(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 107948645 B
(45)授权公告日 2021.12.24

(21)申请号 201711438973.3

(22)申请日 2012.07.16

(65)同一申请的已公布的文献号
申请公布号 CN 107948645 A

(43)申请公布日 2018.04.20

(30)优先权数据
61/508,477 2011.07.15 US

(62)分案原申请数据
201280044181.7 2012.07.16

(73)专利权人 GE视频压缩有限责任公司
地址 美国纽约

(72)发明人 瓦莱里·乔治
阿纳斯塔西娅·亨克尔 (续)

(74)专利代理机构 北京康信知识产权代理有限责任公司 11240
代理人 梁丽超 刘丹

(51)Int.Cl.
H04N 19/13(2014.01) (续)

(56)对比文件

Gordon Clare等.Wavefront Parallel Processing for HEVC Encoding and Decoding.《Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 6th Meeting: Torino, IT, 14-22 July, 2011, JCTVC-F274》.2011,

Gordon Clare等.Wavefront and Cabac Flush: Different Degrees of Parallelism Without Transcoding.《Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 6th Meeting: Torino, IT, 14-22 July, 2011, JCTVC-F275》.2011,

Félix Henry.Wavefront Parallel Processing.《Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 5th Meeting: Geneva, CH, 16-23 March, 2011, JCTVC-E196》.2011, (续)

审查员 赵慧敏

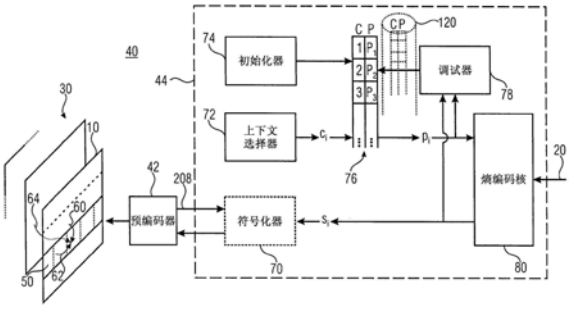
权利要求书4页 说明书25页 附图18页

(54)发明名称

解码器和方法、编码器和方法及存储介质

(57)摘要

本发明提供了解码器和方法、编码器和方法及存储介质，解码器被配置为：对熵编码器数据流内的多个熵片段进行熵解码以重建分别与熵片段相关联的样本阵列的不同部分，其中，针对每个熵片段沿相应的熵编码路径使用相应的概率估计来执行熵解码，沿相应的熵编码路径使用相应的熵片段的先前解码部分来调试相应的概率估计，使用熵片段顺序开始对多个熵片段依次进行熵解码，并且，在对预定熵片段进行熵解码的过程中，基于如使用预定熵片段的先前已解码部分调试的预定熵片段的相应概率估计，以及如在空间相邻熵片段的相邻部分的按熵片段顺序空间相邻的先前熵片段的熵解码中使用的概率估计，来对预定熵片段的当前部分执行熵解码。



CN 107948645 B

[转续页]

[接上页]

(72)发明人 海纳·基希霍弗尔
德特勒夫·马佩 托马斯·席尔

(51)Int.Cl.
H04N 19/43(2014.01)
H03M 7/30(2006.01)
H03M 7/40(2006.01)

(56)对比文件
Chih-Wei Hsu等.Wavefront Parallel Processing with Tiles.《Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 6th Meeting: Torino, IT, 14-22 July, 2011, JCTVC-F063》.2011,

1.一种用于从熵编码数据流重建样本阵列的解码器,所述解码器包括微处理器或电子电路或计算机,所述微处理器或电子电路被配置为,或所述计算机被编程为:

对所述熵编码数据流内的多个熵片段进行熵解码以重建分别与所述熵片段相关联的所述样本阵列的不同部分,其中,每个熵片段在其中对所述样本阵列的对应部分的数据进行熵编码,所述不同部分形成所述样本阵列的区块的行,所述区块按行和列规则排列,使得与所述熵片段对应的部分由相同数量的所述区块组成,其中,所述熵片段被细分为组块,每个所述组块设有揭示其在所述样本阵列中的起始位置的组块报头,其中,对所述多个熵片段进行熵解码包括:

针对每个熵片段沿相应的熵编码路径使用相应的概率估计来执行所述熵解码,其中,所述熵编码路径沿所述区块的行并行指向,

沿所述相应的熵编码路径使用相应的熵片段的先前解码部分来调试所述相应的概率估计,

使用熵片段顺序开始对所述多个熵片段依次进行熵解码,并且

在对预定熵片段进行熵解码的过程中,基于使用所述预定熵片段的先前已解码部分进行所述调试的所述预定熵片段的相应概率估计,来对所述预定熵片段的当前部分执行所述熵解码,

存储与预定熵片段对应的部分的第二区块进行熵解码之后显示出来的概率估计,用于在沿相应的编码路径解码按熵片段顺序与相应熵片段对应的部分的第一区块之前的概率估计初始化,

检查当前组块是否与跟所述预定熵片段对应的部分的沿所述熵编码路径的第一子部分对应,并且

如果对应,则在沿相应的熵编码路径解码与所述预定熵片段对应的部分的第一区块之前执行所述概率估计的初始化,所述概率估计在沿相应的编码路径对按所述熵片段顺序在前的熵片段对应的部分的第二区块进行熵解码之后显现出来,在调试相应的概率估计的过程中熵解码所述当前组块,并当熵解码与所述预定熵片段的部分的沿所述熵编码路径的第二子部分对应的另一组块时,将对所述当前组块进行熵解码结束时显示出来的相应的所述概率估计的状态保持不变纳入考虑,并且

如果不对应,则沿熵编码路径通过使在与对所述预定熵片段的部分中的与所述当前组块对应的子部分之前的子部分对应的组块进行熵解码结束时显现出来的相应的概率估计的状态保持不变,在当前片段恢复对所述预定熵片段的所述熵解码。

2.根据权利要求1所述的解码器,其中,所述微处理器或电子电路被配置为,或所述计算机被编程为:操纵按所述熵片段顺序对直接连续的熵片段进行的熵解码,以防止与沿所述编码路径在区块中测量的所述直接连续的熵片段对应的部分的当前解码区块的距离变得比两个区块小。

3.根据权利要求1所述的解码器,其中,所述微处理器或电子电路被配置为,或所述计算机被编程为:操纵按所述熵片段顺序对直接连续的熵片段进行的熵解码,以与沿所述编码路径在区块中测量的所述直接连续的熵片段对应的部分的当前解码区块的距离仍是两个区块。

4.根据权利要求1所述的解码器,其中,所述解码器包括对所述组块进行解交错的解交

错器并被配置为即使在整体接收所述熵片段中的任意一个之前，沿所述熵解码路径开始对所述熵片段进行并行熵解码。

5.根据权利要求1所述的解码器，其中，所述样本阵列是样本阵列序列的当前样本阵列，并且所述微处理器或电子电路被配置为，或所述计算机被编程为：在对熵片段顺序中第一熵片段进行熵解码的过程中，使用前一阵列的熵解码中使用的概率估计的结束状态初始化所述熵片段顺序中第一熵片段的相应的概率估计。

6.根据权利要求1所述的解码器，其中，所述熵编码数据流是时序分层数据流。

7.一种用于将样本阵列编码成熵编码数据流的编码器，所述编码器包括微处理器或电子电路或计算机，所述微处理器或电子电路被配置为，或所述计算机被编程为：

将多个熵片段熵编码成熵编码器数据流，每个熵片段都分别与所述样本阵列的不同部分相关联，使得每个熵片段在其中对所述样本阵列的对应部分的数据进行熵编码，所述不同部分形成所述样本阵列的区块的行，所述区块按行和列规则排列，使得与所述熵片段对应的部分由相同数量的所述区块组成，其中，所述熵片段被细分为组块，每个所述组块设有揭示其在所述样本阵列中的起始位置的组块报头，其中，对所述多个熵片段进行熵编码包括：

针对每个熵片段沿相应的熵编码路径使用相应的概率估计来执行所述熵编码，其中，所述熵编码路径沿所述区块的行并行指向，

沿所述相应的熵编码路径使用相应的所述熵片段的先前编码部分来调试所述相应的概率估计，

使用熵片段顺序开始对所述多个熵片段依次进行所述熵编码，并且

在对预定熵片段进行所述熵编码的过程中，基于使用所述预定熵片段的先前编码部分进行所述调试的所述预定熵片段的相应概率估计，来对所述预定熵片段的当前部分执行所述熵编码，

存储与预定熵片段对应的部分的第二区块进行熵编码之后显示出来的概率估计，用于在沿相应的编码路径编码按熵片段顺序与相应熵片段对应的部分的第一区块之前的概率估计初始化，

检查当前组块是否与跟所述预定熵片段对应的部分的沿所述熵编码路径的第一子部分对应，并且

如果对应，则在沿相应的熵编码路径编码与所述预定熵片段对应的部分的第一区块之前执行所述概率估计的初始化，所述概率估计在沿相应的编码路径对按所述熵片段顺序在前的熵片段对应的部分的第二区块进行熵编码之后显现出来，在调试相应的概率估计的过程中熵编码所述当前组块，并当熵编码与所述预定熵片段的部分的沿所述熵编码路径的第二子部分对应的另一组块时，将对所述当前组块进行熵解码结束时显示出来的相应的所述概率估计的状态保持不变纳入考虑，并且

如果不对应，则沿熵编码路径通过使在与对所述预定熵片段的部分中的与所述当前组块对应的子部分之前的子部分对应的组块进行熵编码结束时显现出来的相应的概率估计的状态保持不变，在当前片段恢复对所述预定熵片段的所述熵编码。

8.根据权利要求7所述的编码器，其中，所述熵编码数据流是时序分层数据流。

9.一种用于从熵编码数据流重建样本阵列的方法，包括：

对所述熵编码数据流内的多个熵片段进行熵解码以重建分别与所述熵片段相关联的所述样本阵列的不同部分，其中，每个熵片段在其中对所述样本阵列的对应部分的数据进行熵编码，所述不同部分形成所述样本阵列的区块的行，所述区块按行和列规则排列，使得与所述熵片段对应的部分由相同数量的所述区块组成，其中，所述熵片段被细分为组块，每个所述组块设有揭示其在所述样本阵列中的起始位置的组块报头，其中，对所述多个熵片段进行熵解码包括：

针对每个熵片段沿相应的熵编码路径使用相应的概率估计来执行所述熵解码，其中，所述熵编码路径沿所述区块的行并行指向，

沿所述相应的熵编码路径使用相应的所述熵片段的先前解码部分来调试相应的所述概率估计，

使用熵片段顺序开始对所述多个熵片段依次进行所述熵解码，并且

在对预定熵片段进行熵解码的过程中，基于使用所述预定熵片段的先前解码部分进行所述调试的所述预定熵片段的相应的概率估计，来对所述预定熵片段的当前部分执行所述熵解码，

存储与预定熵片段对应的部分的第二区块进行熵解码之后显示出来的概率估计，用于在沿相应的编码路径解码按熵片段顺序与相应熵片段对应的部分的第一区块之前的概率估计初始化，

检查当前组块是否与跟所述预定熵片段对应的部分的沿所述熵编码路径的第一子部分对应，并且

如果对应，则在沿相应的熵编码路径解码与所述预定熵片段对应的部分的第一区块之前执行所述概率估计的初始化，所述概率估计在沿相应的编码路径对按所述熵片段顺序在前的熵片段对应的部分的第二区块进行熵解码之后显现出来，在调试相应的概率估计的过程中熵解码所述当前组块，并当熵解码与所述预定熵片段的部分的沿所述熵编码路径的第二子部分对应的另一组块时，将对所述当前组块进行熵解码结束时显示出来的相应的所述概率估计的状态保持不变纳入考虑，并且

如果不对应，则沿熵编码路径通过使在与对所述预定熵片段的部分中的与所述当前组块对应的子部分之前的子部分对应的组块进行熵解码结束时显现出来的相应的概率估计的状态保持不变，在当前片段恢复对所述预定熵片段的所述熵解码。

10.一种用于将样本阵列编码成熵编码数据流的方法，包括：

将多个熵片段熵编码成所述熵编码器数据流，每个熵片段都分别与所述样本阵列的不同部分相关联，使得每个熵片段在其中对所述样本阵列的对应部分的数据进行熵编码，所述不同部分形成所述样本阵列的区块的行，所述区块按行和列规则排列，使得与所述熵片段对应的部分由相同数量的所述区块组成，其中，所述熵片段被细分为组块，每个所述组块设有揭示其在所述样本阵列中的起始位置的组块报头，其中，对所述多个熵片段进行熵编码包括：

针对每个熵片段沿相应的熵编码路径使用相应的概率估计来执行所述熵编码，其中，所述熵编码路径沿所述区块的行并行指向，

沿所述相应的熵编码路径使用相应的所述熵片段的先前编码部分来调试相应的概率估计，

使用熵片段顺序开始对所述多个熵片段依次进行所述熵编码，并且

在对预定熵片段进行熵编码的过程中，基于使用所述预定熵片段的先前编码部分进行所述调试的所述预定熵片段的相应的概率估计，来对所述预定熵片段的当前部分执行所述熵编码，

存储与预定熵片段对应的部分的第二区块进行熵编码之后显示出来的概率估计，用于在沿相应的编码路径编码按熵片段顺序与相应熵片段对应的部分的第一区块之前的概率估计初始化，

检查当前组块是否与跟所述预定熵片段对应的部分的沿所述熵编码路径的第一子部分对应，并且

如果对应，则在沿相应的熵编码路径编码与所述预定熵片段对应的部分的第一区块之前执行所述概率估计的初始化，所述概率估计在沿相应的编码路径对按所述熵片段顺序在前的熵片段对应的部分的第二区块进行熵编码之后显现出来，在调试相应的概率估计的过程中熵编码所述当前组块，并当熵编码与所述预定熵片段的部分的沿所述熵编码路径的第二子部分对应的另一组块时，将对所述当前组块进行熵解码结束时显示出来的相应的所述概率估计的状态保持不变纳入考虑，并且

如果不对应，则沿熵编码路径通过使在与对所述预定熵片段的部分中的与所述当前组块对应的子部分之前的子部分对应的组块进行熵编码结束时显现出来的相应的概率估计的状态保持不变，在当前片段恢复对所述预定熵片段的所述熵编码。

11.用于解码样本阵列的方法，其中，所述方法包括：

接收并解码通过根据权利要求10所述的方法将视频编码成的数据流。

12.用于解码数据流的方法，其中，所述方法包括：

接收并解码已经编码的数据流，其中，如果样本阵列已经由解码器解码，则使解码器解码通过执行根据权利要求9所述的方法由所述数据流得到的视频。

13.存储样本阵列的方法，包括：

在数字存储介质上存储通过根据权利要求10所述的方法将视频编码成的数据流。

14.用于传输图片的方法，包括：

在传输介质上传输通过根据权利要求10所述的方法将样本阵列编码成的数据流。

15.根据权利要求11至14中任一项所述的方法，其中，所述熵编码数据流是分层数据流。

16.计算机可读数字存储介质，存储有计算机程序，所述计算机程序被配置为在由计算机执行时指示所述计算机执行根据权利要求9至10中任一项所述的方法。

解码器和方法、编码器和方法及存储介质

[0001] 本申请为国际申请日为2012年7月16日、国际申请号为PCT/EP2012/063929、发明名称为“低延迟的样本阵列编码”的中国国家阶段申请的分案申请，该中国国家阶段申请的进入国家阶段日为2014年3月11日、申请号为201280044181.7、发明名称为“低延迟的样本阵列编码”。

技术领域

[0002] 本申请涉及样本阵列编码比如图片或视频编码。

背景技术

[0003] 由于HEVC标准以及视频分辨率预期增加的处理要求提高，对编码器和解码器进行并行化处理是非常重要的。多核架构在广泛的现代电子设备中变得可用。因此，需要能够使用多核架构的有效方法。

[0004] 在光栅扫描过程中可进行LCU的编码或解码，CABAC概率据此适应每个图像的特异性。空间依赖性存在于相邻的LCU之间。由于比如运动矢量、预测、帧内预测等元素不同，每个LCU(最大编码单元)取决于其左边、上方、左上和右上的相邻LCU。由于在解码过程中能够进行并行化处理，因此这些依赖性通常需要被中断或在最新应用中被中断。

[0005] 提出了某些并行化概念，即波阵面。进一步研究的激励因素是执行降低了编码效率损失并由此减轻了对编码器和解码器中的并行化方法的比特流的负担的技术。此外，利用可用技术进行低延迟处理是不可能的。

发明内容

[0006] 由此，本发明的目的在于提供样本阵列的编码概念，由此以编码效率相对较少的损失的方式允许出现低延迟。

[0007] 该目的通过所附的独立权利要求的主题来实现。

[0008] 如果对预定熵片段的当前部分进行熵编码不但基于如使用预定熵片段的先前编码部分调试的预定熵片段的相应概率估计，而且还基于如用于以熵片段顺序在前的空间相邻的熵片段在其相邻部分的进行熵编码的概率估计，则用于熵编码的概率估计更紧密适用于实际符号统计，由此降低通常由较低延迟概念导致的编码效率下降。另外或可选地利用时间相互关系。

[0009] 例如，对如用于按熵片段顺序在前的空间相邻的熵片段进行熵编码的概率估计的依赖性可以涉及在对预定熵片段进行熵编码开始时初始化概率估计。通常，概率估计被初始化为适用于样本阵列材料的代表性混合的符号统计的值。为了避免传输概率估计的初始化值，该初始化值按照惯例对编码器和解码器来说是已知的。然而，这些预先定义的初始化值通常只是边信息位率与编码效率之间的折衷，因为这些初始化值通常或多或少会偏离当前编码样本阵列材料的实际样本统计。编码熵片段期间的概率调试使概率估计适应实际符号统计。通过使用对刚提及的按熵编码顺序在前的空间相邻的熵片段的已经调试的概率估

计在熵编码当前/预定熵片段的开始阶段初始化概率估计来加速该处理，因为前者的值在某种程度上已经适用于当前正在考虑的样本阵列的实际符号统计。然而，通过在对预定/当前熵片段的概率估计进行初始化的过程中使用在其相邻部分使用的概率估计，而不是通过在对先前熵片段进行熵编码结束时显示出概率估计，来实现低延迟编码。采用该措施，波阵面处理是可能的。

[0010] 此外，上述的对如用于按熵编码顺序在前的空间相邻的熵片段进行熵编码的概率估计的依赖性可以涉及调试用于对当前/预定熵片段本身进行熵编码的概率估计的调试处理。概率估计调试涉及使用刚刚编码的部分，即，刚刚编码的符号，以便使概率估计的当前状态适应实际符号统计。采用该措施，初始化的概率估计以某个调试速率(adaption rate)适应实际符号统计。该调试速率通过不仅基于当前/预定熵片段的当前编码符号，而且还根据按熵编码顺序在前的空间相邻的熵片段的相邻部分显示自身的概率估计来执行刚提及的概率估计调试来增加。再者，通过适当选择当前熵片段的当前部分的空间相邻邻域和先前熵片段的相邻部分，波阵面仍然是可能的。因沿当前熵片段耦合自身的概率估计调试与先前熵片段的概率调试而得到的好处是增加的速率，针对实际符号统计的调试以此速率发生，因为在当前和先前熵片段中遍历(traverse)的符号的数量，而不仅仅是当前熵片段的符号的数量，有助于调试。

[0011] 本发明提供了一种用于从熵编码数据流重建样本阵列(10)的解码器，其被配置为：对所述熵编码器数据流内的多个熵片段进行熵解码以重建分别与所述熵片段相关联的所述样本阵列的不同部分(12)，其中，针对每个熵片段沿相应的熵编码路径(14)使用相应的概率估计来执行所述熵解码，沿所述相应的熵编码路径使用相应的熵片段的先前解码部分来调试所述相应的概率估计，使用熵片段顺序(16)开始对所述多个熵片段依次进行熵解码，并且在对预定熵片段进行熵解码的过程中，基于如使用所述预定熵片段的先前已解码部分调试的所述预定熵片段的相应概率估计，以及如在空间相邻熵片段的相邻部分的按所述熵片段顺序空间相邻的先前熵片段的熵解码中使用的概率估计，来对所述预定熵片段的当前部分执行所述熵解码。

[0012] 其中，所述不同部分是所述样本阵列的区块的行。

[0013] 其中，所述熵片段顺序经选择使得沿所述熵片段顺序，所述不同部分在相对于所述熵片段的熵编码路径(14)成角度的方向(16)上相互跟随，所述熵编码路径又基本上彼此并行延伸。

[0014] 其中，每个熵片段在其中对所述样本阵列的对应部分的数据进行熵编码，所述不同部分形成所述样本阵列的区块的行，所述区块按行和列规则排列，使得与所述熵片段对应的部分由相同数量的所述区块组成并且所述熵编码路径沿所述区块的行并行指向，其中，所述解码器被配置为针对每个熵片段(90)在沿相应的编码路径(14)解码与相应的熵片段对应的部分(12)的第一区块之前针对相应的熵片段对所述概率估计(94)执行初始化，所述概率估计在沿相应的编码路径对按所述熵片段顺序(16)在前的熵片段对应的部分(12)的第二区块(50)进行熵解码之后显现出来。

[0015] 其中，所述解码器被配置为存储在沿相应的编码路径对与按所述熵片段顺序在前的熵片段对应的部分的第二区块进行熵解码之后显示出来的概率估计，并在沿相应的编码路径解码与相应熵片段对应的部分的第一区块之前，使用所存储的概率估计进行初始化。

[0016] 其中，每个熵片段在其中对所述样本阵列的对应部分的数据进行熵编码，所述不同部分形成所述样本阵列的区块的行，所述区块按行和列规则排列，使得与所述熵片段对应的部分由相同数量的所述区块组成并且所述熵编码路径沿所述区块的行并行指向，其中，所述解码器被配置为针对每个熵片段沿相应的熵编码路径执行熵解码并沿相应的熵编码路径对相应的所述概率估计执行调试，使得在基于所述预定熵片段的相应的概率估计(94)对所述预定熵片段的当前部分进行熵解码之后，根据所述预定熵片段的当前部分、以及在对空间相邻的所述熵片段的相邻部分进行熵解码的过程中显现出来的概率估计，来调试所述预定熵片段的相应的概率估计(94)。

