对源数据进行编码。一般但不总是，第一级向源数据添加预定量的冗余。第二级然后使用链式反应码或诸如此类以从原始源数据以及由第一级编码计算出的冗余码元产生经编码码元。在一个具体实施例中，使用链式反应解码过程来首先解码收到数据。若该过程没有成功地完整恢复出原始数据，则可应用第二解码步骤。

[0073] 本文中教示的一些实施例可应用于许多其他类型的代码，例如应用于如因特网工程任务组 (IETF) 请求注解 (RFC)5170 中描述的代码 (下文称为“IETF LDPC 码”)、以及应用于美国专利号 6,073,250、6,081,909 和 6,163,870 中描述的代码 (下文称为“Tornado 码”)，从而改善这些类型的代码的可靠性和 / 或 CPU 和 / 或存储器性能。

[0074] 本文中教示的一些实施例的一个优点在于与单独的链式反应编码相比需要较少的算术运算来产生经编码码元。包括第一级编码和第二级编码的一些具体实施例的另一优点在于第一级编码和第二级编码可以在分开的时间和 / 或由分开的设备完成，因此划分了计算负荷，并使总计算负荷还有存储器大小及存取模式要求最小化。在多级编码的实施例中，在第一级编码期间从输入文件生成冗余码元。在这些实施例中，在第二级编码中，从输入文件和冗余码元的组合生成经编码码元。在这些实施例的一些中，可按需生成经编码码元。在其中第二级包括链式反应编码的实施例中，每个经编码码元可以在不考虑其他经编码码元是如何生成的情况下生成。这些经编码码元一旦被生成然后就可被放入分组并被传送到其目的地，其中每个分组包含一个或更多个经编码码元。还可以改为或者同时使用非

分组化传输技术。

[0075] 如本文中使用的，术语“文件”是指存储在一个或更多个源处且将作为一个单元被递送到一个或更多个目的地的任何数据。因此，来自文件服务器或计算机存储设备的文档、图像和文件皆是能被递送的“文件”的示例。文件可以有已知的大小（诸如存储在硬盘上的一兆字节图像）或可以有未知的大小（诸如从流送源的输出获取的文件）。无论是哪种，文件是源码元的序列，其中每个源码元在文件内有位置且具有值。“文件”还可用于指流送源的短部分，即数据流可被划分成一秒区间，且每个此类一秒区间内的源数据块可被视为“文件”。作为另一示例，来自视频流送源的数据块可基于例如由能播放该视频流的视频系统定义的该数据的优先级被进一步划分成多个部分，且每个块的每一部分可被视为“文件”。因此，术语“文件”被一般性地使用且不旨在是广义限定性的。

[0076] 如本文中所使用的，源码元表示待传送或传达的数据，且经编码码元表示基于源码元生成的、在通信网络上传达或被存储以实现对源码元的可靠接收和 / 或再生的数据。中间码元表示在编码或解码过程的中间步骤期间使用或生成的码元，其中典型地存在用于从源码元生成中间码元的方法以及相应的用于从中间码元生成源码元的逆方法。输入码元表示在编码或解码过程期间向一个或更多个步骤输入的数据，且输出码元表示在编码或解码过程期间从一个或更多个步骤输出的数据。

[0077] 在许多实施例中，这些不同类型或标记的码元可以是相同的或者至少部分地包括其他类型的码元，且在一些示例中，这些术语被可互换地使用。在一示例中，假设待传送的文件是有 1,000 个字符的文本文件，每个字符被视为源码元。若这些 1,000 个源码元按原样被提供给编码器，编码器又输出被传送的经编码码元，则源码元也是输入码元。然而，在其中这 1,000 个源码元在第一步骤中被转换成 1,000 个（或者更多或更少个）中间码元且中间码元被提供给编码器以在第二步骤中生成经编码码元的实施例中，则在第一步骤中，源码元是输入码元且中间码元是输出码元，以及在第二步骤中，中间码元是输入码元且经编码码元是输出码元，然而，源码元是该两步骤编码器的总输入码元且经编码码元是该两步骤编码器的总输出码元。在此示例中，若编码器是系统编码器，则经编码码元可包括源码元连同从中间码元生成的修复码元，而中间码元不同于源码元和经编码码元两者。在此示例中，若编码器替代地是非系统编码器，则中间码元可包括源码元连同在第一步骤中使用例如 LDPC 和 / 或 HDPC 编码器从源码元生成的冗余码元，而经编码码元不同于源码元和中间码元两者。

[0078] 在其他示例中，有更多码元且每个码元表示一个以上的字符。在发射机中存在源 - 中间码元转换的任一种情形中，接收机可具有相应的中间 - 源码元转换作为逆转换。

[0079] 传输是通过信道从一个或更多个发送方向一个或更多个接收方传送数据以递送文件的过程。发送方有时也被称为编码器。若一个发送方通过理想信道连接到任何数量的接收方，则由于所有数据都将被正确接收，因此收到数据可能是源文件的确切副本。在此，假设信道不是理想的，这正是大多数实际信道的情形。在许多信道不理想中，两种感兴趣的不理想是数据擦除和数据不完整（其可被视为数据擦除的特殊情况）。数据擦除发生在信道丢失或丢弃数据时。数据不完整发生在接收方在直到一些数据已掠过了接收方才开始接收数据、接收方在传输结束前停止接收数据、接收方选择只接收所传送数据的一部分、和 / 或接收方间歇地停止并再次开始接收数据时。作为数据不完整的示例，移动卫星发送方可

能正在发射表示源文件的数据并在接收方处于射程内之前开始传输。一旦接收方处于射程内，数据可以被接收直到该卫星移出射程，此时接收方可以重定向其卫星天线（它在该时间期间不接收数据）以开始接收由移进射程内的另一卫星发射的关于相同输入文件的数据。如从阅读本描述应明白的，数据不完整是数据擦除的特殊情况，因为接收方可以将数据不完整视为如同接收方整段时间都处在射程内但信道丢失了直至接收方开始接收数据前的所有数据（且接收方有相同的问题）。而且，如在通信系统设计中公知的，可检测差错可通过简单地丢弃所有具有可检测差错的数据块或码元而被认为等效于擦除。

[0080] 在一些通信系统中，接收方接收由多个发送方生成的数据、或由一个发送方使用多个连接生成的数据。例如，为了加快下载，接收方可同时连接到一个以上传送关于相同文件的数据的发送方。作为另一示例，在多播传输中，多个多播数据流可以被传送以允许接收方能连接到这些流中的一个或更多个流，从而将集总传输速率与将这些接收方连接到发送方的信道的带宽匹配。在所有此类情形中，关注的是要确保所有所传送数据对于接收方是能独立使用的，即多个源数据在这些流之间不是冗余的，即使当传输速率对于不同流相差很大时以及当有任意丢失模式时亦然。

[0081] 一般而言，通信信道是连接发送方和接收方以进行数据传输的信道。通信信道可以是实时信道，其中信道在获得数据时就将数据从发送方移到接收方，或者通信信道可能是存储信道，该信道在将数据从发送方运输到接收方时存储一些或全部的数据。后者的示例是盘存储或其他存储设备。在该示例中，生成数据的程序或设备可以被认为是将数据传送给存储设备的发送方。接收方是从存储设备读取数据的程序或设备。发送方用来将数据送到存储设备上的机制、存储设备本身以及接收方用来从存储设备获取数据的机制共同构成了信道。如果存在这些机制或存储设备会丢失数据的机会，则它可被视为通信信道中的数据擦除。

[0082] 当发送方和接收方由其中码元会被擦除的通信信道隔开，则优选不传送输入文件的确切副本，而是改为传送从输入文件生成的帮助擦除恢复的数据。编码器是处理该任务的电路、设备模块或代码段。查看编码器操作的一种方式在于编码器从源码元生成经编码码元，其中源码元值序列表示输入文件。每个源码元因此将在输入文件内有位置且具有值。解码器是从由接收方接收到的经编码码元重构源码元的电路、设备、模块或代码段。在多级编码中，编码器和解码器有时进一步被分成各自执行不同任务的子模块。

[0083] 在多级编码系统的实施例中，编码器和解码器可被进一步分成各自执行不同任务的子模块。例如，在一些实施例中，编码器包括本文所谓的静态编码器和动态编码器。如本文中使用的，“静态编码器”是从源码元集合生成数个冗余码元的编码器，其中冗余码元的数目是在编码前确定的。当在多级编码系统中使用静态编码时，源码元与使用静态编码器从源码元生成的冗余码元的组合往往被称为中间码元。潜在可能的静态编码代码的示例包括 Reed-Solomon 码、Tornado 码、汉明码、诸如 IETF LDPC 码之类的 LDPC 码等。本文中使用术语“静态解码器”来指能解码由静态编码器编码的数据的解码器。

[0084] 如本文中使用的，“动态编码器”是从输入码元集合生成经编码码元的编码器，其中可能的经编码码元的数目独立于输入码元的数目，且其中将生成的经编码码元的数目无需是固定的。通常在多级编码中，输入码元是使用静态编码生成的中间码元，而经编码码元是使用动态编码器从中间码元生成的。动态编码器的一个示例是链式反应编码器，诸如

Luby I 和 Luby II 中教示的编码器。本文中使用术语“动态解码器”来指能解码由动态编码器编码的数据的解码器。

[0085] 在一些实施例中，为多级编码且系统性的编码使用应用于源码元的解码过程基于中间码元之间由静态编码器定义的关系以及中间码元与源码元之间由动态编码器定义的关系来获得中间码元值，并随后使用动态编码器从中间码元生成附加经编码码元或即修复码元。类似地，相应的解码器具有接收经编码码元并基于中间码元之间由静态编码器定义的关系以及中间码元与收到经编码码元之间由动态编码器定义的关系从收到经编码码元解码中间码元值的解码过程，并随后使用动态编码器从中间码元生成任何缺失的源码元。

[0086] 多级编码的实施例无需被限于任何特定类型的码元。典型地，码元值是从对应某个正整数 M 的有 2^M 个码元的字母表中选择的。在此类情形中，源码元可以用来自输入文件的 M 比特数据序列表示。M 的值往往基于例如使用的应用、通信信道和 / 或经编码码元的大小来确定。另外，经编码码元的大小往往基于应用、信道和 / 或源码元的大小来确定。在一些情形中，如果经编码码元值和源码元值有相同大小（即可由相同数目的比特表示、或是从相同的字母表选择的），则编码过程可以被简化。如果是这种情形，则在经编码码元值大小受限时，源码元值大小也受限。例如，可能期望将经编码码元放入大小有限的分组中。如果关于与经编码码元相关联的关键字的一些数据要被传送以在接收机处恢复该关键字，则经编码码元将优选足够小以在一个分组中容纳经编码码元值和关于关键字的数据。

[0087] 作为示例，如果输入文件是多兆字节的文件，则输入文件可能被断成数千、数万或数十万个源码元，其中每个源码元编码几千、几百或仅几个字节。作为另一示例，对于基于分组的因特网信道，有效载荷大小为 1024 字节的分组可能是恰适的（一字节为 8 比特）。在该示例中，假定每个分组包含一个经编码码元和 8 字节的辅助信息，则经编码码元大小为 8128 比特 ((1024-8)*8) 将是恰适的。因此，源码元大小可以被选取为 M = (1024-8)*8，或即 8128 比特。作为另一示例，一些卫星系统使用 MPEG 分组标准，其中每个分组的有效载荷包括 188 个字节。在该示例中，假定每个分组包含一个经编码码元和 4 字节的辅助信息，则经编码码元大小为 1472 比特 ((188-4)*8) 将是恰适的。因此，源码元大小可以被选取为 M = (188-4)*8，或即 1472 比特。在使用多级编码的通用通信系统中，诸如源码元大小（即 M，即由一个源码元编码的比特数）之类的专用参数可以是由应用设定的变量。

[0088] 每个经编码码元具有值。在以下考虑的一个优选实施例中，每个经编码码元还与被称为其“关键字”的标识符相关联。优选地，每个经编码码元的关键字能被接收方简单地确定以允许接收方能将一个经编码码元与其他经编码码元区别开。优选地，一个经编码码元的关键字不同于所有其他经编码码元的关键字。先前技术中讨论了各种形式的关键字法。例如，Luby I 描述了在本发明的实施例中可以采用的各种形式的关键字法。在其他优选实施例中，诸如在附录 A 中描述的一个实施例中，经编码码元的关键字被称为“经编码码元标识符”、或“编码码元标识符”，或更简单地称为“ESI”。

[0089] 多级编码在预期有数据擦除的场合或接收方没有正好在传输开始和结束时开始和结束接收的场合尤其有用。后一种情况在此被称为“数据不完整”。关于擦除事件，多级编码享有在 Luby I 中教示的链式反应编码的许多益处。具体而言，经多级编码的码元是信息加性的，所以可以使用任何合适数量的分组来恢复输入文件以达到合意准确度。在使用多级编码时，这些状况不会负面地影响通信过程，因为用多级编码生成的经编码码元是信

息加性的。例如，若由于噪声突发引起数据擦除而导致一百个分组丢失，则可以在该突发后获取额外的一百个分组来取代被擦除分组的丢失。如果因为在发射机开始传送时接收机没有调谐到该发射机而导致成千个分组丢失，则接收机可以仅从任何其他传输期或甚至从另一发射机获取这成千个分组。使用多级编码，接收机不限于获取任何特定分组集合，所以它可以从一个发射机接收一些分组、切换到另一发射机、丢失一些分组、错过给定传输的起始或结束，且仍能恢复输入文件。在没有接收机 - 发射机协调的情况下加入和离开传输的能力有助于简化通信过程。

[0090] 在一些实施例中，使用多级编码来传送文件可以包括从输入文件生成、形成或提取源码元，计算冗余码元，将源码元和冗余码元编码成一个或更多个经编码码元——其中每个经编码码元是基于其关键字独立于所有其他经编码码元生成的，以及将经编码码元在信道上传送给一个或更多个接收方。另外，在一些实施例中，使用多级编码来接收（以及重构）输入文件的副本可以包括从一个或更多个数据流接收经编码码元的某个集合或子集，并从收到经编码码元的值和关键字来解码源码元。

[0091] 系统码和非系统码

[0092] 系统码是其中源码元在能被传送的经编码码元当中的代码。在这种情形中，经编码码元包括源码元和从源码元生成的也称为修复码元的冗余码元。出于各种原因，对于许多应用而言，系统码优于非系统码。例如，在文件递送应用中，有用的是能够在正使用数据生成修复数据的同时按顺序次序开始传送该数据，其中生成修复数据的过程可能要花费一定量的时间。作为另一示例，许多应用优选以未修改形式按顺序次序向一个信道发送原始源数据，并向另一信道发送修复数据。此举的一个典型原因是为了既支持未纳入 FEC 解码的旧式接收机，与此同时还向纳入了 FEC 解码的增强型接收机提供更好的体验，其中旧式接收机仅加入源数据信道，而增强型接收机加入源数据信道和修复数据信道两者。

[0093] 在这些以及相关类型的应用中，有时可能是以下情形：由接收机接收到的源码元中的丢失模式和丢失分数与收到修复码元中经历的丢失模式和丢失分数有很大不同。例如，当源码元在修复码元之前被发送时，由于信道的突发性丢失状况，源码元中的丢失分数和模式可能与修复码元中相应的丢失分数和模式有很大不同，且源码元中的丢失模式与丢失是均匀随机的情况下可能的典型丢失模式相去甚远。作为另一示例，当在一个信道上发送源数据而在另一信道上发送修复数据时，这两个信道上可能存在相当不同的丢失状况。因此，具有在不同类型的丢失状况下工作良好的系统 FEC 码是合意的。

[0094] 尽管本文中的示例引述系统码（其中输出或经编码码元包括源或输入码元）或非系统码，但除非另行指出，否则本文中的教示应被认为可应用于这两种代码。Shokrollahi III 教示了用于将非系统链式反应码转换成系统码以使得如此构造的系统码维持非系统码的稳健性属性的方法。

[0095] 具体而言，使用 Shokrollahi III 中教示的方法，所构造的系统码具有以下属性：丢失的源码元和丢失的修复码元之间就解码器的可恢复性而言差别很小，即，对于给定的总丢失量，解码恢复概率基本相同，几乎独立于源码元中的丢失比例相比于修复码元中的丢失比例。此外，经编码码元中的丢失模式并不显著影响解码恢复概率。比较而言，对于其他系统码的构造，诸如针对 Tornado 码或针对 IETF LDPC 码描述的那些构造，在许多情形中，丢失的源码元和丢失的修复码元之间就解码器的可恢复性而言差别很大，即，对于给

定的总丢失量，解码恢复概率宽泛地变化，取决于源码元中的丢失比例相比于修复码元中的丢失比例。此外，经编码码元中的丢失模式可能对解码恢复概率有强影响。若经编码码元的丢失在所有经编码码元中是均匀随机的，则 Tornado 码和 IETF LDPC 码有合理良好的恢复属性，但恢复属性随着丢失模型偏离均匀随机丢失而退化。因此，在这种意义上，Shokrollahi III 中教示的实施例具有胜于系统码的其他构造的优点。

[0096] 对于具有取决于丢失的源码元和丢失的修复码元的比例以及取决于丢失模式就解码器的可恢复性而言有强影响的属性的 FEC 码，在适用的情况下克服这种属性的一种办法是按均匀随机次序发送经编码码元，即按均匀随机次序发送源码元和修复码元的组合，因此源码元随机地散布在修复码元之间。按随机次序发送经编码码元具有以下优点：无论信道丢失模型如何、无论丢失是突发性的还是均匀随机的还是其他某种丢失类型，对经编码码元的丢失仍是随机的。然而，如上所述，这种办法对于一些应用不是合意的，例如对于其中期望在修复码元之前按顺序发送源码元、或其中源码元是在与修复码元不同的信道上发送的应用。

[0097] 在此类情形中，希望其中经编码码元中的丢失模式对解码器的恢复属性影响不大的系统码的构造，且本文中提供了一些示例。

[0098] 如本文中所使用的，“随机”和“伪随机”往往是等效的和 / 或可互换的且可取决于上下文。例如，随机丢失可以指信道丢失了哪些码元，其可以真正是随机事件，而对码元邻元的随机选择可能实际上是根据非随机过程的可重复伪随机选择，但与真正随机选择的情形具有相同或相似的属性或行为。除非显式地或通过上下文另行指出，否则将某事表征为随机的不意味着排除伪随机。

[0099] 在针对此类系统 FEC 编码器的一种办法中，由包括多个编码器子块或子过程的编码器获得源码元，这多个编码器子块或子过程之一作用为生成中间码元的解码器，这些中间码元是另一子块或子过程的输入码元。中间码元随后被施加到另一子块或子过程，该另一子块或子过程将这些中间码元编码成经编码码元，从而经编码码元包括从一个一致性过程生成的源码元（连同附加的冗余码元），藉此提供胜于作为使用一个过程（例如，复制）来获取经编码码元集合的源码元并使用另一过程来获取该经编码码元集合的冗余码元的系统编码器的编码器的稳健性益处和其他益处。

[0100] 输出编码可以是链式反应编码器、静态编码器或其他变型。附录 A 描述了系统码实施例。在阅读本公开之后，本领域普通技术人员应能够容易扩展 Shokrollahi III 的教示以应用于诸如 Tornado 码和 IETF LDPC 码之类的系统码，从而产生这些代码的也是系统码但具有更好恢复属性的新版本。具体而言，通过应用以下描述的一般性方法获得的这些代码的新版本被增强以具有以下属性：源码元中的丢失比例相比于修复码元中的丢失比例不会显著影响解码恢复概率，此外丢失模式不会显著影响解码恢复概率。因此，这些代码能被有效地用在要求使用具有不会受源码元和修复码元中的不同分数丢失量或不同丢失模式强烈影响的恢复属性的系统 FEC 码的上述应用中。

[0101] 该新编码方法能被一般性地应用于系统 FEC 码、非系统 FEC 码、固定码率 FEC 码和链式反应 FEC 码的编码以产生用于新的增强型系统 FEC 码的总体编码方法。还存在可应用的相应的新解码方法。

[0102] 编码器中的解码器的示例

[0103] 现在将提供编码器中的解码器的示例。

[0104] 令编码方法 E 为（发射机中或别处的）编码器针对从 K 个源码元生成 N 个经编码码元的固定码率（非系统或系统）FEC 码 E 使用的编码方法，其中 N 至少为 K。类似地，令解码方法 E 为接收机中或别处的解码器使用的针对 FEC 码 E 的相应解码方法。

[0105] 假设 FEC 码 E 具有以下属性：N 个经编码码元中的 K 个经编码码元的随机集合足以使用解码方法 E 以合理概率恢复原始的 K 个源码元，其中合理概率可能例如为概率 1/2。合理概率可以是由使用或应用设定的某个要求且可以为不同于 1/2 的值。应理解，特定码的构造无需因特定恢复概率而异，但应用和系统可针对其特定稳健性程度来设计。在一些实例中，可通过考虑 K 个以上码元并随后使用解码过程确定这些所考虑码元中允许成功解码的 K 个码元的集合来提高恢复概率。

[0106] 假设对于 FEC 码 E，ESI（经编码码元标识符）与每个经编码码元相关联且该 ESI 标识该经编码码元。在不失一般性的情况下，ESI 在本文中用 0、1、2、…、N-1 标记。

[0107] 在用于使用针对 FEC 码 E 的方法生成的系统 FEC 码 F 的系统编码方法 F 的一个实施例中，K 和 N 是输入参数。FEC 码 F 的源码元将具有 ESI 0、…、K-1，且 FEC 码 F 的修复码元将具有 ESI K、…、N-1。针对 FEC 码 F 的系统编码方法 F 使用如下由硬件和 / 或软件执行的针对 FEC 码 E 的编码方法 E 和解码方法 E 从 K 个源码元 C(0)、…、C(K-1) 生成 N 个经编码码元：

[0108] (1) 随机地置换与 FEC 码 E 相关联的 N 个 ESI 以得到 FEC 码 E 经置换 ESI 集合 X(0)、…、X(N-1)，其中该经置换 ESI 集合被组织成使得能关于 ESI 的置换次序 X(0)、…、X(K-1) 从 FEC 码 E 的头 K 个经编码码元解码出 FEC 码 E 的 K 个源码元，

[0109] (2) 对于每个 i ＝ 0，…，N-1，将 FEC 码 F 的 ESI i 与 FEC 码 E 的 ESI X(i) 相关联，

[0110] (3) 对于每个 i ＝ 0，…，K-1，将具有 ESI X(i) 的 FEC 码 E 经编码码元的值设为源码元 C(i) 的值，

[0111] (4) 向具有相应的 FEC 码 E ESI X(0)、…、X(K-1) 的源码元 C(0)、…、C(K-1) 应用解码方法 E 以生成经解码码元 E(0)、…、E(K-1)，以及

[0112] (5) 向经解码码元 E(0)、…、E(K-1) 应用编码方法 E 以生成具有相关联 FEC 码 ESI 0、…、N-1 的 FEC 码 E 经编码码元 D(0)、…、D(N-1)，

[0113] (6) 编码方法 F 的具有 ESI 0、1、…、N-1 的经编码码元为 D(X(0))、D(X(1))、…、D(X(N-1))。

[0114] 注意，编码方法 F 的输出为 N 个经编码码元，其中头 K 个经编码码元为具有相关联 ESI 0、1、…、K-1 的源码元 C(0)、…、C(K-1)。因此，编码方法 F 产生对源数据的系统编码。

[0115] 与刚才描述的编码方法 F 相对应的解码方法 F 的一个实施例如下，其中 K 和 N 是该方法贯穿始终使用的输入参数。该解码方法 F 从具有相关联 FEC 码 F ESI Y(0)、…、Y(K-1) 的 K 个收到经编码码元 D(0)、…、D(K-1) 恢复 K 个源码元 C(0)、…、C(K-1)。收到码元无需正好是所发送码元。由硬件和 / 或软件执行的该方法如下：

[0116] (1) 随机地置换与 FEC 码 E 相关联的 N 个 ESI 以得到 FEC 码 E 经置换 ESI 集合 X(0)、…、X(N-1)，其中该经置换 ESI 集合被组织成使得能关于 ESI 的置换次序 X(0)、…、

X(K-1) 从 FEC 码 E 的头 K 个经编码码元解码出 FEC 码 E 的 K 个源码元，

[0117] (2) 向具有相关联 FEC 码 E ESI X(Y(0))、…、X(Y(K-1)) 的经编码码元 D(0)、…、D(K-1) 应用解码方法 E 以生成经解码码元 E(0)、…、E(K-1)，

[0118] (3) 使用编码方法 E 从 E(0)、…、E(K-1) 生成具有 FEC 码 E ESI X(0)、…、X(K-1) 的经编码码元 C(0)、…、C(K-1)，

[0119] (4) 具有 ESI 0、…、K-1 的 FEC 码 F 的经解码源码元为 C(0)、…、C(K-1)。

[0120] 如刚才所描述地操作的方法和装置具有一些合意属性。例如，考虑 FEC 码 E，其为系统码且具有能以高概率解码 K 个收到经编码码元的随机集合的属性，但还具有当接收到 K 个经编码码元且收到经编码码元中的源码元比例不接近 K/N 时则其不能以高概率来解码的属性。在这种情形中，该实施例描述了使用 FEC 码 E 的编码和解码方法的新 FEC 码 F，并且该新 FEC 码 F 具有以下合意属性：其将从 K 个收到经编码码元的集合独立于作为源码元的收到经编码码元的比例以高概率进行解码。

[0121] 上面的实施例有许多变形。例如，在编码方法 F 的步骤 (1) 中，ESI 的随机置换可以是伪随机的或者基于产生对 ESI 的良好选择但既不是随机也不是伪随机的其他某种方法。在 FEC 码 E 为系统码的情形中，优选该置换中在步骤 (1) 中从系统 ESI 中选择的头 K 个 ESI 的分数与 FEC 码 E 的码率成比例，即与 K/N 成比例。优选由新编码方法 F 在步骤 (1) 中作出的对 ESI 的随机选取可由简洁的数据量来表示，例如由用于公知或协定的伪随机生成器的种子连同用于基于该种子和伪随机生成器如何工作来选取 ESI 的协定方法来表示，从而该新解码方法 F 可在步骤 (1) 中基于相同的种子以及用于生成 ESI 的伪随机生成器和方法作出完全相同的 ESI 置换选取。一般而言，优选新编码方法 F 在步骤 (1) 中生成 ESI 序列使用的过程和新解码方法 F 在步骤 (1) 中生成 ESI 序列使用的过程两者生成相同的 ESI 序列，以确保新解码方法 F 是新编码方法 F 的逆。

[0122] 还存在其他变形，其中例如不使用显式 ESI，经编码码元的唯一性标识符改为藉由其关于其他经编码码元的位置或藉由其他手段。

[0123] 在以上描述中，FEC 码 E 的原始 ESI 被 FEC 码 F 重新映射，从而源码元的有序集合以连贯次序被指派 ESI 0、…、K-1，且修复码元被指派 ESI K、…、N-1。其他变形是可能的，例如 ESI 的重新映射可在发送方处刚好在编码方法 F 已生成经编码码元之后但在传送经编码码元之前发生，并且 ESI 的逆重新映射可在接收方处当经编码码元被接收时但在经编码码元被解码方法 F 处理以恢复原始源码元之前发生。

[0124] 作为另一变形，在新编码方法 F 的步骤 (1) 中，置换可通过首先选择 K+A 个 FEC 码 E ESI，其中 A 是为了确保有高概率的可解码性而选取的值，并随后在解码过程的模拟期间确定 K+A 个 ESI 中哪 K 个 ESI 在解码期间实际被使用，并且所选择的置换可将 K+A 个 ESI 的初始集合中在解码期间实际被使用的 K 个 ESI 选择为该置换的头 K 个 ESI。类似变形应用于新解码方法 F。

[0125] 作为编码方法 F 的另一变形，用于生成随机置换的种子是针对 K 的值预先计算出的，以确保与步骤 (1) 中产生的 ESI 置换相关联的 FEC 码 E 的头 K 个经编码码元是可解码的，且随后该种子总是被用于编码方法 F 和相应的解码方法 F 的步骤 (1) 中的 K 以生成步骤 (1) 中的置换。用于选取此类种子的方法包括随机地选取种子直至找到确保步骤 (1) 中的可解码性的一个种子并随后选择该种子。替换地，可由编码方法 F 动态地生成具有这些

属性的种子，且随后该种子可被传达给解码方法 F。

[0126] 作为编码方法 F 的另一变形，在步骤 (1) 中可选择部分置换，即在新编码方法 F 的步骤 (1) 中无需生成所有 ESI，且若步骤 (5) 和 (6) 中不需要全部经编码码元则无需生成所有经编码码元，例如由于这些经编码码元对应于作为经编码码元的部分的源码元或者由于需要生成少于 N 个经编码码元。在其他变形中，无需重新计算新解码方法 F 的步骤 (3) 和 (4) 中的所有经编码码元，因为一些收到经编码码元可对应于正被恢复的一些源码元。类似地，在新解码方法 F 的步骤 (2) 中，无需解码所有 K 个码元 E(0)、…、E(K-1)，例如若步骤 (2) 中解码出的一些码元在用于生成经编码码元的后续步骤中是不需要的。

[0127] 以上描述的方法和实施例具有许多应用。例如，编码方法 F 和解码方法 F 及其变形可应用于 Tornado 码和 IETF LDPC 码以提供改善的接收开销和解码失败概率性能。一般而言，这些新方法适用于任何固定码率 FEC 码。这些新方法的变形还可应用于无固定码率的 FEC 码，即可应用于其中可生成的经编码码元的数目独立于源码元的数目的诸如链式反应码之类的 FEC 码。

[0128] Shokrollahi III 包含用于创建用于链式反应码的系统编码和解码方法的类似教示。在一些实施例中，用于这些代码的编码和解码方法 E 是 Luby I、Luby II、Shokrollahi I、Shokrollahi II、Luby III、Shokrollahi IV 中教示的那些编码和解码方法。为了描述系统编码器，描述编码方法 E 和解码方法 E 并使用以上描述且从这些参考知晓的一般原理往往就足以将这些方法变换成系统编码方法 F 和系统解码方法 F。因此，在阅读本公开和所引述的参考之际，本领域普通技术人员应当明白如何利用描述编码方法 E 和解码方法 E 的教示并将它们应用于系统编码方法 F 和系统解码方法 F 或诸如此类。

[0129] 钝化

[0130] 如 Shokrollahi II 中教示的，钝化解码是每当从一组已知线性方程值求解一组未知变量时就可结合置信传播来应用的一般方法，并且在实现基于线性方程组的高效编码和解码方法时尤其有益。为了区别 Shokrollahi II 中描述的钝化解码与下文中描述的永久钝化解码，使用“运行中”钝化（在各处缩写为“OTF 钝化”）来指代 Shokrollahi II 的方法和教示，而使用“永久钝化”来指代本文中其中提前选择钝化的方法和教示。

[0131] 置信传播解码的一个原则在于，在解码过程期间每当有可能时，解码器就应当使用依赖于一个剩余未知变量的（可能简化的）方程来求解该变量，且该方程因此与该变量相关联，以及随后通过消元剩余的未使用方程对该解出变量的依赖性来简化这些剩余的未使用方程。例如在 Tornado 码、诸如 Luby I、Luby II、Shokrollahi I、Shokrollahi II、Luby III、Shokrollahi IV 中描述的链式反应码、以及 IETF LDPC 码的一些实施例中已使用了此类简单的基于置信传播的解码过程。

[0132] OTF 钝化解码在多个阶段中进行。在 OTF 钝化解码方法的第一阶段中，每当置信传播解码过程由于不存在依赖于仅一个剩余未知变量的剩余方程而不能继续时，解码器将“OTF 钝化”一个或更多个未知变量并关于置信传播过程将它们视为“解出”并从剩余方程“消元”（即使它们其实未被消元），由此可能允许置信传播解码过程继续进行。在第一阶段期间被 OTF 钝化的变量随后例如在第二阶段中例如使用高斯消元法或计算更高效的方法来求解，并且随后在第三阶段中，这些被 OTF 钝化的变量的值被用来完整地求解与第一解码阶段期间的方程相关联的变量。

[0133] 如 Shokrollahi II 中更详细地教示的 OTF 钝化解码除链式反应码之外还可应用于许多其他类型的代码。例如，其可被应用于一般类的 LDPC 和 LDGM 码，尤其是应用于 IETF LDPC 码和 Tornado 码，从而导致改善这些类型的代码的可靠性（降低解码失败的概率）和 / 或 CPU 和 / 或存储器性能（提高编码和 / 或解码的速度和 / 或减小所要求的存储器大小和 / 或存取模式要求）。

[0134] 链式反应码实施例结合 OTF 钝化解码的一些变形在 Shokrollahi IV 中描述。其他变形在本申请中描述。

[0135] 系统概览

[0136] 图 1 是使用多级编码的通信系统 100 的框图。其类似于 Shokrollahi I 中所示的，但在该情形中，编码器 115 计及哪些中间码元被“永久钝化”的指定并在动态编码过程期间对这些中间码元与没有被永久钝化的中间码元不同地操作。同样，解码器 155 在解码时也计及被永久钝化的中间码元。

[0137] 如图 1 中解说的，K 个源码元 (C(0)、…、C(K-1)) 被输入编码器 115，且若对变得对解码器 155 可用的码元的解码成功，则解码器 115 可输出这 K 个源码元的副本。在一些实施例中，流被解析成有 K 个码元的块，且在一些实施例中，有大于 K 的某个数目的源码元的文件被分成大小为 K 的码元块且如此被传送。在其中优选块大小 $K' > K$ 的一些实施例中，可向 K 个源码元添加 $K' - K$ 个填充码元。这些填充码元可具有值 0，或编码器 115 和解码器 155 两者已知的任何其他固定值（或者能在解码器 155 处以其他方式确定的值）。应理解，编码器 115 可包括多个编码器、模块或诸如此类，且解码器 155 也可以是这种情形。

[0138] 如所解说的，编码器 115 还接收来自动态关键字生成器 120 的动态关键字序列和来自静态关键字生成器 130 的静态关键字序列，动态关键字生成器 120 和静态关键字生成器 130 各自可由随机数生成器 135 驱动。动态关键字生成器 120 的输出可简单地为基数序列，但无需是这种情形。关键字生成器的操作可如 Shokrollahi I 中所示的。

[0139] 应理解，附图中所示的各种功能块可实现为指定输入被提供作为输入信号的硬件，或者它们可由执行存储在指令存储器中并按恰适次序执行以执行相应功能的指令的处理器实现。在一些情形中，使用专门硬件来执行这些功能和 / 或执行程序代码。程序代码和处理器不总是被示出，但普通技术人员在阅读本公开之际将知道如何实现此类细节。

[0140] 编码器 115 还接收来自钝化指定器 125 的输入以及沿本文中别处描述的线向系统 100 输入的其他参数。钝化指定器 125 的输出可包括表示出于解码目的被指定为“永久钝化”的中间码元的数目的值 P（“PI 列表”指示哪 P 个中间码元在该列表上）。如别处解释的，用于编码过程的中间码元在一些实施例中恰好是 K 个源码元，而在其他实施例中，存在除了只复制 K 个源码元之外还从这 K 个源码元生成中间码元的某种类型的处理、转换、编码、解码等。

[0141] 输入参数可包括由关键字生成器和 / 或编码器的编码过程使用的随机种子（以下更详细地描述）、要生成的经编码码元的数目、要生成的 LDPC 码元的数目、要生成的 HDPC 码元的数目、要生成的中间码元的数目、要生成的冗余码元的数目等，和 / 或这些值中的一些是从编码器 115 可用的其他值计算出的。例如，要生成的 LDPC 码元的数目可完全从固定公式和 K 的值演算出来。

[0142] 编码器 115 从其输入生成经编码码元序列 ($B(I_0)$、$B(I_1)$、$B(I_2)$、…) 并将它们提供

给传送模块 140，传送模块 140 还接收来自动态关键字生成器 120 的动态关键字值（I_0、I_1、I_2、…），但若存在传达该信息的其他方法则可能不必如此。传送模块 140 可能以此处无需更详细描述的常规方式来传达给予信道 145 的内容。接收模块 150 接收经编码码元和动态关键字值（在需要的场合）。信道 145 可以是穿过空间的信道（用于从一地传送以在另一地接收）或穿过时间的信道（用于记录到介质以例如在以后的时间回放）。信道 145 可能导致丢失一些经编码码元。因此，解码器 115 从接收模块 150 接收到的经编码码元 $B(I_a)$、$B(I_b)$、…可能不等同于传送模块发送的经编码码元。这由不同的下标索引来指示。

[0143] 解码器 155 优选能够使用动态关键字重新生成器 160、随机数生成器 163 和静态关键字生成器 165 重新生成用于收到码元的关键字（这些关键字可能不同），并接收各种解码参数作为输入。这些输入中的一些输入可能是硬编码的（即，在设备的构造期间输入的）以及一些输入可能是可改变的输入。

[0144] 图 2 是在其他各图中以及贯穿本公开最经常使用的变量、阵列及诸如此类连同标注概要的表。除非另行指出，否则 K 表示用于编码器的源码元的数目，R 表示静态编码器生成的冗余码元的数目，以及 L 是“中间码元”的数目，即源码元与冗余码元的组合且因此 L = K+R。

