(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 105359418 B
(45)授权公告日 2018.11.09

(21)申请号 201480024634.9

(22)申请日 2014.02.27

(65)同一申请的已公布的文献号
申请公布号 CN 105359418 A

(43)申请公布日 2016.02.24

(30)优先权数据
1303661.1 2013.03.01 GB

(85)PCT国际申请进入国家阶段日
2015.10.30

(86)PCT国际申请的申请数据
PCT/EP2014/000510 2014.02.27

(87)PCT国际申请的公布数据
WO2014/131517 EN 2014.09.04

(73)专利权人 古如罗技微系统公司
地址 芬兰土尔库市里南路34号,邮编20100

(72)发明人 奥西·卡雷沃

(74)专利代理机构 北京英赛嘉华知识产权代理有限责任公司 11204
代理人 王达佐 王艳春

(51)Int.Cl.
H03M 7/30(2006.01)

(56)对比文件
US 2008/0030384 A1,2008.02.07,
CN 101795407 A,2010.08.04,
CN 101299611 A,2008.11.05,
CN 101202548 A,2008.06.18,

审查员 葛运滨

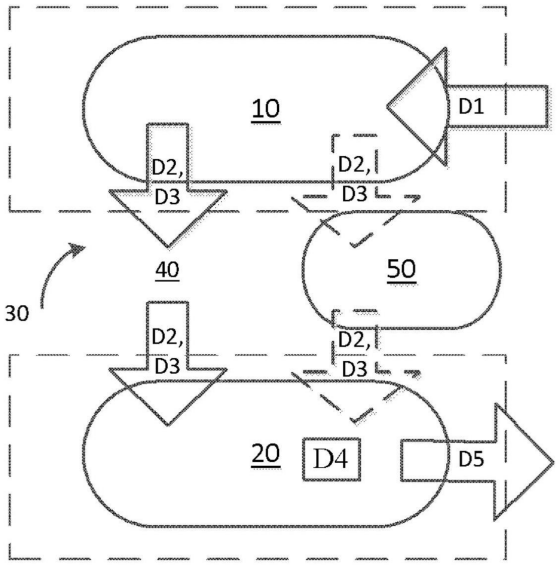
权利要求书4页 说明书24页 附图3页

(54)发明名称

编码器、解码器和编码解码方法

(57)摘要

一种编码器(10),用于编码包括一个数值序列的输入数据(D1),以生成相应的编码输出数据(D2或D3),编码器(10)包括一个数据处理装置,用于差分和/或总和编码形式的输入数据(D1),以生成一个或多个相应的编码序列,其中一个或多个相应的编码序列经历围绕最大值和/或围绕最小值,生成编码输出数据(D2或D3)。此外,还提供了一种解码器(20)解码所述编码数据(D2、D3或D4)生成相应的解码输出数据(D5)。



CN 105359418 B

1.一种编码器(10),用于编码输入数据(D1)以生成相应的已编码输出数据(D2或D3),所述输入数据(D1)包括多个数字值的序列,其特征在于,所述编码器(10)包括数据处理装置,用于对输入数据(D1)进行差分和/或总和形式的编码,以生成一个或多个相应的已编码序列,其中,所述一个或多个相应的已编码序列受到一个围绕最大值的回绕和/或一个围绕最小值的回绕处理,生成已编码输出数据(D2或D3),所述编码器(10)用于为一系列预测值使用默认第一个预测值,所述预测值用于生成所述已编码输出数据(D2或D3),其中所述已编码输出数据通过使用输入值、预测值和编码算子而生成。

2.如权利要求1所述的编码器(10),其特征在于,所述数据处理装置用于分析所述输入数据(D1)和/或一个或多个相应的已编码序列来计算一个或多个偏移值、最小值和/或最大值,从而应用到所述一个或多个相应的已编码序列以用于生成所述已编码输出数据(D2或D3)。

3.如权利要求2所述的编码器(10),其特征在于,所述一个或多个偏移值有一个“0”值。

4.如权利要求1所述的编码器(10),其特征在于,所述编码器(10)用于处理包括一个或多个1-bit值的所述多个值,所述编码器(10)用于以一位一位方式编码所述输入数据(D1)。

5.如权利要求1所述的编码器(10),其特征在于,所述一个或多个相应的已编码序列代表在所述输入数据(D1)的连续的值中的变化。

6.如权利要求1所述的编码器(10),其特征在于,所述编码器(10)用于将所述输入数据(D1)细分为多个部分,所述多个部分被单独编码。

7.如权利要求6所述的编码器(10),其特征在于,所述编码器(10)用于仅当数据压缩在所述已编码输出数据(D2或D3)中是可实现的时,对所述数据的部分有选择地进行编码。

8.如权利要求1所述的编码器(10),其特征在于,所述默认第一个预测值为“0”。

9.如权利要求1所述的编码器(10),其特征在于,所述第一个预测值由以下至少一种值的计算得到:最小值、最大值、(最大值+最小值+1)/2。

10.如权利要求1至9任一所述的编码器(10),其特征在于,所述编码器(10)应用附加的编码,以生成所述已编码输出数据,其中所述附加的编码包括以下中的至少一个:行程长度编码(RLE)、SRLE、熵修正(EM)、可变长度编码(VLC)、哈夫曼编码、算术编码、距离编码。

11.如权利要求6所述的编码器(10),其特征在于,所述编码器(10)用于根据其中互相类似的比特的行程长度将所述输入数据(D1)细分为多个部分,所述的类似的比特的行程长度在使用行程长度编码(RLE)、SRLE、熵修正(EM)、哈夫曼编码、可变长度编码(VLC)、距离编码和/或算术编码进行编码时是有效率的。

12.如权利要求1所述的编码器(10),其特征在于,所述处理装置通过使用计算机硬件实现,所述计算机硬件用于执行一个或多个记录于非瞬时机器可读的数据存储介质上的软件产品。

13.一种使用编码器(10)编码输入数据(D1)以生成相应的已编码输出数据(D2或D3)的方法,所述输入数据(D1)包括多个值的序列,其特征在于,所述方法包括:

(a)使用编码器(10)的数据处理装置,对输入数据(D1)进行差分和/或总和形式的编码,以生成一个或多个相应的已编码序列;并且

(b)使用所述数据处理装置,使得所述一个或多个相应的已编码序列受到一个围绕最大值的回绕和/或一个围绕最小值的回绕处理,以生成所述已编码输出数据(D2或D3),

所述方法包括为一系列预测值使用默认第一个预测值，所述预测值为用于生成所述已编码输出数据(D2或D3)的值，其中所述已编码输出数据通过使用输入值、预测值和编码算子而生成。

14.如权利要求13所述的方法，其特征在于，所述方法包括使用所述数据处理装置来分析所述输入数据(D1)和/或所述一个或多个相应的已编码序列来计算一个或多个偏移值、最小值和/或最大值，从而应用到所述一个或多个相应的已编码序列以用于生成所述已编码输出数据(D2或D3)。

15.如权利要求14所述的方法，其特征在于，所述方法包括为一个或多个偏移值使用“0”值。

16.如权利要求13所述的方法，其特征在于，所述方法包括处理所述包括一个或多个1-bit值的多个值，并以一位一位方式编码所述输入数据(D1)。

17.如权利要求13所述的方法，其特征在于，所述一个或多个相应的已编码序列代表所述输入数据(D1)的连续的值的变化。

18.如权利要求13所述的方法，其特征在于，所述方法包括将所述输入数据(D1)细分为多个部分，所述多个部分被单独编码。

19.如权利要求18所述的方法，其特征在于，所述方法包括仅当数据压缩在所述已编码输出数据(D2或D3)中是可实现的时，对所述数据的部分有选择地进行编码。

20.如权利要求13所述的方法，其特征在于，所述默认第一个预测值为“0”。

21.如权利要求13所述的方法，其特征在于，其特征在于，所述方法包括由以下至少一种值的计算得到所述第一个预测值：最小值、最大值、(最大值+最小值+1)/2。

22.如权利要求13至21任一所述的方法，其特征在于，所述方法包括应用附加的编码，以生成所述已编码输出数据，其中所述附加的编码包括以下中的至少一个：行程长度编码(RLE)、SRLE、熵修正(EM)、可变长度编码(VLC)、哈夫曼编码、算术编码、距离编码。

23.如权利要求18所述的方法，其特征在于，所述方法包括根据其中互相类似的比特的行程长度将所述输入数据(D1)细分为多个部分，所述的类似的比特的行程长度在使用行程长度编码(RLE)、SRLE、熵修正(EM)、哈夫曼编码、可变长度编码(VLC)、距离编码和/或算术编码进行编码时是有效率的。

24.如权利要求13所述的方法，其特征在于，所述方法包括通过使用计算机硬件实现所述处理装置，所述计算机硬件用于执行一个或多个记录于非瞬时机器可读的数据存储介质上的软件产品。

25.一种解码器(20)，用于解码已编码数据(D2、D3或D4)以生成相应的已解码输出数据(D5)，其特征在于，所述解码器(20)包括数据处理装置，用于处理已编码数据(D2、D3或D4)的一个或多个部分，其中所述数据处理装置用于将差分和/或总和形式的编码应用于所述一个或多个部分的一个或多个相应的已编码序列，其中所述一个或多个已编码序列受到一个围绕最大值的回绕和/或一个围绕最小值的回绕处理，生成已解码输出数据(D5)，其中所述数据处理装置用于在由此被解码的一系列数据中假定一个第一个预测值的默认值。

26.如权利要求25所述的解码器(20)，其特征在于，所述解码器(20)用于解码包括一个或多个1-bit值的所述已编码数据(D2、D3或D4)，所述解码器(20)用于以一位一位的方式解码所述已编码数据(D2、D3或D4)。

27.如权利要求25所述的解码器(20),其特征在于,所述数据处理装置用于接收一个或多个偏移值、最小值或最大值,以应用到所述一个或多个已编码序列,从而用于生成所述已解码输出数据(D5)。

28.如权利要求27所述的解码器(20),其特征在于,所述一个或多个偏移值有“0”值。

29.如权利要求25所述的解码器(20),其特征在于,所述一个或多个相应的已编码序列代表在已编码成所述已编码数据的多个值中的变化。

30.如权利要求25所述的解码器(20),其特征在于,所述数据处理装置用于将至少以下一种的逆运算应用于被处理的数据:行程长度编码(RLE)、SRLE、熵修正(EM)、可变长度编码(VLC)、哈夫曼编码、算术编码、距离编码。