[0017] 其中，所述解码器被配置为使得所述预定熵片段的相应概率估计的调试，在基于所述预定熵片段的相应概率估计对所述预定熵片段的当前部分进行熵解码之后，通过根据所述预定熵片段的当前部分的第一调试和对如用于空间相邻的熵片段的相邻部分的熵解码的概率估计的所述第一调试的结果的平均化来执行。

[0018] 其中，所述解码器被配置为操纵按所述熵片段顺序对直接连续的熵片段进行的熵解码，以防止与沿所述编码路径在区块中测量的所述直接连续的熵片段对应的部分的当前解码区块的距离变得比两个区块小。

[0019] 其中，所述解码器被配置为操纵按所述熵片段顺序对直接连续的熵片段进行的熵解码，以与沿所述编码路径在区块中测量的所述直接连续的熵片段对应的部分的当前解码区块的距离仍是两个区块。

[0020] 其中，所述熵片段被细分为组块，所述解码器包括对所述组块进行解交错的解交错器并被配置为即使在整体接收所述熵片段中的任意一个之前，沿所述熵解码路径开始对所述熵片段进行并行熵解码。

[0021] 其中，所述熵片段被细分为组块且所述解码器配置为：检查当前组块是否与所述预定熵片段的部分的沿所述熵编码路径的第一子部分对应，并且如果对应，则在调试相应的概率估计的过程中熵解码所述当前组块，并当熵解码与所述预定熵片段的部分的沿所述熵编码路径的第二子部分对应的另一组块时，将对所述当前组块进行熵解码结束时显示出来的相应的所述概率估计的状态纳入考虑，并且如果不对应，则沿熵编码路径使用取决于在与对所述预定熵片段的部分中的与所述当前组块对应的子部分之前的子部分对应的组块进行熵解码结束时显现出来的概率估计的概率估计来对所述当前组块进行所述熵解码。

[0022] 其中，所述样本阵列(10)是样本阵列序列的当前样本阵列并且所述解码被配置为在对预定熵片段进行熵解码的过程中，基于如使用所述预定熵片段的先前解码部分调试的所述预定熵片段的相应概率估计、如用于对在空间相邻的所述熵片段的相邻部分的空间相邻的按所述熵片段顺序在前的熵片段进行熵解码的概率估计、以及用于解码与除所述当前样本阵列之外的另一样本阵列有关的熵编码数据流的先前解码帧的概率估计，来对所述预定熵片段的所述当前部分进行所述熵解码。

[0023] 本发明提供了一种配置为从熵编码数据流重建样本阵列序列的解码器，其被配置为：对所述熵编码数据流的当前帧进行熵解码以重建所述样本阵列序列的当前样本阵列，沿熵编码路径并使用概率估计来执行所述熵解码，并且沿所述熵编码路径使用所述当前帧的先前解码部分来调试概率估计，其中，所述解码器被配置为基于用于解码所述熵编码数据流的所述先前解码帧的概率估计来初始化或确定所述当前帧的概率估计。

[0024] 其中，所述熵解码阶段被配置为基于在完成所述熵编码数据流的先前解码帧的解码之后产生的概率估计来初始化所述当前帧的概率估计。

[0025] 其中，所述熵解码阶段被配置为基于如使用所述预定熵片段的所述先前解码部分调试的所述预定熵片段的相应的概率估计、以及如用于对所述先前解码帧的熵片段的空间对应部分进行熵解码的概率估计、以及可选地如对在空间相邻的所述熵片段的相邻部分的空间相邻的按熵片段顺序在前的熵片段进行的熵解码中使用的概率估计来执行所述调试。

[0026] 其中，所述熵解码阶段被配置为基于预测参考来选择所述先前解码帧的熵片段的空间对应部分和/或空间相邻的所述熵片段的相邻部分。

[0027] 本发明提供了一种用于将样本阵列(10)编码成熵编码数据流的编码器，被配置为：将多个熵片段熵编码成熵编码器数据流，每个熵片段都分别与所述样本阵列的不同部分(12)相关联，其中，针对每个熵片段沿相应的熵编码路径(14)使用相应的概率估计来执行所述熵编码，沿所述相应的熵编码路径使用相应的所述熵片段的先前解码部分来调试所述相应的概率估计，使用熵片段顺序(16)开始对所述多个熵片段依次进行所述熵编码，并且在对预定熵片段进行所述熵编码的过程中，基于如使用所述预定熵片段的先前编码部分调试的所述预定熵片段的相应概率估计、以及如对在空间相邻的所述熵片段的相邻部分的空间相邻的按熵片段顺序在前的熵片段进行熵编码中使用的概率估计，来对所述预定熵片段的所述当前部分执行所述熵编码。

[0028] 本发明提供了一种配置为将样本阵列序列编码成熵编码数据流的编码器，被配置为：对所述熵编码数据流的当前帧进行熵编码以重建所述样本阵列序列的当前样本阵列，沿熵编码路径并使用概率估计来执行所述熵编码，并且沿所述熵编码路径使用所述当前帧的先前编码部分来调试所述概率估计，其中，所述编码器被配置为基于用于编码所述熵编码数据流的先前编码帧的概率估计来初始化或确定所述当前帧的概率估计。

[0029] 本发明提供了一种用于从熵编码数据流重建样本阵列(10)的方法，包括：对所述熵编码器数据流内的多个熵片段进行熵解码以重建分别与所述熵片段相关联的所述样本阵列的不同部分(12)，其中，针对每个熵片段沿相应的熵编码路径(14)使用相应的概率估计来执行所述熵解码，沿所述相应的熵编码路径使用相应的所述熵片段的先前解码部分来调试相应的所述概率估计，使用熵片段顺序(16)开始对所述多个熵片段依次进行所述熵解码，并且在对所述预定熵片段进行熵解码的过程中，基于如使用所述预定熵片段的先前解码部分调试的所述预定熵片段的相应的概率估计，以及如对在空间相邻的所述熵片段的相邻部分的空间相邻的按所述熵片段顺序在前的熵片段进行的解码中使用的概率估计来对所述预定熵片段的所述当前部分执行所述熵解码。

[0030] 本发明提供了一种配置为从熵编码数据流重建样本阵列序列的方法，包括：对所述熵编码数据流的当前帧进行熵解码以重建所述样本阵列序列的当前样本阵列，沿熵编码路径并使用概率估计来执行所述熵解码，并且沿所述熵编码路径使用所述当前帧的先前解码部分来调试所述概率估计，并且其中，所述方法包括基于用于解码所述熵编码数据流的先前解码帧的概率估计来初始化或确定所述当前帧的概率估计。

[0031] 本发明提供了一种用于将样本阵列(10)编码成熵编码数据流的方法，包括：将多个熵片段熵编码成所述熵编码器数据流，每个熵片段都分别与所述样本阵列的不同部分(12)相关联，其中，针对每个熵片段沿相应的熵编码路径(14)使用相应的概率估计来执行

所述熵编码，沿所述相应的熵编码路径使用相应的所述熵片段的先前解码部分来调试相应的概率估计，使用熵片段顺序(16)开始对所述多个熵片段依次进行所述熵编码，并且在对预定熵片段进行熵编码的过程中，基于如使用所述预定熵片段的先前编码部分调试的所述预定熵片段的相应的概率估计、以及如对在空间相邻的所述熵片段的相邻部分的空间相邻的按熵片段顺序在前的熵片段进行的熵编码中使用的概率估计，来对所述预定熵片段的当前部分执行所述熵编码。

[0032] 本发明提供了一种用于将样本阵列序列编码成熵编码数据流的方法，包括：对所述熵编码数据流的当前帧进行熵编码以重建所述样本阵列序列的当前样本阵列，沿所述熵编码路径并使用概率估计来执行所述熵编码，并且沿所述熵编码路径使用所述当前帧的先前编码部分来调试所述概率估计，其中，所述方法包括基于用于编码所述熵编码数据流的先前编码帧的概率估计来初始化或确定所述当前帧的所述概率估计。

[0033] 本发明提供了一种具有配置为在计算机上运行时执行上述方法的程序代码的计算机程序。

附图说明

[0034] 本发明的实施例的有利实现是从属权利要求的主旨。此外，针对图对优选实施例进行描述，在图中：

[0035] 图1示出了示例性编码器的框图；

[0036] 图2示出了以其中限定的编码顺序将图片划分为片段和片段部分(即，区块或编码单元)的示意图；

[0037] 图3示出了示例性编码器比如图1的示例性编码器的功能的流程图；

[0038] 图4示出了用于阐述示例性编码器比如图1的示例性编码器的功能的示意图；

[0039] 图5示出了编码器和解码器并行操作实现的示意图；

[0040] 图6示出了示例性解码器的框图；

[0041] 图7示出了示例性解码器比如图6的示例性解码器的功能的流程图；

[0042] 图8示出了由图1到图6的编码方案产生的示例性比特流的示意图；

[0043] 图9示意性地示出了如何在其他LCU的帮助下计算概率的实例；

[0044] 图10示出了与HM3.0相比的帧内(4线程)的RD结果的示图；

[0045] 图11示出了与HM3.0相比的低延迟(1线程)的RD结果的示图；

[0046] 图12示出了与HM3.0相比的随机存取(1线程)的RD结果的示图；

[0047] 图13示出了与HM3.0相比的低延迟(4线程)的RD结果的示图；

[0048] 图14示意性地且示例性地示出了熵片段的可能复合物；

[0049] 图15示意性地且示例性地示出了熵片段的可能信令；

[0050] 图16示意性地且示例性地示出了经由组块编码、分割、交错和解码熵片段数据；

[0051] 图17示意性地且示例性地示出了帧之间的可能复合物；

[0052] 图18示意性地且示例性地示出了并置信息的可能用途；

[0053] 图19示意性地示出了波阵面在由连续样本阵列跨越的时空空间内倾斜运行的可能性；

[0054] 图20示意性地示出了将熵片段细分为组块的另一实例。

具体实施方式

[0055] 为了轻松理解下面所列的措施以编码效率的较少损失而提高低延迟的效果，首先用更一般的术语描述了图1的编码器，而没有初步讨论本申请的实施例的有利概念以及如何同样可以构建成图1的实施例。然而，应提及的是，图1中的结构只用作可以使用本申请的实施例的说明性环境。也简要讨论了根据本发明实施例的编码器和解码器的推广和替代。

[0056] 图1示出了用于将样本阵列10编码成熵编码数据流20的编码器。如图1所示，样本阵列10可以是样本阵列序列30中的一个，编码器可以配置为将序列30编码成数据流20。

[0057] 图1的编码器通常用参考标号40表示且包括预编码器42，接着是熵编码阶段(stage)44，其输出端输出数据流20。预编码器42配置为接收并对样本阵列10进行操作以便通过预定语法的语法元素来描述其内容，其中每个语法元素属于预定的一组语法元素类型中的相应一个，这些类型又与相应语义相关联。

[0058] 在使用语法元素描述样本阵列10的过程中，预编码器42可以将样本阵列10细分为编码单元50。出于例如可以是下面更详细陈述的原因，术语“编码单元”可选地称为“编码树单元”(CTU)。图2中示例性地示出了预编码器42如何将样本阵列10细分为编码单元50的一个可能性。根据该实例，细分定期将样本阵列10细分为编码单元50，使得后者按行和按列排列以在不重叠的情况下无间隙地覆盖完整的样本阵列10。换句话说，预编码器42可以配置为通过语法元素来描述每个编码单元50。这些语法元素中的一些可以形成细分信息以更进一步细分各编码单元50。例如，通过多树细分，细分信息可以描述将各编码单元50细分为预测区块52，预编码器42使预测模式与这些预测区块52的每一个的相关联预测参数相关联。该预测细分可以允许预测区块52的尺寸不同，如图2所示。预编码器42还可以使残差(residual)细分信号与预测区块52相关联以便进一步将预测区块52细分为残差区块54以便描述每个预测区块52的预测残差。因此，预编码器可以配置为根据混合编码方案来生成样本阵列10的语法描述。然而，正如上面已经提到的，刚提及的预编码器42通过语法元素来描述样本阵列10的方式仅出于说明目的提出且还可以通过不同的方式实现。

[0059] 预编码器42可以利用样本阵列10的相邻编码单元50的内容之间的空间相互关系。例如，预编码器42可以根据针对与当前编码的编码单元50空间相邻的先前编码的编码单元50确定的语法元素来预测某个编码单元50的语法元素。在图1和图2中，例如，左上邻近部分用于进行预测，如箭头60和62所示。而且，预编码器42在帧内预测模式下可以将相邻编码单元50的已经编码内容外插至当前编码单元50以获得当前编码单元50的样本的预测。如图1所示，预编码器42除了利用空间相互关系之外还可以根据先前编码的样本阵列时间地(temporarily)预测样本和/或当前编码单元50的语法元素，如图1中通过箭头64说明性示出的。也就是说，运动补偿预测可以被预编码器42使用，并且运动矢量本身可以根据先前编码的样本阵列的运动矢量进行时间预测。

[0060] 也就是说，预编码器42可以以编码单元的方式来描述样本阵列10的内容，并且为此可以使用空间预测。空间预测对于每个编码单元50来说要受相同样本阵列10的空间相邻编码单元的限制，使得当遵循在样本阵列10的编码单元50之间的编码顺序66时，用作空间预测的预测参考的相邻编码单元通常在当前编码单元50之前通过编码顺序66来遍历。如图2所示，编码单元50之间限定的编码顺序66例如可以是光栅扫描顺序，编码单元50根据该顺序从上到下逐行遍历。任选地，将阵列10细分为瓦片(tile)阵列可能会导致在进行到瓦片

顺序中的下一个之前，扫描顺序66按光栅扫描顺序首先遍历构成一个瓦片的编码单元50，反过来，瓦片顺序也可以是光栅扫描类型。例如，空间预测只可以涉及当前编码单元50所在的编码单元行上方的编码单元行内的相邻编码单元50，以及相同编码单元行内的但相对于当前编码单元向左的编码单元。如下面更详细的阐述，对空间相互关系/空间预测的限制使得可以进行并行波阵面处理。

[0061] 预编码器42将语法元素转发至熵编码阶段44。如刚才概述的，这些语法元素中的一些已经被预测编码，即表示预测残差。预编码器42因此可以被视为预测编码器。除此之外，预编码器42可以是配置为变换编码单元50的内容的预测的编码残差的变换编码器。

[0062] 图1中还示出了熵编码阶段44的示例性内部结构。如图所示，熵编码阶段44任选可以包括用于转换从预编码器42接收的每个语法元素的符号化器，其可能状态的数量超过符号字母(alphabet)的符号s_i序列的符号字母基数，熵编码引擎44基于该符号字母操作。除任选符号化器70之外，熵编码引擎44可以包括上下文选择器72以及初始化器74、概率估计管理器76、概率估计调试器78及熵编码核80。熵编码核的输出形成熵编码阶段44的输出。此外，熵编码核80包括两个输入，即一个用于接收符号序列的符号s_i，另一个用于接收每个符号的概率估计p_i。

[0063] 由于熵编码的属性，就压缩率方面的编码效率随概率估计的提高而提高：概率估计与实际符号统计越好的匹配，压缩率就越好。

[0064] 在图1的实例中，上下文选择器72配置为针对每个符号s_i在由管理器76管理的一组可用上下文中选择对应的上下文c_i。然而，应注意的是，上下文选择只形成任选特征并且可以通过使用每个符号的相同上下文来省略(leave off)。然而，如果使用上下文选择，上下文选择器72可以配置为至少部分基于关于当前编码单元外侧的编码单元，即关于上文讨论的有限邻近关系内的相邻编码单元的信息来执行上下文选择。

[0065] 管理器76包括存储装置，该存储装置针对每个可用上下文存储相关联的概率估计。例如，符号字母可以是二元字母，使得只有一个概率值可能必须针对每个可用上下文存储。

[0066] 初始化器74可以间歇地初始化或重新初始化针对可用上下文存储在管理器76内的概率估计。下面将进一步讨论可以执行初始化的可能时刻。

[0067] 调试器78访问多对符号s_i以及对应概率估计p_i并因此调试管理器76内的概率估计。也就是说，每当概率估计由熵编码核80应用以便将各符号s_i熵编码成数据流20时，调试器78可以根据当前符号s_i的值来改变概率估计，使得在编码与此概率估计相关联的下一符号(通过上下文)时概率估计p_i更好适应实际符号统计。也就是说，调试器78接收从管理器76中选择的上下文的概率估计和对应符号s_i并因此调试概率估计p_i，使得针对相同上下文c_i的下一符号s_i使用自适应概率估计。

[0068] 熵编码核80例如配置为根据算术编码方案或概率区间划分熵编码方案来操作。在算术编码的过程中，当编码符号序列时，熵编码核80例如连续更新其状态，该状态由概率区间限定，概率区间由概率区间宽度值和概率区间偏移值限定。当在管路(pipe)概念下操作时，熵编码核80例如将概率估计的可能值的域细分为不同区间，同时相对于这些区间的每一个执行固定概率熵编码，由此获得每个子区间的子流，其编码效率被分别定制为相关联的概率区间。就熵编码而言，数据流20输出可以是信号传送解码侧信息的算术编码数据流，

从而可以模仿或重新进行区间细分处理。

[0069] 自然，熵编码状态44可以对与样本阵列10有关的所有信息，即，所有语法元素/符号s_i进行熵编码，在其开始时只初始化概率估计一次，然后通过调试器(adapter)78继续更新概率估计。然而，这会产生必须在解码侧依次解码的数据流20。换句话说，任何解码器不可能将由此产生的数据流细分为几个子部分且并行解码子部分。这又会阻碍任何低延迟作用。

[0070] 因此，如下面更详细所述，最好将描述样本阵列10的一定量的数据细分为所谓的熵片段。这些熵片段的每一个由此覆盖与样本阵列10有关的一组不同的语法元素。然而，如果熵编码阶段44通过首先初始化概率估计一次，然后单独继续更新每个熵片段的概率估计来相互完全独立地对每个熵片段进行熵编码，则编码效率由于与样本阵列10有关并描述该样本阵列10的数据的百分比增加而降低，所使用的概率估计针对此也不太精确适应实际符号统计。

[0071] 为了克服在满足低编码延迟和高编码效率的愿望的过程中出现的刚提及的问题，可以使用以下编码方案，现在针对图3对其进行描述。

[0072] 首先，将描述样本阵列10的数据细分为在下文中称为“熵片段”的部分。细分80可以重叠。另一方面，该细分可以至少部分对应于将样本阵列10分为不同部分的空间细分。也就是说，根据细分80，可以根据编码单元50的各语法元素与之有关的位置来将描述样本阵列10的语法元素分布在不同熵片段上。例如参见图2。图2示出了将样本阵列10示例性细分为不同部分12。每个部分都与各熵片段对应。如示例性所示，每个部分12都与编码单元50的行对应。然而，其他细分也是可行的。然而，最好是将样本阵列10细分为部分12的细分符合上述编码顺序66，使得部分12沿编码顺序66覆盖连续运行的编码单元12。然而，即便如此，沿编码顺序66的部分12的开始位置和结束位置不需要分别与编码单元50行的左侧边缘和右侧边缘一致。甚至与编码单元50的边界一致立即相互符合，编码顺序66也不需要是强制性的。

[0073] 通过如此细分样本阵列10，熵片段顺序16在部分12之间限定，部分12沿此沿编码顺序66相互跟随。而且，针对每个熵片段，限定各熵编码路径14，即运行至各部分12的编码路径66的分段。在图2的实例中，其中部分12与编码单元50的行一致，每个熵片段的熵编码路径14并行于彼此，即这里是从左侧至右侧，沿行方向指向。

[0074] 应注意的是，可以限制由预编码器42执行的空间预测以及由上下文选择器72执行的上下文推导以便不会超越片段边界，即，使得空间预测和上下文选择不取决于与另一个熵片段对应的数据。这样，“熵片段”将对应于H.264中的“片段”的惯常定义，例如，其可独立于彼此全部解码，下面所列的概率初始化/调试依赖性除外。然而，允许空间预测和上下文选择，即一般来说为依赖性，超越片段边界以便利用本地/空间相关性同样是可行的，因为WPP处理仍然可行，即使关于预编码的反演，即基于语法元素的重建，和熵上下文选择也是如此。到目前为止，熵片段将或多或少对应于“依赖性片段”。

[0075] 细分80例如可以由熵编码阶段44执行。细分可以是固定的或者可以在序列30阵列之间变化。细分按照约定可以是固定的或者可以在数据流20内进行信号传送。

[0076] 基于熵片段，实际熵编码可能会发生，即，82。针对每个熵片段，熵编码可以被构造成起始阶段84和连续阶段86。起始阶段84例如涉及初始化概率估计、以及触发各熵片段的

实际熵编码处理。然后在连续阶段86期间执行实际熵编码。沿相应熵编码路径14执行阶段86期间的熵编码。每个熵片段的起始阶段84经控制使得多个熵片段的熵编码使用熵片段顺序16依次启动。

[0077] 现在，为了避免对完全独立于彼此的每个熵片段进行熵编码导致的上述损失，熵编码处理82经控制使得当前部分，例如当前熵片段的当前编码单元基于如使用当前熵片段的先前编码部分(即，就图2而言的至当前编码单元50的左边的当前熵片段的部分)调试的当前熵片段的各概率估计，以及如用于在其相邻部分，即相邻编码单元的按熵片段16的顺序在前的空间相邻的熵片段进行熵编码的概率估计来进行熵编码。

[0078] 为了描述更清楚地描述前述依赖性，现在参照图4。图4按熵片段顺序16用参考标号90示出了第n-1、第n及第n+1熵片段。每个熵片段90都包含描述样本阵列10的相应熵片段90与之相关联的部分的语法元素序列。沿熵编码路径14，将熵片段19分割为区段92序列，其中每一个都对应于熵片段90与之有关的部分12的编码单元50的相应编码单元。