[0145] 如以下解释的，在静态编码器的一些实施例中，生成两种类型的冗余码元。在此处的许多示例中使用的具体实施例中，第一集合包括 LDPC 码元且第二集合包括 HDPC 码元。在不失一般性的情况下，本文中的许多示例将 LDPC 码元的数目称为 S 以及将 HDPC 码元的数目称为 H。可能有两种以上冗余码元，因此不要求 R = S+H。LDPC 码元和 HDPC 码元具有不同的度分布，且本领域普通技术人员在阅读本公开之际将看出如何使用非 LDPC 或 HDPC 码元的冗余码元，但其中冗余码元包括两个（或更多个）码元集合，其中每个集合的度分布与其他集合的度分布不同。如公知的，冗余码元集合的度分布是指度的分布，其中冗余码元的度是指该冗余码元所依赖的源码元的数目。

[0146] P 表示中间码元中的永久钝化码元的数目。永久钝化码元是被指定进行特定处置——即在置信传播网络中为了继续进行置信传播而被“搁置”或“钝化”（并随后在解出钝化码元之后回来求解）——的那些码元，其中永久钝化码元与其他钝化码元的区别在于永久钝化码元是在编码器处被指定进行此类处置。

[0147] N 表示解码器 155 对其进行解码尝试的收到码元的数目，且 A 是“开销”码元的数目，即超出 K 的收到经编码码元数目。因此，A = N-K。

[0148] K、R、S、H、P、N 和 A 是整数，典型地全部大于或等于 1，但在具体实施例中，其中一些可以为 1 或 0（例如，R = 0 是其中没有冗余码元的情形，且 P = 0 退回到 Shokrollahi II 的情形，其中仅存在 OTF 钝化）。

[0149] 源码元矢量记为 (C(0)，…，C(K-1))，且冗余码元矢量记为 (C(K)，…，C(L-1))。因此，在系统情形中，(C(0)，…，C(L-1)) 表示中间码元矢量。这些中间码元中的数目 P 个中间码元被指定为“永久钝化”。“PI 列表”指示中间码元中的哪些是被永久钝化的中间码元。在许多实施例中，PI 列表简单地指向末 P 个中间码元，即 C(L-P)、…、C(L-1)，但这不是必要条件。假设该情形仅是为了简化本描述的其余部分。

[0150] 不在 PI 列表上的中间码元在本文中被称为“LT 中间码元”。在该示例中，LT 中间码元将为 C(0)、…、C(L-P-1)。D(0)、…、D(N-1) 表示收到经编码码元。

[0151] 应注意，在值阵列被描述为“N(0)、…、N(x)”或诸如此类的场合，不应假定这要求至少 3 个值，因为其并非意在排除只有一个或两个值的情形。

[0152] 使用永久钝化的编码方法

[0153] 图 3 是图 1 中所示的编码器 115 的一个具体实施例的框图。如其中所解说的，源码元被存储在输入缓冲器 205 中且被提供给静态编码器 210 和动态编码器 220，静态编码器 210 和动态编码器 220 还接收关键字输入和其他输入。静态编码器 210 可包括用于存储内部值和程序指令的内部存储 215（存储器、缓冲器、虚拟存储器、寄存器存储等）。同样，动态编码器 220 可包括用于存储内部值和程序指令的内部存储 225（存储器、缓冲器、虚拟存储器、寄存器存储等）。

[0154] 在一些实施例中，冗余演算器 230 确定要创建的冗余码元的数目 R。在一些实施例中，静态编码器 210 生成两个相异的冗余码元集合，且在具体实施例中，第一集合是头 S 个冗余码元，即码元 C(K)、…、C(K+S-1) 且它们是 LDPC 码元，而第二集合是接下来 H 个冗余码元，即 C(L-H)、…、C(L-1) 且它们是 HDPC 码元。若 PI 列表是末 P 个冗余码元，则所有 H 个冗余码元可在 PI 列表上（若 $P \geq H$）或者所有 P 个冗余码元可为 HDPC 码元（若 $P < H$）。

[0155] 导致生成这两个码元集合的操作可能相当不同。例如，在以下描述的一些实施例中，用于生成 LDPC 冗余码元的操作是二进制操作，而用于生成 HDPC 码元的操作是非二进制的。

[0156] 动态编码器 220 的操作在图 4 中更详细地解释。根据一个实施例，动态编码器 220 包括两个编码器，PI 编码器 240 和 LT 编码器 250。在一些实施例中，LT 编码器 250 是链式反应编码器，且 PI 编码器 240 是特定类型的链式反应编码器。在其他实施例中，这两个编码器可能非常相似，或者 PI 编码器 240 不是链式反应编码器。无论如何定义这些编码器，它们都生成码元，其中 LT 编码器 250 从被指定为非永久钝化的 LT 中间码元 C(0)、…、C(L-P-1) 生成其码元，而 PI 编码器 240 从永久钝化的中间码元 C(L-P)、…、C(L-1) 生成其码元。这两种所生成的码元进入组合器 260，组合器 260 生成最终经编码码元 270。

[0157] 在本发明的一些实施例中，一些永久钝化码元可参与 LT 编码过程，且不是永久钝化码元的一些码元可参与 PI 编码过程。换言之，PI 列表和包括 LT 中间码元的码元集合不必是非相交的。

[0158] 在优选实施例中，提供给组合器 260 的码元可具有相同长度，且由组合器 260 执行的功能是对这些码元的异或 (XOR) 运算以生成经编码码元 270。然而，这对于本发明的运转不是必要的。能够预想可导致类似结果的其他类型的组合器。

[0159] 在其他实施例中，中间码元被细分成两个以上集合，例如，一个 LT 码元集合和若干个（一个以上）PI 码元集合，每个 PI 码元集合具有其相关联的编码器 240。当然，每个相关联的编码器可被实现为在充当不同集合的不同编码器时根据编码过程对不同指令进行操作的共同计算元件或硬件元件。

[0160] 如可由 PI 编码器 240 执行的 PI 编码过程 241 的示例操作在图 5 中例示。使用与要生成的经编码码元相对应的关键字 I_a，在步骤 261，编码器确定正权重 WP 和包含 WP 个在 L-P 与 L-1 之间（含 L-P 和 L-1）的整数的列表 ALP。在步骤 263，若列表 $ALP = (t(0), \ldots, t(WP-1))$，则码元 X 的值被设为 $X = C(t(0)) \oplus C(t(1)) \oplus \cdots \oplus C(t(WP-1))$，其中 $\oplus$ 表示异或运算。

[0161] 在一些实施例中，权重 WP 固定为某个数字，诸如 3 或 4 或其他某个固定数字。在其他实施例中，权重 WP 可属于可能的此类数字的小集合，诸如被选取成等于 2 或 3。例如，如附录 A 的实施例中所示，权重 WP 取决于通过如可由 LT 编码器 250 执行的 LT 编码过程 251 生成的码元的权重。若 LT 编码器 250 生成的权重为 2，则 WP 取决于关键字 I_a 被选取为 2 或 3，其中 WP 为 2 或 3 的次数比例大约相等；若 LT 编码器 250 生成的权重大于 3，则 WP 被选取为 2。

[0162] 图 6 是根据本发明的实施例之一且使用 Luby I 和 Shokrollahi I 的教示的 LT 编码过程 251 的示例。在步骤 267，使用关键字 I_a 来分别生成权重 WL 和列表 AL。在步骤 269，若列表 ALP = (j(0)，…，j(WL-1))，则码元 X 的值被设为 X = C(j(0)) ⊕ C(j(1)) ⊕ … ⊕ C(j(WL-1))。

[0163] 图 7 解说了演算权重 WL 的操作。如其中所示，在步骤 272，创建与要生成的经编码码元相关联的数字 v，并且可基于该经编码码元的关键字 I_a 来计算数字 v。它可以是经编码码元的索引、代表性标记等，或者是不同的数字，只要编码器和解码器能一致即可。在该示例中，v 在 0 与 2^{20} 之间，但在其他示例中，其他范围是可能的（诸如 0 到 2^{32}）。生成 v 可以使用随机生成表以显式方式进行，但如何生成这些随机数的确切操作可以变化。

[0164] 假定编码器能访问表 M，表 M 的示例在图 8 中提供。称为“度分布查找”表的表 M 包含两列和多行。左列标记有权重 WL 的可能值，且右列标记有 0 与 2^{20} 之间（含 0 和 2^{20}）的整数。对于 v 的任何值，该度分布查找表的 M[d] 列中正好有一个单元，其中 M[d-1] < v ≤ M[d] 为真。对于这一个单元，d 列中存在相应的值，且编码器将它用作经编码码元的权重 WL。例如，在经编码码元具有 v = 900,000 的场合，该经编码码元的权重将为 WL = 7。

[0165] 静态编码器 210 能访问元素 SE(k,J)，其中 k = 0，…，R-1 且 j = 0，…，L-1。这些元素可属于任何有限域，该域的元素 α 与码元 X 之间存在运算 * 以使得 α*X 为码元，且 α*(X ⊕ Y) = α*X ⊕ α*Y，其中 ⊕ 表示异或运算。此类域和运算已在 Shokrollahi IV 中详述。静态编码器 210 的操作可描述为：对于给定的源码元序列 C(0)、…、C(K-1)，计算满足式 1 中所示关系的冗余码元序列 C(K)、…、C(L-1)，其中 Z(0)、…、Z(R-1) 是编码器和解码器已知的值（例如，0）。

[0166]
$$\begin{pmatrix} SE(0,0) & SE(0,1) & \cdots & SE(0,L-2) & SE(0,L-1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ SE(R-1,0) & SE(R-1,1) & \cdots & SE(R-1,L-2) & SE(R-1,L-1) \end{pmatrix} \bullet \begin{pmatrix} C(0) \\ \vdots \\ C(K-1) \\ C(K) \\ \vdots \\ C(L-1) \end{pmatrix} = \begin{pmatrix} Z(0) \\ \vdots \\ Z(R-1) \end{pmatrix},$$

（式 1）

[0167] 在式 1 中，条目 SE(k,j) 可全为二进制的，或者其中一些条目可属于域 GF(2) 而其他一些条目属于其他域。例如，附录 A 的实施例的相应矩阵在图 9 中给出。它包括两个子矩阵，一个有 S 行且一个有 H 行。上面的子矩阵包括两部分：包括末 P 列的子矩阵，其中每一行具有两个连贯的 1（这些位置按模 P 来计数）。该矩阵的头 W = L-P 列包括循环矩阵继以 SxS 单位矩阵。循环矩阵包括 B 列且每一列（可能最后一列除外）具有 S 行。这些循环矩阵的数目为 ceil(B/S)。这些循环矩阵中的列各自正好有 3 个 1。第 k 个循环矩阵的第一列在位置 0、(k+1)mod S 和 (2k+1)mod S 处具有 1。其他列是第一列的循环移位。图 9

中的下 H 行包括具有 GF(256) 中的条目的矩阵 Q 继以 HxH 单位矩阵。

[0168] 若 α 表示具有最小多项式 $x^8+x^4+x^3+x^2+1$ 的 GF(256) 的元素，则矩阵 Q 等同于图 10 中给出的矩阵。此处，Δ_1、…、Δ_{K+S-1} 是权重为 2 的列，其 2 个非零条目的位置是根据附录 A 的第 5.3.3.3 节中略述的规程伪随机地确定的。明智地选取值 S、P 和 H（诸如附录 A 中提供的各个值），图 10 中的矩阵导致相应代码优越的恢复属性。以上描述的规程在图 11 中例示。在步骤 276，矩阵 SE 被初始化为 0。在步骤 278，等于 LDPC 码元数目的输入变量 S 被提供给该过程，且对于对 (i,j)，其中 i = j mod S、或 i = (1+floor(j/S))+j mod S、或 i = 2*(1+floor(j/S))+j mod S，SE(i,j) 的值被设为 1。该步骤负责图 9 中的循环矩阵。

[0169] 在步骤 280，与图 9 中的单位矩阵 I_S 相对应的位置被设为 1。在步骤 282，与图 9 中的矩阵的 PI 部分相对应的位置被设为 1。这些位置是 (i,1) 和 (i,t) 形式的，其中 1 = i mod P 且 t = (i+1)mod P。在步骤 284，与图 9 中的矩阵 Q 相对应的位置被设置。相应地，提供矩阵 Q 作为该步骤的附加输入。在步骤 286，与图 9 的矩阵中的单位矩阵 I_H 相对应的位置被设为 1。

[0170] 对矩阵 SE 的其他选取是可能的且取决于特定应用和整体代码要求的需求。无论如何选取式 1 中的矩阵，静态编码器 210 的任务可用各种方式来完成。例如，高斯消元法可被用作恢复未知值 C(K)、…、C(L-1) 的过程，如本领域普通技术人员在阅读本公开时将明白的。

[0171] 解码和永久钝化

[0172] 解码问题可陈述如下：解码器 155 具有带有相应关键字 I_a、I_b、…的 N 个经编码码元 $B(I_a)$、$B(I_b)$、…。这些经编码码元的整个集合或其子集可能已被解码器接收到，而其他经编码码元可能已通过其他手段被给予解码器。解码器的目标是恢复源码元 C(0、…、C(K-1)。为了简化表示，将收到经编码码元记为 D(0)、…、D(N-1)。

[0173] 许多解码操作可使用矩阵语言以及对这些矩阵的操作——具体而言是求解带有此类矩阵的方程组——来简洁地描述。在以下描述中，方程可对应于收到经编码码元，且变量可对应于将基于收到经编码码元来求解的源码元或者源码元与从源码元生成的冗余码元（常常称为中间码元）的组合集合。在作为附录 A 提供的规范中，经编码码元可被称为“编码码元”（且存在其他变型），但是在阅读整个说明书和附录之后应当明白这些参考如何相关。还应理解，矩阵以及对方程的运算和求解可实现为与这些数学运算相对应的计算机指令，且在没有计算机、处理器、硬件或某种电子元件的情况下进行此类运算确实是不切实际的。

[0174] 在发起解码过程的第一阶段之前，在解码器处使用永久钝化来确定要钝化的一组变量，其被称为永久钝化码元或变量。以下描述的永久钝化解码方法可应用于现有代码，或者代码可被专门设计成结合永久钝化解码甚至更好地工作。永久钝化解码方法可被应用于求解任何线性方程组，且具体而言可被应用于链式反应码、IETF LDPC 码和 Tornado 码。

[0175] 永久钝化解码是每当从一组已知线性方程值求解一组未知变量时就可结合置信传播解码和 / 或 OTF 钝化解码来应用的一般方法，并且在实现基于线性方程组的高效编码和解码方法时尤其有益。在第一阶段，基于已知编码方法的结构或基于收到方程，申明一组未知变量将被永久钝化，且永久钝化变量被从线性方程移除且在解码过程的第二阶段中被视为“解出”（除了当第二阶段线性方程被简化时，对永久钝化变量执行相同的简化）。

[0176] 在第二阶段，使用先前描述的置信传播解码对未被永久钝化的未知变量应用置信传播解码，或者对未被永久钝化的未知变量应用类似于针对 OTF 钝化解码方法的第一阶段描述的 OTF 钝化解码，藉此产生一组简化的经编码码元或方程。从第二阶段得到的简化的经编码码元或方程具有以下属性：其对未被钝化的变量或码元的依赖性已被消除，因此这些简化的经编码码元或方程仅依赖于钝化变量或码元。注意，在一些实现中，也可以保持原始经编码码元或方程，从而原始经编码码元和简化的经编码码元两者皆是可用的。

[0177] 在第三阶段，使用简化的经编码码元或方程来求解永久钝化变量连同在第二阶段中使用 OTF 钝化解码生成的任何附加 OTF 钝化变量，例如使用高斯消元法来求解，或者使用永久钝化变量与线性方程之间的关系的特殊结构（若存在）比使用高斯消元法更高效地求解。

[0178] 在第四阶段，解出的钝化变量 (OTF 钝化变量或永久钝化变量）的值结合原始经编码码元或方程（或重新推导出的原始经编码码元或方程）被用于求解未被钝化的变量。

[0179] 永久钝化解码方法的一个优点在于，除了永久钝化之外，OTF 钝化的次数 w 一般可以较小或为 0，且很大程度上可独立于接收到哪些经编码码元。这可使得解码复杂性独立于接收到哪些经编码码元而一致地较小，允许更可靠的解码，以及允许可以更高效地调度的更加可预测且更少的存储器存取。由于第二阶段中仅存在少量 OTF 钝化，且由于一般仅在解码过程期间确定会使得码元操作模式在某种程度上不可预测的第二阶段中的 OTF 钝化，因此存储器存取模式在解码期间更加可预测，总体上允许更加可预测的高效解码过程。

[0180] 上述内容有许多变形。例如，各阶段可按非顺序的交织次序执行。作为另一示例，钝化码元可进而在第三阶段中使用多个附加阶段中的 OTF 钝化解码或永久钝化解码来求解。作为另一示例，永久钝化解码可应用于可用于纠错码、或擦除校正码的线性方程组和变量，或用于可使用线性方程组来求解的其他应用。作为另一示例，这些方法可应用于系统码和非系统码两者。作为另一示例，这些方法还可在编码过程期间应用，例如在使用 Shokrollahi III 中教示的方法从非系统码生成系统码时。

[0181] 在一些情形中，有可能设计编码过程使得永久钝化解码方法将尤为有效。例如，已知置信传播解码每当能被应用时就是计算高效的，但还知道它在单独使用时不能提供高可靠性解码。当在 OTF 钝化解码内使用置信传播解码时，可非常高效地处理置信传播步骤，但散布在置信传播步骤内的 OTF 钝化步骤会使解码减慢，且此类 OTF 钝化步骤越多，解码过程就越慢。

[0182] 在 OTF 钝化解码的典型实施例中，在尝试使用 N+R 个线性方程值求解 K+R 个未知变量时，在 N = K 时即尝试使用 0 开销求解这些变量时，OTF 钝化步骤的数目通常最大。另一方面，随着 N 增大到大于 K，典型地是这种情形：OTF 钝化解码的复杂性由于较少的 OTF 钝化步骤而降低，直至 N 足够大从而在一些情形中不存在 OTF 钝化步骤且钝化解码像或几乎像置信传播解码那样计算高效。在 OTF 钝化解码的其他实施例中，即使在 N 比 K 大相当多时，OTF 钝化的数目可能仍较大。

[0183] 在永久钝化解码的一个优选实施例中，永久钝化变量的数目 P 以及线性方程的结构被设计成使得在使用 OTF 钝化解码从 K+R 个线性方程值求解未被永久钝化的 L-P 个变量时，OTF 钝化解码期间的 OTF 钝化步骤的数目很小且在一些情形中为 0，因此 OTF 钝化解码步骤几乎像置信传播那样计算高效。

[0184] 在优选实施例中，线性方程的结构被设计成使得 OTF 钝化解码阶段几乎像置信传播解码那样高效。在此类优选实施例中，永久钝化变量与线性方程的关系使得求解钝化变量（包括永久钝化变量连同来自 OTF 钝化解码阶段的任何 OTF 钝化变量）的阶段能被高效地执行。此外，在优选实施例中，永久钝化码元的结构使得完成从解出的钝化码元来求解未被钝化的变量的阶段是计算高效的。

[0185] 带有永久钝化的链式反应码解码

[0186] 图 12 解说将由解码器使用 N 个收到经编码码元或方程以及 R 个已知静态码元或方程来求解的一组变量的矩阵表示。解码器的任务是求解该图中给出的线性方程组。典型地，码元 / 方程由解码器可访问的存储器或存储中存储着的值来表示，且以下描述的矩阵运算由解码器可执行的指令来实现。

[0187] 图 12 中所示的矩阵包括 L = K+R 列和 N+R 行。LT 子矩阵表示 N 个经编码码元与 L 个中间码元中通过 LT 编码过程 251 确定的 L-P 个 LT 码元之间的关系。PI 子矩阵表示 N 个经编码码元与 L 个中间码元中通过 PI 编码过程 241 确定的 P 个 PI 码元之间的关系。式 1 的矩阵 SE 表示由静态编码器 210 确定的中间码元之间的关系。解码器能基于收到经编码码元的关键字并从码构造确定这些关系。

[0188] 图 12 的线性方程组通过使用 Shokrollahi II 中教示的 OTF 钝化方法对上述矩阵作行 / 列置换以变换成图 13 中所示的形式来求解。它包括下三角矩阵 LO 310、包括与 OTF 钝化相对应的矩阵 320（称为 OTFI）的数个列、与永久钝化中间码元集合或其子集相对应的矩阵 330PI、以及与得到矩阵 LO 的三角化过程中没有使用的经编码或静态码元相对应的矩阵 340EL。

[0189] 图 14 是描述可执行得到图 12 中的矩阵的过程的元件的框图。它包括 LT 矩阵生成器 347、PI 矩阵生成器 349、以及静态矩阵生成器 350。一旦接收到关键字 I_a、I_b…，LT 矩阵生成器就创建图 12 中的矩阵 LT，而 PI 矩阵生成器 349 创建图 12 的矩阵 PI。这两个矩阵的级联被转发给静态矩阵生成器 350，静态矩阵生成器 350 可将静态关键字 S_0、S_1、… 作为附加提示。静态矩阵生成器的任务是创建矩阵 SE，且其输出是图 12 中给出的完整矩阵。

[0190] LT 矩阵生成器 347 和 PI 矩阵生成器 349 的操作分别与图 15 中 LT 编码器 250 和 PI 编码器 240 的操作紧密关联。静态矩阵生成器 350 的操作是重新创建用于静态编码的式 1 的矩阵 SE。

[0191] LT 矩阵生成器 347、PI 矩阵生成器 349 和静态矩阵生成器现在将参照它们可能执行的操作来进一步详细地描述。

[0192] 图 16 是解说 LT 矩阵生成器 347 采用的方法的一个实施例 500 的流程图。在步骤 505，LT 矩阵生成器 347 将 N x(L-P) 格式的矩阵 LT 初始化为全 0。接下来，在步骤 510，使用关键字 I_a、I_b、…来分别生成权重 WL(0)、…、WL(N-1) 和列表 AL(0)、…、AL(N-1)。列表 AL(i) 中的每个列表包括 WL(i) 个在范围 0、…、L-P-1 中的整数 (j(0)，…，j(WL(i)-1))。在步骤 515，使用这些整数将条目 LT(i，j(0))、…、LT(i，j(WL(i)-1)) 设为 1。如以上解释的，矩阵 LT 在收到码元 (D(0)，…，D(N-1)) 的意义上贡献于未知数 (C(0)，…，C(L-1)) 的方程组。

[0193] 如本领域技术人员可领会的，如此处描述的 LT 矩阵生成器的操作类似于图 6 的 LT

编码过程 251 的操作。

[0194] 图 17 是解说 PI 矩阵生成器 349 采用的方法的一个实施例 600 的流程图。在步骤 610，PI 矩阵生成器 349 将 N x P 格式的矩阵 PI 初始化为全 0。接下来，在步骤 615，使用关键字 I_a、I_b、…来分别生成权重 WP(0)、…、WP(N-1) 和列表 ALP(0)、…、ALP(N-1)。列表 ALP(i) 中的每个列表包括 WP(i) 个在范围 0、…、P-1 中的整数 (j(0)，…，j(WP(i)-1))。在步骤 620，使用这些整数将条目 PI(i，j(0))、…、PI(i，j(WP(i)-1)) 设为 1。PI 矩阵生成器的操作类似于图 5 中的 PI 编码过程 241 的操作。

[0195] 如以上解释的，矩阵 LT 和 PI 在收到码元 (D(0)，…，D(N-1)) 的意义上贡献于未知数 (C(0)，…，C(L-1)) 的方程组。理由如下：一旦 LT 编码器选取权重 WL(i) 并关联列表 AL(i) = (j(0)，…，j(WL(i)-1))，且 PI 编码器选取权重 WP(i) 并关联列表 ALP(i) = (t(0)，…，t(WP(i)-1))，则如下所示地获得相应的经编码码元 D(i)。针对 i 在 0 和 N-1 之间的所有值累积的这些方程得到式 2 中表示的合意方程组。

[0196] $D(i) = C(j(0)) \oplus \cdots \oplus C(j(WL(i)-1)) \oplus C(t(0)) \oplus \cdots \oplus C(t(WP(i)-1))$ （式 2）

[0197] 权重 WL 可以使用与图 7 中给出的规程相似的规程来演算。本领域普通技术人员在查阅本公开之际将看出如何将此举扩展到其中有两个以上编码器、每个编码器有不同的度分布来操作的情形。

[0198] 在图 18 中提供矩阵生成器的略微不同的流程图。它包括 LT 矩阵生成器 710、静态矩阵生成器 715、以及 PI 矩阵生成器 720。一旦接收到关键字 I_a、I_b…，LT 矩阵生成器 710 就创建图 15 中解说的矩阵 LT，而静态矩阵生成器 715 创建图 15 中解说的矩阵 SE 并将附加的静态关键字 S_0、S_1、…作为其另一输入。这两个矩阵的级联被转发给 PI 矩阵生成器 720，PI 矩阵生成器 720 创建矩阵 PI。LT 矩阵生成器 710 的操作可与图 16 中详述的 LT 矩阵生成器 347 的操作完全相同。静态矩阵生成器 715 的操作可与图 14 中的静态矩阵生成器 350 的操作不同。具体地，图 19 详述了此类操作的示例性实施例。

[0199] 在步骤 725，矩阵 SE 被初始化为 0。在步骤 730，等于 LDPC 码元数目的输入变量 S 被提供给该过程，且对于对 (i，j) 在 i = j mod S、i = (1+floor(j/S))+j mod S、或 i = 2*(1+floor(j/S))+j mod S 时，SE(i，j) 的值被设为 1。在步骤 735，与图 9 中的单位矩阵 IS 相对应的位置被设为 1。在步骤 740，与矩阵 T 相对应的位置被提供作为该步骤的附加输入。该矩阵可具有多个有限域中的条目，且对于不同的应用可以不同。它可以基于代码要求的需求来选取。

[0200] 图 20 是解说 PI 矩阵生成器 720 采用的方法的一个实施例的简化流程图。在步骤 745，PI 矩阵生成器 349 将 (N+R)x P 格式的矩阵 PI 初始化为全 0。接下来，在步骤 750，使用关键字 I_a、I_b、…来分别生成权重 WP(0)、…、WP(N-1) 和列表 ALP(0)、…、ALP(N-1)。列表 ALP(i) 中的每个列表包括 WP(i) 个在范围 0、…、P-1 中的整数 (j(0)，…，j(WP(i)-1))。在步骤 755，使用这些整数将条目 PI(i，j(0))、…、PI(i，J(WP(i)-1)) 设为 1。图 20 中的 PI 矩阵生成器的操作类似于图 17 的 PI 矩阵生成器的操作，不同之处在于该矩阵生成器创建具有多 R 行的矩阵且与图 15 的矩阵紧密相关。

[0201] 图 12 或图 15 中的方程组典型地是稀疏的，即所涉及矩阵中非零条目的数目典型地远小于可能的条目的一半。在这种情形中，可能无需直接存储这些矩阵，而是可存储帮助

重新创建这些矩阵的每个个体条目的指示。例如，对于矩阵 LT 或 PI 的每一行，过程可能想要存储如图 5-6 中计算出的权重和邻元列表。其他方法也是可能的，且其中许多方法已在本文中或在通过援引纳入于此的公开中作了解释。

[0202] 一旦矩阵生成器已创建图 12 或图 15 给出的形式的方程组，解码器的任务将是求解该方程组的未知值 C(0)、…、C(L-1)。可应用数种不同方法来达成此目标，包括但不限于高斯消元法、或 Luby I、Luby II、Shokrollahi I、II、III、IV、或 V 中描述的任何方法。

[0203] 现在参照图 21-26 略述用于求解图 12 或图 15 中的方程组的可能方法。根据本发明的一些实施例的解码器的操作的流程图在图 21 中给出。在步骤 1305，使用早先描述的一些方法创建解码矩阵。在步骤 1310，使用行和列置换来重新安排该矩阵。如以上提及的，此类矩阵可从图 12 或图 15 的矩阵通过应用行和列置换来获得。链式反应解码结合 Shokrollahi II 的运行中钝化解码可被用于达成此目的。因此存在对集合 {0,1，…,L-1} 操作的置换 pi 以及对集合 {0,1，…,N+R-1} 操作的置换 tau 以使得满足图 22 中的方程。

[0204] 在本文中，w 表示图 13 中的矩阵 LO 的行和列的数目，即既未被永久钝化也未被 OTF 钝化的中间码元的数目。在步骤 1315，使用图 13 的矩阵 LO 将矩阵 LO 在对角线以下的所有条目消零。通过这样做，图 23 中的方程右侧的码元集合需要遵守相同的操作，从而通过对一些 D(tau(i)) 的异或来获得该方程组新的右侧。

[0205] 如图 24 中解说的，在此操作之后，矩阵 810 变成单位矩阵，840 中的矩阵 EL 将不被触及，且矩阵 OTFI 和 PI 将变为 820 中的 OTFI-2 和 830 中的 PI-2，因为解码过程需要根据为了将矩阵 LO 简化为单位矩阵所必需的操作将这些矩阵的行异或在一起。

[0206] 解码过程的下一步可为步骤 1320，其中剩余矩阵在 LO 之下的其余部分被消元以获得图 25 中所指示形式的矩阵。将原始码元 D(0)、…、D(N_R-1) 在此步骤之后的经置换和简化值记为 E(0)、…、E(N+R-1)，将矩阵 EL_2 的行数记为 u，以及将 EL_2 的列数记为 g，图 25 中的矩阵的结构根据式 3 关于 C(pi(L-g))、…、C(pi(L-1)) 的值得到较小的有 u 个线性方程的方程组。

[0207] $$(EL\_2)\bullet\begin{pmatrix} C(pi(L-g)) \\ \vdots \\ C(pi(L-1)) \end{pmatrix}=\begin{pmatrix} E(N+R-u) \\ \vdots \\ E(N+R-1) \end{pmatrix}$$ 式 3

[0208] 诸如图 21 中描述的解码过程可在步骤 1330 中通过各种手段求解该方程组，例如通过使用高斯消元过程、或链式反应编码与高斯消元法的组合、或通过钝化解码的其他应用、或通过其他手段来求解。若矩阵 EL 具有属于多个域的元素，则高斯消元法可被修改从而将 GF(2) 中的计算与较大域诸如 GF(256) 中的计算分开，如举例而言在 Shokrollahi IV 中教示的。

[0209] 若式 3 的方程组不是使用解码器采用的这些过程可解的，则解码器可在步骤 1335 中应用对策。此类对策可包括标记差错并停止该过程，或者可包括请求更多经编码码元，或者可停止该过程并向使用该解码器的应用返回它迄今为止已能够恢复的中间码元或源码元的列表。若该方程组是可解的，则解码器可恢复钝化中间码元 C(pi(L-g))、…、C(pi(L-1)) 的值。在一些变形中，在步骤 1330 中还可恢复除钝化中间码元之外的其他一些中间码元。

[0210] 一旦恢复出这些码元的值，解码器就行进到涉及回代的步骤 1340。恢复出

C(pi(L-g))、…、C(pi(L-1)) 的值得到图 26 中给出的类型的方程组。该方程组比一般方程组更容易求解。例如，解码器可使用图 23 中指示的过程来这样做。获得图 23 右侧的第一矢量的过程可被称为回代，因为它是将已知码元的值代入该方程组的过程。如本领域普通技术人员在阅读本公开后可以看出的，图 23 和 26 中给出的方程组在算术上是等效的。

[0211] 在图 23 中，解码器通过实现其中使用矩阵乘法规则将矩阵右侧的条目乘以已解出矢量 C(pi(L-g))、…、C(pi(L-1)) 的条目并将所获得的条目与 E(0)、…、E(L-g-1) 异或的过程来获得未知值 C(pi(0))、…、C(pi(L-g-1))。将所获得的条目与 E(0)、…、E(L-g-1) 异或且因此恢复 C(pi(0))、…、C(pi(L-g-1)) 的值的过程包括图 21 中的解码器的步骤 1345。

[0212] 尽管在一些应用中是有用的，但该方法在一些优选实施例中可能导致较大的计算开销，因为图 23 右侧的矩阵典型地不是稀疏的，因此为了获得元素 C(pi(j)) 之一，不得不执行与 g 成比例的数次异或。在一些实施例中，该数目可能很大，例如由于永久钝化的数目 P 开始被选取成较大，且 g 可至少为 P 这么大。这可能对永久钝化码元的数目即 P 的值造成严格限制，且若使用较小的 P 值，则这可能导致 OTF 钝化中间码元的数目增大。

[0213] 图 27 描述了可能比图 21 中描述的过程在计算上更高效的经修改解码过程。该过程的步骤 1405 到 1435 可与图 14 中的过程的相应步骤相同。可任选地，该过程可在附加存储器位置保存图 12 或图 15 中的原始矩阵的副本、或此矩阵的相关部分、以及原始码元 D(0)、…、D(N+R-1) 供将来使用。这对于该过程的运转不是必要的，但若应用具有足够的存储器资源来保存这些副本则可能导致进一步的速度优点。替换地，该过程可仅保存原始码元 D(0)、…、D(N+R-1) 的副本而不保存矩阵，并在需要矩阵时重新创建矩阵。步骤 1440 使用矩阵的所存储副本或者撤销步骤 1415 中的过程以再获得图 22 中的原始方程组、或者图 28 中给出的仅该方程组的上部。此时，图 29 中给出的矩阵 1510 是稀疏的，且 C(pi(w))、…、C(pi(L-1)) 的值是已知的，其中 w = L-g。

[0214] 如公知的，图 29 中的方程的右侧可经由涉及码元的少量异或——即等于矩阵 OTFI 中非零条目的数目加上矩阵 PI 中非零条目的数目——的计算高效过程来计算。该过程的此步骤在图 27 中由 1445 表示。在该步骤完成之后，图 29 中方程的右侧已计算出，且将求解其中未知数为 C(pi(0))、…、C(pi(w-1)) 的值的方程组。该方程组可在步骤 1450 中使用链式反应解码来求解，因为右侧的下三角 LO 是稀疏的，即为了求解该方程组而对码元进行异或的次数等于矩阵 LO 中非零条目的数目且该数目典型地远小于 w*w，w*w 为可能的非零条目的最大数目。

[0215] 永久钝化数目的选取

[0216] 永久钝化数目的选取会影响整体性能，因此会是重要的。另一方面，该数目需要被选取为尽可能大：若该数目是大的，则 OTF 钝化的数目可能减少到非常小的数目，有时甚至为 0。这是因为图 15 中 LT 和 SE 矩阵的组合（或者其在图 23 中的相应变形）实际上是具有大开销的链式反应码的解码矩阵。这一事实使得 OTF 钝化的数目很小。在某些实施例中，OTF 钝化可能更难管理，因此减少其数目可能在速度和 / 或存储器的意义上得到优点。

[0217] 另一方面，增加永久钝化的数目可能对运行时间有不利影响：例如，图 21 的解码过程中的步骤 1330 以及图 27 的过程中的相应步骤 1430 要求求解具有至少 P 行和 P 列的方程组。这样做的一种方式将是标识图 25 中的矩阵 EL-2 的可逆子矩阵，对该矩阵求逆，并

使用该逆矩阵来获得中间码元 C(pi(L-g-1))、…、C(pi(L-1)) 的值。由于在许多实施例中矩阵 EL-2 可能不是稀疏的，因此获得这些中间码元的值可能引起对码元 g x g 次异或的量级。由于 g 至少为 P，因此对码元的异或次数可能至少为 P x P，因此若对码元的异或的总次数要在 K 内保持线性，则好的选择是将数目 P 设为与 K 的平方根成比例。附录 A 的具体实施例将 P 选取为 2.5*sqrt(K) 的量级，并保持与该观察一致。这是 P 的良好选取，因为在 P 的该选择下，典型地 OTF 钝化的数目相当小，从约 P 变化到非常接近或等于 0。

[0218] 另一个感兴趣的量是经编码码元或静态码元存在的钝化中间码元邻元的平均数 I。图 27 中的解码过程的步骤 1445 可能需要每个未恢复中间码元平均有多达 I 次码元异或来完成该步骤。若 I 是大的，则该异或次数对于执行解码的过程或编码过程的存储器和计算资源来说可能过多。另一方面，若 I 太小，则图 25 的矩阵 EL-2 可能不具有满秩，且可解码性可能受到危害。

[0219] 更详细的分析揭示了永久钝化的重要方面是使得图 15 的矩阵 PI 的行为是各列彼此线性独立，即该矩阵尽可能是满秩的。本领域技术人员公知的是，若 PI 是随机二进制矩阵，则可达成可能的最大限度的满秩。另一方面，PI 可在每列中平均具有与 K 的平方根成反比的 1 的分数，且仍满足与纯随机矩阵相同的秩属性。出于该原因，附录 A 中的具体实施例将 I 选取为 2 与 3 之间的数字，因此 P 选取为与 K 的平方根成比例，这意味着 PI 的每列中 1 的数目平均与 K 的平方根成反比。

[0220] 存在这些方法的许多变形，如本领域技术人员在阅读本公开之际将认识到的。例如，异或可用其他运算来代替，例如，较大有限域上的线性算子，或者这些算子可以是不同算子的混合，例如大有限域上的一些线性算子用于一些操作，而较小的大有限域上的其他线性算子用于其他操作。

[0221] 参照附录 A 的具体示例

[0222] 如以上详述的，在没有永久钝化（即，关于哪些经编码码元将不是将作为确定链式反应解码的序列的一部分的矩阵操纵的部分的预定决定）的情况下，OTF 钝化的数目可能是相当随机的且在存储器消耗的意义上导致潜在问题。在源码元的数目非常大且开销很小的场合，差错概率可能难以接受地接近 1。

[0223] 由于小开销的高差错概率，在源码元数目很大时要找出良好的系统信息可能变得愈加困难。本文中，系统信息是指需要提供给编码器和解码器才能在 Shokrollahi III 的意义上构造系统码的信息。此外，每当获得系统信息时，可预期代码行为远远偏离其平均行为，因为“平均”来说该码应在 0 开销时失败。