31.如权利要求25所述的解码器(20),其特征在于,所述默认值有“0”值。

32.如权利要求25所述的解码器(20),其特征在于,所述解码器(20)用于使用所述第一个预测值,所述第一个预测值根据以下中的至少一种值的计算得到:最小值、最大值、(最大值+最小值+1)/2、(最大值/2)+最小值。

33.如权利要求25至32任一所述的解码器(20),其特征在于,所述处理装置通过使用计算机硬件实现,所述计算机硬件用于执行一个或多个记录于非瞬时机器可读的数据存储介质上的软件产品。

34.一种使用解码器(20)解码已编码数据(D2、D3或D4)以生成相应的已解码输出数据(D5)的方法,其特征在于,所述方法包括:

使用数据处理装置来处理所述已编码数据(D2、D3或D4)的一个或多个部分,其中所述数据处理装置用于将差分和/或总和形式的编码应用于所述一个或多个部分的一个或多个相应的已编码序列,其中所述一个或多个已编码序列受到一个围绕最大值的回绕和/或一个围绕最小值的回绕处理,来生成所述已解码输出数据(D5),其中所述数据处理装置用于在由此被解码的一系列数据中假定一个第一个预测值的默认值。

35.如权利要求34所述的方法,其特征在于,所述方法包括解码包括一个或多个1-bit值的所述已编码数据(D2、D3或D4),并使用所述解码器(20)以一位一位的方式解码所述已编码数据(D2、D3或D4)。

36.如权利要求34所述的方法,其特征在于,所述方法包括使用所述数据处理装置来接收一个或多个偏移值、最小值和最大值,从而应用到所述一个或多个已编码序列以用于生成所述已解码输出数据(D5)。

37.如权利要求36所述的方法,其特征在于,所述一个或多个偏移值有“0”值。

38.如权利要求34所述的方法,其特征在于,所述一个或多个相应的已编码序列代表在已编码成所述已编码数据(D2、D3或D4)的多个值中的变化。

39.如权利要求34所述的方法,其特征在于,所述数据处理装置用于将以下至少一种的逆运算应用于被处理数据:行程长度编码(RLE)、SRLE、熵修正(EM)、可变长度编码(VLC)、哈夫曼编码、算术编码、距离编码。

40.如权利要求34所述的方法,其特征在于,所述默认值有“0”值。

41.如权利要求34至40任一所述的方法,其特征在于,所述处理装置通过使用计算机硬件实现,所述计算机硬件用于执行一个或多个记录于非瞬时机器可读的数据存储介质上的软件产品。

42.一种使用解码器(20)解码已编码数据(D2、D3或D4)以生成相应的已解码输出数据(D5)的方法,其特征在于,所述方法包括:

(a)使用数据处理装置处理所述已编码数据(D2、D3或D4)以解码所述已编码数据(D2、D3或D4)的一个或多个部分,考虑到所述已编码数据(D2、D3或D4)包括至少一种已编码序列,所述已编码序列代表在已转换的数据的多个值中变化,并使用一个围绕最大值的回绕和/或一个围绕最小值的回绕;并且

(b)使用所述数据处理装置来生成相应的已处理的数据,并通过使用至少一个预偏移值和/或后偏移值来转换所述一个或多个部分以生成所述已解码输出数据(D5)。

43.一种编解码器,包括至少一个如权利要求1所述的编码器(10)以及至少一个如权利要求25所述的解码器,所述编码器(10)用于编码输入数据(D1)以生成相应的已编码数据,所述解码器用于解码所述已编码数据以生成相应的已解码输出数据(D5)。

编码器、解码器和编码解码方法

技术领域

[0001] 本发明涉及编码器，例如可采用直接ODelta算子的编码器。此外，本发明涉及编码数据的方法，例如采用直接ODelta算子的数据编码方法。此外，本发明还涉及用于对编码数据进行解码的解码器，例如采用逆ODelta算子的解码器。此外，本发明涉及编码数据的解码方法，例如采用逆ODelta算子对编码数据进行解码的方法。更进一步地，本发明涉及记录在非瞬时机器可读的数据存储介质上的软件产品，其中，软件产品可在计算机硬件上执行来实现上述方法。

背景技术

[0002] 克劳德·香农(Claude E.Shannon)提出了一种数学理论，它为现代通信系统提供了基础。此外，在上述数学理论知识上，产生了各种现代编码方法。表1提供了一个概述当代技术知识的信息来源的列表。

[0003] 表1:已知技术

[0004]

文件编号	细节
P1	Variable-length code “长度可变的编码”，维基百科 (2012 年 11 月 28 日访问) URL： http://en.wikipedia.orq/wiki/Variable-lenqth_code
P2	Run-length encoding “行程长度编码”，维基百科 (2012 年 11 月 28 日访问)

[0005]

	URL: http://en.wikipedia.orq/wiki/Run-lenqth_encoding
P3	Huffman coding “哈夫曼编码”，维基百科 (2012 年 11 月 28 日访问) URL：http://en.wikipedia.orq/wiki/Huffman_coding
P4	Arithmetic coding “算术编码”，维基百科 (2012 年 11 月 28 日访问) URL: http://en.wikipedia.orq/wiki/Arithmetic_coding
P5	A Mathematic Theory of Communication “数学理论的交流”，克劳德.香农（1948 年）（2012 年 11 月 28 日访问） URL: http://cm:bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf
P6	Delta encoding “德尔塔编码”，维基百科 (2012 年 11 月 28 日访问) URL: http://en.wikiDedia.orq/wiki/Delta_coding
P7	Shannon's source coding theorem “香农信源编码定理”，维基百科（2012 年 11 月 28 日访问） URL: http://en.wikipedia.orq/wiki/Source_coding_theorem
P8	Entropy “熵”，维基百科（2012 年 11 月 28 日访问） URL: http://en.wikipedia/wiki/Entropv

[0006] 在欧洲专利EP1376974B1(“数据包报头压缩方法和装置”,申请人是Alcatel Lucent(FR)),描述了一种传输数据包的方法,其中数据包由包含压缩值的字段(CF)组成。所述压缩值表示在两个连续数据包之间不断变化的值,并且包括一个预定义的间隔(以下简称“解释区间(interpretation interval)”)。方法包括以下步骤:

[0007] (i)如果要压缩的值与预定义的回绕(wraparound)边界之间的距离低于预设阈值,给所述压缩值追加一个附加位,所述附加位明确地表示该值被压缩到回绕边界的相对位置;

[0008] (ii)在一个接收器中,根据所有接收到的压缩值的比特位计算第一解释区间;

[0009] (iii)如果第一解释区间的不止一个值与接收到的压缩值相匹配,用信号指示回绕(wraparound);

[0010] (iv)在用信号指示回绕的时候,根据压缩值的除所述附加位之外的所有接收到的比特位计算第二解释区间;

[0011] (v)使用附加位,在第二解释区间消除解压缩值的歧义。

[0012] 香农熵(Shanon entropy)的定义是由表1中所列的P7和P8文件提供的。有许多不同的压缩方法,可以用于压缩给定数据的熵,这些方法有时被用于修改熵,例如,为了获得给定数据的更大的无损压缩率之目的;这种修改熵的方法包括,例如,在表1中P2文件描述的行程长度编码(RLE),表1中P1文件描述的可变长度编码(Variable-length code,VLC),表1中P3文件描述的哈夫曼编码(Huffman coding),表1中P6文件中描述的德尔塔编码(Delta encoding),和表1中P4文件描述的算术编码(Arithmetic coding),也称为距离编码(range coding)。这些方法有利于压缩代表字母、数字、字节和字的数据。然而,这些方法并不适用于压缩比特级的给定数据,并且由于这个原因这些方法不能很好的压缩那些容易一位一位地变化的给定数据。

[0013] 德尔塔编码,例如表1中P6文件所述,可以从正数的原始数据值中生成增量值,这些增量值可以是正数或者负数。此外,有一些德尔塔编码的实现是基于所采用的数据元素的尺寸而用于8位、16位或32位回绕。然而,目前缺乏针对8位、16位、32位以外环境而优化的德尔塔编码器。特别是,当编码原始比特值时,也即“0”和“1”,现有德尔塔编码器是尤其低效的,例如逐位(一位一位地,bit-by-bit)编码,它典型地形成三个不同的值,即“-1”、“0”和“1”。

[0014] 所有类型的数据都会占用存储空间,当数据从一个位置移到另一个位置时,需要通信系统的传输容量。随着数据量的增加,例如三维视频内容等多媒体的发展,相应的需要更多的存储空间和传输容量来处理数据,并且随着数据量的增加还需要更多的能源。在全球范围内,被传输的数据量正在随着时间不断增加;例如,因特网包含大量的数据,其中一些存储于多个副本。此外,有些方法可同时用于压缩与数据相关的熵E,例如用在减少数据的大小的时候。此外,也有一些用于修正熵的方法,例如德尔塔编码和行程长度编码,但是,仍然需要改进的方法,来提供比当前方法更大的数据压缩。

[0015] 还有需要对已知的德尔塔编码方法进行优化的使用,来实现对原始数据更快、更有效的编码,例如逐位编码,例如,在原始数据的数据元素中的所有值没有被使用,和/或,与德尔塔编码方法相结合使用的之前或之后的编码方法需要比初始动态用于待编码的数据的比特位更高的比特位格式。

发明内容

[0016] 本发明旨在于提供一种德尔塔编码器的改进形式,即一种“直接ODelta编码器”,当对个别比特进行编码,即一位一位的方式进行编码,以及对其它数据值进行时,这种编码器是更有效的。