[0079] 如上所述，用于对熵片段90进行熵编码的概率估计在连续阶段86期间沿熵编码路径14不断更新，使得概率估计越来越好地适应相应熵片段90的实际符号统计，即，概率估计与相应熵片段相关联。当不断更新在连续阶段86期间用于对熵片段90进行熵编码的概率估计94时，在图4中，下面只进一步示出并提及了区段92的开始位置和结束位置出现的概率估计94的状态。具体地，在96中示出了在对第一区段92进行熵编码之前如开始相84期间初始化的状态，在98中示出了在编码第一区段之后显示自身的状态，在100中指示在编码头两个区段之后显示自身的状态。同样针对第n-1熵片段，下一个熵片段，即n+1熵片段，按熵片段顺序16在图4中示出了相同元素。

[0080] 现在，为了实现上面所列的依赖性，根据在编码先前熵片段n-1期间显示出来的概率估计94的任何中间状态来设置对第n熵片段90进行熵编码的初始状态96。“中间状态”应指概率估计94的任何状态，不包括初始状态96和在对完整的熵片段n-1进行熵编码之后显示出来的最终状态。通过这样做，沿熵片段顺序16对熵片段90序列进行熵编码可以被并行化，且并行化程度通过在用于初始化对下一熵片段进行熵编码的概率估计94的状态之前的区段92数(即，a)，和在该级之后的区段92数(即，b)之比来确定。具体地，在图4中，a示例性地设置为在初始化期间相等，即调试状态100以使得将当前熵片段的状态96设置为等于先前熵片段的状态100，用箭头104示出。

[0081] 采用该措施，按熵编码路径顺序14对符合状态100的任何区段92进行熵编码可取决于如在连续阶段86期间基于相同熵片段的先前区段调试的概率估计94和如用于对先前熵片段90的第三区段92进行熵编码的概率估计。

[0082] 因此，对熵片段90进行熵编码在流水线调度过程中可并行执行。强加于时间调度的仅有的限制是只在完成对先前熵片段的第a区段92的熵编码之后才可以开始某些熵片段的熵编码。按熵片段顺序16直接彼此跟随的熵片段90不会受到与连续阶段86期间的熵编码程序的时间对准有关的任何其他限制的影响。

[0083] 然而，根据另一实施例，另外地或可选地可使用更强的耦合。具体地，如在图4中用代表性箭头106所示，连续阶段86期间的概率估计调试导致与某个区段92对应的编码单元的数据根据各区段92开始时的状态改变概率估计94直至该区段92结束，由此改善如上所示的实际符号统计的逼近(approximation)。也就是说，只根据熵片段n-1的数据对熵片段n-1

执行调试106，并且这同样适用于熵片段n等的概率估计调试106。例如，可以执行上文针对箭头104阐述的初始化，在熵片段90之间没有任何进一步干扰的情况下执行概率估计调试106。然而，为了加快实际符号统计的概率估计逼近，当调试当前熵片段n的概率估计调试时，连续熵片段的概率估计调试106可以经耦合使得先前熵片段n-1的概率估计调试106也有影响或被纳入考虑。在图4中用从对第n-1熵片段90进行熵编码的空间相邻概率估计94的状态110指向对第n熵片段90进行熵编码的概率估计94的状态100的箭头108示出了这种情况。例如，可以在使用上面所列的状态96初始化时，在对先前熵片段的b个区段92进行熵编码之后显示出来的b个概率估计状态的任意一个下使用概率调试耦合108。更确切地说，在对当前熵片段的第一区段92进行熵编码之后显示出来的概率估计可以是由常见概率调试106，以及将在对先前熵片段n-1的第(a+1)区段92进行熵编码期间由概率估计调试106产生的概率估计状态纳入考虑108造成的。例如，“纳入考虑”可以涉及某些平均化操作。下面将进一步概述实例。换句话说，在其熵编码区段92开始时，平均化对第n熵片段90进行熵编码的概率估计94的状态98可能是由对当前熵片段n进行熵编码的如使用调试106适应的概率估计94的先前状态96，与在对根据概率调试106修改的先前熵片段n-1的第a+1区段92进行熵编码之前的状态造成的。类似地，状态100可能是由平均化在对当前熵片段n进行熵编码期间执行的调试106的结果以及在对先前熵片段n-1的第a+2区段92进行熵编码期间的概率调试的结果造成的。

[0084] 更具体地，令：

[0085] $p(n)\rightarrow\{i,j\}$，其中i，j表示任意编码单元的位置(其中(0,0)位于左上位置，(I,J)位于右下位置)，$i\in\{1\cdots I\}$且$j\in\{1\cdots J\}$，I是列数，J是行数，p()限定路径编码器66，

[0086] $P_{\{i,j\}}$是用于对编码单元{i，j}进行熵编码的概率估计；并且

[0087] $T(P_{\{i,j\}})$是基于编码单元{i，j}的$P_{\{i,j\}}$的概率调试106的结果。

[0088] 然后，连续熵片段90的概率估计106可以组合为根据$P_{p(n+1)}=T(P_{p(n)})$用

[0089] $$P_{p(n+1)}=\text{average}(T(P_{p(n)}),T(P_{\{i,j\}_1}),\ldots,T(P_{\{i,j\}_N}))$$

[0090] 来取代常见的熵片段内部调试，

[0091] 其中，N可以是1或大于1，$\{i,j\}_{1\cdots N}$分别选自任意先前的熵片段90(按熵片段顺序16)或在其范围内。函数“average”可以是加权和、中值函数等中的一个。$p(n)=\{i,j\}$是当前编码单元，p(n+1)分别遵循编码顺序14和66。在所提出的实施例中，$p(n+1)=\{i+1,j\}$。

[0092] 优选地，满足$\{i,j\}_{1\cdots N}$，针对每个$k\in\{1\cdots N\}$，$\{i,j\}_{1\cdots N}=\{i_k,j_k\}$且$i_k<i+3$且$j_k<j$，其中$p(n)=\{i,j\}$是当前编码单元(即，当前熵片段的第二或之后编码单元中的任意一个)，即，不会超出波阵面。

[0093] 在后一种选择中，在对熵片段90进行并行熵编码时对时间调度进行熵编码彼此更紧密地耦合。也就是说，在连续阶段86期间，当前熵片段行中的下一区段92只可以在完成先前熵片段的对应区段时才开始，其是进一步按熵编码路径顺序14的排序位置。

[0094] 换句话说，上面的讨论示出了示例，其中，解码器40，具体为熵编码阶段44，配置为在沿相应编码路径14解码与对应于第n熵片段的部分12的第一编码单元/第一区块50对应的第一区段92之前，针对熵片段90比如第n熵片段，对概率估计94执行初始化，其中概率估计在与沿相应编码路径14按熵片段顺序16在先的熵片段对应的部分12的第二编码单元/区块50进行熵解码之后显示出来。另外地或可选地，解码器40，特别是熵编码阶段44，可以配

置为针对每个熵片段90执行熵解码和概率估计调试，使得在基于当前熵片段90的各概率估计94对当前熵片段的当前部分/区块/编码单元50进行熵解码之后，根据当前熵片段的当前部分和在对空间相邻的先前熵片段，比如至当前熵片段的当前部分/区块/编码单元的右边的第二列中的上述相邻行中的部分的相邻部分/区块/编码单元50执行熵解码的过程中显示出来的概率估计来调试当前熵片段的各概率估计94。

[0095] 如从上面的讨论可以看出，必须单独针对每个熵片段50调试/管理概率估计。这可以通过依次处理熵片段并将如针对图4示例性示出并提及的概率估计状态，即96、98、100、110及102存储在各概率估计存储装置120中来进行(参见图1)。可选地，可以设置一个以上熵编码阶段44以便并行解码熵片段。在图5中示出了这种情况，其中示出了熵编码阶段44的多个例示，每个都分别与熵片段的各个熵片段和样本阵列10的对应部分12相关联。图5还通过使用各熵解码阶段130的并行例示示出了解码处理机器可能实现。这些熵解码阶段130的每一个都通过经由数据流20传递的熵片段的各个熵片段来提供。

[0096] 图5示出了熵编码阶段44和相应解码级130不完全独立于彼此并行操作。相反，所存储的概率估计状态比如存储在存储装置120中的状态，从与各熵片段对应的一个级传递至与遵循熵片段顺序16的熵片段有关的另一级。

[0097] 图5出于说明目的还示出了遍历编码单元50的可能细分的可能遍历顺序，比如在一个编码单元50内的预测区块52之间限定的遍历顺序140。对于所有这些预测区块52，对应语法元素包含在各区段92内并与这些语法元素的值对应，在遍历路径140期间调试概率估计94，在遍历编码单元50期间的调试限定上述“T”。在根据H.264和HEVC的CABAC中，“T”基于表，即通过“表移(table walk)”来执行，该“表移(table walk)”限定从当前上下文的概率估计的当前状态至取决于与此上下文相关联的当前符号值的下一状态的转换。

[0098] 在继续进行示出了与图1的编码器实施例对应的解码器实施例的图6之前，应注意的是，预编码器42的预测特性仅充当说明性实施例。根据替代实施例，当熵编码阶段44所操作的语法元素例如是样本阵列10的原始样本值的情况下，预编码器42可以被省略。甚至可选地，预编码器42可以配置为比如以JPEG格式形成样本阵列10的子带分解(subband decompositon)。上面已经提及了上下文选择器72的任选特征。这同样适用于初始化器74。这同样可以以不同方式进行仪表化。

[0099] 图6示出了与图1的编码器40对应的解码器200。从图6可见，解码器200的构造基本上体现了编码器40的构造。也就是说，解码器200包括用于接收数据流20的输入端202，随后是熵解码阶段204和构造器206的级联。熵解码阶段204对传递至数据流202内的熵片段进行熵解码并又将解码符号s_i和语法元素分别转发至构造器206，该构造器206又通过相应请求208来从熵解码阶段204请求语法元素。换句话说，构造器206还要承担在编码器内解析由预编码器42产生的语法元素流的责任。因此，构造器206依次从熵解码阶段204请求语法元素。熵解码阶段204的构造基本上与熵编码阶段44相同。因此，重新使用熵解码阶段204的内部区块的相同参考标号。符号化器70(如果存在)将语法元素请求变为符号请求，熵解码核80利用各符号值s_i应答，符号化器70将形成有效符号字的接收符合序列映射到语法元素并将其转发至构造器206。构造器206根据从熵解码阶段24接收的语法元素流比如如上所述的那些，使用预测解码等来重建样本阵列10。更确切地说，构造器206还要使用编码顺序66并以编码单元的方式执行解码，同时执行预测60，62及64。任选利用从语法元素流的语法元素获

得的预测残差来组合(比如累计)语法元素或样本值的一个或多个预测。熵解码核80,如与熵编码核80一样,可以遵循算术解码概念或概率区间划分熵解码概念。就算术解码而言,熵解码核80可以以指向该部分区间的部分区间宽度值和诸如偏移值等值的形式来不断更新内部状态。使用入站数据流来进行更新。当前部分区间使用通过上下文选择器72和概率估计管理器76为每个符号s_i提供的概率估计p_i类似地被细分为熵编码核80。调试器78使用解码符号值s_i来执行概率估计调试以便通过上下文选择器72更新与符号s_i相关联的上下文c_i的概率估计值p_i。在相同示例中通过与编码侧相同的方式来进行初始化器74进行的初始化。

[0100] 图6的解码器以非常类似于上文中针对图3描述的编码器的功能的方式进行操作。在步骤230中,将样本阵列10细分为熵片段。例如参见图8。图8示出了到达输入端202的数据流20以及其中包含的具体熵片段90。在步骤230中,这些熵片段90的每一个都与各熵片段90与之相关联的部分12相关联,使得可以执行基于进行中的各熵片段的上述概率估计初始化和概率估计调试。样本阵列细分,或更确切地说,熵片段90与相应部分12的相关性,可以由构造器206执行。相关性可以采用不同措施实现,比如未进行熵编码的部分中的数据流20内包含的边信息,或按照惯例。

[0101] 熵片段90然后在熵解码处理232中以反映编码处理82的方式进行熵解码,即同时针对每个熵片段90执行起始阶段234和连续阶段236,概率估计初始化和调试以相同的方式且在相同示例中进行,如与在编码程序中一样。

[0102] 在解码侧可能存在上文中针对编码的所述的相同并行化。图5中示出的熵解码阶段例示130可以分别被实施,如针对图6中的熵解码阶段204所示。可以使用概率估计存储装置240以便存储用于熵解码阶段130的概率估计的状态,该熵解码阶段负责按熵编码顺序16对后继熵片段进行熵解码。

[0103] 在描述了本申请的实施例之后,在下文中再次使用不同措辞来描述目前为止讨论的概念。在下文中,将描述本申请的几个进一步方面。在下文中,前述编码单元50称为LCU(最大编码单元),由此使措辞适应即将出现的HEVC标准。

[0104] 首先,再次参照图9简要讨论上文讨论的概率调试106。

[0105] 当前LCU使用在编码先前的左边LCU之后可用的概率估计比如CABAC概率。例如,图9的LCU,用x表示,被视为使用概率估计p1进行熵编码,该概率估计如直到将左侧LCU熵编码至LCUx的左边结束时才进行的调试。然而,如果该信息不但从左边一个开始使用,而且还从已经进行处理且可用的一个或多个其他LCU开始使用,则可以达到更好的概率调试结果。

[0106] 如上所述,在对熵片段进行熵解码和熵编码的过程中,在来自其他LCU的已有概率(概率估计)的帮助下,在编码或解码每个LCU之前计算新的概率估计。更确切地说,概率调试不但根据当前熵片段的任意先前LCU执行,而且还根据在熵片段顺序之前的熵片段的LCU执行。再次在图9中表示该方法。因此可以计算在图9中用X表示的当前LCU的开始概率:

[0107] $$p_{new}=a_1\cdot p_1+a_2\cdot p_2+a_3\cdot p_3+a_4\cdot p_4+a_5\cdot p_5+a_{(5+1)}\cdot p_{(5+1)}\cdots+a_k\cdot p_k \quad (1)$$

[0108] 其中$a_1,\cdots a_k$是LCU的加权因子。

[0109] 试验表明哪些概率加权提供最佳结果。在该实验中,只使用了相邻LCU。研究表明使用来自左边LCU的75%和来自右上方LCU的25%的加权。在图10到图13中,呈现了结果。以“prob.Adapt.(概率调试)”为标题的示图使用上述概率调试。

[0110] 然而，在概率估计调试中，不仅可以使用相邻区块。每个最近的LCU具有自己的特别的邻近部分，其概率最佳化用途可能相当重要。换句话说，不仅可以利用来自上部最近的行的LCU。在图9中，人们可以看到概率估计的推导首先从邻近部分中选择并且可以从每个下一个上行的每个右上方LCU中选择，参见p_5和p_k。

[0111] 应该承认，上述概率重新计算或概率估计调试引起了一些复杂性。例如通过三个步骤对概率估计进行新计算：首先，必须从每个上下文状态获取每个候选者的概率估计。这种情况通过分别存储在存储装置120和240中，或通过操纵并行熵片段n/解码处理来进行，使得这些状态并发可用。其次，通过使用方程(1)，创建最佳化概率(p_{new})。也就是说，例如可以使用某些平均化以便组合不同熵片段的调试概率估计。在最后一个步骤中，从p_{new}中转换新的上下文状态并替换旧的上下文状态。也就是说，概率估计管理器76采用由此获得的新概率估计。每个语法元素的性能，尤其使用乘法运算，可以增加巨大的复杂性。减少损失的唯一途径是试图避免这三个步骤。如果确定候选者的数量及其加权，则可以逼近每种情况的预先计算好的表。因此，只需要在候选者的索引(上下文状态)的帮助下简单访问制表日期。

[0112] 可认为该技术可以为使用或不使用熵片段的应用提供良好的结果。对于第一应用每帧只使用一个片段，由此在没有任何其他变化的情况下最佳化概率调试。就熵片段而言，概率调试发生在独立于其他片段的每个片段内。这允许可以快速了解当前LCU的概率。

[0113] 在上述描述中，还呈现了上行的第二LCU的使用，即，概率估计初始化的第二LCU的使用。如果完成了上文提及的比特流(熵片段)的某些条件，则编码和解码并行化是可能的。必须破坏LCU之间的CABAC概率依赖性。通过波阵面并行处理对使每个行的第一LCU独立于前一行的最后LCU来说是很重要的。例如，如果CABAC概率在LCU的每行开始时被重新初始化，则这可以被实现。然而，该方法不是最佳的，原因是每次重新初始化丢失所实现的适应图像的特异性的CABAC概率。如果每行的第一LCU的CABAC概率的初始化以在前一行的第二LCU之后获得的概率而发生，则可以减少该损失。

[0114] 如上文所描述的，通过耦合空间相邻熵片段的概率调试可实现概率调试速度的增益。具体地，换句话说，上述讨论还期望解码器比如图6的解码器来从熵编码数据流重建样本阵列(10)，该解码器配置为对熵编码器数据流内的多个熵片段进行熵解码(由熵解码阶段执行)以便分别重建与熵片段相关联的样本阵列的不同部分(12)，同时针对每个熵片段沿各熵编码路径(14)使用各概率估计来执行熵解码，沿各熵编码路径使用各熵片段的先前解码部分来调试各概率估计(由调试器78执行)，使用熵片段顺序(16)开始对多个熵片段依次进行熵解码，并在对预定熵片段进行熵解码的过程中，基于如使用预定熵片段(例如包括p1)的先前解码部分调试的预定熵片段的各概率估计，以及如用于在空间相邻熵片段的相邻部分(比如p4)的进行对按熵片段顺序在前的空间相邻的熵片段(例如包括X的片段)的熵编码中所使用的概率估计来对预定熵片段的当前部分(x)执行熵解码。

[0115] 不同部分可以是样本阵列的区块的行(例如，LCU或宏块)。后者可以是视频的图片。熵编码路径可以按行延伸。熵编码是上下文自适应的，因此概率调试也可以是上下文自适应的。熵片段顺序一般经选择使得，沿熵片段顺序，不同部分沿相对于熵片段的熵编码路径(14)倾斜的方向(16)相互跟随，其又基本上彼此并行延伸。采用该措施，熵片段的当前解码部分(比如图中的p1，p4，p5+1及pk)的“波阵面”一般可以沿形成小于这些部分的序列方

向的熵片段路径的角的行排列。波阵面可能必须具有1y/2x区块位置的斜率以便左上参考总是支持所有线程(thread)并行处理片段。

[0116] 解码器可以配置为针对每个熵片段，沿各熵编码路径以相应熵片段的样本阵列部分的多个部分为单位执行熵解码，使得熵片段分别由相同数量的部分组成，沿熵片段路径的部分的部分序列沿横向于熵片段路径的方向彼此对齐。预定熵片段部分的当前部分属于由此产生的部分网格(例如LCU或宏块)。在针对每个熵片段沿各熵编码路径执行熵解码的过程中，解码器可以保留熵片段顺序中直接连续的熵片段之间的起始时间之间的偏移/移位，使得熵片段的当前解码部分的波阵面针对熵片段路径和熵片段顺序方向形成比如直到0.5x区块位置斜率的行的对角线。偏移/移位可以与所有成对的直接连续的熵片段的两个部分对应。可选地，解码器可以只防止直接连续的(关于其样本阵列部分12，则是直接相邻的)熵片段的当前解码部分的距离变成小于两个部分。参见上图：一旦解码该部分/区块p4，其右边的部分/区块就根据路径顺序16解码，并发地，如果存在，则解码X或解码在其之前的部分/区块中的任意一个。采用该措施，解码器可以使用如已经基于部分/区块p4的内容调试的概率估计，即，空间相邻部分内的部分与预定熵片段的部分12的沿路径顺序16继当前部分X之后的部分对齐，以便确定概率估计用于解码X。就在解码两个部分的熵片段的过程中的常量偏移而言，解码器能够使用如已经基于部分/区块p4的内容调试的概率估计，以便并发地对空间相邻熵片段的后继部分(即，至p4的右边的部分)进行熵解码。

[0117] 如上所述，可以使用已经调试的概率估计的加权和以便确定将被用于解码X的概率估计。

[0118] 同样如上所述，熵片段顺序还可以越过帧边界。

[0119] 应注意的是，可以针对当前/预定熵片段的每个部分执行上述的来自先前熵片段的概率调试，先前熵片段中的这些相邻部分可用于此。也就是说，对沿路径方向16的第一部分也是如此，并且对于这第一部分/区块(图中每个熵片段的最左边)，调试等同于上述初始化。

[0120] 为了更好调试，同样在这种情况下，可以将上述两种方法组合在一起。在图10到图13中示出了1线程和4线程的处理结果，即并行使用的处理单元(2LCU+Prob.Adap或2LCU)。

[0121] 为了更好理解上述实施例，尤其是下面描述的进一步实施例的上下文，具体是LCU的用途，可以首先查看H.264/AVC的结构。

[0122] H.264/AVC中的编码视频序列由NAL单位流中收集的一系列访问单元组成，并且只使用一个序列参数集。可以独立解码每个视频序列。编码序列由编码图片序列组成。编码帧可以是整个帧或单个字段。将每个图片划分为固定尺寸的宏块(在HEVC中：LCU)。几个宏块或LCU可以一起并入一个片段。图片因此是一个或多个片段的集合。该数据分离的目的是允许对用片段表示的图片区域中的样本进行独立解码，而不使用来自其他片段的数据。

[0123] 通常也称为“熵片段”的技术是将传统片段分成额外在子片段。具体地，指的是将单个片段的熵编码数据切片。一片一片地排列熵片段可以具有不同种类。最简单的一种是使用帧中的每行LCU/宏块作为一个熵片段。可选地，列或独立区域可以被用作熵片段，其甚至可以被中断并彼此切换，例如图14中的片段1。