[0224] 用于通过永久钝化来构造链式反应码的一些参数可包括用于图 4 的 LT 编码器 250 的度分布 Ω、用于 PI 编码器 240 的参数、关于永久钝化码元数目的确定、关于冗余静态码元的数目及其结构的确定，以及可生成并在图 1 的编码器 115 和解码器 155 之间共享随机数的特定方式。

[0225] 使用 RQ 码的编码器和解码器

[0226] 使用本文中描述的方法的在下文被称为“RQ 码”的代码的优选实施例在附录 A 的第 5 节中更详细地指出。附录 A 的其余部分描述了应用 RQ 码在广播或多播网络上进行可靠的对象递送的一种方法。

[0227] RQ 码使用先前以及下文描述的方法来实现系统码，意味着所有源码元都在会被生

成的经编码码元中，因此经编码码元可被视为原始源码元与由编码器生成的修复码元的组合。

[0228] 尽管一些先前代码具有良好的属性，但存在将提高其实际应用的一些改善。两项重要的潜在改善是更陡峭的开销－失败曲线以及每源块更大数目的所支持源码元。开销是收到的经编码码元数目与源块中的源码元数目之差，例如开销 2 意味着接收 K+2 个经编码码元来解码具有 K 个源码元的源块。给定开销处的失败概率是当收到经编码码元数目对应于该开销时解码器未能完全恢复源块的概率。开销－失败曲线是失败概率作为从开销 0 开始递增的开销的函数如何下降的标绘。若解码器的失败概率作为开销的函数下降得快或陡峭，则开销－失败曲线较好。

[0229] 随机二进制码具有其中失败概率对于每个额外的开销码元基本上按因子 2 下降的开销－失败概率曲线，具有不可行的计算复杂性，但本讨论的主题限于开销－失败概率曲线而非计算复杂性。在一些应用中，这是充分的开销－失败曲线，但对于其他一些应用，优选更陡峭的开销－失败曲线。例如，在流送应用中，源块中的源码元数目的范围可能很宽，例如 K = 40、K = 200、K = 1,000、K = 10,000。为了提供良好的流送体验，可能要求失败概率较低，例如失败概率为 10^{-5} 或 10^{-6}。由于带宽对于流送应用而言往往是珍贵的，因此作为源码元的分数发送的修复码元百分比应当最小化。例如，假设在使用具有 K = 200 的源块时应保护发送流的网络避免高达 10％的分组丢失，且要求失败概率至多为 10^{-6}。随机二进制码要求至少 20 的开销来达成 10^{-6} 的失败概率，即接收机需要 220 个经编码码元才能以该失败概率进行解码。每个源块需要发送总共 245 个经编码码元以满足该要求，因为 ceil(220/(1-0.1)) = 245。因此，修复码元对该流的带宽要求增加了额外的 22.5％。

[0230] 对于值 K = K′（针对所有所支持的 K′ 值）以及对于 K = 1 和 K = K′ +1（针对除了 K 的最末所支持值以外的所有值），本文中以及附录 A 的第 5 节中描述的 RQ 码对于开销 0、1 和 2 分别达成小于 10^{-2}、10^{-4} 和 10^{-6} 的失败概率。已对各种丢失概率进行了测试，例如丢失概率 10％、20％、50％、70％、90％和 95％。

[0231] 对于使用 RQ 码的以上示例，开销 2 足以达成失败概率 10^{-6}，因此对于每个源块只需要发送总共 225 个经编码码元就能满足该需求，因为 ceil(202/(1-0.1)) = 225。在这种情形中，修复码元对该流的带宽要求增加了额外的 12.5％，即比随机二进制码要求的带宽开销少 10％。因此，改善了开销－失败曲线的 RQ 码具有一些非常正面的实际结果。

[0232] 存在其中期望支持每源块的大量源码元的应用。例如，在移动文件广播应用中，从网络效率的观点来看有利的是将文件作为单个源块来编码，或更一般地，将文件分成可行的尽量少的源块。例如假设要广播有 5 千万字节的文件且用于携带经编码码元的每个分组内的可用大小为一千字节。为了将该文件作为单个源块来编码要求支持 K = 50,000 的值。（注意，存在如先前描述的允许使用少得多的存储器来解码的分子块技术）。

[0233] 代码支持的源码元数目可能受限有若干原因。一个典型的原因是诸如对于 Reed-Solomon 码，计算复杂性随着 K 增加而变得不合理，但诸如链式反应码之类的代码不是这种情形。另一个原因可能是 0 开销处的失败概率随着 K 增加而增大到几乎 1，使得更加难以找出产生良好系统码构造的系统索引。0 开销处的失败概率可指示推导良好码构造的难度，因为这实质上是在随机地选取系统索引时结果得到的系统码构造具有头 K 个经编码码元能解码 K 个源码元的属性的概率。

[0234] 由于RQ码设计的开销-失败曲线对于K的所有值都如此陡峭，因此可能容易找出良好的系统索引且因此支持大得多的K值。如附录A第5节中描述的RQ码支持高达56,403的K值，且还支持每源块最高达16,777,216的经编码码元总数。对RQ码的所支持值的这些限制是由于基于感知到的应用要求的实际考虑来设定的，而不是由于RQ码设计的局限性。除附录A中示出的那些以外的其他实施例可具有不同的值。

[0235] RQ码如下限制所支持的不同源块大小的数目。给定要编码或解码的具有K个源码元的源块，基于附录A第5.6节中示出的表来选择K′值。该表的第一列列出了K′的可能值。所选择的K′值是可能性中使得K≤K′的最小值。K个源码元C′(0)、…、C′(K-1)用值被设为0的K′-K个码元C′(K)、…、C′(K′-1)来填充，以产生包括K′个源码元C′(0)、…、C′(K′-1)的源块，并且随后对该被填充的源块执行编码和解码。

[0236] 以上办法具有减少需要支持的系统索引数目的益处，即只有几百个而非几万个。对于K而言在开销-失败概率的意义上没有缺点，因为其与所选K′的开销-失败曲线相同：给定K的值，解码器就能计算出的值K′，并将C′(K)、…、C′(K′-1)的值设为0，且因此其只需要解码源块的K′个源码元中其余的K个。唯一的潜在缺点是用略多的源码元来编码和解码可能需要略多的存储器或计算资源。然而，K′的连贯值之间的间距对于K′的较大值约为1％，因此该潜在缺点是可忽略的。

[0237] 由于源块从K填充到K′，因此RQ码内的经编码码元C′(0)、C′(1)…的标识符被称为内部码元标识符，缩写为ISI，其中C′(0)、…、C′(K′-1)是源码元而C′(K′)、C′(K′+1)…是修复码元。

[0238] 采用该编码器和解码器的外部应用使用范围从0到K-1的经编码码元标识符（也称为编码码元标识符，缩写为ESI）来标识原始源码元C′(0)、…、C′(K-1)，且使用继续K、K+1…的经编码码元标识符来标识修复码元C′(K′)、C′(K′+1)…。因此，在RQ码内用ISI X标识的修复码元C′(X)在外部用ESI X-(K′-K)来标识。这在附录A第5.3.1节中更详细地描述。

[0239] RQ码的编码和解码由两种类型的关系来定义：中间码元之间的约束关系以及中间码元与经编码码元之间的LT-PI关系。该约束关系对应于由例如图12或图15中示出的SE矩阵定义的中间码元之间的关系。LT-PI关系对应于由例如图12或图15中示出的LT矩阵和PI矩阵定义的中间码元与经编码码元之间的关系。

[0240] 编码通过基于以下各项确定中间码元值而进行：(1)源码元值；(2)源码元与中间码元之间的LT-PI关系；以及(3)中间码元之间的约束关系。修复码元的值可基于中间码元与修复码元之间的LT-PI关系从中间码元生成。

[0241] 类似地，解码通过基于以下各项确定中间码元值而进行：(1)收到经编码码元值；(2)收到经编码码元与中间码元之间的LT-PI关系；以及(3)中间码元之间的约束关系。缺失源码元的值可基于中间码元与缺失源码元之间的LT-PI关系从中间码元生成。因此，编码和解码基本上是对称规程。

[0242] 示例硬件组件

[0243] 图30-31解说了可用来实现以上描述的方法的硬件的框图。每个元件可以是硬件、由通用或专用处理器或组合执行的程序代码或指令。

[0244] 图30解说示例编码系统1000，其可实现为硬件模块、软件模块、或存储在程序存

储 1002 中并由处理器 1004 执行的可能作为并非如图中所示地分开的综合代码单元的程序代码的部分。编码系统 1000 接收传达源码元和参数信息的信号输入，并输出传达该信息的信号。

[0245] 输入接口 1006 将传入的源码元存储到源码元缓冲器 1008 中。源 - 中间码元生成器 1010 从源码元生成中间码元。这在一些实施例中可以是直通器而在其他实施例（诸如“系统”实施例）中可以是解码器模块。

[0246] 冗余码元生成器 1012 从源码元生成冗余码元。这可实现为链式反应编码器、LDPC 编码器、HDPC 编码器或类似的编码器。钝化器 1014 接收源码元、中间码元和 / 或冗余码元，视情形而定，并将其中一些——永久钝化码元——存储在 PI 缓冲器 1018 中并将其他码元提供给输出编码器 1016。该过程可能仅是逻辑上的而非物理上的。

[0247] 诸如异或算子之类的算子 1020 对来自输出编码器 1016 的一个或更多个经编码码元（在某些实施例中为 1 个）以及来自 PI 缓冲器 1018 的一个或更多个 PI 码元（在某些实施例中为 1 个）进行运算，且该运算的结果被提供给传送接口 1030，传送接口 1030 从系统 1000 输出信号。

[0248] 图 31 解说示例解码系统 1100，其可实现为硬件模块、软件模块、或存储在程序存储 1102 中并由处理器 1104 执行的可能作为并非如图中所示地分开的综合代码单元的程序代码的部分。一些过程可能仅是在逻辑上而非物理上实现的。

[0249] 解码系统 1100 取得输入信号以及可能的其他信息并且若能够则输出源数据。信号输入被提供给接收接口 1106，接收接口 1106 将收到码元存储在缓冲器 1108 中。收到码元的 ESI 被提供给矩阵生成器 1110，矩阵生成器 1110 如本文中所描述地依赖于接收到的特定码元生成矩阵，并将结果存储在矩阵存储器 1112 中。

[0250] 调度器 1114 可从矩阵存储器 1112 读取矩阵细节并生成调度，该调度存储在调度存储器 1016 中。调度 1114 还可在完成时生成完成信号并将 PI 矩阵传达给 PI 求解器 1118。PI 求解器 1118 将解出的 PI 码元值提供给求解器 1120，求解器 1120 还使用该调度从收到码元、调度和 PI 码元解码中间码元。

[0251] 中间码元被提供给中间 - 源码元生成器 1122，中间 - 源码元生成器 1122 可以是编码器或直通器。中间 - 源码元生成器 1122 的输出被提供给输出接口 1124，输出接口 1124 输出源数据或可供输出的任何源数据。

[0252] 其他考虑

[0253] 在某些情形中，可能需要增强的可解码性。在本文中别处提供的示例中，当经编码码元具有 LT 邻元和 PI 邻元两者时，LDPC 码元仅具有不在 HDPC 码元中的 LT 邻元或 PI 邻元。在一些实例中，若 LDPC 码元也具有包括 HDPC 码元的 PI 邻元，则可解码性得到改善。有了包括 HDPC 码元的所有 PI 码元之间的邻元，LDPC 码元的解码价值可能更类似于经编码码元的解码价值。如本文中别处解释的，码元依赖于 LT 码元（其能容易地编码和解码）且还依赖于包括 HDPC 码元（其可提供高可靠性解码）的 PI 码元，因此可存在两种优点。

[0254] 在一示例中，每个 LDPC 码元具有两个 PI 邻元，即 LDPC 码元的值取决于两个 PI 码元的值。

[0255] 在减少重复经编码码元的出现的一些情形中，可解码性也可得到改善，其中两个经编码码元若具有完全相同的总邻元集合则是重复，其中经编码码元的总邻元集合包括 LT

邻元集合和 PI 邻元集合。具有相同的总邻元集合的重复经编码码元携带完全相同的关于生成这些经编码码元的中间源块的信息，因此收到一个以上重复经编码码元与收到重复经编码码元之一相比在解码时并不存在更好的机会，即接收一个以上重复码元增加了接收开销且重复中的经编码码元中的仅一个对于解码是有用的。

[0256] 优选属性是每个收到经编码码元不是任何其他收到经编码码元的重复，因为这意味着每个收到经编码码元对于解码可能都是有用的。因此，可能优选减少此类重复的数目或减少发生重复的概率。

[0257] 一种办法是限制每个经编码码元可具有的 LT 邻元的数目。例如，若存在 W 个可能的邻元，则邻元的最大数目可被限制为 W-2。这在一些情形中减少了总邻元集合将被重复的机会，因为包括所有 W 个可能邻元的邻元集合将不被允许。在约束是 Deg[v] = min(d,W-2) 的场合，存在度为 W-2 的 W*(W-1)/2 个不同的邻元集合。因此，不大可能为经编码码元生成重复的总邻元集合。可改为使用其他约束，诸如对于除 Wg = 2 以外的一些 Wg 为 min(d, W-Wg)，或者其他某种约束。

[0258] 可单独使用或与以上的重复减少技术一起使用的另一种技术是为每个经编码码元选取一个以上 PI 邻元，使得经编码码元不大可能有重复 PI 邻元，且不大可能为经编码码元生成重复的总邻元集合。PI 邻元可按如何生成 LT 邻元的类似方式生成，例如通过首先根据以下代码片断如附录 A 第 5.3.5.4 节中所示地生成 (d1, a1, b1)：

[0259] if(d < 4)then{d1 = 2+Rand[y,3,2]}else{d1 = 2}；

[0260] a1 = 1+Rand[y,4,P1-1]；

[0261] b1 = Rand[y,5,P1]；

[0262] 注意在该示例中，存在意义重大的针对 PI 邻元数目 d1 定义的随机度分布，且该分布依赖于所选取的 LT 邻元数目 d，且当 LT 邻元数目较小时 PI 邻元数目很可能较大。这提供了以下属性：经编码码元的总体度使得减少将生成以及因此接收的重复经编码码元的机会。

[0263] 经编码码元值可如附录 A 第 5.3.5.3 节中所示地并通过以下代码片断使用由 (d1, a1, b1) 定义的邻元来生成：

[0264]

```
while (b1 >= P) do {b1 = (b1+a1) % P1};
    result = result ^ C[W + b1];
For j = 1, ..., d1-1 do
    b1 = (b1+a1) % P1;
    while (b1 >= P) do {b1 = (b1+a1) % P1};
    result = result ^ C[W + b1];
    Return result;
```

[0265] 为了支持这些可解码性特征或者分开地提供可解码性，可以使用 K′ 值的不同系统索引 J(K′)，诸如附录 A 第 5.6 节的表 2 中示出的系统索引。

[0266] 可在传输和 / 或接收系统中执行以生成系统索引 J(K′) 的过程的示例如下解说。

对于可能 K′ 列表中的每个 K′，典型地可由恰适地编程的电路或处理器执行的一个过程是检查索引数目的合适性。例如，该电路 / 处理器可检查，对于 J = 1…1000[或其他某个极限]，关于可能的系统索引 J 是否满足以下准则：

[0267] (a) 在 0 开销处从 K′ 个源码元解码是可能的？

[0268] 若是，则记录运行中钝化的数目

[0269] (b) 头 K′ /0.06 个可能的经编码码元（具有 ESI 0、…、K′ /0.06）之间存在重复的总邻元集合？[可改为使用其他阈值。]

[0270] (c) 当使用头 K′ 个收到经编码码元在 10,000 个循环 [或其他某种测试] 内解码时，若每个经编码码元与其他经编码码元独立地在每个循环内有 0.93[或其他某个阈值] 的概率丢失，解码失败概率低于 0.007[或其他某个阈值]？

[0271] 该电路 / 处理器随后在满足以上准则 (a)、(b) 和 (c) 的可能系统索引 J 中进行选择，从而选取在步骤 (a) 中记录运行中钝化的平均数目的系统索引。

[0272] 注意，上面的选择准则有许多变形。例如，在一些情形中，可能优选选取满足上面的 (a)、(b) 和 (c) 且在指定的循环数目内在步骤 (c) 中产生最少数目的解码失败的系统索引。作为另一示例，在选取系统索引时可考虑运行中钝化的数目与解码失败概率的组合。作为另一示例，每个 K′ 值可能有多个系统索引可用，则在特定应用内随机地选取它们之一。

[0273] 附录 A 第 5.6 节中的表 2 中列出的 K′ 值的系统索引是附录 A 中描述的代码的系统索引的一个潜在可能的列表。

[0274] 分子块过程的变形

[0275] 出于各种目的已知将块分成较小单位以在物理上或逻辑上进行进一步处理的分子块。例如，它在 IETF RFC 5053 中使用。从美国专利号 7,072,971 也知道它。分子块方法的一个主要用途是允许 FEC 码将较大的数据块作为单个实体来保护，同时在接收机处使用 FEC 解码器通过使用比该数据块的大小小得多的存储器量来恢复该数据块。

[0276] 用于选取 IETF RFC 5053 中描述的子块数目的一种方法对于许多合理的参数设置提供良好的源块划分和子块划分，但在一些环境中可能产生可能不严格满足子块大小的上限 WS 的解（但即使在这些情形中，也产生其中子块大小比子块大小的给定约束 WS 大适度因子的解）。作为另一示例，在 draft-luby-rmt-bb-fec-raptorg-object-00（其中源块中的最大源码元数目远大于 IETF RFC 5053 中源块中的最大源码元数目）第 4.2 节中，提供以下处方来演算 T、Z 和 N，其中 T 是码元大小，Z 是将文件（或数据块）分成的源块的数目，而 N 是子块的数目。另外，P′ 是码元的分组有效载荷大小，F 是以字节计的文件大小，K′ _max 是所支持的最大源码元数目（例如，56,404），A1 是指定码元或子码元的大小应为 A1 字节的倍数以允许更高效解码的对准因子，例如对于现代 CPU 而言 A1 = 4 是优选的，且 WS 是以字节计的子块大小的合意上限。

[0277] 注意，参数 T、Z 和 N 的推导可在发送方或替换性服务器上基于 F、A1 和 P′ 的值来进行。接收机仅需要知晓 F、A1、T、Z 和 N 的值就能在关于文件或数据块的收到分组中确定该文件或数据块的子块和源块结构。接收机可从收到分组的大小确定 P′。注意，所发送和接收的分组典型地还包含标识分组内容的其他信息，例如典型地大小为 4 字节且携带源块编号 (SBN) 的 FEC 有效载荷 ID、以及分组中携带的第一个码元的经编码码元标识符 (ESI)。

[0278] 在 draft-luby-rmt-bb-fec-raptorg-object-00 第 4.2 节中描述的用于演算 T、

Z、N 的先前方法将它们设为以下值：

[0279] ● T = P′

[0280] ● Kt = ceil(F/T)

[0281] ● Z = ceil(Kt/K′ _max)

[0282] ● N = min{ceil(ceil(Kt/Z)*T/WS), T/Al}

[0283] 在这些演算中，ceil() 是输出大于或等于其输入的最小整数的函数，而 floor() 是输出小于或等于其输入的最大整数的函数。另外，min() 是输出其输入中的最小值的函数。

[0284] 这样推导源块和子块划分的一些参数设置的一个问题在于若 T/Al 小于 ceil(ceil(Kt/Z)*T/WS)，则可能不遵守子块大小的上限 W。

[0285] 潜在的附属问题在于这允许子码元小达 Al，Al 典型地设为 4 字节，且可能太小而实际上不是高效的。典型地，子码元大小越小，解码或编码子块的处理开销越多。此外，尤其是在接收机处，子码元大小越小意味着需要解复用和解码的子块越多，且这会消耗诸如 CPU 循环和存储器存取之类的接收机资源。另一方面，可允许子码元大小越小意味着可将源块分成越多遵守子块大小的指定上限 WS 的子块。因此，子码元越小允许支持的源块越大，且因此跨该源块提供的 FEC 保护产生更好的保护和更好的网络效率。实际上，在许多情形中，优选确保子码元至少为指定的最小大小，这提供在接收机处更好地平衡处理需求与存储器需求的机会，以及对网络资源的高效利用。

[0286] 作为所推导参数的示例，使用 draft-luby-rmt-bb-fec-raptorg-object-00 第 4.2 节中描述的先前方法来演算 T、Z、N：

[0287] 输入：

[0288] F = 56,404KB

[0289] P′ = 1KB = 1,024 字节

[0290] WS = 128KB

[0291] Al = 4

[0292] K′ _max = 56,404

[0293] 演算：

[0294] T = 1KB

[0295] Kt = 56,404

[0296] Z = 1

[0297] N = 256（由于 min 函数的第二输入）

[0298] 在该示例中，将有一个源块，包括 256 个子块，其中每个子块约为 220KB（大于 WS），其中至少一些子块具有的子码元大小为 4 字节（极其小）。

[0299] 第三个问题在于 AL-FEC 解决方案可能不支持所有可能的源码元数目，即它可能仅支持所选的 K′ 值列表，其中 K′ 是源块中所支持的源码元数目，且若源块中合意的实际源码元数目 K 不在这些 K′ 值中，则 K 被填充到最接近的 K′ 值，这意味着所使用的源块的大小可在一定程度上大于从以上演算出的 K 值。

[0300] 现在描述新的分子块方法，其是对以上描述的先前方法的改进。出于描述目的，用于分子块的模块可将待划分的数据 F 以及包括 WS、Al、SS 和 P′ 的值作为其输入，其中这些

变量的涵意在以下更详细地描述。

[0301] WS 表示在接收机的工作存储器中可解码的最大大小的子块的规定约束，可能以字节为单位。A1 表示存储器对准参数。由于接收机存储器若码元和子码元在存储器中沿存储器对准边界对准则可能更高效地工作，因此追踪 A1 并以 A1 字节的倍数来存储值可能是有用的。例如，典型地 A1 = 4，因为许多存储器设备天生在四字节边界上寻址存储器中的数据。A1 的其他值也是可能的，例如 A1 = 2 或 A1 = 8。典型地，A1 可被设为所有许多可能的接收机的最小公倍数存储器对准。例如，若一些接收机支持 2 字节存储器对准但其他接收机支持 4 字节存储器对准，则将推荐 A1 = 4。

[0302] 基于优选的子码元大小下限确定参数 SS，以使得子码元大小下限为 SS*A1 字节。可能优选的是使子码元大小为 A1 的倍数，因为解码操作典型地是对子码元执行的。

[0303] 以下是用于将数据 F 划分成 Z 个源块并然后将这 Z 个源块划分成 N 个子块的方法的详细解释。在该描述中，P′ 是指存储在存储器中的（或暗含的）表示用于要发送的码元的分组内的可用字节的变量，且假定 P′ 是 A1 的倍数。T 是表示要放在所发送分组内的码元的大小的变量。其他变量可从本文中推出。

[0304] 用于确定 T、Z 和 N 的新的分子块方法

[0305] ● T = P′

[0306] ● Kt = ceil(F/T)；

[0307] ● N_max = floor(T/(SS*A1))；

[0308] ●对于所有 n = 1，…，N_max

[0309] ○ KL(n) 是作为源块中的可能源码元数目所支持的最大 K′ 值，其满足

[0310] ■ K′ ≤ WS/(A1*(ceil(T/(A1*n))))；

[0311] ● Z = ceil(Kt/KL(N_max))；

[0312] ● N =使得 ceil(Kt/Z) ≤ KL(n) 的最小 n；

[0313] 一旦确定了这些参数，则 Z 个源块中每个源块的大小以及每个源块的 N 个子块的子码元大小就可如 IETF RFC 5053 中描述地确定，即 Kt = ceil(F/T)，(KL，KS，ZL，ZS) = Partition[Kt，Z]，以及 (TL，TS，NL，NS) = Partition[T/A1，N]。

[0314] Kt 是文件中源码元的数目。在分子块模块中，Kt 个源码元被划分成 Z 个源块：各自有 KL 个源码元的 ZL 个源块以及各自有 KS 个源码元的 ZS 个源块。随后，KL 被向上舍入到 KL′，其中 KL′ 是至少为 KL 的最小所支持源码元数目（且出于编码和解码目的将附加的 KL′ -KL 个零填充码元添加到源块，但这些附加码元典型地不被发送或接收），且类似地，KS 被向上舍入到 KS′，其中其中 KS′ 是至少为 KS 的最小所支持源码元数目（且出于编码和解码目的将额外的 KS′ -KS 个零填充码元添加到源块，但这些附加码元典型地不被发送或接收）。

[0315] （由分子块模块、其他软件模块、或硬件执行的）这些演算确保了各源块的源码元数目尽可能相等，受其数目总计为文件中的源码元数目 Kt 的约束。这些演算还确保子块的子码元大小尽可能相等，受它们是 A1 的倍数且其大小总计为码元大小的约束。

[0316] 随后，演算子码元参数 TL、TS、NL 和 NS，其中存在 NL 个使用较大子码元大小 TL*A1 的子块且存在 NS 个使用较小子码元大小 TS*A1 的子块。函数 Partition[I，J] 是以软件或硬件实现的且被定义为其输出是四个整数的序列 (IL，IS，JL，JS) 的函数，其中 IL =

ceil(I/J)、IS = floor(I/J)、JL = I-IS*J 以及 JS = J-JL。

[0317] 这些新方法的一些属性相当值得注意。分子块模块能确定关于所使用的最小子码元大小推导出的下限。根据上式，知道 TS = floor(T/(Al*N))，其中 TS*Al 是所使用的最小子码元大小，因为 TS ≤ TL。注意，在 N = N_max 时，使用最小子码元。对正 X 和 Y 使用 X/(floor(Y)) ≥ X/Y，TS 至少为 floor(T/(Al*floor(T/(SS*Al))))，其进而至少为 floor(SS) = SS。由于这些事实，由本文中描述的划分方法产生的最小子码元大小将至少为 TS*Al = SS*Al，如所需的。

[0318] 分子块模块能确定关于最大子块大小推导出的上限。所使用的最大子块大小为 TL*Al*KL′，其中 KL′ 是上表中的最小 K′ 值，其至少为 KL = ceil(Kt/Z)。注意，通过 N 的定义，KL′ ≤ KL(N) 且 TL = ceil(T/(Al*N))。由于 KL(N) ≤ WS/(Al*(ceil(T/(Al*N))))，因此得到 WS ≥ KL(N)*Al*ceil(T/(Al*N)) ≥ KL′ *Al*TL。

[0319] 变量 N_max 可表示可将大小为 T 的源码元划分成的子码元的最大数目。将 N_max 设为 floor(T/(SS*Al)) 确保了最小子码元大小至少为 SS*Al。KL(n) 是当源块的码元各自被划分成 n 个子码元以确保源块的每个子块的大小至多为 WS 时能支持的源块中的最大源码元数目。

[0320] 源块的数目 Z 可被选取为尽可能小，受每个源块中的源码元数目至多为 KL(N_max) 的约束，这确保了每个源码元可被划分成大小至少为 SS*Al 的子码元且结果得到的子块的大小至多为 WS。分子块模块从 Z 的值确定源块数目和这 Z 个源块中每个源块中的码元数目。

[0321] 注意，若使用比通过该划分方法产生的更小的任何 Z 值，则将存在源块之一的大于 WS 的子块，或者将存在源块之一的具有小于 SS*Al 的子码元大小的子块。另外，该划分方法产生的最小源块在受这两种约束的情况下尽可能大，即没有其他方法将文件或数据块划分成遵守这两种约束的源块，以使得最小源块大于通过该划分方法产生的最小源块。因此，在这种意义上，通过此划分方法产生的 Z 值是最优的。

[0322] 受对于每个子块而言子块的子码元数目乘以子块所划分的源块中的源码元数目至多为 WS 的约束，将源块划分成的子块的数目 N 可被选取为尽可能小。

[0323] 注意，若使用比通过此划分方法从 Z 值产生的更小的任何 N 值，则将存在其大小将超过 WS 的至少一个子块。另外，受最大子块大小不应超过 WS 的约束，该划分方法从给定的 Z 值产生的最小子码元大小尽可能大，即没有其他方法产生由 Z 值确定的源块的子块，其遵守最大子块约束以使得最小子码元大小大于通过该划分方法产生的最小子码元大小。因此，在这种意义上，通过此方法产生的 N 值是最优的。

[0324] 在以下示例中，假定支持所有可能的 K′ 值作为源块中的源码元数目。

[0325] 示例 1

[0326] 输入：

[0327] SS = 5

[0328] Al = 4 字节

[0329] （最小子码元大小= 20 字节）

[0330] WS = 128KB = 131,072 字节

[0331] P′ = 1,240 字节

[0332] F = 6MB = 6,291,456 字节

[0333] 演算：

[0334] T = 1,240 字节

[0335] Kt = 5,074

[0336] N_max = 62

[0337] KL(N_max) = 6,553

[0338] Z = 1

[0339] KL = ceil(Kt/Z) = 5,074

[0340] N = 52(KL(N) = 5,461)

[0341] TL = 6,较大子码元= 24 字节

[0342] TS = 5,较小子码元= 20 字节

[0343] TL*AL*KL = 121,776

[0344] 示例 2

[0345] 输入：

[0346] SS = 8

[0347] Al = 4 字节

[0348] （最小子码元大小= 32 字节）

[0349] WS = 128KB = 131,072 字节

[0350] P′ = 1KB = 1,024 字节

[0351] F = 56,404KB = 57,757,696 字节

[0352] 演算：

[0353] T = 1,024 字节

[0354] Kt = 56,404

[0355] N_max = 32

[0356] KL(N_max) = 4,096

[0357] Z = 14

[0358] KL = ceil(Kt/Z) = 4,029

[0359] N = 32(KL(N) = 4,096)

[0360] TL = 8,较大子码元= 32 字节

[0361] TS = 8,较小子码元= 32 字节

[0362] TL*AL*KL = 128,928

[0363] 上述方法有许多变形。例如，对于一些 FEC 码，在源块中具有至少最小数目的源码元以使得 FEC 码的源块接收开销最小化是合意的。由于对于实在小的文件大小或数据块大小 F，源码元的大小可能变得太小，因此可能还存在将分组划分成的最大源码元数目。例如，在 IETF RFC 5053 中，源块中的合意最小源码元数目为 Kmin = 1024，且分组划分成的最大源码元数目为 Gmax = 10。

[0364] 以下是上述新的分子块方法的计及刚才描述的附加参数 Kmin 和 Gmax 的另一变形，其中 G 是每个分组中携带的源块的码元数目，该变形可由分子块模块或更一般地由编码器、解码器、发射机和 / 或接收机上的某个模块或软件或硬件执行。

[0365] 在该变形中，每个分组携带分组中第一个码元的 ESI，且随后该分组中的每个后续码元隐式地具有比该分组中的前一码元大 1 的 ESI。

[0366] 用于确定 G、T、Z 和 N 的新的分子块方法

[0367] ● G = min(ceil(P′ *Kmin/F), floor(P′ /(SS*Al)), Gmax) ;

[0368] ● T = floor(P′ /(Al*G))*Al ;

[0369] ● Kt = ceil(F/T) ;

[0370] ● N_max = floor(T/(SS*Al)) ;

[0371] ●对于所有 n = 1, …, N_max

[0372] ○ KL(n) 是作为源块中的可能源码元数目所支持的最大 K′ 值，其满足

[0373] ■ K′ ≤ WS/(Al*(ceil(T/(Al*n)))) ;

[0374] ● Z = ceil(Kt/KL(N_max)) ;

[0375] ● N =使得 ceil(Kt/Z) ≤ KL(n) 的最小 n ;

[0376] 注意，通过演算 G 的方式，保证了码元大小至少为 SS*Al，即码元大小至少为最小子码元大小。还注意，应当是这种情形 :SS*Al 至少为 P′ ，以确保码元大小可至少为 SS*Al (且若不是，则 G 将评估为 0)。

[0377] 示例 3

[0378] 输入 :

[0379] SS = 5

[0380] Al = 4 字节

[0381] (最小子码元大小 = 20 字节)

[0382] WS = 256KB = 262,144 字节

[0383] P′ = 1,240 字节

[0384] F = 500KB = 512,000 字节

[0385] Kmin = 1,024

[0386] Gmax = 10

[0387] 演算 :

[0388] G = 3

[0389] T = 412

[0390] Kt = 1,243

[0391] N_max = 20

[0392] KL(N_max) = 10,992

[0393] Z = 1

[0394] KL = ceil(Kt/Z) = 1,243

[0395] N = 2(KL(N) = 1,260)

[0396] TL = 52，较大子码元 = 208 字节

[0397] TS = 51，较小子码元 = 204 字节

[0398] TL*AL*KL = 258,544

[0399] 如现在已描述的，这些新方法引入了对用于任何子块的最小子码元大小的约束。本公开提供了遵守该附加约束同时严格遵守最大子块大小约束的用于分子块的新方法。这

些方法产生满足在受最小子码元大小约束的情况下将文件或数据块划分成尽可能少的源块、且随后在受最大子块大小约束的情况下将源块拆分成尽可能少的子块的目标的分源块和分子块解决方案。

[0400] 变型

[0401] 在一些应用中，不能解码所有源码元或能够但是以相对低的概率来解码所有源码元可能是可接受的。在此类应用中，接收机可在接收到 K+A 个经编码码元之后停止尝试解码所有源码元。或者，接收机可在接收到少于 K+A 个经编码码元之后停止接收经编码码元。在一些应用中，接收机甚至可以仅接收 K 个或更少的经编码码元。因此，将理解，在本发明的一些实施例中，合意的准确度无需是完全恢复所有源码元。

[0402] 此外，在其中不完全恢复是可接受的一些应用中，数据可能被编码成使得不能恢复所有源码元，或者使得完全恢复这些源码元将要求接收比源码元数目多得多的经编码码元。此类编码一般将要求较少的计算花销，且因此可能是降低编码的计算花销的可接受途径。

[0403] 将理解，上述附图中的各种功能块可由硬件和 / 或软件的组合来实现，且在具体实现中，一些块的一些或所有功能性可被组合。类似地，还将理解，本文中教示的各种方法可由硬件和 / 或软件的组合来实现。

[0404] 以上描述仅是解说性的而非限制性的。在审阅本公开之际，本发明的许多变型对本领域技术人员将变得明显。因此，本发明的范围不应以上面的描述为准来确定，而是应改为以所附权利要求连同其等效技术方案的完整范围为准来确定。

[0405] 附录 A

[0406]

可靠多播传输
因特网草案
期望状态：标准跟踪
过期：2011 年 2 月 12 日

M. Luby
高通公司
A. Shokrollahi
EPFL
M. Watson
高通公司
T. Stockhammer
Nomor 研究所
L. Minder
高通公司
2010 年 8 月 11 日

[0407] 用于对象递送的 RaptorQ 前向纠错方案

[0408] draft-ietf-rmt-bb-fec-raptorq-04

[0409] 摘要

[0410] 本文档描述对应于 FEC 编码 ID 6(待确认 (tbc)) 的用于 RaptorQ 前向纠错码及其可靠数据对象递送应用的全规范的 FEC 方案。

[0411] RaptorQ 码是提供比 RFC5053 中的 Raptor 码更优越的灵活性、支持更大的源块大小和更好的编码效率的新代码系列。RaptorQ 也是喷泉码，即编码器可在运行中从源数据块的源码元生成如所需的那么多编码码元。解码器对于大多数情况能够从等于源码元数目的编码码元的任何集合恢复源块，且在极少数情况下能够用稍微多于源码元数目的编码码元的任何集合恢复源块。

[0412] 此处描述的 RaptorQ 码是系统码，意味着所有源码元都在能生成的编码码元当中。

[0413] 该备忘录的状态

[0414] 该因特网草案是完全遵照 BCP 78 和 BCP 79 的规定提交的。

[0415] 因特网草案是因特网工程任务组 (IETF) 的工作文档。注意，其他组也可分发工作文档作为因特网草案。当前因特网草案的列表在 http://datatracker.ietf.org/drafts/current/。

[0416] 因特网草案是在至多 6 个月内有效且可在任何时间被其他文档更新、取代或废弃的草案文档。将因特网草案作为参考资料或将它们引述为非“进行中”是不合适的。

[0417] 本因特网草案将在 2011 年 2 月 12 日过期。

[0418] 版权申明

[0419] 版权 (c)2010IETF Trust 和人员被标识为文档作者。保留所有权利。

[0420] 本文档符合 BCP 78 以及 IETF Trust 关于 IETF 文档对本文档发布日期的影响的法律条款 (http://trustee.ietf.org/license-info)。请仔细审阅这些文档，因为它们描述了读者关于本文档的权利和限制。从本文档提取的代码分量必须包括如 Trust 法律条款第 4.e 节中描述的简化 DSD 许可文本，且不保证如简化 BSD 许可中描述中那样提供。

[0421] 目录

[0473] 1. 介绍

[0474] 本文档规定了用于对象递送应用的 RaptorQ 前向纠错码的 FEC 方案。FEC 方案的概念在 RFC5052[RFC5052] 中定义且本文档遵循其中规定的格式并使用该文档的术语。本

文中描述的 RaptorQ 码是 RFC5053[RFC5053] 中描述的 Raptor 码的下一代。RaptorQ 码提供比 RFC5053[RFC5053] 的 Raptor 码更优越的可靠性、更好的编码效率，且支持更大的源块大小。这些改进与 RFC5053[RFC5053] 相比简化了在对象递送内容递送协议中使用 RaptorQ 码。

