(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 113473128 A
(43)申请公布日 2021.10.01

(21)申请号 202110338203.1

(22)申请日 2021.03.30

(30)优先权数据
2004590.2 2020.03.30 GB

(71)申请人 想象技术有限公司
地址 英国赫特福德郡

(72)发明人 彼得·莱西 S·菲尼

(74)专利代理机构 北京安信方达知识产权代理有限公司 11262
代理人 陆建萍 杨明钊

(51)Int.Cl.
H04N 19/13(2014.01)
H04N 19/186(2014.01)
H04N 19/42(2014.01)

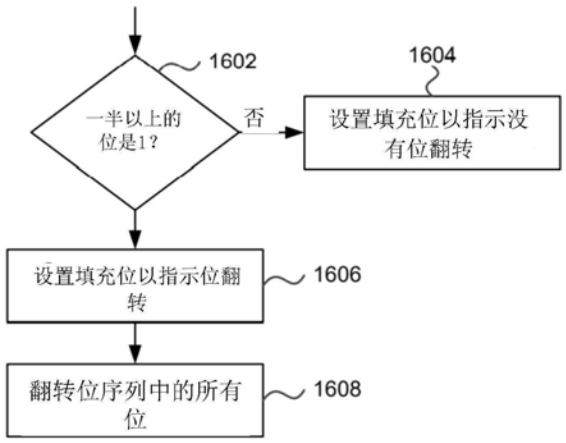
权利要求书2页 说明书33页 附图21页

(54)发明名称

高效编码方法

(57)摘要

描述了一种编码数据值的方法,其中将数据值布置成字,每个字包括多个输入值和一个或多个填充位。通过确定字的一部分中的一半以上的位是否为1来编码字,其中所述部分可包括字中的输入值的位中的一些或全部,以及响应于确定所述部分中的一半以上的位为1,反转所述部分中的所有位且将对应填充位设置为指示所述反转的值。



CN 113473128 A

1.一种编码数据值的方法,所述方法包括:

接收多个输入字,每个输入字包括一个或多个输入值和一个或多个填充位;

将所述输入字中的所述一个或多个输入值的所述位划分成多个部分,所述部分中的至少一个具有所述输入字中的对应填充位,

对于在所述输入字中具有对应填充位的部分中的每一个,确定所述部分中的一半以上的所述位是否具有预定义位值;以及

响应于确定所述部分中的一半以上的所述位为1,通过将所述部分中的所有位反转并且将填充位设置为指示所述反转的值来生成输出字。

2.根据权利要求1所述的方法,还包括:

响应于确定输入字的一部分中的一半以上的所述位具有所述预定义位值,通过不使所述部分中的所述位反转并且通过将填充位设置为指示所述部分中的所述位尚未被反转的值来生成输出字。

3.根据权利要求1所述的方法,其中所述一个或多个输入值包括多个输入值。

4.根据权利要求1所述的方法,其中所述输入字的所述部分包括所述一个或多个输入值。

5.根据权利要求1所述的方法,其中所述多个部分中的每一个具有所述输入字中的对应填充位。

6.根据权利要求1所述的方法,其中所述输入值具有非均匀概率分布。

7.根据权利要求1所述的方法,其中所述输入值具有均匀的概率分布。

8.根据权利要求1所述的方法,还包括:

接收多个初始输入字,每个初始输入字包括一个或多个初始输入值;以及

对所述初始输入字中的初始输入值去相关,从而生成所述输入字的所述输入值,对于所述输入字执行所述确定,即确定所述输入字的一部分中的一半以上的所述位是否具有所述预定义位值。

9.根据权利要求1所述的方法,其中所述预定义位值为1。

10.根据权利要求1所述的方法,其中所述预定义位值为0。

11.一种包括编码硬件块的计算实体,所述编码硬件块包括:

输入端,其被配置为接收多个输入值,每个输入字包括一个或多个输入值以及一个或多个填充位;

硬件逻辑,其被布置成将所述输入字中的所述一个或多个输入值的位划分成多个部分,所述部分中的至少一个部分在所述输入字中具有对应填充位,并且对于所述部分中的每一个部分在所述输入字中具有对应填充位,确定所述部分中的一半以上的所述位是否具有预定义位值,以及响应于确定所述部分中的一半以上的所述位具有预定义位值,通过将所述部分中的所有位进行反转并且将填充位设置为指示所述反转的值来生成输出字;

以及输出端,其用于输出所述输出字。

12.根据权利要求11所述的计算实体,其中所述预定义位值为1。

13.根据权利要求11所述的计算实体,其中所述预定义位值为0。

14.一种解码数据值的方法,所述方法包括:

接收多个输入字,每个输入字包括多个位段和对应于每个段的填充位;以及

对于输入字的每个段：

读取并分析对应填充位的值；

响应于确定所述填充位指示所述段在编码过程期间被翻转，翻转所述段中的所有位并且将所述填充位复位到其默认值；

响应于确定所述填充位指示所述段在所述编码过程期间未翻转，使所述段中的所述位保持不变且将所述填充位复位到其默认值；以及

将所得到的位作为经解码的字输出。

15.一种包括解码硬件块的计算实体，所述解码硬件块包括：

输入端，其被配置为接收多个输入字，每个输入字包括

多个位段和对应于每个段的填充位；

硬件逻辑，其被布置成对于输入字的每个段：

读取并分析对应填充位的值；

响应于确定所述填充位指示所述段在编码过程期间翻转，翻转所述段中的所有位并且将所述填充位复位到其默认值；以及

响应于确定所述填充位指示所述段在所述编码过程期间未翻转，使所述段中的所述位保持不变且将所述填充位复位到其默认值；

以及

输出端，其用于将所得到的位作为经解码的字输出。

16.一种硬件逻辑块，其被配置为执行根据权利要求1-10和14中任一项所述的方法。

17.一种集成电路定义数据集，所述集成电路定义数据集当在集成电路制造系统中处理时配置所述集成电路制造系统以制造如权利要求16所述的硬件逻辑块。

18.一种集成电路制造系统，包括：

非暂时性计算机可读存储介质，其上存储有集成电路的计算机可读描述，所述计算机可读描述对如权利要求16所述的硬件逻辑块进行描述；

布局处理系统，其被配置为处理所述集成电路描述，以便生成体现所述图形处理系统的集成电路的电路布局描述；以及

集成电路生成系统，其被配置为根据所述电路布局描述来制造所述图形处理系统。

19.一种计算机可读代码，其被配置成使得在运行所述代码时执行如权利要求1-10和14中任一项所述的方法。

20.一种计算机可读存储介质，其上编码有权利要求19所述的计算机可读代码。

高效编码方法

背景技术

[0001] 在计算系统中,处理单元(例如CPU或GPU)通常将数据写入外部存储器或从外部存储器读取数据,并且此外部存储器存取消耗大量功率。例如,外部DRAM存取与类似的内部SRAM存取相比可能要消耗50-100倍的功率。对此的一种解决方案是使用总线反转编码。总线反转编码涉及通过添加一个或多个额外总线线路且使用这些额外的一个或多个总线线路来传输指示总线值对应于数据值还是经反转数据值的码而减少所传输数据中的转变的数目。为了确定要通过总线发送哪个(即,数据值或经反转的值),确定当前数据值与下一数据值之间不同的位数,并且如果该位数大于数据值中的总位数的一半,则将在额外总线线路上传输的码设置为1,并且将下一总线值设置为反转的下一数据值。然而,如果不同的位数不大于数据值中的总位数的一半,则将通过附加总线线路发送的码设置为0,并且将下一总线值设置为下一数据值。

[0002] 下面描述的实施方案仅以举例的方式提供,而不构成对解决已知编码(或重新编码)数据方法的任何或所有缺点的实现方式的限制。

发明内容

[0003] 提供本发明内容是为了以简化的形式介绍下文在具体实施方式中进一步描述的一系列概念。本发明内容不旨在标识所要求保护的主题的关键特征或必要特征,也不旨在用于限制所要求保护的主题的范围。

[0004] 描述了一种编码数据值的方法,其中将数据值布置成字,每个字包括多个输入值和一个或多个填充位。通过确定字的一部分中的一半以上的所述位是否为1来编码字,其中所述部分可包括字中的输入值的位中的一些或全部,以及响应于确定所述部分中的一半以上的所述位为1,反转所述部分中的所有位并且将对应填充位设置为指示所述反转的值。

[0005] 第一方面提供了一种编码数据值的方法,所述方法包括:接收多个输入字,每个输入字包括一个或多个输入值和一个或多个填充位;确定输入字的一部分中的一半以上的所述位是否具有预定义位值;以及响应于确定输入字的一部分中的一半以上的所述位为1,通过将所述部分中的所有位反转并且将填充位设置为指示所述反转的值来生成输出字。

[0006] 第二方面提供了一种包括编码硬件块的计算实体,所述编码硬件块包括:输入端,其被配置为接收多个输入值,每个输入字包括一个或多个输入值以及一个或多个填充位;硬件逻辑,其被布置成确定输入字的一部分中的一半以上的所述位是否具有预定义位值,以及响应于确定输入字的一部分中的一半以上的所述位具有预定义位值,通过将所述部分中的所有位反转并且将填充位设置为指示所述反转的值来生成输出字;以及输出端,其用于输出所述输出字。

[0007] 第三方面提供了一种对数据值进行解码的方法,所述方法包括:接收多个输入字,每个输入字包括一个或多个位段和对应于每个段的填充位;以及对于输入字的每个段:读取并分析对应填充位的值;响应于确定所述填充位指示所述段在编码过程期间被翻转,翻转所述段中的所有位并且将所述填充位复位到其默认值;响应于确定所述填充位指示所述

段在所述编码过程期间未翻转，使所述段中的所述位保持不变且将所述填充位复位到其默认值；以及将所得到的位作为经解码的字输出。

[0008] 第四方面提供了一种包括解码硬件块的计算实体，所述解码硬件块包括：输入端，其被配置为接收多个输入字，每个输入字包括一个或多个位段和对应于每一段的填充位；硬件逻辑，其被布置成对于输入字的每个段：读取并分析对应填充位的值；响应于确定所述填充位指示所述段在编码过程期间翻转，翻转所述段中的所有位并且将所述填充位复位到其默认值；以及响应于确定所述填充位指示所述段在所述编码过程期间未翻转，使所述段中的所述位保持不变且将所述填充位复位到其默认值；以及输出端，其用于将所得到的位作为经解码的字输出。

[0009] 被布置为执行如本文所述的方法的硬件逻辑可被体现在集成电路上的硬件中。可以提供一种在集成电路制造系统处制造被布置为执行如本文所述的方法的硬件逻辑(诸如处理器或其部分)的方法。可以提供一种集成电路定义数据集，当在集成电路制造系统中处理时，该集成电路定义数据集配置该系统以制造被布置为执行如本文所述的方法的硬件逻辑(诸如处理器或其部分)。可以提供一种非暂时性计算机可读存储介质，其上存储有集成电路的计算机可读描述，当被处理时，该计算机可读描述使布局处理系统生成在集成电路制造系统中使用的电路布局描述，以制造被布置为执行如本文所述的方法的硬件逻辑(诸如处理器或其部分)。

[0010] 可以提供一种集成电路制造系统，包括：非暂时性计算机可读存储介质，其上存储有计算机可读集成电路描述，其描述被布置为执行如本文所述的方法的硬件逻辑(诸如处理器或其部分)；布局处理系统，其被配置为处理集成电路描述，以便生成体现被布置为执行如本文所述的方法的硬件逻辑(诸如处理器或其部分)的集成电路的电路布局描述；以及集成电路生成系统，其被配置为制造被布置成执行如本文根据电路布局描述所描述的方法的硬件逻辑(诸如处理器或其部分)。

[0011] 可提供用于执行本文中描述的任一方法的计算机程序码。可提供非暂时性计算机可读存储介质，在其上存储有计算机可读指令，所述计算机可读指令在计算机系统处执行时使所述计算机系统执行本文中描述的任何方法。

[0012] 如对本领域的技术人员显而易见的，上述特征可以适当地组合，并且可以与本文所述的示例的任何方面组合。

附图说明

[0013] 现在将参考附图详细描述示例，在附图中：

[0014] 图1是对数据进行功率高效编码的第一示例方法的流程图；

[0015] 图2是示出图1的方法的映射操作的示例实现的流程图；

[0016] 图3示出可以用于实现图2的映射操作的示例LUT；

[0017] 图4是示出图2的方法的映射操作的另一示例实现的流程图；

[0018] 图5A和5B是示出标识一组预定义码的子集的示例方法的流程图；

[0019] 图6A和6B是示出生成码的两种示例方法的流程图；

[0020] 图7示出图6的方法的替代表示；

[0021] 图8A和8B是示出图2的方法的映射操作的另外的示例实现的流程图；

[0022] 图9是示出图1的方法的映射操作的另一示例实现的流程图；

[0023] 图10A示出示例差值的概率分布图；

[0024] 图10B示出示例移位参考值的概率分布图；

[0025] 图11A示出符号重新映射的示例差值的概率分布图；

[0026] 图11B示出符号重新映射的示例移位参考值的概率分布图；

[0027] 图12示出对于P的各种值，映射到具有P位填充的前2^{L}N位二项式码的一组均匀随机L位输入值(其中L＝10)的平均汉明权的图；

[0028] 图13是示出图1的方法的映射操作的另一示例实现的流程图；

[0029] 图14A和14B示出可以用来实现图2的映射操作的两个其他示例逻辑阵列；

[0030] 图15A和15B是示出标识一组预定义码的子集的另外的示例方法的流程图；

[0031] 图16是对数据进行功率高效编码的第二示例方法的流程图；

[0032] 图17是对数据进行功率高效编码的第三示例方法的流程图；

[0033] 图18和19示出包括被配置为执行这里描述的方法之一的硬件逻辑的两个示例硬件实现；

[0034] 图20示出其中实现了被布置为执行如本文所述的方法的硬件逻辑(诸如处理器或其部分)的计算机系统；以及

[0035] 图21示出用于生成包含硬件逻辑(诸如处理器或其部分)的集成电路制造系统，该硬件逻辑被布置为执行如本文所述的方法；

[0036] 图22是示出生成码的另一示例方法的流程图；

[0037] 图23是示出生成码的又一示例方法的流程图；以及

[0038] 图24是示出当对数据进行解码时使用的示例映射方法的流程图。

[0039] 附图示出了各种示例。技术人员将理解，附图中所示的元件边界(例如，框、框的组，或其他形状)表示边界的一个示例。在一些示例中，情况可能是一个元件可以被设计为多个元件，或者多个元件可以被设计为一个元件。在适当的情况下，贯穿各附图使用共同附图标记来指示相似特征。

具体实施方式

[0040] 通过示例的方式给出以下描述，以使本领域的技术人员能够制造和使用本发明。本发明不限于本文中描述的实施方案，并且对所公开的实施方案的各种修改对于所属领域的技术人员而言将是显而易见的。

[0041] 现在仅通过示例的方式来描述实施方案。

[0042] 如上所述，外部存储器存取消耗大量功率，因此可能是计算系统的功率预算的很大一部分。该功耗至少部分地是数据在其上行进的总线的电容的结果，并且这意味着改变状态比维持状态消耗更多的功率。这是已知的总线反转编码方法背后的基本原理，所述方法寻求减少所传输数据中(即，在一个传输数据值和下一个传输数据值之间)的转换数目。然而，如上所述，所述方法需要一个或多个附加总线线路，并且还需要额外的硬件，例如专用存储器(例如，具有在存储所接收的数据之前可以反转任何位反转的硬件)和CPU/GPU中的附加编码器/解码器。此外，随着每次比较的位的数目增加(以便确定是发送所述位还是其经反转的值)，总线反转编码的总效率显著降低。

[0043] 本文描述了对数据值进行功率高效编码的各种替代方法。除了降低当通过外部(即,片外)总线(例如,向外部存储器或另一外部模块,诸如显示控制器)传输数据时消耗的功率之外,通过使用本文描述的方法,还可以降低当通过内部总线(同时比外部总线低得多)传输时消耗的功率。所述方法可另外减少在存储数据(例如,在芯片上高速缓冲存储器或外部存储器中)时消耗的功率,尤其在其中存储装置在存储0时相对于存储一时消耗较少功率的实施方案中。因此,这些方法特别适合于功率受限的应用和环境,例如移动或其他电池供电的设备。

[0044] 与已知的总线反转编码方法不同,这里描述的方法不需要附加的总线线路。此外,本文描述的许多方法可以与非专用存储器一起使用,因为即使当随后随机访问数据时,数据也可以以其有效编码的形式存储。特别地,在所得到的码是固定长度的情况下(例如,在编码长度与初始数据值长度匹配的情况下),存储器不需要被专门化。使用固定长度的码使得随机访问变得简单,因为数据可以直接被索引。即使在所得到的码不是固定长度的情况下,在它们是特定单元(例如半字节)大小的倍数的情况下),也不需要专用存储器,而只需要可以以给定粒度(例如半字节步幅)读/写的存储器。

[0045] 图1是对数据进行功率高效编码的第一示例方法的流程图。所述方法包括接收输入值(框102),基于输入值的概率分布将每个输入值映射到一组码中的一者(框104),以及输出与所接收的输入值相对应的码(框106)。

[0046] 输入数据(即,在框102中接收的输入值)可以是任何类型的数据,并且输入值可以具有任何位宽(例如,4、8、10、16位或包括偶数位的其他位宽和/或包括奇数位的位宽)。在各种示例中,数据可以是具有相关联的非均匀概率分布的数据(并且因此可以以有意义的方式通过其概率来排序)。相关的非均匀概率分布不需要完全精确到其实际概率分布。在其他示例中,数据可以是具有均匀随机的相关联概率分布的数据,或者是不具有概率分布的数据,并且由于这些在实践中是等效的,因此它们被等同地对待。因此,在下面的描述中,短语“没有概率分布的数据”和“具有均匀随机概率分布的数据”可互换使用。

[0047] 在各种示例中,输入数据可以是图形数据(例如,像素数据)、音频数据、工业传感器数据或纠错码(ECC)。在输入数据是图形数据的各种示例中,输入值可以是颜色通道的扫描线数据,例如RGB、RGBX、RGBA、YUV、平面Y、平面U、平面V或UVUV或诸如帧缓冲器的内容或高度/深度/法线映射的其他像素数据。

[0048] 在本文所述的许多示例中,输入值是表示字符的无符号值或无符号码;然而,输入值可以可选地是其他数据类型(例如,带符号或浮点值)。在输入值不是无符号值的情况下,可能需要考虑不同的概率分布,并且可能需要相应地修改所述方法的各个方面(例如,去相关和概率排序)。例如,带符号值像去相关的无符号值那样分布在0周围,因此可以简单地进行符号重新映射,并且浮点值将通常类似于无符号或带符号的定点值而分布(取决于它们表示什么)(例如,均匀地分布在中间值周围),但是不同地编码,因此(在适当的去相关/移位之后)需要不同的符号重新映射(例如,将符号位从MSB移动到LSB)。下文更详细地描述此情况。

[0049] 在各种示例中,一组预定义码中的码可各自包括与输入值相同数目的位,并且出于以下描述的目的,N是输出码的位长度,而L是输入值的位长度。然而,在其他示例中,这些码中的一些或全部可包括比输入值多的位(即,N>L)。在各种示例中,该码集可包括多个子

集,其中每个码子集包括不同位长度的码。例如,该码集可包括包含10位码的第一子集和包含12位码的第二子集。如下所述,基于码的特性,例如基于码中的1的数目(即,码的汉明权HW)或码内的位翻转的数目(即,表示码的位序列内的1-0或0-1转变的数目),可将具有相同位长度的码的子集各自进一步划分成较小子集。

[0050] 在各种示例中,可将一个或多个输入值分组成任何大小(例如,8、16、32个位)的数据字,包含0个、一个或多个填充位。例如,在输入数据是像素数据的情况下,输入数据可以包括数据字,每个数据字包括3或4个分别为10位长度或8位长度的输入值(例如,对于YUV/RGB或RGBX/RGBA数据)。在这样的示例中,数据字中的输入值可以各自被分别映射到一组码中的一者,并且然后可以组合所得码以形成输出数据字(例如,包括三个或四个级联码)。在各种示例中,除了多个输入值之外,数据字可包含一或多个填充位,例如,10位和8位数据值可被包装成分别包含三个输入值以及2或8个填充位的32位数据字。如下文更详细地描述,当执行本文描述的编码方法时,填充位可保持不变(例如,其可在输入值的映射之前从输入数据字移除且接着在组合所得码时包含于输出数据字中),或填充位中的一个或多个(且在一些示例中,所有填充位)可用于允许数据字中的输入值中的一个或多个的较长(且更有效)码(其中N>L)。

[0051] 可以以许多不同的方式执行映射(在框104中)。在各种示例中,所述映射可使用将可能输入值映射到来自一组预定义码的码的预先计算的LUT。LUT可以基于输入值的概率分布来预先计算,其中这是预先已知的(例如,对于字母字符)。LUT对于较短的输入值(例如,包括多达最大约10位的输入值)比对于较长的输入值(例如,包括32位的输入值)更适合,因为否则需要多个门来实现LUT,并且以另一种方式执行映射可能更有效(例如,就硅面积或功率而言),并且下面描述其他方法。

[0052] 术语“逻辑阵列”在这里用于指被配置为将一组输入二进制码(例如输入值)映射到一组输出二进制码(例如来自一组预定义码的码)的门的布局。术语“数据阵列”在这里用来指由输入值索引的二进制结果的阵列。逻辑阵列和数据阵列都是二进制映射的实现,并且它们的使用可以是可互换的(例如,在这里描述了数据阵列的使用的情况下,可以替换地使用逻辑阵列,反之亦然)。通常,逻辑阵列是更加以硬件为中心的解决方案,而数据阵列适合于以硬件为中心的解决方案或以软件为中心的解决方案。术语“查找表”(LUT)在这里用于指逻辑阵列或数据阵列。

[0053] 图2是示出图1的方法的映射操作(框104)的另一示例实现的流程图。在该示例中,(框104的)映射操作包括两个阶段。首先,基于输入值的概率分布,即基于特定输入值出现或预期出现的频率,为输入值确定概率索引(框202)。这个操作可以被称为“概率排序”。如下所述,根据数据的类型,概率索引可能不能准确地将输入值排序为概率递减的顺序;然而,这确实导致输入值的排序,使得它们从最可能的值(具有更接近于0的索引)到最不可能的值(具有较大索引)近似排序。排序中的任何误差降低了通过使用所述方法而获得的效率增益,但是不阻止所述方法操作并提供益处。对于具有均匀随机概率分布的输入值(即,其中2^L个输入值中的每一个具有概率2^{-L})并且其中N=L,仍然可以使用所述方法;然而,平均汉明权将不会减小。

[0054] 在确定了输入值的概率索引(在框202中)之后,基于存在于预定义码中的1的数目或位翻转的数目,将概率索引(以及因此的输入值)映射到一组预定义码中的一者(框204),

然后输出所得到的码(即,由映射操作标识的码)。根据该映射操作,具有较低概率索引(并且因此具有较高发生概率)的那些输入值可以被映射到来自一组预定义码的具有较少的1的那些码。或者,代替基于码中的1的数目(即,基于码的HW)分配码,可将具有较低概率索引(且因此较高发生概率)的输入值映射到来自预定义码集合的具有较少位翻转的那些码。在本文所述的方法中,在映射(在框204中)中使用的位翻转的数目是指当被认为是位序列时(而不是特定码与另一码之间的汉明距离)特定码内的位翻转。

[0055] 虽然图2中未示出,但是映射(在框104中)还可以包括在概率排序之前的预处理操作(在框202中)和/或在映射之后的后处理操作(在框204中)。这些附加的、可选的操作在下面描述。

[0056] 概率索引x可以用许多不同的方式确定(在框202中),并且如上所述,概率索引的目标是如果p_x是输入值被映射到x的概率,则$p_0 \geqslant p_1 \geqslant p_2 \geqslant p_3 \ldots$,尽管如上所述,在各种示例中,概率索引的确定可以仅是对由数据的实际概率分布给出的排序的近似。例如,最常见的k个输入值可以映射到索引0、…、k-1,并且所有其他输入值可以单射地映射到任何其他索引(以保证可逆性)。