[0124] 熵片段概念的明确目标是可以使用并行CPU/GPU和多核架构，以便改进解码处理的时间，即以便加速处理。可以将当前片段划分为可以在不参考其他片段数据的情况下进

行解析和重建的分割。虽然两个优点可以利用熵片段方法来实现，但是也出现出了一些损失。

[0125] 首先，主要目的是创建适用于并行编码和解码处理的比特流。必须考虑LCU只有在相邻LCU(左边，上部，右上)已经可作为编码版本时才可以进行编码，以便使用空间和运动信息来预测。为了通过切片来启用并行化，必须实现处理片段之间的移位(例如2个LCU的移位，对波阵面方法来说是典型的)。由于CABAC概率调试，LCU使用可从先前编码LCU得到的概率。至于光栅扫描顺序，对图片进行切片产生的问题是对并行化的抑制，因为每行的第一LCU都取决于前一行的最后一个LCU。影响在于必须破坏片段之间的CABAC概率依赖性，以便同时启动几个片段。其中一个办法是重新初始化普通CABAC，然而，将丢失所有采用的数据。结果，位率可能会增加。

[0126] 其次，每个片段都要生成可以串联放入主流中的自身子比特流。但是必须传递解码器专用信息，使得可以正确识别主流中的这些片段和位置。支持信号传送的两个场景。位置信息可以存储在图片报头中(片段长度信息)或存储在每个片段报头中(作为起始码的点)。在每个熵片段和位置信息结束时字节对齐会增加损失。

[0127] 为了减少针对熵片段进行信号传送引起的损失，必须使用信号传送的良好编码技术。如果起始码用于每个片段，即为比特流添加太多额外的字节(例如每个片段最少4个字节)，则在帧中将引起熵片段信号传送的明显损失。当然，使用起始码插入熵片段90的优点在于低延迟场景，其中编码器应该能够即刻输出熵片段。在这些情况下，不可能存在进入点的预先信号传送。

[0128] 然而，在严格较少的低延迟场景下，存储片段长度(偏移)的机会看起来更适合。一种已知的编码此信息的方法是可变长码(VLC)或指数哥伦布编码。VLC的主要特征是在真信息之前添加空信息(零)。在这些零的帮助下，可以确定存储关于偏移长度的信息的代码。提出另一种实现这种情况的技术，在图15中示出其方案，其中X＝EntropySliceSize是熵片段中包含的字节数量。接下来的每一个X(偏移)都被定义为先前已经编码并信号传送的熵片段偏移与当前偏移之间的尺寸差。该概念的主要特征是根据尺寸X形成差异，由此可以减少数据量，总共达到3位，这允许通过解码器提取关于每个熵片段的尺寸的合适信息。与VLC相比较，可以获取熵片段报头中节省的位数量。

[0129] 也就是说，根据图15的方面，例如，提供了对信息信号进行熵编码的概念，不管并行处理能力如何，与目前为止可用的概念相比，其实现压缩率更高。根据这方面，由样本阵列编码成的熵编码数据流部20包括由样本阵列的不同部分12熵编码成的熵片段比如图8中的90，以及在图9中用虚线300所示的报头，包括在熵编码数据流中在熵解码域中测得的揭示熵片段90的起始位置302的信息，该信息针对预定熵片段包括先前熵片段n-1的起始位置与预定熵片段n的起始位置之间的差，该差包括在数据流中作为VLC位序列。

[0130] 具体地，VLC位序列可以具有可变长度y的前缀，表示差x落入z个区间$[0,2^a-1]$，$[2^a,2^b+2^a-1]$，$[2^b+2^a,2^c+2^b+2^a-1]$等等序列中的第y个，以及长度a，b，c……序列中的第y个的PCM编码后缀。如果a，b，c……被选择为是2的幂，如果添加对应的y，即使得a+1，b+2，c+3等都是2的幂，则可以保持字节对齐。数量z不限于三个，如图5中示例性选择的。

[0131] 编码器，比如图1中的编码器，被适当设计为将连续起始位置的差变为VLC位序列，即通过首先确定前缀，即差位于哪个子区间(第y个)，然后将后缀加(suffix plus)设置为

等于起始位置差减去0，2^a，2^b+2^a等等中的第y个。解码器，比如图6中的解码器，被适当设计为从VLC位序列中得出当前熵片段n的起始位置，即通过首先检查前缀以获得y，然后将差设置为前缀的值加上0、2^a、2^b+2^a等等中的第y个，然后将差与先前熵片段n-1的起始点相加。

[0132] 进一步优点可以通过分割熵片段来实现，即用于低延迟传输并解码加速。

[0133] 在视频流中，启用更高的分辨率(Full-HD，QUAD-HD等)会产生必须传输的更大量的数据。对于时间灵敏的场景来说，所谓的低延迟用例(<145ms)，传输时间成为关键因素。考虑用于视频会议应用的ADSL的上行传输链路。这里，所谓的流的随机存取点，通常这些都指I帧，将是在传输期间导致瓶颈的诱因之一。

[0134] 为了解决此问题并最小化传输延迟和解码时间，即，端到端延迟，可以应用用于并行传输和处理的交错熵片段方案的信息技术。

[0135] HEVC允许在解码器侧进行所谓的波阵面。这将利用熵片段来实现。在正常情况下，一次传递整个片段的数据。一旦编码数据到达波阵面解码器引擎，就开始解码处理。为了缩短解码器可以开始并结束一个帧的时间，根据本实施例采用使用交错方法将熵片段分割为小组块的分割。因此，编码器可以比在正常情况下更早地将与特定熵片段对应的数据传递至传输层。这随后会导致传输更快且在客户端更早开始并行解码处理。

[0136] 创建片段组块可以进一步通过以保持所有依赖性的进一步片段(依赖性片段)细分熵片段来实现。如果在每个最大编码单元(LCU)/编码树单元(CTU)中这样做，这些组块可以通过额外使用系统层技术来进行交错，系统层技术允许以交错的形式传输组块并经由额外信令恢复或至少提供组块的原始解码顺序的知识。此信令可以是解码顺序号(DON)，比如H.264/AVC(RFC 3984)的IETF RTP有效载荷格式中所定义的。另一系统方法可以为如在MEPG-2系统中，通过向其中每一个分配不同PID，进一步进行多路复用，并通过在传输信道上对它们进行交错，将波阵面子流的组块应用于不同传输流。

[0137] 由于使用帧间参考的可用性，如果例如已经以波阵面的方式，基于对解码以下帧的熵片段的所需信息的了解，已解码了随后帧的片段或熵片段，则该方法还可以用在帧边界上。按解码顺序接着出现的帧的已经可解码数据可以根据流中的指示数据部分对先前帧的依赖性的最大容许/信号传送运动矢量长度或额外信息，或指示在序列固定位置比如参数集中的信号传送的位置的固定参考方案得出。下面将进一步概述这种情况。

[0138] 图片可以利用每个最大编码单元(LCU)行一个熵片段来编码，如图2和图9所示。在解码器侧有利于使用波阵面技术。在编码处理期间，可以将每个片段的比特流划分为恒定尺寸的区段。然后得到的区段被交错并且可以传递至传输装置。恒定尺寸的区段在比特流结束时由于其可变长度而可能会造成问题。

[0139] 可能存在两种普通的解决方案。第一种是生成一字节区段(片段的比特流表征通常字节对齐)并控制每个解码器引擎消耗的字节，即，解码器揭示熵片段的完成。

[0140] 第二种是在片段结束时使用最终化代码。在后一种情况下，可能存在可变长度区段，但这也会产生更大量的数据。另一种方法是信号传送熵片段长度。下面将描述此选择的一种方式。

[0141] 区段的尺寸和交错模式可以在一个SEI消息或在SPS中用信号传送。

[0142] 图16中示出了传输方案。

[0143] 由此，根据图16的方面，例如，提供了对样本阵列进行熵编码的概念，与目前为止

可用的概念相比，其实现了较低延迟。根据这方面，信息信号编码成的编码数据流20包括比如熵片段90或信息信号的不同部分12(预测地和/或熵)编码成的完全独立可编码片段(左侧所示)的片段，其中将片段90细分为以交错方式排列在编码数据流20内的组块(阴影框310)，交错用括号312表示。

[0144] 如上文所表示的且针对其他方面所述的，片段可以是熵片段90，熵片段又是帧的片段的真子集，因此，编码数据流可以是熵编码数据流20。

[0145] 片段90的交错312实现低延迟，原因是负责解码任意片段90的解码器不需要等待其他解码器的先前片段所消耗的持续时间(根据片段顺序16)。相反，一旦第一组块310可用并潜在的片间依赖性被解析，所有可用解码器就能够开始解码相关联的片段，参见波阵面方法。

[0146] 可以使用熵片段之间独立设置的概率估计，或使用如上所述的概率估计值的熵片段间调试来将不同部分12熵编码成熵片段90，例如，即沿各熵编码路径14将不同部分12编码成熵片段90，并且下级熵片段使用尤其从如前所述的对应部分的空间相邻部分的更高等级的熵片段中使用的概率估计值得出的概率估计值在其中编码对应部分12。

[0147] 熵编码数据流20可以另外包括如在图16中示为选项的报头。报头300可以与序列30的帧(样本阵列)相关联，报头300包括揭示熵片段的长度的信息。关于熵片段90的长度的信息可以在如上所述的报头300中使用VLC代码进行编码。使用关于熵片段的长度的知识，在解码侧，可以识别与每个熵片段90及其长度相关联的最后组块。然而，还可以使用起始码或其指示方案。片段的起始位置还可以通过了解片段中断的解码处理来识别。因此，可能仅仅依靠解码器的指示，但这需要在解码器之间传送信号，在某些情况下，“较早的”熵片段中断迟于流中的“后一个(later)”片段。针对特定情况可能需要流中的“自适应”信令，其可以基于起始码。

[0148] 例如，在如图16所示的报头300之后可以排列熵片段90。

[0149] 至少对于所关注的沿组块在熵编码数据流20内排列的顺序从组块的第一个开始的组块310序列的开始连续部分，组块310的长度可以相等。后继组块的长度可以改变。后继组块可以等于序列的开始连续部分的长度或小于其长度。后继组块的长度可以从报头300内揭示熵片段90的长度或起始位置的前述信息得出。组块310可以根据熵片段之间限定的顺序以循环方式排列在熵编码数据流20内。在当前和后继周期中可以跳过其组块全部在先前周期内的熵片段，。

[0150] 其他信息信号，随后样本阵列序列比如视频信号经由数据流20传递。不同部分12由此不必是预定样本阵列的部分比如图片/帧。

[0151] 如上所述，熵片段90可以具有经由使用熵片段间预测和/或帧间预测进行的预测编码，而将熵片段间预测和/或帧间预测的预测残差的熵编码编码其中的样本阵列10的不同部分12。也就是说，如上所述，不同部分可以是帧10或多个帧30的空间上有区别的部分。如果例如已经以波阵面的方式，基于对解码以下帧的熵片段的所需信息由于可用的帧间参考的了解解码了以下帧的片段或熵片段，则后一种情况适用。按解码顺序接着出现的帧的已经可解码数据可以根据指示数据部分对先前帧的依赖性的流中的最大容许/信号传送运动矢量长度或额外信息得出，并且熵片段间预测可以涉及帧内预测，同时帧间预测可以涉及运动补偿预测。下面将概述实例。

[0152] 熵片段之间的概率估计设置的前述独立性可能与概率调试以及上下文建模有关。也就是说，在熵片段内选择的上下文可以独立于其他熵片段进行选择，上下文的概率估计也可以独立于任何其他熵片段进行初始化和调试。

[0153] 对应熵解码器的构造可以如下。

[0154] 熵解码器的配置可以如图6所示并且可以另外包括配置为对组块310进行解交错且在图16中用314标识的解交错器，该熵解码器配置为将包括熵片段90的熵编码数据流20解码为被熵编码是帧的不同部分，其中将熵片段细分为组块310，所示组块以交错方式排列在熵编码数据流20内。

[0155] 具体地，如图5所示，熵解码器可以包括多个熵解码器130比如在不同处理内核上运行的线程，其中，解交错器可以配置为针对每个熵片段将其组块310转发至与各熵片段相关联的熵解码器44。

[0156] 换句话说，可以将熵片段细分为组块，还可以对所述组块进行交错，解码器可以包括对组块进行解交错的解交错器并且即使在整体接收任意熵片段之前也可以开始沿路径16在熵片段上并行操作。应该牢记，组块的长度优选在熵编码域中而不是在语法域中进行测量以便对应于图片中的数个空间部分/区块，但是后一选项也是可用的。

[0157] 接下来，将描述时间依赖性的可能用途。除了目前为止所述的概率估计增强实施例之外或作为其替代，同样可以使用。

[0158] 波阵面处理模式正如现在所描述的一样可以扩展至熵编码，其中每个LCU的新概率计算同样使用帧之间的时间依赖性。

[0159] 众所周知，概率将在每个帧开始时(第一LCU)被重新初始化。由此已经在先前帧中获取的概率会丢失。为了降低编码效率的丢失，人们可以将图片的最终状态(参见320)，或就熵片段用途而言，概率的片段的最终状态(参见322)分别从参考帧324传递至当前帧10的第一LCU 50或熵片段12(图17)。参考帧中的对应片段的这个数据不但可以在结束位置进行推导，而且还在参考片段中的先前位置进行推导，原因是并行波阵面处理也可以越过帧边界，即，在编码帧片段的同时，可能尚未完成先前帧的片段的编码处理。因此，信令可以用于指示参考位置，或可以用方案来指示。

[0160] 使用上述符号，人们由此可以分别针对在开始阶段84和234中进行初始化，设置$P_{\{i,j\}}$，{i,j}表示等于或至少取决于任何$T(P_{\{i,j\}'})$的当前样本阵列的第k个熵片段中的第一CU 50，{i,j}'表示先前(按可以等于表示呈现顺序的样本阵列编码顺序)样本阵列或几个$T(P_{\{i,j\}'})$'的组合内的CU。这种情况只针对k＝0或每个熵片段k∈{1…K}进行，其中K表示当前帧中的熵片段数量。除了上述空间初始化之外或替代其还可以进行时间初始化。也就是说，可以将$P_{\{i,j\}}$设置为等于$T(P_{\{i,j\}'})$和$\mathbf{T}(P_{\{i,j\}_{spatial}})$的某些组合(比如sme平均值)，或几个$T(P_{\{i,j\}'})$和$\{i,j\}_{spatial}$的组合，其中{i,j}表示第k熵片段中的第一CU 50，{i,j}'表示先前(先前编解码的)样本阵列内的CU，$\{i,j\}_{spatial}$表示当前样本阵列的先前熵片段内的CU。只要关注{i,j}'的位置，可以将$P_{\{i,j\}}$设置为等于$T(P_{\{i,j\}'})$，其中{i,j}表示当前样本阵列的第k熵片段(按熵编码顺序14)中的第一CU 50(按熵编码顺序14)，{i,j}'表示先前样本阵列(按样本阵列编码顺序)中的第k熵片段(按熵片段顺序)内的最后CU(按熵编码顺序14)或先前样本阵列(按样本阵列编码顺序)中的最后熵片段(按熵片段顺序)内的最后CU。再者，时间初始化可以只针对样本阵列中的第一熵片段执行。

[0161] 利用概率调试方法来测试参考帧的最终状态的解析处理，在图10到图19中示出了其结果(时序(temporal)图)。

[0162] 使用来自其他帧的日期的另一机会是在并置LCU之间交换所获得的概率。主要思想基于下述看法，参考帧的属性与当前帧没有太大区别。为了加快了解沿帧中的LCU的概率，人们可以试图将每个LCU的最终状态传递至当前帧中的合适LCU。在图18上示出了该提案。

[0163] 借助参考帧，可以理解不同的机会。例如，最后编码的帧可以被用作参考帧。否则，只来自相同时序层的最后编码的帧可以适用于参考。

[0164] 而且，该方法可以与上文已经提出的方法合并，作为来自参考帧的最后(片段)信息的用途，概率调试和上行的第二LCU的用途。

[0165] 上面的空间调试处理可以修改为

[0166] $$P_{p(n+1)}= \text{average}(T(P_{p(n)}), T(P_{\{i,j\}_1}),\ldots,T(P_{\{i,j\}_N}), T(P_{\{i,j\}'_1}),\ldots,T(P_{\{i,j\}'_M}),)$$

[0167] 其中，N可以是1或大于1，$\{i,j\}_{1\cdots N}$分别选自当前样本阵列10及其相关联部分12中的任意先前的熵片段90(按熵片段顺序16)或在其范围内，M可以是1或大于1，$\{i,j\}'_{1\cdots M}$在先前样本阵列350的范围内。这可能是因为CU's 50$\{i,j\}'_{1\cdots M}$中的至少一个共同定位至p(n)。关于CU's 50$\{i,j\}_{1\cdots N}$的可能选择，参照上文描述。函数“average”可以是加权和、中值函数等中的一个。

[0168] 上述空间调试处理可以用$P_{p(n+1)}=\text{average}(T(P_{p(n)}), T(P_{\{i,j\}'_1}),\ldots,T(P_{\{i,j\}'_M}),)$替代，

[0169] 其中M可以是1或大于1，$\{i,j\}'_{1\cdots M}$在先前样本阵列的范围内。这可能是因为$\{i,j\}'_{1\cdots M}$中的至少一个共同定位至p(n)。关于$\{i,j\}_{1\cdots N}$的可能选择，参照上文描述。函数“average”可以是加权和、中值函数等中的一个。这可能是因为$\{i,j\}'_{1\cdots M}$中的至少一个共同定位至p(n)。

[0170] 可以采用利用从来自一个或甚至多个参考帧的其他区块获得的日期的方法作为并置信息的用途的特定扩展。

[0171] 前面提到的技术只使用从当前帧或参考帧中的直接邻近部分获得的信息。然而，这并不意味着在这种情况下获得的概率是最佳概率。相邻LCU，根据图片分割(残差)，不总是具有最佳概率模型。据推测，可以在区块的帮助下实现最佳结果，可以据此进行预测。因此，该适当的区块可以被用作当前LCU的参考。

[0172] 因此，在上述调试实例中，可以根据充当p(n)预测因子提供者的CU来选择$\{i,j\}_{1\cdots N}$和/或$\{i,j\}'_{1\cdots M}$。

[0173] 可以对每帧在熵片段或单一熵片段的情况下使用所呈现的时间概率调试/初始化方案。

[0174] 根据后一方面，概率调试速度的增益通过耦合时间相邻/有关的帧的概率调试来实现。所描述的是存在一种解码器比如图6的解码器，其中解码器配置为从熵编码器数据流重建样本阵列序列，并配置为对熵编码器数据流的当前帧进行熵解码以便重建样本阵列序列的当前样本阵列，沿熵编码路径并使用概率估计来执行熵解码并沿熵编码路径使用当前帧的先前解码部分来调试概率估计，其中，熵解码阶段配置为基于用于解码熵编码数据流

的先前解码帧的概率估计来初始化或确定当前帧的概率估计。

[0175] 也就是说，例如，当前帧的概率估计基于在完成解码熵编码数据流的先前解码帧之后产生的概率估计来初始化。因此，缓冲要求较低，原因是只有概率估计值的最终状态才需要缓冲直至当前帧解码开始为止。当然，这方面可以与图1到图9的方面组合，其中针对每个部分12的第一部分，不但是用于先前熵片段(如果可用)中的空间相邻部分的概率估计值，而且以加权方式，例如，可以使用先前帧中的(例如空间)对应熵片段的概率估计值的最终状态。参考帧中的对应片段的这个数据不但可以在结束位置进行推导，而且还在参考片段中的先前位置进行推导，原因是并行波阵面处理也可以越过帧边界，即，在编码帧片段的同时，可能尚未完成先前帧的片段的编码处理。因此，信令可以用于指示参考位置，或可以用方案来指示。

[0176] 进一步地，例如，用于编码先前解码帧的部分/区块的概率估计全部被缓冲，不仅是最终状态，解码器，在对预定熵片段进行熵解码的过程中(参照空间耦合概率推导的上文描述)，基于如使用预定熵片段的先前解码部分调试的预定熵片段的各概率估计(例如包括p1)，以及如用于对先前解码帧的熵片段的空间对应部分进行熵解码的概率估计来对预定熵片段的当前部分(X)执行熵解码，任选地，另外地使用如用于在空间相邻熵片段的相邻部分(比如p4)的熵片段(例如包括X的片段)之前按熵片段顺序对空间相邻部分进行熵解码的概率估计，如上所述。同样正如上面所描述的，部分之间的空间对应，以及先前解码帧中的当前帧的概率调试的合适的识别可以在当前部分/区块的运动信息，比如运动索引、运动矢量等的帮助下来限定。

[0177] 到目前为止，在波阵面处理期间延伸的波阵面主要被描述成倾斜延伸通过一个样本阵列10，其中，编码/解码可以一个样本阵列接一个样本阵列地执行。然而，这不是必须的。参照图19。图19示出了超出样本阵列序列范围的部分，其中，在其中定义了序列的样本阵列，并被描述成按样本阵列编码顺序380排列，该顺序可以与呈现时间顺序一致或不一致。图19示例性地示出了分别将样本阵列10细分为四个熵片段。已经编码/解码的熵片段用阴影线示出。四个编码/解码线程(编码/解码阶段)382当前在具有指数n的样本阵列的四个熵片段12上操作。然而，图19示出了保留的线程数5，并且在图19中该具有数量5的进一步线程382被操作用于编码/解码下一样本阵列，即，n+1，保证当前编码/解码帧n中的各参考部分已经可用于此的部分，即，已经由线程1到4中任意一个处理。这些部分例如在图1中的64处示出的预测中被引用。