[0475] RaptorQ FEC 方案是与 FEC 编码 ID 6(tbc) 相对应的全规范 FEC 方案。

[0476] 编者注：最终 FEC 编码 ID 仍待定，但‘6(tbc)’将在本因特网草案中用

[0477] 作临时值，期望在 IANA 注册过程中顺序地使用 FEC 编码 ID。

[0478] RaptorQ 是喷泉码，即编码器可在运行中从块的源码元生成如所需的那么多编码码元。解码器能够从在数量上仅比源码元数目略多的编码码元的任何集合恢复源块。

[0479] 本文档中描述的代码是系统码，即原始源码元可不修改地连同数个修复码元从发送方发送给接收方。关于在可靠多播中使用前向纠错码的更多背景参见 [RFC3453]。

[0480] 2. 必要注释

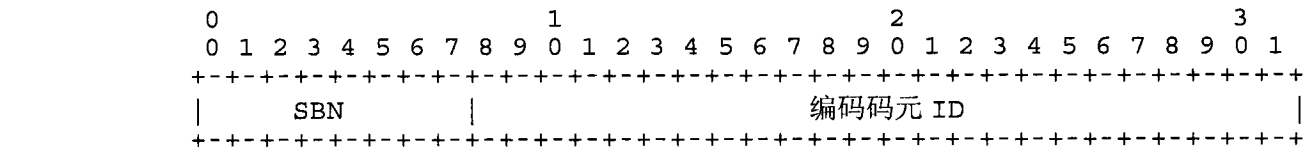
[0481] 本文档中的关键词“必须”、“必须不”、“要求”、“应当”、“不应”、“应该”、“应该不”、“推荐”、“可以”以及“可任选的”将如 [RFC2119] 中所描述地解释。

[0482] 3. 格式和代码

[0483] 3.1.FEC 有效载荷 ID

[0484] FEC 有效载荷 ID 必须为有 4 个八位字节的字段，定义如下：

[0485]



[0486] 图 1：FEC 有效载荷 ID 格式

[0487] ○源块编号 (SBN)，(8 比特，无符号整数）：分组内的编码码元涉及的源块的非负整数标识符。

[0488] ○编码码元 ID(ESI)，(24 比特，无符号整数）：分组内的编码码元的非负整数标识符。

[0489] 对源块编号和编码码元标识符的解释在第 4 节中定义。

[0490] 3.2.FEC 对象传输信息

[0491] 3.2.1. 强制性的

[0492] FEC 编码 ID 的值必须为 6，如是由 IANA 指派的（参见第 7 节）。

[0493] 3.2.2. 共同的

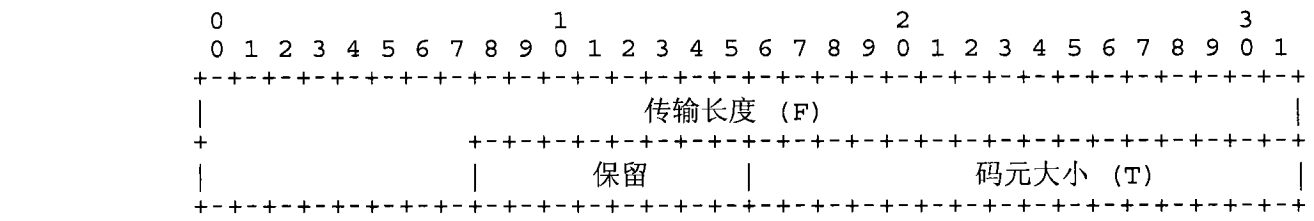
[0494] 该 FEC 方案使用的共同 FEC 对象传输信息元素为：

[0495] ○传输长度 (F)，(40 比特，无符号整数）：至多为 946270874880 的非负整数。这是以八位字节为单位的对象传输长度。

[0496] ○码元大小 (T)，(16 比特，无符号整数）：小于 2^^16 的正整数。这是以八位字节为单位的码元大小。

[0497] 经编码的共同 FEC 对象传输信息格式如图 2 中所示。

[0498]



[0499] 图 2 :用于 RaptorQ FEC 方案的经编码共同 FEC OTI

[0500] 注 1 :关于传输长度的极限 946270874880 是由于将码元大小限制为 2^^16-1、将源块中的码元数限制为 56403 以及将源块数目限制为 2^^8。

[0501] 3.2.3. 因方案而异的

[0502] 以下参数携带在用于该 FEC 方案的因方案而异的 FEC 对象传输信息元素中 :

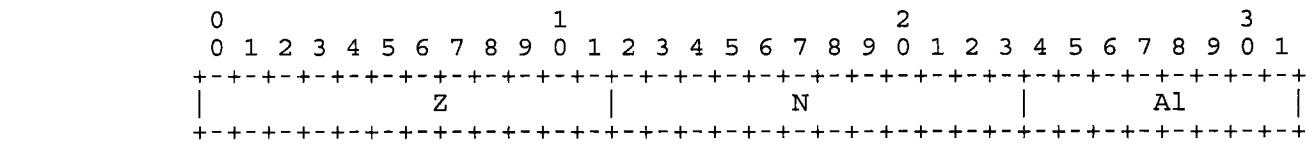
[0503] ○源块数目 (Z)(12 比特,无符号整数)

[0504] ○子块数目 (N)(12 比特,无符号整数)

[0505] ○码元对准参数 (A1)(8 比特,无符号整数)

[0506] 这些参数解为正整数。经编码的因方案而异的对象传输信息为由如图 3 中所示的参数 Z、N 和 A1 构成的有 4 个八位字节的字段。

[0507]



[0508] 图 3 :经编码的因方案而异的 FEC 对象传输信息

[0509] 经编码 FEC 对象传输信息为由经编码共同 FEC 对象传输信息和经编码的因方案而异的 FEC 对象传输信息的级联构成的有 12 个八位字节的字段。

[0510] 这三个参数定义了如第 4.4.1.2 节中描述的源块划分。

[0511] 4. 规程

[0512] 4.1. 介绍

[0513] 对于本节中使用的任何未定义符号或函数,具体而言函数“ceil”和“floor”,参见第 5.1 节。

[0514] 4.2. 内容递送协议要求

[0515] 本节描述 RaptorQ FEC 方案与利用 RaptorQ FEC 方案进行对象递送的任何内容递送协议 (CDP) 之间的信息交换。

[0516] 用于对象递送的 RaptorQ 编码器方案和 RaptorQ 解码器方案需要来自 CDP 的以下信息 :

[0517] ○以八位字节计的对象传输长度 F

[0518] ○码元对准参数 A1

[0519] ○以八位字节计的码元大小 T,其必须为 A1 的倍数

[0520] ○源块数目 Z

[0521] ○每个源块中的子块数目 N

[0522] 用于对象递送的 RaptorQ 编码器方案还需要 :

[0523] —要编码的对象为 F 个八位字节

[0524] RaptorQ 编码器方案向 CDP 提供关于要发送的每个分组的以下信息 :

[0525] ○源块编号 (SBN)

[0526] ○编码码元 ID(ESI)

[0527] ○编码码元

[0528] CDP 必须将此信息传达给接收机。

[0529] 4.3. 示例参数推导算法

[0530] 本节提供关于推导三个传输参数 T、Z 和 N 的推荐。该推荐基于以下输入参数：

[0531] ○ F,对象传输长度,以八位字节计

[0532] ○ WS,工作存储器中可解码的块的最大大小,以八位字节计

[0533] ○ P′,以八位字节计的最大有效载荷大小,其假定为 A1 的倍数

[0534] ○ A1,码元对准参数,以八位字节计

[0535] ○ SS,其中对子码元大小的合意下限是 SS*A1 的参数

[0536] ○ K′_max,每源块的最大源码元数目。

[0537] 注:第 5.1.2 节将 K′_max 定义为 56403。

[0538] 基于以上输入,传输参数 T、Z 和 N 演算如下：

[0539] 令：

[0540] ○ T = P′

[0541] ○ Kt = ceil(F/T)

[0542] ○ N_max = floor(T/(SS*A1))

[0543] ○对于所有 n = 1, …, N_max

[0544] *KL(n) 是第 5.6 节的表 2 中使 K′ <= WS/(A1*(ceil(T/(A1*n)))) 的最大 K′ 值

[0545] ○ Z = ceil(Kt/KL(N_max))

[0546] ○ N 是使得 ceil(Kt/Z) <= KL(n) 的最小 n = 1, …, N_max

[0547] 推荐每个分组正好包括一个码元。然而,接收机应当支持包含多个码元的分组。

[0548] 值 Kt 是表示对象的源数据所必需的码元总数。

[0549] 上面的以及第 4.4.1.2 节中定义的算法确保子码元大小是码元对准参数 A1 的倍数。这是有用的,因为用于编码和解码的求和运算一般每次执行若干个八位字节,例如在 32 位处理器上每次至少 4 个八位字节。因此,若子码元大小是该数目个八位字节的倍数,则编码和解码能更快速地执行。

[0550] 输入参数 A1 的推荐设置是 4。

[0551] 参数 WS 可用来生成在解码器处用有限的工作存储器就能高效地解码的经编码数据。注意,对于给定的 WS 值,实际的最大解码器存储器需求取决于实现,但使用仅比 WS 略大的工作存储器就有可能实现解码。

[0552] 4.4. 对象递送

[0553] 4.4.1. 源块构造

[0554] 4.4.1.1. 综述

[0555] 为了向源对象应用 RaptorQ 编码器,对象可被断为 Z >= 1 个被称为源块的块。RaptorQ 编码器被独立地应用于每个源块。每个源块由唯一性源块编号 (SBN) 来标识,其中第一源块具有的 SBN 为 0,第二源块具有的 SBN 为 1,依此类推。每个源块被划分成数目 K 个各自大小为 T 个八位字节的源码元。每个源码元由唯一性编码码元标识符 (ESI) 来标

识，其中源块的第一源码元具有的 ESI 为 0，第二源码元具有的 ESI 为 1，依此类推。

[0556] 具有 K 个源码元的每个源块被划分成 N ＞＝ 1 个子块，这些子块足够小以在工作存储器中解码。每个子块被划分成 K 个大小为 T′ 的子码元。

[0557] 注意 K 的值对于对象的每个源块不一定相同，且 T′ 的值对于源块的每个子块不一定相同。然而，码元大小 T 对于对象的所有源块都是相同的，且码元数目 K 对于源块的每个子块都是相同的。将对象确切地划分成源块和子块在下节 4.4.1.2 中描述。

[0558] 4.4.1.2. 源块和子块划分

[0559] 源块和子块的构造是基于 5 个输入参数 F、Al、T、Z 和 N 以及函数 Partition[] 来确定的。这 5 个输入参数定义如下：

[0560] ○ F，对象传输长度，以八位字节计

[0561] ○ A1，码元对准参数，以八位字节计

[0562] ○ T，码元大小，以八位字节计，其必须为 A1 的倍数

[0563] ○ Z，源块数目

[0564] ○ N，每个源块中的子块数目

[0565] 这些参数必须设置成使得 ceil(ceil(F/T)/Z) ＜＝ K′ _max。对这些参数的推导的推荐在第 4.3 节中提供。

[0566] 函数 Partition[] 将一对正整数 (I，J) 作为输入并推导出 4 个非负整数 (IL，IS，JL，JS) 作为输出。具体地，Partition[I，J] 的值是序列 (IL，IS，JL，JS)，其中 IL ＝ ceil(I/J)、IS ＝ floor(I/J)、JL ＝ I-IS*J 以及 JS ＝ J-JL。Partition[] 推导用于将大小为 I 的块划分成 J 个大致相同大小的块的参数。具体地，JL 个长度为 IL 的块和 JS 个长度为 IS 的块。

[0567] 源对象必须如下被划分成源块和子块：

[0568] 令：

[0569] ○ Kt ＝ ceil(F/T)，

[0570] ○ (KL，KS，ZL，ZS) ＝ Partition[Kt，Z]，

[0571] ○ (TL，TS，NL，NS) ＝ Partition[T/Al，N].

[0572] 那么，对象必须被划分成 Z ＝ ZL+ZS 个毗连的源块，头 ZL 个源块各自具有 KL*T 个八位字节，即 KL 个各自为 T 个八位字节的源码元，且其余 ZS 个源块各自具有 KS*T 个八位字节，即 KS 个各自 T 个八位字节的源码元。

[0573] 若 Kt*T ＞ F，则对于编码过程，最后一个源块的最后一个码元必须在末尾用 Kt*T-F 个 0 八位字节来填充。

[0574] 接下来，具有 K 个源码元的每个源块必须被划分成 N ＝ NL+NS 个毗连的子块，头 NL 个子块各自由 K 个大小为 TL*A1 个八位字节的毗连子码元构成，且其余 NS 个子块各自由 K 个大小为 TS*A1 个八位字节的毗连子码元构成。码元对准参数 A1 确保这些子码元总是 A1 个八位字节的倍数。

[0575] 最后，源块的第 m 个码元由来自 N 个子块中每个子块的第 m 个子码元的级联构成。注意，这意味着当 N ＞ 1 时，码元不是对象的毗连部分。

[0576] 4.4.2. 编码分组构造

[0577] 每个编码分组包含以下信息：

[0578] ○源块编号 (SBN)

[0579] ○编码码元 ID(ESI)

[0580] ○编码码元

[0581] 每个源块与其他源块独立地被编码。源块从 0 连贯地编号。

[0582] 从 0 到 K-1 的编码码元 ID 按顺序次序标识源块的源码元，其中 K 是源块中的源码元数目。编码码元 ID K 向前标识使用 RaptorQ 编码器从源码元生成的修复码元。

[0583] 每个编码分组要么全部由源码元构成（源分组）要么全部由修复码元构成（修复分组）。分组可包含来自相同源块的任何数目个码元。在源分组中的最后一个源码元包括出于 FEC 编码目的而添加的填充八位字节的情形中，这些八位字节无需被包括在该分组中。否则，仅全体码元必须被包括。

[0584] 每个源分组中携带的编码码元 ID X 是该分组中携带的第一个源码元的编码码元 ID。该分组中的后续源码元按顺序次序具有编码码元 ID X+1 至 X+G-1，其中 G 是该分组中的码元数目。

[0585] 类似地，放入修复分组中的编码码元 ID X 是修复分组中第一个修复码元的编码码元 ID，且该分组中的后续修复码元按顺序次序具有编码码元 ID X+1 至 X+G-1，其中 G 是该分组中的码元数目。

[0586] 注意，接收机不必知道修复分组的总数。

[0587] 5. RaptorQ FEC 码规范

[0588] 5.1. 定义、符号和缩写

[0589] 出于本节中的 RaptorQ FEC 码规范的目的，应用以下定义、符号和缩写

[0590] 5.1.1. 定义

[0591] ○源块：出于 RaptorQ 编码和解码目的一起考虑的 K 个源码元的块

[0592] ○扩展源块：从源块以及零个或更多个填充码元构造的 K′ 个源码元的块，其中 K′ ＞＝ K。

[0593] ○码元：数据单位。码元以八位字节计的大小称为码元大小。码元大小总是正整数。

[0594] ○源码元：编码过程期间使用的最小数据单位。源块内的所有源码元具有相同大小。

[0595] ○填充码元：向源块添加以形成扩展源块的具有全零比特的码元。

[0596] ○编码码元：可作为对源块进行编码的一部分被发送的码元。源块的编码码元由源块的源码元以及从源块生成的修复码元构成。从源块生成的修复码元具有与该源块的源码元相同的大小。

[0597] ○修复码元：源块的不是源码元的编码码元。修复码元是基于源块的源码元生成的。

[0598] ○中间码元：使用逆编码过程从源码元生成的码元。随后从中间码元直接生成修复码元。编码码元不包括中间码元，即中间码元不作为对源块进行编码的一部分被发送。中间码元被划分成 LT 码元和 PI 码元。

[0599] ○ LT 码元：可以是编码码元的 LT 邻元的中间码元的子集。

[0600] ○ PI 码元：可以是编码码元的 PI 邻元的中间码元的子集。

[0601] ○系统码：其中所有源码元被包括作为源块的编码码元的一部分的代码。如本文中所描述的 RaptorQ 码是系统码。

[0602] ○编码码元 ID(ESI)：出于发送和接收目的唯一性地标识与源块相关联的每个编码码元的信息。

[0603] ○内部码元 ID(ISI)：出于编码和解码目的唯一性地标识与扩展源块相关联的每个码元的信息。

[0604] ○对八位字节和码元以及矩阵的算术运算：使用这些运算来从源码元产生编码码元以及反之。参见第 5.7 节。

[0605] 5.1.2. 符号

[0606] i、j、u、v、h、d、a、b、d1、a1、b1、v、m、x、y 取决于上下文表示一种类型或另一种类型的值或变量。

[0607] X 取决于上下文标示要么为 ISI 值要么为 ESI 值的非负整数值。

[0608] ceil(x) 标示大于或等于 x 的最小整数，其中 x 是实值。

[0609] floor(x) 标示小于或等于 x 的最大整数，其中 x 是实值。

[0610] min(x，y) 标示值 x 和 y 中的最小值，且一般而言表示所有自变量值中的最小值。

[0611] max(x，y) 标示值 x 和 y 中的最大值，且一般而言表示所有自变量值中的最大值。

[0612] i％ j 标示 i 模 j。

[0613] i+j 标示 i 与 j 之和。若 i 和 j 是八位字节 / 码元，这指定对八位字节 / 码元的算术，如第 5.7 节中定义的。若 i 和 j 是整数，则这表示普通的整数加法。

[0614] i*j 标示 i 与 j 之积。若 i 和 j 是八位字节，这指定对八位字节的算术，如第 5.7 节中定义的。若 i 是八位字节而 j 是码元，这表示码元与八位字节的乘法，同样如第 5.7 节中定义的。最后，若 i 和 j 是整数，则 i*j 表示普通的整数乘积。

[0615] aˆˆb 标示运算 a 的 b 次幂。若 a 是八位字节而 b 是非负整数，应理解这意味着 a*a*…*a(b 项)，其中‘*’是如第 5.7 节中定义的八位字节乘积。

[0616] uˆv 标示对于等长度比特串 u 和 v，u 与 v 的逐比特异或。

[0617] Transpose[A] 标示矩阵 A 的转置矩阵。在本规范中，所有矩阵都具有为八位字节的条目。

[0618] Aˆˆ-1 标示矩阵 A 的逆矩阵。在本规范中，所有都具有八位字节作为条目，因此应理解，对矩阵条目的运算将如第 5.7 节中所陈述地进行，且 Aˆˆ-1 是 A 关于八位字节算术的矩阵逆。

[0619] K 标示单个源块中的码元数目。

[0620] K′ 标示扩展源块中的源码元加上填充码元的数目。对于本规范的大部分，填充码元被视为附加源码元。

[0621] K′ _max 标示单个源块中可存在的最大源码元数目。设为 56403。

[0622] L 标示关于单个扩展源块的中间码元数目。

[0623] S 标示关于单个扩展源块的 LDPC 码元的数目。这些是 LT 码元。对于第 5.6 节表 2 中所示的每个 K′ 值，S 的相应值为质数。

[0624] H 标示关于单个扩展源块的 HDPC 码元的数目。这些是 PI 码元。

[0625] B 标示除 LDPC 码元以外为 LT 码元的中间码元的数目。

[0626] W 标示为 LT 码元的中间码元的数目。对于第 5.6 节中所示的表 2 中的每个 K′ 值，W 的相应值为质数。

[0627] P 标示为 PI 码元的中间码元的数目。这些包含所有 HDPC 码元。

[0628] P1 标示大于或等于 P 的最小质数。

[0629] U 标示为 PI 码元的非 HDPC 中间码元的数目。

[0630] C 标示中间码元阵列 C[0]、C[1]、C[2]、…、C[L-1]。

[0631] C′ 标示扩展源块的码元阵列，其中 C′ [0]、C′ [1]、C′ [2]、…、C′ [K-1] 是源块的源码元，而 C′ [K]、C′ [K+1]、…、C′ [K′ -1] 是填充码元。

[0632] V0、V1、V2、V3 标示 4 个 32 位无符号整数的阵列，V0[0]、V0[1]、…、V0[255]；V1[0]、V1[1]、…、V1[255]；V2[0]、V2[1]、…、V2[255]；以及 V3[0]、V3[1]、…、V3[255]，如第 5.5 节中所示的。

[0633] Rand[y，i，m] 标示伪随机数生成器

[0634] Deg[v] 标示度生成器

[0635] Enc[K′ ，C，(d，a，b，d1，a1，b1)] 标示编码码元生成器

[0636] Tuple[K′ ，X] 标示元组生成器函数

[0637] T 标示以八位字节计的码元大小。

[0638] J(K′) 标示与 K′ 相关联的系统索引。

[0639] G 标示任何生成器矩阵。

[0640] I_S 标示 SxS 单位矩阵。

[0641] 5.1.3. 缩写

[0642] ESI 编码码元 ID

[0643] HDPC 高密度奇偶校验

[0644] ISI 内部码元 ID

[0645] LDPC 低密度奇偶校验

[0646] LTLuby 变换

[0647] PI 永久钝化

[0648] SBN 源块编号

[0649] SBL 源块长度（以码元计）

[0650] 5.2. 概览

[0651] 本节定义系统 RaptorQ FEC 码。

[0652] 码元是编码和解码过程的基本数据单位。对于每个源块，所有码元有相同大小，称为码元大小 T。针对编码和解码两者对码元执行的原子运算是第 5.7 节中定义的算术运算。

[0653] 基本编码器在第 5.3 节中描述。编码器首先从源块的源码元推导中间码元块。该中间块具有以下属性：可使用相同的过程从中间块生成源码元和修复码元两者。编码器使用高效过程从中间块产生修复码元，其中每个此类修复码元是来自该块的少量中间码元的异或。也可使用相同的过程从中间块再生源码元。编码码元是源码元与修复码元的组合。

[0654] 解码器的示例在第 5.4 节中描述。用于从中间块产生源码元和修复码元的过程被设计成使得能从任何充分大的编码码元集合独立于该集合中源码元与修复码元的混合来恢复中间块。一旦恢复出中间块，就可使用编码过程恢复源块的缺少源码元。

[0655] 对 RaptorQ 兼容解码器的要求在第 5.8 节中提供。数个解码算法有可能达成这些要求。达成这些要求的高效解码算法在第 5.4 节中提供。

[0656] 中间码元和修复码元的构造部分地基于第 5.3 节中描述的伪随机数生成器。该生成器基于有 1024 个随机数的固定集合，其必须为发送方和接收方两者可用。这些数字在第 5.5 节中提供。RaptorQ 的编码和解码运算使用关于八位字节的运算。第 5.7 节描述了如何执行这些运算。

[0657] 最后，从源码元构造中间码元由“系统索引”来管理，对于 6 到 K′ _max ＝ 56403 个源码元之间的具体扩展源块大小，“系统索引”的值在第 5.6 节中提供。因此，RaptorQ 码支持有 1 到 56403 之间个源码元的源块。

[0658] 5.3. 系统 RaptorQ 编码器

[0659] 5.3.1. 介绍

[0660] 对于有 K 个源码元的给定源块，出于编码和解码目的，源块被扩增 K′ -K 个附加的填充码元，其中 K′ 是至少为第 5.6 节的系统索引表 2 中的 K 的最小值。用于将源块拉长到 K′ 的倍数的原因是为了实现更快的编码和解码，以及使需要存储在编码器和解码器中的表信息量最少。

[0661] 出于传送和接收数据的目的，K 的值被用来确定源块中的码元数目，因此 K 需要在发送方和接收方处已知。在这种情形中，发送方和接收方可从 K 计算 K′ 且这 K′ -K 个填充码元可在不需要额外通信的情况下被自动添加到源块。发送方和接收方使用编码码元 ID(ESI) 来标识源块的编码码元，其中源块的编码码元由源码元和与源块相关联的修复码元构成。对于有 K 个源码元的源块，源码元的 ESI 为 0、1、2、…、K-1 而修复码元的 ESI 为 K、K+1、K+2、…。使用 ESI 在传输中标识编码码元确保了 ESI 值在源码元和修复码元之间连贯地延续。

[0662] 出于编码和解码数据的目的，从 K 推导出的 K′ 的值被用作对其执行编码和解码操作的扩展源块的源码元的数目，其中 K′ 个源码元由原始的 K 个源码元和附加的 K′ -K 填充码元构成。内部码元 ID(ISI) 被编码器和解码器用来标识与扩展源块相关联的码元，即用于生成编码码元以及用于解码。对于有 K 个原始源码元的源块，原始源码元的 ISI 为 0、1、2、…、K-1，K′ -K 个填充码元的 ISI 为 K、K+1、K+2、…、K′ -1，以及修复码元的 ISI 为 K′ 、K′ +1、K′ +2、…。使用 ISI 来进行编码和解码允许扩展源块的填充码元以与扩展源块的其他源码元相同的方式被对待，且修复码元的给定前缀是按照与扩展源块中独立于 K 的给定数目 K′ 个源码元一致的方式生成的。

[0663] ESI 和 ISI 之间的关系是简单的：原始 K 个源码元的 ESI 和 ISI 是相同的，K′ -K 个填充码元具有 ISI 但不具有相应的 ESI（由于它们是既不被发送也不被接收的码元），以及修复码元 ISI 简单地为修复码元 ESI 加上 K′ -K。用于标识所发送和接收的编码码元的 ESI 与用于编码和解码的相应 ISI 之间的转换、以及以用于编码和解码的填充码元恰适地填充扩展源块是 RaptorQ 编码器 / 解码器中的填充函数的责任。

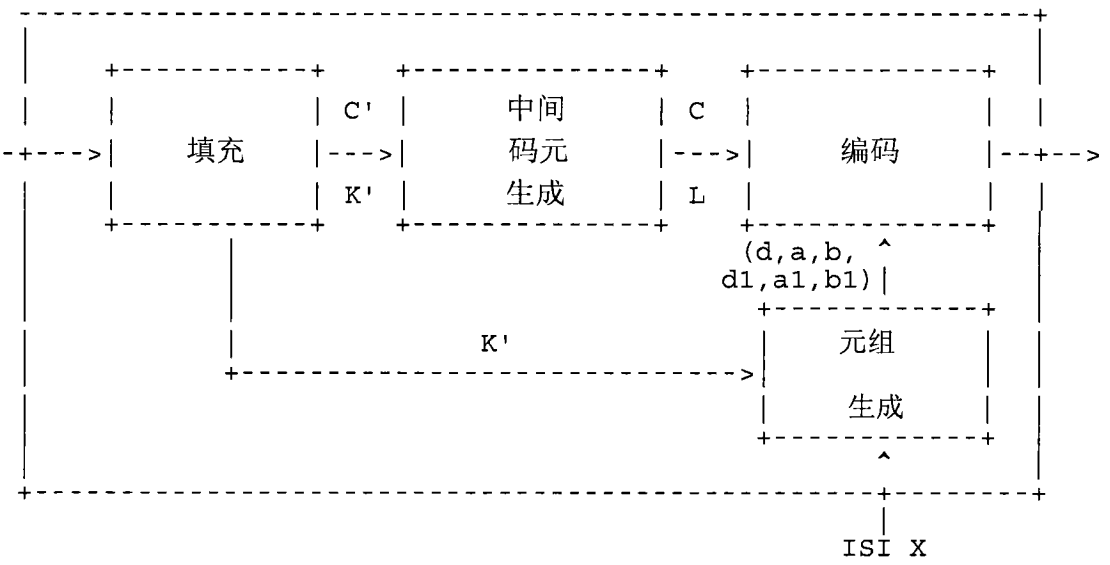
[0664] 5.3.2. 编码概览

[0665] 使用系统 RaptorQ 编码器从由放入扩展源块 C′ 的 K 个源码元构成的源块生成任何数目个修复码元。图 4 示出编码概览。

[0666] 编码的第一步骤是通过添加零个或更多个填充码元来构造扩展源块，以使得码元

总数K′是第5.6节中列出的值之一。每个填充码元由T个八位字节构成，其中每个八位字节的值为0。K′必须被选为来自第5.6节的表的大于或等于K的最小K′值。

[0667]



[0668] 图4：编码概览

[0669] 令C′[0]、…、C′[K-1]标示K个源码元。

[0670] 令C′[K]、…、C′[K′-1]标示K′-K个填充码元，其全部被设为0比特。那么，C′[0]、…、C′[K-1]是对其执行编码和解码的扩展源块。

[0671] 在本描述的其余部分中，这些填充码元将被视为附加源码元且如此引述。然而，这些填充码元不是编码码元的一部分，即它们不作为编码的一部分被发送。在接收机处，K′的值可基于K计算出来，随后接收机可在有K′个源码元的源块末尾插入K′-K个填充码元并从收到编码码元恢复源块的其余K个源码元。

[0672] 编码的第二步骤是从K′个源码元生成数目L＞K′个中间码元。在该步骤中，使用如第5.3.5.4节中描述的Tuple[]生成器生成K′个源元组(d[0]，a[0]，b[0]，d1[0]，a1[0]，b1[0])，…，(d[K′-1]，a[K′-1]，b[K′-1]，d1[K′-1]，a1[K′-1]，b1[K′-1])。这K′个源元组和与这K′个源码元相关联的ISI被用于使用逆编码过程从源码元确定L个中间码元C[0]、…、C[L-1]。该过程可由RaptorQ解码过程实现。

[0673] 在这L个中间码元内，某些“预编码关系”必须成立。第5.3.3.3节描述了这些关系。第5.3.3.4节描述了如何从源码元生成中间码元。

[0674] 一旦生成了中间码元，可能产生修复码元。对于ISI为X＞K′的修复码元，可使用第5.3.5.4节中描述的Tuple[]生成器生成非负整数元组(d,a,b,d1,a1,b1)。随后，该(d,a,b,d1,a1,b1)元组和ISI X被用于使用第5.3.5.3节中描述的Enc[]生成器从中间码元生成相应的修复码元。该修复码元的相应ESI则为X-(K′-K)。注意，也可使用相同的过程生成扩展源块的源码元，即对于任何X＜K′，使用该过程生成的码元具有与C′[X]相同的值。

[0675] 5.3.3. 第一编码步骤：中间码元生成

[0676] 5.3.3.1. 综述

[0677] 该编码步骤是从源码元C′[0]、…、C′[K′-1]生成L个中间码元C[0]、…、

C[L-1] 的预编码步骤，其中 L ＞ K′ 在第 5.3.3.3 节中定义。中间码元由两组约束唯一性地定义：

[0678] 1. 中间码元通过源码元元组集合和源码元的 ISI 与源码元相关。源码元元组的生成在第 5.3.3.2 节中使用第 5.3.5.4 节中描述的 Tuple[] 生成器来定义。

[0679] 2. 在源码元自身内，数个预编码关系成立。这些在第 5.3.3.3 节中定义。

[0680] L 个中间码元的生成随后在第 5.3.3.4 节中定义。

[0681] 5.3.3.2. 源码元元组

[0682] K′ 个源码元中的每个源码元与源码元元组 (d[X]，a[X]，b[X]，d1[X]，a1[X]，b1[X]) 相关联，其中 0 ＜＝ X ＜ K′ 。源码元元组使用第 5.3.5.4 节中定义的元组生成器被确定为：

[0683] 对于每个 X，0 ＜＝ X ＜ K′

[0684] (d[X]，a[X]，b[X]，d1[X]，a1[X]，b1[X]) ＝ Tuple[K，X]

[0685] 5.3.3.3. 预编码关系

[0686] L 个中间码元之间的预编码关系通过要求这些中间码元的 S+H 个线性组合的集合估值为 0 来定义。有 S 个 LDPC 码元和 H 个 HDPC 码元，因此 L ＝ K′ +S+H。L 个中间码元的另一种划分是划分成两个集合，一个集合有 W 个 LT 码元而另一个集合有 P 个 PI 码元，因此也是 L ＝ W+P 的情形。在编码过程中，P 个 PI 码元与 W 个 LT 码元不同地对待。P 个 PI 码元由 H 个 HDPC 码元连同其他 K′ 个中间码元中的 U ＝ P-H 个的集合构成。W 个 LT 码元由 S 个 LDPC 码元连同其他 K′ 个中间码元中的 W-S 个构成。这些参数的值是如下描述地从 K′ 确定的，其中 H(K′)、S(K′) 和 W(K′) 是从第 5.6 节表 2 推导出来的。

[0687] 令：

[0688] ○ S ＝ S(K′)

[0689] ○ H ＝ H(K′)

[0690] ○ W ＝ W(K′)

[0691] ○ L ＝ K′ +S+H

[0692] ○ P ＝ L-W

[0693] ○ P1 标示大于或等于 P 的最小质数

[0694] ○ U ＝ P-H

[0695] ○ B ＝ W-S

[0696] ○ C[0]、…、C[B-1] 标示为 LT 码元但非 LDPC 码元的中间码元。

[0697] ○ C[B]、…、C[B+S-1] 标示也为 LT 码元的 S 个 LDPC 码元。

[0698] ○ C[W]、…、C[W+U-1] 标示为 PI 码元但非 HDPC 码元的中间码元。

[0699] ○ C[L-H]、…、C[L-1] 标示也为 PI 码元的 H 个 HDPC 码元。

[0700] 称为 LDPC 关系的第一组预编码关系在以下描述且要求在该过程结束时码元集合 D[0]、…、D[S-1] 全为 0：

[0701] ○初始化码元 D[0] ＝ C[B]、…、D[S-1] ＝ C[B+S-1]。

[0702] ○ For i ＝ 0，…，B-1 do

[0703] *a ＝ 1+floor(i/S)

[0704] *b ＝ i％ S

[0705] *D[b] = D[b]+C[i]

[0706] *b = (b+a)% S

[0707] *D[b] = D[b]+C[i]

[0708] *b = (b+a)% S

[0709] *D[b] = D[b]+C[i]

[0710] ○ For i = 0, …, S-1 do

[0711] *a = i% P

[0712] *b = (i+1)% P

[0713] *D[i] = D[i]+C[W+a]+C[W+b]

[0714] 回想码元添加将如第 5.7 节中指定地执行。

[0715] 注意,以上算法中定义的 LDPC 关系是线性的,因此存在 S x B 矩阵 G_LDPC 1 和 S x P 矩阵 G_LDPC 2,以使得

[0716] G_LDPC,1*Transpose[(C[0], …, C[B-1])]+G_LDPC,2*Transpose(C[W], …, C[W+P-1])+Transpose[(C[B], …, C[B+S-1])] = 0

[0717] (矩阵 G_LDPC 1 是在上面的算法的第一循环中定义的,以及 G_LDPC 2 可从第二循环推出。)

[0718] 中间码元 C[0]、…、C[L-1] 之间的第二组关系是 HDPC 关系且它们定义如下 :

[0719] 令 :

[0720] ○ α 标示由整数 2 表示的元组,如第 5.7 节中定义的。

[0721] ○ MT 标示八位字节的 H x(K′ +S) 矩阵,其中对于户 0, …, K′ +S-2,若 i = Rand[j+1,6,H] 或 i = (Rand[j+1,6,H]+Rand[j+1,7,H-1]+1)% H,则 MT[i,j] 的条目是由整数 1 表示的八位字节,且对于 i 的所有其他值,MT[i,j] 为 0 元素,以及对于 j = K′ +S-1, MT[i,j] = a^^i,其中 i = 0, …, H-1。

[0722] ○ GAMMA 标识八位字节的 (K′ +S)x(K′ +S) 矩阵,其中

[0723] GAMMA[i,j] =

[0724] a^^(i-j),对于 i >= j,

[0725] 0,否则。

[0726] 那么,头 K′ +S 个中间码元 C[0]、…、C[K′ +S-1] 与 H 个 HDPC 码元 C[K′ +S]、…、C[K′ +S+H-1] 之间的关系由下式给出 :

[0727] Transpose[C[K′ +S], …, C[K′ +S+H-1]]+MT*GAMMA*Transpose[C[0], …, C[K′ +S-1]] = 0,

[0728] 其中′ *′ 表示利用八位字节乘法来定义八位字节矩阵与码元矩阵 (具体而言,码元的列向量) 之间的乘法的标准矩阵乘法,以及′ +′ 标识八位字节向量上的加法。

[0729] 5.3.3.4. 中间码元

[0730] 5.3.3.4.1. 定义

[0731] 给定 K′ 个源码元 C′ [0]、C′ [1]、…、C′ [K′ -1],L 个中间码元 C[0]、C[1]、…、C[L-1] 是满足以下条件的唯一性地定义的码元值 :

[0732] 1.K′ 个源码元 C′ [0]、C′ [1]、…、C′ [K′ -1] 满足 K′ 约束

[0733] C′ [X] = Enc[K′ , (C[0], …, C[L-1]), (d[X], a[X], b[X], d1[X], a1[X],

b1[X])],对于所有 X,0 ＜＝ X ＜ K′ ,

[0734] 其中 (d[X],a[X],b[X],d1[X],a1[X],b1[X])) ＝ Tuple[K′ ,X],Tuple[] 在第 5.3.5.4 节中定义,以及 Enc[] 在第 5.3.5.3 节中描述。

[0735] 2.L 个中间码元 C[0]、C[1]、…、C[L-1] 满足第 5.3.3.3 节中定义的预编码关系。

[0736] 5.3.3.4.2. 演算中间码元的示例方法

[0737] 本节描述用于演算满足第 5.3.3.4.1 节中的约束的 L 个中间码元 C[0]、C[1]、…、C[L-1] 的可能方法。

[0738] L 个中间码元可如下计算 :

[0739] 令 :

[0740] ○ C 标示 L 个中间码元 C[0]、C[1]、…、C[L-1] 的列向量。

[0741] ○ D 标示由 S+H 个零码元继以 K′ 个源码元 C′ [0]、C′ [1]、…、C′ [K′ -1] 构成的列向量。

[0742] 那么以上约束定义八位字节的 L x L 矩阵 A 以使得 :

