(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 106060549 B
(45)授权公告日 2019.07.12

(21)申请号 201610457834.4

(22)申请日 2011.09.30

(65)同一申请的已公布的文献号
申请公布号 CN 106060549 A

(43)申请公布日 2016.10.26

(30)优先权数据
12/895,676 2010.09.30 US

(62)分案原申请数据
201180047285.9 2011.09.30

(73)专利权人 夏普株式会社
地址 日本国大阪府大阪市阿倍野区长池町22番22号545-8522

(72)发明人 吉兰·米拉
克里斯多佛·A·西盖

(74)专利代理机构 中科专利商标代理有限责任公司 11021
代理人 赵伟

(51)Int.Cl.
H04N 19/196(2014.01)
H04N 19/176(2014.01)
H04N 19/70(2014.01)
H04N 19/46(2014.01)
H04N 19/91(2014.01)
H04N 19/174(2014.01)
H04N 19/436(2014.01)
H03M 7/40(2006.01)
H03M 7/42(2006.01)

(56)对比文件
CN 101076114 A,2007.11.21,
JP 2000285259 A,2000.10.13,
CN 101682786 A,2010.03.24,
JP 2005223533 A,2005.08.18,

审查员 吴峰

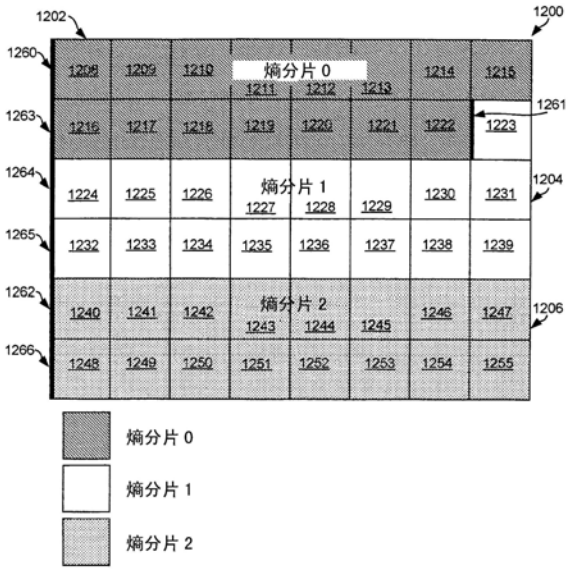
权利要求书1页 说明书38页 附图32页

(54)发明名称

视频编码和解码中上下文初始化的方法和系统

(57)摘要

公开了用于熵编码器和解码器中上下文模型初始化的系统和方法。在一些示例性实施例中,在当前宏块是行中第一个宏块时可以重置上下文模型,在熵分片内处理多个bin或比特。



CN 106060549 B

1.一种对视频进行解码的方法,包括如下步骤:

(a)确定当前位置是否是分片中最大编码单元LCU行的开始处;

(b)识别分片类型是否是I分片类型;以及

(c)初始化与所述分片类型相关联的上下文自适应二进制算术编码CABAC上下文,

其中,在确定当前位置是分片中LCU行的开始处且所识别的分片类型是I分片类型的情况下,所述步骤(c)在不使用用于确定初始化表的信息的情况下,初始化所述CABAC上下文。

2.一种对视频进行编码的方法,包括如下步骤:

(a)确定当前位置是否是分片中最大编码单元LCU行的开始处;

(b)识别分片类型是否是I分片类型;以及

(c)初始化与所述分片类型相关联的上下文自适应二进制算术编码CABAC上下文,

其中,在确定当前位置是分片中LCU行的开始处且所识别的分片类型是I分片类型的情况下,所述步骤(c)在不使用用于确定初始化表的信息的情况下,初始化所述CABAC上下文。

3.一种对视频进行解码的装置,包括:

(a)确定部,确定当前位置是否是分片中最大编码单元LCU行的开始处;