[0057] 在各种示例中,可使用输入值与概率索引之间的LUT(例如,预先计算的逻辑阵列)来确定概率索引。LUT可以基于输入值的概率分布来预先计算,其中这是预先已知的(例如,对于字母字符)。LUT对于较短的输入值(例如,包括多达最大约10位的输入值)比对于较长的输入值(例如,包括32位的输入值)更适合,因为否则需要多个门来实现逻辑阵列,并且以另一种方式(例如,使用迭代算法)生成概率索引可能更有效(例如,在硅面积或功率方面)。

[0058] 虽然使用LUT来确定概率索引对于输入值的类型或格式是不可知的,但是在其他示例中(例如,其中通过变换输入值本身来确定输入值的概率索引),确定概率索引的方法(在框202中)可以至少部分地取决于输入值的类型或格式(例如,带符号还是无符号、浮点还是定点等)和/或输入值的概率分布和/或生成输入值的方式。例如,在输入值关于0近似对称分布,概率分布的峰值在0或接近0的情况下,则可以通过对输入值应用符号重新映射来确定概率索引。在输入值关于另一值近似对称地分布(使得概率分布的峰值处于该另一值)的示例中,则在将符号重新映射应用于移位后的输入值之前,可以首先对输入值进行移位(使得它们关于0近似对称地分布)。用于定点值的符号重新映射包括将输入值向左移位一位位置(其涉及添加0作为新的最低有效位LSB)、移除最高有效位(MSB)且接着对所有剩余位与刚移除的MSB进行异或。在(带符号)浮点值的情况下,符号重新映射包括将符号位从MSB移动到LSB:对于(带符号)浮点格式,首先正值被递增排序,随后负值被递减排序,因此将符号位移动到LSB使这些值交错且通过其与0的距离对所述值进行排序。如果对浮点输入值使用下面描述的去相关操作(参考图9),则除了将符号位从MSB移位到LSB之外,其余位还必须由符号位进行异或以便撤消初始的异或。这样,输入值和结果概率索引包括相同数目的位。在其他示例中,可以产生具有与输入值不同数目个位的索引的方式产生概率索引,例如,其中一些数目个0填充概率索引的MSB末尾。

[0059] 在另一个示例中,在输入值是使用无损编码方法(例如霍夫曼编码(Huffman encoding))产生的并且输入值具有可变长度的情况下,概率索引可以基于输入值的长度来确定(在框202中)(例如最短的输入值是最可能的并且因此被分配较低的概率索引,并且最长的输入值是最不可能的并且因此被分配较高的概率索引)。在使用无损编码方法生成输

入值但是通过添加尾部(例如,后面跟随着无、一个或多个0的尾部)将编码值填充到固定位长以生成输入值的另一示例中,可以基于输入值的尾部的长度(例如,具有最长尾部的输入值是最可能的并且因此被分配较低概率索引,并且具有最短尾部的输入值是最不可能的并且因此被分配较高概率索引)来确定概率索引(在框202中)。下面描述涉及霍夫曼编码的详细示例。

[0060] 在另一示例中,可以使用概率分布构建器来确定(在框202中)概率索引,该概率分布构建器累积每个可能值的频率并且按它们对值排序。这种方法可以用在概率分布事先不知道的情况下。对于编码和解码操作,将以相同的方式生成分布,以确保正确的解码。在第一示例实现中,使用第一X输入值(其中X是整数)来生成分布(由概率分布构建器),然后将该分布用于所有剩余数据(同时保持固定)。

[0061] X的值可以取决于输入位的数目L,并且可以更适合于较小的输入,其中明确地存储分布将不需要过多的存储。例如,对于L=4,存在16个可能的值,并且因此所存储的分布是针对这16个可能的输入。因此,可以选择X的值,使得X>>16,例如X=256,以确保所生成的分布提供对实际分布的良好近似,并且在这种情况下,所生成的概率分布可以包括针对每个输入的4位频率,总共64位。更一般地,在L位输入和各输入的F位频率中,为了存储随着L呈索引函数性增加的分布,需要2^L*F位的合计。在这种情况下,X可以选择为$2^L*2^F=2^{(L+F)}$的数量级,但在分布特别偏斜的情况下,X的值越小越好(该偏斜预先已知)。

[0062] 在第二示例实现中,分布是完全动态的并且由概率分布构建器连续更新,例如,可以针对每个输入值、字或字块更新分布,其中在溢出点处(例如,在任何频率计数达到可以以F位频率存储的最大值的点处),在继续之前按比例缩小(例如,通过除以二)所有频率。在其他示例中,可以不同地处理溢出情况(例如,通过将频率计数钳位在最大值处;然而,这将导致比按比例缩小频率更不精确的概率分布)。

[0063] 每当需要对输入进行编码时(例如,对于动态分布),可以从分布推断概率索引,或者替代地,可以预先参考分布一次,以生成每个输入的概率索引并将其存储在预先计算的LUT中(例如,对于静态分布)。这可以与下面参考图3描述的LUT组合。在各种示例中,概率分布构建器可以用“最佳猜测”分布来播种,使得它更快地收敛。这可以提供更有效的硬件实现,例如,用于其中数据随着时间显著改变的动态分布。

[0064] 在可能特别适合于大值L(其中先前描述的实现可能不是最优的)的又一示例实现中,可以实现简化的概率排序,使得具有超过预定义阈值的频率的所有输入被标识,并且这些输入被分配给最低概率索引,并且所有其他输入被单射分配给剩余概率索引。

[0065] 在动态地生成概率分布的示例中,指示输入值如何与概率索引相关的数据可被存储和重用以使得数据能够被随后解码。或者,可以以相同的方式在编码和解码两者时生成概率分布。对于静态(即已知)分布(例如,其从先前的观察中已知),在去相关和概率排序操作(例如,移位和符号重新映射)的功能中隐式地编码分布,并且编码器和解码器可以使用对应的逆映射。

[0066] 可以以许多不同的方式执行映射(在框204中)。在各种示例中,所述映射可使用在概率索引与一组预定义码之间映射的LUT(例如,预先计算的逻辑阵列),并且下文参考图3描述示例。该LUT仅包括10位码,并且这可能是因为一组预定义码仅包括10位码,或者因为为每个码子集提供了单独的LUT,其中每个子集对应于一个位长度并且包括具有特定位长

度的码。然而，将了解，在其他示例中，单个LUT可包含一组预定义码中的所有码(例如，按位长度分组)。

[0067] 图3中所示的示例LUT 300包括10位概率索引和10位码，并且可以用于长度不超过10位的输入值(使得映射可以保持可逆)。描述了用于映射操作(在框204中)的该示例和后续方法，其中基于预定义码中存在的1的数目，将概率索引(以及因此的输入值)映射到一组预定义码中的一者。随后描述对其中基于预定义码中存在的位翻转的数目来替代地执行映射的方法的修改。

[0068] 如图3所示，10位码的组(其可以是整个一组预定义码或其子集)被细分为多个子集301-311，每个子集包括一个或多个包含相同数目的1(即具有相同汉明权)的码。在每个子集内，具有相同HW的码可以以任何方式排序，并且在所示的示例中，它们按字典排序。如图3所示，虽然在一些情况下，10位二进制形式的概率索引与10位码匹配，但是在大多数情况下，两者是不同的。

[0069] 在其他示例中，如图4所示，映射可通过首先基于概率索引来标识码子集(框402)并随后选择所标识的码子集中的一个码来执行，其中这些子集是图3中所示并在以上描述的那些子集(即，每个子集对应于不同的HW并且仅包括来自具有特定HW的码集的那些码)(框404)。假定在子集与汉明权之间存在一对一的关系，则图4的方法可被替换地描述为计算码的HW(在框402中)并随后用所计算的HW来标识码(在框404中)。

[0070] 例如，可以通过从概率索引x中迭代地减去二项式系数来标识码的子集(在框402中)$\binom{N}{r}$的两倍来识别码的子集(在方框402中)，其中N是码中的位的数目，并且初始地r＝0。如图5A所示，概率索引x初始地与r＝0的二项式系数比较(框502)，该系数等于1(不考虑N的值)。如果概率索引严格小于1(框502中的“是”)，即它是0，则选择汉明权R为0的第一子集301(框504)。否则，从概率索引中减去二项式系数的值(即，在该第一次迭代中为1)，并且将r的值加1(框506)。在随后的迭代中，将来自前一迭代的更新后的概率索引(即，更新后的x值)与二项式系数和r的当前值进行比较(在框502中)，并且如果更新后的概率索引x严格小于二项式系数(框502中的“是”)，则选择HW等于r的当前值的子集，即，其中R＝r(框504)。然而，如果更新的概率索引x不严格小于二项式系数(框502中的“否”)，则从更新的概率索引中减去二项式系数的值(具有r的当前值)，并且r的值加1(框506)。在选择子集(在框502和506中)中使用的二项式系数的值可以被计算或者可以从包含预先生成的值的列表的LUT中获得。

[0071] 下面示出了二项式系数的示例LUT，其中列对应于从0到12的N(或N，n于计算$\binom{n}{r}$的后续方法)的不同值，而行对应于从0到5的r的不同值。该LUT被设计成在L＝10和N＝12的情况下使用；然而，它们也可以用于任何更小的L和/或N值)。实际上，所使用的LUT可省略以下列或行中的一个或多个，其中这些列或行不被使用(例如，其中N＜12)，并且LUT可以以任何格式存储，该格式可使用适当的索引(例如，基于n和r的值)来访问，例如按行或按列。

[0072]

r \ N	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	2	3	4	5	6	7	8	9	10	11	12
2	0	0	1	3	6	10	15	21	28	36	45	55	66
3	0	0	0	1	4	10	20	35	56	84	120	165	220
4	0	0	0	0	1	5	15	35	70	126	210	330	495
5	0	0	0	0	0	1	6	21	56	126	252	462	792

[0073] 此外，通过注意到对于所有$\mathrm{N}\binom{N}{0}=1$，可省略第一行(例如，因为对于宽LUT，节省用于在LUT中存储行的空间比执行对索引r的比较所需的额外逻辑更显著)且通过注意到对于所有$\mathrm{N}\binom{N}{1}=N$，可省略第二行。另外(或代替)作为对于$r>1$，$\binom{0}{r}=\binom{1}{r}=0$，可以省略前两列。省略前两行和列导致更小的LUT：

[0074]

r \ N	2	3	4	5	6	7	8	9	10	11	12
2	1	3	6	10	15	21	28	36	45	55	66
3	0	1	4	10	20	35	56	84	120	165	220
4	0	0	1	5	15	35	70	126	210	330	495
5	0	0	0	1	6	21	56	126	252	462	792

[0075] 在各种实例中，如果LUT的最后一列存储累加值$\sum_{r=0}^{R}\binom{12}{r}$而不是$\binom{12}{r}$，则可简化用于实现本文所描述的方法的硬件逻辑，如图5B中所示，从而得到如下LUT：

[0076]

r \ N	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	2	3	4	5	6	7	8	9	10	11	13
2	0	0	1	3	6	10	15	21	28	36	45	55	79
3	0	0	0	1	4	10	20	35	56	84	120	165	299
4	0	0	0	0	1	5	15	35	70	126	210	330	794
5	0	0	0	0	0	1	6	21	56	126	252	462	1586

[0077] 或者，在前两行和列都被移除的情况下：

[0078]

r \ N	2	3	4	5	6	7	8	9	10	11	12
2	1	3	6	10	15	21	28	36	45	55	79
3	0	1	4	10	20	35	56	84	120	165	299
4	0	0	1	5	15	35	70	126	210	330	794
5	0	0	0	1	6	21	56	126	252	462	1586

[0079] 上述后两个示例LUT(即,在最后一列中具有累积系数)可以特别适合于使用N的单个值(例如,在所示示例中N＝12)的实现,因为它们然后可以与实现图5B的方法的逻辑一起使用。尽管这只使用LUT的最后一列(即,累积值),但是其他列仍然可以包括在LUT中,因为它们可以在后续阶段中使用,即,从子集中标识特定码(在框404中,这使用所有n＜N的正常二项式系数)。或者,最后一列可作为单独LUT存储。

[0080] 在使用N的多个值(例如,N＝10和N＝12)的实现中,使用上述两个前面的示例LUT(即,在所有列中存储正常二项式系数,而不是在LUT的任何列中存储累积二项式系数)并且在两种情况下使用图5A的逻辑可能更有效,因为码的标识(在框404中)使用所有n<N的正常二项式系数,否则需要实现用于N＝12的图5B的逻辑和用于N＝10的图5A的逻辑。

[0081] 如图5B所示,通过使用累积值(在框512的比较中),循环中的更新步骤(框516)仅包括r值的增量(而不是如图5A的框506中那样还更新概率索引的值),并且存在计算最终的、更新的概率索引x值的单个减法操作(框514)。此减法运算(在框514中)从初始概率索引减去可从LUT读取的累加值$\sum_{i=0}^{i=r-1}\binom{N}{i}$,以在单个步骤中产生概率索引的最终更新版本。尽管图5B包括图5A中的流程图的一个附加框,但是由于将循环简化为单个增量(框516中),所以减少了处理量(以及因此的延迟),这在硅面积方面展开的成本也更低。

[0082] 从上述LUT(其也可以被称为“二进制系数表”)访问值可以如以下伪码中所示地实现:

[0083]
```
无符号 nCr (无符号 n、无符号 r)
{
 如果(r=0)返回 1;
 如果(r=1)返回 n;
 如果(n<2)返回 0;

 //二进制数系数表
 无符号 coeffs[4][11] =
 {
 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 79,
 0, 1, 4, 10, 20, 35, 56, 84, 120, 165, 299,
 0, 0, 1, 5, 15, 35, 70, 126, 210, 330, 794,
 0, 0, 0, 1, 6, 21, 56, 126, 252, 462, 1586
 };
 返回 bin_coeffs[r - 2][n - 2];
}
```

[0084] 在其他示例中,可以使用LUT来标识(在框402中)码的子集,该LUT将概率索引映射到子集,或者映射到HW,HW本身标识针对特定码长度的子集,并且下面示出N=10的示例。

[0085]

最小索引	最大索引	汉明权R
0	0	0
1	10	1
11	55	2
56	175	3
176	385	4
386	637	5
638	847	6
848	967	7
968	1012	8
1013	1022	9
1023	1023	10

[0086] 例如,在N=10且x=10的情况下,选择具有R=1的子集,并且在N=10且x=37的情况下,选择具有R=2的子集。

[0087] 代替使用上述LUT,可以使用包括累积二项式系数的较早LUT的右手列。上述“最大索引”比来自较早LUT的其相应的累积二项式系数小一。在这种情况下,代替确定该索引是否小于或等于最大索引,改变比较以确定该索引是否严格小于累积二项式系数。

[0088] 在例如使用上述方法之一标识了子集(在方框402中)之后,被确定为子集的标识的一部分的信息可以被用于从所标识的子集中选择码(在方框404中)。例如,在使用图5A或5B的方法的情况下,概率索引x的最终更新值可用于在标识的子集中选择码中的一个,因为

它提供了要在子集内选择的码的相对位置(例如,通过选择子集中的第(x+1)个码)。在使用上述LUT的情况下,尚未计算概率索引的最终更新值(在框402中);然而,这可以通过从概率索引中减去所标识的子集的最小索引(在与图5B的框514中的单次减法相同的操作中)来获得,然后用作要在所标识的子集内选择的码的相对位置的指示符(在框404中)。返回参考前面的示例,其中N=10并且概率索引的初始值为10(即,初始地x=10),概率索引x=10-1=9的最终更新值并且因此输入值被映射到具有R=1的子集中的位置9中的码(假设子集中的第一码在位置0中)。类似地,在N=10并且概率索引的初始值为37(即,初始地x=37)的情况下,概率索引x的最终更新值=37-11=26,因此输入值被映射到具有R=2的子集中的位置26中的码。

[0089] 每个子集内的码可存储在LUT中,并基于概率索引x的最终更新值来选择,或者可替代地,码可使用概率索引x的最终更新值在迭代过程中一次一位地生成。图6A中所示的这种迭代方法可以特别适合于N的值,其中码的数目足够大,使得将它们全部存储在LUT中是低效或不切实际的,然而它也可以用于N的其他较小值(即,用于N的任何值)。如图6A所示,迭代方法使用两个迭代值n和r,并且初始地n=N(其中如上所述,N是码的位长度)和r=R(其中如上所述,R是所标识的子集的HW),并且在每次迭代开始时,n递减1(框602),使得正好存在N次迭代。在每次迭代中,将更新的概率索引x与具有当前n和r值的二项式系数$\binom{n}{r}$进行比较(框604)。如果更新的概率索引x大于或等于二项式系数$\binom{n}{r}$(在方框604中的“是”),则将1附加到码,从概率索引x中减去二项式系数$\binom{n}{r}$的值,并且r递减1(框606)。然而,如果更新的概率索引x不大于或等于二项式系数$\binom{n}{r}$(在方框604中为“否”),则将0附加到码,并且x和r的值不变(方框608)。当n=0时(框610中的“是”),所述方法停止。

[0090] 前面两个示例可以用于说明图6A的操作。在以上第一示例中,N=10并且概率索引的初始值是10(即x=10),并且因此R=1并且当前子集的概率索引的起始值x=9。

[0091]

	x	n	r	$\binom{n}{r}$	$x \geq \binom{n}{r}$?	码
第 1 次迭代	9	9	1	9	是	1

[0092]

第 2 次迭代	0	8	0	1	否	10
第 3 次迭代	0	7	0	1	否	100
第 4 次迭代	0	6	0	1	否	1000
第 5 次迭代	0	5	0	1	否	10000
第 6 次迭代	0	4	0	1	否	100000
第 7 次迭代	0	3	0	1	否	1000000
第 8 次迭代	0	2	0	1	否	10000000
第 9 次迭代	0	1	0	1	否	100000000
第 10 次迭代	0	0	0	1	否	1000000000

[0093] 在以上第二示例中,N＝10并且概率索引的初始值是37(即x＝37),并且因此R＝2并且当前子集的概率索引的起始值x＝26。

[0094]

	x	n	r	$\binom{n}{r}$	$x \geqslant \binom{n}{r}$?	码
第 1 次迭代	26	9	2	36	否	0
第 2 次迭代	26	8	2	28	否	00
第 3 次迭代	26	7	2	21	是	001
第 4 次迭代	5	6	1	6	否	0010
第 5 次迭代	5	5	1	5	是	00101
第 6 次迭代	0	4	0	1	否	001010
第 7 次迭代	0	3	0	1	否	0010100
第 8 次迭代	0	2	0	1	否	00101000

[0095]

第 9 次迭代	0	1	0	1	否	001010000
第 10 次迭代	0	0	0	1	否	0010100000

[0096] 图7中示出图6A的方法的替代表示。图7中的表编码整个方法，并且被示出为N＝10，其中列对应于被确定的当前位的位置，即n的当前值(在初始递减之后)，并且因此存在从0到9编号的N列。索引0对应于LSB，索引N-1对应于MSB。行对应于仍待附加的1的数目，即r的当前值。可以有n+1行；然而，在码中的位数N等于输入值中的位数L的情况下，如下所述，预处理操作(其也可以被称为输入优化)可以用于近似地将行数减半(从N+1行减小到$\lceil (N+1)/2 \rceil$行)，因此行被编号为0到5。粗体值是二项式系数$\binom{n}{r}$，对应于列数n和行数r。二项式系数之间的1和0表示在每个步骤中哪个位值被附加到码。

[0097] 该过程从右向左操作，开始于最右列(在所示的示例中n＝N-1＝9)并且在由所标识的子集的汉明权给定的行中，即其中r＝R(例如，在所示的示例中r＝R＝2)。如果更新后的概率索引x的当前值大于或等于当前二项式系数$\binom{n}{r}$，则更新后的概率索引x减小二项式系数的值，并且n和r都递减1，这对应于对角地向左和向上到下一个二项式系数的步长，并且如表中所示，1被附加到码。若更新后的概率索引x的当前值不大于或等于当前二项式系数$\binom{n}{r}$，则更新后的概率索引x不变，并且仅n递减1，这对应于下一个二项式系数的剩余步长，并且如表中所示，将0附加到码。由于比较x≥0将总是返回真，所以图7中的表的左下部中的灰度值永远不会到达具有汉明权R的码必须在其中具有R个1，并且一旦n、仍待附加的位数变为小于r、仍待附加的1的数目时，这就是不可能的。

[0098] 用于以上第二示例(即，用于N＝10并且概率索引的初始值是x＝37，并且因此R＝2并且用于当前子集的概率索引的起始值也可以被称为概率索引的修改或更新值是x＝26)的示例路径由图7中的阴影单元指示。一对阴影单元对应于二项式系数(以粗体示出)，该二项式系数将与x的当前值进行比较，基于该比较，与随后的位选择1或0耦合。

[0099] 在硬件中，图5A、图5B、图6A和图6B的逻辑中的循环可以展开，即循环的每个步骤可以一个接一个地顺序放置，和/或流水线化以便权衡等待时间与面积。

[0100] 如上所述，在N＝L的情况下，作为映射操作的一部分(在框204中)并且在概率排序之后(在框202中)发生的预处理操作与对应的后处理操作(也在框204中)相结合可以用于减小图7中所示的表的大小，并且可以附加地或替代地用于减小在映射操作内使用的LUT中的一些或全部的大小。这个预处理和后处理操作对可能对于N的较大值特别有用，因为虽然使用某种形式的LUT可能导致较小的映射硬件逻辑和对于N的较小值的较快操作，但是随着N的值增加，LUT的大小增加。存储该表所需的位数随N成立方增长。

[0101] 图8A示出向图4的方法(如上所述)添加预处理和后处理操作对(框802和804)的映射操作(框204中)的示例实现。在预处理阶段(框802)，确定是否反转输入，即概率索引(如框202中所确定的)。如果概率索引x的值在范围的上半部分中，即$x \geq 2^{N-1}$(框806中的“是”)，则翻转概率索引中的所有位，即0变为1且1变为0(框808)。然而，如果概率索引x不在范围的

上半部分(框806中的“否”),即,它在范围的下半部分,则概率索引保持不变(框810)。概率索引x是否在范围的上半部分中的确定(在框806中)可通过在以二进制写入时检查概率索引的MSB来实现。如果是1,则该值在该范围的上半部分,而如果是0,则不是。这意味着框806中的判定和随后的处理框(框808-810)等同于框824(下面描述)。

[0102] 然后,如上参考图4所述,通过首先基于概率索引标识码的子集(框402),其中每个子集对应于不同的HW并且仅包括来自具有特定HW的码集的那些码,然后选择所标识的码子集中的码之一(框404),来继续映射。

[0103] 在选择了码(在方框404中)之后,后处理级(方框804)确定是否反转所述码。如果输入概率索引被反转(框812中的“是”),则在输出码之前翻转所选码中的所有位(框814);然而,如果输入概率索引没有被反转(框812中的“否”),则输出所选择的码或保持其不变(框816)。图8A的这种方法可以使用单个标志位来实现,该标志位根据输入是否被反转来设置(在框802中),然后用来决定(在框812中)在输出码之前是否修改该码。一旦生成了码(在框804中),标志位就不再需要被存储并且可以被丢弃。

[0104] 图8A的方法内的分支(在框802和804中)可以如图8B所示被消除。图8B示出向图4的方法(如上所述)添加预处理和后处理操作对(框802和804)的映射操作(框204中)的另一示例实现。如图8B所示,提取x的MSB(框822),并与x的剩余位进行异或(框824)。然后,使用x的这些修改位(即,框824的异或操作的输出)来生成码(在框402-404中),并且将x的MSB(如框822中所提取的)存储为标志(框826)在使用x的修改位选择了码(在框404中)之后,访问存储的MSB(框832)并与选择的码进行异或(框834)。然后输出该第二异或运算的结果(在框834中)。这意味着,根据是使用图8A的方法还是图8B的方法,使用不同的位集来标识(即,输入到框402的)码子集;然而,这种差异不影响结果。在图8A的方法中,将标志作为直接信号/线从方框806到方框812单独存储到输入位。输入的所有位(在框802中的预处理之后)作为输入被提供给框402,然而MSB总是被设置为0,因此它不包含信息并且不影响码子集的标识(在框402中)。相比之下,在图8B的方法中,在异或(在框824中)之后,标志留作MSB,并且剩余的N-1位用作框402的输入(具有隐式前导0)。