[0017] 此外,本发明旨在于提供一种德尔塔编码数据的改进方法,即一种直接ODelta编码数据的方法。

[0018] 此外,本发明旨在于提供一种用于解码已编码数据的改进的解码器,例如ODelta

已编码数据,即一种逆ODelta解码器。

[0019] 进一步地,本发明旨在于提供一种解码已编码数据的改进的方法,例如ODelta已编码数据,即一种逆ODelta解码数据的方法。

[0020] 根据第一个方面,提供了一种如权利要求1所述的编码器:一种编码器(10),用于编码输入数据(D1)以生成相应的已编码输出数据(D2或D3),所述输入数据(D1)包括多个数字值的序列,其特征在于,所述编码器(10)包括数据处理装置,用于对输入数据(D1)进行差分和/或总和形式的编码的,以生成一个或多个相应的已编码序列,其中,所述一个或多个相应的已编码序列受到一个围绕最大值的回绕和/或一个围绕最小值的回绕处理,生成已编码输出数据(D2或D3),所述编码器(10)用于为一系列预测值使用默认第一个预测值,所述预测值用于生成所述输出已编码数据(D2或D3),其中所述已编码数据通过使用输入值、预测值和编码算子而生成。

[0021] 优选地,所述回绕(wrap around)用于避免在实施所述编码器时产生负偏差或太大的值。所述术语"数值"(numerical value)被解释为包括个别比特位(二进制),和比特组(比二进制更高的阶)。

[0022] 本发明的有益效果在于,差分和/或总和编码的结合,和所述回绕提供所述输入数据(D1)的有用的熵修正以生成所述已编码数据(D2),能够在一个给定的熵编码器内从已编码数据(2)生成已编码数据(D3)时增强数据压缩。

[0023] 可选地,所述数据处理装置用于分析所述输入数据(D1)和/或一个或多个相应的已编码序列来计算一个或多个偏移值、最小值和/或最大值,用于所述一个或多个相应的已编码序列,以生成所述已编码输出数据(D2或D3)。进一步可选地,所述一个或多个偏移值有一个"0"值。

[0024] 可选地,所述编码器(10)用于处理包括一个或多个1-bit值的所述多个值,所述编码器(10)用于以一位一位方式编码所述输入数据(D1)。

[0025] 可选地,所述一个或多个相应的已编码序列代表在所述输入数据(D1)的连续的值中的变化。

[0026] 可选地,所述编码器用于使用一个回绕值(wrap Value),其中所述回绕值是最大值(high Value)-最小值(low Value)+1。当生成所述已编码输出数据(D2或D3)时,这样的回绕值被优选地使用,以避免产生比所述最小值(low Value)更小的值、负值、或比最大值(high Value)更大的值。

[0027] 可选地,所述编码器(10)用于将所述输入数据(D1)细分为多个部分,所述多个部分被单独编码。进一步可选地,所述编码器(10)用于仅当数据压缩在所述已编码输出数据(D2或D3)中是可实现的时,对所述数据的部分有选择地进行编码。

[0028] 进一步可选地,所述默认第一个预测值为"0"。进一步可选地,所述第一个预测值由以下至少一种值的计算得到:最小值(low Value)、最大值(high Value)、(最大值(high Value)+最小值(low Value)+1)/2。

[0029] 可选地,所述编码器(10)应用附加的编码,以生成所述已编码输出数据(D2),其中所述附加的编码包括以下中的至少一个:行程长度编码(RLE)、SRLE,熵修正(EM)、可变长度编码(VLC)、哈夫曼编码、算术编码、距离编码。

[0030] 可选地,所述编码器(10)用于根据其中互相类似的比特的行程长度将所述输入数

据(D1)细分为多个部分,所述的类似的比特的行程长度在使用行程长度编码(RLE)、SRLE、熵修正(EM)、哈夫曼编码、可变长度编码(VLC)、距离编码和/或算术编码进行编码时是有效率的。

[0031] 可选地,所述处理装置通过使用计算机硬件实现,所述计算机硬件用于执行一个或多个记录于非瞬时机器可读的数据存储介质上的软件产品。

[0032] 根据第二个方面,提供了一种使用编码器(10)编码输入数据(D1)以生成相应的已编码输出数据(D2或D3)的方法,所述输入数据(D1)包括多个值的序列,其特征在于,所述方法包括:

[0033] (a)使用编码器(10)的数据处理装置,对输入数据(D1)进行差分和/或总和形式的编码,以生成一个或多个相应的已编码序列;并且

[0034] (b)使用所述数据处理装置,使得所述一个或多个相应的已编码序列受到一个围绕最大值的回绕和/或一个围绕最小值的回绕处理,以生成所述已编码输出数据(D2或D3),

[0035] 所述方法包括为一系列预测值使用默认第一个预测值,所述预测值为用于生成所述输出已编码数据(D2或D3)的值,其中所述已编码数据通过使用输入值、预测值和编码算子而生成。

[0036] 优选地,所述回绕用于避免在实施所述编码器时产生负偏差或太大的值。所述术语“数值”(numerical value)被解释为包括个别比特位(二进制),和比特组(比二进制更高的阶)。

[0037] 可选地,所述方法包括使用所述数据处理装置来分析所述输入数据(D1)和/或所述一个或多个相应的已编码序列来计算一个或多个偏移值、最小值和/或最大值,用来作用于所述一个或多个相应的已编码序列以生成所述已编码输出数据(D2或D3)。进一步可选地,所述方法包括为一个或多个偏移值使用“0”值。

[0038] 可选地,所述方法包括处理所述包括一个或多个1-bit值的多个值,并以一位一位方式编码所述输入数据(D1)。进一步可选地,所述方法包括为一个或多个偏移值使用“0”值。

[0039] 可选地,当实施所述方法时,所述一个或多个相应的已编码序列代表所述输入数据(D1)的连续的值的变化。

[0040] 可选地,所述方法包括执行所述编码器以使用一个回绕值(wrap Value),所述回绕值是最大值(high Value)-最小值(low Value)+1。当生成所述已编码输出数据(D2或D3)时,这样的回绕被优选地使用,以避免产生比所述最小值(low Value)更小的值、负值、或比最大值(high Value)更大的值。

[0041] 可选地,所述方法包括将所述输入数据(D1)细分为多个部分,所述多个部分被单独编码。进一步可选地,所述方法包括仅当数据压缩在所述已编码输出数据(D2或D3)中是可实现的时,对所述数据的部分有选择地进行编码。

[0042] 进一步可选地,在所述方法中,所述默认第一个预测值为“0”。进一步可选地,所述方法包括由以下至少一种值的计算得到所述第一个预测值:最小值(low Value)、最大值(high Value)、(最大值(high Value)+最小值(low Value)+1)/2。

[0043] 可选地,所述方法包括应用附加的编码,以生成所述已编码输出数据(D2),其中所述附加的编码包括以下中的至少一个:行程长度编码(RLE)、SRLE、熵修正(EM)、可变长度编

码(VLC)、哈夫曼编码、算术编码、距离编码。

[0044] 进一步可选地，所述方法包括根据其中互相类似的比特的行程长度将所述输入数据(D1)细分为多个部分，所述的类似的比特的行程长度在使用行程长度编码(RLE)、SRLE、熵修正(EM)、哈夫曼编码、可变长度编码(VLC)、距离编码和/或算术编码进行编码时是有效率的。

[0045] 进一步可选地，所述方法包括通过使用计算机硬件实现所述处理装置，所述计算机硬件用于执行一个或多个记录于非瞬时机器可读的数据存储介质上的软件产品。

[0046] 根据第三个方面，提供了一种解码器(20)，用于解码已编码数据(D2、D3或D4)以生成相应的已解码输出数据(D5)，其特征在于，所述解码器(20)包括数据处理装置，用于处理已编码数据(D2、D3或D4)的一个或多个部分，其中所述数据处理装置用于将差分和/或总和形式的编码应用于所述一个或多个部分的一个或多个相应的已编码序列，其中所述一个或多个已编码序列受到一个围绕最大值的回绕和/或一个围绕最小值的回绕处理，生成已解码输出数据(D5)，其中所述数据处理装置用于在由此被解码的一系列数据中假定一个第一个预测值的默认值。

[0047] 可选地，所述解码器(20)用于解码包括一个或多个1-bit值的所述已编码数据(D2、D3或D4)，所述解码器(20)用于以一位一位的方式解码所述已编码数据(D2、D3或D4)。

[0048] 可选地，所述数据处理装置用于接收一个或多个偏移值、最小值或最大值，用于所述一个或多个已编码序列，以生成所述已解码输出数据(D5)。进一步可选地，所述接收的数据为偏移值、最小值和/或最大值。进一步可选地，所述一个或多个偏移值有“0”值。

[0049] 可选地，所述一个或多个相应的已编码序列代表在已编码成所述已编码数据(D2或D3)的多个值中的变化。

[0050] 可选地，所述解码器用于使用一个回绕值(wrap Value)，其中所述回绕值是最大值(high Value)-最小值(low Value)+1。当生成所述已解码输出数据(D5)时，这样的回绕值被优选地使用，以避免产生比所述最小值(low Value)更小的值、负值、或比最大值(high Value)更大的值。

[0051] 可选地，所述数据处理装置用于将至少以下一种的逆运算应用于被处理的数据：行程长度编码(RLE)、SRLE、熵修正(EM)、可变长度编码(VLC)、哈夫曼编码、算术编码、距离编码。

[0052] 进一步可选地，所述默认值有“0”值。然而，所述预测值可以为：最小值(low Value)、最大值(high Value)、(最大值(high Value)+最小值(low Value)+1)/2。

[0053] 可选地，在所述解码器中，所述处理装置通过使用计算机硬件实现，所述计算机硬件用于执行一个或多个记录于非瞬时机器可读的数据存储介质上的软件产品。

[0054] 根据第四个方面，提供了一种使用解码器(20)解码已编码数据(D2、D3或D4)以生成相应的已解码输出数据(D5)的方法，其特征在于，所述方法包括：

[0055] 使用数据处理装置来处理所述已编码数据(D2、D3或D4)的一个或多个部分，其中所述数据处理装置用于将差分和/或总和形式的编码应用于所述一个或多个部分的一个或多个相应的已编码序列，其中所述一个或多个已编码序列受到一个围绕最大值的回绕和/或一个围绕最小值的回绕处理，来生成所述已解码输出数据(D5)，其中所述数据处理装置用于在由此被解码的一系列数据中假定一个第一个预测值的默认值。

[0056] 可选地，所述方法包括解码包括一个或多个1-bit值的所述已编码数据(D2、D3或D4)，并使用所述解码器(20)以一位一位的方式解码所述已编码的数据(D2、D3或D4)。进一步可选地，所述一个或多个偏移值有“0”值。