(b)识别部,识别分片类型是否是I分片类型;以及

(c)初始化部,初始化与所述分片类型相关联的上下文自适应二进制算术编码CABAC上下文,

其中,在确定当前位置是分片中LCU行的开始处且所识别的分片类型是I分片类型的情况下,所述初始化部在不使用用于确定初始化表的信息的情况下,初始化所述CABAC上下文。

4.一种对视频进行编码的装置,包括:

(a)确定部,确定当前位置是否是分片中最大编码单元LCU行的开始处;

(b)识别部,识别分片类型是否是I分片类型;以及

(c)初始化部,初始化与所述分片类型相关联的上下文自适应二进制算术编码CABAC上下文,

其中,在确定当前位置是分片中LCU行的开始处且所识别的分片类型是I分片类型的情况下,所述初始化部在不使用用于确定初始化表的信息的情况下,初始化所述CABAC上下文。

视频编码和解码中上下文初始化的方法和系统

[0001] 本申请是2013年3月29日(申请日:2011年9月30日)向中国专利局递交并进入中国国家阶段的题为“视频编码和解码中上下文初始化的方法和系统”的发明专利申请No.201180047285.9(PCT国际申请No.PCT/JP2011/073150)的分案申请。

技术领域

[0002] 本发明的实施例总体上涉及视频编码,具体地,本发明的一些实施例涉及并行视频编码和并行视频解码中用于上下文初始化的方法和系统。

背景技术

[0003] 现有的视频编码方法和标准,例如H.264/MPEG-4 AVC(H.264/AVC)以及JCT-VC提议测试模型(TMuC)与早期方法和标准相比可以以更高复杂性为代价提供更高的编码效率。对视频编码方法和标准的质量要求和分辨率要求的提高也会增加它们的复杂性。支持并行解码的解码器可以提高解码速度并降低存储器要求。此外,多核处理器的发展可以使支持并行解码的编码器和解码器令人期待。

[0004] H.264/MPEG-4 AVC[Joint Video Team of ITU-T VCEG and ISO/IEC MPEG,“H.264:Advanced video coding for generic audiovisual services,”ITU-T Rec.H.264and ISO/IEC 14496-10(MPEG4-Part 10),November 2007]是一种视频编解码器(编码器/解码器)规范,其全部内容通过引用合并于此,为了压缩效率,该规范在残差编码之前使用宏块预测来降低视频序列中的时间和空间冗余。

[0005] 提议测试模型(TMuC)[JCT-VC A205,“Test Model under Consideration,”June 16,2010]是JCT-VC的初始测试模型,其全部内容通过引用合并于此。使用具有可变大小的基本编码单元(被称作编码树块(CTB))的TMuC可以提供比H.264/AVC更大的灵活性。

发明内容

[0006] 本发明的一些实施例包括用于并行熵编码的方法和系统。本发明的一些实施例包括用于并行熵解码的方法和系统。

[0007] 根据本发明的第一方面,可以在熵分片的开始处使用上下文表来初始化用于熵编码的多个上下文模型。

[0008] 根据本发明的第二方面,可以在熵分片的行中的开始基本单元处使用上下文表来初始化用于熵编码的多个上下文模型。

[0009] 本发明的一个实施例公开了一种对视频序列中的视频帧进行解码的方法,所述方法包括:

[0010] 在视频解码器中,接收熵分片;