[0178] 图19示例性地用虚线384示出，延伸通过样本阵列n+1的线，该样本阵列n+1共同定位至已经处理的、即样本阵列n的已经编码/解码的部分，即一方面是样本阵列n内的阴影线部分，与尚未处理的部分，即另一方面样本阵列的未加阴影线部分之间的界限的样本阵列n+1。利用双箭头，图19还示出了按列和按行方向测量的运动矢量的最大可能长度，即分别为y_{max}及x_{max}。因此，图19还用点划线386示出了线384的取代版本，即与线384间隔开最小可能距离的线386，使得距离不会沿列方向降至y_{max}以下，也不会沿行方向降至x_{max}以下。可以看出，在样本阵列n+1中存在编码单元50，样本阵列n中的任意参考部分针对其被保证发现完全包含在该样本阵列n的已经处理过的部分内，即，位于样本阵列n+1的二分之一处，样本阵列n+1相对于线386位于上游侧。因此，线程5能够已经用于解码/编码如图19所示的这些编码单元。可以看出，甚至是第六线程也可按样本阵列n+1的熵片段顺序16在第二熵片段上操

作。由此，波阵面不但以空间方式，而且还以时间方式延伸穿过样本阵列序列30跨越的时空空间。

[0179] 请注意，刚提及的波阵面方面还结合熵片段界限上的上述概率估计耦合工作。

[0180] 此外，参照上述组块方面，还应注意的是，将熵片段细分为较小片，即组块，不受要在熵编码域中、即在熵编码域中执行的限制。想一想上述讨论：如上所述的熵片段具有降低编码效率损失的优点，尽管实现了由于概率估计偏离相同的或先前编码/解码帧的先前编码/解码熵片段造成的波阵面处理，即，基于这些先前熵片段的概率估计来初始化和/或调试概率估计。假设就波阵面处理而言，这些熵片段的每一个都被熵编码/解码一个线程。也就是说，当细分熵片段时，没有必要使组块并行编码/解码。相反，编码器只应具有在完成熵编码之前输出熵片段的比特流的子部分的机会，解码器应具有在接收相同熵片段的剩余组块之前在这些子部分，即组块上操作的机会。此外，应在接收侧启用交错。然而，为了启用后一种解交错，没有必要在熵编码域中执行细分。具体地，在没有严重的编码效率损失的情况下，仅仅通过间歇重置熵编码/解码核的概率区间的内部状态，即分别为概率区间宽度值、偏移值来执行上文呈现的将熵片段细分为更小的组块。然而，不重置概率估计。相反，分别从熵片段的开始端至末端不断进行更新/调试。采用该措施，可以将熵片段细分为单独组块，细分在语法元素域中执行，而不是在压缩比特流域中执行。细分可能符合如下所列的空间细分以便缓解组块接口至解码器的信号传送。每个组块都设有揭示其在样本阵列中的起始位置的自身组块报头，例如针对编码顺序14相对于各熵片段的起始位置和熵片段索引，或相对于样本阵列10的突出位置比如左上角来测量。

[0181] 为了更清楚地描述根据后一实施例的将熵片段细分为组块，参照图20。图20仅出于说明性目的示出了样本阵列10被细分为四个熵片段。样本阵列10的当前编码部分用阴影线示出。三个线程当前在对熵编码样本阵列10起作用并基于立即处理来输出熵片段的组块：例如参见按熵片段顺序16的与样本阵列10的部分12对应的第一熵片段。在编码部分12的子部分12a之后，编码器由此形成组块390，即，熵编码核80执行完成程序以完成就算术编码而言到目前为止从子部分12a产生的算术比特流以形成组块390。然后针对熵片段12的后继子部分12b按编码顺序14来恢复编码程序，同时启动新的熵比特流。例如，这意味着重置熵编码核80的内部状态，比如概率区间宽度值和概率区间偏移值。然而，没有重置概率估计。仍然保持不变。在图20中用箭头392示出。在图20中，示例性地示出将熵片段或部分12细分为两个以上子部分，因此第二组块1b在沿编码顺序14到达部分12的末端之前要进行熵完成处理，由此启动行中的下一组块。

[0182] 并发地，另一线程对按熵片段顺序16的第二熵片段或部分12起作用。在完成第二熵片段/部分12的第一子部分时，输出组块2a，由此开始对第二熵片段的剩余部分进行熵编码，然而，同时保持概率估计在组块2a末端有效。

[0183] 利用时间轴394，图20试图示出组块390一被完成就被输出。这会造成与图16中描述的交错类似的交错。每个组块可以被分组化为数据包并按任何顺序经由某个传输层传输至解码侧。使用箭头396示出了传输层。

[0184] 解码器必须将组块重新分配给其子部分12a、12b、等等。为此，每个组块390都可以具有揭示其相关联子部分12a或12b、即对其进行描述的语法元素在各组块中被熵编码成的子部分的开始端的位置的报头部398。通过使用该信息，解码器能够使每个组块390与其熵

片段和此熵片段的部分12内的子部分相关联。

[0185] 出于说明目的，图20还示例性地示出了熵片段12的连续子部分12a和12b之间的交界处不一定与连续编码单元50的界限一致的可能性。相反，交界处可以在编码单元的前述示例性多树细分的更深层面定义。报头398中包含的位置信息可以足够精确地指示子部分与当前组块390相关联的开始端，以便识别各编码单元的各子区块，即，据此描述各子区块的语法元素序列内的位置。

[0186] 如从上述讨论可看出，几乎没有编码效率损失是由将熵片段细分为组块造成的。只有熵完成处理和分组化可能涉及某些编码效率损失，但另一方面低延迟增益是极大的。

[0187] 再者，请注意，刚提及的空间细分组块方面以空间和时间的方式还结合熵片段界限上的上述概率估计耦合工作。

[0188] 解码器比如图6的解码器可以撤消组块传输，如下所述。具体地，解码器可以检查当前组块属于哪个熵片段。该检查基于前述位置信息来进行。然后，可以检查当前组块沿熵编码路径14是否与对应熵片段的部分的第一子部分对应。如果对应，则当沿熵编码路径对与预定熵片段的部分的第二子部分对应的另一组块进行熵解码时，解码器可以在调试各概率估计的过程中对当前组块进行熵解码并将在对当前组块进行熵解码结束时显示出来的各概率估计的状态纳入考虑。“纳入考虑”可能涉及将组块1b的开始端的概率估计设置为等于通过开始于组块1a的开始端，组块1a的子部分12a的末端的概率估计状态的概率调试显示出来的概率估计，或等于来自如上所述的其他熵片段的熵概率估计的组合。只要关注第一组块12a的开始端的概率初始化，就可参照上述讨论，因为这同样形成对应熵片段的开始端。换句话说，如果当前片段按顺序14是第二组块或下一组块，则解码器使用概率估计对当前组块进行熵解码，所述概率估计取决于在沿熵编码路径14对在对应于当前组块的子部分之前的预定熵片段的部分的子部分对应的组块进行熵解码结束时显示出来的概率估计。

[0189] 上述描述揭示了不同方法，所述方法可用于并行编码和解码并可用于最佳化即将出现的HEVC视频编码标准中的原有处理。已对熵片段进行了简短概述。示出熵片段形成的方式，所述优点可通过切片来实现并且这些技术还会导致什么样的损失。提出了数种方法，假设通过更好地利用LCU之间的本地依赖以及不同帧的LCU之间的时间依赖改善了以帧为单位的LCU(最大编码单元)的概率了解过程。断言不同组合对进行编码和解码并行化和不进行编码和解码并行化的这两个概念提供了改进。

[0190] 例如采用所提出的最佳组合方法进行的高效率性能改进为与没有使用熵片段的HM3.0相比，帧内为-0.4％，低延迟为-0.78％，随机访问为-0.63％，且与利用一般重新初始化的熵片段方法相比，帧内为-0.7％，低延迟为-1.95％，随机访问为-1.5％。

[0191] 具体地，上文中尤其提出了以下技术。

[0192] • 不仅使用LCU的本地依赖而且还使用其时间依赖来最佳化CABAC概率在编码每个LCU之前的调试，参见图1到图9，图17和图18。

[0193] • 实现更大的解码灵活性，还可以使用熵片段，使得帧中的某些区域独立于彼此。

[0194] • 允许片段/熵片段起始位置的最小信令传送以便进行并行化，例如波阵面处理，参见图15。

[0195] • 允许在并行化编码器-发射器-接收器-解码器环境下通过熵片段/片段的交错传输进行的低延迟传输，参见图16。

[0196] 上文提及的所有方法已经在HM3.0中进行了整合和测试。在表1和表2中呈现了所获得的结果，其中参考点为没有进行任何熵片段实现的HM3.0(其中2LCU—上行的第二LCU的用途；2LCU+概率调试—利用概率调试方法融合的2LCU；时间—每个LCU都进行概率调试的时间依赖(参考帧的结束状态)的用途)。

[0197] 表1 1线程的RD总结果

[0198]

1 线程	2LCU	2LCU+ 概 率调试	概率调试	时间
帧内	0.14	-0.31	-0.4	
低延迟	0.66	0.09	-0.12	-0.78
随机	0.48	-0.1	-0.24	-0.63

[0199] 表2 4线程的总RD结果

[0200]

4 线程	2LCU	2LCU+ 概 率调试	概率调试	时间
帧内	0.19	-0.27	-0.27	
低延迟	1.01	0.54	0.63	0.17
随机	0.74	0.2	0.24	0.01

[0201] 然而，感兴趣的是了解所提出的方法如何通过重新初始化LCU的每行的开始端的概率来影响波阵面。在表3和表4中示出了这些结果(其中orig_neiInit是不使用熵片段、通过熵片段的用途、通过重新初始化进行的比较HM3.0)。

[0202] 表3 1线程的总RD结果，参考是新的初始化

[0203]

1 线程	orig_neiInit	2LCU	2LCU+ 概率调试	概率调试	时间
帧内	0.3	-0.15	-0.62	-0.7	
随机访问	0.9	-0.4	-0.99	-1.13	-1.51
低延迟	1.19	-0.51	-1.08	-1.29	-1.93

[0204] 表4 4线程的总RD结果，参考是新的初始化

[0205]

4 线程	orig_neiInit	2LCU	2LCU+ 概率调试	概率调试	时间
帧内	0.34	-0.14	-0.61	-0.61	
随机访问	1.18	-0.43	-0.96	-0.96	-1.51
低延迟	1.61	-0.5	-0.92	-1.05	-1.41

[0206] 上述结果表明更多的使用帧内和帧之间的依赖以及合理应用已经获得的信息可

防止平均损失。

[0207] HEVC视频编码和解码的波阵面处理方法将使用相邻LCU之间的依赖以及时间帧依赖的可能性与波阵面并行处理的概念融合在一起。通过这种方式，可以减少损失并且可以实现性能提高。

[0208] 概率调试速度的增益通过计算空间相邻熵片段的概率调试来实现。

[0209] 如上所述，上述的所有方面可以彼此组合，因此针对某个方面提及某些实现可能性当然也适用于其他方面。

[0210] 虽然已经就装置的上下文描述了若干方面，但显然这些方面也表示对应的方法的描述，其中，方框或装置是与方法步骤或方法步骤的特征对应的。同理，在方法步骤的上下文中描述的方面也表示对应的方框或对应的装置的项目或特征结构的描述。部分或全部方法步骤可通过(或使用)硬件装置执行，类似例如微处理器、可编程计算机或电子电路。在某些实施例中，最重要方法步骤中的某一者或多者可通过这种装置执行。

[0211] 上文提及的本发明的编码信号可以存储在数字存储介质上或者可以在传输介质比如无线传输介质或有线传输介质比如互联网上传输。

[0212] 根据某些实现的要求，本发明的实施例已经在硬件或软件中实现。实现可使用数字存储介质进行，例如软盘、DVD、蓝光盘、CD、ROM、PROM、EPROM、EEPROM或闪存，其上存储有电子可读取控制信号，其与可编程计算机系统协作(或可协作)，从而执行各个方法。因此数字存储介质可为计算机可读介质。

[0213] 根据本发明的某些实施例包括具有电子可读控制信号的数据载体，其可与可编程计算机系统协作，以便执行本文所述方法中的一者。

[0214] 通常情况下，本发明的实施例可实现为具有程序代码的计算机程序产品，当该计算机程序产品在计算机上运行时，该程序代码可操作用来执行这些方法中的一个。该程序代码例如可存储在机器可读载体上。

[0215] 其它实施例包括用于执行本文所述方法中的一个且存储在机器可读载体上的计算机程序。

[0216] 换句话说，因此本发明方法的实施例为具有程序代码的一种计算机程序，用于当该计算机程序在计算机上运行执行本文所述方法中的一个。

[0217] 因此，本发明方法的进一步实施例为一种数据载体(或数字存储介质或计算机可读介质)包括于其上记录的用于执行本文所述方法中的一者的计算机程序。数据载体、数字存储介质或记录介质典型地是有形的和/或非过渡的。

[0218] 因此，本发明方法的进一步实施例为表示用于执行本文所述方法中的一者的计算机程序的数据流或信号序列。数据流或信号序列例如可配置来经由数据通讯连接例如透过互联网传送。

[0219] 进一步实施例包括一种处理构件，例如计算机或可编程逻辑设备，被配置来或适用于执行本文所述方法中的一者。

[0220] 进一步实施例包括其上安装有用于执行本文所述方法中的一者的计算机程序的计算机。

[0221] 根据本发明的进一步实施例包括一种装置或系统，其配置为(例如电子地或光学地)将用于执行本文描述方法的其中之一的计算机程序传递到接收器。接收器可以例如是

计算机、移动设备、存储器设备等等。该装置或系统可以例如包括用于将计算机程序传递到接收器的文件服务器。

[0222] 在某些实施例中，可编程逻辑设备(例如现场可编程门阵列)可用来执行本文所述方法的一部分或全部功能。在某些实施例中，现场可编程门阵列可与微处理器协作以执行本文所述方法中的一者。通常情况下，该等方法优选由硬件装置执行。

[0223] 上述实施例仅用于举例说明本发明的原理。应理解，本文所述的配置及细节的修改及变化对本领域的技术人员来说是显而易见的。因此，意图仅受随附的专利权利要求的范围所限制，而不受由本文实施例的描述及解说所呈现的特定细节所限制。

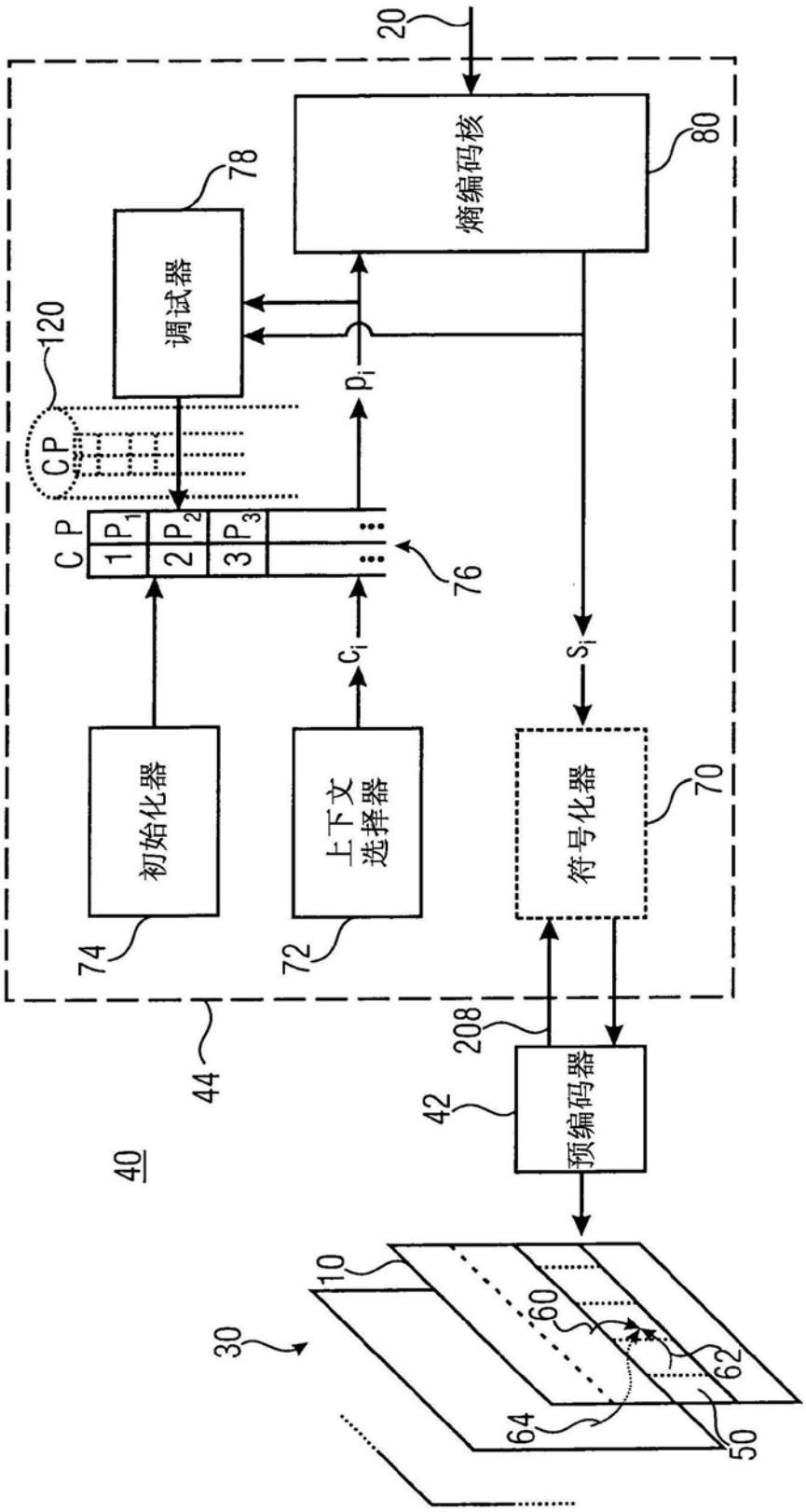

20
熵编码核
80
78
调试器
120
CP
p_i
C P
1 P_1
2 P_2
3 P_3
76
c_i
s_i
初始化器
上下文选择器
符号化器
70
74
72
208
44
42
预编码器
40
10
60
62
64
50
30