[0105] 使用这个预处理和后处理操作对(框802和804)显著地减少了N=L时所需的二项式系数的数目,因此提供了对图4的方法的优化。例如,在N=L并且不使用该对运算的情况下,对于R存在N+1个选项,并且因此存储二项式系数的LUT包括N+1行(因此(N+1)*(N+1)个条目)相比之下,在使用这对运算的情况下,对于R仅存在•(N+1)/2•个选项,因此存储二项式系数的LUT包括•(N+1)/2•行(并且因此(N+1)*•(N+1)/2•个条目)。在存在一些填充的情况下,即N>L,对于R存在最多•(N+1)/2•个选项,因此存储二项式系数的LUT包括•(N+1)/2•行。通过如上文所描述地消除前两个行和列,可减小LUT的大小。在所有情况下,存储该表所需的位数是$O(N^3)$。虽然这种优化可以在N>L的情况下使用,但是它没有提供任何益处,因为当N>L时,x从不在所有值的上半部(例如,在框806中它总是"否",并且x的MSB总是0,因此框824和834中的异或操作对这些值没有影响)。

[0106] 图9示出图2的方法的变型,其包括预处理操作(框902),该预处理操作可以在示例中的概率排序(框202中)之前实现,在该示例中,一个或多个输入值被分组成数据字。预处理操作(在框902中)是去相关操作,其目的在于去除或至少减少相同数据字中的输入值之间的相似性,例如去除或减少空间或通道相关性。在这种变化中,用于确定概率索引(在框

202中)的输入值是从去相关操作(在框902中)输出的修改的(即去相关的)输入值。

[0107] 有许多不同的方式来执行去相关(在框902中),并且可以根据输入值的类型(即,它是什么类型的数据,例如,它是否是自然视频数据、视频游戏数据、GUI数据等来选择所使用的方法,并且这可以在数据格式中反映出来,因为YUV更有可能被用于自然视频数据,而RGB更有可能被用于视频游戏和GUI)。在使用取决于数据类型(或其他标准)的多种不同的去相关方法中的一种的系统中,一个或多个自由位(例如,一个或多个填充位)可以用于指示所使用的去相关方法。对于一些数据类型,诸如具有更饱和值的那些数据类型(例如,饱和视频),可以省略去相关(在框902中)(如图2中)。类似地,对于具有空通道或alpha(例如RGBX或RGBA)的格式,对于额外通道(如图2中)可以省略去相关操作(在框902中),否则去相关可以引入额外信息。

[0108] 在各种示例中,通过取每个值与所选参考值之间的差来执行去相关(在框902中),并且所述方法可以适合于具有较少极值的数据,例如,诸如自然视频数据的较不饱和值(即,较少极值)。参考值可以是固定常数,或者可以例如基于单词中的输入值(例如输入值的平均值)来计算,或者可以以另一种方式来确定。在一些示例中,参考值可以从数据字内的输入值中选择。例如,如果数据字包括K个输入值,则可以将参考值指定为索引为J=•K/2•(即,中间索引)的输入值。在其他示例中,参考索引J可以被选择为用于特定格式的输入值的固定常数,并且被存储在系统中或者以其他方式被设置为默认值。例如,对于3或4值图像格式(例如RGB、RGBX、RGBA或平面Y/U/V),参考索引J可以被设置为1-这对应于绿色(其是最重要的颜色)以及中间值。在参考值的索引(或者如果参考值是计算的而不是使用索引选择的,则参考值本身)在字之间变化的任何示例中,则可能需要存储标识参考值选择的位(例如,索引的值或参考值本身)(例如,在填充位中)。因此,使用参考索引J的固定(并且因此是默认)值可以提供更简单的实现方式。

[0109] 在以某种方式标识或选择了参考值之后,如果数据字是固定常数或存储在填充位中,则从数据字中的每个输入值中减去参考值,或者如果数据字中的索引值给出了数据字中的每个其他输入值,则从数据字中的每个其他输入值中减去参考值,其中以2^L为模。在后一种情况下,可以从参考值本身中减去固定常数,以2^L为模,例如当输入值分布在平均值周围时(例如对于YUV数据),可以减去2^{L-1}。

[0110] 对于大的L值,可以应用小波(例如Haar或线性)来执行去相关,或者可以使用由线性映射和/或移位组成的一些其他更复杂的线性或仿射去相关,只要映射是可逆的(即,仿射变换的非奇异线性)。

[0111] 例如,给定范围从0到1023的值和包括三个10位值485、480和461(K=3,L=10)的数据字,则该字的二进制表示是01111001010111100000111001101,其具有汉明权16。参考索引被指定为J=•K/2•=1,因此值480是参考值。从485和461中减去参考值480(以1024为模),并且从参考值480中减去中值512(给定值0-1023的范围)。因此,这三个值从485、480、461映射(在框902中)到5、992、1005。

[0112] 除了对定点值使用上述去相关方法之外,还可以对浮点值使用去相关操作,只要注意符号位并且该操作是无损的,即可逆的。在一个示例中,去相关可以包括当考虑所有值以2^L为模时,通过其符号位对每个值进行异或以将0对准为紧接着负0,并且以降序将负值分布在正值以下。此后,可以从参考值中以2^L为模减去一些值,就像用于固定点去相关一

样。这有效地在伪对数空间中执行操作，因为MSB中的索引被线性处理，但是仍然旨在将变换后的输入分布在0附近。如果字中的任何输入值已经表示了带符号(而不是无符号)值，则通常不需要对这些值进行去相关，因为假设它们已经分布在0附近。

[0113] 图10A和10B示出对于所有的Y、所有的U、所有的V或U和V数据的混合的字的去相关的输入值(如从框902输出的)的概率分布图。图10A示出差值的概率分布，即对于不是参考值的去相关输入值，而图10B示出移位参考值的概率分布。在这两种情况下，去相关值分布在0周围，因此符号重新映射可用于执行概率排序(在框202中)，如上所述。符号重新映射将三个值5、992、1005映射到10、63和37，并且在此示例中，如通常情况，经变换参考值最大，因为其不如差约为0那样紧密地分布在平均值周围(如图10A和10B中所示)。这在图11A和11B中进一步示出，其示出符号重新映射差值(图11A中)和符号重新映射参考值(图11B中线的厚度是不完美的排序的结果，这导致概率的小尺度变化，与图11A中的情况下整齐地减小的概率相反)，再次针对Y、U或V数据。

[0114] 在上述示例中，N＝L＝10，如果N＜L，则码将是有损的，因此使用大于或等于L的N的值。然而，在其他示例中，可以使用一个或多个填充位，例如以允许更长的码，使得N＞L并且N＝L+P，其中P是用于对输入值进行编码的填充位的数目。如上所述，在各种示例中，除了多个输入值之外，数据字可包含一个或多个填充位，例如，YUV和RGB数据值可被包装成32位数据字，其分别包含三个输入值以及2或8个填充位，并且当对数据字中的输入值中的一个或多个进行编码时可利用这些填充位中的一个或多个。虽然所述方法在N的值远大于L(例如N＝2L)的情况下仍然有效，但是这可能增加复杂度，同时提供码的效率的较小增加。这在图12中示出，其是绘制了对于P的各种值，映射到具有P位填充的前2^LN位二项式码的一组均匀随机L位输入值(其中L＝10)的平均汉明权的图，其中N＝L+P.水平轴表示填充位P的数目(从0到22)，并且垂直轴表示前2^LN位码的平均HW。如图12所示，P＝0的平均汉明权如所期望的是5，因为没有填充，1的平均数由L＝10位的一半给出。对于0和22之间的P，平均HW从5下降到大约2.5，梯度下降到0。这从视觉上表明，通过添加填充的前几位可以获得最大的益处。对于P，性能的增益可以达到2^L-L-1，此时平均HW为$(2^L-1)/2^L$～1，然而在许多示例中，N＜2L，否则N≥2L，并且在这种情况下，可以以一半位数来存储码，这是一种更简单的优化。

[0115] 在各种示例中，当对参考值进行编码时可使用一个或多个填充位，并且当对差值进行编码时可不使用填充位。这是因为，如图10B所示，移位的参考值的概率分布没有差值那样紧密地分布在0周围。参考前面的示例，其中数据字包括三个10位输入值(K＝3，L＝10)，可以存在两个填充位(形成32位数据字)，并且这两个填充位可以用于编码参考值，使得对于第一和第三输入值，结果码包括10位(N_0＝N_2＝10)，并且对于第二输入值，结果码包括12位(N_1＝12)在以上示例中，输入值485、480、461被去相关(在框902中)以给出5、992、1005，并且这些然后被符号重新映射(在框202中)以生成三个概率索引：x_0＝10，x_1＝63并且x_2＝37。然后，通过使用图5A的方法识别码的子集(在框402中)，使得R_0＝1，R_1＝2并且R_2＝2，并且更新的概率索引是x_0＝9，x_1＝50并且x_2＝26，并且然后使用LUT或图6A的方法来识别(或生成)具有指定汉明权重的码，可以将这些映射到码(在框204中)。得到的码是1000000000、0100000100000和0010100000。

[0116] 虽然在此示例中，所有填充位(例如，用于填充到32位数据字中的三个10位输入值的可用填充位两者)用于编码数据字中的输入值中的仅一者，但在其他示例中，填充位可在

输入值中的两者或两者以上之间共享。例如，在三个8位输入值被打包成32位数据字的情况下，每个输入值可以被映射到10位码(N＝10)并且可以存在两个未使用的填充位，或者可替代地，两个输入值可以被映射到10位码并且一个输入值可以被映射到12位码，使得所有填充位都被使用。

[0117] 在输入值具有非均匀概率分布或输入值具有均匀随机概率分布的情况下，可以使用上述方法；但是，对于具有均匀随机概率分布的输入值，并且其中N＝L，上述方法在减少传送和/或存储数据时消耗的功率方面可能不提供任何明显的益处。然而，上述方法的变型可以用于减少在发送和/或存储数据时消耗的功率，其中对于具有均匀概率分布的数据(例如，对于随机数据)存在一个或多个填充位(即，其中N＞L)。在这样的示例中，如图17所示，不是基于单独确定的概率分布来映射输入值(如在框104中)，而是将L位输入值映射到N位码，其中N＞L(框1702)。作为该映射操作的一部分(在框1702中)，代替确定单独的概率索引(例如在框202中)，L位输入值本身被用作概率索引。或者，在使用去相关操作(例如在框902中)并导致具有均匀随机概率分布的输入值的罕见情形中，在去相关操作中从L位输入值生成的L位值被用作概率索引。在图17的方法中，基于L位输入值本身，L位值被映射到N位码，其中N＞L(在框1702中)。

[0118] 在图4所示和上文所述的变型中，可以首先基于L位值标识码的子集(框1704)，然后标识码的子集中的一个(框404)。这可以使用LUT(例如，如上所述，例如参考图3)和/或使用迭代过程(例如，如上参考图6所述)来完成。作为基于N位码的HW对N位码进行分组和排序的结果(如上文详述)，随机输入值数据被映射到最功率高效的码子集。虽然图17没有示出图8A和8B的可选的一对预处理和后处理操作(框802和804)，但是这些也可以用于输入数据值没有概率分布的情况(例如，对于基本上随机的输入数据)。

[0119] 如上文所描述，在一些示例中(不管数据是否具有非均匀概率分布或为实质上随机的)，可将多个输入值分组成数据字。在这样的示例中，结果码(每个输入值一个)可以以任何顺序被分组(例如，复用)到输出数据字中，因为该顺序不影响输出字的HW。例如，如在上面的示例中，在结果码是1000000000、0100000100000和0010100000的情况下，由参考值生成的码可以被打包在数据字的末尾，因为它使用附加的P个填充位，这可以简化码的复用以形成输出数据字。汉明权为5的结果输出数据字由下式给出：

[0120] 10000000000010100000010000100000

[0121] 如果节省功率的最大因素是使字的汉明权最小化，则该输出数据字可以通过外部总线被发送到例如外部存储器或另一外部模块，诸如显示控制器。

[0122] 然而，如果当在外部总线上传输时节省功率的较大因素是最小化位翻转的数目，则可使用如上所述的相同方法，并且可实现额外的后处理操作(框1302)，如图13中所示。此后处理操作(在框1302中)涉及“异或”操作，如下所述。虽然图13是图9的方法的变型，但是(框1302的)后处理操作也可以与图17的方法一起使用，和/或(框1302的)后处理操作可以在没有任何去相关的情况下使用(即省略框902)。此外，该后处理可以结合图8A和8B的方法中的优化来使用。或者，异或操作可以不作为后处理操作(如框1302中)执行，而是作为生成码的过程的一部分执行，如图6B所示。如图6B所示，当与图6A所示的初始方法(以及上面描述的)比较时，除了第一位之外，附加所有的翻转的前一位(框606)，而不是附加1(框616)，并且附加前一(未翻转)位(框618)，而不是附加0(框608)。

[0123] 在后处理操作(框1302)中,码可以被看作差的列表,并且每个位(除了第一个位之外)可以依次与其先前的相邻位进行异或,其中先前可以取决于数据如何被发送而在向左或向右的方向上。在以下示例中,前一相邻者在左边,并且将码:

[0124] 10000000000010100000010000100000

[0125] 变换成:

[0126] 11111111111100111111100000111111

[0127] 然而,如果码在B位总线中传送(其中B是总线宽度),则不将每个位(除前B个位中的每一个位外)与其前一相邻者进行异或,而是依次与位B之前的位置进行异或,并且在此示例中,码:

[0128] 10000000000010100000010000100000

[0129] 其可如下经由四位总线(B=4)传输:

[0130] 通道0:10010000(3位翻转)

[0131] 通道1:00000100(2位翻转)

[0132] 通道2:00010010(4位翻转)

[0133] 通道3:00000000(0位翻转)

[0134] 被变换成:

[0135] 10001000100000100010011001000100

[0136] 其可如下经由四位总线传输:

[0137] 通道0:11100000(1位翻转)

[0138] 通道1:00000111(1位翻转)

[0139] 通道2:00011100(2位翻转)

[0140] 通道3:00000000(0位翻转)

[0141] 可以看出,该后处理将位翻转的总数从9减少到4。

[0142] 如果码在不同宽度的总线上传送,即,不存在一致的总线大小B,则B的值可以被选择为所有总线大小的最高公因子。例如,如果先前的码(1000000000001010000001000010000000)通过4位总线和8位总线两者传输,则使用B=4对其进行后处理,并且该后处理减少了4位总线上的位翻转的总数(从9到4),并保持8位总线上的位翻转的初始数目(总共8)。

[0143] 无需后处理:

[0144] 通道0:1000(1位翻转)

[0145] 通道1:0000(0位翻转)

[0146] 通道2:0001(1位翻转)

[0147] 通道3:0000(0位翻转)

[0148] 通道4:0100(2位翻转)

[0149] 通道5:0010(2位翻转)

[0150] 通道6:0100(2位翻转)

[0151] 通道7:0000(0位翻转)

[0152] 采用后处理:

[0153] 通道0:1100(1位翻转)

[0154] 通道1:0001(1位翻转)

[0155] 通道2:0010(2位翻转)

[0156] 通道3:0000(0位翻转)

[0157] 通道4:1000(1位翻转)

[0158] 通道5:0011(1位翻转)

[0159] 通道6:0110(2位翻转)

[0160] 通道7:0000(0位翻转)

[0161] 将了解,此32位示例相当短(就每通道的位的数目而言),并且对于更代表实际实施方案的较大数目的位,通常将存在8位总线以及4位总线上的位翻转的数目的减少。

[0162] 在进一步可选的后处理操作中(在框1302中),如果数据正被流式传输并且在编码点和解码点两者处对先前B位的访问可用,则可以通过将第一B位中的每一个与其在位流中的先前位进行异或来实现进一步的功率节省,并且这可以通过因子M/(M-B)来提高效率,其中M是数据字中的位数(例如M=32)。在该示例中,在存储编码数据之前,至少反转该最终异或级;除非数据随后将以与写入相同的顺序读出,例如当流式传输时。

[0163] 如图13所示,框1302的后处理操作可以与框902的去相关预处理操作结合使用,或者可以独立于去相关操作而使用。

[0164] 除了图13中的后处理操作(框1302)之外,上述许多方法被描述为进行操作以便最小化(或至少减小)输出码中的汉明权(即1的数目)。在这些情况下,可以说系统具有用于输出码的汉明权的目标值0。然而,可能存在例如由于某些硅工艺中固有的不对称性而导致的实施方案,其中传输或存储一比传输或存储0更具功率效率。在这些情况下,可以说系统具有输出码的汉明权的目标值,该目标值是最大值(并且取决于码的长度)。当以最大汉明权为目标时,可以使用相同的方法,其中一个附加操作是在方法结束时翻转输出码的所有位。或者,可修改以上方法以将对0的参考替换为对1的参考,并且反之亦然(在图6A的框606和608中,图22的框2202中,图23中的框2302和2307中,以及图24中的框2402中)。当最大化1而不是0时,仍然可以使用上面参考图8A、8B和9(例如在框802、804和902中)描述的预处理和/或后处理操作。然而,如果类似的改变被应用于后处理操作1302(或如图6B所示的其等效操作),则这将替代地导致最大化(而不是最小化)位翻转,并且因此这可以仅在这是期望的结果的情况下被实现。

[0165] 在一些实现场景中,系统可以在一些情况下从最大化1而在其他情况下最大化0中受益。为了实现这一点,可以提供标记(例如,单个位),该标记在最大化1的方法的变体和最大化0的方法的变体之间切换操作。

[0166] 如上所述,在映射操作(在框204中)中,可以基于预定义码中存在的1的数目或者基于预定义码中存在的多个位翻转的数目,将概率索引(以及因此的输入值)映射到一组预定义码中的一者。上面的示例(除了图6B和框1302)都涉及这些选项中的第一个(即,基于1的数目);然而,也可以使用上述映射操作(包括如图17中存在均匀概率分布的情况),其中基于预定义码中存在的位翻转的数目(即基于码内的两个连续位中的0与1之间或1与0之间的转变的数目)来执行映射。例如,与图3的LUT 300相比,这将导致码重新分组成子集,并且因此也导致码的重新排序。然而,现在码的重新分组取决于码是否被流式传输。如果码正在被流式传输,则前一相邻者可用于异或,并且以一个开始的每个码(取决于流的方向,在左边或右边)指示隐式位翻转。图14A中示出示例LUT 1420,并且可以从第二子集,子集1422中

看到该隐式位翻转，该第二子集包含具有一个位翻转的码，包括码1111111111。如果码没有被流式传输，则前一相邻者不可用于异或，并且图14B中示出示例LUT 1400。

[0167] 如图14B所示，10位码的集合或子集被细分为多个子集1401-1410，每个子集包括两个或更多个包含相同数目翻转的码，例如第一子集1401包括两个没有翻转的码，第二子集1402包括18个具有一个翻转的码，等等。在每个子集内，具有相同数目的翻转的码可以以任何方式排序，并且在所示的示例中，它们按字典排序。如图14B所示，虽然在非常少的情况下，概率索引的10位二进制版本匹配10位码，但是在大多数情况下，两者是不同的。

[0168] 在使用翻转的数量而不是HW的情况下，例如可以通过从概率索引x中迭代地减去二项式系数$\binom{N-1}{r}$的两倍来识别码的子集(在方框402中)，其中N是码中的位的数目，并且初始地r＝0。图15A示出与图5A等效的方案，其中使用翻转的数目F代替汉明权R。如图15A所示，概率索引x初始地与r＝0的二项式系数的两倍进行比较(框1502)，r＝0等于二(与N的值无关)-如图15A所示，使用翻转次数的二项式系数是$\binom{N-1}{r}$而非$\binom{N}{r}$，因为在N位码中有N个位位置，在可以发生翻转的位之间只有N-1个点(除非跨相邻码的异或发生，例如在流式传输中，在这种情况下使用图6B的先前方法，框1302或图14A)，但是(较小的)二项式系数被乘以二，因为对于初始位的值有两个选项:1或0，然后所有其他位由位翻转的数目和那些位翻转的位置确定。如果概率索引严格小于二(框1502中的“是”)，即，它是0或1，则选择翻转次数F为0的第一子集1401(框1504)。否则，从概率索引中减去二项式系数的两倍的值(即，在该第一迭代中为2)，并且将r的值加1(框1506)。在随后的迭代中，将来自先前迭代的更新后的概率索引x与二项式系数的两倍和r的当前值进行比较(在框1502中)，并且如果更新后的概率索引x严格小于二项式系数的两倍(在框1502中的“是”)，则选择翻转次数等于r的当前值的子集，即其中F＝r(框1504)。然而，如果更新的概率索引x不严格小于二项式系数的两倍(框1502中的“否”)，则从更新的概率索引中减去二项式系数的两倍的值(具有r的当前值)，并且r的值加1(框1506)。在选择子集(在框1502和1506中)中使用的二项式系数的值(或者是那些二项式系数的两倍)可以被计算或者可以被预先生成并且从LUT获得(例如，以与上面参考图5A所描述的类似的方式)。此外，以与从图5A所示的方法到图5B所示的方法的修改类似的方式，图15A的方法可以被修改为使用双二项式系数的累积版本，和/或LUT中的值可以是累积版本，以减少处理量、等待时间和功率(如上参考图5B所述)，如图15B所示。如图15B所示，通过使用累积值(在框1512的比较中)，循环中的更新步骤(框1516)仅包括r值的增量(而不是如图15A的框1506中那样还更新概率索引的值)，并且存在计算最终的、更新的概率索引x值的单个减法操作(框1514)。此减法运算(在框1514中)从初始概率索引减去可从LUT读取的累加值$2\sum_{i=0}^{i=r-1}\binom{N}{i}$，以在单个步骤中产生概率索引的最终更新版本。在另一变型中，代替存储双二项式系数，可以仅将二项式系数存储在LUT中(即，不乘以二)，并且可以动态地执行乘以二，即，通过将位移动一个位位置。

[0169] 一旦例如使用上述方法之一标识了子集(在框402中)，则可以使用作为子集的标识的一部分而确定的信息来从所标识的子集中选择码(在框404中)，如前所述。每个子集内的码可存储在LUT中，并基于概率索引x的最终更新值来选择，或者可替代地，码可使用概率

索引x的最终更新值在迭代过程中一次一位地生成。如图22所示。如图22所示，迭代方法使用两个迭代值N和r，并且初始地n＝N-1(其中如上所述，N是码的位长度)和r＝F(其中如上所述，F是所标识的子集中的翻转的数目)，并且，在预处理阶段(框2202-2206)之后，N递减1(框2208)，使得正好存在N-1次迭代(n＝N-2、N-3、...、1、0中的每一个一次)。

[0170] 在图22的方法的预处理阶段，将零附加到代码(框2202)，并且如果概率索引x的最终更新值大于或等于二项式系数$\binom{n}{r}$(框2204中的'是')，则设置标志，并且通过将x设置为等于$2\binom{n}{r}-1-x$来减小概率索引的值(框2206)。在每个随后的迭代中，n递减1(框2208)，并且将更新的概率索引x与具有当前n和r值的二项式系数$\binom{n}{r}$进行比较(框2210)。如果更新的概率索引x大于或等于二项式系数$\binom{n}{r}$(方框2210中的"是")，则将前一位的翻转版本(即，最近附加到该码的位)附加到该码，从概率索引x中减去二项式系数$\binom{n}{r}$的值，并且r递减1(框2212)。然而，如果更新的概率索引x不大于或等于二项式系数$\binom{n}{r}$(在框2210中为“否”)，则将前一位附加到码，并且x和r的值不变(框2214)。当n＝0(框2216中的“是”)并且然后存在后处理阶段时，迭代循环停止。在后处理阶段中，如果在预处理阶段中设置了标志(框2218中的“是”)，则翻转码中的所有位(框2220)。如果标志未被设置(框2218中的“否”)，则码中的位不翻转。该标志可以被编码为框2206和2218之间的直接信号，或者它可以被存储为输出码旁边的附加位，该附加位在后处理期间被丢弃。

[0171] 图23示出生成码的另一迭代方法，该码是图22所示的码的变型并且如上所述。与图22的方法不同，图23的方法避免了对后处理操作的需要，并且不涉及标志的存储(作为系统状态或作为每个码旁边的额外位)。因此，图23的方法比图22的方法更有效。如图23所示，1或0初始地被附加(在框2307或2302中)取决于第一比较(在框2204中)的结果，并且通过以这种方式设置码的初始位，不需要根据存储的标志值翻转码中的所有位。