[0011] 在所述熵分片中识别分片开始基本单元;并且

[0012] 将与所述分片开始基本单元的熵解码相关联的上下文模型初始化为第一上下文模型。

[0013] 本发明的一个实施例公开了一种对视频序列中的视频帧进行解码的方法，所述方法包括：

[0014] 在视频解码器中，接收熵分片；

[0015] 接收上下文表重置标志的上下文表重置标志值；

[0016] 在所述熵分片中识别第一行分片基本单元；并且

[0017] 当所述上下文表重置标志值是第一值时，将与所述第一行开始基本单元的熵解码相关联的上下文模型初始化为第一上下文模型。

[0018] 本发明的一个实施例公开了一种对视频序列的视频帧进行编码的方法，所述方法包括：

[0019] 在编码器中，将视频序列的帧划分成至少一个重构分片，从而产生第一重构分片；

[0020] 形成与所述重构分片相对应的第一熵分片；

[0021] 识别与所述第一熵分片相关联的第一行开始基本单元；并且

[0022] 在与所述视频帧相关联的比特流中通知与所述第一行开始基本单元相关联的第一上下文模型。

[0023] 在结合附图考虑本发明的以下详细描述时，可以更容易地理解本发明的上述和其他目标、特征和优点。

附图说明

[0024] 图1是示出了H.264/AVC视频编码器的图(现有技术)；

[0025] 图2是示出了H.264/AVC视频解码器的图(现有技术)；

[0026] 图3是示出了示例性分片结构的图(现有技术)；

[0027] 图4是示出了示例性分片组结构的图(现有技术)；

[0028] 图5是示出了根据本发明实施例的示例性分片划分的图，其中，画面可以被划分成至少一个重构分片，并且重构分片可以被划分成多于一个熵分片；

[0029] 图6是示出了包括熵分片的本发明示例性实施例的图；

[0030] 图7是示出了包括在分片重构之前对多个熵分片进行并行熵解码的本发明示例性实施例的图；

[0031] 图8是示出了包括针对熵分片重构的画面级上预测数据/残差数据复用的本发明示例性实施例的图；

[0032] 图9是示出了包括针对熵分片重构的画面级上颜色平面复用的本发明示例性实施例的图；

[0033] 图10是示出了包括通过熵解码、形成熵分片和熵编码对比特流进行代码转换的本发明示例性实施例的图；

[0034] 图11是示出了包括将重构分片划分成多个熵分片的本发明实施例的图，其中，与多个熵分片中的每个熵分片相关联的bin的数目不超过bin的预定数目；

[0035] 图12是示出了包括将重构分片划分成多个熵分片的本发明示例性实施例的图，其中，bin可以与熵分片相关联，直到熵分片中bin的数目超过基于bin的预定最大数目的阈值为止；

[0036] 图13是示出了包括将重构分片划分成多个熵分片的本发明示例性实施例的图，其

中,与多个熵分片中的每个熵分片相关联的bin的数目不超过bin的预定数目,并且每个重构分片包含不多于预定数目的宏块;

[0037] 图14是示出了包括将重构分片划分成多个熵分片的本发明示例性实施例的图,其中,bin可以与熵分片相关联,直到熵分片中bin的数目超过基于bin的预定最大数目的阈值并且每个重构分片包含不多于预定数目的宏块为止;

[0038] 图15是示出了包括将重构分片划分成多个熵分片的本发明示例性实施例的图,其中,与多个熵分片中的每个熵分片相关联的比特的数目不超过比特的预定数目;

[0039] 图16是示出了包括将重构分片划分成多个熵分片的本发明示例性实施例的图,其中比特可以与熵分片相关联,直到熵分片中比特的数目超过基于比特的预定最大数目的阈值为止;

[0040] 图17是示出了包括多个bin编码器的本发明示例性实施例的图;

[0041] 图18是示出了包括多个上下文适应单元的本发明示例性实施例的图;

[0042] 图19是示出了包括多个bin编码器和多个上下文适应单元的本发明示例性实施例的图;

[0043] 图20是示出了包括将重构分片划分成多个熵分片的本发明示例性实施例的图,其中,熵分片的大小被约束以限制熵分片中由每个受约束熵编码器单元操作的bin的数目;

[0044] 图21是示出了将重构分片划分成多个熵分片的本发明示例性实施例的图,其中,熵分片的大小被约束以限制熵分片中由每个受约束熵编码器单元操作的bin的数目;

[0045] 图22是示出了包括多个bin解码器的本发明示例性实施例的图;

[0046] 图23是示出了包括多个上下文适应单元的本发明示例性实施例的图;

[0047] 图24是示出了包括多个bin解码器和多个上下文适应单元的本发明示例性实施例的图;