图1

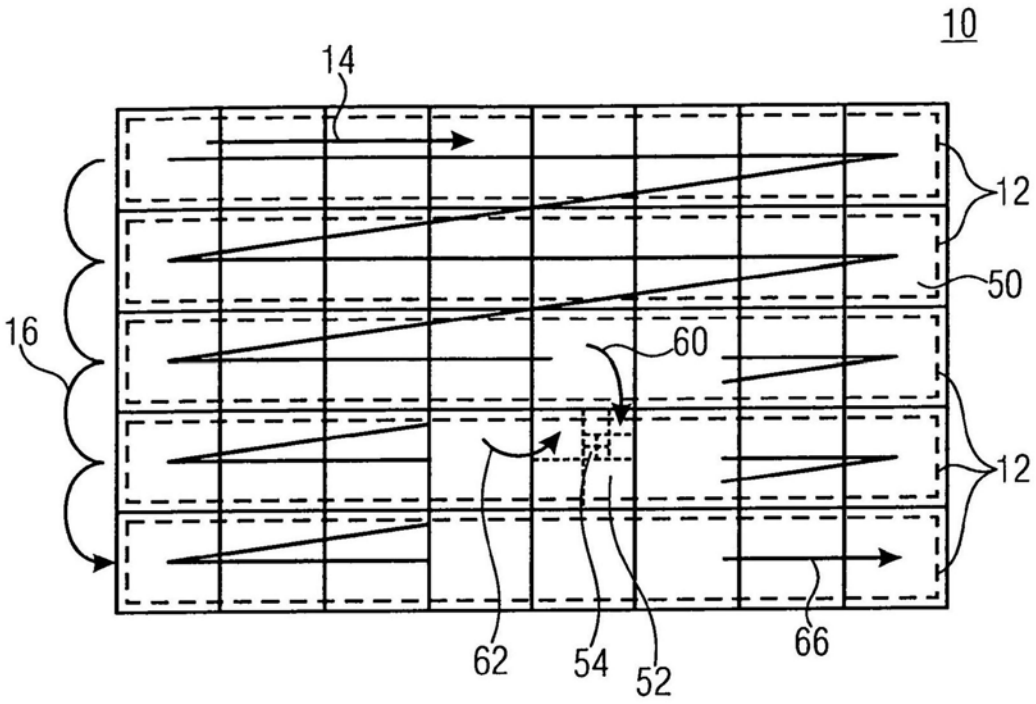

10
14
12
50
16
60
12
62
54
52
66

图2

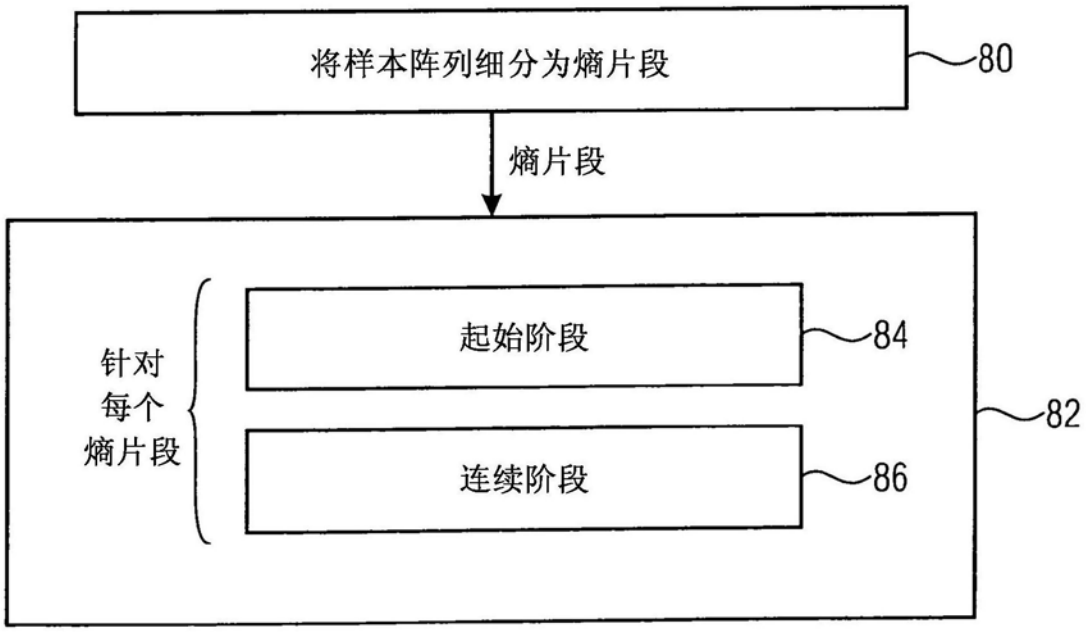

将样本阵列细分为熵片段
80
熵片段
针对
每个
熵片段
起始阶段
84
连续阶段
86
82

图3

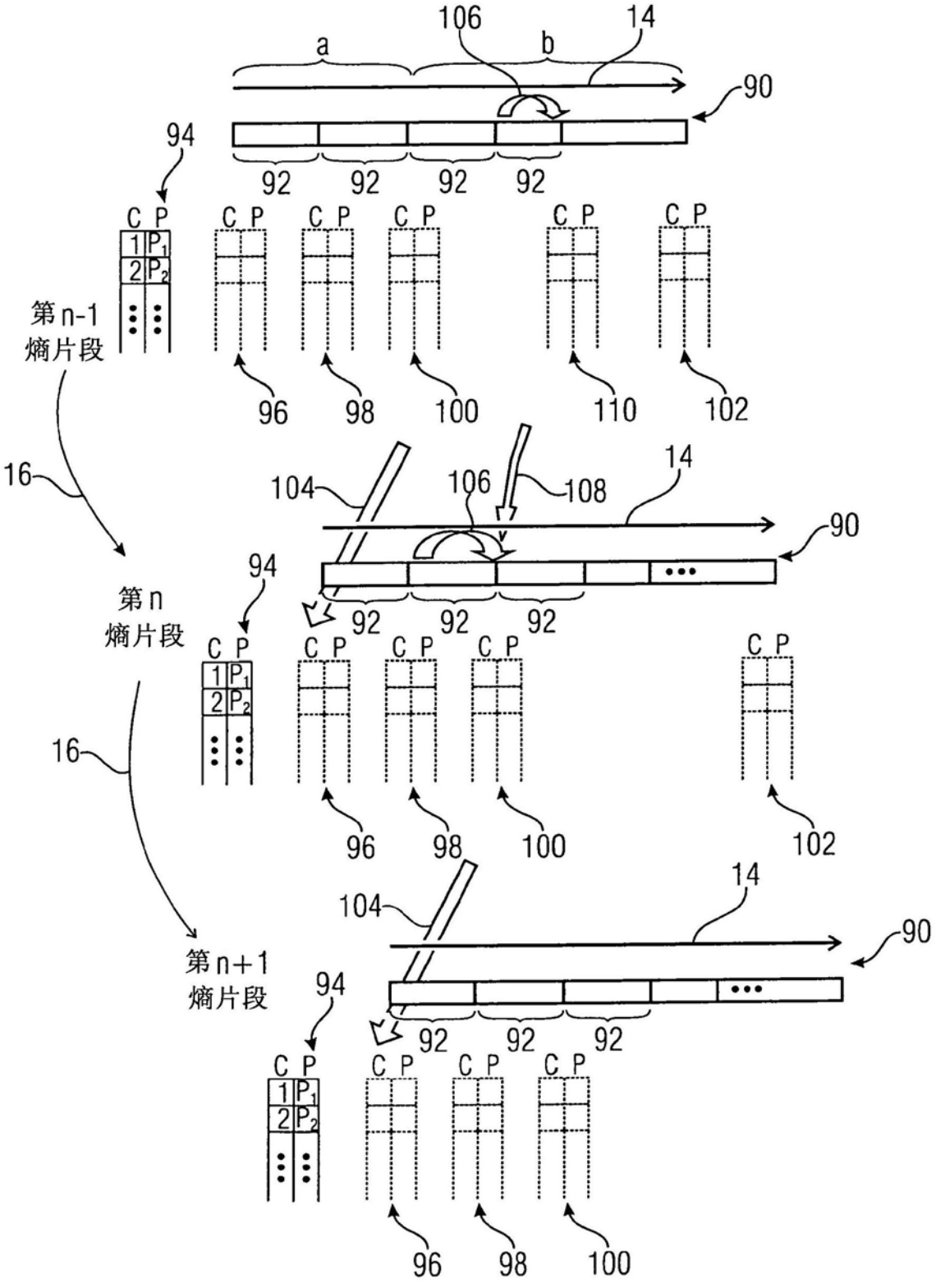

106
14
a
b
90
94
92
92
92
92
C P
1 P1
2 P2
第n-1
熵片段
96
98
100
110
102
16
104
106
108
14
90
94
第n
熵片段
92
92
92
C P
96
98
100
102
16
104
14
90
第n+1
熵片段
94
92
92
92
C P
96
98
100

图4

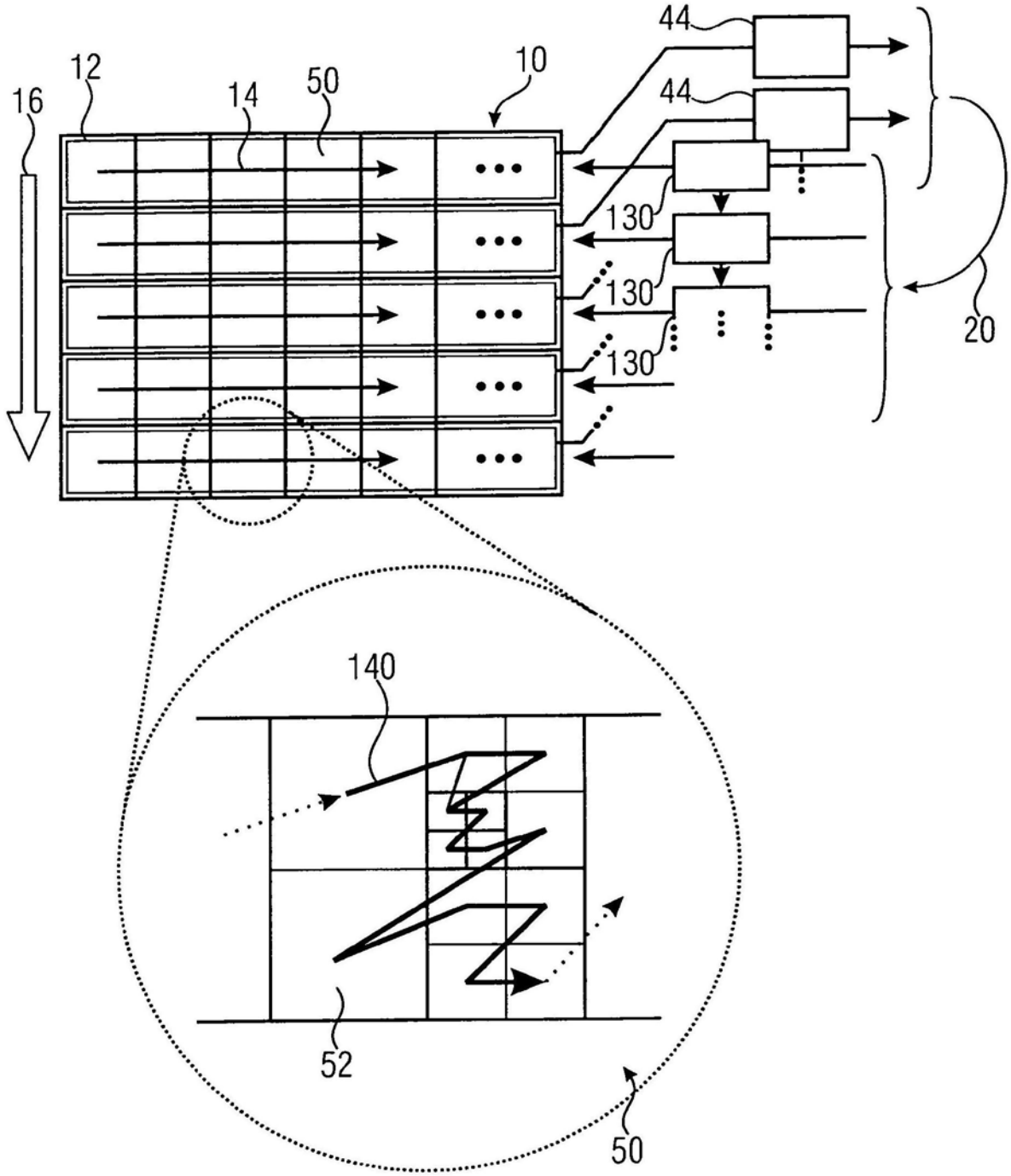


图5

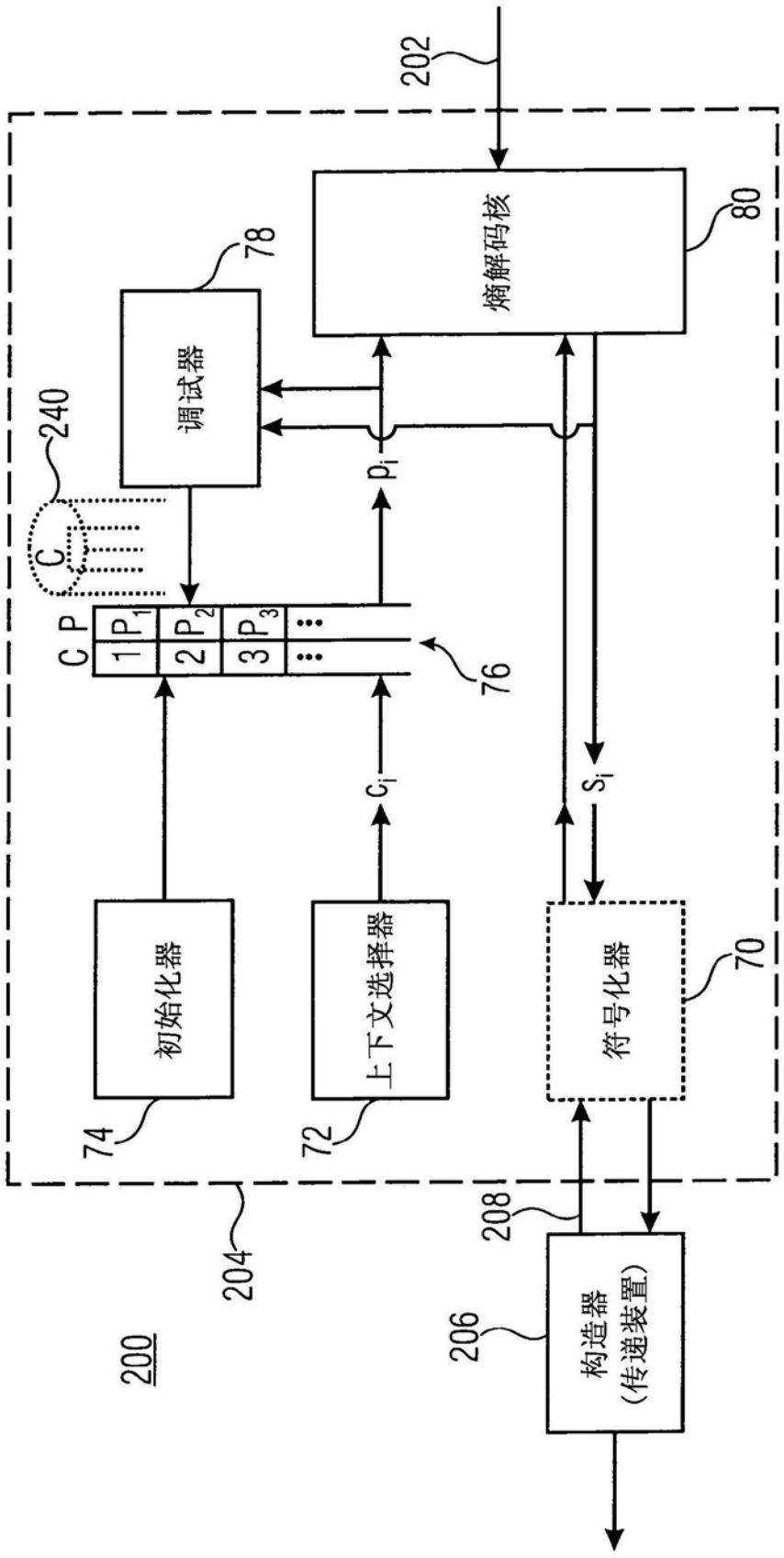

202
熵解码核
80
78
调试器
240
C
p_i
C P
1 P_1
2 P_2
3 P_3
76
c_i
s_i
初始化器
上下文选择器
符号化器
70
74
72
208
204
206
构造器
(传递装置)
200