[0057] 可选地，所述方法包括使用所述数据处理装置来接收一个或多个偏移值、最小值和最大值，用于所述一个或多个已编码序列，以生成所述已解码输出数据(D5)。

[0058] 可选地，在所述方法中，所述一个或多个相应的已编码序列代表在已编码成所述已编码的数据(D2、D3或D4)的多个值中的变化。

[0059] 可选地，所述方法包括利用所述数据处理装置来使用一个围绕最大值的回绕或围绕一个最小值的回绕，以避免在生成所述已解码输出数据(D5)时产生负值或太大的值。

[0060] 可选地，所述数据处理装置用于将以下至少一种的逆运算应用于被处理数据：行程长度编码(RLE)、SRLE、熵修正(EM)、可变长度编码(VLC)、哈夫曼编码、算术编码、距离编码。

[0061] 进一步可选地，所述默认值有“0”值。

[0062] 可选地，在所述方法中，所述处理装置通过使用计算机硬件实现，所述计算机硬件用于执行一个或多个记录于非瞬时机器可读的数据存储介质上的软件产品。

[0063] 根据第五个方面，提供了一种使用解码器(20)解码已编码数据(D2、D3或D4)以生成相应的已解码输出数据(D5)的方法，其特征在于，所述方法包括：

[0064] (a)使用数据处理装置处理所述已编码数据(D2、D3或D4)以解码所述已编码数据(D2、D3或D4)的一个或多个部分，考虑到所述已编码数据(D2、D3或D4)包括至少一种已编码序列，所述已编码序列代表在已转换的数据的多个值中变化，并使用一个围绕最大值的回绕和/或一个围绕最小值的回绕；并且

[0065] (b)使用所述数据处理装置来生成相应的已处理的数据，并通过使用至少一个预偏移值和/或后偏移值来转换所述一个或多个部分以生成所述已解码的输出数据(D5)。

[0066] 参考前述方面，关于偏移值，可选地，在所述解码器中，所述数据处理装置用于接收在所述已编码数据(D2、D3或D4)中的一系列值，得到至少一个预偏移值和/或后偏移值。所述后偏移值可以用于在一个逆算子使用之前生成所述已转换数据，例如一个直接逆Odelta算子，所述预偏移值用于在所述逆算子之后转换数据。可选地，至少一个偏移值与所述处理的数据结合以生成所述已解码输出数据(D5)。

[0067] 所述偏移值在所述解码器中被可选地支持。当它在编码器使用时同样在所述解码器中使用。所述回绕在一个使用范围(low Value到high Value)的参数(wrap Value)内，所述逆算子(总和vs.差分)和逆预测(输入值vs.输出值)为所述解码器的重要因素。

[0068] 可选地，在所述解码器中，所述数据处理装置用于将至少以下一种的逆运算应用于被处理的数据：行程长度编码(RLE)、SRLE、熵修正(EM)、可变长度编码(VLC)、哈夫曼编码、算术编码、距离编码。所述处理被执行以从数据(D3)生成数据(D4)。

[0069] 在所述解码器中，所述数据处理装置用于当用逆解码形式实施时，例如逆ODelta解码，使用一个回绕，所述回绕在一个使用范围(low Value到high Value)的参数(wrap Value)内。

[0070] 根据第六个方面，提供了一种编解码器，包括至少一个如第一个方面所述的编码器(10)以及至少一个如第三个方面所述的解码器，所述编码器(10)用于编码输入数据(D1)

以生成相应的已编码数据(D2或D3)，所述解码器用于解码所述已编码数据(D2、D3或D4)以生成相应的已解码数据(D5)。

[0071] 根据第七个方面，提供了一种记录于非瞬时机器可读的数据存储介质上的软件产品，其特征在于，所述软件产品执行于计算机硬件上以执行所述第二个方面的编码数据方法。

[0072] 根据第八个方面，提供了一种记录于非瞬时机器可读的数据存储介质上的软件产品，其特征在于，所述软件产品执行于计算机硬件上以执行所述第四个方面的编码数据方法。

[0073] 应当理解，本发明的实施例是容易以各种组合被结合，而不脱离本发明的权利要求的保护范围。

附图说明

[0074] 下面结合附图，对本发明的实施例加以描述，其中：

[0075] 图1是实现了本发明的功能的、包括编码器和解码器的编解码器的示意图；

[0076] 图2是图1中编码器执行的数据编码方法的示意图；

[0077] 图3是图1中解码器执行的数据解码方法的示意图。

[0078] 在附图中，使用有下划线的数字表示有下划线的数字位于其上的项目，或者表示与有下划线的数字相邻的项目。不带下划线的数字涉及一个项目，该项目通过将不带下划线的数字连接至所述项目的线标识。当数字不带下划线并且伴有一个相关的箭头时，所述不带下划线的数字是用来识别这个箭头所指向的整个项目。

具体实施方式

[0079] 在描述本申请实施例时，以下首字母缩写词和定义将会被使用，如表2所提供。

[0080] 表2：首字母缩写词和定义

[0081]

首字母缩写词	定义
ADC	Analog-to-digital converter，模数转换器
Codec	编码器和针对数字数据的相应的解码器
DAC	Digital-to-analog converter，数模转换器
DB	Database in Random Access Memory (RAM) or Read Only Memory (ROM)，随机存取存储器或只读存储器的数据库
DC	给定图像中的 DC 成分，即，图像的均值，对应于图像的平均亮度，代表图像的最低空间频率成分
RLE	Run-length encoding，行程长度编码
ROI	Region of interest，兴趣区
ROM	Read Only Memory，只读存储器
VLC	Variable-length code，可变长度编码

[0082]

[0083] 笼统地讲，参照图1，本发明涉及编码器10及其相关的操作方法；有益地，编码器10作为一个直接的ODelta编码器来实现。此外，本发明还涉及相应的解码器20；有益地，解码器20作为逆ODelta解码器来实现。本发明的实施例中采用直接Odelta算子，它是作为上述已知德尔塔编码的位优化版本，它也是为其他数据的距离优化版本。ODelta编码被用于采用可变长度数据字的计算硬件或专用数字硬件，例如8/16/32/64位，和/或采用8/16/32/64位数据元素的可变长度编码，这些数据元素的原始值被表示在1至64比特的范围内，并生成1到64比特的相应的编码值。当然，编码器10和解码器20，在任何情况下，都知道哪种数字值被包含在数据D1中，例如原始数据，因此它的定义或传输将不会进一步在这里阐明。只是被假定数字范围(MIN和MAX)是已知的，并且数据D1可供利用。

[0084] 已知德尔塔编码方法增加值的范围，从初始(MIN到MAX)到结果(MIN-MAX到MAX-MIN)。这意味着，当原始数据仅包含正数时，它也会产生负值。根据本发明的ODelta算子永远不会产生一个不在相应的原始值范围内的值，所以它不会增加已经使用的数据范围，因而适用于熵降低和相关的数据压缩的情况。例如，已知德尔塔编码方法用5位的数据流来工作，也即在从0到31范围的值，这样这种德尔塔编码方法所生成的数据值将在-31到+31范围之间，即使用6位(也即符号位+5位)极大表述的63值；相区别地，当上述5位的数据流生成时，直接ODelta生成的值仍在0到31范围内。而且，鉴于已知德尔塔编码方法不太可能递归，直接，或逆实现，根据本发明的直接或逆ODelta算子则易被递归实现，然而它仍然保留值的已使用范围。值的范围不需要是与位(bit)准确对应。例如0到31的值由5位定义；ODelta算子能够使用值的任何范围，例如从0到25范围的值，同时仍然正常运行。

[0085] 原则上，这里所描述的ODelta方法总是能够直接在现有的数据范围基础上起作用，下文将给出一个示例。ODelta方法还可以被加强，它可以传递信息，所述信息指示在数据中出现的最低数字值(“low Value”)和在数据中出现的最高数字值(“high Value”)。值得注意的是，low Value＞＝MIN和high Value＜＝MAX，并且这些值是可选的。

[0086] 下面将描述根据本发明的直接和逆ODelta算子的两个例子。直接和逆ODelta算子的第一个例子是高效的并且是相对简单实现的，例如在电子硬件和/或计算机硬件上可操作的执行一个或多个记录于非瞬时(非暂时的)机器可读数据存储介质上的软件产品。

[0087] 当根据本发明实施直接或逆ODelta算子时，数据值的所有原始序列都是正数且最低值为0。可选地，一些偏移值，也即预偏移值或后偏移值，可以用来转移(shift)数据值，使得他们均为正值且最低值为“0”。根据本发明的ODelta算子可以以直接的方式被用于所有类型的数据；它通常能够提供数据压缩，即减少需要传输的数据率，因为，当偏移值添加到所有值中或从所有值中减去时，数据值的范围可能被使用更少的比特定义。例如，原始数据值，在直接或逆ODelta算子的应用之前，是在从-11至+18的范围内；这样的范围可以通过使用+11的偏移值转化为0至29的范围，并且所述转化后的范围此后由5-bits描述。当这样的预偏移值或后偏移值没有被使用时，原始数据值至少需要6比特来描述它们，并且通常情况下，在实践中，完整的8比特的有符号的(signed)字节便于使用。

[0088] 当使用广义的直接或逆ODelta算子时，数据范围的类似优化也是可能的。因此，如果直接或逆ODelta算子，或一些其他方法，生成数据值，通过偏移值，这些数据可以呈现为

小于值的全部范围的形式，则所述范围优化可以在ODelta编码方法的任何阶段实施。当所述偏移值被使用时，无论符号是正还是负，它都会被从编码器10传递至解码器20，如后面参照图1、图2和图3所阐述。

[0089] 直接ODelta算子可以以1-bit方式实施，例如为了以逐位方式编码所述原始数据D1；以这样的1-bit方式，如下面被更详细描述的方法1和方法3，当图1中原始数据D1中比特值没有变化时生成“0”，并当原始数据D1中比特值发生变化时生成“1”。可选地，在原始数据中第一比特的预测为“0”，因此在原始数据D1中第一比特的值被保留。可选地，也可能在原始数据中第一比特使用预测值为“1”，但这种选择不提供任何编码的好处；为此，当所述预测总是为1-bit数据被默认定义为“0”，即预定义的“0”被编码器10和解码器20使用时，不需要传递任何选择，从而避免了传达所述预测这一需要，由此改进数据压缩。