[0743] A*C ＝ D

[0744] 矩阵 A 可如下构造 :

[0745] 令 :

[0746] ○ G_LDPC 1 和 G_LDPC 2 是如第 5.3.3.3 节中定义的 S x B 矩阵和 S x P 矩阵。

[0747] ○ G_HDPC 是 H x(K′ +S) 矩阵,以使得

[0748] G_HDPC*Transpose(C[0],…,C[K′ +S-1]) ＝ Transpose(C[K′ +S],…,C[L-1]),即 G_HDPC ＝ MT*GAMMA

[0749] ○ I_S 是 S x S 单位矩阵

[0750] ○ I_H 是 H x H 单位矩阵

[0751] ○ G_ENC 是 K′ x L 矩阵,以使得

[0752] G_ENC*Transpose[(C[0], …, C[L-1])] ＝ Transpose[(C′ [0],C′ [1], …, C′ [K′ -1])],

[0753] 即,当且仅当 C[j] 被包括在被求和以产生 Enc[K′ ,(C[0], …,C[L-1]),(d[i], a[i],b[i],d1[i],a1[i],b1[i])] 的码元中时,G_ENC[i,j] ＝ 1,否则 G_ENC[i,j] ＝ 0。

[0754] 那么 :

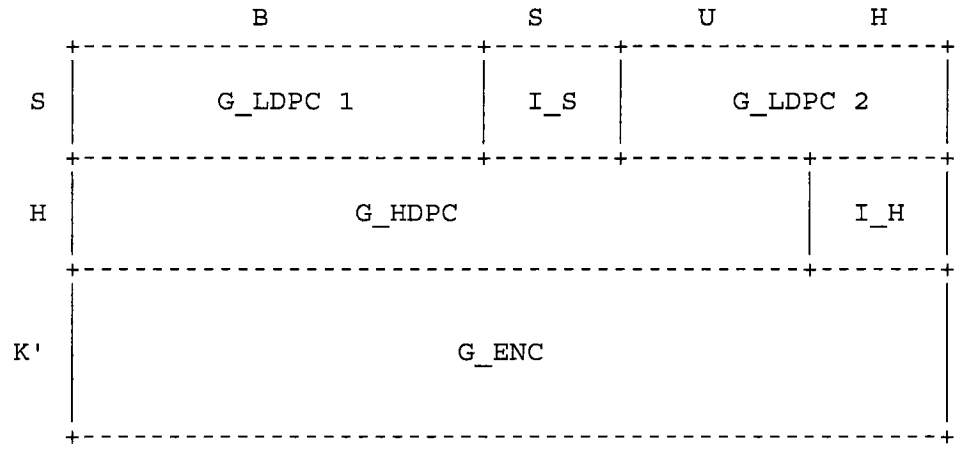
[0755] ○ A 的头 S 行等于 G_LDPC 1|I_S|G_LDPC 2。

[0756] ○ A 的接下来 H 行等于 G_HDPC|I_H。

[0757] ○ A 的其余 K′ 行等于 G_ENC。

[0758] 矩阵 A 在下图 (图 5) 中描绘 :

[0759]



[0760] 图 5 :矩阵 A

[0761] 中间码元随后可计算为 :

[0762] C = (A^^-1)*D

[0763] 生成源元组以使得对于任何 K′ ,矩阵 A 具有满秩且因此是可逆的。该演算可通过向 K′ 个源码元 C′ [0]、C′ [1]、…、C′ [K′ -1] 应用 RaptorQ 解码过程以产生 L 个中间码元 C[0]、C[1]、…、C[L-1] 来实现。

[0764] 为了高效地从源码元生成中间码元,推荐使用诸如第 5.4 节中描述的高效解码器实现。

[0765] 5.3.4. 第二编码步骤 :编码

[0766] 在第二编码步骤中,通过使用元组 (d, a, b, d1, a1, b1) = Tuple[K′ , X] 向 L 个中间码元 C[0], C[1], …, C[L-1] 应用第 5.3.5.3 节中定义的生成器 Enc[K′ , (C[0], C[1], …, C[L-1]), (d, a, b, d1, a1, b1)] 来生成 ISI 为 X(X > = K′) 的修复码元。

[0767] 5.3.5. 生成器

[0768] 5.3.5.1. 随机数生成器

[0769] 随机数生成器 Rand[y,i,m] 定义如下,其中 y 是非负整数,i 是小于 256 的非负整数,以及 m 是正整数,且产生的值是 0 到 m-1 之间的整数。令 V0、V1、V2 和 V3 为第 5.5 节中提供的阵列。

[0770] 令 :

[0771] ○ x0 = (y+i)mod 2^^8

[0772] ○ x1 = (floor(y/2^^8)+i)mod 2^^8

[0773] ○ x2 = (floor(y/2^^16)+i)mod 2^^8

[0774] ○ x3 = (floor(y/2^^24)+i)mod 2^^8

[0775] 那么

[0776] Rand[y, i, m] = (V0[x0]^V1[x1]^V2[x2]^V3[x3]) % m

[0777] 5.3.5.2. 度生成器

[0778] 度生成器 Deg[v] 定义如下,其中 v 是小于 2^^20 = 1048576 的非负整数。给定 v,找出表 1 中使得 f[d-1] <= v < f[d] 的索引 d,且设置 Deg[v] = min(d,W-2)。回想 W 是如第 5.3.3.3 节中描述地从 K′ 推导出的。

[0779]

索引 d	f[d]	索引 d	f[d]
0	0	1	5243
2	529531	3	704294
4	791675	5	844104
6	879057	7	904023
8	922747	9	937311
10	948962	11	958494
12	966438	13	973160
14	978921	15	983914
16	988283	17	992138
18	995565	19	998631
20	1001391	21	1003887
22	1006157	23	1008229
24	1010129	25	1011876
26	1013490	27	1014983
28	1016370	29	1017662
30	1048576		

[0780] 表 1 :定义用于编码码元的度分布

[0781] 5.3.5.3. 编码码元生成器

[0782] 编码码元生成器 Enc[K′ ,(C[0],C[1], …,C[L-1]),(d,a,b,d1,a1,b1)] 取以下输入 :

[0783] ○ K′ 是扩展源块的源码元数目。令 L、W、B、S、P 和 P1 是如第 5.3.3.3 节中描述地从 K′ 推导出的。

[0784] ○ (C[0],C[1],…,C[L-1]) 是如第 5.3.3.4 节中描述地生成的 L 个中间码元 (子码元) 的阵列。

[0785] ○ (d, a, b, d1, a1, b1) 是使用第 5.3.5.4 节中定义的元组生成器从 ISI X 确定的,从而

[0786] *d 是标示编码码元 LT 度的正整数

[0787] *a 是 1 到 W-1 之间 (含 1 和 W-1) 的正整数

[0788] *b 是 0 到 W-1 之间 (含 1 和 W-1) 的非负整数

[0789] *d1 是标示编码码元 PI 度的具有值 2 或 3 的正整数

[0790] *a1 是 1 到 P1-1 之间 (含 1 和 P1-1) 的正整数

[0791] *b1 是 0 到 P1-1 之间 (含 1 和 P1-1) 的非负整数

[0792] 编码码元生成器根据以下算法产生单个编码码元作为输出 (称为结果 (result)) :

[0793] ○ result = C[b]

[0794] ○ For j = 1, …, d-1 do

[0795] *b = (b+a) % W

[0796] *result = result+C[b]

[0797] ○ While(b1 >= P)do b1 = (b1+a1) % P1

[0798] ○ result = result+C[W+b1]

[0799] ○ For j = 1, …, d1-1 do

[0800] *b1 = (b1+a1) % P1

[0801] *While(b1 >= P)do b1 = (b1+a1) % P1

[0802] *result = result+C[W+b1]

[0803] ○ Return result

[0804] 5.3.5.4. 元组生成器

[0805] 元组生成器 Tuple[K′, X] 取以下输入：

[0806] ○ K′ -扩展源块中的源码元数目

[0807] ○ X-ISI

[0808] 令：

[0809] ○ L 是如第 5.3.3.3 节中描述地从 K′ 确定的

[0810] ○ J = J(K′) 是与 K′ 相关联的系统索引，如第 5.6 节表 2 中定义的元组生成器的输出是元组 (d, a, b, d1, a1, b1)，定义如下：

[0811] ○ A = 53591+J*997

[0812] ○ if(A% 2 == 0){A = A+1}

[0813] ○ B = 10267*(J+1)

[0814] ○ y = (B+X*A) % 2^^32

[0815] ○ v = Rand[y, 0, 2^^20]

[0816] ○ d = Deg[v]

[0817] ○ a = 1+Rand[y, 1, W-1]

[0818] ○ b = Rand[y, 2, W]

[0819] ○ If(d < 4){d1 = 2+Rand[X, 3, 2]}else{d1 = 2}

[0820] ○ a1 = 1+Rand[X, 4, P1-1]

[0821] ○ b1 = Rand[X, 5, P1]

[0822] 5.4. 示例 FEC 解码器

[0823] 5.4.1. 综述

[0824] 本节描述用于本规范中介绍的 RaptorQ 码的高效解码算法。注意，每个收到编码码元是中间码元的已知线性组合。因此，每个收到编码码元提供中间码元之间的线性方程，其连同中间码元之间的已知线性预编码关系给出了线性方程组。因此，用于求解线性方程组的任何算法能成功解码出中间码元并因此解码出源码元。然而，所选取的算法对解码的计算效率有很大影响。

[0825] 5.4.2. 解码扩展源块

[0826] 5.4.2.1. 综述

[0827] 假定解码器知道它要解码的源块的结构，包括码元大小 T、源块中的码元数目 K 以及扩展源块中的源码元数目 K′ 。

[0828] 根据第 5.3 节中描述的算法，RaptorQ 解码器可演算总共数目 L ＝ K′ +S+H 个中间码元并确定它们是如何从要解码的扩展源块生成的。在本描述中，假定要解码的扩展源块的收到编码码元被传递给解码器。此外，对于每个此类编码码元，假定其总和等于编码码元的中间码元的数目和集合被传递给解码器。在包括填充码元的源码元的情形中，第 5.3.3.2 节中描述的源码元元组指示其总和给定每个源码元的中间码元的数目和集合。

[0829] 令 N ＞＝ K′ 为要用于解码的收到编码码元（包括扩展源块的填充码元）的数目，且令 M ＝ S+H+N。那么通过第 5.3.3.4.2 节中的符号，得到 A*C ＝ D。

[0830] 解码扩展源块等效于从已知的 A 和 D 解码 C。显然，当且仅当 A 的秩为 L 时，才能解码 C。一旦解码出 C，就可通过使用源码元元组确定必须求和以获得每个缺失源码元的中间码元的数目和集合来获得缺失源码元。

[0831] 解码 C 的第一步骤是形成解码调度。在该步骤中，使用高斯消元法（使用行操作以及行和列重新排序）且在丢弃 M-L 行之后将 A 转换成 L x L 单位矩阵。解码调度包括高斯消元过程期间的行操作以及行和列重新排序的顺序，且仅取决于 A 而不取决于 D。从 D 解码 C 可与形成解码调度并发地发生，或者解码可在后来基于解码调度发生。

[0832] 解码调度与解码 C 之间的对应关系如下。初始令 c[0] ＝ 0、c[1] ＝ 1、…、c[L-1] ＝ L-1 以及 d[0] ＝ 0、d[1] ＝ 1、…、d[M-1] ＝ M-1。

[0833] ○每次在解码调度中向行 i′ 添加 A 的行 i 的倍数 β 时，则在解码过程中，向码元 D[d[i′]] 添加码元 β*D[d[i]]。

[0834] ○每次 A 的行 i 乘以八位字节 β 时，则在解码过程中，码元 D[d[i]] 也乘以 β 。

[0835] ○每次在解码调度中将行 i 与行 i′ 交换时，则在解码过程中，将 d[i] 的值与 d[i′] 的值交换。

[0836] ○每次在解码调度中将列 j 与列 j′ 交换时，则在解码过程中，将 c[j] 的值与 c[j′] 的值交换。

[0837] 从该对应关系清楚的是，在解码扩展源块时对码元的总运算次数是高斯消元中的行操作（非交换）次数。由于 A 在高斯消元后且在丢弃末 M-L 行之后为 L x L 单位矩阵，显然，在成功解码结束时，L 个码元 D[d[0]]、D[d[1]]、…、D[d[L-1]] 是 L 个码元 C[c[0]]、C[c[1]]、…、C[c[L-1]] 的值。

[0838] 执行高斯消元以形成解码调度的次序与解码是否成功无关。然而，解码速度严重取决于执行高斯消元的次序。（此外，维持 A 的稀疏表示是重要的，尽管此处未描述）。本节的其余部分描述可执行高斯消元的相对高效的次序。

[0839] 5.4.2.2. 第一阶段

[0840] 在高斯消元的第一阶段，矩阵 A 在概念上被划分成子矩阵，且附加地创建矩阵 X。该矩阵具有与 A 同样多的行和列，且其贯穿第一阶段将为下三角矩阵。在该阶段开头，矩阵 A 被复制到矩阵 X 中。

[0841] 子矩阵大小被参数化为非负整数 i 和 u，它们分别被初始化为 0 和 PI 码元时间 P。A 的子矩阵为 ：

[0842] 1. 由头 i 行与头 i 列的交集定义的子矩阵 I。这在该阶段每一步骤结束时为单位

矩阵。

[0843] 2. 由头 i 行与除第 i 列和末 u 列以外的所有列的交集定义的子矩阵。该子矩阵的所有条目皆为 0。

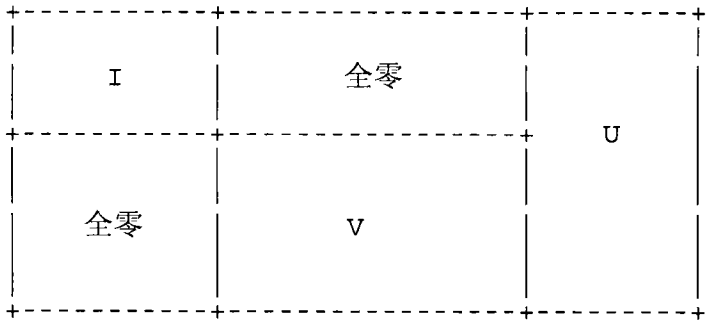
[0844] 3. 由头 i 列与除第 i 行以外的所有行的交集定义的子矩阵。该子矩阵的所有条目皆为 0。

[0845] 4. 由所有行与末 u 列的交集定义的子矩阵。

[0846] 5. 由除头 i 列和末 u 列以外的所有列与除头 i 行以外的所有行的交集定义的子矩阵 V。

[0847] 图 6 解说了 A 的子矩阵。在第一阶段开头，V ＝ A。在每一步骤中，选取 A 的一行。

[0848]



[0849] 图 6 ：第一阶段中 A 的子矩阵

[0850] 由 V 的结构定义的下图被用来确定选取 A 的哪一行。与 V 相交的列是该图中的节点，以及在 V 中具有恰好 2 个非零条目且不是 HDPC 行的那些行是该图中在这两个 1 的位置处连接这两列（节点）的边缘。该图中的分量是节点（列）和边缘（行）的最大集合，以使得该图中每一对节点 / 边缘之间存在路径。分量的大小是该分量中的节点（列）数目。

[0851] 第一阶段中至多有 L 个步骤。当 i+u ＝ L 时，即当 V 和 V 上方的全零子矩阵已消失且 A 由 I、I 下方的全零子矩阵和 U 构成时，该阶段成功结束。若在 V 消失前的某个步骤，在该步骤里 V 中没有非零行供选择，则该阶段以解码失败而不成功地结束。在每一步骤中，如下选取 A 的一行 ：

[0852] ○若 V 的所有条目皆为 0，则不选取行且解码失败。

[0853] ○令 r 为最小整数，使得 A 的至少一行在 V 中具有正好 r 个 1。

[0854] ＊若 r ！＝ 2，则选取在 V 中具有正好 r 个 1 的在所有此类行当中具有最小原始度的行，除了在所有非 HDPC 行已被处理之前不应选取 HDPC 行。

[0855] ＊若 r ＝ 2，则选取在 V 中具有正好 2 个 1 的是上面描述的图中由 V 定义的最大大小分量的一部分的任一行。

[0856] 在该步骤中选取了行之后，将 A 中与 V 相交的第一行与所选行交换，从而所选行是与 V 相交的第一行。A 中与 V 相交的那些列被重新排序，从而所选行中的 r 个 1 之一出现在 V 的第一列，且其余 r-1 个 1 出现在 V 的最后各列。也对矩阵 X 执行相同的行和列操作。随后，所选行的恰适倍数被加到 A 中所选行下方在 V 的第一列中具有非零条目的所有其他行。具体地，若所选行下方的行在 V 的第一列中具有条目 β，且所选行在 V 的第一列中具有条目 α，则 β / α 乘以所选行被加到该行以使 V 的第一列剩下零值。最后，i 递增 1 且 u 递增 r-1，完成该步骤。

[0857] 注意，若以上标识出的行操作在受影响行自身在解码过程期间被选取之前实际没有执行，则效率可得以改善。这避免了在解码过程中最终没有使用的行的行操作的处理，且具体而言在实际需要之前避免了 β ！ ＝ 1 的那些行。此外，HDPC 行需要的行操作可通过使用第 5.3.3.3 节中描述的算法在一个过程中对所有此类行执行。

[0858] 5.4.2.3. 第二阶段

[0859] 此时，X 中在头 i 行和 i 列外的所有条目都被丢弃，因此 X 具有下三角形式。X 的末 i 行和 i 列被丢弃，因此 X 现在具有 i 行 i 列。子矩阵 U 被进一步划分成头 i 行即 U_ 上、以及其余 M-i 行即 U_ 下。在第二阶段中对 U_ 下执行高斯消元以确定其秩小于 u（解码失败）或将它转换成其中头 u 行是单位矩阵的矩阵（第二阶段成功）。将该 u x u 单位矩阵称为 I_u。A 中与 U_ 下 -I_u 相交的 M-L 行被丢弃。该阶段之后，A 具有 L 行和 L 列。

[0860] 5.4.2.4. 第三阶段

[0861] 在第二阶段之后，A 中需要消零以完成将 A 转换成 L x L 单位矩阵的那部分是 U_ 上。子矩阵 U_ 上的行数 i 一般远大于 U_ 上的列数 u。此外，这时矩阵 U_ 上一般是密集的，即该矩阵的非零条目数量很大。为了将该矩阵简化为稀疏形式，需要逆反获得矩阵 U_ 下所执行的操作序列。为此，将矩阵 X 乘以由 A 的头 i 行构成的 A 的子矩阵。在该操作之后，由头 i 行与 i 列的交集构成的 A 的子矩阵等于 X，其中矩阵 U_ 上被变换成稀疏形式。

[0862] 5.4.2.5. 第四阶段

[0863] 对于 U_ 上的头 i 行中的每一行，进行以下操作 ：若该行在位置 j 具有非零条目且若该非零条目的值为 b，则向该行加 b 乘以 I_u 的行 j。在该步骤之后，由头 i 行与 i 列的交集构成的 A 的子矩阵等于 X，子矩阵 U_ 上由 0 构成，由末 u 行与头 i 列的交集构成的子矩阵由 0 构成，以及由末 u 行与 u 列构成的子矩阵为矩阵 I_u。

[0864] 5.4.2.6. 第五阶段

[0865] 对于 j 从 1 到 i，执行以下操作 ：

[0866] 1. 若 A[j, j] 不是 1，则将 A 的行 j 除以 A[j, j]。

[0867] 2. 对于 1 从 1 到 j-1，若 A[j, 1] 是非零，则将 A[j, 1] 乘以 A 的行 1 加到 A 的行 j。

[0868] 在该阶段之后，A 为 L x L 单位矩阵，且完整的解码调度已成功形成。随后，可执行包括求和已知编码码元的相应解码以基于解码调度来恢复中间码元。根据第 5.3.3.2 节计算与所有源码元相关联的元组。关于收到源码元的元组被用于解码。关于缺失源码元的元组被用于确定需要将哪些中间码元求和以恢复缺失的源码元。

[0869] 5.5. 随机数

[0870] 第 5.3.5.1 节中使用的 4 个阵列 V0、V1、V2 和 V3 在以下提供。这 4 个阵列中的每个阵列中有 256 个条目。进入每个阵列的索引始于 0，且这些条目是 32 位的无符号整数。

[0871] 5.5.1. 表 V0

[0872] 251291136,3952231631,3370958628,4070167936,123631495,3351110283,

[0873] 3218676425,2011642291,774603218,2402805061,1004366930,

[0874] 1843948209,428891132,3746331984,1591258008,3067016507,

[0875] 1433388735,504005498,2032657933,3419319784,2805686246,

[0876] 3102436986,3808671154,2501582075,3978944421,246043949,

[0877] 4016898363,649743608,1974987508,2651273766,2357956801,689605112,

[0878] 715807172,2722736134,191939188,3535520147,3277019569,1470435941,

[0879] 3763101702,3232409631,122701163,3920852693,782246947,372121310,

[0880] 2995604341,2045698575,2332962102,4005368743,218596347,

[0881] 3415381967,4207612806,861117671,3676575285,2581671944,

[0882] 3312220480,681232419,307306866,4112503940,1158111502,709227802,

[0883] 2724140433,4201101115,4215970289,4048876515,3031661061,

[0884] 1909085522,510985033,1361682810,129243379,3142379587,2569842483,

[0885] 3033268270,1658118006,932109358,1982290045,2983082771,

[0886] 3007670818,3448104768,683749698,778296777,1399125101,1939403708,

[0887] 1692176003,3868299200,1422476658,593093658,1878973865,

[0888] 2526292949,1591602827,3986158854,3964389521,2695031039,

[0889] 1942050155,424618399,1347204291,2669179716,2434425874,

[0890] 2540801947,1384069776,4123580443,1523670218,2708475297,

[0891] 1046771089,2229796016,1255426612,4213663089,1521339547,

[0892] 3041843489,420130494,10677091,515623176,3457502702,2115821274,

[0893] 2720124766,3242576090,854310108,425973987,325832382,1796851292,

[0894] 2462744411,1976681690,1408671665,1228817808,3917210003,

[0895] 263976645,2593736473,2471651269,4291353919,650792940,1191583883,

[0896] 3046561335,2466530435,2545983082,969168436,2019348792,

[0897] 2268075521,1169345068,3250240009,3963499681,2560755113,

[0898] 911182396,760842409,3569308693,2687243553,381854665,2613828404,

[0899] 2761078866,1456668111,883760091,3294951678,1604598575,

[0900] 1985308198,1014570543,2724959607,3062518035,3115293053,

[0901] 138853680,4160398285,3322241130,2068983570,2247491078,

[0902] 3669524410,1575146607,828029864,3732001371,3422026452,

[0903] 3370954177,4006626915,543812220,1243116171,3928372514,

[0904] 2791443445,4081325272,2280435605,885616073,616452097,3188863436,

[0905] 2780382310,2340014831,1208439576,258356309,3837963200,

[0906] 2075009450,3214181212,3303882142,880813252,1355575717,207231484,

[0907] 2420803184,358923368,1617557768,3272161958,1771154147,

[0908] 2842106362,1751209208,1421030790,658316681,194065839,3241510581,

[0909] 38625260,301875395,4176141739,297312930,2137802113,1502984205,

[0910] 3669376622,3728477036,234652930,2213589897,2734638932,

[0911] 1129721478,3187422815,2859178611,3284308411,3819792700,

[0912] 3557526733,451874476,1740576081,3592838701,1709429513,

[0913] 3702918379,3533351328,1641660745,179350258,2380520112,

[0914] 3936163904,3685256204,3156252216,1854258901,2861641019,

[0915] 3176611298,834787554,331353807,517858103,3010168884,4012642001,

[0916] 2217188075,3756943137,3077882590,2054995199,3081443129,

[0917] 3895398812,1141097543,2376261053,2626898255,2554703076,

[0918] 401233789,1460049922,678083952,1064990737,940909784,1673396780,

[0919] 528881783,1712547446,3629685652,1358307511

[0920] 5.5.2. 表 V1

[0921] 807385413,2043073223,3336749796,1302105833,2278607931,541015020,

[0922] 1684564270,372709334,3508252125,1768346005,1270451292,

[0923] 2603029534,2049387273,3891424859,2152948345,4114760273,

[0924] 915180310,3754787998,700503826,2131559305,1308908630,224437350,

[0925] 4065424007,3638665944,1679385496,3431345226,1779595665,

[0926] 3068494238,1424062773,1033448464,4050396853,3302235057,

[0927] 420600373,2868446243,311689386,259047959,4057180909,1575367248,

[0928] 4151214153,110249784,3006865921,4293710613,3501256572,998007483,

[0929] 499288295,1205710710,2997199489,640417429,3044194711,486690751,

[0930] 2686640734,2394526209,2521660077,49993987,3843885867,4201106668,

[0931] 415906198,19296841,2402488407,2137119134,1744097284,579965637,

[0932] 2037662632,852173610,2681403713,1047144830,2982173936,910285038,

[0933] 4187576520,2589870048,989448887,3292758024,506322719,176010738,

[0934] 1865471968,2619324712,564829442,1996870325,339697593,4071072948,

[0935] 3618966336,2111320126,1093955153,957978696,892010560,1854601078,

[0936] 1873407527,2498544695,2694156259,1927339682,1650555729,

[0937] 183933047,3061444337,2067387204,228962564,3904109414,1595995433,

[0938] 1780701372,2463145963,307281463,3237929991,3852995239,

[0939] 2398693510,3754138664,522074127,146352474,4104915256,3029415884,

[0940] 3545667983,332038910,976628269,3123492423,3041418372,2258059298,

[0941] 2139377204,3243642973,3226247917,3674004636,2698992189,

[0942] 3453843574,1963216666,3509855005,2358481858,747331248,

[0943] 1957348676,1097574450,2435697214,3870972145,1888833893,

[0944] 2914085525,4161315584,1273113343,3269644828,3681293816,

[0945] 412536684,1156034077,3823026442,1066971017,3598330293,

[0946] 1979273937,2079029895,1195045909,1071986421,2712821515,

[0947] 3377754595,2184151095,750918864,2585729879,4249895712,

[0948] 1832579367,1192240192,946734366,31230688,3174399083,3549375728,

[0949] 1642430184,1904857554,861877404,3277825584,4267074718,

[0950] 3122860549,666423581,644189126,226475395,307789415,1196105631,

[0951] 3191691839,782852669,1608507813,1847685900,4069766876,

[0952] 3931548641,2526471011,766865139,2115084288,4259411376,

[0953] 3323683436,568512177,3736601419,1800276898,4012458395,1823982,

[0954] 27980198,2023839966,869505096,431161506,1024804023,1853869307,

[0955] 3393537983,1500703614,3019471560,1351086955,3096933631,

[0956] 3034634988,2544598006,1230942551,3362230798,159984793,491590373,

[0957] 3993872886,3681855622,903593547,3535062472,1799803217,772984149,

[0958] 895863112,1899036275,4187322100,101856048,234650315,3183125617,

[0959] 3190039692,525584357,1286834489,455810374,1869181575,922673938,

[0960] 3877430102,3422391938,1414347295,1971054608,3061798054,

[0961] 830555096,2822905141,167033190,1079139428,4210126723,3593797804,

[0962] 429192890,372093950,1779187770,3312189287,204349348,452421568,

[0963] 2800540462,3733109044,1235082423,1765319556,3174729780,

[0964] 3762994475,3171962488,442160826,198349622,45942637,1324086311,

[0965] 2901868599,678860040,3812229107,19936821,1119590141,3640121682,

[0966] 3545931032,2102949142,2828208598,3603378023,4135048896

[0967] 5.5.3. 表 V2

[0968] 1629829892,282540176,2794583710,496504798,2990494426,3070701851,

[0969] 2575963183,4094823972,2775723650,4079480416,176028725,

[0970] 2246241423,3732217647,2196843075,1306949278,4170992780,

[0971] 4039345809,3209664269,3387499533,293063229,3660290503,

[0972] 2648440860,2531406539,3537879412,773374739,4184691853,

[0973] 1804207821,3347126643,3479377103,3970515774,1891731298,

[0974] 2368003842,3537588307,2969158410,4230745262,831906319,

[0975] 2935838131,264029468,120852739,3200326460,355445271,2296305141,

[0976] 1566296040,1760127056,20073893,3427103620,2866979760,2359075957,

[0977] 2025314291,1725696734,3346087406,2690756527,99815156,4248519977,

[0978] 2253762642,3274144518,598024568,3299672435,556579346,4121041856,

[0979] 2896948975,3620123492,918453629,3249461198,2231414958,

[0980] 3803272287,3657597946,2588911389,242262274,1725007475,

[0981] 2026427718,46776484,2873281403,2919275846,3177933051,1918859160,

[0982] 2517854537,1857818511,3234262050,479353687,200201308,2801945841,

[0983] 1621715769,483977159,423502325,3689396064,1850168397,3359959416,

[0984] 3459831930,841488699,3570506095,930267420,1564520841,2505122797,

[0985] 593824107,1116572080,819179184,3139123629,1414339336,1076360795,

[0986] 512403845,177759256,1701060666,2239736419,515179302,2935012727,

[0987] 3821357612,1376520851,2700745271,966853647,1041862223,715860553,

[0988] 171592961,1607044257,1227236688,3647136358,1417559141,

[0989] 4087067551,2241705880,4194136288,1439041934,20464430,119668151,

[0990] 2021257232,2551262694,1381539058,4082839035,498179069,311508499,

[0991] 3580908637,2889149671,142719814,1232184754,3356662582,

[0992] 2973775623,1469897084,1728205304,1415793613,50111003,3133413359,

[0993] 4074115275,2710540611,2700083070,2457757663,2612845330,

[0994] 3775943755,2469309260,2560142753,3020996369,1691667711,

[0995] 4219602776,1687672168,1017921622,2307642321,368711460,

[0996] 3282925988,213208029,4150757489,3443211944,2846101972,

[0997] 4106826684,4272438675,2199416468,3710621281,497564971,285138276,

[0998] 765042313,916220877,3402623607,2768784621,1722849097,3386397442,

[0999] 487920061,3569027007,3424544196,217781973,2356938519,3252429414,

[1000] 145109750,2692588106,2454747135,1299493354,4120241887,

[1001] 2088917094,932304329,1442609203,952586974,3509186750,753369054,

[1002] 854421006,1954046388,2708927882,4047539230,3048925996,

[1003] 1667505809,805166441,1182069088,4265546268,4215029527,

[1004] 3374748959,373532666,2454243090,2371530493,3651087521,

[1005] 2619878153,1651809518,1553646893,1227452842,703887512,

[1006] 3696674163,2552507603,2635912901,895130484,3287782244,

[1007] 3098973502,990078774,3780326506,2290845203,41729428,1949580860,

[1008] 2283959805,1036946170,1694887523,4880696,466000198,2765355283,

[1009] 3318686998,1266458025,3919578154,3545413527,2627009988,

[1010] 3744680394,1696890173,3250684705,4142417708,915739411,

[1011] 3308488877,1289361460,2942552331,1169105979,3342228712,

[1012] 698560958,1356041230,2401944293,107705232,3701895363,903928723,

[1013] 3646581385,844950914,1944371367,3863894844,2946773319,

[1014] 1972431613,1706989237,29917467,3497665928

[1015] 5.5.4. 表 V3

[1016] 1191369816,744902811,2539772235,3213192037,3286061266,

[1017] 1200571165,2463281260,754888894,714651270,1968220972,3628497775,

[1018] 1277626456,1493398934,364289757,2055487592,3913468088,

[1019] 2930259465,902504567,3967050355,2056499403,692132390,186386657,

[1020] 832834706,859795816,1283120926,2253183716,3003475205,1755803552,

[1021] 2239315142,4271056352,2184848469,769228092,1249230754,

[1022] 1193269205,2660094102,642979613,1687087994,2726106182,446402913,

[1023] 4122186606,3771347282,37667136,192775425,3578702187,1952659096,

[1024] 3989584400,3069013882,2900516158,4045316336,3057163251,

[1025] 1702104819,4116613420,3575472384,2674023117,1409126723,

[1026] 3215095429,1430726429,2544497368,1029565676,1855801827,

[1027] 4262184627,1854326881,2906728593,3277836557,2787697002,

[1028] 2787333385,3105430738,2477073192,748038573,1088396515,

[1029] 1611204853,201964005,3745818380,3654683549,3816120877,

[1030] 3915783622,2563198722,1181149055,33158084,3723047845,3790270906,

[1031] 3832415204,2959617497,372900708,1286738499,1932439099,

[1032] 3677748309,2454711182,2757856469,2134027055,2780052465,

[1033] 3190347618,3758510138,3626329451,1120743107,1623585693,

[1034] 1389834102,2719230375,3038609003,462617590,260254189,3706349764,

[1035] 2556762744,2874272296,2502399286,4216263978,2683431180,

[1036] 2168560535,3561507175,668095726,680412330,3726693946,4180630637,

[1037] 3335170953,942140968,2711851085,2059233412,4265696278,

[1038] 3204373534,232855056,881788313,2258252172,2043595984,3758795150,

[1039] 3615341325,2138837681,1351208537,2923692473,3402482785,

[1040] 2105383425,2346772751,499245323,3417846006,2366116814,

[1041] 2543090583,1828551634,3148696244,3853884867,1364737681,

[1042] 2200687771,2689775688,232720625,4071657318,2671968983,

[1043] 3531415031,1212852141,867923311,3740109711,1923146533,

[1044] 3237071777,3100729255,3247856816,906742566,4047640575,

[1045] 4007211572,3495700105,1171285262,2835682655,1634301229,

[1046] 3115169925,2289874706,2252450179,944880097,371933491,1649074501,

[1047] 2208617414,2524305981,2496569844,2667037160,1257550794,

[1048] 3399219045,3194894295,1643249887,342911473,891025733,3146861835,

[1049] 3789181526,938847812,1854580183,2112653794,2960702988,

[1050] 1238603378,2205280635,1666784014,2520274614,3355493726,

[1051] 2310872278,3153920489,2745882591,1200203158,3033612415,

[1052] 2311650167,1048129133,4206710184,4209176741,2640950279,

[1053] 2096382177,4116899089,3631017851,4104488173,1857650503,

[1054] 3801102932,445806934,3055654640,897898279,3234007399,1325494930,

[1055] 2982247189,1619020475,2720040856,885096170,3485255499,

[1056] 2983202469,3891011124,546522756,1524439205,2644317889,

[1057] 2170076800,2969618716,961183518,1081831074,1037015347,

[1058] 3289016286,2331748669,620887395,303042654,3990027945,1562756376,

[1059] 3413341792,2059647769,2823844432,674595301,2457639984,

[1060] 4076754716,2447737904,1583323324,625627134,3076006391,345777990,

[1061] 1684954145,879227329,3436182180,1522273219,3802543817,

[1062] 1456017040,1897819847,2970081129,1382576028,3820044861,

[1063] 1044428167,612252599,3340478395,2150613904,3397625662,

[1064] 3573635640,3432275192

[1065] 5.6. 系统索引和其他参数

[1066] 下表 2 指定了所支持的 K′ 值。该表还指定了关于每个所支持的 K′ 值的系统索引 J(K′)、HDPC 码元数目 H(K′)、LDPC 码元数目 S(K′)、以及 LT 码元数目 W(K′)。对于 K′ 的每个值,S(K′) 和 W(K′) 的相应值为质数。

[1067] 系统索引 J(K′) 被设计成具有以下属性 :源码元元组集合 (d[0], a[0], b[0], d1[0], a1[0], b1[0]), …, (d[K′ -1], a[K′ -1], b[K′ -1], d1[K′ -1], a1[K′ -1], b1[K′ -1]) 使得 L 个中间码元被唯一性地定义,即图 6 中的矩阵 A 具有满秩且因此是可逆

的。

[1068]

K'	J(K')	S(K')	H(K')	W(K')
10	254	7	10	17
12	630	7	10	19
18	682	11	10	29
20	293	11	10	31
26	80	11	10	37
30	566	11	10	41
32	860	11	10	43
36	267	11	10	47
42	822	11	10	53
46	506	13	10	59
48	589	13	10	61
49	87	13	10	61
55	520	13	10	67
60	159	13	10	71
62	235	13	10	73
69	157	13	10	79
75	502	17	10	89
84	334	17	10	97
88	583	17	10	101
91	66	17	10	103
95	352	17	10	107
97	365	17	10	109
101	562	17	10	113
114	5	19	10	127
119	603	19	10	131

[1069]

125	721	19	10	137
127	28	19	10	139
138	660	19	10	149
140	829	19	10	151
149	900	23	10	163
153	930	23	10	167
160	814	23	10	173
166	661	23	10	179
168	693	23	10	181
179	780	23	10	191
181	605	23	10	193
185	551	23	10	197
187	777	23	10	199
200	491	23	10	211
213	396	23	10	223
217	764	29	10	233
225	843	29	10	241
236	646	29	10	251
242	557	29	10	257
248	608	29	10	263
257	265	29	10	271
263	505	29	10	277
269	722	29	10	283
280	263	29	10	293
295	999	29	10	307
301	874	29	10	313
305	160	29	10	317
324	575	31	10	337
337	210	31	10	349
341	513	31	10	353
347	503	31	10	359
355	558	31	10	367

[1070]