图6

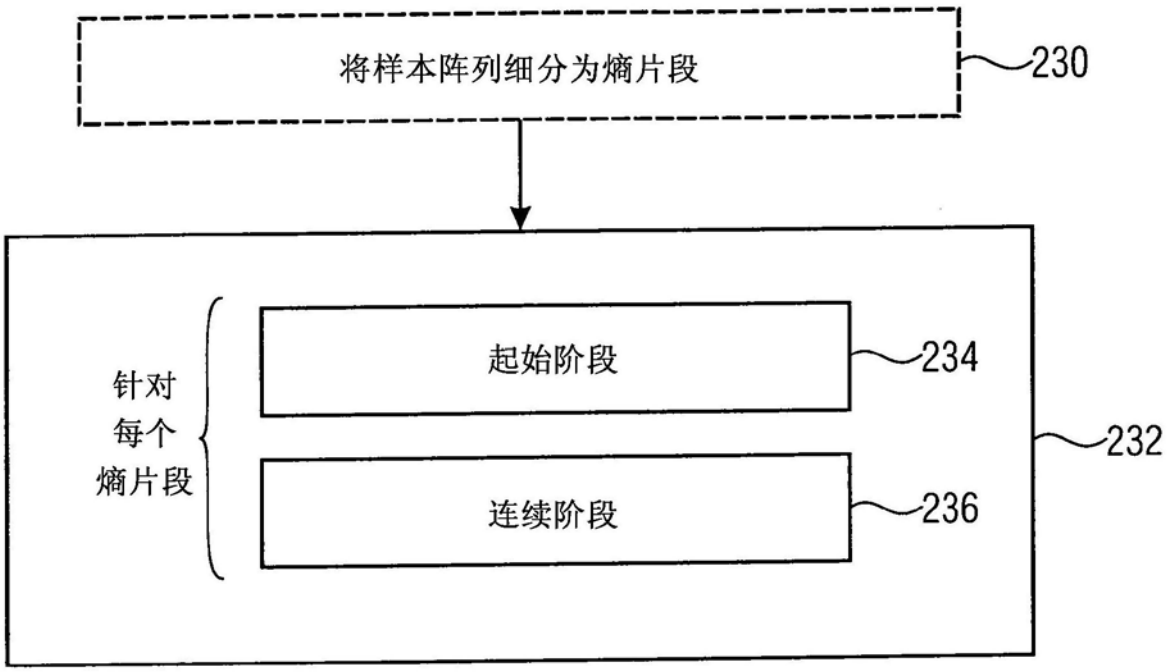


图7

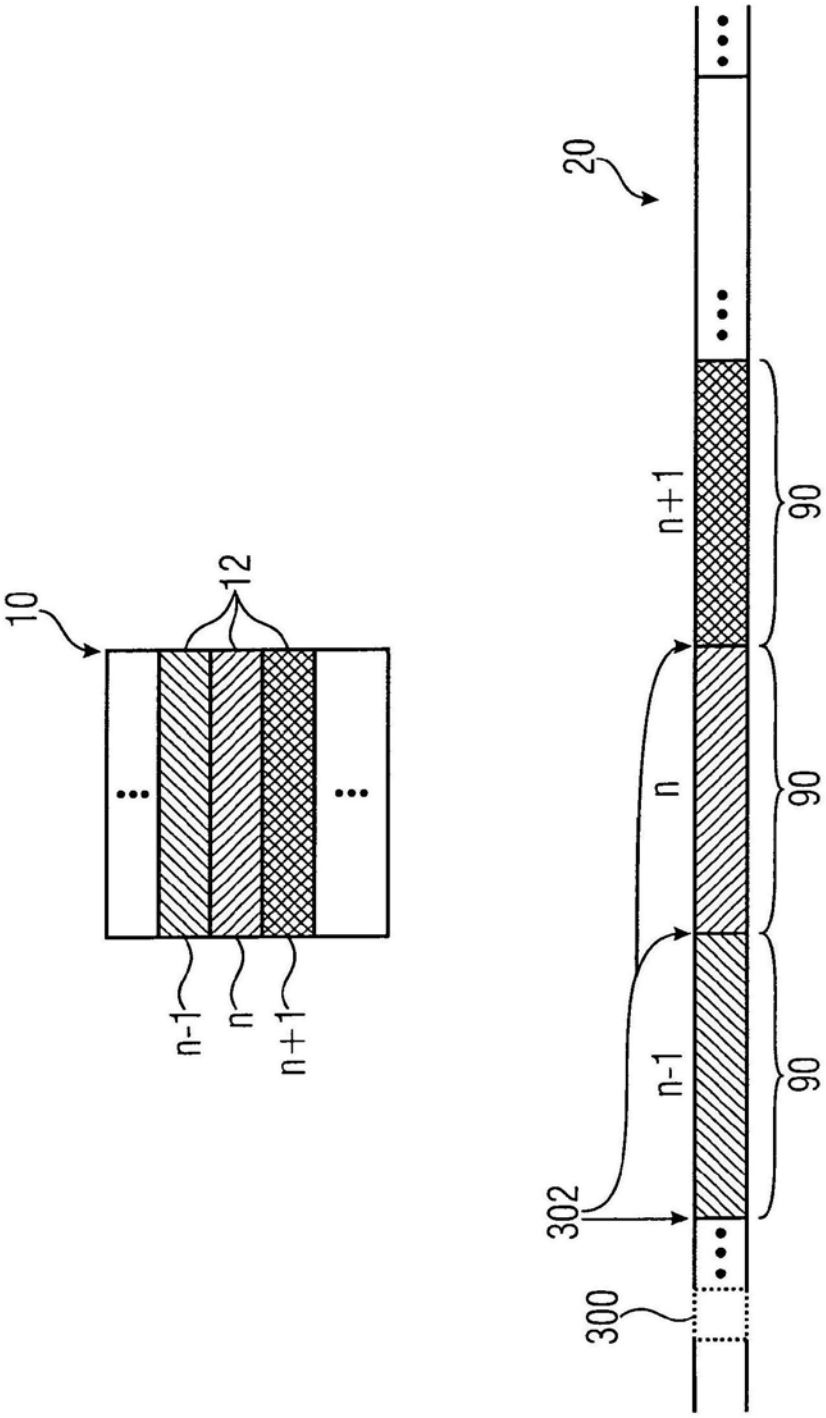


图8

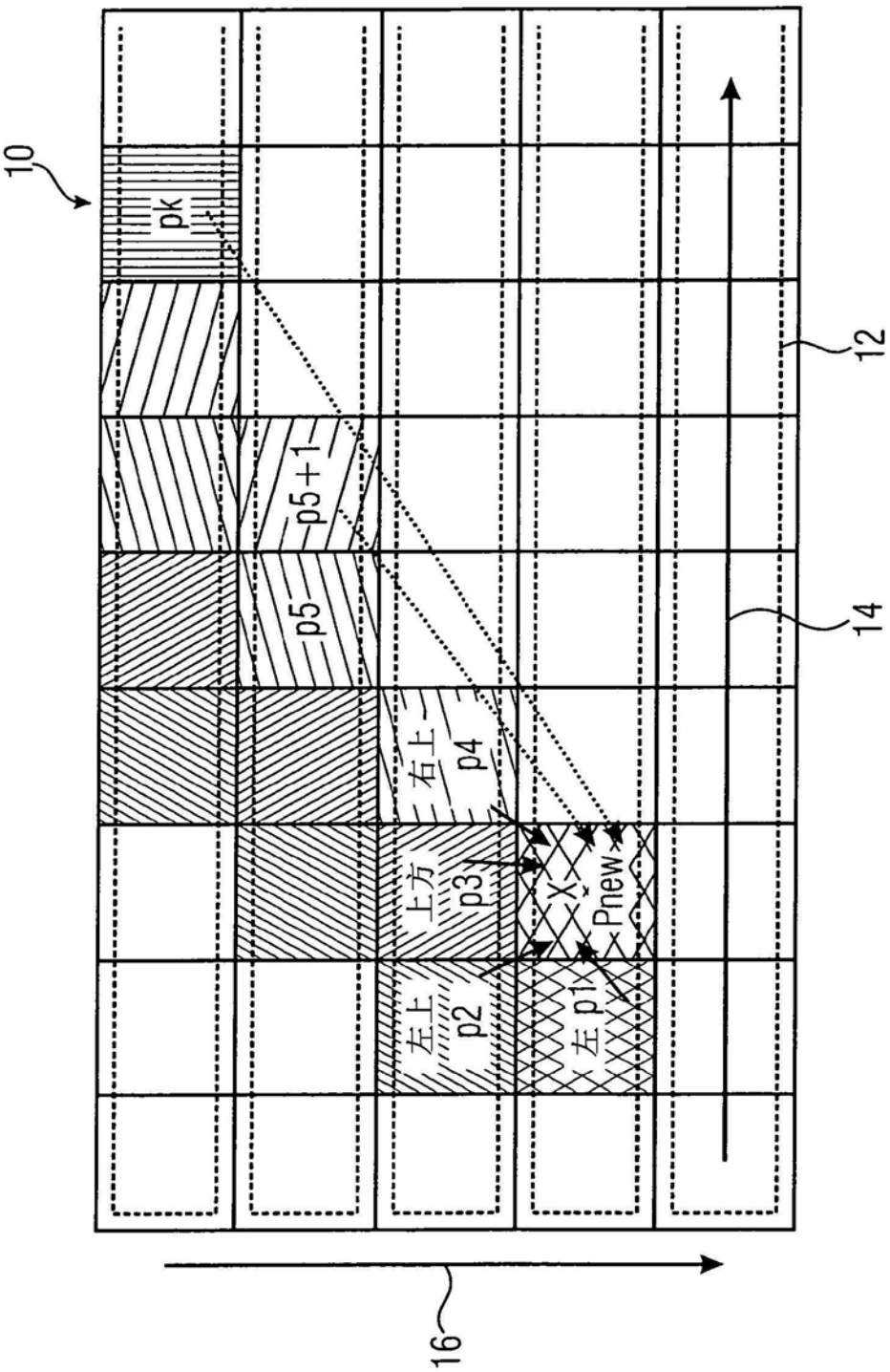


图9

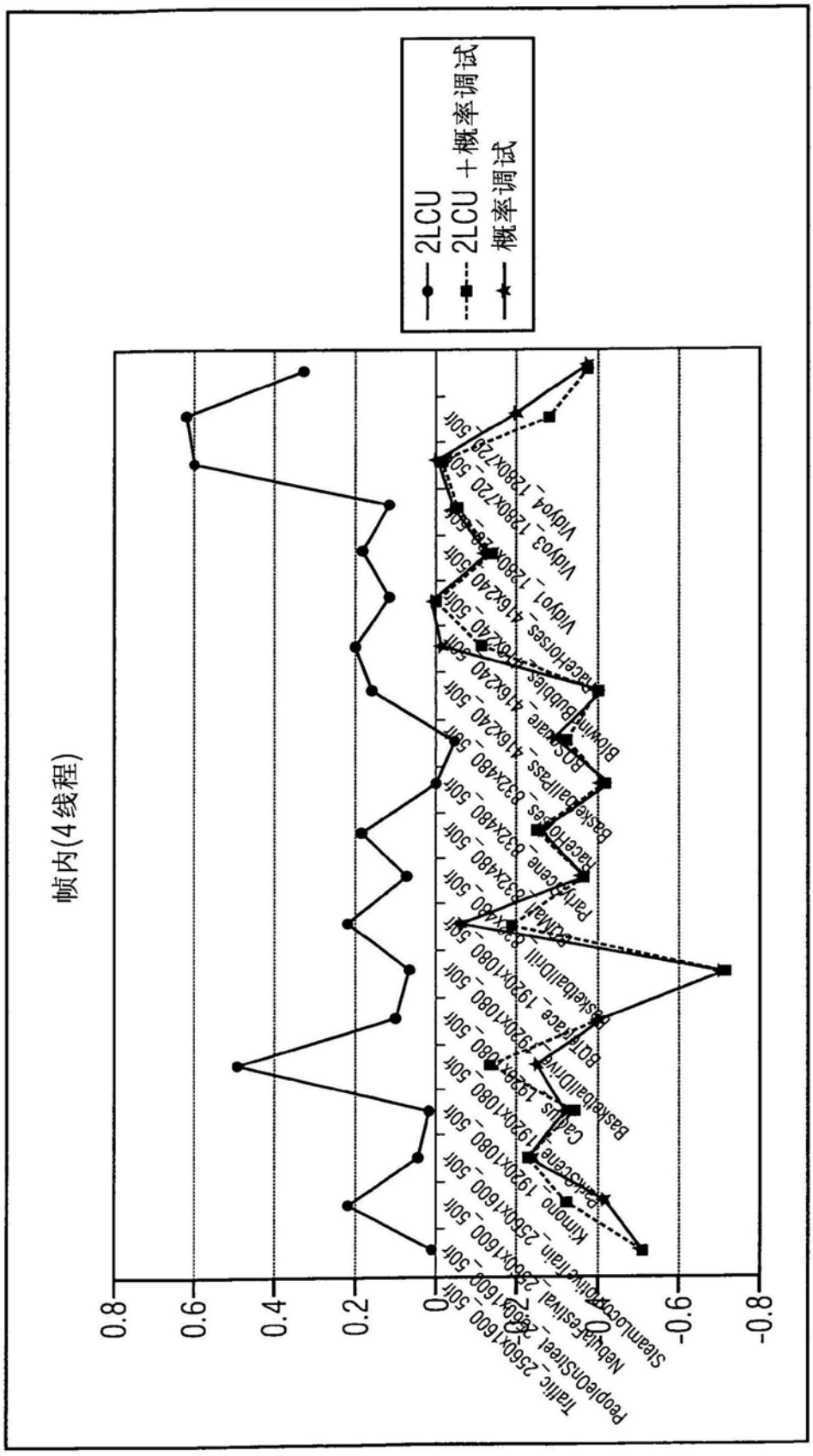

2LCU
2LCU +概率调试
概率调试
帧内(4线程)
0.8
0.6
0.4
0.2
0
-0.2
-0.6
-0.8
Traffic_2560x1600_50fr
PeopleOnStreet_2560x1600_50fr
NebulaFestival_2560x1600_50fr
SteamLocomotive_2560x1600_50fr
ParkScene_1920x1080_50fr
Kimono_1920x1080_50fr
Cactus_1920x1080_50fr
BQTerrace_1920x1080_50fr
BasketballDrive_1920x1080_50fr
BasketballDrill_832x480_50fr
BQMall_832x480_50fr
PartyScene_832x480_50fr
RaceHorses_832x480_50fr
BasketballPass_832x480_50fr
BQSquare_416x240_50fr
BlowingBubbles_416x240_50fr
RaceHorses_416x240_50fr
Vidyo1_1280x720_50fr
Vidyo3_1280x720_50fr
Vidyo4_1280x720_50fr

图10

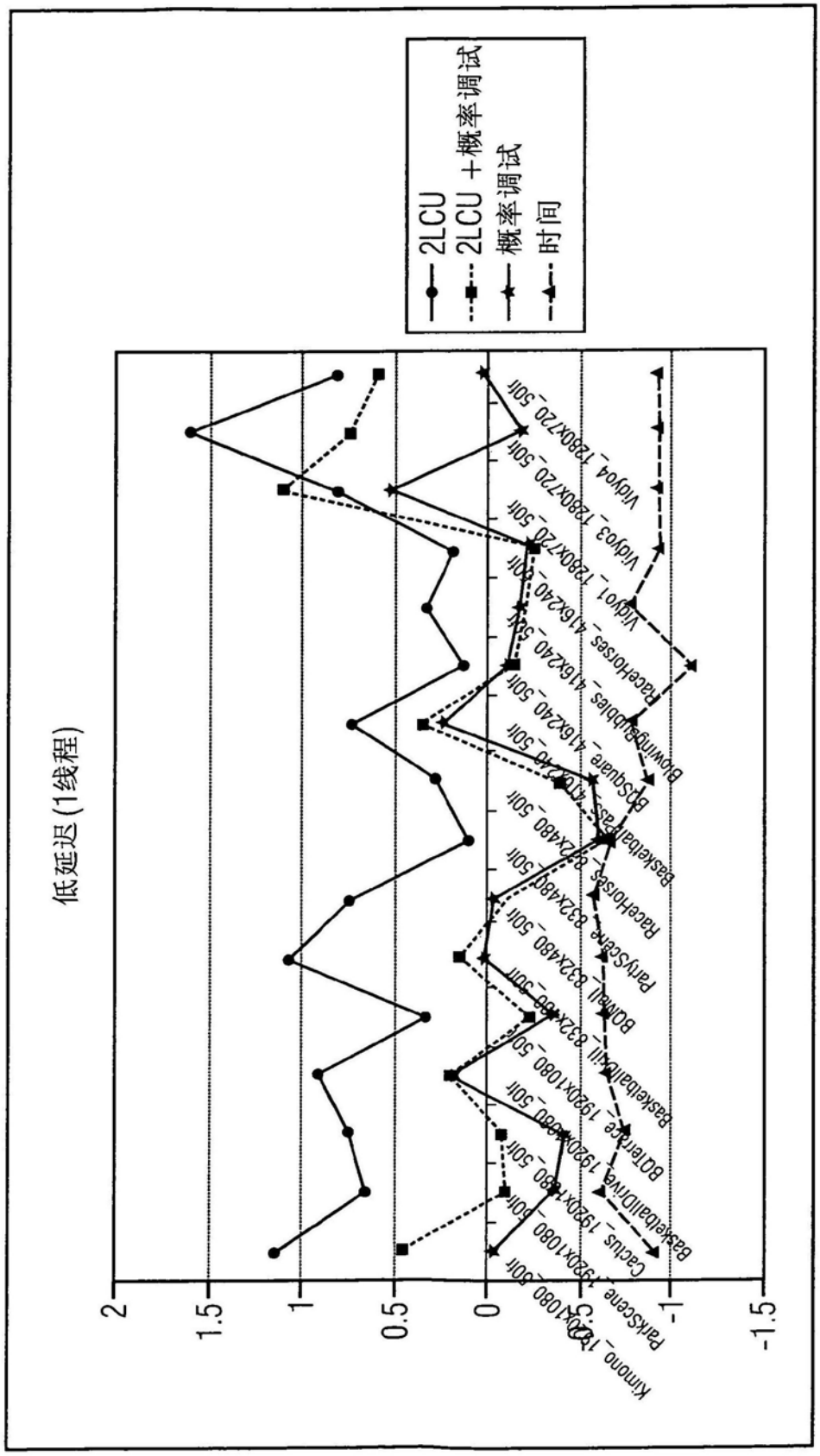

2LCU
2LCU +概率调试
概率调试
时间
低延迟(1线程)
2
1.5
1
0.5
0
-0.5
-1
-1.5
Kimono_1920X1080_50fr
ParkScene_1920X1080_50fr
Cactus_1920X1080_50fr
BasketballDrive_1920X1080_50fr
BQTerrace_1920X1080_50fr
BasketballDrill_832X480_50fr
BQMall_832X480_50fr
PartyScene_832X480_50fr
RaceHorses_832X480_50fr
BasketballPass_416X240_50fr
BQSquare_416X240_50fr
BlowingBubbles_416X240_50fr
RaceHorses_416X240_50fr
Vidyo1_1280X720_50fr
Vidyo3_1280X720_50fr
Vidyo4_1280X720_50fr

图11

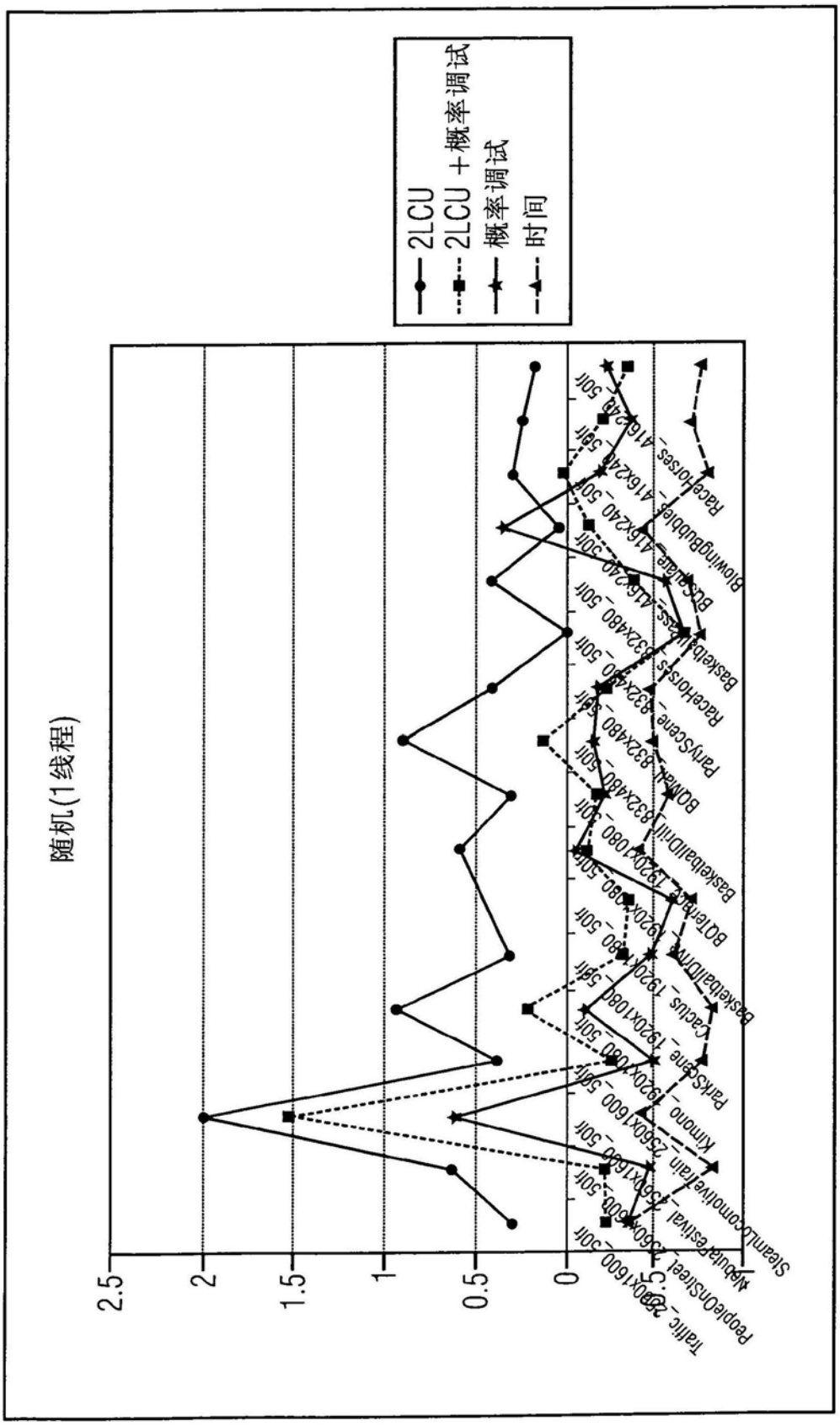

2LCU
2LCU +概率调试
概率调试
时间
随机(1线程)
2.5
2
1.5
1
0.5
0
-0.5
Traffic_2560x1600_50fr
PeopleOnStreet_2560x1600_50fr
MobileFestival_2560x1600_50fr
SteamLocomotiveTrain_2560x1600_50fr
Kimmono_1920x1080_50fr
ParkScene_1920x1080_50fr
Cactus_1920x1080_50fr
BasketballDrive_1920x1080_50fr
BQTerrace_1920x1080_50fr
BasketballDrill_832x480_50fr
BQMall_832x480_50fr
PartyScene_832x480_50fr
RaceHorses_832x480_50fr
BasketballPass_832x480_50fr
BQSquare_416x240_50fr
BlowingBubbles_416x240_50fr
RaceHorses_416x240_50fr

图12

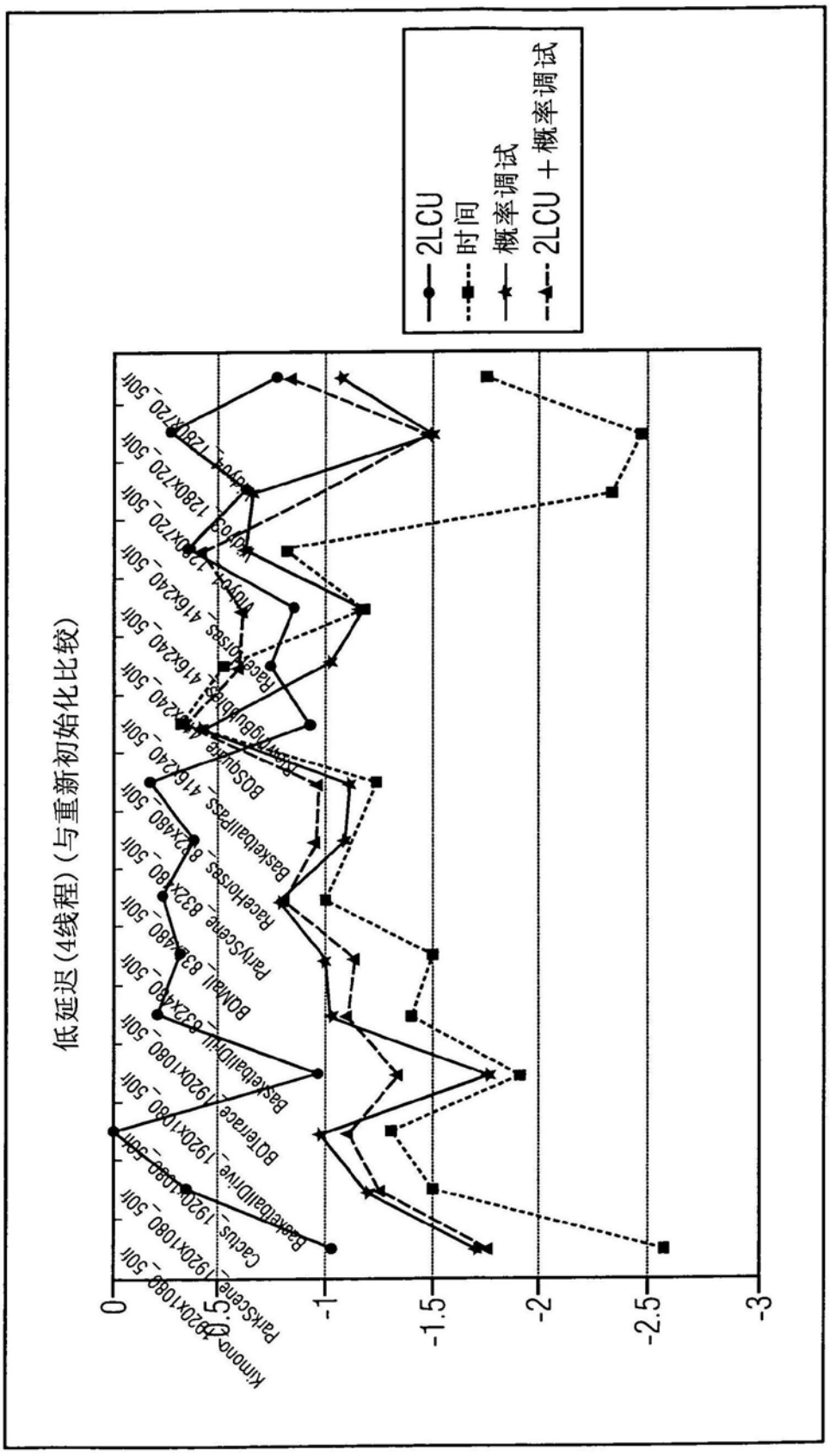

2LCU
时间
概率调试
2LCU + 概率调试
低延迟(4线程)(与重新初始化比较)
0
-0.5
-1
-1.5
-2
-2.5
-3
Kimono_1920X1080_50fr
ParkScene_1920X1080_50fr
Cactus_1920X1080_50fr
BasketballDrive_1920X1080_50fr
BQTerrace_1920X1080_50fr
BasketballDrill_832X480_50fr
BQMall_832X480_50fr
PartyScene_832X480_50fr
RaceHorses_832X480_50fr
BasketballPass_416X240_50fr
BQSquare_416X240_50fr
BlowingBubbles_416X240_50fr
RaceHorses_416X240_50fr
Vidyo1_1280X720_50fr
Vidyo3_1280X720_50fr
Vidyo4_1280X720_50fr