[0090] 现在将介绍根据本发明的直接ODelta编码的实例。作为示例，比特的原始序列，即37位，包括17个“1”和20个“0”，在如下的等式1(Eq.1)中提供：

[0091] 0101011001000101000000000001111111111 Eq.1

[0092] 熵E是从等式2(Eq.2)计算得到：

[0093] $$E = 17 * \log_{10}\left(\frac{37}{17}\right) + 20 * \left(\frac{37}{20}\right) = 11.08523 \quad \text{Eq.2}$$

[0094] 一些比特数，即最小比特数(Min_bit)，需要用来编码等式2(Eq.2)中的熵E，它可以根据香农的信源编码定理计算得到，如上述D7和D8文件描述，如等式3(Eq.3)中提供的：

[0095] $$\text{Min\_bits} = \frac{E}{\log_{10}(2)} = 36.82 \text{ bits} \quad \text{Eq.3}$$

[0096] 当如上所述的直接ODelta算子被于比特的原始序列时，即方法1和方法3，生成一个比特序列如下，包括37位，其中有13个“1”和24个“0”。

[0097] 0 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 Eq.4

[0098] 熵E是从等式5(Eq.5)计算得到：

[0099] $$E = 13 * \log_{10}\left(\frac{37}{13}\right) + 20 * \left(\frac{37}{24}\right) = 10.41713 \quad \text{Eq.5}$$

[0100] 按照等式6(Eq.6)，表示为比特的最小数，即Min_bit：

[0101] $$\text{Min\_bits} = \frac{E}{\log_{10}(2)} = 34.60 \text{ bits} \quad \text{Eq.6}$$

[0102] 在等式4(Eq.4)中比特的序列被更进一步地编码来实现数据压缩，例如利用以下至少一种：行程长度编码(RLE)、哈夫曼编码、算术编码、范围编码、熵编码或SRLE编码。

[0103] ODelta算子，当它与所用的熵编码方法一同使用时，减少了一些呈现始数据D1所需要的比特数量，例如RLE或SRLE被用于等式4(Eq.4)中那样的操作过的数据，而不是如等式1(Eq.1)中那样的原始数据；这个1-bit直接ODelta算子，即方法1和方法3，当在等式1(Eq.1)中比特的原始序列中有许多改变时生成“1”，当在等式1(Eq.1)中比特的原始序列中

有一长串互相类似的比特时产生“0”。

[0104] 对于ODelta算子的逆版本，即方法1和方法3的逆，当在编码数据流，即数据D2中有“1”时，将一个比特值从“0”改为“1”，或在适当的时候从“1”改为“0”，且当在编码数据流D2中有“0”时不改变比特值。当ODelta运算被执行于ODelta运算过的(ODelta-operated)数据比特流D2时，原始数据流数据D1被再生成为解码数据流D5。然而，如上所述，额外的编码方式如VLC或哈夫曼编码也可以被使用，这也需要被考虑；这意味着通过使用熵编码器的正向运算由数据D2生成数据D3，使用熵解码器的逆运算由数据D3生成数据D4。

[0105] 原始数据流D1在应用于编码之前被分为两个或更多部分。这种细分为原始数据流D1的编码提供了更多的优化的机会。例如，这种细分的益处在于，在直接ODelta编码，即利用方法1和方法3时，数据D1中的可变序列生成更多的“1”，反之，平坦的不可变序列，即平坦(flat)序列，生成更多的“0”，这是后续V的RL编码或哈夫曼编码所期望的，所以，通过把数据D1分成可以如上所述被独立编码的多个部分，针对构成数据D1的整个比特流，熵E可以被减小。

[0106] 下一步将描述根据本发明的直接ODelta编码的示例，它针对采用了被相互独立编码的多个部分的情形。。包括原始单个比特序列的第一部分总共包括16比特，即7个“1”和9个“0”，如下面的等式7(Eq.7)所示：

[0107] 0101011001000101 Eq.7

[0108] 其中，H(X)＝4.7621，B＝15.82；“H”表示熵，“B”表示Min_bit。当直接ODelta算子被用于等式7(Eq.7)的原始比特序列时，相应的转化比特序列如等式8(Eq.8)所示：

[0109] 0111110101100111 Eq.8

[0110] 其中H(X)＝4.3158，B＝14.34。

[0111] 包括原始单个比特序列的第二部分如下面等式9(Eq.9)所示：

[0112] 00000000001111111111 Eq.9

[0113] 其中H(X)＝6.313，B＝20.97。当直接ODelta算子被用于等式9(Eq.9)的原始比特序列时，相应的转化比特序列如等式10(Eq.10)所示：

[0114] 00000000001000000000 Eq.10

[0115] 其中H(X)＝1.7460，B＝5.80。在这些例子中，如上所述，H(X)代表熵E，B代表编码所需要的比特的最小数。

[0116] 在等式7(Eq.7)和等式10(Eq.10)的这个例子中，把直接ODelta算子分别用于两个部分时，达到最好的压缩(即编码到14.34比特+5.80比特＝总计20.14比特)；这需要的比特比原来所需要的36.82比特更少，直接ODelta运算过的比特需要34.60比特，也少于细分后所需的比特的原始数(＝15.82比特+20.97比特＝36.79比特)。通过逐块地(piece-by-piece)分析原始数据D1的熵E和修改后的-即，包括在数据D2中-数据的相应的熵H，，可以自动地将数据D1中比特的原始流分成多个部分。

[0117] 数据压缩也可以以粗糙的方式实现，假如有一个足够大的数据空间，其中的比特值沿着序列快速地变化，当在数据D1中有多个长行程(long run)部分时，仅将数据D1的部分细分成一个新的部分进行编码。可选地，数据D1的部分数据进行编码，无需使用直接ODelta算子，例如，如果有长的相互类似的比特，其间有相对较少的个别不同的比特，在这种情况下，直接ODelta算子并没有为数据压缩给予显著的好处。

[0118] 将数据D1细分成较小的部分有一个缺点，它产生额外的开销，使得编码数据D2的数据更多。例如，这种开销包括与每个新的部分相关的数据比特或数据字节的数量的信息指示。然而，传输至少一定量的额外开销数据值往往是必要的，从而当给定数据细分为两个数据部分时只有一个额外开销数据值。

[0119] 为了实现以后可以解码的编码的比特流，在直接ODelta算子之后实施熵编码，例如VLC，哈夫曼编码，算术编码，距离编码，RLE，SRLE，EM和其他类似的。执行基于计算熵E和与实际数据编码相比最小的比特估计值的优化计算是更容易和更有效的。这样的顺序的执行可以实现相当快的速度优化，并在编码数据D2中实现最佳的数据压缩。或者，以一种方式执行熵优化是可行的，这种方式为原始比特、字母、数字、字节和字数据，即在数据D1中，首先与其他一些方法编码来生成熵优化的比特流，并此后使用直接ODelta算子修正熵优化的比特流来提供相应的已编码的数据，即数据D2。此外，所述ODelta操作数据仍然可以用其他编码方法从数据D2编码生成数据D3。

[0120] 广义的直接ODelta算子使用一个参数，该参数描述了用于数据D1中的值的范围，即表示这些值所需的一个值或比特数。此外，所述ODelta算子被用于允许使用正负偏移值的方法，换句话说正负“pedestal”值(这里称主基础值)。例如，如果数据D1是七位数据，即支持“0”到“127”的值，但它仅包含从“60”到“115”范围内的值，然后，-60的偏移值被用于数据D1时，由此生成从“0”到“55”范围的转换数据，“0”到“55”也可以表示为仅包含6比特的值，即从而可以实现一定程序的数据压缩。因此，当数据值的全部范围存在数据D1中，即以7比特表示通常以8位字节表示时，广义的直接ODelta算子改善了结果。

[0121] 根据本发明，直接的ODelta值，即方法1，可以使用由如下的示例软件代码的摘录所描述的过程来计算，它是针对于仅具有正值的数据(low Value＝MIN＝0和high Value＝MAX＝127，wrap Value＝127-0+1＝128)：

[0122]

```
wrapValue = power(2, bits) = power(2, 7) = 128
prediction Value = (lowValue + highValue + 1) div 2 = (wrapValue + 1) div 2 +
lowValue = 64
for all pixels
begin
    if(originalValue >= predictionValue) then
        ODelta Value = originalValue - predictionValue
    else
        ODeltaValue = wrapValue + originalValue— predictionValue
    predictionValue = originalValue
end
```

[0123] 现在将提供一个例子来进一步澄清上述ODelta算子。

[0124] 值的原始序列如下面的等式11(Eq.11)所示：

[0125] 65,80,126,1,62,45,89,54,66 Eq.11

[0126] 相应的德尔塔编码值如下面的等式12(Eq.12)所示：

[0127] 65,15,46,-125,61,-17,44,-35,12 Eq.12

[0128] 相应的直接ODelta编码值如下面的等式13(Eq.13)：

[0129] 1,15,46,3,61,111,44,93,12 Eq.13

[0130] 其中使用了在参数wrapValue之内的回绕。

[0131] 逆ODelta算子,即方法1,可用于生成逆ODelta值,例如利用如下的软件代码示例实现：

[0132]
```
wrapValue = power(2, bits) = power(2, 7) = 128
predictionValue = (wrapValue + 1) div 2 + low Value = 64
for all pixels
begin
    ODeltaValue = originalValue + predictionValue
    if (ODeltaValue >= wrapValue) then
        ODeltaValue = ODeltaValue - wrapValue
    predictionValue = ODeltaValue
end
```

[0133]

[0134] 当这个软件代码被执行并被应用于等式13(Eq.13)时,生成如等式14(Eq.14)提供的值：

[0135] 65,80,126,1,62,45,89,54,66 Eq.14

[0136] 这个例子使用wrapValue作为2的幂值。这不是强制性,wrapValue也可以是比最大的数据值更大的任何值或比已使用的范围更大的值(如果负值也可用),或者范围可以在数据给定序列中根据前偏移量修改。稍后会有说明此功能的另一个例子。