[0172] 在上述许多示例中，输入值都具有相同的长度，即L位，以及输出码的长度，即N位。如上所述，在各种示例中，输入值和/或输出值可以在长度上变化，并且这可以取决于生成输入值的方式。在一个示例中，输入值可使用霍夫曼编码来生成，并且在使用字母符号及其相关联的概率的特定示例中，这些可通过对于每个输入字符以分配给叶节点的空输出二进制串开始来编码。这些叶节点被用作建立二叉树的基础。通过在每一步寻找具有最小概率的两个未处理的节点并创建新节点来处理数据，新节点的概率由前两个节点的和给出。第一/第二先前节点(以及分层结构中低于它的所有节点)的输出二进制串具有被推到其输出码前面的0/1，其中具有较高概率的节点被分配0，而具有较低概率的节点被分配1(通过以这种方式而不是任意地分配1和0，导致在以下示例中平均汉明权的高达9％的降低)。然后，这两个先前节点被认为是处理过的，并且之后在选择节点时被忽略。这被迭代地重复，直到仅剩下单个未处理的节点(被称为概率为1的根节点)。该过程建立二叉树，其中每个新创建的节点连接到生成它的两个先前节点，并且其中在层级结构更下游的输入字符(对应于叶节点)具有分配给它们的更长的输出二进制码。

[0173] 所得到的二进制码具有任意数目的位长，因此可能希望将码填充到4位的倍数，以便减少硬件中的多路复用(即复用)逻辑，并且使得在该示例中，希望将每个编码的输入字符填充到4、8或12位长。由于用0自然地填充霍夫曼码不提供增益，例如HW的减小，所以编码的输入字符可以用作这里描述的方法的输入，以将它们映射到更功率有效的长度为4、8或12位的输出码。

[0174] 霍夫曼编码的输入字符形成这里描述的编码方法的输入值，并且霍夫曼编码中使用的概率数据可能不再可用于概率排序操作(在框202中)。虽然在这种情况下，由初始概率数据给出的排序是不可用的，但是它被隐含地存储在来自码的长度和值的霍夫曼编码的输入值中，并且因此可以使用排序算法从它们中推断出，例如通过执行输入值的重复的成对比较(例如，冒泡或快速排序)以按照概率从最高到最低地放置输入值：如果两个输入值具有不同的长度，则较短的一个具有较高的概率，并且如果两个值具有相同的长度，则最小的按字典方式具有较高的概率。对于长度L_a和L_b的两个输入值a和b，这可以用未知的概率p_a和p_b写为：

[0175] $$p_a > p_b \leftrightarrow L_a < L_b \vee (L_a = L_b \wedge a < b)$$

[0176] 其中通过将相等长度的两个码视为L_a位整数来给出最终不等式。此分类和随后的二进制码生成(如下所述)可离线(而不是即时)执行并用于生成LUT。在运行时，可通过在LUT中编索引来标识码，并且在运行时，这在处理和等待时间方面更高效，尤其是对于大数据集。

[0177] 如上所述，在对输入值排序并因此隐式分配概率索引之后，将它们映射到一组预定义码中的一者(在框204中)，其中在该示例中，给定输入值的输出码的长度是输入值四舍五入到下一倍数的4位(即4、8或12位)的长度。因此，输出码满足半字节步幅的要求，而且还保持可变长度熵编码的大部分益处。

[0178] 为了使码是无前缀的(即，如果连接，则不需要单独存储长度)，8/12位码的前4/8位必须不同于4/8位码本身中的任何一个。因此，8位码字可被设置为全部以1010的相同前缀开始，其从1001的“r”的最不可能的4位码字继续，并且所有12位码字可被设置为以10101101开始，其从10101011的“k”的最不可能的8位码字继续。

[0179] 对于该示例，字符根据其概率排序的结果映射如下：

[0180]

字母	输入值	码	字母	输入值	码
e	011	0000	m	1100,0	1010,1000
t	111	0001	w	1100,1	1010,0011
a	0001	0010	f	1101,0	1010,0101
o	0010	0100	g	0011,00	1010,0110
i	0100	1000	y	0011,01	1010,1001
n	0101	0011	p	0011,10	1010,1010
s	1000	0101	b	0011,11	1010,1100
h	1001	0110	v	1101,11	1010,0111
r	1010	1001	k	1101,100	1010,1011
d	0000,0	1010,0000	j	1101,1010,0	1010,1101,0000
l	0000,1	1010,0001	x	1101,1010,1	1010,1101,0001

c	1011,0	1010,0010	q	1101,1011,0	1010,1101,0010
u	1011,1	1010,0100	z	1101,1011,1	1010,1101,0100

[0181] 从这可以看出，虽然在一些情况下输入值和输出码可以相同，但是在大多数情况下它们不同，并且平均相对HW(即，其中“平均”表示对由其关联概率加权的所有输出码的相对HW求，并且“相对HW”将HW称为码长度的比例，这是对于可变长度码比简单地HW更好的度量)在上述示例中与仅填充初始霍夫曼码相比减少了多于15％，因此这提供了相应的功率节省。

[0182] 可参考图16描述对数据进行功率高效编码的另一示例方法。虽然图16的方法可以用作上述方法的后处理操作，但是在大多数示例中，所述方法是独立使用的。与上述方法相比，图16的方法实现起来更简单，但是较早的方法提供了改进的性能(例如，功耗的较大改进)。

[0183] 图16的方法可用于减少将在外部总线上存在一个或多个未使用的填充位的任何地方传输的数据的汉明权。通过减小所传输的数据的HW，独立于发送数据的总线(其可为内部或外部总线)的宽度(例如，在总线宽度已知或总线宽度未知的情况下)而改进数据传输的功率效率。即使输入值具有均匀随机概率分布，并且例如可以用于上述类型的数据或诸如指针数据的其他类型的数据(例如，在其被对准到特定粒度级别的情况下)，所述方法也提供了功率效率的改进。与已知技术不同，图16的方法不需要总线(或存储器行)宽度的任何知识或对数据写入/读取的次序的任何控制(例如，其中存在由硬件执行的高速缓存或写入组合)。

[0184] 虽然下面对图16的方法的描述涉及基于确定多少(或什么比例的)位是1来降低数据的汉明权，但是更一般地，所述方法可以基于确定多少(或什么比例的)位具有预定义值(例如，在此将其设置为1或0)，使得降低(在此将预定义值设置为1)或增加(在此将预定义值设置为1)汉明权。预定值的选择可以取决于所使用的特定架构和硬件。

[0185] 在存在一个未使用的填充位的情况下，图16的方法可以应用于数据字中的所有其他位。在其他示例中，可将数据字中的其他位细分为若干部分，其中所述部分中的一者对应于未使用的填充位，并且可将图16的方法应用于所述部分而不应用于其他部分。在存在一个以上未使用的填充位的情况下，数据字中的其他位可细分为若干部分，其中每一部分或所述部分的子集对应于未使用的填充位中的一者，并且可独立于其他部分将图16的方法应用于具有对应的未使用的填充位的每一部分。在其中数据字被细分成多个部分的各种示例中，部分大小可被选择成使得每一部分具有偶数位长度，并且在各种示例中，不同部分可具有不同的位长度。

[0186] 如图16所示，所述方法包括确定位序列(如上所述，其可以包括数据字中的除了相应的未使用填充位之外的一些或所有位)是否包括比0多的1(框1602)，并且如果1比0少，或者具有相同数目的1和0(框1602中的“否”)，则将相应的填充位设置为指示位序列未翻转的值(例如0)(框1604)。相比之下，如果位序列中存在比0更多的1(框1602中的“是”)，则将对应的填充位设置为指示位序列已经翻转的值(例如1)(框1606)，并且位序列中的所有位都翻转，即反转(框1608)。因此，这减小了在外部总线上传输的位序列的汉明权，并且将最大汉明权封顶到单元(N/2)，其中N是包括填充位的位序列中的位数。

[0187] 虽然图16的此方法可仅在输入值很少具有高汉明权(例如，HW>•N/2•)的情况下

产生较小益处，但所述方法在输入值经常具有高汉明权的示例中提供较大益处。例如，如果输入值代表无符号整数，则具有高HW的值将主要是集合中的最大值。例如，如果输入值代表带符号整数，则具有高HW的值将主要是最小负值。对于具有均匀随机概率的输入值，所述方法提供了改进的效率，但是该改进小于分布向更高HW偏斜的情况，因为每个输入值具有相等的概率。

[0188] 通过独立于本文所述的其他方法而使用图16的方法，视视频数据的性质而定，视频数据的HW可减小高达约20％。相比之下，通过使用本文所述的其他方法(例如，具有各种任选的预处理阶段和后处理阶段的图1的方法)，视频数据的HW可降低50％以上，并且对于一些类型的视频数据可降低80％之多。

[0189] 虽然在图16中未示出，但是在各种示例中，上述去相关操作(框902)可以被用作图16的方法的预处理步骤。这提供了使用所述方法所获得的益处的额外改进，因为图16的位翻转对具有高HW的值(例如小的负值)进行了最大改进。

[0190] 虽然上述方法已经被描述为提高外部总线上的数据传输效率和外部存储器中的数据存储效率，但是在数据通过内部总线传输和/或存储在本地高速缓存中的情况下，功率效率可以被提高。在此类示例中，本文所描述的方法可在产生输入值的处理操作结束时实现，例如在处理图形数据的图块结束时实现。

[0191] 上述方法涉及数据值的编码。在使用经编码数据值之前，并且在一些示例中，在存储经编码数据值之前，可使用解码方法来恢复初始输入值。对使用本文所述的任何方法生成的编码数据值进行解码可以通过反转方法的每个阶段并以反转的顺序执行这些阶段来执行。

[0192] 参考图2的编码方法，对应的解码方法包括将每个码映射到概率索引(即，在与框204的概率索引相反的操作中)，然后确定与每个概率索引对应的输入值(即，在与框202的概率索引相反的操作中)。从码到概率索引的映射(在框204的逆中)使用与码生成中使用的相同的二项式系数(并且因此可以使用相同的LUT)。

[0193] 参考图5A、5B、6A和6B描述的映射方法，通过如图24所示的那样反向执行这些方法来确定码在其相应子集内的位置，可以将码映射回概率索引。所述方法使用三个参数n、x和r进行迭代，这三个参数初始地都可以被设置为0。所述方法检查码的LSB，并且如果它是一(框2402中的“是”)，则x和r的值被更新(框2404A或2404B，其中框2404B的实现更有效，因为它节省了加法操作)。不考虑LSB的值，然后从码中移除LSB(框2406)，递增n的值(框2407)，并且所述方法继续在每个后续迭代中检查新的LSB，直到码中没有剩余的位(框2408中的“否”)。在这一点上(即，在框2408中的“否”之后)，r的最终值被设置为汉明权R(在框2410中)，并且x的最终值可以在框2412中被计算为$x=x+\sum_{r=0}^{R-1}\binom{N}{r}$其中累积二项式系数的值可以从LUT读取。然后输出x的最终值(如在框2412中计算的)(框2414)。

[0194] 虽然图24的方法涉及在循环的每次迭代中移除LSB(框2406中)，但是在其他示例中，所述方法可以从LSB到MSB步进通过二进制码。在这样的示例中，循环结束时的决策框(框2408)可被表达为“n＜N?’并且该变化也可用于移除LSB的情况(如图24所示)。

[0195] 在图24的方法中，在循环内x的递增(在框2404A或2404B中)对应于图6A的反向操作，即，找到对应于给定汉明权R的子集内的输出码，在循环之后将最终二项式系数加到x

(在框2412中)对应于取消图5A或5B的操作,即,找到初始x落入的由汉明权R给定的子集,并相应地将其修改为相对索引。

[0196] 参考图7所示的表,该表可以用于将概率索引映射到码以及将码映射到概率索引,码中的1和0描述了通过该表的路径(从左到右),其中在位值一之后到达的每个二项式系数被累加到当前总数(其初始地为0)。从码字的LSB开始,每个LSB对应于一个向下和向右的步长,并且该步长达到的二项式系数被加到当前总数中。码中的每个0仅对应于右边的一个步长,并且在这种情况下,当前总数不被修改。由于在码中有N个位和R个1,所以该过程在位置$\binom{N-1}{R}$处终止。因此,这对应于图24的循环内的框。将最终的总和加到累积二项式系数$\sum_{r=0}^{R-1}\binom{N}{r}$以产生码字的概率索引,并且这对应于图24的循环之后的框。

[0197] 在生成概率索引之后,将其映射回输入值。在这种概率排序(在框202中)使用符号重新映射的情况下,可以通过取消符号重新映射来标识输入值,这通过:移除LSB,将剩余位向右移位一位位置(其涉及添加0作为MSB),并且接着将所有L位(即,移除LSB之后剩余的位加上新添加的MSB)与所移除的LSB进行异或。在修改的符号重映射用于浮点输入的情况下,通过仅将LSB移动到MSB来反转该操作。因此,概率索引和输入值都包括L位。在当对输入值进行编码时执行去相关(在框902中)的那些示例中,通过反号重映射生成的值是去相关的输入值,并且因此所述方法还包括例如通过将参考值移位中值并且然后将其他值移位修改的参考值或一般地取消初始线性或仿射变换来反转去相关。然而,在对输入值进行编码时没有执行去相关的情况下,通过符号逆重映射生成的值是实际输入值,并且解码完成。

[0198] 在通过参考LUT从输入值生成概率索引的情况下,通过反向参考LUT同样从概率索引解码输入值。

[0199] 参考图16的编码方法,当独立于本文所述的任何其他方法使用时,对应的解码方法包括:对于每一位段(其中数据字中可存在一个或多个段),读取对应填充位的值,并且如果填充位指示段在编码过程期间翻转,则翻转段中的所有位并将填充位复位到其默认值,并且如果填充位指示段在编码过程期间未翻转,则使段中的位保持不变并将填充位复位到其默认值。

[0200] 图3、14A和14B示出示例LUT,并且将理解,在其他示例中,每个子集(例如,图3和14A的每个HW子集)内的条目可以被重新布置以减少对LUT进行编码所需的门的数目。

[0201] 图18和19示出其中可以实现本文描述的方法的两个示例硬件实现场景1800、1900。在第一示例中,如图18所示,示出两个计算实体1802、1804,并且这些计算实体可以是外围设备、处理器或存储器。图18中的第一计算实体1802包括提取/输出硬件框1812,其被布置成将数据值输出到编码硬件框1822,该编码硬件块被布置成执行本文描述的编码方法之一。所得到的编码数据在总线1806上被携带并在第二计算实体1804处被解码。第二计算实体包括解码硬件框1814,其被布置为执行由第一计算实体1802实现的编码方法的逆方法,并且经解码的数据项随后被输入到输入/存储硬件框1824。

[0202] 在各种示例中,编码硬件框1822包括输入1826、输出1828和硬件逻辑1830,其可被称为映射硬件逻辑。输入1826被配置为接收多个输入值,映射硬件逻辑1830被布置为基于输入值的概率分布和码的特性将每个输入值映射到一组预定义码中的一者,其中码的特性

包括码的汉明权或码内的位翻转的数目，并且输出1828被布置为输出与所接收的输入值相对应的码。在这样的示例中，解码硬件框1814包括输入1832、输出1834和硬件逻辑1836，其可以被称为映射硬件逻辑。输入1832被配置为接收多个输入码，映射硬件逻辑1836被布置为基于解码值的概率分布和码的特性将每个输入码映射到一组预定义解码值中的一者，其中码的特性包括码的汉明权或码内的位翻转的数目，并且输出1834被布置为输出与所接收的输入码相对应的解码值。

[0203] 在其他示例中，编码硬件框1822仍包括输入1826和输出1828以及硬件逻辑1830。如上所述，输入1826被配置为接收多个输入值，然而，在这些示例中，每个输入字包括一个或多个输入值和一个或多个填充位。在这些示例中，硬件逻辑1830不同地操作，并且替代地被布置成确定输入字的一部分中的一半以上的位是否是1，并且响应于确定输入字的一部分中的一半以上的位是1，通过将该部分中的所有位反转并且将填充位设置为指示该反转的值来生成输出字。在这些示例中，输出1828被布置为输出字。在这些示例中，解码硬件框1814包括输入1832、输出1834和硬件逻辑1836。输入1832被配置为接收多个输入字，其中每个输入字包括一个或多个位的段及对应于每一段的填充位。硬件逻辑1836被布置成针对输入字的每个部分：读取并分析对应填充位的值；响应于确定所述填充位指示所述段在所述编码过程期间翻转，翻转所述段中的所有位且将所述填充位复位到其默认值；以及响应于确定所述填充位指示所述段在所述编码过程期间未翻转，使所述段中的所述位保持不变且将所述填充位复位到其默认值。输出1834被布置成将所得到的位作为经解码的字输出。

[0204] 在图19所示的第二示例中，存在三个计算实体1802、1804、1908。这些中的前两个如上面参考图18所述，并且第三实体1908是存储器。在该第二示例中，与第一示例不同，编码数据在总线1806上携带并存储在存储器1908中，并且数据不被解码，直到第二计算实体1804需要它为止。在该第二示例中，可以理解，第一实体1802和第二计算实体1804可以是同一计算实体。

[0205] 应当理解，图18和19中所示的计算实体1802、1804可以包括图18和19中未示出的许多附加元件，例如，编码和解码块可以分别由其他对编码/压缩和解码/解压块进行预处理和后处理，以进一步修改编码，从而独立于HW的降低而改善其他属性。

[0206] 对于本文所述的方法，存在许多示例使用情况。例如，当从缓冲器向屏幕写出像素数据时，并且在该示例中，数据通常是相关的和填充的。在解码正常的可变长度压缩之后，在被传递到屏幕或其他设备之前，这里描述的方法还可以用作视频系统的一部分。这里描述的方法可以代替任何压缩/解压缩流水线或与任何压缩/解压缩流水线耦合(因为该数据通常将具有相关的分布)。这将系统从仅仅带宽节省改进到功率节省。压缩/解压缩流水线被广泛地使用，因为压缩通常被应用于图像/视频/纹理/音频等。所述方法还可用于其他类型的相关数据或填充到2位的幂的数据(例如，不填充二字节的幂的用户界定的结构)。

[0207] 以上描述的图12示出附加填充位对本文描述的第一方法(例如，图1-16的方法)的影响。如图12所示，所述方法继续获得高达总共2^L-L-1个填充位的益处，尽管每个附加的填充位具有越来越小的影响。相比之下，第二方法(如上文参看图16所描述)不受益于具有多于L/2个填充位(因为通过在小于2个位的输入值上使用旗标位不获得益处)。下面针对L的各种值给出了两种方法的填充位的一些示例最大数目：

[0208]

位数L	Max P, 第一方法(图 1-15A、15B)	Max P, 第二方法(图16)
1	0	0
2	1	1
3	4	1
4	11	2
5	26	2
6	57	3
7	120	3
8	247	4

[0209] 使用上述方法(除了图16的方法之外)可以实现的效率可以如下所述地说明。

[0210] 令$i\in 2^N=\{0,1,\ldots 2^N-1\}$是编码的输入值

[0211] 令$p_i\in[0,1]$是i的概率,使得$\sum_{i\in 2^N} p_i = 1$

[0212] 令$H:2^N\to N, i \mapsto H_i$是汉明权函数

[0213] 则平均汉明权W由下式给出

[0214] $$W = \sum_{i\in 2^N} p_i H_i$$

[0215] 令H^n是汉明权$n\in\{0,1,\ldots N\}$的码集

[0216] 令p^n是输入码的概率的总和H^n则

[0217] $W = \sum_{n=0}^{N}\sum_{i\in H^n} p_i H_i = \sum_{n=0}^{N} n\sum_{i\in H^n} p_i = \sum_{n=0}^{N} np^n$,使得$\sum_{n=0}^{N} p^n = 1$

[0218] 可以证明,当满足以下标准时,W被精确地最小化:

[0219] $p_{i_0}\geq p_{i_1}\geq\cdots\geq p_{i_N}$对于所有$i_0\in H^0, i_1\in H^1,\cdots i_n\in H^N$(*)

[0220] 假设码已被分配给概率,使得它们满足上述准则(*),第一选择是由任意两个汉明权桶索引i和j在{0,1,…N}中,,wlog j≤k并且选择是由来自这些桶的任意两个索引$i_j\in H^j$和$i_k\in H^k$构成,分别具有概率p_{i_j}和p_{i_k}。这给出:

[0221] $k=j+1, l\in N_0$

[0222] $(*)\Longrightarrow p_{i_j} = p_{i_k} + \epsilon, \epsilon\in R_{\geq 0}$

[0223] 交换概率i_j和i_k给出新的平均汉明权W′,其大于或等于W如下:

$$W' = \sum_{n=0}^{N} \sum_{i \in H^n} p_i H_i = \Sigma + p_{i_j} H_{i_k} + p_{i_k} H_{i_j}$$

$$= \Sigma + p_{i_j} k + p_{i_k} j$$

[0224]
$$= \Sigma + (p_{i_k} + \epsilon)(j + l) + p_{i_k} j$$

$$= \Sigma + p_{i_k} j + p_{i_k} l + \epsilon j + \epsilon l + p_{i_k} j$$

$$= \Sigma + p_{i_k}(j + l) + (p_{i_k} + \epsilon) j + \epsilon l$$

$$= \Sigma + p_{i_k} k + p_{i_j} j + \epsilon l$$

$$= W + \epsilon l \geq W$$

[0225] 因此W′≥W具有相等性的条件是，当且仅当k＝j(即，两个编码值已经具有相同的汉明权)或者$p_{i_j} = p_{i_k}$(即，两个编码值具有相同的概率)。在任一情况下，转置码也满足(*)。

[0226] 因此，已经表明，以(*)给出的准则确定了给定概率分布的H.W最小编码的集合，并且都是局部最小值。通过考虑具有不以这种方式排序的概率的任何其他编码，相同的逻辑清楚的是，存在对一对概率的转置，其将平均汉明权减少某个非0量-因此它们实际上是所有全局最小值。

[0227] 如果使用上述第二方法(即，图16的位翻转方法)作为减少编码值的平均汉明权的替换方案(即，如果设置了一半以上的位，则所有位被翻转，并且填充位之一被标记以标识这一点)，则由于以固定方式修改编码值而不管数据的概率分布如何，所以需要特定类型的数据有效。这是具有向具有少量1或少量0的码偏斜的概率分布的数据，并且相反地，最差种类的数据是具有向具有类似数目的0和1的码偏斜的概率的那些数据。带符号数据非常适合有效类型的分布，因此去相关数据也应该有益于所述方法。

[0228] 这可以由以下逻辑明确地示出，其中N＝L+1即，单个填充位用于标记，并且N和L分别是具有和不具有填充位标记的码的长度：

[0229]
$$W' = \sum_{n=0}^{\lfloor L/2 \rfloor} \sum_{i \in H^n} p_i H_i + \sum_{n=\lfloor L/2 \rfloor + 1}^{L} \sum_{i \in H^n} p_i (N - H_i)$$

[0230] 这在以下情况下是最佳的：

[0231] $p_{i_0} \geq p_{i_1} \geq \cdots \geq p_{i_{\lfloor L/2 \rfloor}}$对于所有$i_0 \in H^0, i_1 \in H^1 \cup H^L, \ldots i_{\lfloor L/2 \rfloor}$

[0232] $\in H^{\lfloor L/2 \rfloor} \cup H^{L - \lfloor L/2 \rfloor}$

[0233] 图20示出其中可以实现本文所述的方法的计算机系统2000。计算机系统包括CPU 2002(其可对应于图18中所示且如上所述的CPU 1800)和GPU 2004(其可对应于图19中所示且如上所述的GPU1900)。另外，计算机系统2000包括存储器2006和其他设备2014，诸如显示器2016、扬声器2018和照相机2020。处理框910(对应于处理框110)在GPU904上实现。计算机系统的部件通过通信总线2024彼此进行通信。

[0234] 图18和19的处理器和图20的计算机系统被示为包括多个功能块。这仅是示意性的，并不旨在限定此类实体的不同逻辑元件之间的严格划分。每个功能块可以任何合适的方式提供。

[0235] 本文所描述的处理器可体现在集成电路上的硬件中。本文所描述的处理器可配置成执行本文所描述的任何方法。一般来讲，上文所述的功能、方法、技术或部件中的任一者可在软件、固件、硬件(例如，固定逻辑电路系统)或其任何组合中实现。本文可以使用术语“模块”、“功能”、“部件”、“元件”、“单元”、“块”和“逻辑”来概括地表示软件、固件、硬件或其任何组合。在软件实现方式的情况下，模块、功能、部件、元件、单元、块或逻辑表示程序码，所述程序码在处理器上执行时执行指定任务。本文中所描述的算法和方法可以由执行码的一个或多个处理器执行，所述码使处理器执行算法/方法。计算机可读存储介质的示例包括随机存取存储器(RAM)、只读存储器(ROM)、光盘、闪存存储器、硬盘存储器以及可以使用磁性、光学和其他技术来存储指令或其他数据并且可以由机器存取的其他存储器装置。