[0048] 图25是示出了将重构块划分为多个熵分片的示例图,其中,熵分片内的宏块连续;

[0049] 图26是示出了将重构块划分为多个熵分片的示例图,其中,熵分片中的宏块不连续;

[0050] 图27是示出了针对重构块到多个熵分片的示例性划分,在熵解码中使用的非连续相邻块的图,其中,熵分片中的宏块不连续;

[0051] 图28是示出了针对重构块到多个熵分片的示例性划分,在熵分片内块的熵解码和重构中使用的相邻块的图,其中,熵分片中的宏块不连续;

[0052] 图29是示出了熵分片首部位置约束的示例性比特流的示例性部分的图形表示。

[0053] 图30是示出了熵分片首部位置约束的示例性比特流的示例性部分的图形表示;

[0054] 图31是示出了包括对比特流的受约束部分进行处理以识别熵分片首部的熵解码器的本发明示例性实施例的图;

[0055] 图32是示出了包括对比特流的受约束部分进行处理以识别熵分片首部的熵解码器的本发明示例性实施例的图;

[0056] 图33是示出了包括对比特流的受约束部分进行处理以识别熵分片首部的熵解码器的本发明示例性实施例的图;以及

[0057] 图34是示出了根据本发明实施例的熵分片内示例性上下文模型初始化方案的图。

具体实施方式

[0058] 参照附图更好地理解本发明的实施例，其中，贯穿附图类似的部分有类似的附图标记。以上所列附图明确地合并作为该详细描述的一部分。

[0059] 应容易理解，在本文中总体上描述和示意的本发明的部件能够在各种不同配置中布置和设计。因此，以下对本发明方法和系统的实施例的更详细描述并不意在限制本发明的范围，而是仅代表本发明的当前优选实施例。

[0060] 本发明实施例的元件可以以硬件、固件和/或软件来实现。尽管本文披露的示例性实施例可以仅描述这些形式之一，但是应当理解，本领域技术人员能够以这些形式中的任何形式实现这些元件，而同时落在本发明发明的范围内。

[0061] 尽管使用熵编码/解码的任何视频编码器/解码器(编解码器)均可以由本发明的实施例来涵盖，但与H.264/AVC编码器和H.264/AVC解码器相关地示意本发明的许多示例性实施例。这意在示意本发明的实施例而并不作为限制。

[0062] 可以与宏块作为基本单元相关地描述本发明的许多示例性实施例。这意在示意并不作为限制。

[0063] 2008年3月28日递交的题为“Methods and Systems for Parallel Video Encoding and Decoding”的美国专利申请No.12/058,301的全部内容通过引起合并于此。2009年10月14日递交的题为“Methods and Systems for Parallel Video Encoding and Decoding”的美国专利申请No.12/579,236的全部内容通过引用合并于此。

[0064] 现有的视频编码方法和标准，例如，H.264/AVC和TMuC，与早期方法和标准相比可以以更高复杂性为代价提供更高的编码效率。对视频编码方法和标准的质量要求和分辨率要求的提高也会增加它们的复杂性。支持并行解码的解码器可以提高解码速度并降低存储器要求。此外，多核处理器的发展可以使支持并行解码的编码器和解码器令人期待。

[0065] H.264/AVC和许多其他视频编码标准和方法以基于块的混合视频编码方法为基础，其中，源编码算法是画面间(也被视为帧间)预测、画面内(也被视为帧内)预测以及预测残差的变换编码的混合。帧间预测可以利用时间冗余，并且帧内预测和预测残差的变换编码可以使用空间冗余。