[0137] 参照图1总结上面所述,本发明涉及编码器10和解码器20。可选地,编码器10和解码器20相结合作为由编号30指示的编解码器。编码器10是可以接收原始输入数据D1,D1被使用例如直接的ODelta方法来编码以生成相应的编码数据D2或D3。编码数据D2或D3通过通信网络40传送或存储于数据存储介质50上,例如数据的载体,如光盘只读存储器(ROM)或类似的。解码器20可以接收编码数据D2或D3,例如通过通信网络40或数据存储介质50上提供,并使用逆方法,例如逆ODelta方法,来生成相应的解码数据D5,它基本上类似于原始数据D1。编码器10和解码器20使用数字硬件实现,例如可以执行一个或多个软件产品的计算机硬件,例如在此描述的作为示例实施例所提供的代码。可选地,编码器10和/或解码器20使用专用数字硬件实现。

[0138] 如在编码器10中执行的ODelta方法,采用如图2所示的步骤。可选地,第一步100,处理输入数据D1来找到它的数据元素的值的范围。在可选的第二步110中,从所述值的范围,计算偏移量,即前偏移量,用于把数据元素转换为一个有利的形态从而生成对应的一组转换元素(translated elements)。在第三步120中,所述在第二步110中转换的元素,受到

直接ODelta编码来生成相应的ODelta编码值。在第四步130中，ODelta编码值和可选的偏移值，最小值(low Value)，和/或最大值(high Value)分别被编码，例如行程长度编码(RLE)、距离编码，或哈夫曼编码，来从数据D2生成数据D3。偏移值、最小值(low Value)，和/或最大值(high Value)并不总是可压缩的，因此需要使用一定量的比特从编码器10传递到解码器20。此外，偏移值、最小值(low Value)，和/或最大值(high Value)是直接ODelta算子的可选功能；例如，偏移值，在某些情况下有“0”值，low Value有MIN值，high Value有MAX值，即没有进行转换，其全部范围被用。特别是，当直接ODelta算子为1比特数据数实现时，即为一位一位编码，它根本不需要偏移值，此时步骤100和110通常被忽略。当偏移值也用于步骤110时，其中的代表最高和最低值的范围值应更新。解码器20应已知不同值的数量，即wrap Value，或者编码器10应当在压缩数据内把它传递至解码器20。可选地，默认wrap Value(＝high Value-low Value+1)用于编码器和解码器。可选地，至少编码器10和解码器20以递归方式操作，例如为了找到一种最佳的方式，将输入数据D1再分成用于编码的部分以提供数据D1的优化压缩来生成编码数据D2。

[0139] 在解码器20中执行的逆ODelta方法采用如图3所示的步骤。在第一步200中，数据D2/D3或D4被使用上述步骤130进行逆编码来生成解码ODelta数据，其中已解码的ODelta数据有已ODelta编码的值并可选地有单独的偏移值。在第二步210中，ODelta编码值被解码来产生数据元素序列。在第三步220中，数据元素的序列使用可选的预偏移值来转换生成解码数据D5；在某些情况下，这种转换被设置为“0”，即没有转换被有效地使用。再者，无需使用偏移值来执行所述方法也是可以的，例如当执行1比特编码时，即一位一位编码，步骤220被省略。更进一步地，解码器20也应当知道wrapValue，从而能够以适当的方式解数被接收的数据元素。

[0140] 通过采用偏移量来实现只有正值，在数据D2或D3中更有效的数据压缩是能够实现的。如果所有数据值已经是正值，就无需添加任何偏移值。当然，可选地，负偏移值可用来减少可用的范围，如下一个示例所展示的，但这不是强制性的。

[0141] 可选地，图2和图3中的方法可以仅用经过ODelta编码的可用的值来进一步优化。这种优化要求所使用的值是已知的。例如，以上述情况为例，只有从1(＝原始最低值)到126(＝原始最高值)的值存在于原始数据集D1中。偏移值是1(-low Value＞＝原始最低值-偏移量＝1-1＝0且high Value＝原始最高值-偏移量＝126-1＝125)。当预偏移值已经从原始数据D1中被减少时，从而产生了在等式15(Eq.15)中的以下值：

[0142] 64,79,125,0,61,44,88,53,65 Eq.15

[0143] 从等式15(Eq.15)，确定125的最大值(high Value＝原始max-偏移量＝126-1＝125)，这样，“number”(＝最大增量值＝high Value-low Value)可以是125，或者wrap Value可以最小为126(＝number+1＝high Value-low Value+1)。现在，就必须要存储和/或传递这些值，然后可以通过更改如下过程值，来修改前面的示例：

[0144] wrap Value＝126(″0″到″125″＝＞126不同的值)

[0145] prediction Value＝(high Value+low Value+1)div 2＝(wrap Value+1)div 2+low Value＝63

[0146] 等式16(Eq.16)中提供了相应的直接ODelta算子值：

[0147] 1,15,46,1,61,109,44,91,12 Eq.16

[0148] 现在所有的“负增量值(Delta values)”削减了2的量级(a factor of 2)(即＝范围变化＝128-126)。同样地,在解码器20中,过程值必须改为如下:

[0149] wrap Value＝126

[0150] prediction Value＝(wrap Value+1)div 2+low Value＝63

[0151] 相应的逆ODelta值如等式17(Eq.17):

[0152] 64,79,125,0,61,44,88,53,65 Eq.17

[0153] 当预偏移值添加到等式17(Eq.17)时,等式18(Eq.18)中的如下结果对应于在等式15(Eq.15)中的原始数据,即:

[0154] 65,80,126,1,62,45,89,54,66 Eq.18

[0155] 在此示例中,值的范围几乎已满,所以有一个中等的好处,它来自于使用跟随偏移量和最大值(high Value)的直接ODelta算子。然而,熵E的降低仍然可以实现,即当他们被适当的传递时,导致在频率表中或在代码表中更少的值。当更少地地使用该范围时,可以达到最大的好处。

[0156] 编码和解码数据的实用的1-bit直接和逆ODelta方法的示例实施例,即方法1或方法3,现在将通过可执行的计算机软件代码提供。该方法采用上述直接和逆ODelta算子,即方法1或方法3。当执行于计算机硬件上时,软件代码可以从一个字节缓冲区到另一个字节缓冲区处理比特。在软件代码中,GetBit、SetBit和ClearBit函数总是更新HeaderBits值。当下一位会在下一个字节时,HeaderIndex值也被更新。可选地,可以优化软件代码,以便只有一个HeaderIndex和HeaderBits值的集合被用于源和目标,这样这些值仅在给定比特被写入目标缓冲区时才会被更新。

[0157]

```
procedure EncodeODelta1u(APtrSrc : PByte; ASrcDstBitLen : PCardinal;
APtrDst : PByte)
var
iSrcHeaderIndex, iSrcHeaderBits, iIndex,
iDstHeaderIndex, iDstHeaderBits : Cardinal;
bBit, bLastBit : Boolean;
begin
// Reset offsets
iSrcHeaderIndex := 0;
iSrcHeaderBits := 0;
iDstHeaderIndex := 0;
iDstHeaderBits := 0;

// Initialise delta value
```

[0158]

```
    bLastBit := False;

    // Go through all bits
    for ilndex := 0 to ASrcDstBitLen do
    begin
      // Read bit
      bBit := GetBit(APtrSrc, ©iSrcHeaderlndex, ©iSrcHeaderBits);

      // Set destination bit if current source bit is different than previous source bit
      if (bBit <> bLastBit) then
      begin
        SetBit(APtrDst, ©iDstHeaderlndex, ©iDstHeaderBits);
        bLastBit := bBit;
      end
      else ClearBit(APtrDst, ©iDstHeaderlndex, ©iDstHeaderBits);
    end;
    end;
    function DecodeO Delta 1u(APtrSrc ; PByte; ASrcDstBitLen : PCardinal;
APtrDst : PByte) : Boolean;
    var
    iSrcHeaderlndex, iSrcHeaderBits, ilndex,
    iDstHeaderlndex, iDstHeaderBits : Cardinal;
    bBit, bLastBit : Boolean;
    begin
    // Reset offsets
    iSrcHeaderlndex := 0;
    iSrcHeaderBits := 0;
    iDstHeaderlndex := 0;
    iDstHeaderBits := 0;
```

```
// Initialise delta value
bLastBit := False;

// Go through all bits
for ilndex := 0 to ASrcDstBitLen do
begin
// Read bit
bBit := GetBit(APtrSrc, ©iSrcHeaderlndex, ©iSrcHeaderBits);

// Change bit value if source bit is true
if (bBit = True) then
```

[0159]

```
begin
if (bLastBit = True) then
bLastBit := False
else bLastBit := True;
end;

// Set destination bit based on bit value (True or False)
if (bLastBit) then
SetBit(APtrDst, ©iDstHeaderlndex, ©iDstHeaderBits)
else ClearBit(APtrDst, ©iDstHeaderlndex, ©iDstHeaderBits);
end;
end;
```

[0160] 上述直接和逆ODelta算子，即方法1或方法3，用于压缩在数字格式中的数据的任何类型，例如视频数据、图像数据、音频数据、图形数据、地震数据、医学数据、测量值、参考数和遮罩(mask)。此外，当首先转换为相应的数字数据时，例如在压缩之前使用模数转换器(ADC)，一个或多个模拟信号也是使用直接ODelta算子压缩的。当使用逆ODelta算子时，如果需要将数据转换回一个或更多的模拟信号，可以在操作后使用数模转换器(DAC)。然而，直接ODelta算子经常在压缩数据方面不是很有效，但当结合其他编码方法使用时能够提供有效的数据压缩，例如可变长度编码(VLC)、算术编码、距离编码、行程长度编码、SRLE、熵修正等。在编码器10中采用直接ODelta算子之后，对数据D2使用这些编码方法。编码数据D2必

须在由此产生的数据被传递至解码器20中的逆ODelta算子之前进行相应地解码。ODelta算子也可以与熵修正器的其他类型一起使用。在某些情况下，直接ODelta算子可以导致熵E的增加，且只有当它提供了有益地数据压缩性能时，数据压缩算法可以有选择地使用直接ODelta算子进行数据编码，例如根据被压缩数据的类型有选择的使用，例如有选择性地应用于上述输入数据D1的可选部分。