[0236] 如本文中所使用的术语计算机程序码和计算机可读指令是指供处理器执行的任何种类的可执行码，包含以机器语言、解释语言或脚本语言表达的码。可执行码包含二进制码、机器码、字节码、定义集成电路的码(例如硬件描述语言或网表)，以及用例如C、Java或OpenCL等编程语言码表达的码。可执行码可以是例如任何种类的软件、固件、脚本、模块或库，当在虚拟机或其他软件环境中被适当地执行、处理、解释、编译、运行时，这些软件、固件、脚本、模块或库使支持可执行码的计算机系统的处理器执行由所述码指定的任务。

[0237] 处理器、计算机或计算机系统可以是任何种类的装置、机器或专用电路，或其集合或一部分，它具有处理能力使得可以执行指令。处理器可以是任何种类的通用或专用处理器，例如CPU、GPU、片上系统、状态机、媒体处理器、专用集成电路(ASIC)、可编程逻辑阵列、现场可编程门阵列(FPGA)、物理处理单元(PPU)、无线电处理单元(RPU)、数字信号处理器(DSP)、通用处理器(例如通用GPU)、微处理器、旨在加速CPU之外的任务的任何处理单元等。计算机或计算机系统可以包括一个或多个处理器。本领域技术人员将认识到，这种处理能力被结合到许多不同的设备中，因此术语“计算机”包括机顶盒、媒体播放器、数字收音机、PC、服务器、移动电话、个人数字助理和许多其他设备。

[0238] 本发明还意图包围限定如本文中所描述的硬件的配置的软件，例如硬件描述语言(HDL)软件，用于设计集成电路或用于配置可编程芯片以执行所要功能。即，可提供一种计算机可读存储介质，其上编码有呈集成电路定义数据集形式的计算机可读程序码，所述集成电路定义数据集在集成电路制造系统中处理(即，运行)时将所述系统配置成制造被配置为执行本文所描述的任何方法的处理器，或制造包括本文所描述的任何设备的处理系统。集成电路定义数据集可以是例如集成电路描述。

[0239] 因此，可以提供一种在集成电路制造系统处制造被配置为执行如本文所述的方法之一的处理器或其他硬件逻辑的方法。此外，可以提供一种集成电路定义数据集，当在集成电路制造系统中处理时，其使得执行制造这种处理器或硬件逻辑的方法。在各种示例中，可在软件/固件中实现编码方法(例如，使用图5A或5B中以及图6A或6B中的框202加上迭代过程)。在其他示例中，所述方法可实现为使用数据结构(例如，一组门或表条目阵列)来编码所述映射的部分或全部(作为LUT)的固定功能硬件单元及/或一组连接的算术单元以执行图5A或5B及图6A或6B的方法。

[0240] 集成电路定义数据集可呈计算机码形式，例如作为网表、用于配置可编程芯片的码，作为在任何层级定义集成电路的硬件描述语言，包含作为寄存器传送级(RTL)码、作为例如Verilog或VHDL的高级电路表示，和作为例如OASIS(RTM)和GDSII的低级电路表示。在

逻辑上定义集成电路的更高级表示法(例如RTL)可以在配置成在软件环境的上下文中生成集成电路的制造定义的计算机系统处处理,所述软件环境包括电路元件的定义和用于组合那些元件以便生成由所述表示法定义的集成电路的制造定义的规则。如通常软件在计算机系统处执行以便定义机器的情况一样,可能需要一个或多个中间用户步骤(例如,提供命令、变量等),以便将计算机系统配置成生成集成电路的制造定义,以执行定义集成电路以便生成所述集成电路的制造定义的码。

[0241] 现在将参照图21描述在集成电路制造系统处处理集成电路定义数据集以便配置该系统来制造被配置为执行如本文所述的方法之一的处理器或其他硬件逻辑的示例。

[0242] 图21示出被配置为制造被配置为执行如本文中描述的方法中的一者的处理器或其他硬件逻辑的集成电路(IC)制造系统2102的示例。特别地,IC制造系统2102包括布局处理系统2104和集成电路生成系统2106。IC制造系统2102被配置为接收IC定义数据集(例如,定义如本文的任何示例中所述的处理器1800、1900),处理IC定义数据集,并根据IC定义数据集生成IC(例如,体现被布置为执行如本文的任何示例中所述的方法的硬件逻辑)。IC定义数据集的处理配置IC制造系统2102以制造体现硬件逻辑的集成电路,该硬件逻辑被布置成执行如本文的任何示例中所述的方法。

[0243] 布局处理系统2104被配置为接收和处理IC定义数据集以确定电路布局。根据IC定义数据集确定电路布局的方法在本领域中是已知的,并且例如可以涉及合成RTL码以确定要生成的电路的门级表示,例如就逻辑部件(例如NAND、NOR、AND、OR、MUX和FLIP-FLOP部件)而言。通过确定逻辑部件的位置信息,可以根据电路的门级表示来确定电路布局。这可以自动完成或者在用户参与下完成,以便优化电路布局。当布局处理系统2104已经确定电路布局时,其可将电路布局定义输出到IC生成系统2106。电路布局定义可以是例如电路布局描述。

[0244] 如本领域已知的,IC生成系统2106根据电路布局定义来生成IC。例如,IC生成系统2106可实现生成IC的半导体装置制造工艺,其可涉及光刻和化学处理步骤的多步骤序列,在此期间,在由半导体材料制成的晶片上逐渐形成电子电路。电路布局定义可呈掩码的形式,其可以在光刻工艺中用于根据电路定义来生成IC。替代地,提供给IC生成系统1006的电路布局定义可呈计算机可读码的形式,IC生成系统1006可使用所述计算机可读码来形成用于生成IC的合适掩码。

[0245] 由IC制造系统2102执行的不同过程可全部在一个位置例如由一方来实施。替代地,IC制造系统2102可以是分布式系统,使得一些过程可在不同位置执行,并且可由不同方来执行。例如,以下阶段中的一些可以在不同位置和/或由不同方来执行:(i)合成表示IC定义数据集的RTL码,以形成要生成的电路的门级表示;(ii)基于门级表示来生成电路布局;(iii)根据电路布局来形成掩码;以及(iv)使用掩码来制造集成电路。

[0246] 在其他示例中,在集成电路制造系统处对集成电路定义数据集的处理可以将该系统配置为制造被布置为执行如本文所述的方法的硬件逻辑,而无需处理IC定义数据集以便确定电路布局。例如,集成电路定义数据集可以定义例如FPGA的可重新配置的处理器的配置,并且对所述数据集进行的处理可以将IC制造系统配置成(例如,通过将配置数据加载到FPGA)生成具有所述定义的配置的可重新配置的处理器。

[0247] 在一些实施方案中,当在集成电路制造系统中处理时,集成电路制造定义数据集

可以使集成电路制造系统生成如本文中描述的装置。例如，通过集成电路制造定义数据集，以上文关于图21描述的方式对集成电路制造系统的配置，可制造出如本文中所述的装置。

[0248] 在一些示例中，集成电路定义数据集可包括在数据集处定义的硬件上运行的软件，或者与在数据集处定义的硬件组合运行的软件。在图21所示的示例中，IC生成系统还可以由集成电路定义数据集另外配置，以在制造集成电路时根据在集成电路定义数据集中定义的程序码将固件加载到所述集成电路上，或者以其他方式向集成电路提供与集成电路一起使用的程序码。

[0249] 本领域技术人员将认识到用来存储程序指令的存储装置可分布在网络上。例如，远程计算机可以将所描述的过程的示例存储为软件。本地或终端计算机可以访问远程计算机，并下载软件的一部分或全部以运行程序。替代地，本地计算机可以根据需要下载软件的片段，或者在本地终端处执行一些软件指令，而在远程计算机(或计算机网络)处执行另一些软件指令。本领域技术人员还将认识到通过利用本领域技术人员已知的常规技术，软件指令的全部或一部分可以由诸如DSP、可编程逻辑阵列等的专用电路执行。

[0250] 本文描述的方法可以由配置有存储在有形存储介质上的机器可读形式的软件的计算机执行，例如，软件采用包括用于将计算机配置为执行所述方法的组成部分的计算机可读程序码的计算机程序的形式，或采用包括计算机程序码装置的计算机程序的形式，当程序在计算机上运行时以及在计算机程序可以在计算机可读存储介质上实现的情况下，所述码装置适于执行本文所述任何方法的所有步骤。有形(或非暂时性)存储介质的示例包括磁盘、拇指驱动器、存储卡等，并且不包括传播信号。软件可适于在并行处理器或串行处理器上执行，使得所述方法步骤可以任何适当顺序执行或同时执行。

[0251] 本文描述的硬件部件可以由其上编码有计算机可读程序码的非暂时性计算机可读存储介质生成。

[0252] 存储用于实现所公开方面的机器可执行数据的存储器可以是非暂时性介质。非暂时性介质可以是易失性或非易失性的。易失性非暂时性介质的示例包括基于半导体的存储器，诸如SRAM或DRAM。可用于实现非易失性存储器的技术的示例包括光学和磁存储器技术、闪存、相变存储器、电阻RAM。

[0253] 对“逻辑”的特定引用是指执行一个或多个功能的结构。逻辑的示例包括被布置成执行这些功能的电路。例如，这种电路可以包括在制造过程中可用的晶体管和/或其他硬件元件。这种晶体管和/或其他元件可用于形成实现和/或包含存储器的电路或结构，例如寄存器、触发器或锁存器，逻辑运算器，例如布尔运算，数学运算器，例如加法器、乘法器，或者，移位器和互连，作为示例。这些元件可以作为定制电路或标准单元库、宏来提供或在其他抽象级别提供。这些元件可以特定布置互连。逻辑可以包括具有固定功能的电路，并且电路可以被编程为执行一个或多个功能；可以从固件或软件更新或控制机制提供这样的编程。被标识为执行一个功能的逻辑还可以包括实现组成性功能或子过程的逻辑。在一个示例中，硬件逻辑具有实现一个或多个固定功能操作，状态机或过程的电路。

[0254] 与已知的实现方式相比，在本申请中阐述的概念在装置、设备、模块和/或系统中(以及在本文中实现的方法中)的实现方式可以引起性能改进。性能改进可以包含计算性能提高、等待时间缩短、处理量增大和/或功耗降低中的一个或多个。在制造此类装置、设备、模块和系统(例如在集成电路中)期间，可以在性能改进与物理实施方案之间进行权衡，从

而改进制造方法。例如，可以在性能改进与布局面积之间进行权衡，从而匹配已知实现方式的性能，但使用更少的硅。例如，这可以通过以串行方式重复使用功能块或在装置、设备、模块和/或系统的元件之间共享功能块来完成。相反，在本申请中阐述的引起装置、设备、模块和系统的物理实现方式的改进(诸如硅面积减小)的概念可以针对性能提高进行权衡。这例如可以通过在预定义面积预算内制造模块的多个实例来完成。

[0255] 如对本领域技术人员显而易见的，可以延长或改变本文给出的任何范围或设备值而不丧失所寻求的效果。

[0256] 应当理解，上述益处和优点可以涉及一个实施方案，或者可以涉及多个实施方案。实施方案不限于解决任何或所有所述问题的那些或具有任何或所有所述益处和优点的那些。

[0257] 对“一个”项目的任何引用都指的是这些项目中的一个或多个。术语“包括”在本文中用于表示包括所标识的方法块或元件，但是这些块或元件不包括排他列表，并且设备可包括附加的块或元件，并且方法可包括附加的操作或元件。此外，并不暗示块、元件和操作本身是关闭的。

[0258] 本文所述的方法的步骤可以任何合适的顺序或在适当时被同时执行。图中框之间的箭头示出方法步骤的一个示例序列，但并不旨在排除其他序列或并行执行多个步骤。另外，在不脱离本文描述的主题的实质和范围的情况下，可以从任何方法中删除单个块。上述任何示例的一些方面可以与所描述的任何其他示例的一些方面组合以形成进一步的示例而不会丧失所寻求的效果。在图的元件被示出通过箭头连接的情况下，应当理解，这些箭头仅示出元件之间的通信(包括数据和控制消息)的一个示例流向。元件之间的流向可以是任一方向或两个方向。

[0259] 申请人据此独立地公开了本文中所描述的每个单独的特征以及两个或更多个此类特征的任意组合，到达的程度使得此类特征或组合能够根据本领域的技术人员的普通常识基于本说明书整体来实行，而不管此类特征或特征的组合是否解决本文中所公开的任何问题。鉴于前文描述，本领域的技术人员将清楚，可以在本发明的范围内进行各种修改。

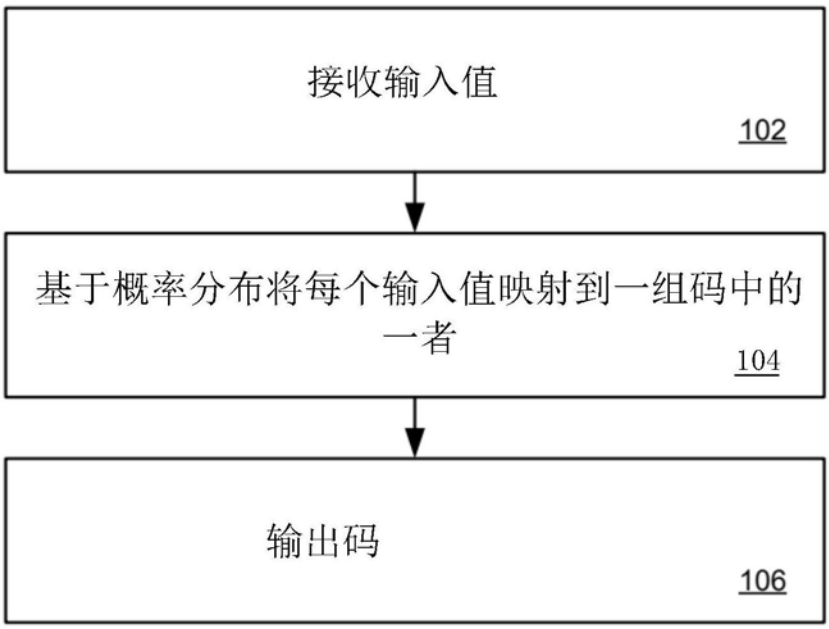

接收输入值
102
基于概率分布将每个输入值映射到一组码中的一者
104
输出码
106

图1

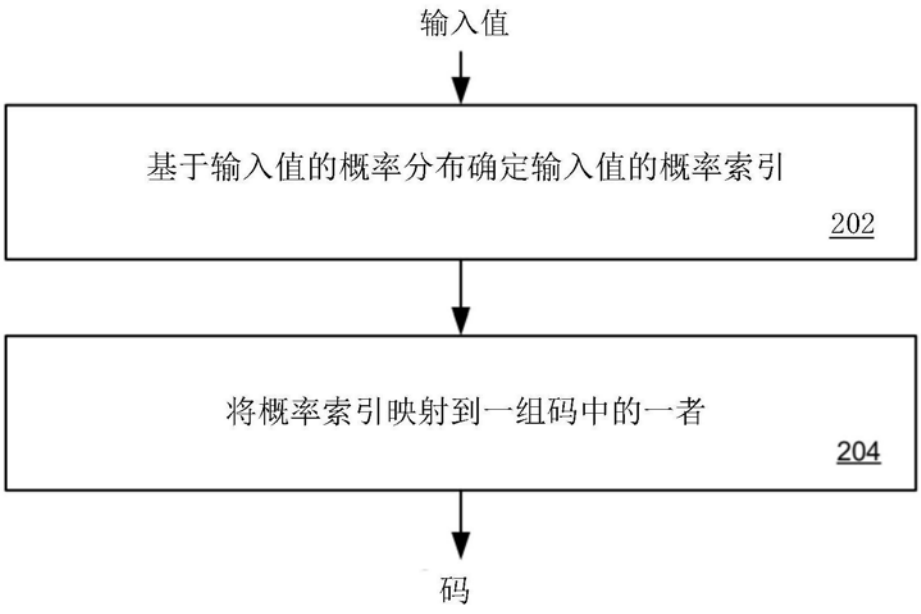
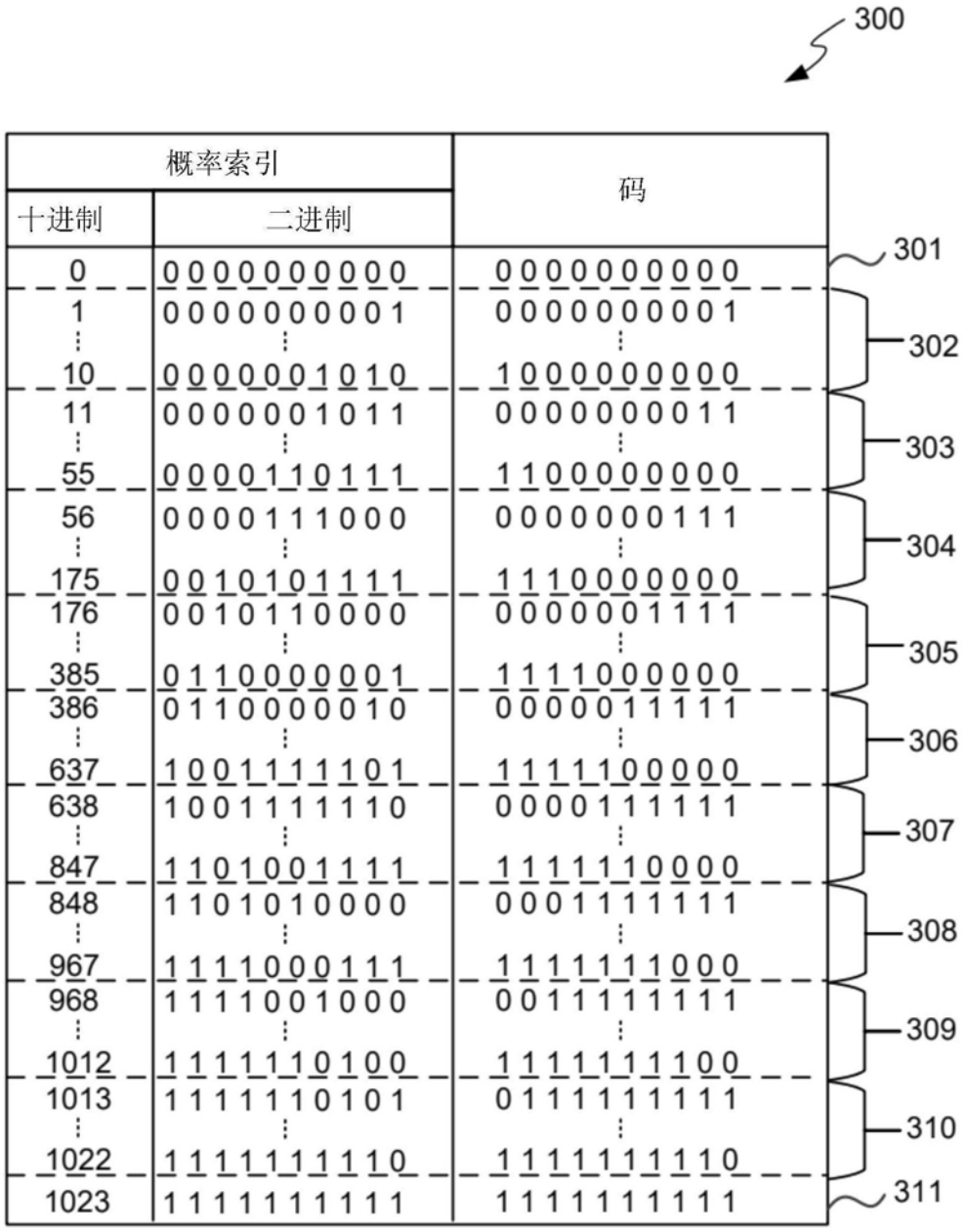

输入值
基于输入值的概率分布确定输入值的概率索引
202
将概率索引映射到一组码中的一者
204
码

图2



概率索引		码	
十进制	二进制		
0	0000000000	0000000000	301
1	0000000001	0000000001	302
⋮	⋮	⋮	
10	0000001010	1000000000	
11	0000001011	0000000011	303
⋮	⋮	⋮	
55	0001101111	1100000000	
56	0000111000	0000000111	304
⋮	⋮	⋮	
175	0010101111	1110000000	
176	0010110000	0000001111	305
⋮	⋮	⋮	
385	0110000001	1111000000	
386	0110000010	0000011111	306
⋮	⋮	⋮	
637	1001111101	1111100000	
638	1001111110	0000111111	307
⋮	⋮	⋮	
847	1101001111	1111110000	
848	1101010000	0001111111	308
⋮	⋮	⋮	
967	1111001111	1111111000	
968	1111001000	0011111111	309
⋮	⋮	⋮	
1012	1111101100	1111111100	
1013	1111110101	0111111111	310
⋮	⋮	⋮	
1022	1111111110	1111111110	
1023	1111111111	1111111111	311

图3

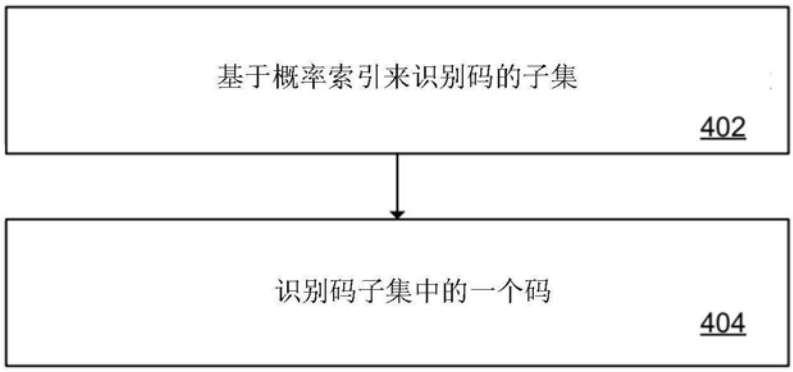

基于概率索引来识别码的子集
402
识别码子集中的一个码
404

图4

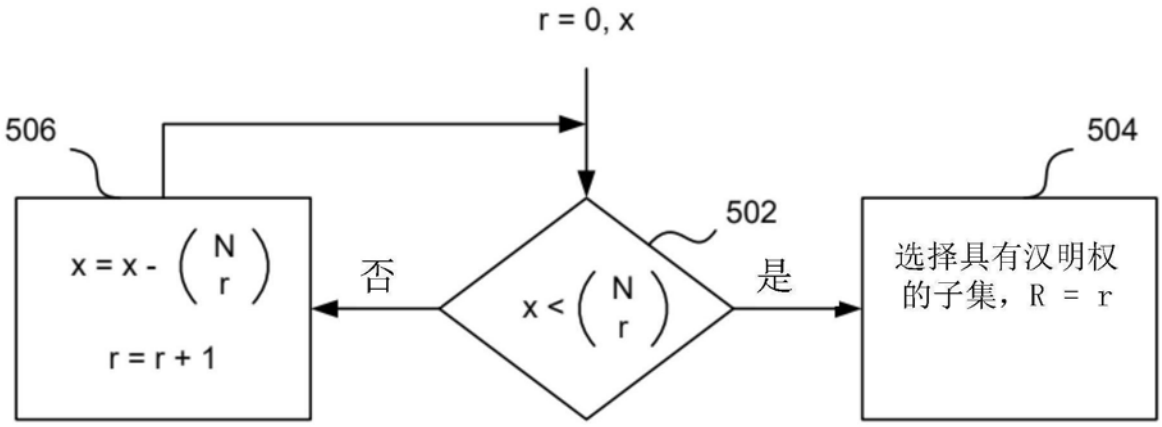

r = 0, x
506
504
502
x = x - (N r)
r = r + 1
否
x < (N r)
是
选择具有汉明权的子集，R = r

图5A

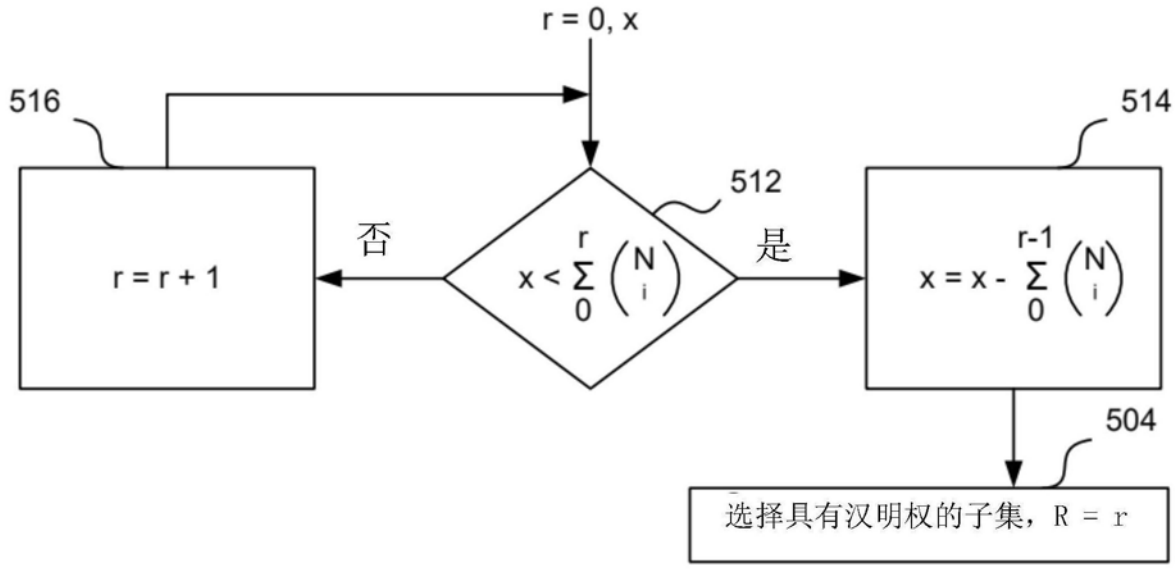

r = 0, x
516
514
512
r = r + 1
否
x < Σ(0 to r) (N i)
是
x = x - Σ(0 to r-1) (N i)
504
选择具有汉明权的子集，R = r