362	932	31	10	373
368	404	31	10	379
372	520	37	10	389
380	846	37	10	397
385	485	37	10	401
393	728	37	10	409
405	554	37	10	421
418	471	37	10	433
428	641	37	10	443
434	732	37	10	449
447	193	37	10	461
453	934	37	10	467
466	864	37	10	479
478	790	37	10	491
486	912	37	10	499
491	617	37	10	503
497	587	37	10	509
511	800	37	10	523
526	923	41	10	541
532	998	41	10	547
542	92	41	10	557
549	497	41	10	563
557	559	41	10	571
563	667	41	10	577
573	912	41	10	587
580	262	41	10	593
588	152	41	10	601
594	526	41	10	607
600	268	41	10	613
606	212	41	10	619
619	45	41	10	631
633	898	43	10	647
640	527	43	10	653

[1071]

648	558	43	10	661
666	460	47	10	683
675	5	47	10	691
685	895	47	10	701
693	996	47	10	709
703	282	47	10	719
718	513	47	10	733
728	865	47	10	743
736	870	47	10	751
747	239	47	10	761
759	452	47	10	773
778	862	53	10	797
792	852	53	10	811
802	643	53	10	821
811	543	53	10	829
821	447	53	10	839
835	321	53	10	853
845	287	53	10	863
860	12	53	10	877
870	251	53	10	887
891	30	53	10	907
903	621	53	10	919
913	555	53	10	929
926	127	53	10	941
938	400	53	10	953
950	91	59	10	971
963	916	59	10	983
977	935	59	10	997
989	691	59	10	1009
1002	299	59	10	1021
1020	282	59	10	1039
1032	824	59	10	1051

[1072]

1050	536	59	11	1069
1074	596	59	11	1093
1085	28	59	11	1103
1099	947	59	11	1117
1111	162	59	11	1129
1136	536	59	11	1153
1152	1000	61	11	1171
1169	251	61	11	1187
1183	673	61	11	1201
1205	559	61	11	1223
1220	923	61	11	1237
1236	81	67	11	1259
1255	478	67	11	1277
1269	198	67	11	1291
1285	137	67	11	1307
1306	75	67	11	1327
1347	29	67	11	1367
1361	231	67	11	1381
1389	532	67	11	1409
1404	58	67	11	1423
1420	60	67	11	1439
1436	964	71	11	1459
1461	624	71	11	1483
1477	502	71	11	1499
1502	636	71	11	1523
1522	986	71	11	1543
1539	950	71	11	1559
1561	735	73	11	1583
1579	866	73	11	1601
1600	203	73	11	1621
1616	83	73	11	1637
1649	14	73	11	1669
1673	522	79	11	1699

[1073]

1698	226	79	11	1723
1716	282	79	11	1741
1734	88	79	11	1759
1759	636	79	11	1783
1777	860	79	11	1801
1800	324	79	11	1823
1824	424	79	11	1847
1844	999	79	11	1867
1863	682	83	11	1889
1887	814	83	11	1913
1906	979	83	11	1931
1926	538	83	11	1951
1954	278	83	11	1979
1979	580	83	11	2003
2005	773	83	11	2029
2040	911	89	11	2069
2070	506	89	11	2099
2103	628	89	11	2131
2125	282	89	11	2153
2152	309	89	11	2179
2195	858	89	11	2221
2217	442	89	11	2243
2247	654	89	11	2273
2278	82	97	11	2311
2315	428	97	11	2347
2339	442	97	11	2371
2367	283	97	11	2399
2392	538	97	11	2423
2416	189	97	11	2447
2447	438	97	11	2477
2473	912	97	11	2503
2502	1	97	11	2531

[1074]

2528	167	97	11	2557
2565	272	97	11	2593
2601	209	101	11	2633
2640	927	101	11	2671
2668	386	101	11	2699
2701	653	101	11	2731
2737	669	101	11	2767
2772	431	101	11	2801
2802	793	103	11	2833
2831	588	103	11	2861
2875	777	107	11	2909
2906	939	107	11	2939
2938	864	107	11	2971
2979	627	107	11	3011
3015	265	109	11	3049
3056	976	109	11	3089
3101	988	113	11	3137
3151	507	113	11	3187
3186	640	113	11	3221
3224	15	113	11	3259
3265	667	113	11	3299
3299	24	127	11	3347
3344	877	127	11	3391
3387	240	127	11	3433
3423	720	127	11	3469
3466	93	127	11	3511
3502	919	127	11	3547
3539	635	127	11	3583
3579	174	127	11	3623
3616	647	127	11	3659
3658	820	127	11	3701
3697	56	127	11	3739
3751	485	127	11	3793

[1075]

3792	210	127	11	3833
3840	124	127	11	3881
3883	546	127	11	3923
3924	954	131	11	3967
3970	262	131	11	4013
4015	927	131	11	4057
4069	957	131	11	4111
4112	726	137	11	4159
4165	583	137	11	4211
4207	782	137	11	4253
4252	37	137	11	4297
4318	758	137	11	4363
4365	777	137	11	4409
4418	104	139	11	4463
4468	476	139	11	4513
4513	113	149	11	4567
4567	313	149	11	4621
4626	102	149	11	4679
4681	501	149	11	4733
4731	332	149	11	4783
4780	786	149	11	4831
4838	99	149	11	4889
4901	658	149	11	4951
4954	794	149	11	5003
5008	37	151	11	5059
5063	471	151	11	5113
5116	94	157	11	5171
5172	873	157	11	5227
5225	918	157	11	5279
5279	945	157	11	5333
5334	211	157	11	5387
5391	341	157	11	5443

[1076]

5449	11	163	11	5507
5506	578	163	11	5563
5566	494	163	11	5623
5637	694	163	11	5693
5694	252	163	11	5749
5763	451	167	11	5821
5823	83	167	11	5881
5896	689	167	11	5953
5975	488	173	11	6037
6039	214	173	11	6101
6102	17	173	11	6163
6169	469	173	11	6229
6233	263	179	11	6299
6296	309	179	11	6361
6363	984	179	11	6427
6427	123	179	11	6491
6518	360	179	11	6581
6589	863	181	11	6653
6655	122	181	11	6719
6730	522	191	11	6803
6799	539	191	11	6871
6878	181	191	11	6949
6956	64	191	11	7027
7033	387	191	11	7103
7108	967	191	11	7177
7185	843	191	11	7253
7281	999	193	11	7351
7360	76	197	11	7433
7445	142	197	11	7517
7520	599	197	11	7591
7596	576	199	11	7669
7675	176	211	11	7759
7770	392	211	11	7853

[1077]

7855	332	211	11	7937
7935	291	211	11	8017
8030	913	211	11	8111
8111	608	211	11	8191
8194	212	211	11	8273
8290	696	211	11	8369
8377	931	223	11	8467
8474	326	223	11	8563
8559	228	223	11	8647
8654	706	223	11	8741
8744	144	223	11	8831
8837	83	223	11	8923
8928	743	223	11	9013
9019	187	223	11	9103
9111	654	227	11	9199
9206	359	227	11	9293
9303	493	229	11	9391
9400	369	233	11	9491
9497	981	233	11	9587
9601	276	239	11	9697
9708	647	239	11	9803
9813	389	239	11	9907
9916	80	239	11	10009
10017	396	241	11	10111
10120	580	251	11	10223
10241	873	251	11	10343
10351	15	251	11	10453
10458	976	251	11	10559
10567	584	251	11	10667
10676	267	257	11	10781
10787	876	257	11	10891
10899	642	257	12	11003

[1078]

11015	794	257	12	11119
11130	78	263	12	11239
11245	736	263	12	11353
11358	882	269	12	11471
11475	251	269	12	11587
11590	434	269	12	11701
11711	204	269	12	11821
11829	256	271	12	11941
11956	106	277	12	12073
12087	375	277	12	12203
12208	148	277	12	12323
12333	496	281	12	12451
12460	88	281	12	12577
12593	826	293	12	12721
12726	71	293	12	12853
12857	925	293	12	12983
13002	760	293	12	13127
13143	130	293	12	13267
13284	641	307	12	13421
13417	400	307	12	13553
13558	480	307	12	13693
13695	76	307	12	13829
13833	665	307	12	13967
13974	910	307	12	14107
14115	467	311	12	14251
14272	964	311	12	14407
14415	625	313	12	14551
14560	362	317	12	14699
14713	759	317	12	14851
14862	728	331	12	15013
15011	343	331	12	15161
15170	113	331	12	15319
15325	137	331	12	15473

[1079]

15496	308	331	12	15643
15651	800	337	12	15803
15808	177	337	12	15959
15977	961	337	12	16127
16161	958	347	12	16319
16336	72	347	12	16493
16505	732	347	12	16661
16674	145	349	12	16831
16851	577	353	12	17011
17024	305	353	12	17183
17195	50	359	12	17359
17376	351	359	12	17539
17559	175	367	12	17729
17742	727	367	12	17911
17929	902	367	12	18097
18116	409	373	12	18289
18309	776	373	12	18481
18503	586	379	12	18679
18694	451	379	12	18869
18909	287	383	12	19087
19126	246	389	12	19309
19325	222	389	12	19507
19539	563	397	12	19727
19740	839	397	12	19927
19939	897	401	12	20129
20152	409	401	12	20341
20355	618	409	12	20551
20564	439	409	12	20759
20778	95	419	13	20983
20988	448	419	13	21191
21199	133	419	13	21401
21412	938	419	13	21613

[1080]

21629	423	431	13	21841
21852	90	431	13	22063
22073	640	431	13	22283
22301	922	433	13	22511
22536	250	439	13	22751
22779	367	439	13	22993
23010	447	443	13	23227
23252	559	449	13	23473
23491	121	457	13	23719
23730	623	457	13	23957
23971	450	457	13	24197
24215	253	461	13	24443
24476	106	467	13	24709
24721	863	467	13	24953
24976	148	479	13	25219
25230	427	479	13	25471
25493	138	479	13	25733
25756	794	487	13	26003
26022	247	487	13	26267
26291	562	491	13	26539
26566	53	499	13	26821
26838	135	499	13	27091
27111	21	503	13	27367
27392	201	509	13	27653
27682	169	521	13	27953
27959	70	521	13	28229
28248	386	521	13	28517
28548	226	523	13	28817
28845	3	541	13	29131
29138	769	541	13	29423
29434	590	541	13	29717
29731	672	541	13	30013
30037	713	547	13	30323

[1081]

30346	967	547	13	30631
30654	368	557	14	30949
30974	348	557	14	31267
31285	119	563	14	31583
31605	503	569	14	31907
31948	181	571	14	32251
32272	394	577	14	32579
32601	189	587	14	32917
32932	210	587	14	33247
33282	62	593	14	33601
33623	273	593	14	33941
33961	554	599	14	34283
34302	936	607	14	34631
34654	483	607	14	34981
35031	397	613	14	35363
35395	241	619	14	35731
35750	500	631	14	36097
36112	12	631	14	36457
36479	958	641	14	36833
36849	524	641	14	37201
37227	8	643	14	37579
37606	100	653	14	37967
37992	339	653	14	38351
38385	804	659	14	38749
38787	510	673	14	39163
39176	18	673	14	39551
39576	412	677	14	39953
39980	394	683	14	40361
40398	830	691	15	40787
40816	535	701	15	41213
41226	199	701	15	41621
41641	27	709	15	42043

[1082]

42067	298	709	15	42467
42490	368	719	15	42899
42916	755	727	15	43331
43388	379	727	15	43801
43840	73	733	15	44257
44279	387	739	15	44701
44729	457	751	15	45161
45183	761	751	15	45613
45638	855	757	15	46073
46104	370	769	15	46549
46574	261	769	15	47017
47047	299	787	15	47507
47523	920	787	15	47981
48007	269	787	15	48463
48489	862	797	15	48953
48976	349	809	15	49451
49470	103	809	15	49943
49978	115	821	15	50461
50511	93	821	16	50993
51017	982	827	16	51503
51530	432	839	16	52027
52062	340	853	16	52571
52586	173	853	16	53093
53114	421	857	16	53623
53650	330	863	16	54163
54188	624	877	16	54713
54735	233	877	16	55259
55289	362	883	16	55817
55843	963	907	16	56393
56403	471	907	16	56951

[1083] 表 2 :系统索引和其他参数

[1084] 5.7. 八位字节、码元和矩阵的操作

[1085] 5.7.1. 综述

[1086] 本节的其余部分描述了用于从源码元生成编码码元以及从编码码元生成源码元

的算术运算。在数学上，八位字节可被视为有限域的元素，即具有 256 个元素的有限域 GF(256)，因此定义了加法和乘法运算以及这两种运算上的单位元素和逆。矩阵运算和码元运算基于对八位字节的算术运算来定义。这允许在不必理解有限域的根本数学的情况下完全实现这些算术运算。

[1087] 5.7.2. 对八位字节的算术运算

[1088] 八位字节以普通方式被映射到范围 0 到 255 中的非负整数：来自码元的单个八位字节数据 B[7]、B[6]、B[5]、B[4]、B[3]、B[2]、B[1]、B[0]，其中 B[7] 是最高阶位而 B[0] 是最低阶位，被映射到整数 i = B[7]*128+B[6]*64+B[5]*32+B[4]*16+B[3]*8+B[2]*4+B[1]*2+B[0]。

[1089] 零个八位字节 u 和 v 的加法被定义为异或运算，即，

[1090] u+v = u^v。

[1091] 减法以相同的方式定义，因此得到

[1092] u-v = u^v。

[1093] 0 元素（加性单位）是由整数 0 表示的八位字节。u 的加性逆简单地为 u，即

[1094] u+u = 0。

[1095] 零个八位字节的乘法在两个表 OCT_EXP 和 OCT_LOG 的辅助下定义，这两个表分别在第 5.7.3 和 5.7.4 节中给出。表 OCT_LOG 将八位字节（非零元素）映射到非负整数，以及 OCT_EXP 将非负整数映射到八位字节。对于两个八位字节 u 和 v，定义

[1096] u*v =

[1097] 0，若 u 或 v 为 0，

[1098] OCT_EXP[OCT_LOG[u]+OCT_LOG[v]]，否则。

[1099] 注意，上面右侧的' +' 是普通的整数加法，因为其自变量是普通的整数。

[1100] 两个八位字节 u 和 v 的除法 u/v 且其中 v ! = 0 定义如下：

[1101] u/v =

[1102] 0，若 u == 0，

[1103] OCT_EXP[OCT_LOG[u]-OCT_LOG[v]+255]，否则。

[1104] 1 元素（乘性单位）是由整数 1 表示的八位字节。对于不是零元素的八位字节 u，即 u 的乘性逆为

[1105] OCT_EXP[255-OCT_LOG[u]]。

[1106] 由 α 标示的八位字节是具有整数表示 2 的八位字节。若 i 是非负整数 0 <= i < 256，得到

[1107] α^^i = OCT_EXP[i].

[1108] 5.7.3. 表 OCT_EXP

[1109] 表 OCT_EXP 包含 510 个八位字节。索引从 0 开始且范围直至 509，且条目是具有以下正整数表示的八位字节：

[1110] 1,2,4,8,16,32,64,128,29,58,116,232,205,135,19,38,76,152,45,90,180,117,234,201,143,3,6,12,24,48,96,192,157,39,78,156,37,74,148,53,106,212,181,119,238,193,159,35,70,140,5,10,20,40,80,160,93,186,105,210,185,111,222,161,95,190,97,194,153,47,94,188,101,202,137,15,30,60,120,240,253,231,211,187,107,

214,177,127,254,225,223,163,91,182,113,226,217,175,67,134,17,34,68,136,13,26,52,104,208,189,103,206,129,31,62,124,248,237,199,147,59,118,236,197,151,51,102,204,133,23,46,92,184,109,218,169,79,158,33,66,132,21,42,84,168,77,154,41,82,164,85,170,73,146,57,114,228,213,183,115,230,209,191,99,198,145,63,126,252,229,215,179,123,246,241,255,227,219,171,75,150,49,98,196,149,55,110,220,165,87,174,65,130,25,50,100,200,141,7,14,28,56,112,224,221,167,83,166,81,162,89,178,121,242,249,239,195,155,43,86,172,69,138,9,18,36,72,144,61,122,244,245,247,243,251,235,203,139,11,22,44,88,176,125,250,233,207,131,27,54,108,216,173,71,142,1,2,4,8,16,32,64,128,29,58,116,232,205,135,19,38,76,152,45,90,180,117,234,201,143,3,6,12,24,48,96,192,157,39,78,156,37,74,148,53,106,212,181,119,238,193,159,35,70,140,5,10,20,40,80,160,93,186,105,210,185,111,222,161,95,190,97,194,153,47,94,188,101,202,137,15,30,60,120,240,253,231,211,187,107,214,177,127,254,225,223,163,91,182,113,226,217,175,67,134,17,34,68,136,13,26,52,104,208,189,103,206,129,31,62,124,248,237,199,147,59,118,236,197,151,51,102,204,133,23,46,92,184,109,218,169,79,158,33,66,132,21,42,84,168,77,154,41,82,164,85,170,73,146,57,114,228,213,183,115,230,209,191,99,198,145,63,126,252,229,215,179,123,246,241,255,227,219,171,75,150,49,98,196,149,55,110,220,165,87,174,65,130,25,50,100,200,141,7,14,28,56,112,224,221,167,83,166,81,162,89,178,121,242,249,239,195,155,43,86,172,69,138,9,18,36,72,144,61,122,244,245,247,243,251,235,203,139,11,22,44,88,176,125,250,233,207,131,27,54,108,216,173,71,142

[1111] 5.7.4. 表 OCT_LOG

[1112] 表 OCT_LOG 包含 255 个非负整数。该表由解释为整数的八位字节来索引。由整数 0 表示的与零元素相对应的八位字节不被作为索引，因此索引从 1 开始且范围直至 255，且条目如下：

[1113] 0,1,25,2,50,26,198,3,223,51,238,27,104,199,75,4,100,224,14,52,141,239,129,28,193,105,248,200,8,76,113,5,138,101,47,225,36,15,33,53,147,142,218,240,18,130,69,29,181,194,125,106,39,249,185,201,154,9,120,77,228,114,166,6,191,139,98,102,221,48,253,226,152,37,179,16,145,34,136,54,208,148,206,143,150,219,189,241,210,19,92,131,56,70,64,30,66,182,163,195,72,126,110,107,58,40,84,250,133,186,61,202,94,155,159,10,21,121,43,78,212,229,172,115,243,167,87,7,112,192,247,140,128,99,13,103,74,222,237,49,197,254,24,227,165,153,119,38,184,180,124,17,68,146,217,35,32,137,46,55,63,209,91,149,188,207,205,144,135,151,178,220,252,190,97,242,86,211,171,20,42,93,158,132,60,57,83,71,109,65,162,31,45,67,216,183,123,164,118,196,23,73,236,127,12,111,246,108,161,59,82,41,157,85,170,251,96,134,177,187,204,62,90,203,89,95,176,156,169,160,81,11,245,22,235,122,117,44,215,79,174,213,233,230,231,173,232,116,214,244,234,168,80,88,175

[1114] 5.7.5. 对码元的运算

[1115] 对码元的运算具有与本规范中对长度为 T 的八位字节向量的运算相同的语义。因此，若 U 和 V 是分别由八位字节 u[0]、…、u[T-1] 和 v[0]、…、v[T-1] 形成的两个码元，则码元和 U+V 被定义为八位字节的逐分量求和，即等于由八位字节 d[0]、…、d[T-1] 形成的码元，使得

[1116] d[i] = u[i]+v[i], 0 <= i < T。

[1117] 此外，若 β 是八位字节，则乘积 β*U 被定义为通过将 U 的每个八位字节乘以 β 获得的码元 D，即，

[1118] d[i] = beta*u[i], 0 <= i < T。

[1119] 5.7.6. 对矩阵的运算

[1120] 本规范中的所有矩阵具有为八位字节的条目，且因此矩阵运算和定义是以底层八位字节算术的方式定义的，例如对矩阵的运算、矩阵秩和矩阵逆。

[1121] 5.8. 对兼容解码器的要求

[1122] 若 RaptorQ 兼容解码器接收在数学上充分的根据第 5.3 节中的编码器规范生成的编码码元集合以重构源块，则此类解码器应当恢复整个源块。

[1123] 对于具有 K′ 个源码元的源块，对于第 5.6 节表 2 中的所有 K′ 值，RaptorQ 兼容解码器应具有以下恢复属性。

[1124] 1. 若解码器接收根据第 5.3 节中的编码器规范生成的具有独立且均匀随机地从可能的 ESI 范围选取的相应 ESI 的 K′ 个编码码元，则平均来说，解码器在 100 次中至多有 1 次将不能恢复整个源块。

[1125] 2. 若解码器接收根据第 5.3 节中的编码器规范生成的具有独立且均匀随机地从可能的 ESI 范围选取的相应 ESI 的 K′ +1 个编码码元，则平均来说，解码器在 10,000 次中至多有 1 次将不能恢复整个源块。

[1126] 3. 若解码器接收根据第 5.3 节中的编码器规范生成的具有独立且均匀随机地从可能的 ESI 范围选取的相应 ESI 的 K′ +2 个编码码元，则平均来说，解码器在 1,000,000 次中至多有 1 次将不能恢复整个源块。

[1127] 注意，第 5.4 节中指定的示例 FEC 解码器履行两个要求，即

[1128] 1. 只要接收到在数学上充分的根据第 5.3 节中的编码器规范生成的编码码元集合，它就能重构源块；

[1129] 2. 它履行上面的强制恢复属性。

[1130] 6. 安全性考虑

[1131] 数据递送可受到攻击者的拒绝服务攻击，攻击者发送被接收方作为合法的来接收的讹误分组。这对于多播递送尤其是个问题，因为讹误分组可能被注入靠近多播树的根的会话中，在这种情形中，讹误分组将抵达许多接收方。这在使用本文档中描述的代码时尤其是个问题，因为使用甚至一个包含编码数据的讹误分组就会导致解码完全讹误和无用的对象。因此，推荐在将对象递送给应用之前向经解码对象应用源认证和完好性检查。例如，可在传输前追加对象的 SHA-1 散列 [SHA1]，且在解码对象之后但在将它递送给应用之前计算和检查 SHA-1 散列。例如，应当通过包括可由接收方验证的在散列值上计算出的数字签名来提供源认证。还推荐使用诸如 TESLA[RFC4082] 之类的分组认证协议来在抵达之际检测和丢弃讹误分组。该方法也可用来提供源认证。此外，推荐在沿着从发送方到接收方的路

径上的所有网络路由器和交换机中实现反向路径转发检查，以限制不良代理向多播树数据路径中注入讹误分组的可能性。

[1132] 另一安全问题是一些 FEC 信息可能被带外接收方在会话描述中获得，且若会话描述是伪造的或讹误的，则接收方将不会使用正确的协议来从收到分组解码内容。为了避免这些问题，推荐采取措施防止接收方接受不正确的会话描述，例如通过使用源认证来确保接收方仅接受来自授权发送方的合法会话描述。

[1133] 7. IANA 考虑

[1134] FEC 编码 ID 和 FEC 实例 ID 的值要进行 IANA 注册。对于 IANA 考虑在应用于本文档时的一般性指南，参见 [RFC5052]。本文档依照 ietf:rmt:fec :编码命名空间到“RaptorQ 码”来指派全规范 FEC 编码 ID 6(tbc)。

[1135] 8. 感谢

[1136] 感谢Ranganathan(Ranga)Krishnan。Ranga Krishnan非常支持发现和解决实现细节以及发现系统索引。此外，皆来自慕尼黑工业大学 (TUM) 的Habeeb Mohiuddin Mohammed和Antonios Pitarokoilis、以及Alan Shinsato已作出有助于阐明和用本规范解决问题的 RaptorQ 编码器 / 解码器的独立实现。

[1137] 9. 参考文献

[1138] 9.1. 标准性参考文献

[1139] [RFC2119]Bradner,S.,″ Key words for use in RFCs to Indicate Requirement Levels(用于 RFC 以指示要求等级的关键词)″，BCP 14,RFC 2119,1997 年 3 月。

[1140] [RFC4082]Perrig,A.、Song,D.、Canetti,R.、Tygar,J. 和 B.Briscoe, ″ Timed Efficient Stream Loss-Tolerant Authentication(TESLA):Multicast Source Authentication Transform Introduction(时间高效流耐丢失认证 (TESLA):多播源认证变换介绍)″，RFC 4082,2005 年 6 月。

[1141] [SHA1]″ Secure Hash Standard(安全散列标准)″，Federal Information Processing Standards Publication(联邦信息处理标准公布)(FIPS PUB)180-1,2005 年 4 月。

[1142] [RFC5052]Watson,M.、Luby,M. 和 L.Vicisano, ″ Forward Error Correction(FEC)Building Block(前向纠错 (FEC) 构建块)″，RFC 5052,2007 年 8 月。

[1143] 9.2. 信息性参考文献

[1144] [RFC3453]Luby,M.、Vicisano,L.、Gemmell,J.、Rizzo,L.、Handley,M. 和 J.Crowcroff,″ The Use of Forward Error Correction(FEC)in Reliable Multicast(前向纠错 (FEC) 在可靠多播中的使用)″，RFC 3453,2002 年 12 月。

[1145] [RFC5053]Luby,M.、Shokrollahi,A.、Watson,M. 和 T.Stockhammer, ″ Raptor Forward Error Correction Scheme for Object Delivery(用于对象递送的 Raptor 前向纠错方案)″，RFC 5053,2007 年 10 月。

[1146] 作者地址

[1147] Michael Luby

[1148] Qualcomm Incorporated 3165 Kifer Road

[1149] Santa Clara,CA 95051

[1150] U.S.A.
[1151] Email:luby@qualcomm.com
[1152] Amin Shokrollahi
[1153] EPFL
[1154] Laboratoire d′ algorithmique
[1155] EPFL
[1156] Station 14
[1157] Batiment BC
[1158] Lausanne 1015
[1159] Switzerland
[1160] Email:amin.shokrollahi@epfl.ch
[1161] Mark Watson
[1162] Qualcomm Incorporated
[1163] 3165 Kifer Road
[1164] Santa Clara,CA 95051
[1165] U.S.A.
[1166] Email:watson@qualcomm.com
[1167] Thomas Stockhammer
[1168] Nomor Research
[1169] Brecherspitzstrasse 8
[1170] Munich 81541
[1171] Germany
[1172] Email:stockhammer@nomor.de
[1173] Lorenz Minder
[1174] Qualcomm Incorporated
[1175] 3165 Kifer Road
[1176] Santa Clara,CA 95051
[1177] U.S.A.
[1178] Email:lminder@qualcomm.com

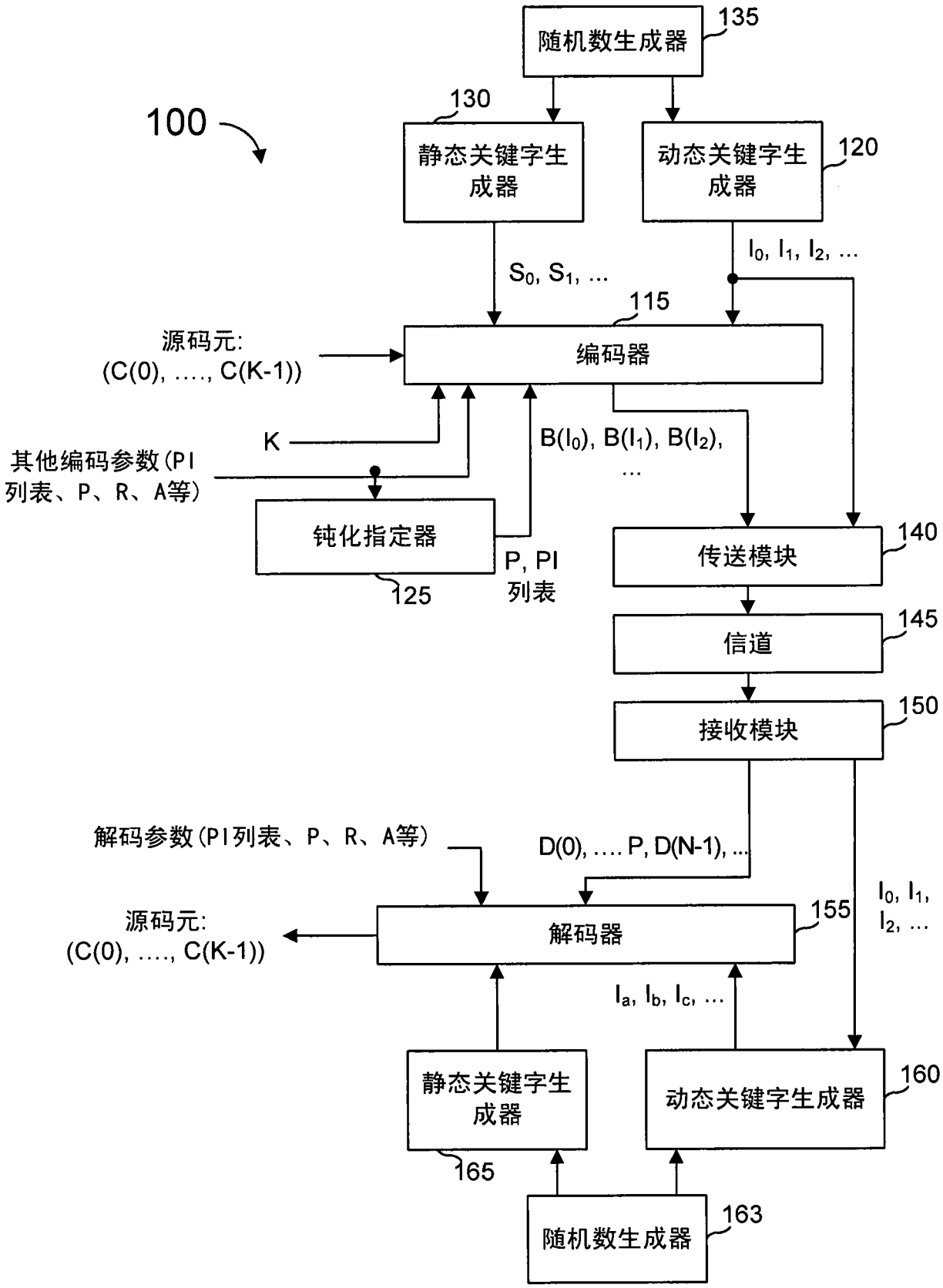

随机数生成器
135
100
130
静态关键字生成器
动态关键字生成器
120
$I_0, I_1, I_2, \ldots$
$S_0, S_1, \ldots$
115
源码元:
(C(0), ..., C(K-1))
编码器
K
其他编码参数(PI列表、P、R、A等)
$B(I_0), B(I_1), B(I_2), \ldots$
钝化指定器
P, PI列表
125
传送模块
140
信道
145
接收模块
150
解码参数(PI列表、P、R、A等)
D(0), ... P, D(N-1), ...
$I_0, I_1, I_2, \ldots$
源码元:
(C(0), ..., C(K-1))
解码器
155
$I_a, I_b, I_c, \ldots$
静态关键字生成器
动态关键字生成器
160
165
随机数生成器
163

图 1

使用的数据值

K	源码元数目
R	冗余码元数目
L	中间码元数目(L = K+R)
S	LDPC 码元数目
H	HDPC 码元数目
P	永久钝化(PI)码元数目
N	收到经编码码元数目
A	开销码元数目(A = N-K)

码元阵列

(C(0), ..., C(K-1))	源码元
(C(K), ..., C(L-1))	冗余码元
(C(0), ..., C(L-1))	中间码元
(C(0), ..., C(L-P-1))	LT 中间码元
(C(L-P), ..., C(L-1))	永久钝化(PI)码元(即，“PI 列表”)
(D(0), ..., D(N-1))	解码器 155 接收到的经编码码元

图 2

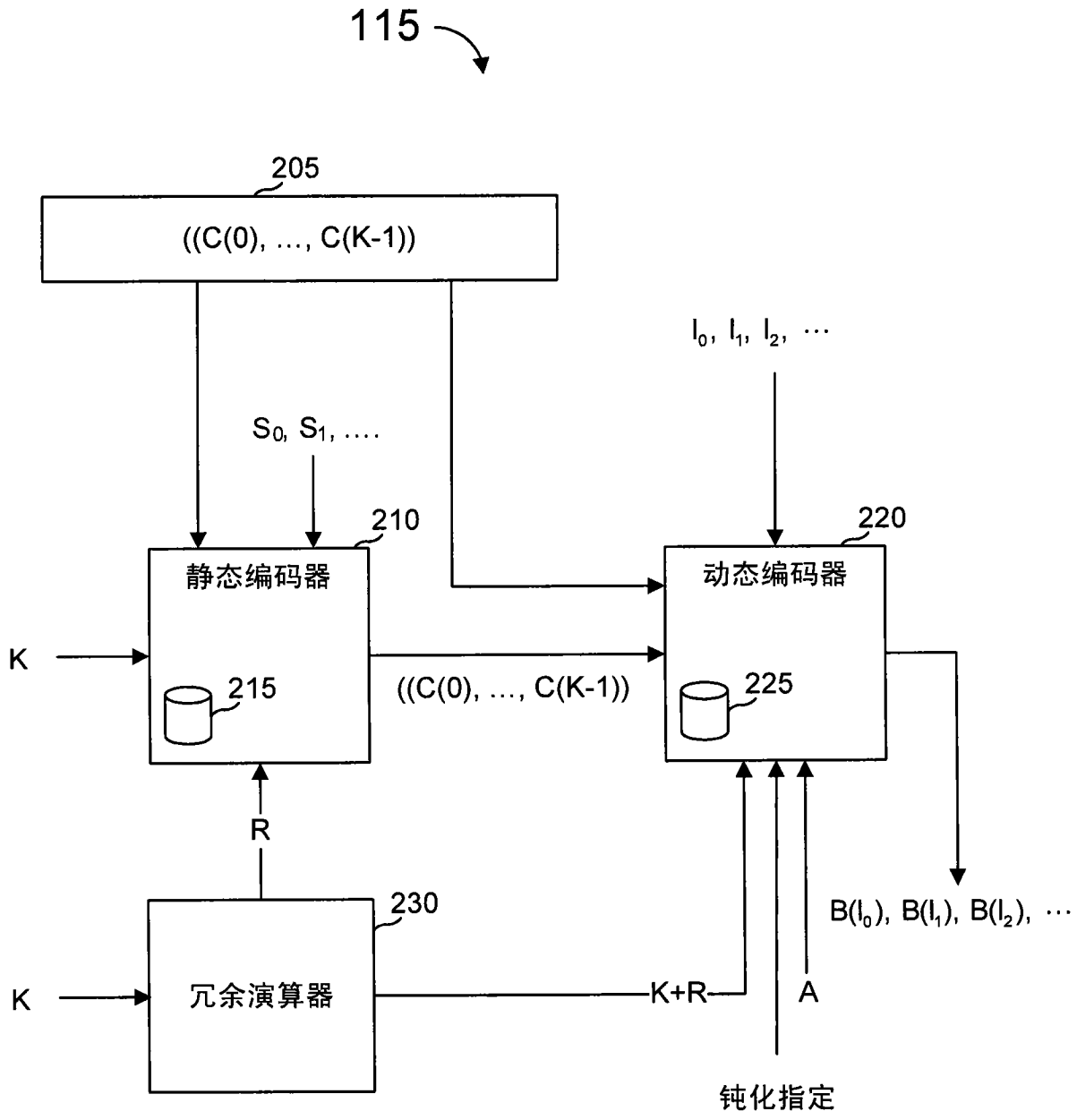

115
205
((C(0), …, C(K-1))
I_0, I_1, I_2, …
S_0, S_1, ….
210
220
静态编码器
动态编码器
K
215
((C(0), …, C(K-1))
225
R
230
$B(I_0)$, $B(I_1)$, $B(I_2)$, …
K
冗余演算器
K+R
A
钝化指定