[0161] 直接ODelta算子已经被提出结合块编码使用，例如，包含在申请号为US13/584,005的美国专利所描述，并且逆ODelta算子已经被提出结合块解码使用，如包含在申请号为US13/584,047的美国专利所描述。可选地，直接ODelta算子和逆ODelta算子与如包含在申请号为US13/657,382的美国专利所描述的多级编码方法结合使用。包括二进制状态的1-bit数据的所有类型，例如在数据D1中呈现的，都受直接ODelta算子的1-bit版本的作用，生成相应的转换后的数据，这些转换后的数据之后受到实际的熵编码作用，生成编码数据D2或D3。可选地，如上面提到的，有选择的使用直接ODelta算子取决于原始数据D1的类型。

[0162] 可选地，在直接ODelta算子之前或之后使用修正数据熵的其他方法是可行的。例如，直接ODelta算子也可以直接用于直接ODelta算子的广义版本内的多比特数据。此外，在所有已使用的比特都先放入比特串序列后，上述直接ODelta算子的1-bit版本被用于多比特数据。

[0163] 当多个方法与编码器10中的直接ODelta算子一起被用于数据压缩时，相应的逆运算在解码器20中以相反的顺序进行，例如：

[0164] 在编码器10中使用下面的方法序列：

[0165] [data D1]＝＞direct ODelta(method 2)

[0166] ＝＞VLC-

[0167] ＝＞EM

[0168] ＝＞Arithmetic coding

[0169] ＝＞[dara D3] Eq.19

[0170] 在解码器20中使用下面的方法逆序列：

[0171] [data D3]＝＞inverse Arithmetic coding

[0172] ＝＞inverse EM

[0173] ＝＞inverse VLC

[0174] ＝＞inverse ODelta(method2)

[0175] ＝＞[data D5] Eq.20

[0176] 其中，“VLC”表示可变长度编码，而“EM”表示熵修正。

[0177] 如上所述的ODelta算子是可逆的和无损的。此外，ODelta算子可以专门为1-bit数据流实施，例如当执行一位一位编码时，但它也为其他数据实施。数据的所有类型都容易使用直接ODelta算子的广义版本处理。当数据要被压缩时使用直接ODelta算子，并当压缩的数据要被解压缩时使用相应的逆ODelta算子。可选地，当ODelta算子被使用时，直接ODelta算子与和它相应的逆运算以相反的顺序被使用；换句话说，逆ODelta算子首先临时地在原始比特流上执行，其后紧随直接ODelta算子，以再生成原始比特流。一个ODelta算子增加了熵且其他ODelta算子减小了熵。非常罕见的情况是，直接ODelta算子不应修正熵，逆ODelta算子也不应修正熵。值得注意的是，当直接和逆ODelta算子被使用时，例如方法1，这些操作

的逆顺序与方法4的正常顺序相似。顺序的类似变化也可能用于方法2和方法3。

[0178] 在1-bit版本中，即以一位一位的方式编码数据，直接ODelta算子在没有预测的情况下开始，即，它默认假定初始“0”值的预测。在广义的版本中，ODelta算子在有表示可用数据范围一半的预测的情况下开始；例如，如果5-bits用于数据D1中的输入数据值，即从“0”至“31”范围的32个不同的值，预测值是32/2＝16。ODelta算子需要为使用算子处理的数据元素提供有关可用数据范围的信息。

[0179] 上述本发明的实施例使在数据D1中以比特或任何数字值表示的熵E的降低成为可能。直接ODelta算子相比于德尔塔编码来说几乎总是提供了改善的熵降低。只有德尔塔编码与字节回绕(wraparound)结合使用的情况下，带有原始预测(方法1)的不同的ODelta操作使用wrap Value＝256，low Value＝MIN＝0，和high Value＝MAX＝255，在它里面产生相同的输出结果。如果使用另一直接ODelta方法，或者如果在输入数据中不是整个数据范围都被用了，那么ODelta算子通过发送所选的方法或low Value和/或high Value产生更好的结果，即也自动修正wrap Value。更小的熵能够使数据以更高的数据压缩率压缩。更高的数据压缩率能够使更小的数据存储空间被使用，且也能够使得压缩数据被传送时使用更慢的数据带宽，相应地也减少能源消耗。

[0180] 在上述中，在编码器10中执行一种形式的差分、总和的计算，且在解码器20中进行相应的逆计算。可以在编码器10中使用另一种预测方法，然后在解码器20中进行相应的逆预测。这意味着实际上是有至少四种不同的直接ODelta方法，以及至少四种相应的逆ODelta方法。这些方法的详细和精确的描述如下文。可选地，以递归方式进行计算以在编码数据D2(或D3)中获得更高程度的数据压缩。当执行这种递归计算时，使用不断变化的数字范围作为已经使用了多少递推计算的函数。例如，在编码器10，对数据D1中执行以下顺序的计算以生成编码数据D2(或D3)：

[0181] [Data D1]edirect ODelta(method 3)＝＞

[0182] edirect ODelta(method 3)＝＞

[0183] eEM＝＞

[0184] edirect ODelta(method 1)＝＞

[0185] eVLC [Data D3] Eq.21

[0186] 且在解码器20中进行相应的逆操作：

[0187] [Data D3]dVLC＝＞

[0188] din verse ODelta(method 1)＝＞

[0189] dEM＝＞

[0190] dinverse ODelta(method 3)

[0191] dinverse ODelta(method 3) [Data D5] Eq.22

[0192] 在这四种方法中每次数据被操作，如等式21(对应方法1的Eq.21)、等式22(对应方法2的Eq.22)、等式23(对应方法3的Eq.23)、等式24(对应方法4的Eq.24)所描述的，尝试使用所有方法是可能的，因为这些方法的一种可能降低正在被处理的数据的熵超过了其他方法。在编码器10和/或解码器20内优化使用方法，多次使用相同或不同的方法是有利的，只要选定的一个方法，或多个方法，与在所需数据中信息的数量相比降低了熵。因此方法1到方法4可用于编码多个数值(numerical values)，其中“numerical values”(数据)在其定

义中包括1-bit数据以及非二进制数字，就像以一位一位方式的编码流，以及多比特的值。

[0193] 差分运算表示一个连续的数字值的差数；相应地，总和操作表示连续的数字值的总和。在编码器10中执行的这些操作有它们自己相应的在解码器20中执行的逆运算。差分或总和可以基于当前的输入值和预输入值或用作预测值的结果值来计算。也可以用其他的预测值，例如，它们可能在编码器中使用较早的输入和输出值来生成预测，只要它在解码器中是可逆的。

[0194] 这种方法在编码器10和解码器20内没有显著地压缩数据，但所有方法都被用来降低熵，这样一些其他压缩方法可以更有效地压缩的已降低熵的数据。可选地，这种其他压缩方法至少为以下的一个：哈夫曼编码、算术编码、距离编码、RLE编码、SRLE编码、熵修正编码。然而，对于所有的方法，需要传送一些数据值，使用这些数据值，可以准确地执行操作和其逆操作，例如如果要实现数据的无损压缩和随后的无损解压缩。当然，编码器10和解码器20有关于包括在输入数据D1中的数字值的类型的信息。假定数字范围是已知的，即由MIN和MAX定义。原则上，这些方法总是能够直接根据现有的数据范围来工作。操作需要的数值是出现的最低数字值(low Value)和出现的最高数字值(high Value)；low Value大于或等于MIN，high Value小于或等于MAX。

[0195] 在这些值的基础上，可以导出其他必要的数字值。以多种形式传送这些值，其中缺少的值被计算。例如，如果集合[“low Value”、“high Value”、“number”]中的两个值是已知的，所述“number”是[high Value-low Value]，那么可以由此计算第三个值。省略数据D2中的某些值，然后在解码器20中派生它们，这能够在数据D2中提供更大的数据压缩。

[0196] 除了这些值，需要在第一个值的计算中用作预值的字母P，即“prediction”(预测)。“0”和“number”之间的值通常能够被选作字母P，即“prediction”。此外，上述操作需要提供“wrapValue”值，为了当在解码器20中解码数据D2/D3或D4时进行恢复性的工作，即尽可能地缩小运算所产生的值的范围。然而，这个“wrap Value”必须大于“number”，且它将有“number”+1这样的值。可选地，取决于数据D1的类型，第一个“预测”值可以选择“0”，如上文所述，例如，如果假定数据D1包含的小值多于它所包含的大值；可选地，第一个“预测”值可以选择为等于所述“number”，如果假定数据D1包含的大值多于它所包含的小值。在一个不是针对值的大小而作出假定的情况下，就需要为了“prediction”值而使用一个“(wrap Value+1)div 2+low Value”值。

[0197] 现在将描述实现本发明的实施例的计算机硬件的操作示例。

[0198] 在编码器10中，第一次直接差分操作在软件循环中被计算，即方法1，实施步骤如下；对于所有数据值，输出值，即“result”，对应于输入值，即“original”：

[0199] result＝original-prediction

[0200] if result＜low Value then result＝result+wrap Value

[0201] 最终，下一次输入的预测值设置为与当前输入相等，即：

[0202] prediction＝original

[0203] 在解码器20中，第一次逆差分操作在软件循环中被计算，即方法1，如下实施：对于所有数据值，对应于输入值即“original”的输出值，即“result”：

[0204] result＝original+prediction

[0205] if result＞high Value then result＝result-wrap Value

[0206] 最终,下一个输入的预测值设置为与当前输出相等,即:

[0207] prediction=result

[0208] 在编码器10中,第二次直接差分操作在软件循环中被计算,即方法2,如下实施;对于所有数据值,对应于输入值即“original”的输出值,即“result”:

[0209] result=original-prediction

[0210] if result<low Value then result=result+wrap Value

[0211] 最终,下一个输入的预测值设置为与当前输出相等,即:

[0212] prediction=result

[0213] 在解码器20中,第二次逆差分操作在软件循环中被计算,即方法2,如下实施;对于所有数据值,对应于输入值即“original”的输出值,即“result”:

[0214] result=original+prediction

[0215] if result>highValue then result=result-wrapValue

[0216] 最终,下一个输入的预测值设置为与当前输入相等,即:

[0217] prediction=original

[0218] 在编码器10中,第一次直接总和操作,即方法3,如下实施:对于所有数据值,对应于输入值即“original”的输入值,即“result”:

[0219] result=original+prediction

[0220] if result>high Value then result=result-wrap Value

[0221] 最终,下一次输入的预测值设置为与当前输入相等,即:

[0222] prediction=original

[0223] 在解码器20中,第一次逆总和操作,即方法3,如下实施:对于所有数据值,对应于输入值即“original”的输入值,即“result”:

[0224] result=original-prediction

[0225] if result<low Value then result=result+wrap Value

[0226] 最终,下一次输入的预测值设置为与当前输出相等,即:

[0227] prediction=result

[0228] 在编码器10中,第二次直接总和操作,即方法4,如下实施:对于所有数据值,对应于输入值即“original”的输入值,即“result”:

[0229] result=original+prediction

[0230] if result>high alue then result=result-wrap Value

[0231] 最终,下一次输入的预测值设置为与当前输出相等,即:

[0232] prediction=result

[0233] 在解码器20中,第二次逆总和操作,即方法4,如下实施:对于所有数据值,对应于输入值即“original”的输入值,即“result”:

[0234] result=original-prediction

[0235] if result<low Value then result=result+wrap Value

[0236] 最终,下一次输入的预测值设置为与当前输出相等,即:

[0237] prediction=result

[0238] 这样的总和与差分操作,所有这四种方法,也适用于1-bit数据,即一位一位的,即

当实现编码器10和解码器20的ODelta版本时。1-bit数据的情况，下一个值已经是编码器10和解码器20已知的，即MIN＝0，MAX＝1。此外，假定lowValue＝MIN＝0，且highValue＝MAX＝1。更进一步地，在这样的例子中，“number”是[high Value-low Value＝1-0＝1]，且wrap Value被选为“number”+1＝1+1＝2。预测值被选为“0”，因为只有那些从low Value＝MIN＝0开始仅有正值的1-bit数据才会被考虑。对于1-bit数据，方法1和方法3产生互相类似的编码结果。相似的，方法2和方法4产生互相类似的编码结果。了解了这些内容，简化那些在数据D2中需要被发送的信息，如各种默认值可以被假定，即只有发送那些与差分操作的执行次数的数字有关的信息是必要的，无论是方法1还是方法2，还是所选的预测(input value(方法1)或result value(方法2))，以至于当解码数据D2、D3或D4来产生解码数据D5时，解码器20可以执行必要次数的正确的逆差分操作。

[0239] 也可以通过使用方法2或方法4处理通过方法1或方法2实现的第一个例子，产生类似的输出，当应用于数据Eq1时，以下所示的结果可以通过这些方法实现：

[0240] 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1

[0241] 这时，处理过的数据有24个“1”和13个“0”，即熵与第一个例子一样，但是“1”和“0”变化的地方不一样。这并不是总发生，相反的，在不同的方法之间也经常熵变化。例如，在数据的前四个元素之后，方法1和/或方法3产生3个“1”和1个“0”，而在原始数据和已经用方法2和/或方法4处理过的数据中有2个“1”和2个“0”。因此，在这种情况下方法1和/或方法3产生比方法2和/或方法4更小的熵，且比原始的熵更小。

[0242] 在多比特(multi-bit)实施中，如果数据D1包括范围从-64到+63的值，那么MIN＝-64，且MAX＝63。假设：low Value＝MIN且high Value＝MAX，“number”＝127且wrapvalue被选为128。然而，当数据D1随机变化时，“prediction”设置为[(wrapvalue+1)div 2+low Value＝64+-64＝0]。

[0243] 值得注意的是，如果第一个值是-1，第一个用直接ODelta方法1和/或方法2编码的值，应为-1-0＝-1，且相应的，用直接ODelta方法3和/或方法4编码的值为-1+0＝-1。那么下面的值将根据数据如何进行的而改变，例如，如果第二个值为5，那么直接ODelta方法1将产生5--1＝6，直接ODelta方法2将产生5--1＝6，直接ODelta方法3将产生5+-1＝5，直接ODelta方法4将产生5+-1＝4。在这个例子中，当使用逆ODelta方法1和/或方法2时，解码器20能够产生如第一个值，-1+0＝-1，用逆ODelta方法3和/或方法4则为-1-0＝-1。相应地，第二个值用逆ODelta方法1将为6+-1＝6，用逆ODelta方法2则为6+-1＝5，用逆ODelta方法3则为4--1＝5，用逆ODelta方法4则为4--1＝5。

[0244] 如果数字范围实际上只包括从-20到+27的值时，这个方法可以优化。在本示例中，传输是可行的，例如，low Value＝-20且high Value＝27。如果两个都被传输，计算number＝47且wrap Value被选为48是可行的。现在，为了预测计算值48 div 2+-20＝4。然后，当使用ODelta方法1或方法2时，前面的示例将产生例如-1:-1-4＝-5，当使用ODelta方法3或方法4时，-1+4＝3。类似地，根据ODelta方法第二个值将为(5--1)＝6，(5--5)＝10，(5+-1)＝4和(5+3)＝8。解码器20再次正确起作用并根据方法1和/或方法2产生第一个值，-5+4＝-1，根据方法3和/或方法4产生第一个值为3-4＝-1。相应地，根据不同方法产生的第二个值将被解码为(6+-1)＝5，(10+-5)＝5，(4--1)＝5和(8-3)＝5。

[0245] 值得注意的是，在上面这些例子中所有值都在范围内，即从-64到+63或从-20到+

27,并且没有必要在这些示例值内执行修正项,但如果任何负的或正的变化足够大,那么就不得不通过给定的等式21到24(Eq.21到Eq.24)来修正数据值以保证结果值在范围内。要指出的是,这里的修正项是指回绕值。

[0246] 当low Value是已知的,为了简化必须随着熵编码数据D3从编码器10发送至解码器20的编码表,已编码的值被安排从0开始、结束于“number”。这个操作就是所谓的后偏移,在熵解码之后和数据D4逆ODelta操作之前,这个后偏移值必须从所述已编码的数值中删除。

[0247] 正如前面提到的,用预偏移功能实现偏移量也是可能的,在原始输入数据(D1)转化为正元素的地方,所述正元素可以在ODelta方法的实际执行之前已经包含从0到“number”的值。也在这种情况下,这个操作所需要的信息传输以这样的一种方式有利的进行,“预偏移”和ODelta方法不重复传输相同的信息,或忽略那些由于其他方法已知的信息。这个预偏移影响应当在逆ODelta操作之后从已解码的数据中删除来产生适当的D5输出数据。

[0248] 不背离本发明如所附权利要求所定义的范围,对本发明上面所述的实例的修改是可能的。例如“包括”,“包含”、“由...组成”,“有”,“是”这些用法在本发明的描述中应当被理解为是非排它的含义,即允许为那些未显性描述的项目、组件或元素的存在。有关单数的描述应当被理解为可以包含复数的含义。包括在权利要求中括号内的数字标记旨在协助权利要求的理解,不应以任何方式限制这些权利要求的保护范围。

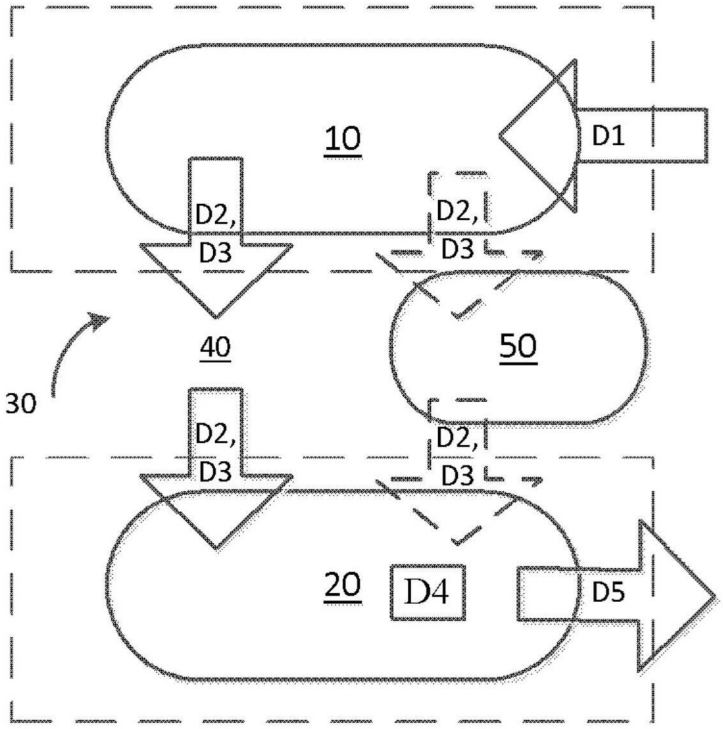


图1

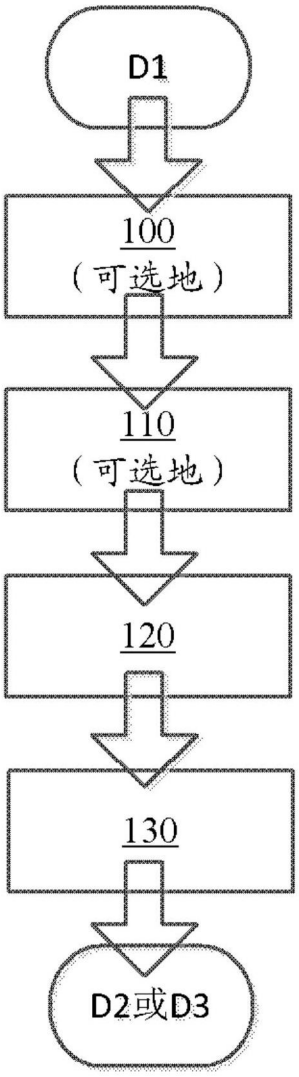


图2

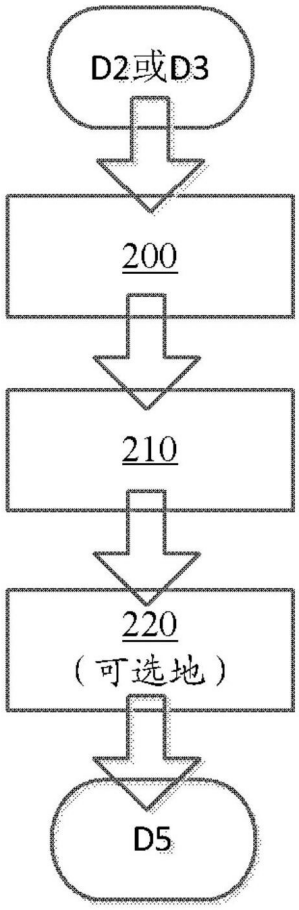


图3