图5B

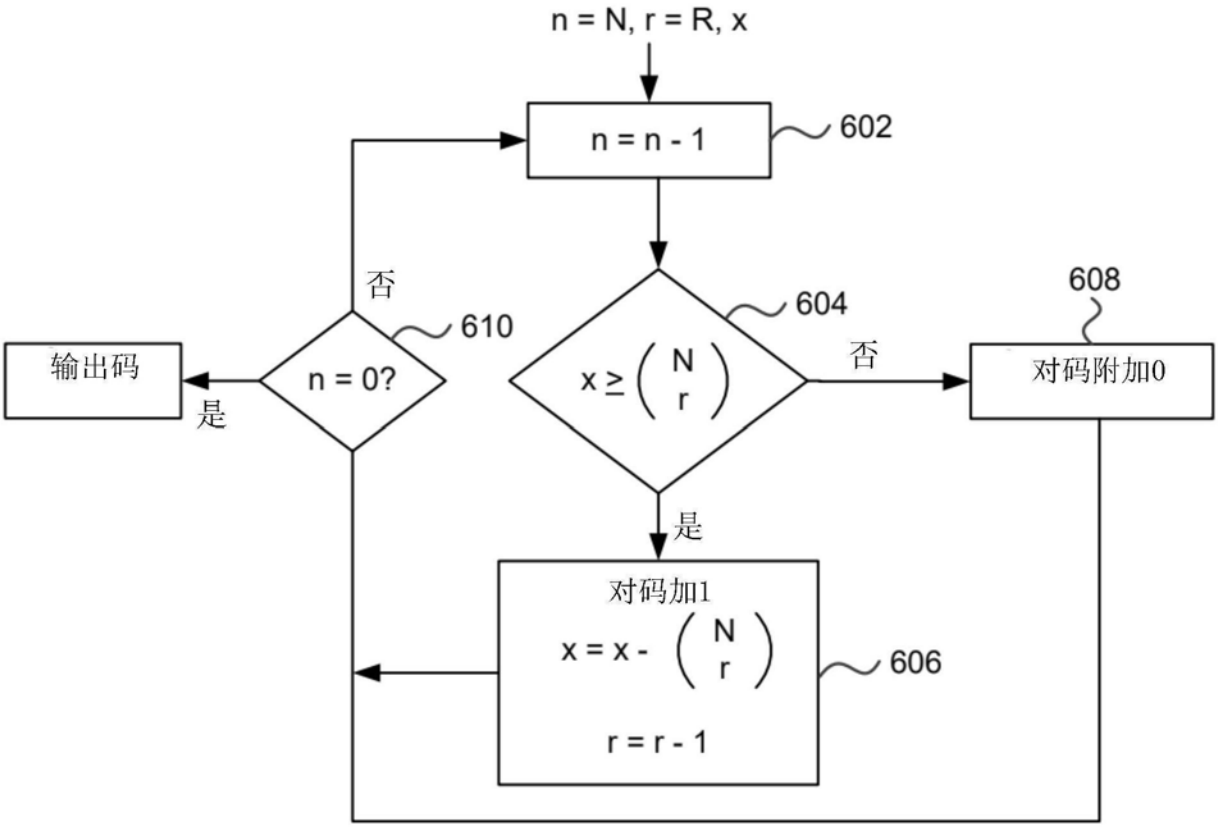


图6A

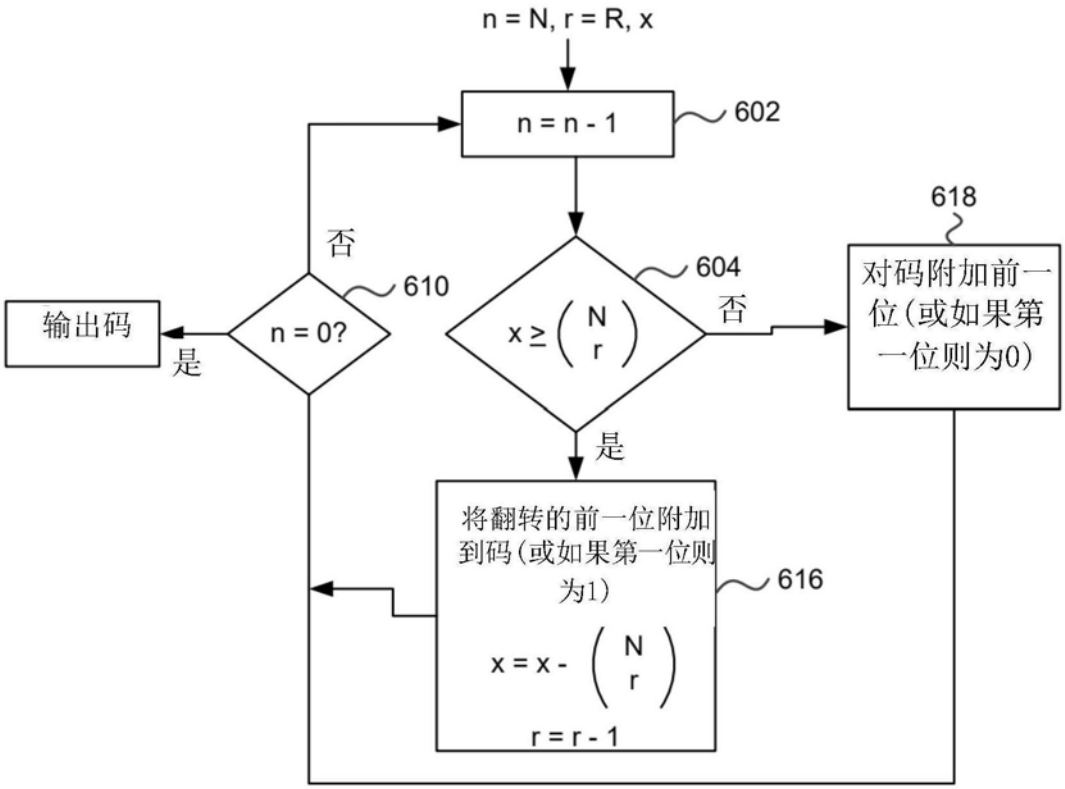


图6B

r/n	0		1		2		3		4		5		6		7		8		9		R
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
	1		1		1		1		1		1		1		1		1		1		
1	0	0	0	1	0	2	0	3	0	4	0	5	0	6	0	7	0	8	0	9	1
	1		1		1		1		1		1		1		1		1		1		
2	0	0	0	0	0	1	0	3	0	6	0	10	0	15	0	21	0	28	0	36	2
	1		1		1		1		1		1		1		1		1		1		
3	0	0	0	0	0	0	0	1	0	4	0	10	0	20	0	35	0	56	0	84	3
	1		1		1		1		1		1		1		1		1		1		
4	0	0	0	0	0	0	0	0	0	1	0	5	0	15	0	35	0	70	0	126	4
	1		1		1		1		1		1		1		1		1		1		
5	0	0	0	0	0	0	0	0	0	0	0	1	0	6	0	21	0	56	0	126	5

图7

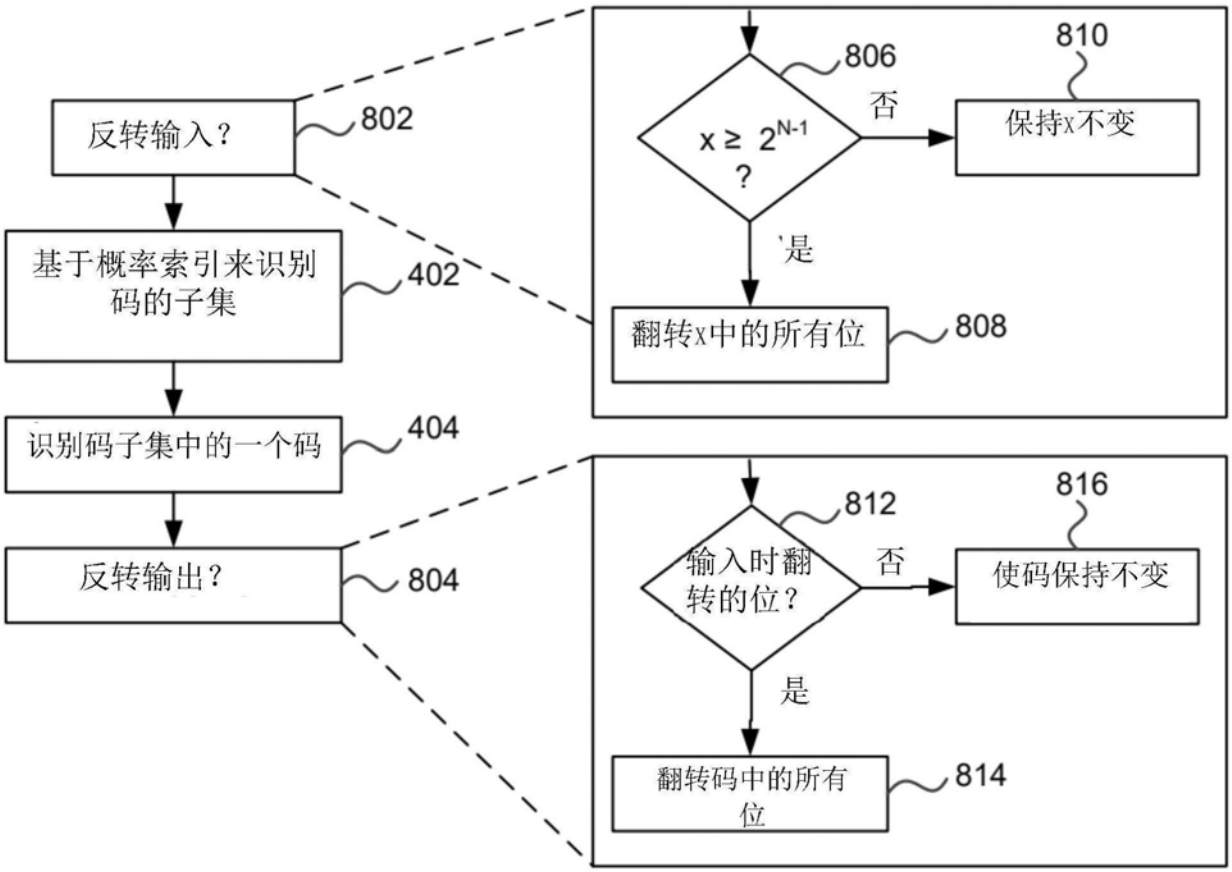

反转输入？
802
基于概率索引来识别码的子集
402
识别码子集中的一个码
404
反转输出？
804
806
x ≥ 2^N-1 ?
否
保持x不变
810
是
翻转x中的所有位
808
812
输入时翻转的位？
否
使码保持不变
816
是
翻转码中的所有位
814

图8A

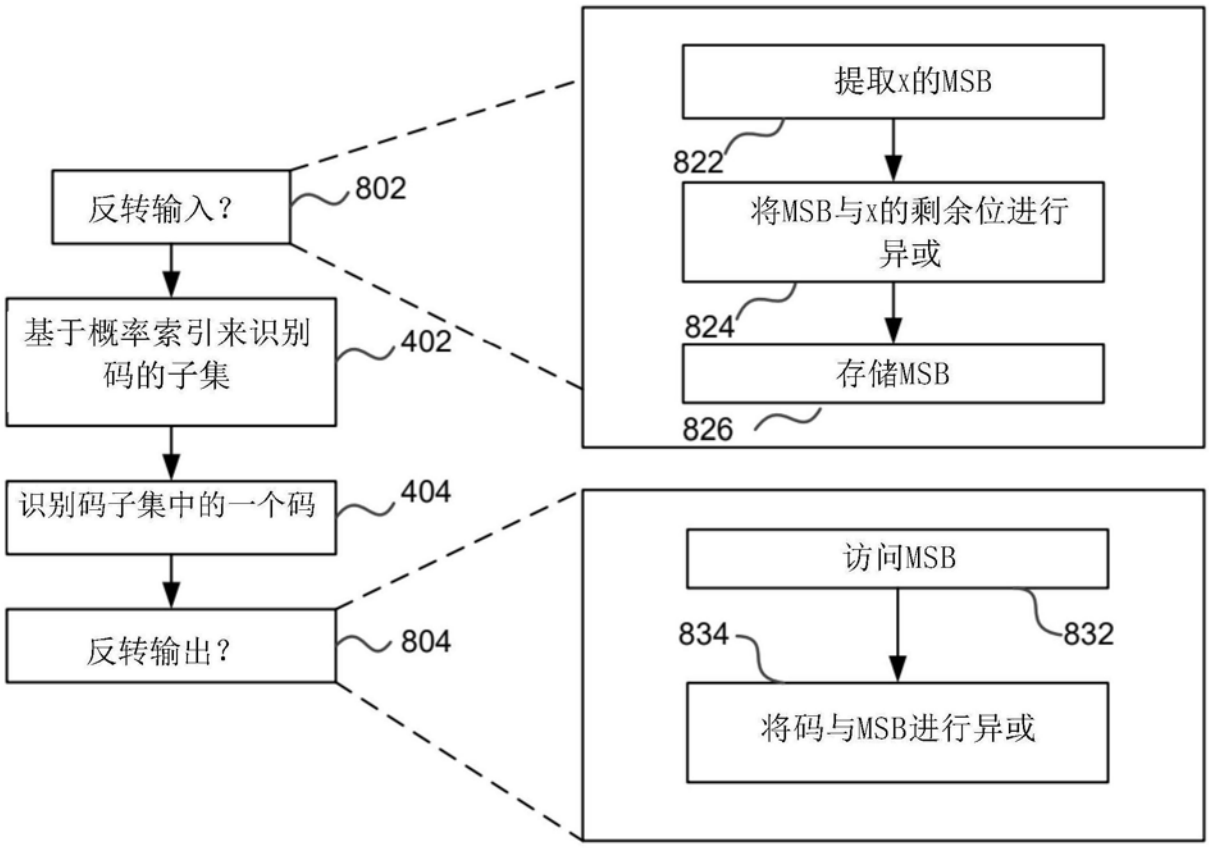

反转输入？
802
基于概率索引来识别码的子集
402
识别码子集中的一个码
404
反转输出？
804
提取x的MSB
822
将MSB与x的剩余位进行异或
824
存储MSB
826
访问MSB
832
834
将码与MSB进行异或

图8B

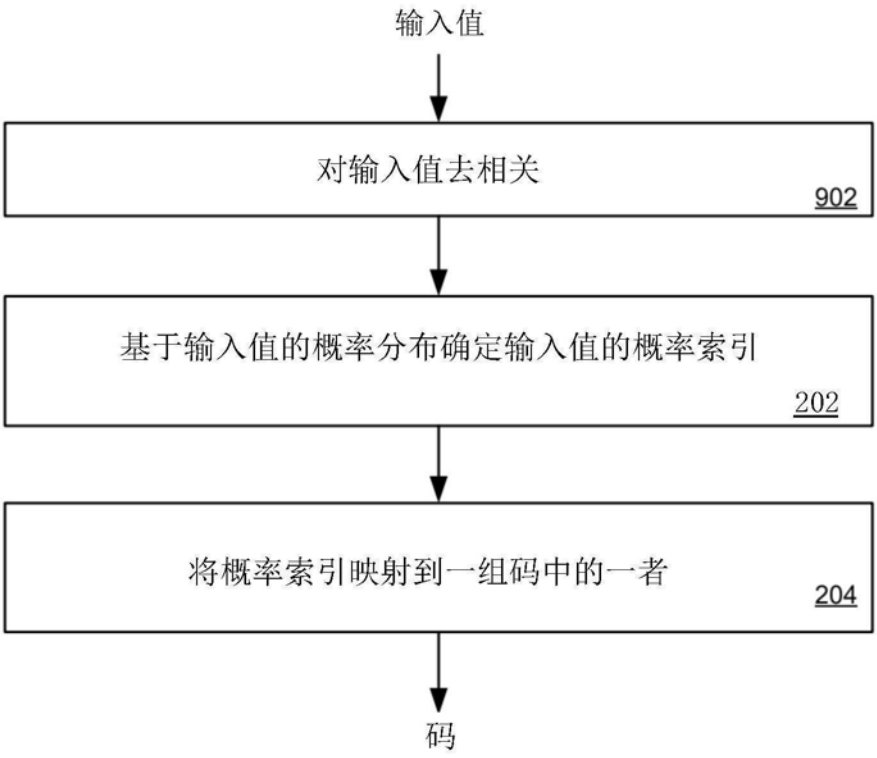

输入值
对输入值去相关
902
基于输入值的概率分布确定输入值的概率索引
202
将概率索引映射到一组码中的一者
204
码

图9

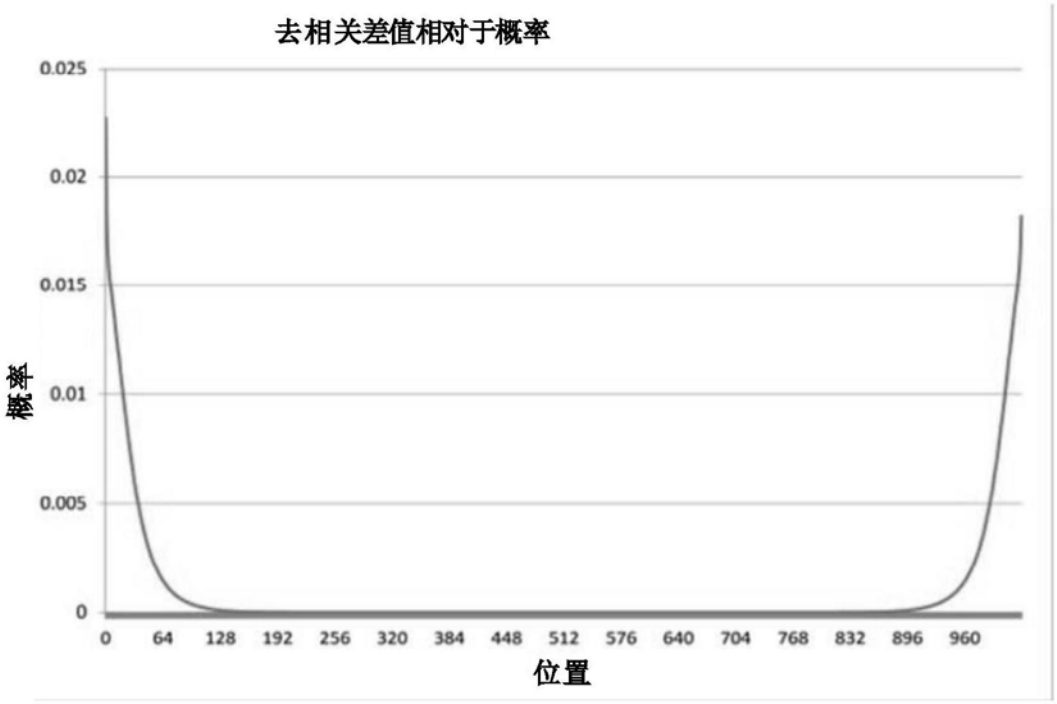

去相关差值相对于概率
概率
0.025
0.02
0.015
0.01
0.005
0
0 64 128 192 256 320 384 448 512 576 640 704 768 832 896 960
位置

图10A

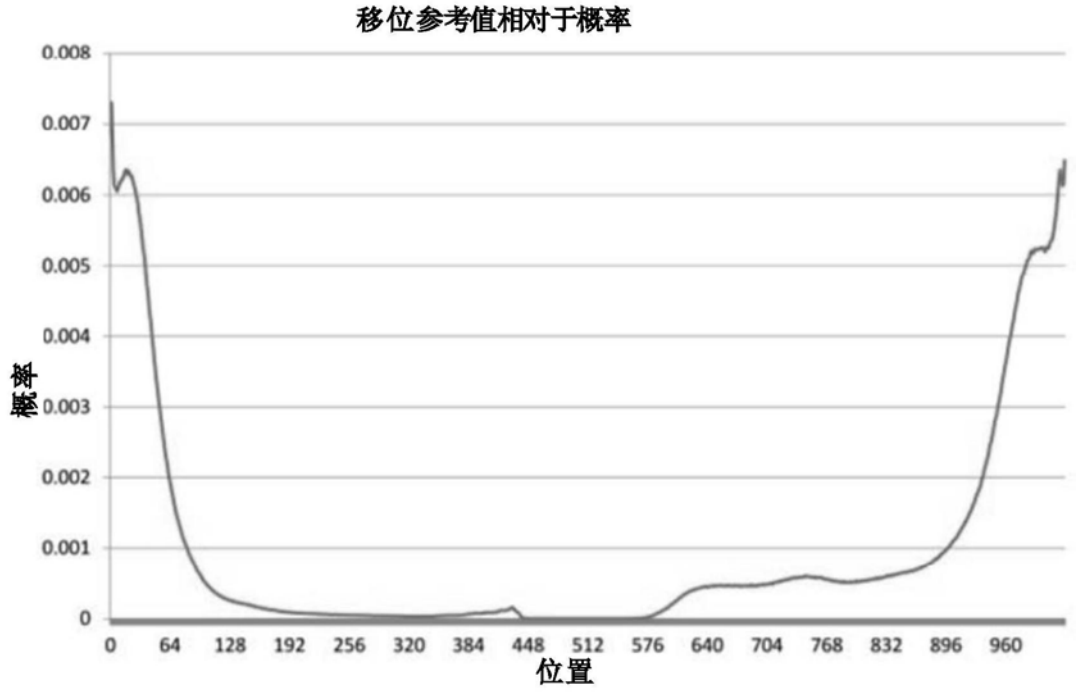

移位参考值相对于概率
概率
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
0 64 128 192 256 320 384 448 512 576 640 704 768 832 896 960
位置