图 3

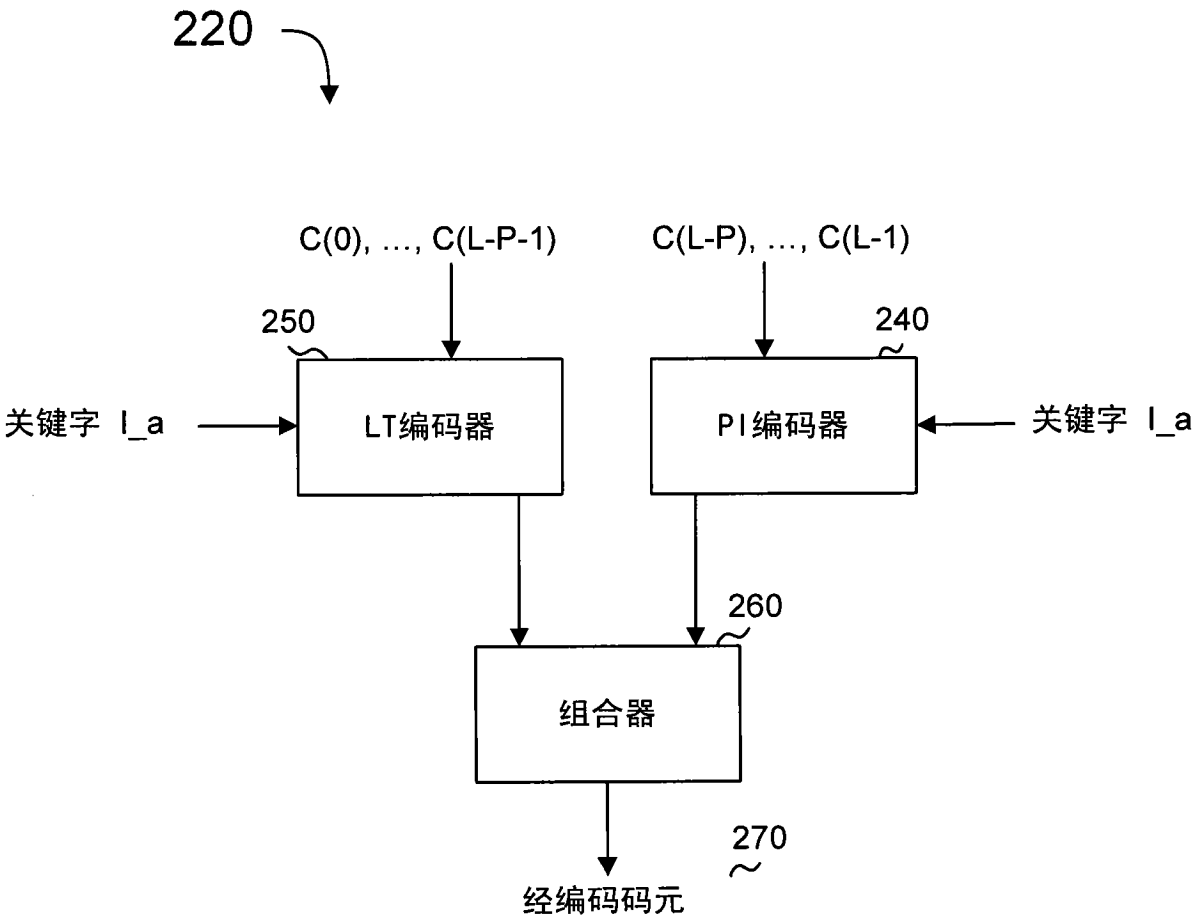

220
C(0), ..., C(L-P-1)
C(L-P), ..., C(L-1)
250
240
关键字 I_a
LT编码器
PI编码器
关键字 I_a
260
组合器
270
经编码码元

图 4

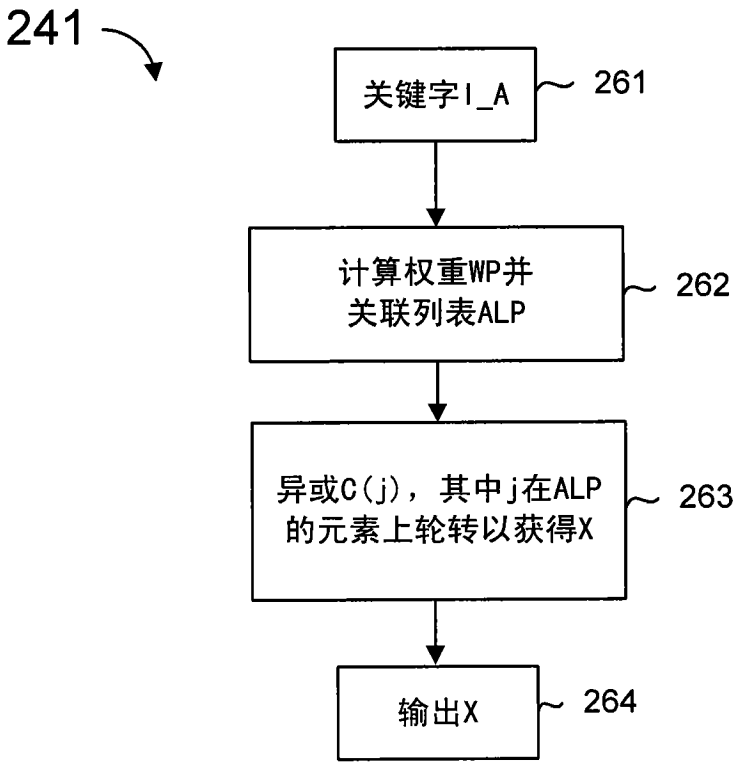

241
关键字I_A
261
计算权重WP并关联列表ALP
262
异或C(j)，其中j在ALP的元素上轮转以获得X
263
输出X
264

图 5

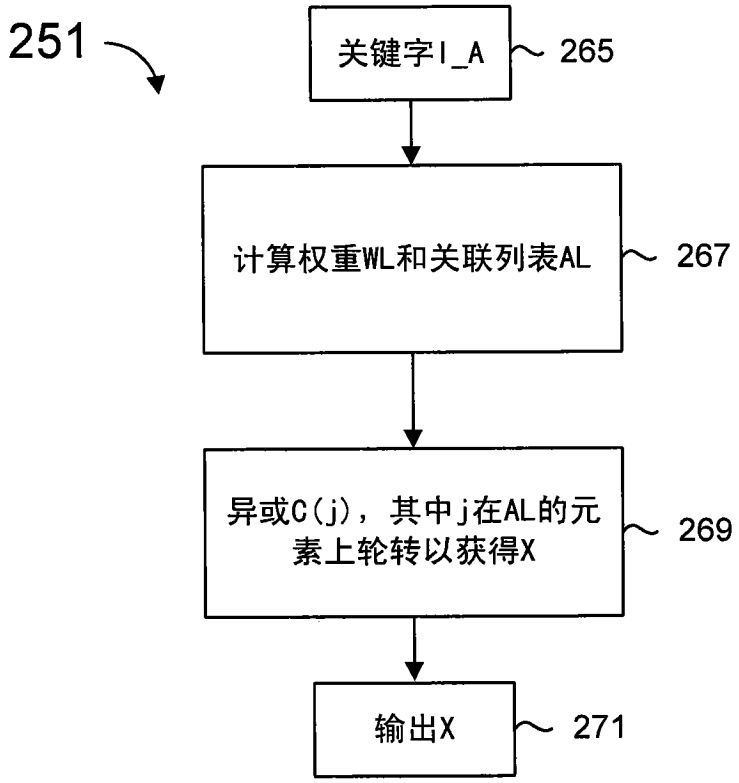

251
关键字I_A
265
计算权重WL和关联列表AL
267
异或C(j)，其中j在AL的元素上轮转以获得X
269
输出X
271

图 6

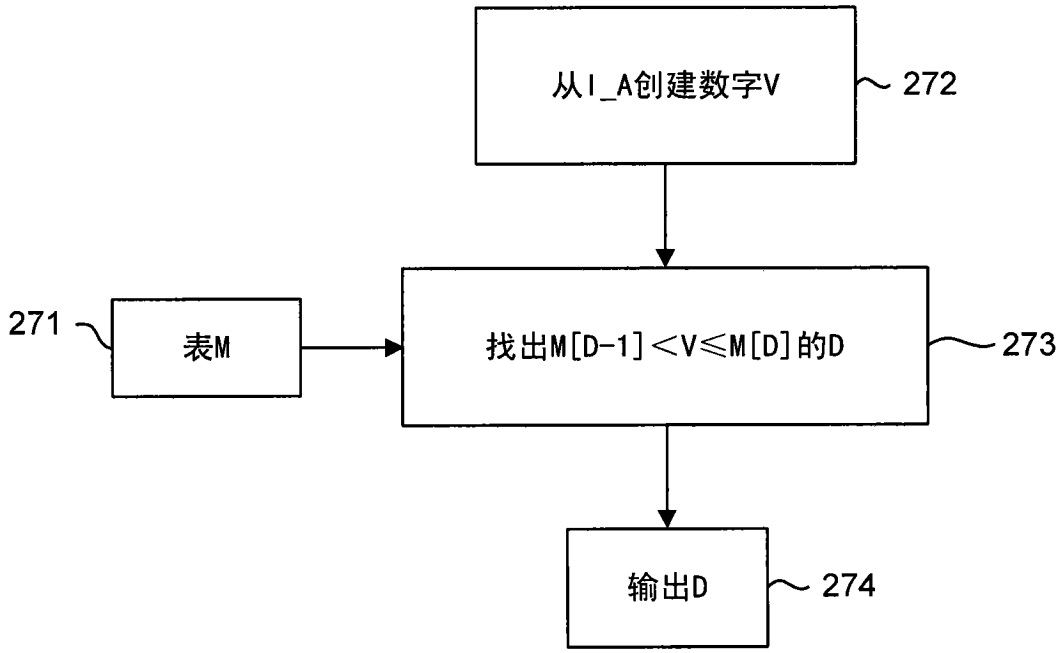


图 7

d	m[d]
30	1048576
29	1017662
28	1016370
27	1014983
26	1013490
25	1011876
24	1010129
23	1008229
22	1006157
21	1003887
20	1001391
19	998631
18	995565
17	992138
16	988283
15	983914
14	978921
13	973160
12	966438
11	958494
10	948962
9	937311
8	922747
7	904023
6	879057
5	844104
4	791675
3	704294
2	529531
1	5243
0	0

图 8

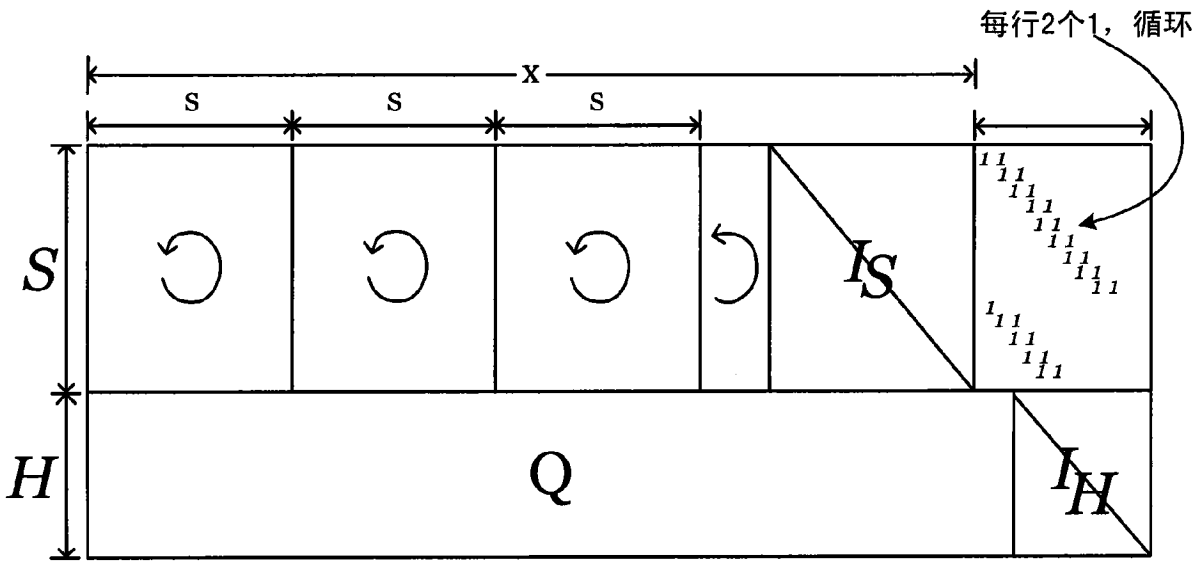


图 9

$$Q=(\Delta_1|\Delta_2|\cdots|\Delta_{k+S-1}|Y)\cdot\Gamma,\Gamma=\begin{pmatrix}1&0&0&\cdots&0&0\\ \alpha&1&0&\cdots&0&0\\ 0&\alpha&1&\cdots&0&0\\ \vdots&\vdots&\vdots&\ddots&\vdots&\vdots\\ 0&0&0&\cdots&1&0\\ 0&0&0&\cdots&\alpha&1\end{pmatrix}^{-1},Y=\begin{pmatrix}a^0\\ a^1\\ \vdots\\ \alpha^{H-2}\\ a^{H-1}\end{pmatrix}$$

图 10

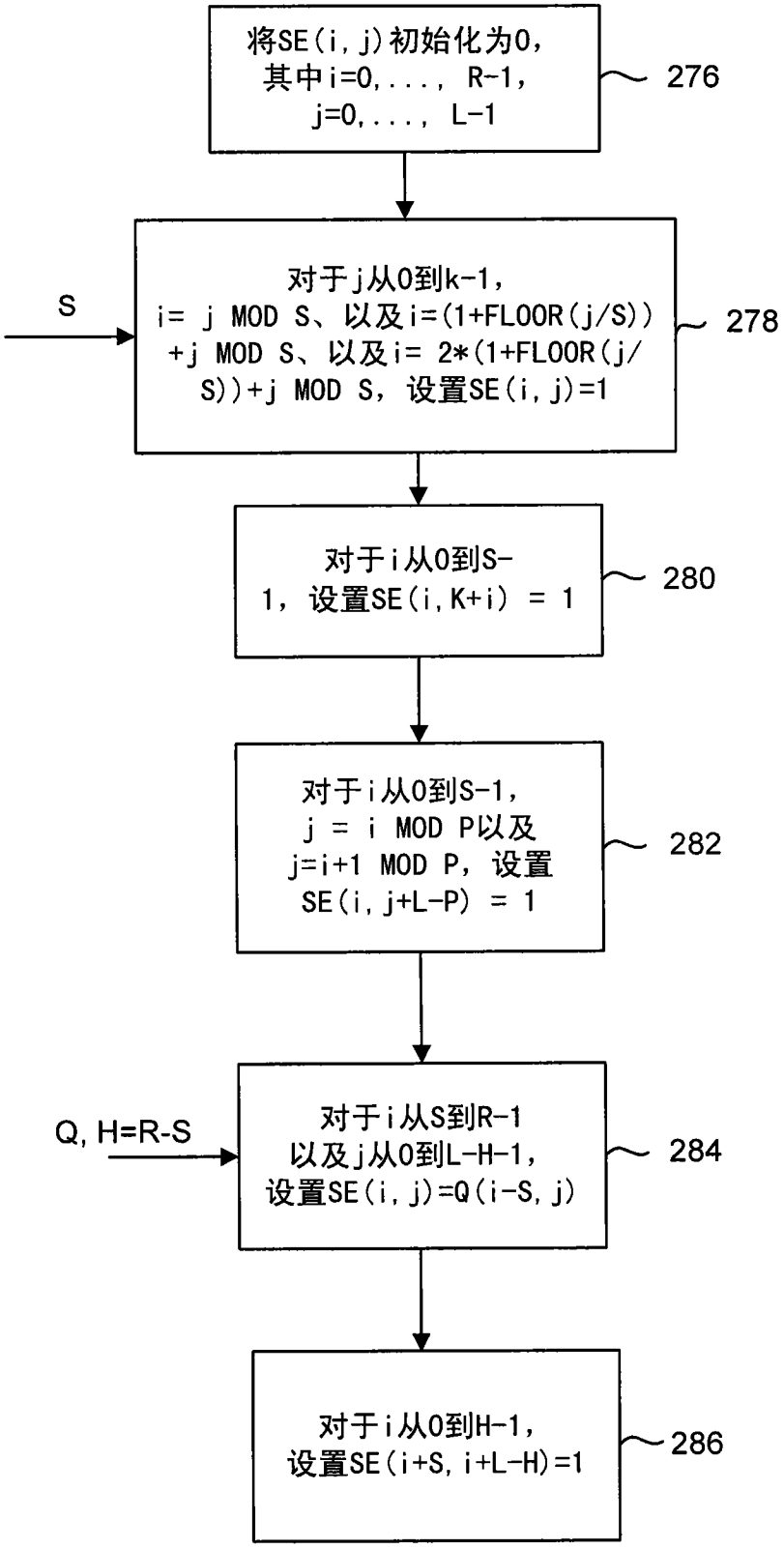

将SE(i, j)初始化为0，
其中i=0, . . . , R-1，
j=0, . . . , L-1
276
S
对于j从0到k-1，
i= j MOD S、以及i=(1+FLOOR(j/S))
+j MOD S、以及i= 2*(1+FLOOR(j/
S))+j MOD S，设置SE(i, j)=1
278
对于i从0到S-
1，设置SE(i, K+i) = 1
280
对于i从0到S-1，
j = i MOD P以及
j=i+1 MOD P，设置
SE(i, j+L-P) = 1
282
Q, H=R-S
对于i从S到R-1
以及j从0到L-H-1，
设置SE(i, j)=Q(i-S, j)
284
对于i从0到H-1，
设置SE(i+S, i+L-H)=1
286

图 11

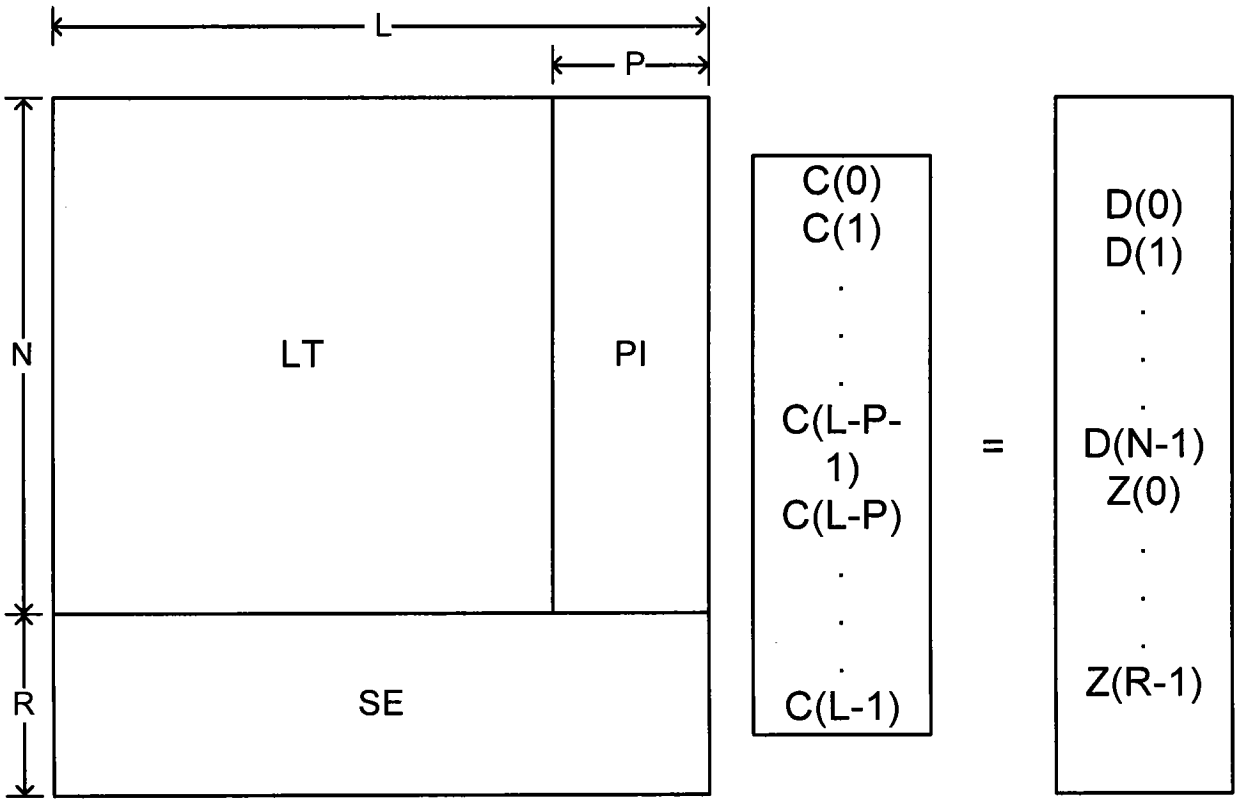


图 12

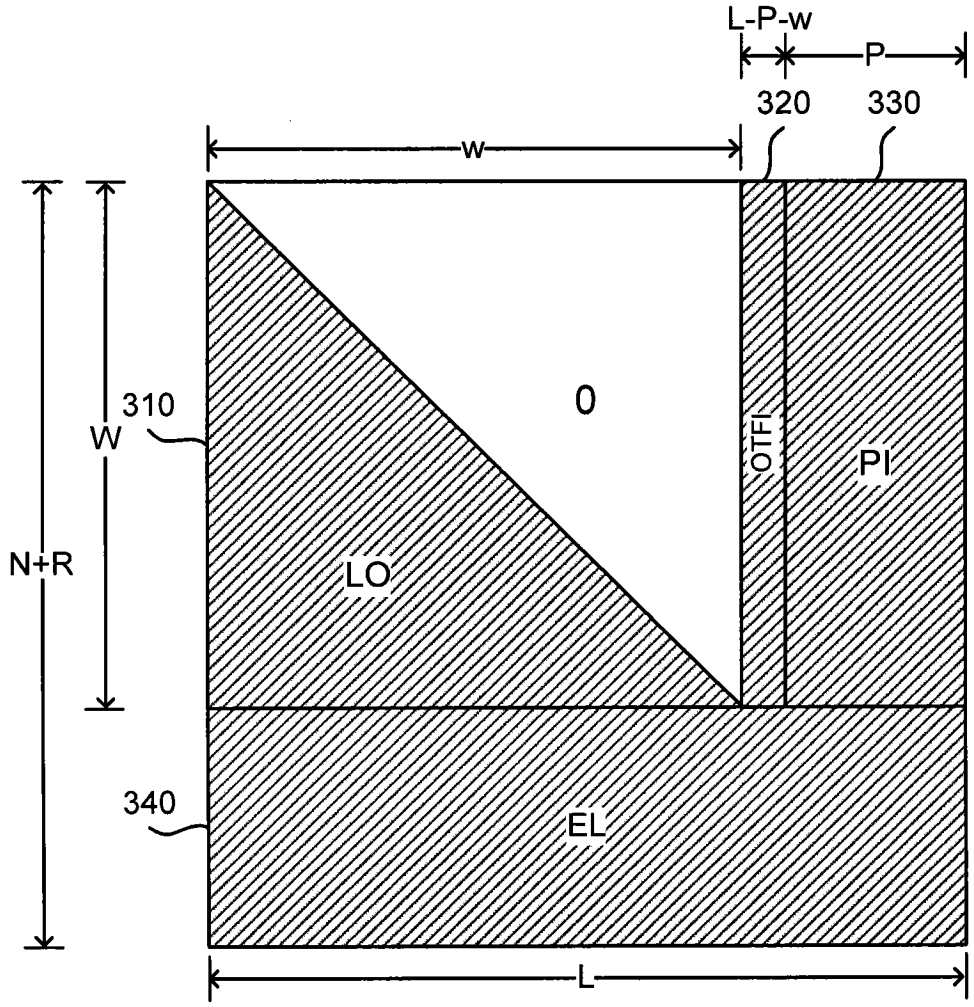


图 13

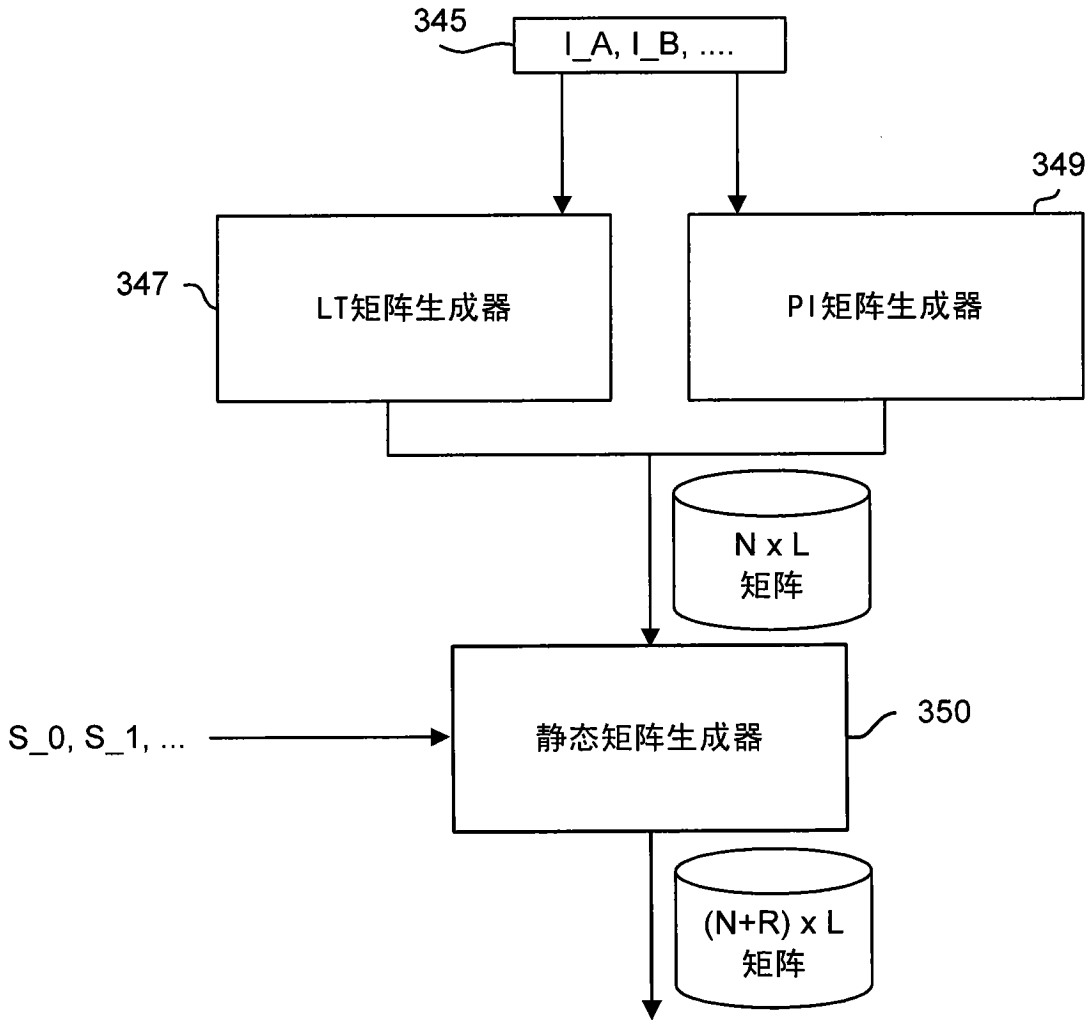

345
I_A, I_B, ...
349
347
LT矩阵生成器
PI矩阵生成器
N x L
矩阵
S_0, S_1, ...
静态矩阵生成器
350
(N+R) x L
矩阵

图 14

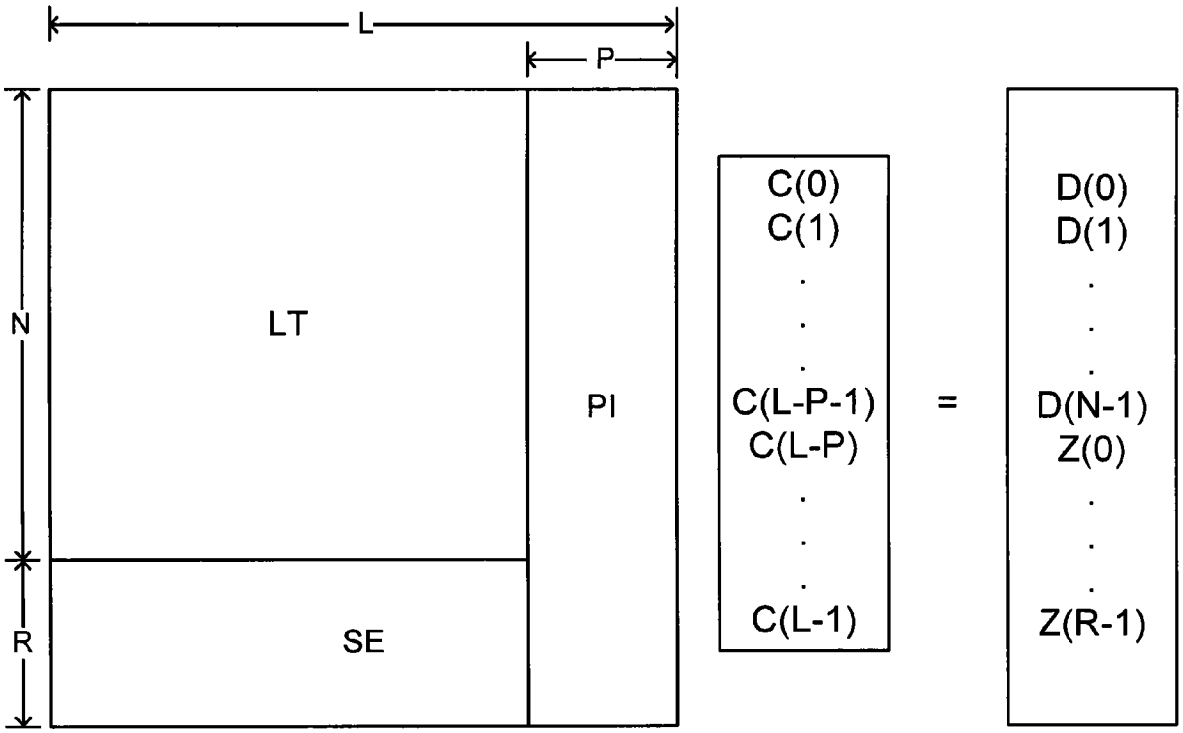


图 15

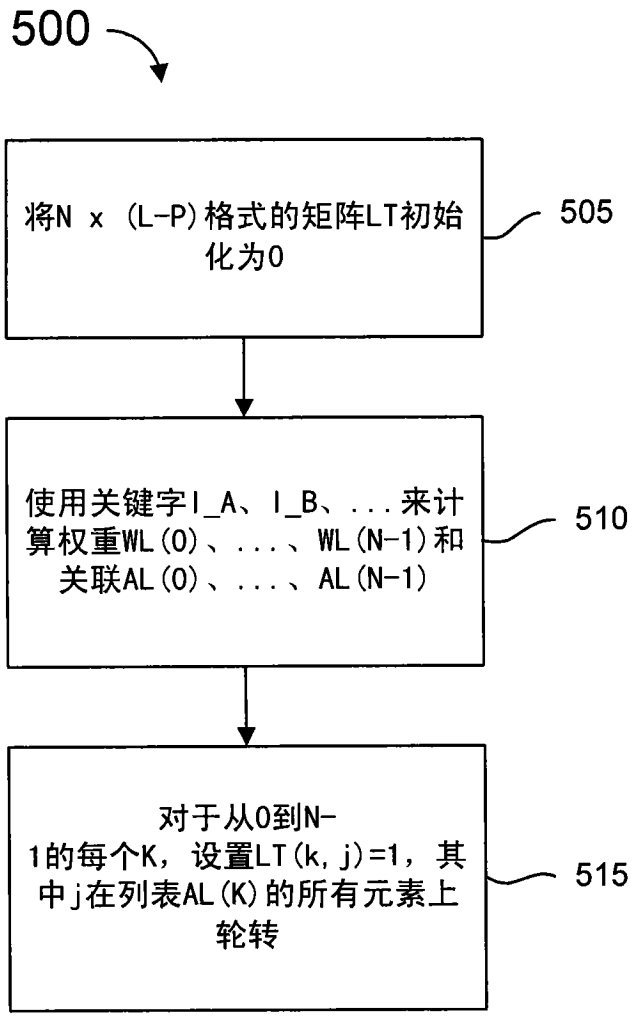

500
将N x (L-P)格式的矩阵LT初始化为0
505
使用关键字I_A、I_B、...来计算权重WL(0)、...、WL(N-1)和关联AL(0)、...、AL(N-1)
510
对于从0到N-1的每个K，设置LT(k,j)=1，其中j在列表AL(K)的所有元素上轮转
515

图 16

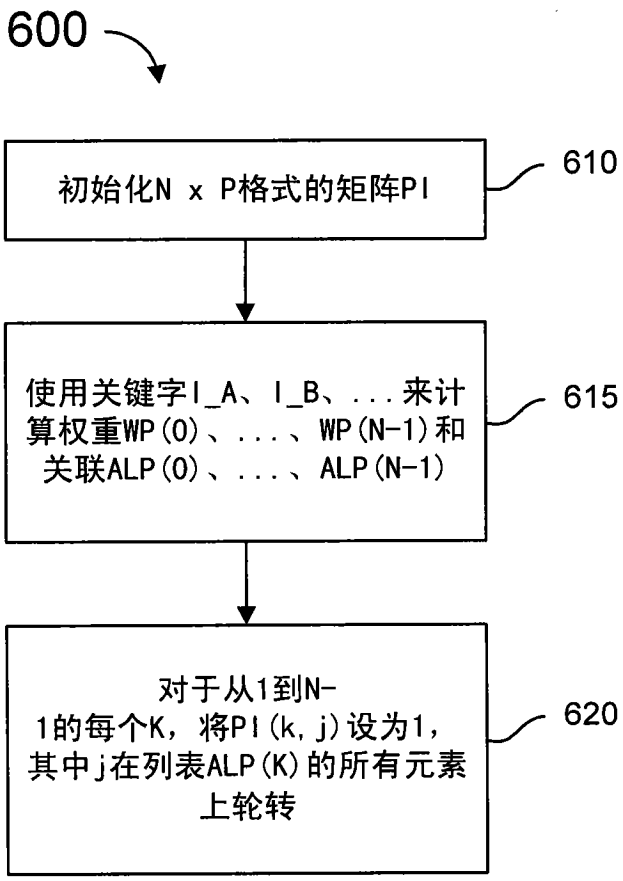

600
初始化N x P格式的矩阵PI
610
使用关键字I_A、I_B、...来计算权重WP(0)、...、WP(N-1)和关联ALP(0)、...、ALP(N-1)
615
对于从1到N-1的每个K，将PI(k, j)设为1，其中j在列表ALP(K)的所有元素上轮转
620

图 17

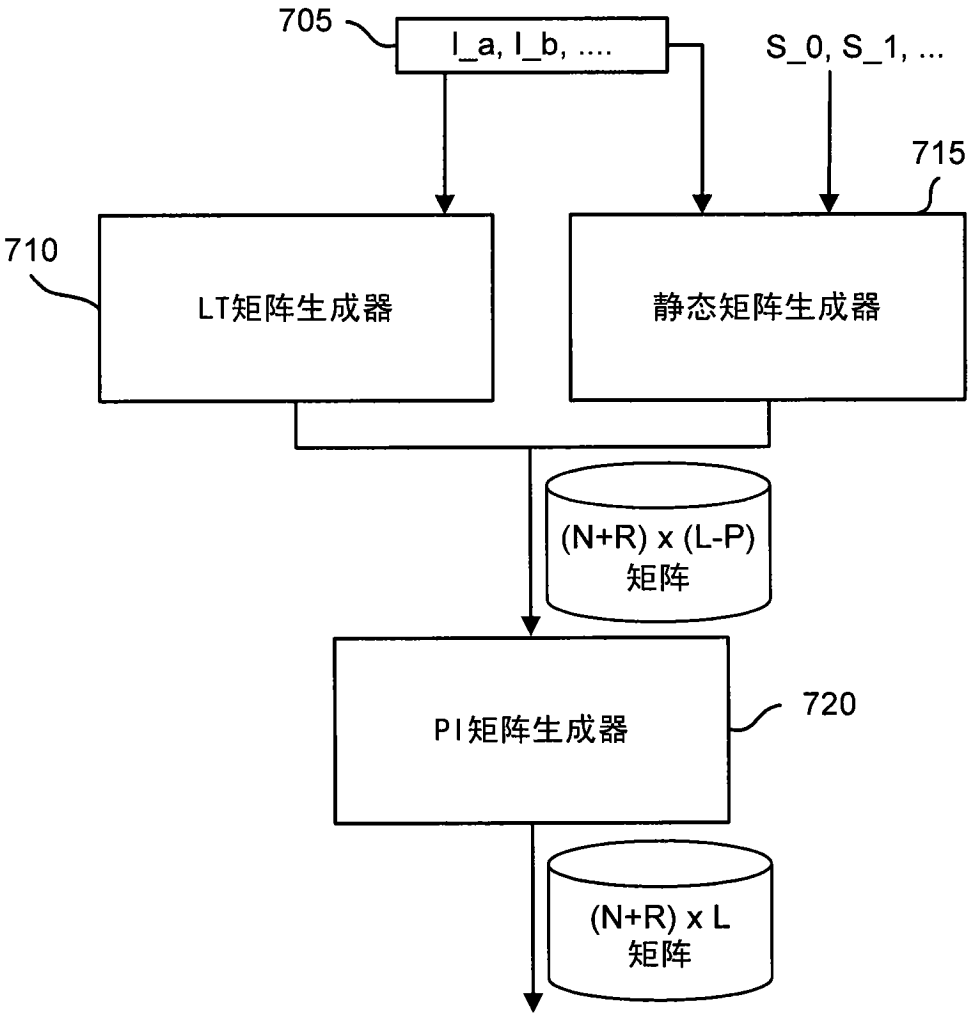

705
I_a, I_b, ...
S_0, S_1, ...
715
710
LT矩阵生成器
静态矩阵生成器
(N+R) x (L-P)
矩阵
PI矩阵生成器
720
(N+R) x L
矩阵

图 18

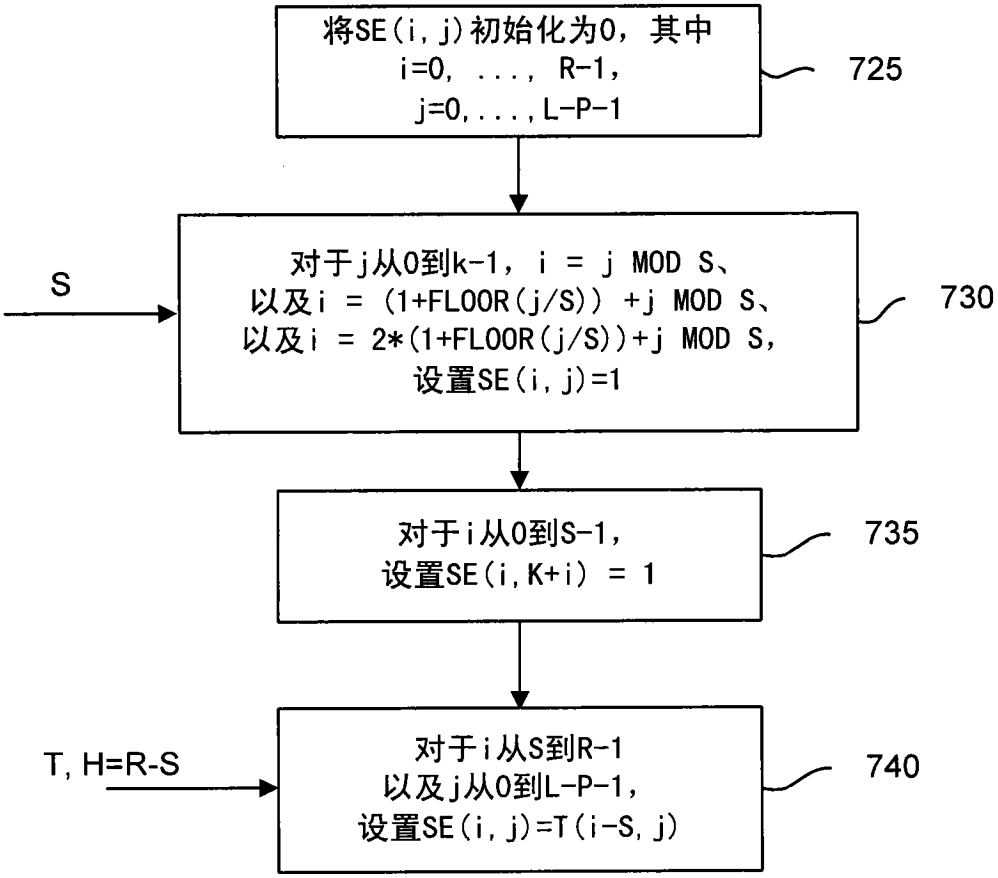

将SE(i,j)初始化为0，其中
i=0, ..., R-1,
j=0,...,L-P-1
725
S
对于j从0到k-1，i = j MOD S、
以及i = (1+FLOOR(j/S)) +j MOD S、
以及i = 2*(1+FLOOR(j/S))+j MOD S，
设置SE(i,j)=1
730
对于i从0到S-1，
设置SE(i,K+i) = 1
735
T, H=R-S
对于i从S到R-1
以及j从0到L-P-1，
设置SE(i,j)=T(i-S,j)
740