[0066] 图1示出了示例性H.264/AVC视频编码器2的框图。可以提供输入画面4(也被视为帧)用于编码。可以产生预测信号6和残差信号8，其中预测信号6可以基于帧间预测10或帧内预测12。可以使用存储的参考画面16(也被视为参考帧)以及由运动估计部18确定的运动信息19通过运动补偿部14来在输入帧14和参考帧4之间确定帧间预测10。可以使用解码信号22通过帧内预测部20来确定帧内预测12。残差信号8可以通过从预测6中减去输入4来确定。通过变换/缩放/量化部24对残差信号8进行变换、缩放和量化，从而产生量化的变换系数26。可以通过将预测信号6与逆(变换/缩放/量化)部30使用量化的变换系数26产生的信号28相加，来产生解码信号22。运动信息19和量化的变换系数26可以由熵编码部32来进行熵编码，并且写入压缩的视频比特流34。可以使用重构的预滤波信号22通过解块滤波器36在编码器2处产生输出图像区域38(例如，参考帧的一部分)。

[0067] 图2示出了示例性H.264/AVC视频解码器50的框图。可以提供输入信号52(也被视为比特流)用于解码。接收到的符号可以由熵解码部54进行熵解码，从而产生运动信息56以及经量化和缩放的变换系数58。可以通过运动补偿部60对运动信息56和驻留在帧存储器64

中的参考帧84进行组合，并且可以产生帧间预测68。可以通过逆(变换/缩放/量化)部62对经量化和缩放的变换系数58进行逆量化、缩放和逆变换，从而产生解码的残差信号70。可以将残差信号70与预测信号78(帧间预测信号68或帧内预测信号76)相加。可以根据当前帧72中先前的解码信息通过帧内预测部74来预测帧内预测信号76。组合的信号72可以由解块滤波器80来滤波，并且经滤波的信号82可以写到帧存储器64。

[0068] 在H.264/AVC中，将输入画面划分成固定大小的宏块，其中，每个宏块覆盖矩形画面区域，该矩形画面区域包括亮度分量的16x16个采样、以及两个色度分量中每一个色度分量的8x8个采样。在其他编解码器和标准中，可以使用与宏块不同的基本单元或基本编码单元，例如编码树块。指定H.264/AVC标准的解码处理以用于处理作为宏块的单元。熵解码器54对压缩视频比特流52的语法元素进行解析，并且进行解复用。H.264/AVC指定两种备选熵解码方法：低复杂度技术，其基于对上下文自适应切换的可变长度码集合(也被称作CAVLC)的使用；以及计算要求更高的算法，是基于上下文的自适应二进制算术编码，被称作CABAC。在两种熵解码方法中，对当前符号的解码可以依赖于先前校正的解码符号以及自适应更新的上下文模型。此外，可以将不同的数据信息，例如，预测数据信息、残差数据信息，以及不同颜色平面复用在一起。在对元素进行熵解码之前可以不进行解复用。

[0069] 在熵解码之后，可以通过获得经逆量化和逆变换的残差信号、预测信号(帧内预测信号或帧间预测信号)来重构宏块。可以通过对每个解码的宏块进行解块滤波来降低块失真。在对输入信号进行熵解码之前可能不开始任何处理，从而使得熵解码成为解码中的潜在瓶颈。

[0070] 类似地，在可以允许备选预测机制的编解码器中，例如，H.264/AVC中的层间预测或其他可缩放编解码器中的层间预测，在解码器侧的所有处理之前熵解码是必需的，从而使得熵解码成为潜在瓶颈。

[0071] 在H.264/AVC中，可以将包括多个宏块的输入画面划分成一个或多个分片。如果在编码器和解码器侧使用的参考画面相同，可以对画面区域中分片所表示的采样的值进行正确解码，而无需使用来自其他分片的数据。因此，针对分片的熵解码和宏块重构不依赖于其他分片。具体地，将每个分片的开始处重置熵编码状态。当针对熵解码和重构二者定义邻域可用性时，将其他分片中的数据标记为不可用。在H.264/AVC中，可以并行地熵解码和重构分片。在分片边界上不允许帧内预测和运动矢量预测。解块滤波可以使用分片边界上的信息。