图10B

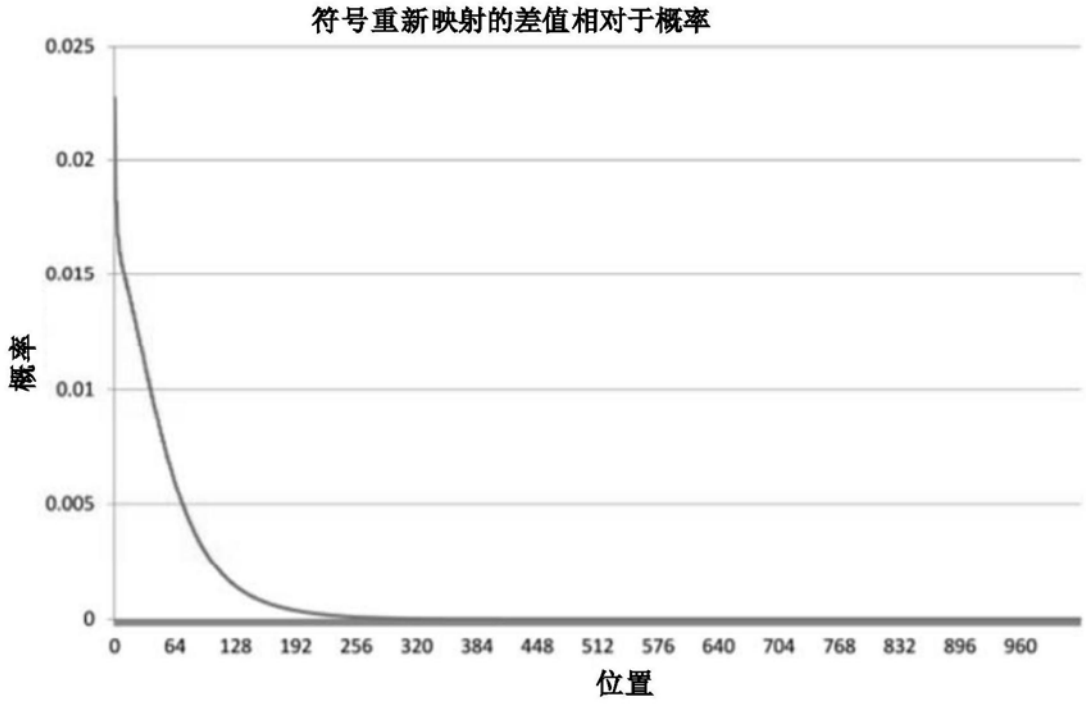

符号重新映射的差值相对于概率
概率
0.025
0.02
0.015
0.01
0.005
0
0 64 128 192 256 320 384 448 512 576 640 704 768 832 896 960
位置

图11A

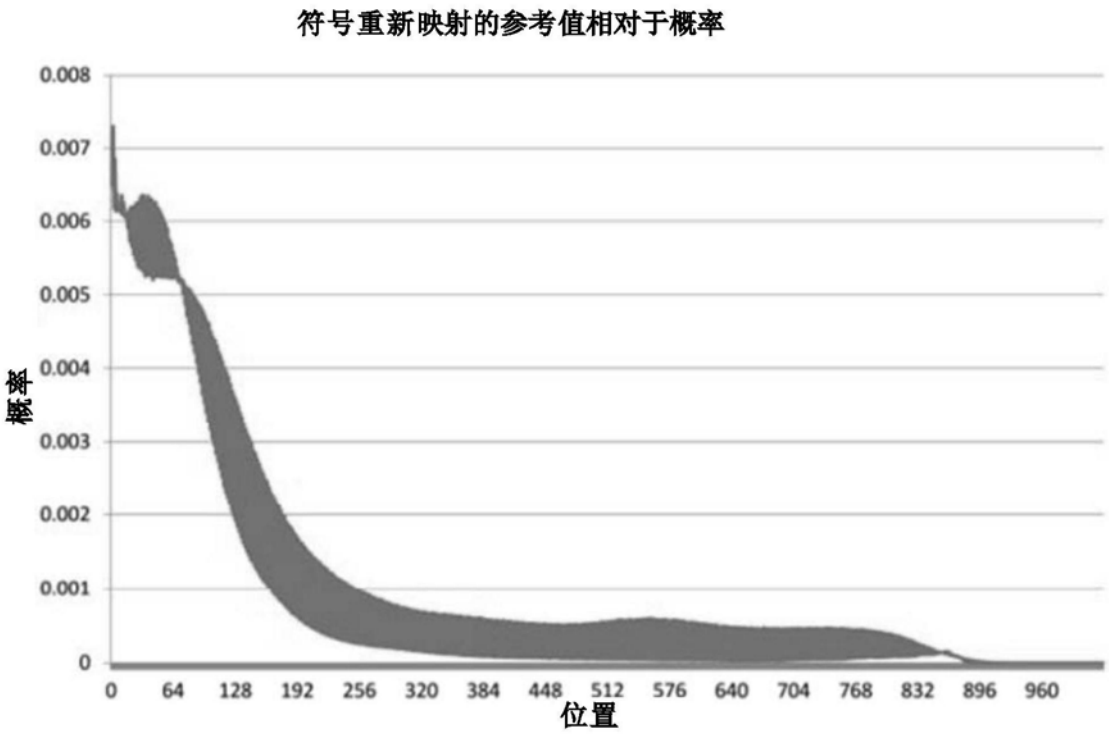

符号重新映射的参考值相对于概率
概率
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
0 64 128 192 256 320 384 448 512 576 640 704 768 832 896 960
位置

图11B

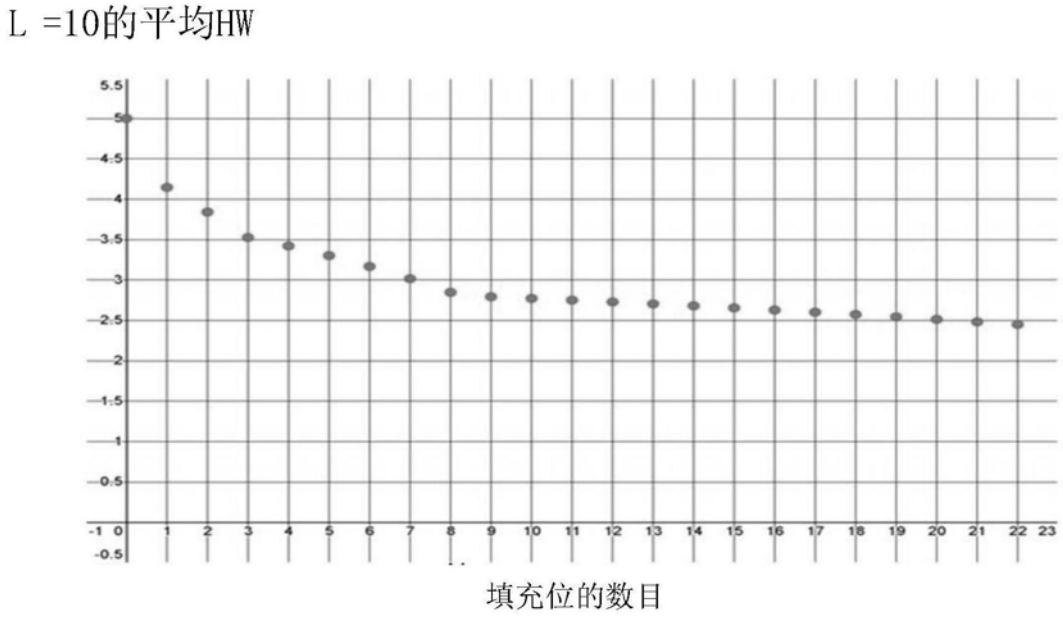

L =10的平均HW
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
-0.5
-1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
填充位的数目

图12

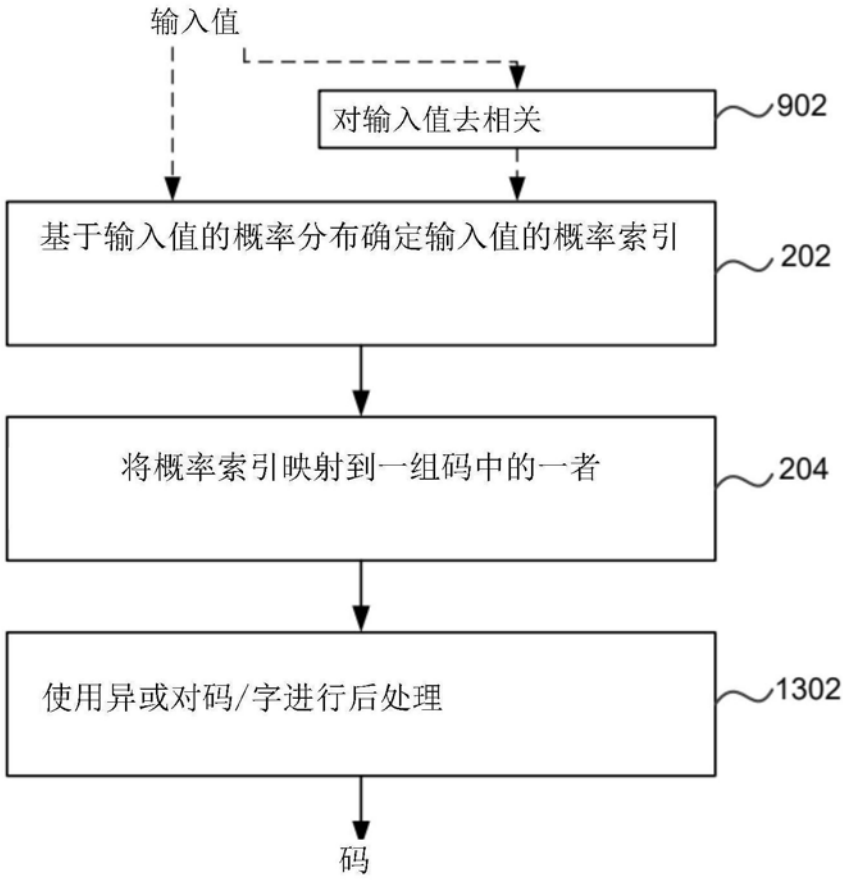
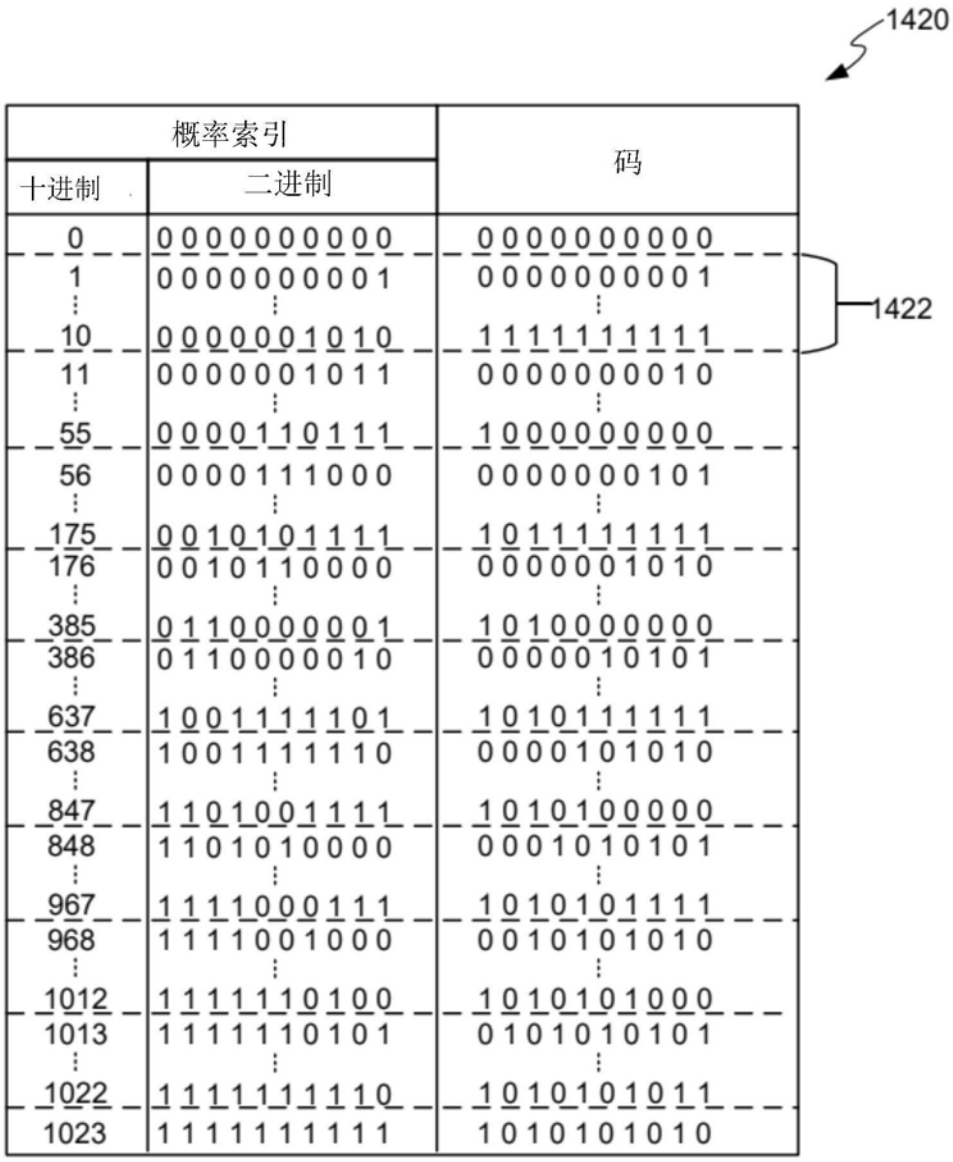

输入值
对输入值去相关
902
基于输入值的概率分布确定输入值的概率索引
202
将概率索引映射到一组码中的一者
204
使用异或对码/字进行后处理
1302
码

图13



概率索引		码
十进制	二进制	
0	0000000000	0000000000
1	0000000001	0000000001
⋮	⋮	⋮
10	0000001010	1111111111
11	0000001011	0000000010
⋮	⋮	⋮
55	0000110111	1000000000
56	0000111000	0000000101
⋮	⋮	⋮
175	0010101111	1011111111
176	0010110000	0000001010
⋮	⋮	⋮
385	0110000001	1010000000
386	0110000010	0000010101
⋮	⋮	⋮
637	1001111101	1010111111
638	1001111110	0000101010
⋮	⋮	⋮
847	1101001111	1010100000
848	1101010000	0001010101
⋮	⋮	⋮
967	1111000111	1010101111
968	1111001000	0010101010
⋮	⋮	⋮
1012	1111101000	1010101000
1013	1111110101	0101010101
⋮	⋮	⋮
1022	1111111110	1010101011
1023	1111111111	1010101010

图14A

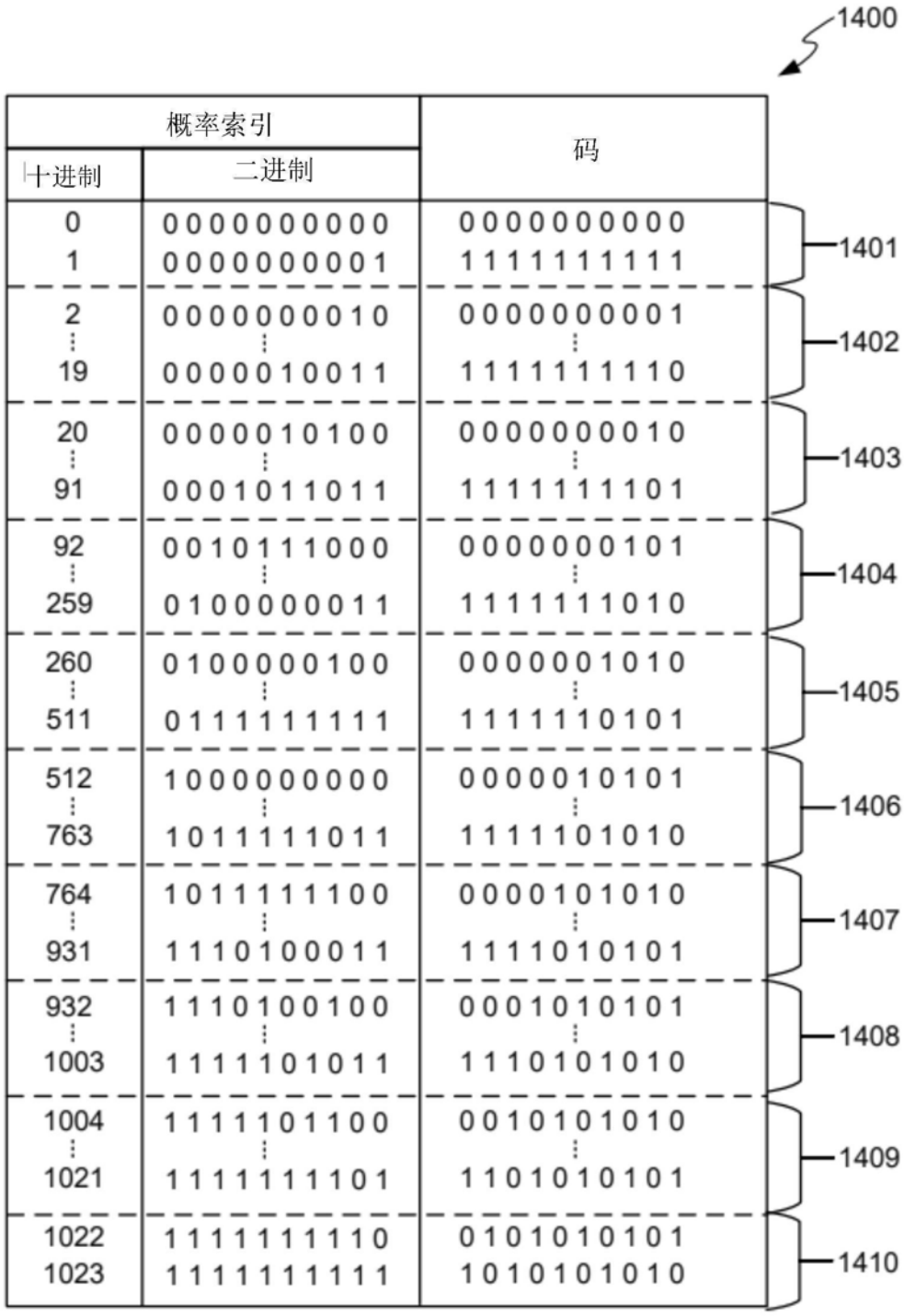

1400
概率索引
十进制
二进制
码
0 0000000000 0000000000
1 0000000001 1111111111
1401
2 0000000010 0000000001
19 0000010011 1111111110
1402
20 0000010100 0000000010
91 0001011011 1111111101
1403
92 0010111000 0000000101
259 0100000011 1111111010
1404
260 0100000100 0000001010
511 0111111111 1111110101
1405
512 1000000000 0000010101
763 1011111011 1111101010
1406
764 1011111100 0000101010
931 1110100011 1111010101
1407
932 1110100100 0001010101
1003 1111101011 1110101010
1408
1004 1111101100 0010101010
1021 1111111101 1101010101
1409
1022 1111111110 0101010101
1023 1111111111 1010101010
1410

图14B

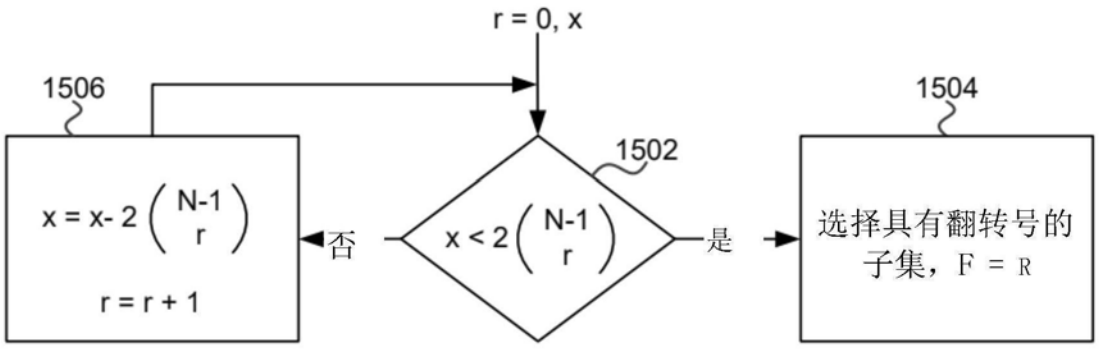

r = 0, x
1506
1504
1502
x = x- 2 (N-1 r)
r = r + 1
否
x < 2 (N-1 r)
是
选择具有翻转号的子集，F = R

图15A

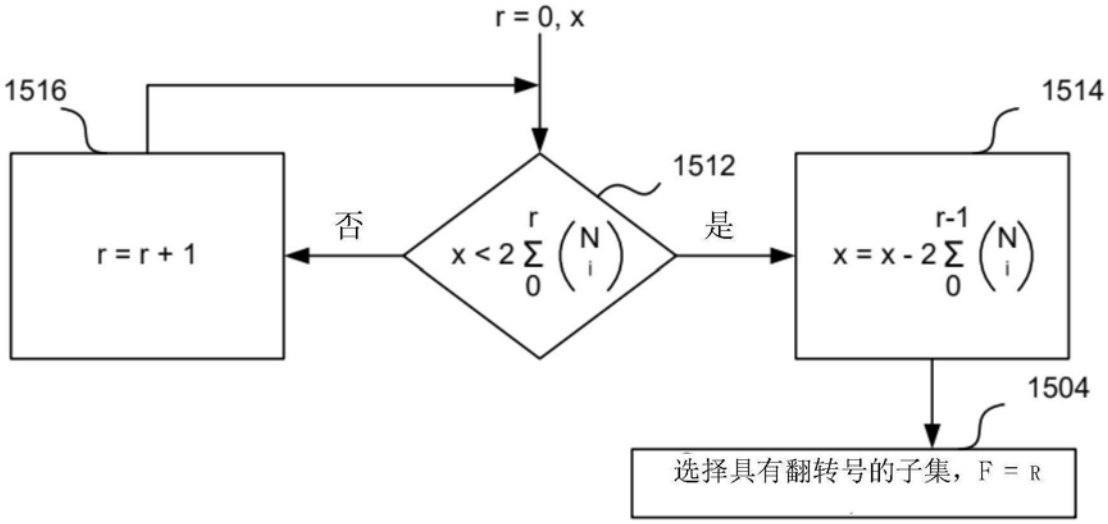

r = 0, x
1516
1514
1512
r = r + 1
否
x < 2 r 0 ∑ (N i)
是
x = x - 2 r-1 0 ∑ (N i)
1504
选择具有翻转号的子集，F = R

图15B

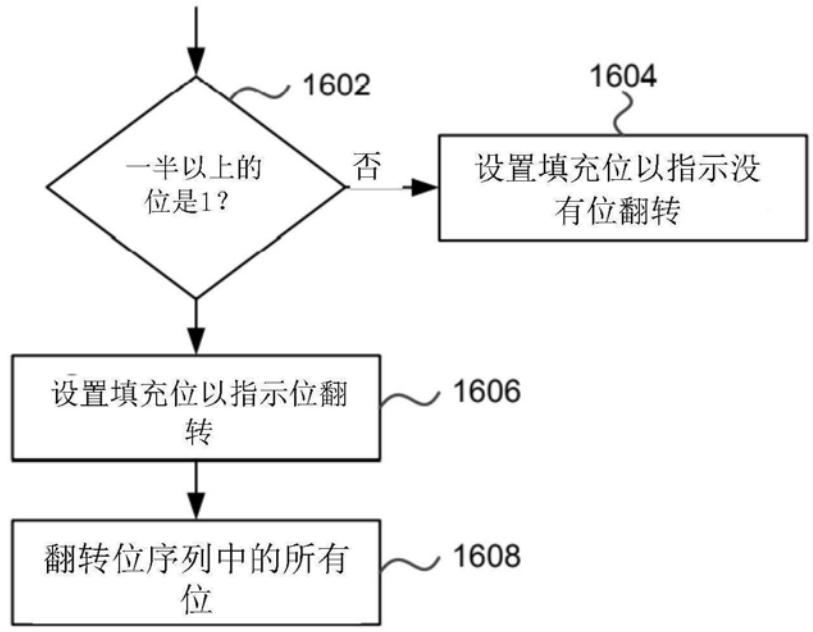

1602
一半以上的位是1?
否
1604
设置填充位以指示没有位翻转
设置填充位以指示位翻转
1606
翻转位序列中的所有位
1608

图16

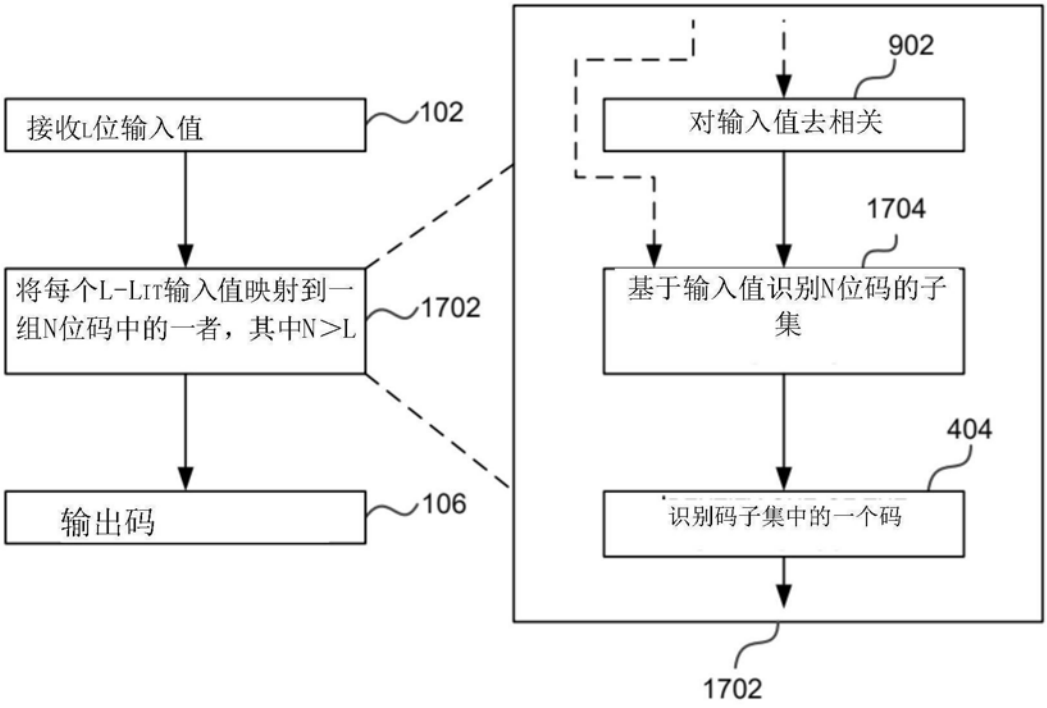

接收L位输入值
102
将每个L-LIT输入值映射到一组N位码中的一者，其中N>L
1702
输出码
106
902
对输入值去相关
1704
基于输入值识别N位码的子集
404
识别码子集中的一个码
1702