图13

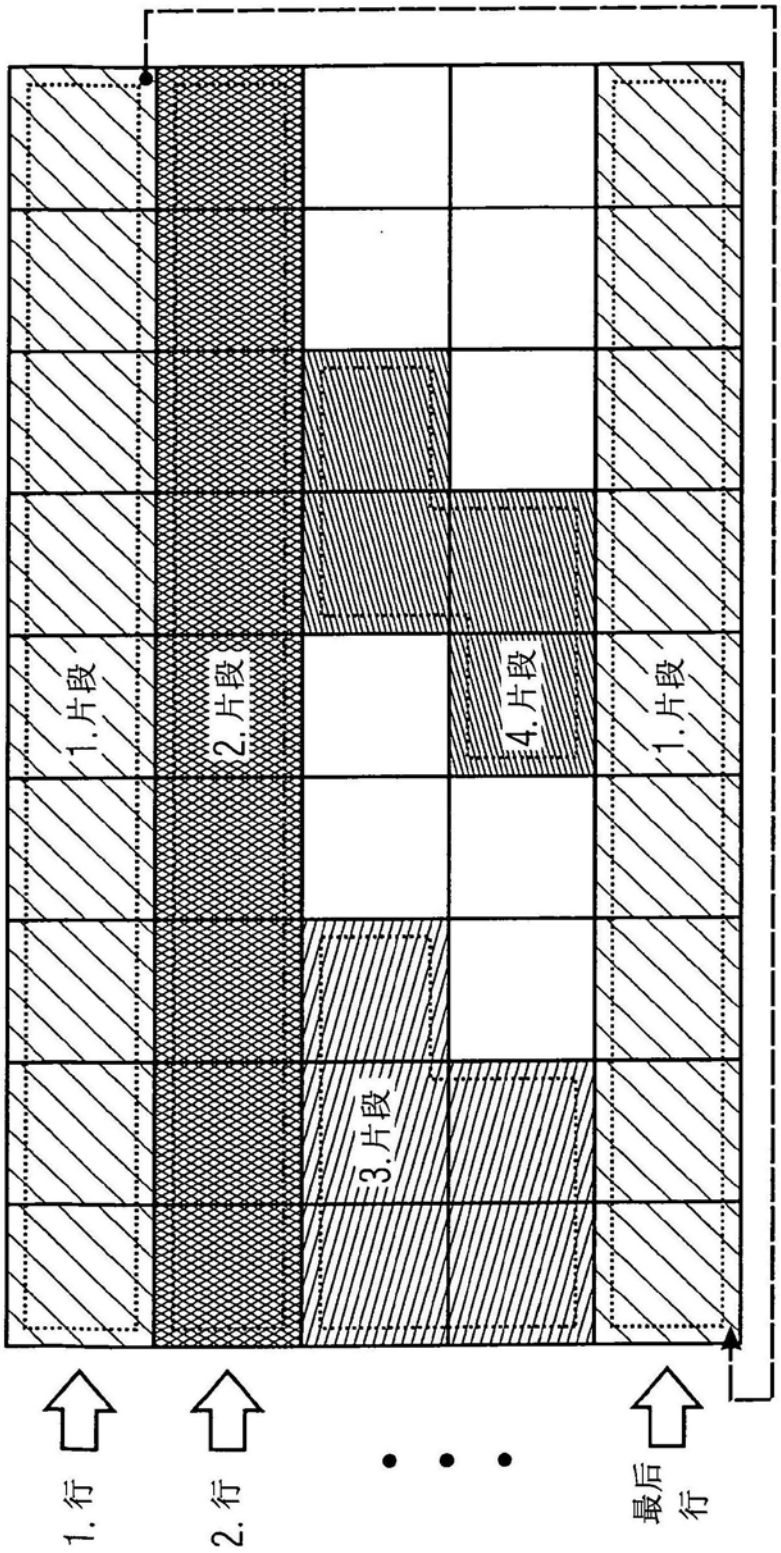


图14

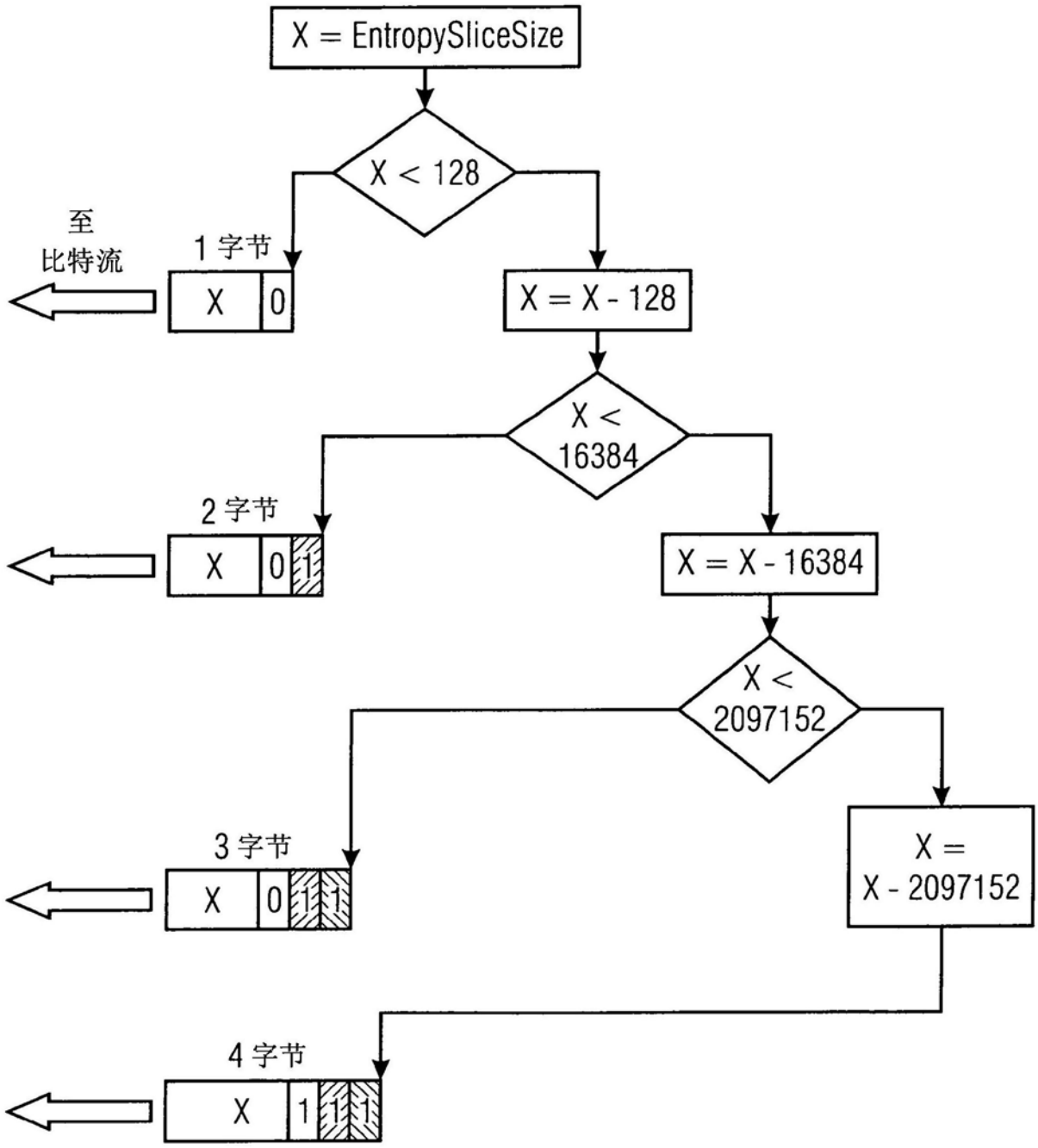

X = EntropySliceSize
X < 128
至
比特流
1 字节
X
0
X = X - 128
X < 16384
2 字节
X
0
1
X = X - 16384
X < 2097152
3 字节
X
0
1
1
X = X - 2097152
4 字节
X
1
1
1

图15

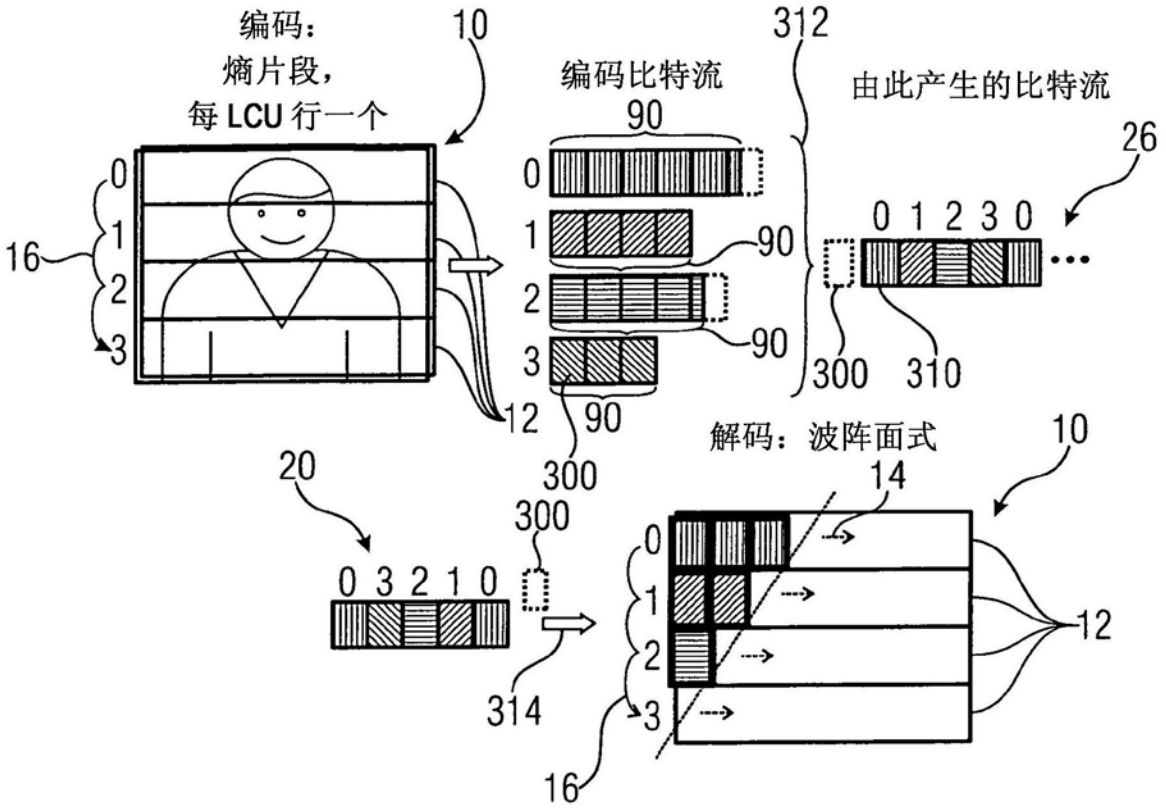

编码：
熵片段，
每 LCU 行一个
10
312
编码比特流
由此产生的比特流
90
26
0
1
2
3
16
12
300
310
解码：波阵面式
20
14
0 3 2 1 0
314

图16

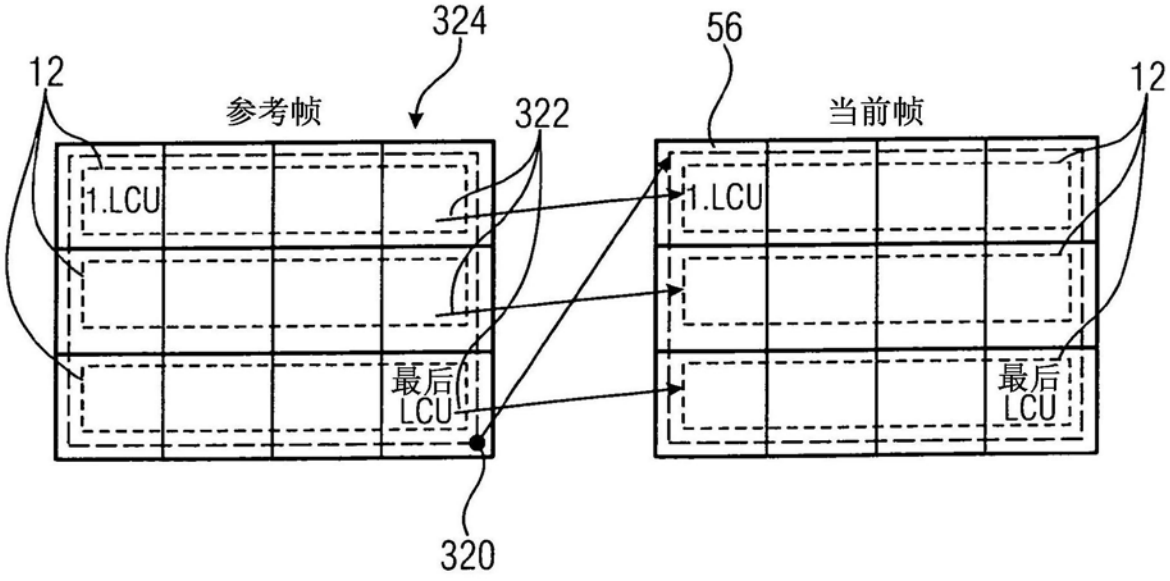

324
56
12
参考帧
322
当前帧
1.LCU
最后 LCU
320

图17

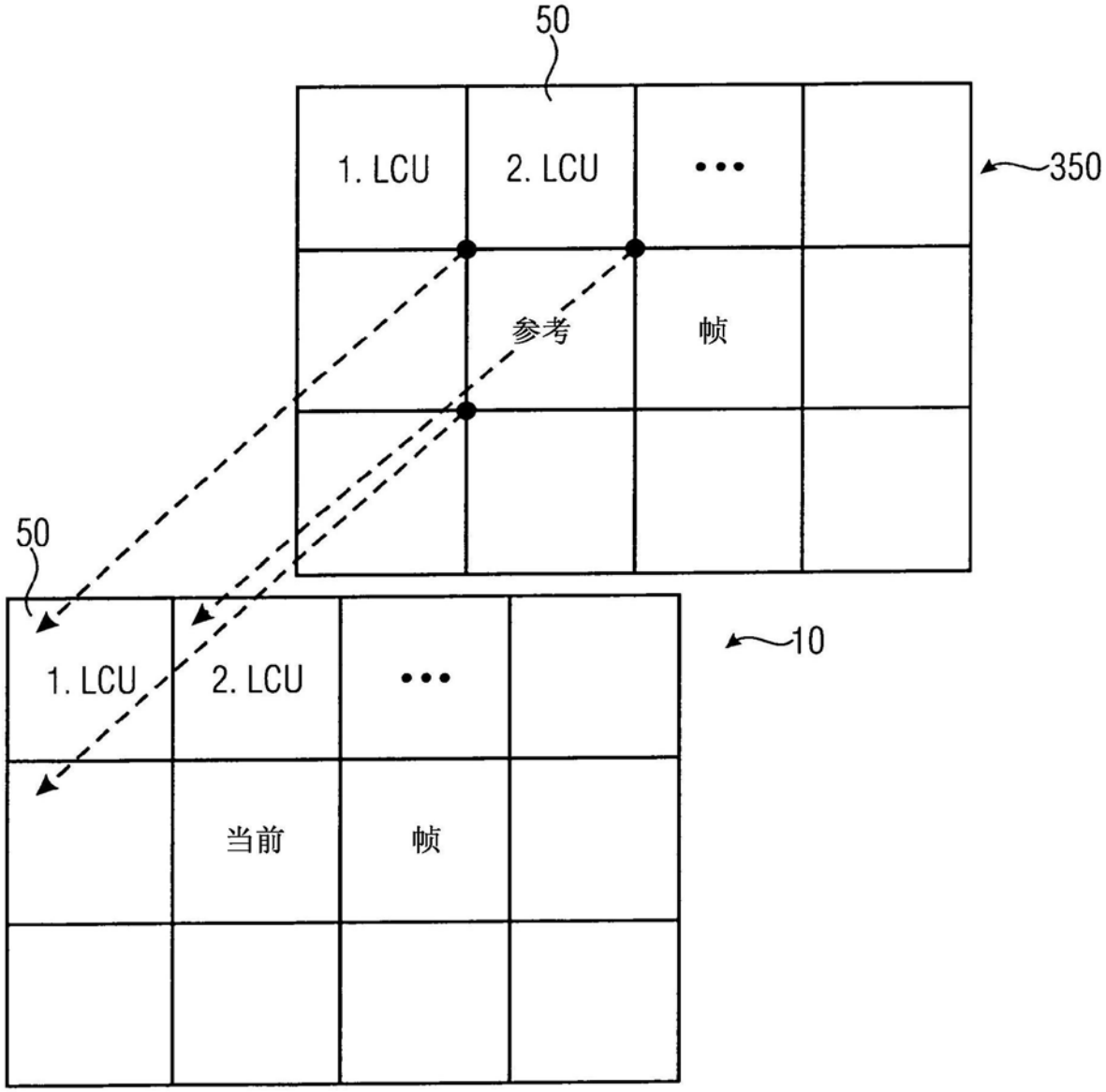


图18

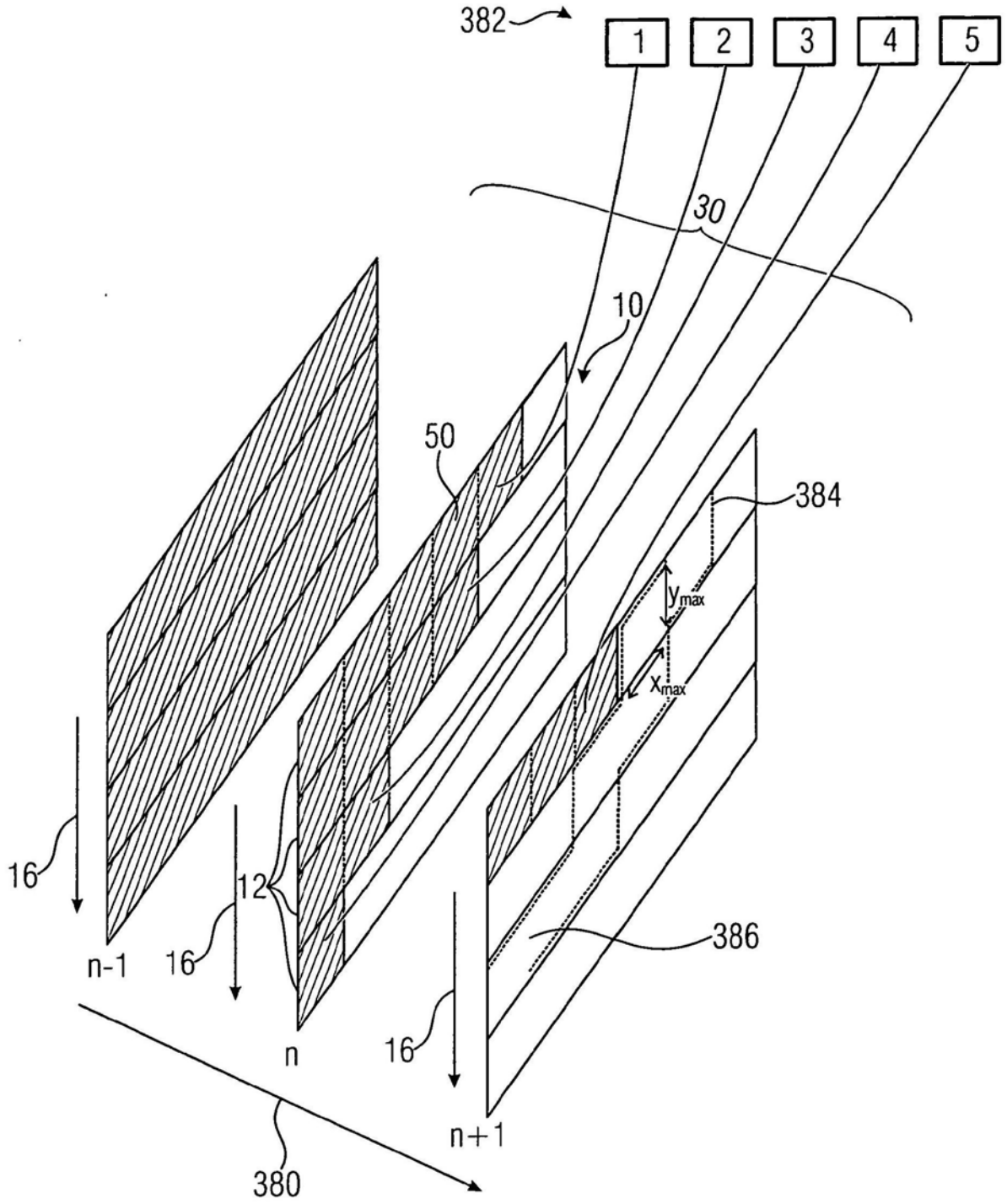


图19

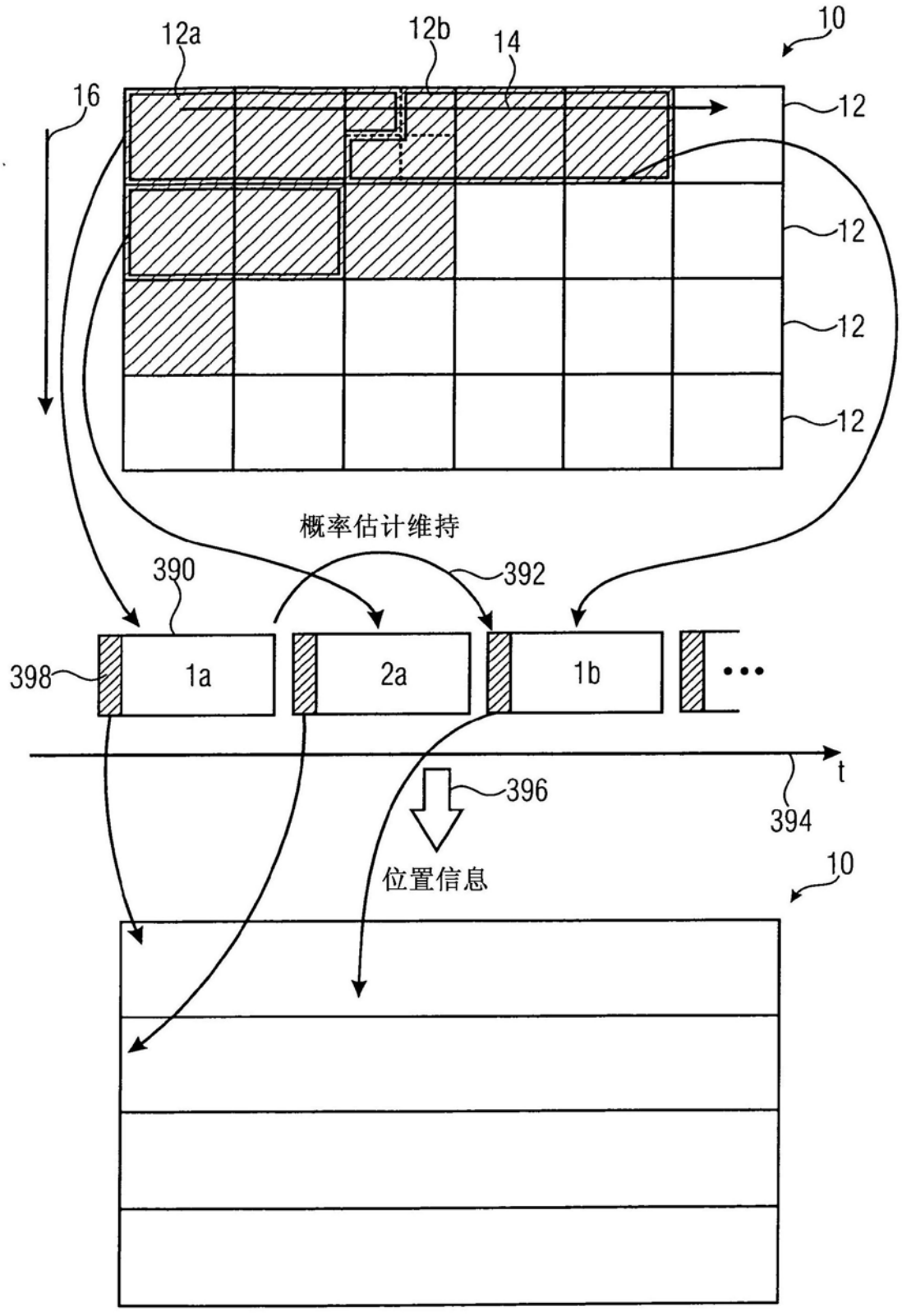

10
12a
12b
14
16
12
12
12
12
概率估计维持
390
392
398
1a
2a
1b
t
394
396
位置信息
10

图20