[0072] 图3示出了包括水平方向上的11个宏块和垂直方向上的9个宏块(标记为91-99的9个示例性宏块)的示例性视频画面90。图3示出了三个示例性分片：表示为“分片#0”100的第一分片，表示为“分片#1”101的第二分片，以及表示为“分片#2”102的第三分片。H.264/AVC解码器可以并行地解码和重构三个分片100、101、102。在针对每个分片的解码/重构处理开始时，针对熵解码和宏块重构二者，初始化或重置上下文模型并且将其他分片中的宏块标记为不可用。因此，对于宏块，例如，“分片#1”中标记为93的宏块，“分片#0”中的宏块(标记为91和92的宏块)不可以用于上下文模型选择或重构。而对于宏块，例如“分片#1”中标记为95的宏块，“分片#1”中的其他宏块(例如，标记为93和94的宏块)可以用于上下文模型选择或重构。因此，在分片内必须串行地进行熵解码和宏块重构。除非使用灵活宏块排序(FMO)来定义分片，否则按照光栅扫描顺序来处理分片内的宏块。

[0073] 灵活宏块排序定义分片组，来修改如何将画面划分成分片。分片组中的宏块可以由宏块至分片组图来定义，宏块至分片组图由分片首部中画面参数集和附加信息来通知。宏块至分片组图包括针对画面中每个宏块的分片组标识号。分片组标识号指定关联宏块属于哪个分片组。可以将每个分片组划分成一个或更多个分片，其中，分片是相同分片组内的宏块的序列，在具体分片组的宏块集合内按照光栅扫描顺序来处理宏块序列。在分片内必须串行地进行熵解码和宏块重构。

[0074] 图4示出了至三个分片组(由“分片组#0”103表示的第一分片组，由“分片组#1”104表示的第二分片组以及由“分片组#2”105表示的第三分片组)的示例性宏块分配。这些分片组103、104、105可以分别与画面90中的两个前景区域和一个背景区域相关联。

[0075] 本发明的一些实施例可以包括将画面划分成一个或更多个重构分片，其中，重构分片可以是自足的，即，如果在编码器和解码器侧使用的参考画面相同，则可以正确地对重构分片所表示的画面区域中的采样值进行重构，而无需使用来自其他重构分片的数据。重构分片内的所有重构的宏块在针对重构的邻域限定中是可用的。

[0076] 本发明的一些实施例可以包括将重构分片划分成多于一个熵分片，其中，熵分片可以是自足的，即，可以正确地对熵分片所表示的画面区域中的符号值进行熵解码，而无需使用来自其他熵分片的数据。在本发明的一些实施例中，可以在每个熵分片的解码开始时重置熵编码状态。在本发明的一些实施例中，当定义熵解码的邻域可用性时，可以将其他熵分片中的数据标记为不可用。在本发明的一些实施例中，在当前块的上下文模型选择中可以不使用其他熵分片中的宏块。在本发明的一些实施例中，可以仅在熵分片内更新上下文模型。在本发明的这些实施例中，与熵分片相关联的每个熵解码器可以保持自己的上下文模型集合。

[0077] ITU电信标准部，研究组16-2008年4月题为“Entropy slices for parallel entropy decoding”的投稿405的全部内容通过引用合并于此。

[0078] 本发明的一些实施例可以包括CABAC编码/解码。CABAC编码处理包括以下四个基本步骤：二进制化；上下文模型选择；二进制算术编码；和概率更新。

[0079] 二进制化：将非二值化符号(例如，变换系数、运动矢量、或其他编码数据)转换成二进制码，也被称作bin串或二进制化符号。当给出二值化语法元素时，可以绕过二进制化初始步骤。二值化符号元素或二进制化符号的元素可以被称作bin。

[0080] 对于每个bin，也可以执行以下操作：

[0081] 上下文模型选择：上下文模型是针对一个或更多个bin的概率模型。对于每个bin，上下文模型包括bin为“1”或“0”的概率。可以依据最近编码数据符号的统计，通常基于左侧和上侧相邻符号(如果可用)，对于可用模型选择来挑选模型。