图17

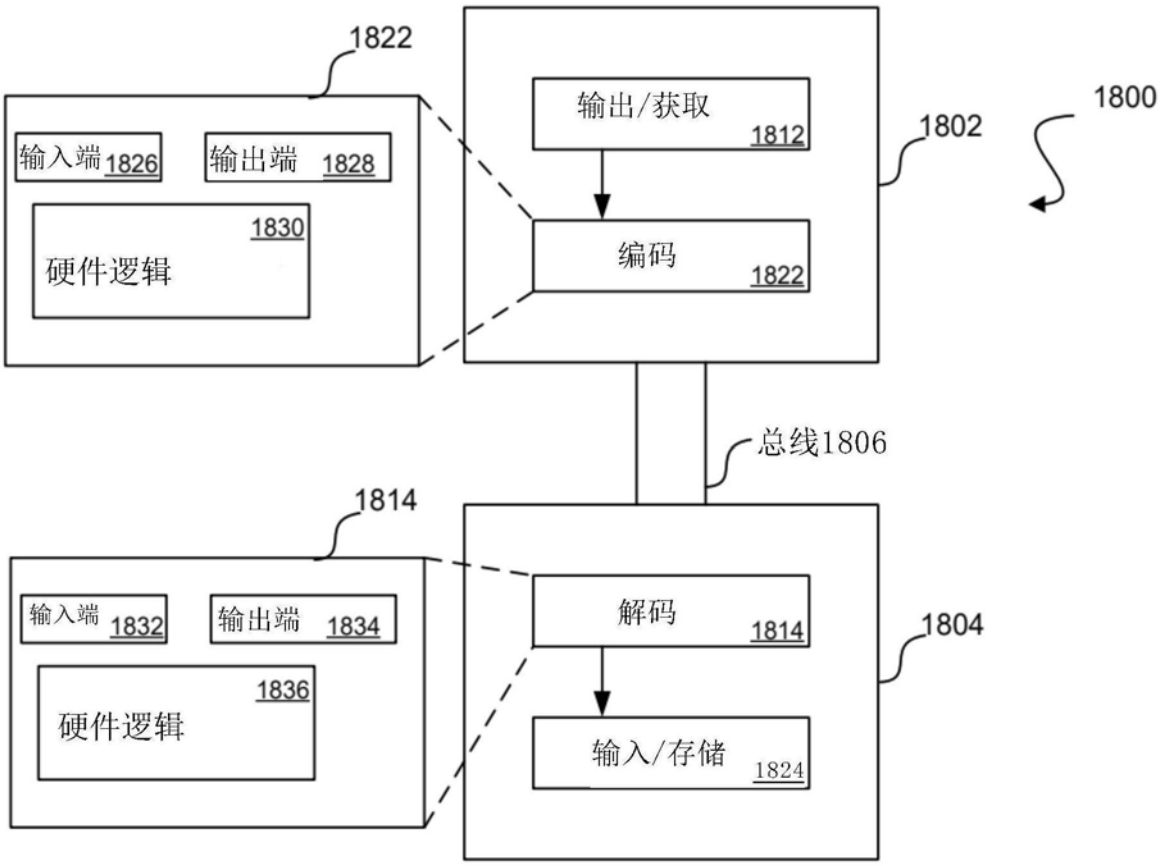

1822
输入端 1826
输出端 1828
1830
硬件逻辑
输出/获取
1812
编码
1822
1802
1800
总线1806
1814
输入端 1832
输出端 1834
1836
硬件逻辑
解码
1814
输入/存储
1824
1804

图18

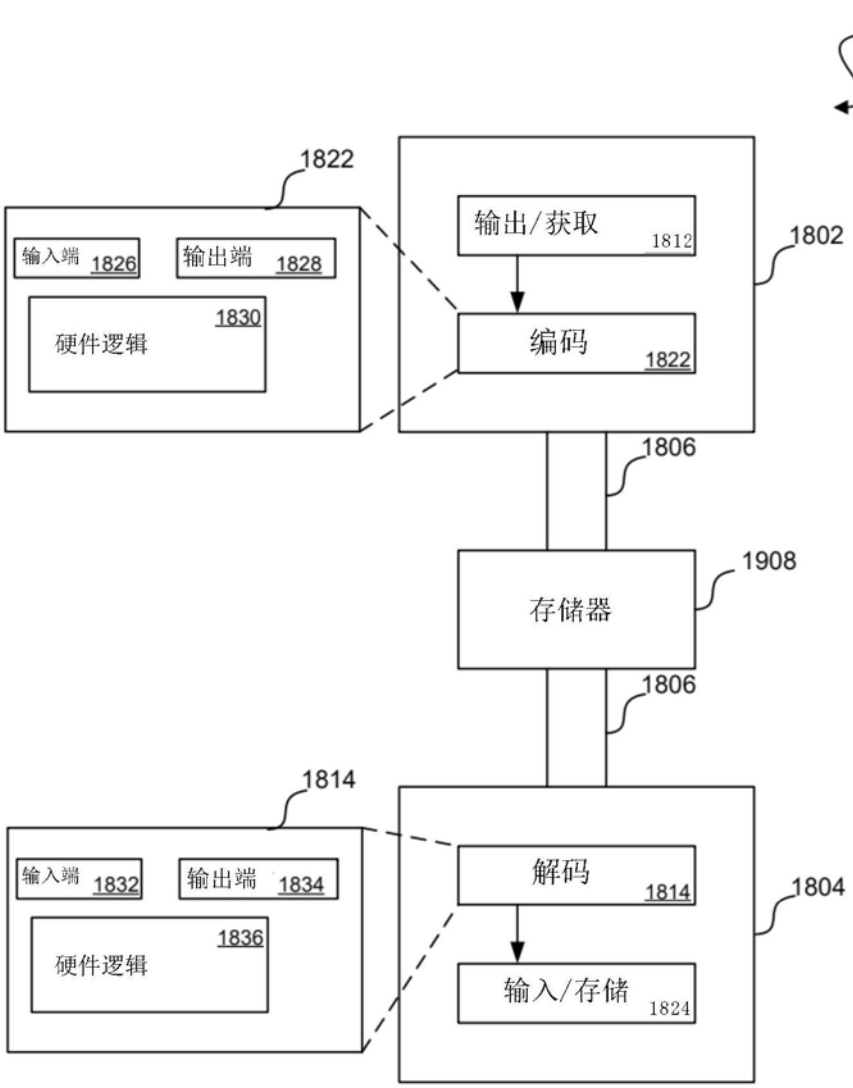

1900
1822
输入端 1826
输出端 1828
1830
硬件逻辑
输出/获取 1812
编码 1822
1802
1806
存储器
1908
1806
1814
输入端 1832
输出端 1834
1836
硬件逻辑
解码 1814
输入/存储 1824
1804

图19

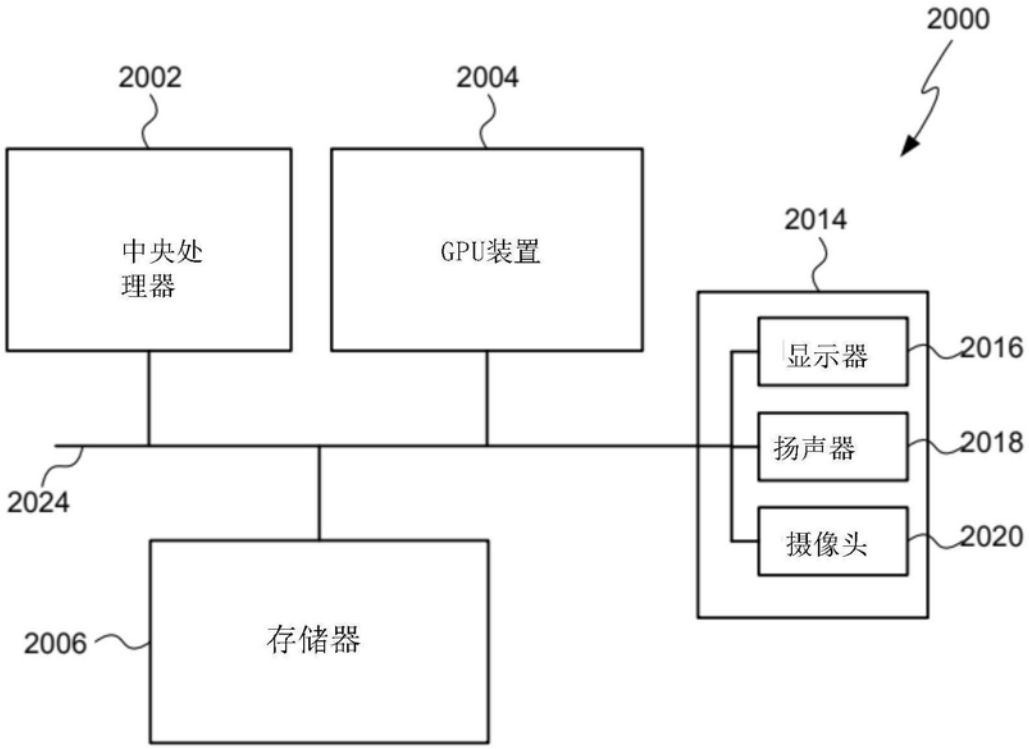

2000
2002
2004
中央处理器
GPU装置
2014
显示器
2016
扬声器
2018
摄像头
2020
2024
2006
存储器

图20

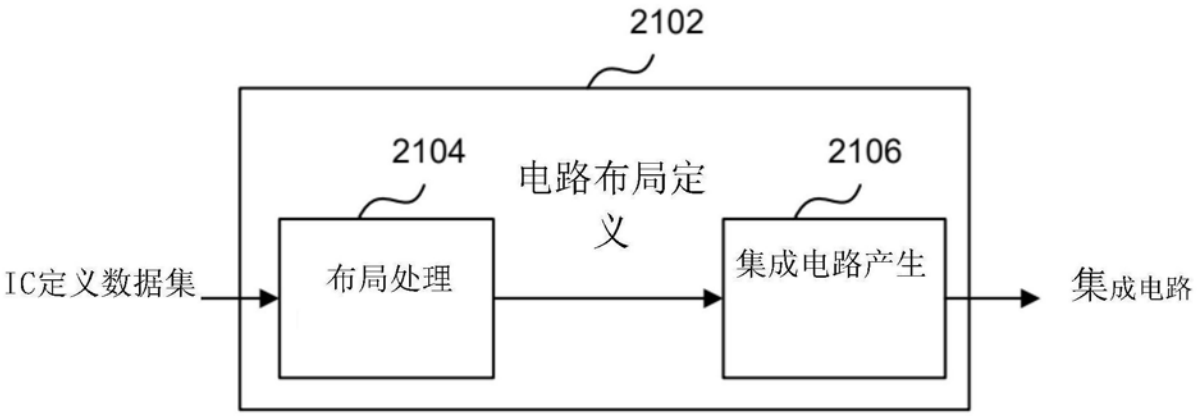

2102
2104
电路布局定义
2106
IC定义数据集
布局处理
集成电路产生
集成电路

图21

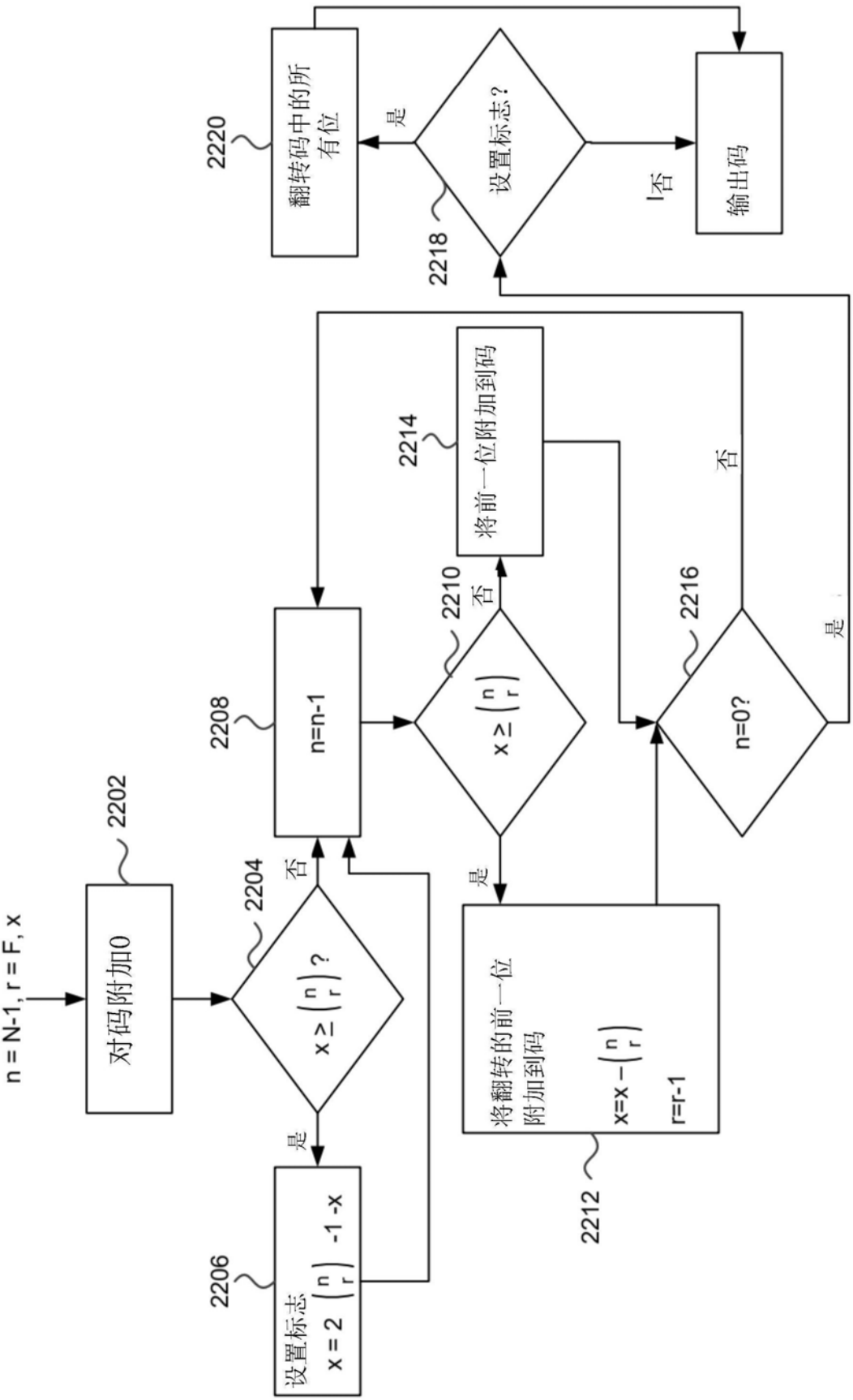

n = N-1, r = F, x
对码附加0
2202
2204
$x \geq \binom{n}{r}$?
否
是
2206
设置标志
$x = 2\binom{n}{r} - 1 - x$
2208
n=n-1
2210
$x \geq \binom{n}{r}$
否
是
2212
将翻转的前一位
附加到码
$x = x - \binom{n}{r}$
r=r-1
2214
将前一位附加到码
2216
n=0?
否
是
2218
设置标志?
是
否
2220
翻转码中的所
有位
输出码

图22

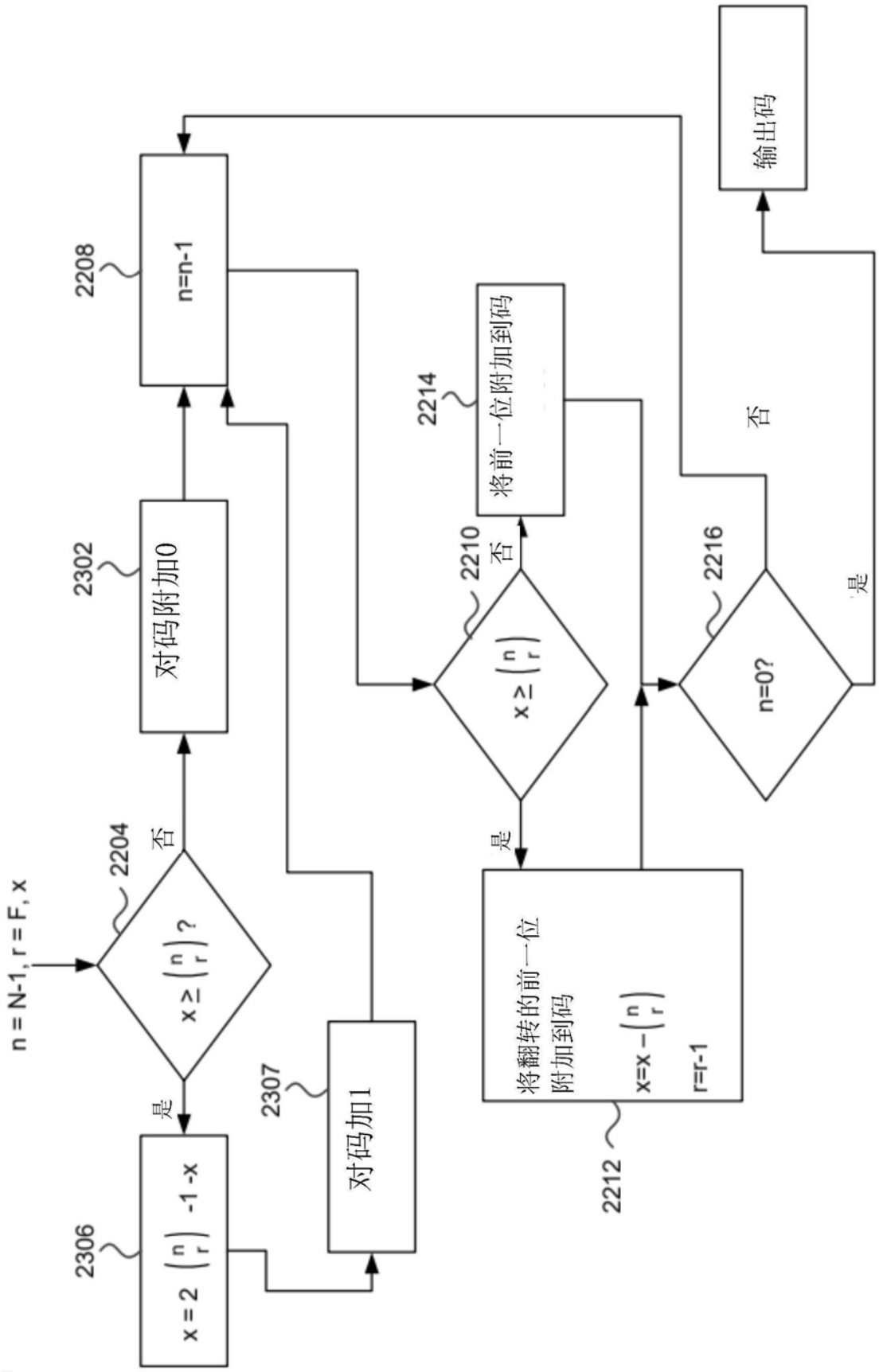

n = N-1, r = F, x
2204
$x \geq \binom{n}{r}$?
否
是
2306
$x = 2\binom{n}{r} - 1 - x$
2307
对码加1
2302
对码附加0
2208
n=n-1
2210
$x \geq \binom{n}{r}$
否
是
2214
将前一位附加到码
2212
将翻转的前一位
附加到码
$x = x - \binom{n}{r}$
r=r-1
2216
n=0?
否
是
输出码

图23

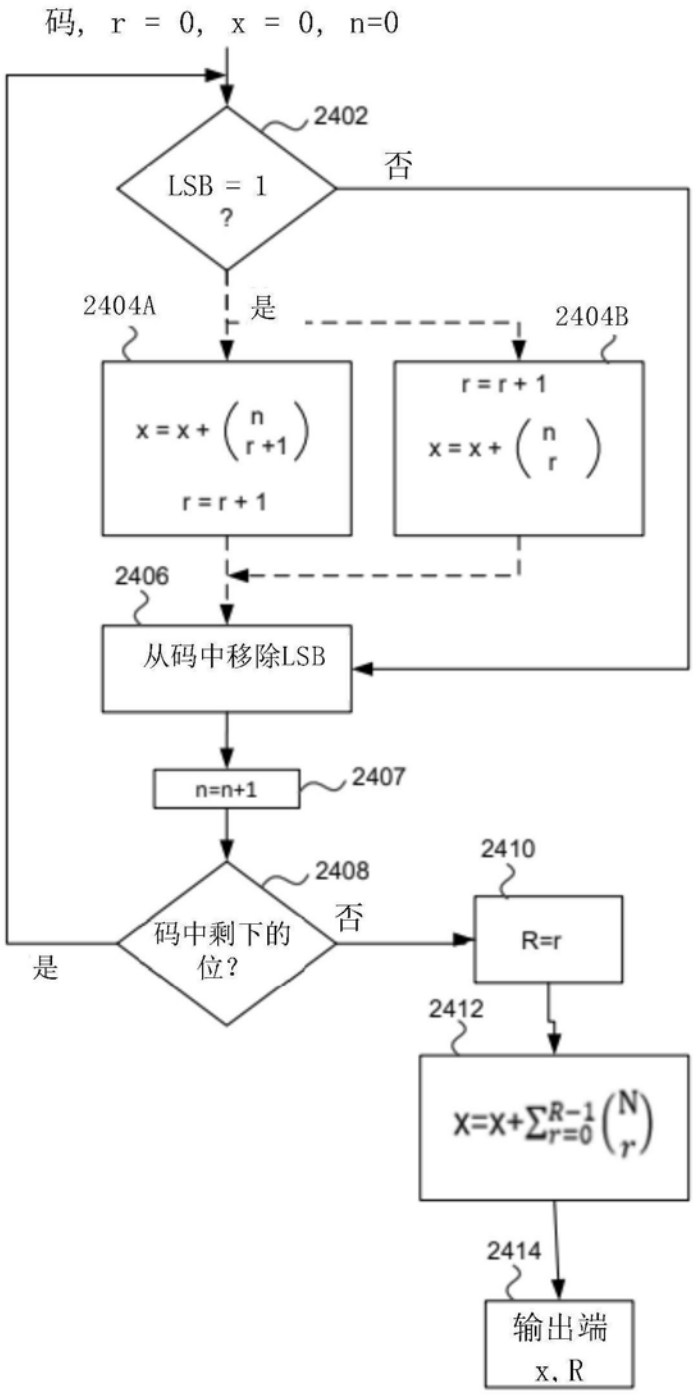

码, r = 0, x = 0, n=0
2402
LSB = 1 ?
否
是
2404A
2404B
x = x + (n r+1)
r = r + 1
r = r + 1
x = x + (n r)
2406
从码中移除LSB
n=n+1
2407
2408
码中剩下的位?
否
是
2410
R=r
2412
X=X+∑r=0R-1 (N r)
2414
输出端
x、R

图24