图 19

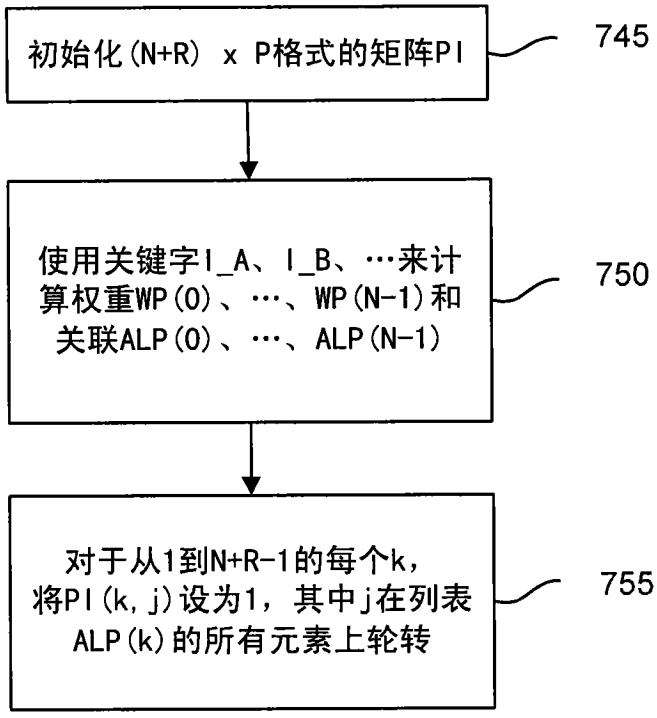

初始化(N+R) x P格式的矩阵PI
745
使用关键字I_A、I_B、…来计
算权重WP(0)、…、WP(N-1)和
关联ALP(0)、…、ALP(N-1)
750
对于从1到N+R-1的每个k，
将PI(k,j)设为1，其中j在列表
ALP(k)的所有元素上轮转
755

图 20

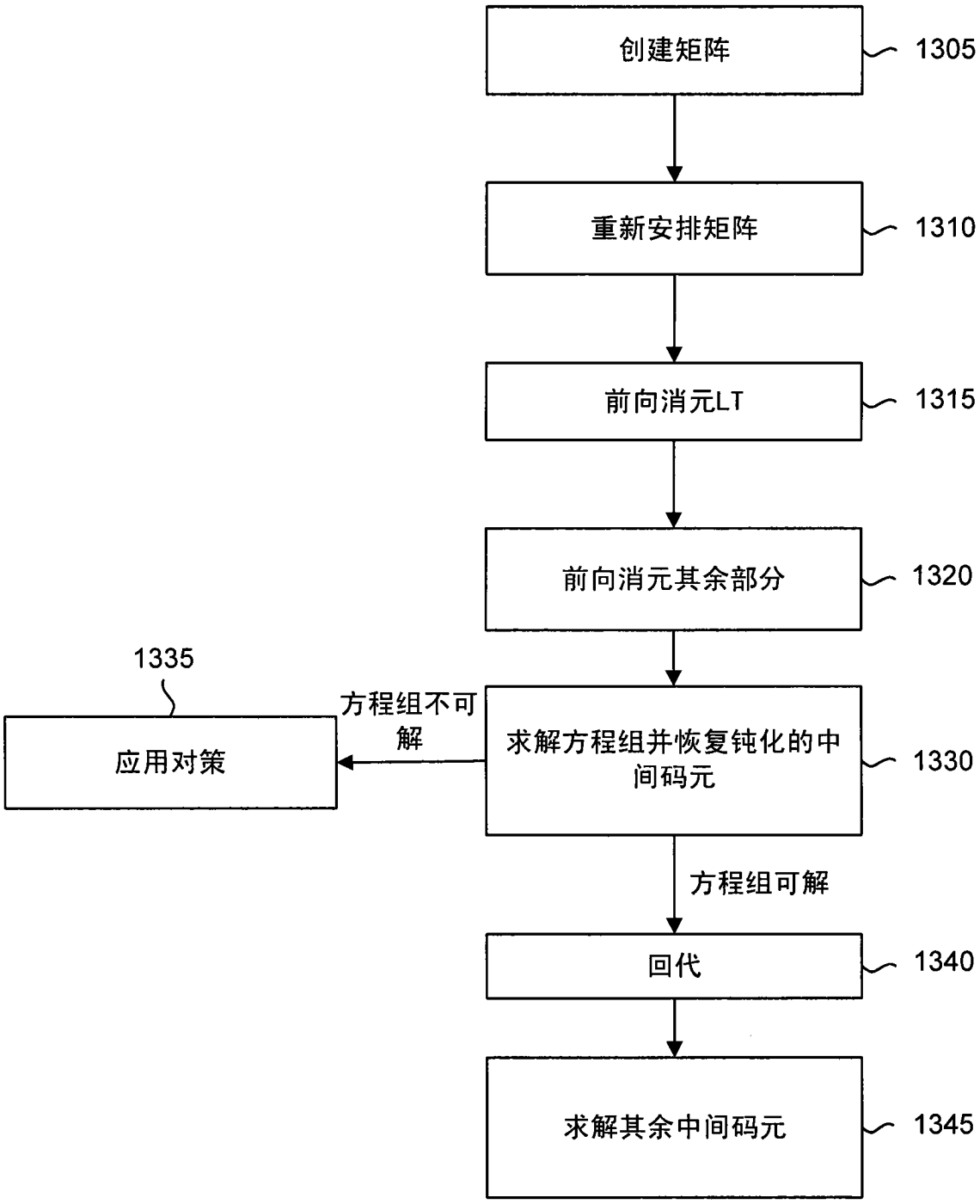

创建矩阵
1305
重新安排矩阵
1310
前向消元LT
1315
前向消元其余部分
1320
求解方程组并恢复钝化的中间码元
1330
方程组不可解
1335
应用对策
方程组可解
回代
1340
求解其余中间码元
1345

图 21

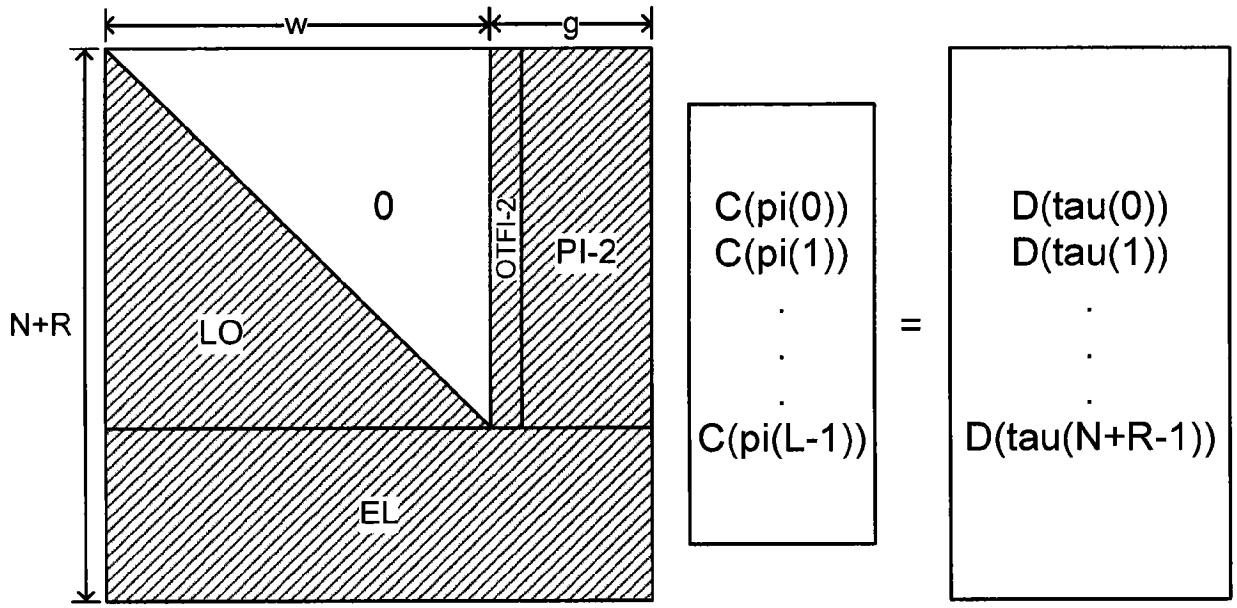

w
g
N+R
0
LO
EL
OTFI-2
PI-2
C(pi(0))
C(pi(1))
C(pi(L-1))
=
D(tau(0))
D(tau(1))
D(tau(N+R-1))

图 22

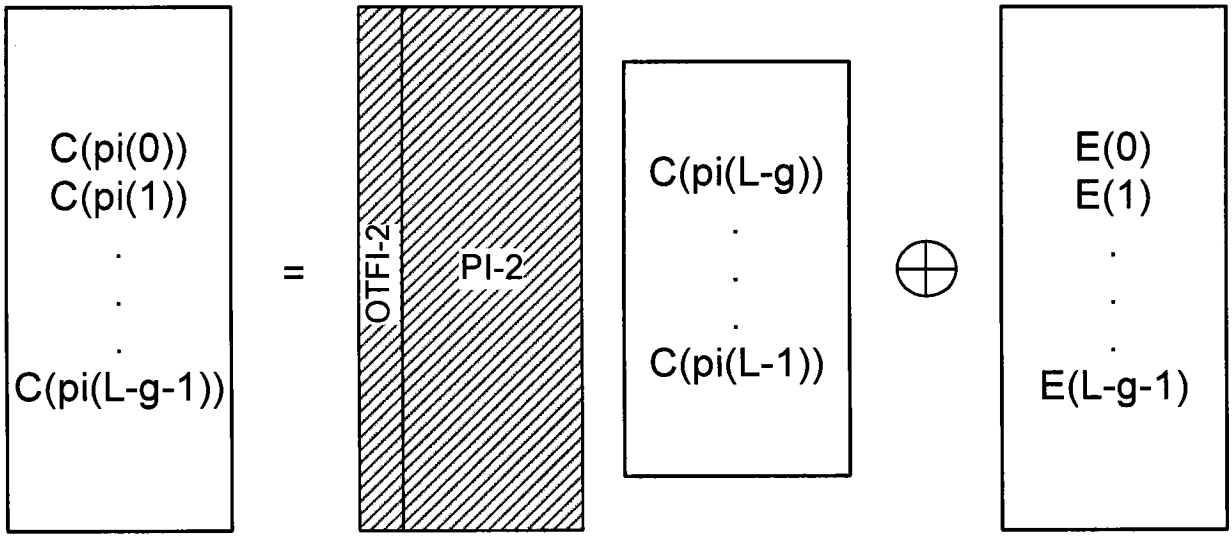

C(pi(0))
C(pi(1))
C(pi(L-g-1))
=
OTFI-2
PI-2
C(pi(L-g))
C(pi(L-1))
E(0)
E(1)
E(L-g-1)

图 23

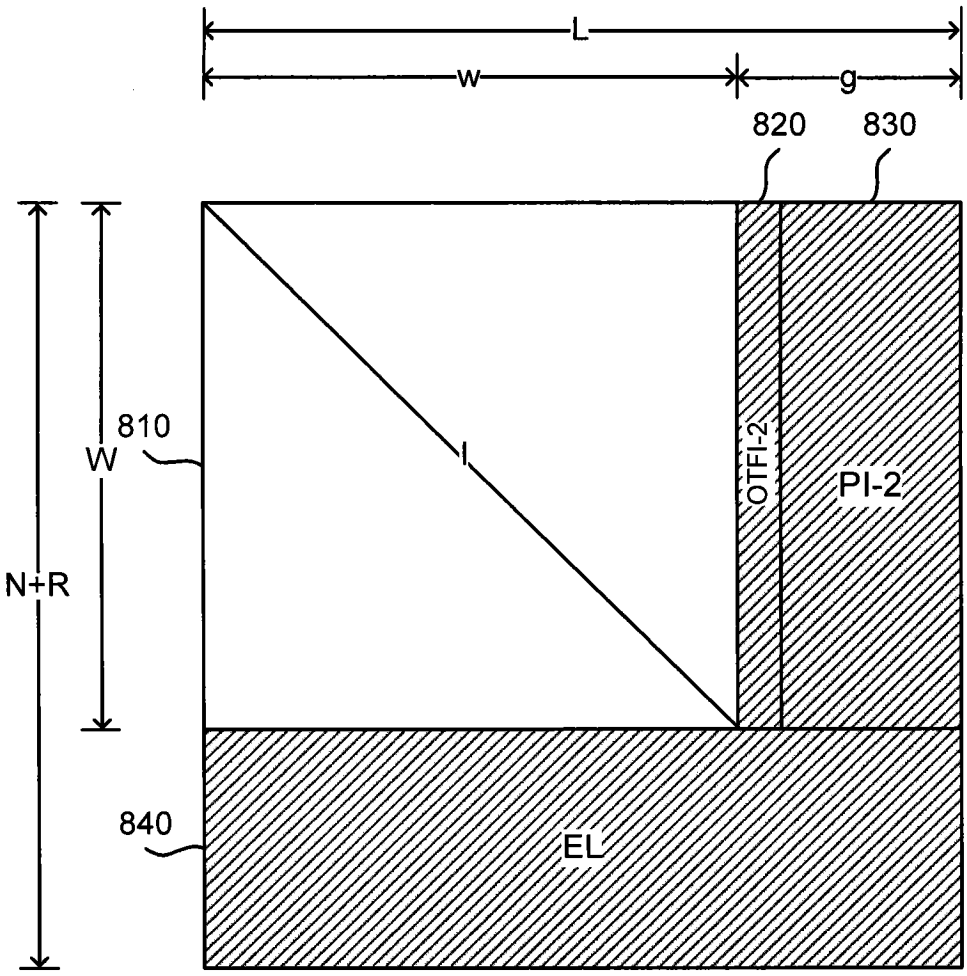


图 24

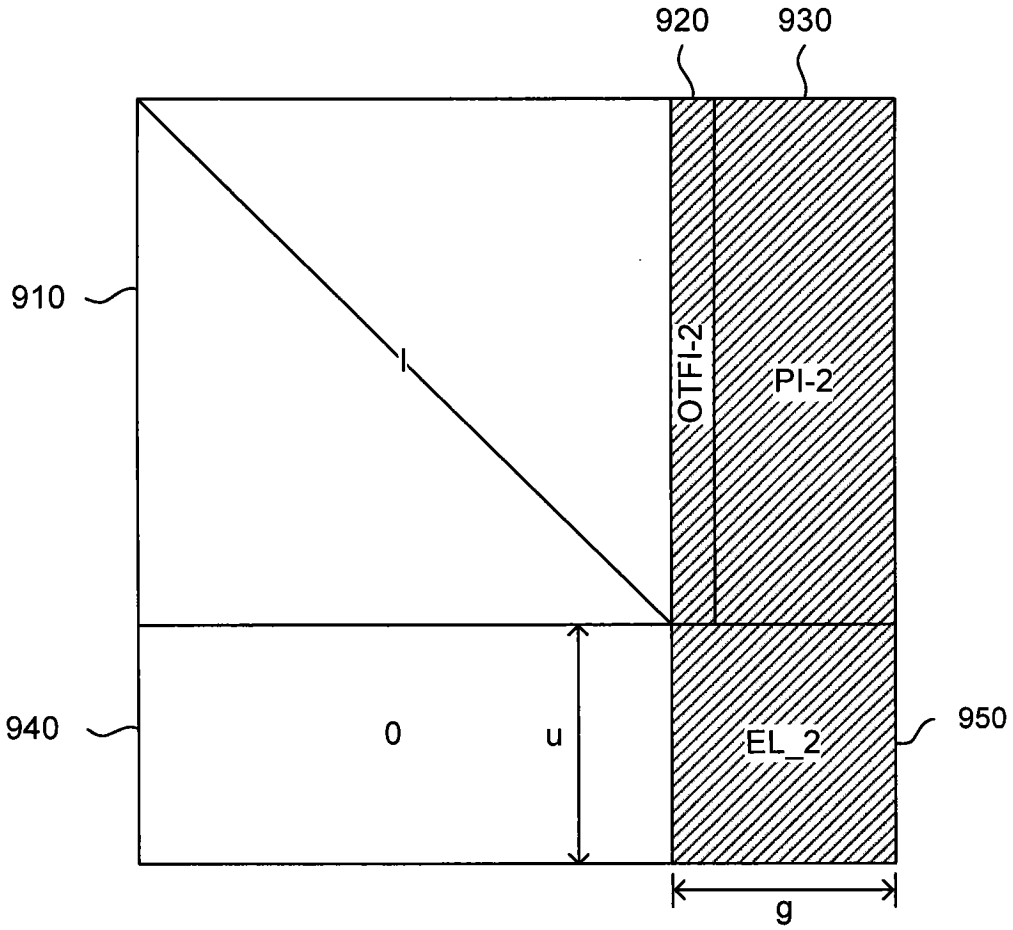

920
930
910
I
OTFI-2
PI-2
940
0
u
EL_2
950
g

图 25

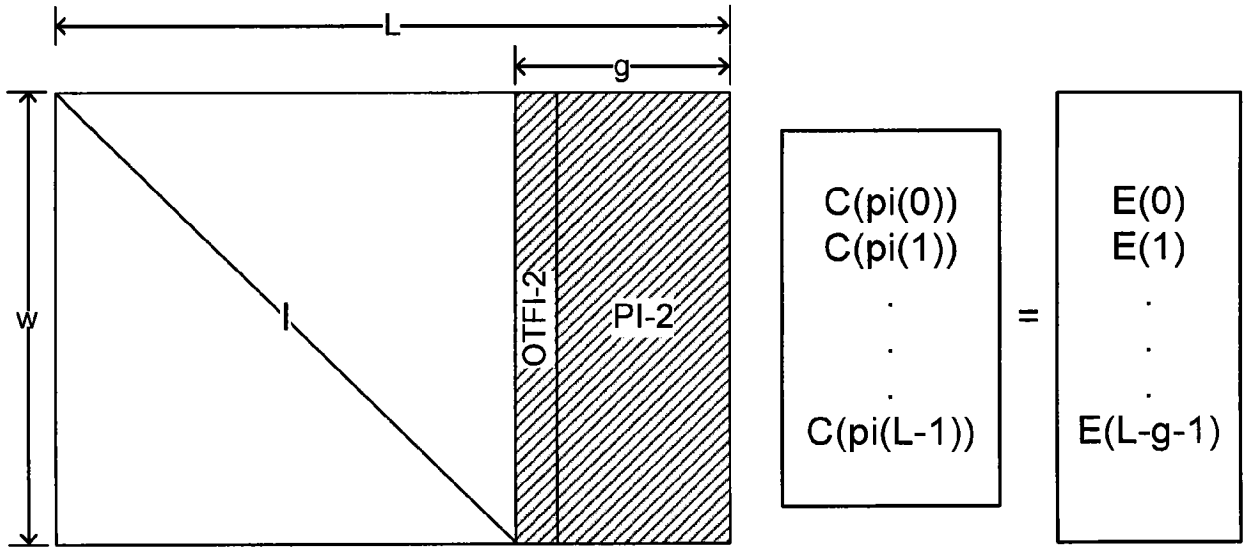

L
g
w
I
OTFI-2
PI-2
C(pi(0))
C(pi(1))
C(pi(L-1))
=
E(0)
E(1)
E(L-g-1)

图 26

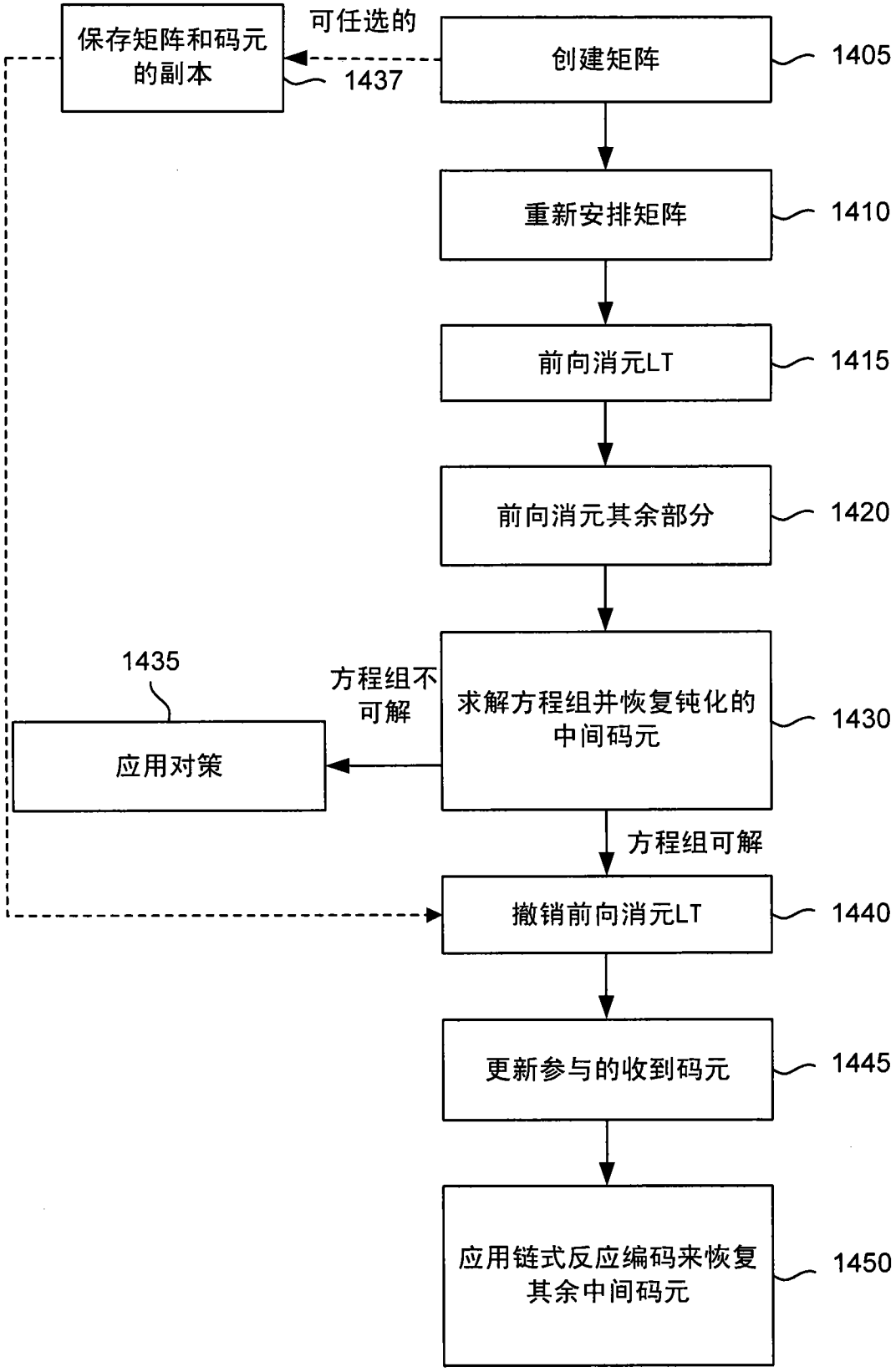

保存矩阵和码元的副本
可任选的
1437
创建矩阵
1405
重新安排矩阵
1410
前向消元LT
1415
前向消元其余部分
1420
1435
方程组不可解
求解方程组并恢复钝化的中间码元
1430
应用对策
方程组可解
撤销前向消元LT
1440
更新参与的收到码元
1445
应用链式反应编码来恢复其余中间码元
1450

图 27

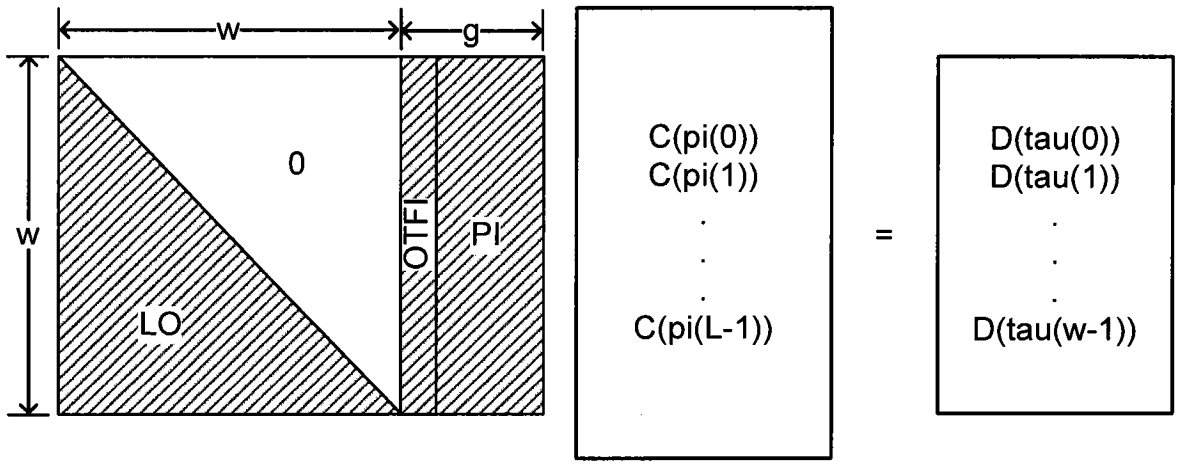

w
g
w
0
LO
OTFI
PI
C(pi(0))
C(pi(1))
C(pi(L-1))
=
D(tau(0))
D(tau(1))
D(tau(w-1))

图 28

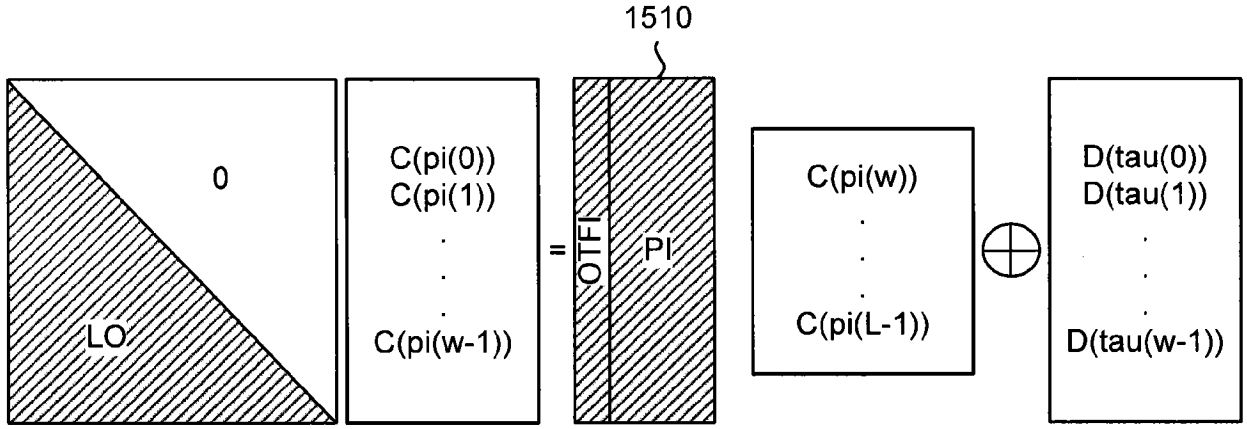

1510
0
LO
C(pi(0))
C(pi(1))
C(pi(w-1))
=
OTFI
PI
C(pi(w))
C(pi(L-1))
D(tau(0))
D(tau(1))
D(tau(w-1))

图 29

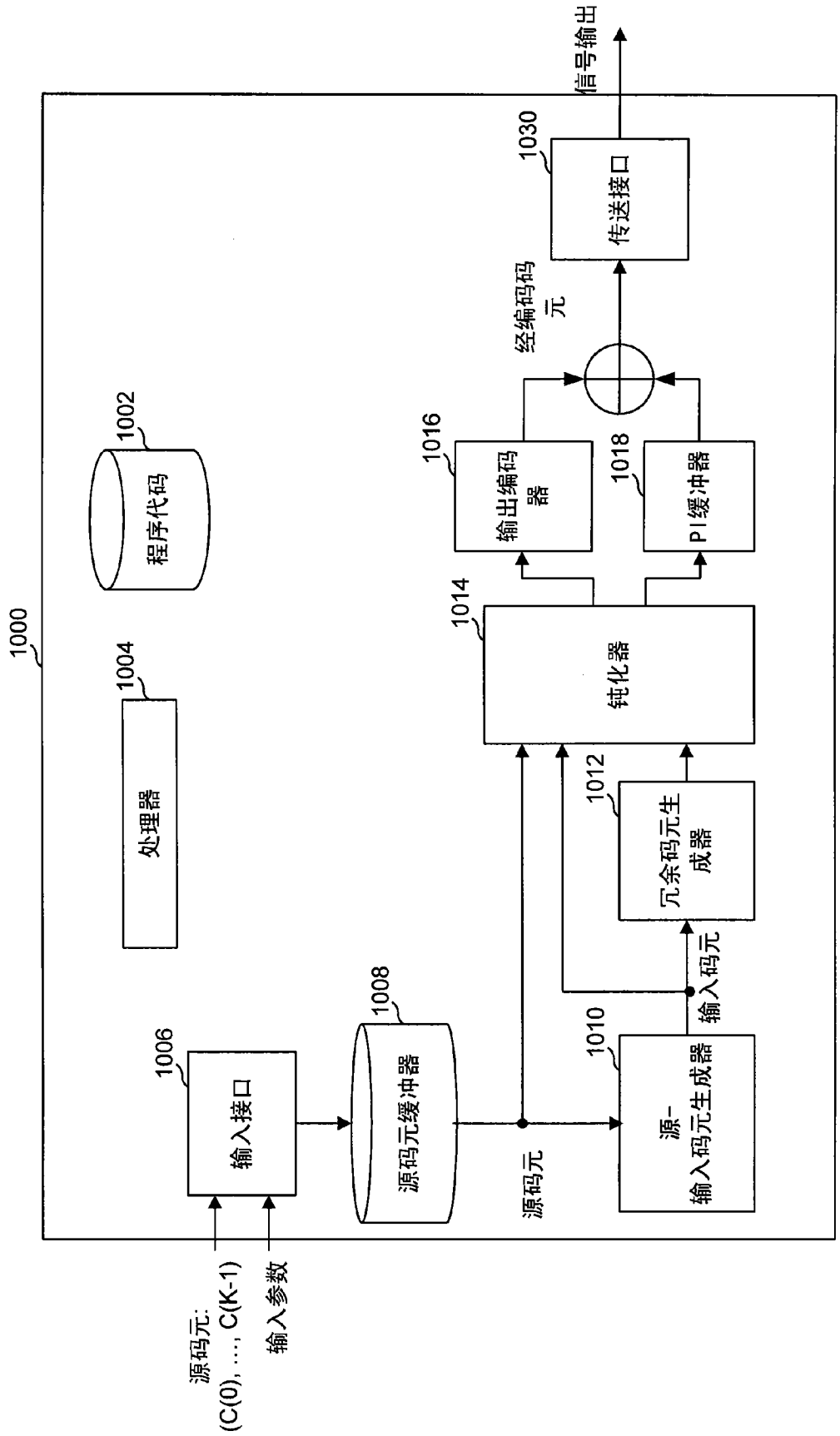

1000
1002
程序代码
1004
处理器
1006
输入接口
1008
源码元缓冲器
1010
源-输入码元生成器
1012
冗余码元生成器
1014
钝化器
1016
输出编码器
1018
PI缓冲器
1030
传送接口
信号输出
经编码码元
源码元
输入码元
源码元:
(C(0), ..., C(K-1)
输入参数

图 30

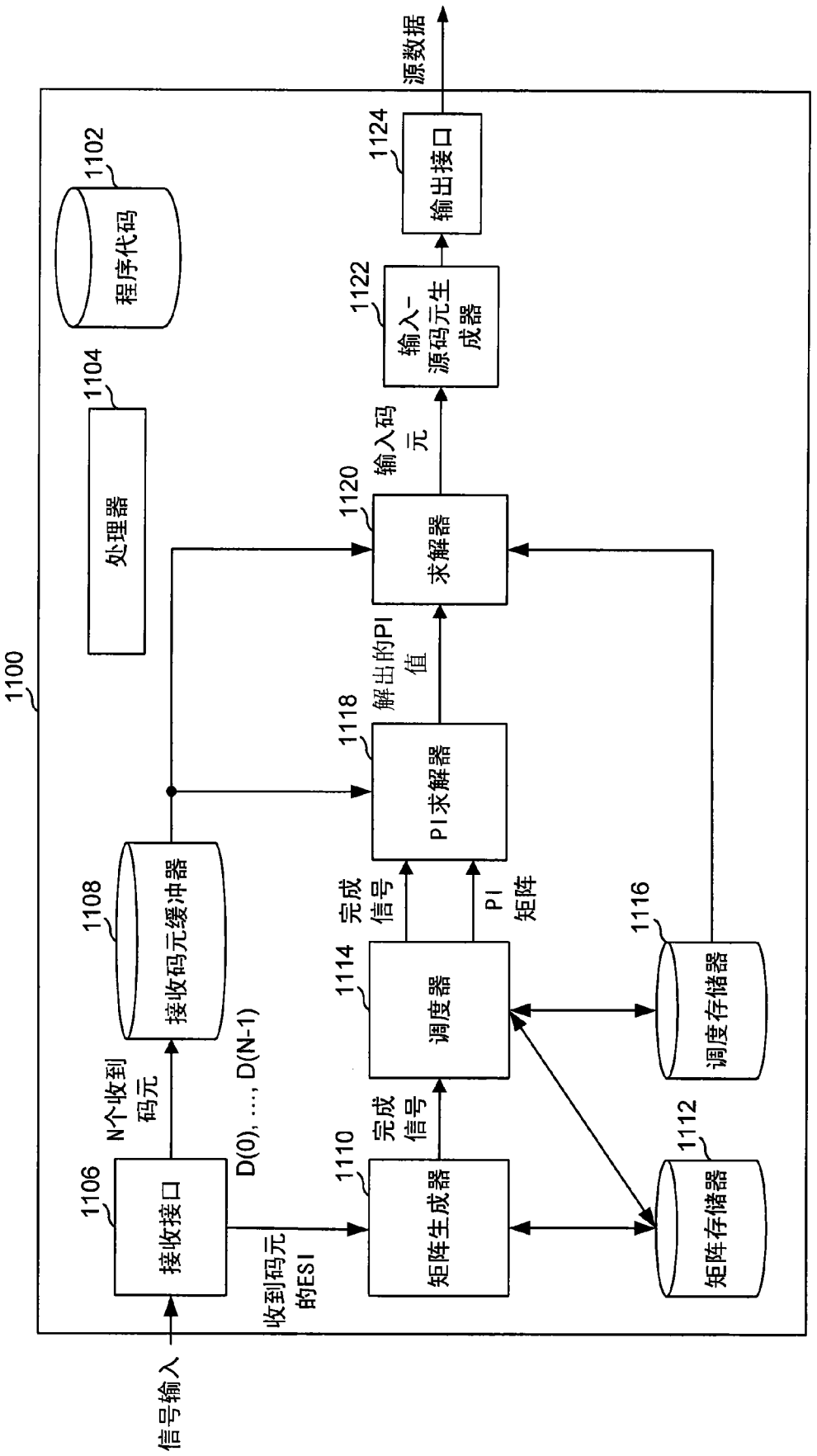

1100
1102
程序代码
1104
处理器
1106
接收接口
1108
接收码元缓冲器
1110
矩阵生成器
1112
矩阵存储器
1114
调度器
1116
调度存储器
1118
PI求解器
1120
求解器
1122
输入-源码元生成器
1124
输出接口
信号输入
N个收到码元
D(0), ..., D(N-1)
收到码元的ESI
完成信号
完成信号
PI矩阵
解出的PI值
输入码元
源数据

图 31