[0082] 二进制算术编码：算术编码器根据所选的概率模型来编码每个bin，并且基于递归间隔再划分。

[0083] 概率更新：基于实际编码值来更新所选的上下文模型。

[0084] 上下文适应可以指以下处理：基于相邻符号值选择与bin相关联的上下文模型状态(也被称作状态)，并且更新分配给给定符号的模型概率分布。可以根据上下文模板来定义相邻符号的位置。

[0085] 在包括CABAC编码/解码的本发明的一些实施例中，在熵分片的解码开始时，可以

将所有上下文模型初始化或重置为预定模型。

[0086] 可以关于图5理解本发明的一些实施例。图5示出了包括水平方向的11个宏块和垂直方向上的9个宏块(标记为115-123的9个示例性宏块)的示例性视频帧110。图5示出了三个示例性重构分片:由“R_SLICE#0”111表示的第一重构分片,由“R_SLICE#1”112表示的第二重构分片以及由“R_SLICE#2”113表示的第三重构分片。图5还示出了第二重构分片“R_SLICE#1”112到三个熵分片的划分:以十字影线112-1示出表示为“E_SLICE#0”的第一熵分片,以垂直影线112-2示出表示为“E_SLICE#1”的第二熵分片,以斜影线112-3示出表示为“E_SLICE#2”的第三熵分片。可以并行地熵解码每个熵分片112-1、112-2、112-3。

[0087] 在本发明的一些实施例中,在对熵分片的熵解码期间,仅有来自熵分片内宏块的数据对于上下文模型选择是可用的。所有其他宏块可以标记为不可用。对于该示例性划分,当解码与标记为119的宏块区域相对应的符号时标记为117和118的宏块对于上下文模型选择是不可用的,这是因为标记为117和118的宏块在包含宏块119的熵分片的外部。然而,这些宏块117、118在重构宏块119时是可用的。

[0088] 在本发明的一些实施例中,编码器可以确定是否将重构分片划分成熵分片,并且编码器可以在比特流中发信号通知该决定。在本发明的一些实施例中,该信号可以包括熵分片标志,在本发明的一些实施例中可以表示为“entropy_slice_flag”。

[0089] 关于图6描述本发明的一些解码器实施例。在这些实施例中,可以检查130熵分片标志,并且如果熵分片标志指示不存在132与画面相关联的熵分片或重构分片,则可以将首部解析134成常规分片首部。可以重置136熵解码器状态,并且可以定义138用于熵解码和重构的邻域信息。然后对分片数据进行熵解码140,并且重构142分片。如果熵分片标志指示存在146与画面相关联的熵分片或重构分片,则可以将首部解析148为熵分片首部。可以重置150熵解码器状态,可以定义152用于熵解码的邻域信息,并且可以熵解码154熵分片数据。然后可以定义156用于重构的邻域信息,并且可以重构142分片。在分片重构142之后,可以检查158下个分片或画面。

[0090] 可以关于图7描述本发明的一些备选解码器实施例。在这些实施例中,解码器能够并行地解码,并且可以定义其自己的并行度,例如,考虑能够并行地解码N个熵分片的解码器。解码器可以识别170N个熵分片。在本发明的一些实施例中,如果在当前画面或重构分片中少于N个熵分片可用,则解码器可以根据后续画面或重构分片(如果可用)解码熵分片。在备选实施例中,在对后续画面或重构分片的部分解码之前,解码器可以等待,直到完全处理了当前画面或重构分片。在识别170多达N个熵分片之后,可以独立地对每个识别的熵分片进行熵解码。可以对第一熵分片进行解码172-176。对第一熵分片的解码172-176可以包括重置解码器状态172。在包括CABAC熵解码的一些实施例中,可以重置CABAC状态。可以定义174用于第一熵分片的熵解码的邻域信息,并且可以解码176第一熵分片数据。对于多达N个熵分片中的每一个,可以执行这些步骤(对于第N个熵分片的178-182)。在本发明的一些实施例中,解码器可以在对所有熵分片进行了熵解码时重构184熵分片。在本发明的备选实施例中,解码器可以在解码了一个或更多个熵分片之后开始重构184。

[0091] 在本发明的一些实施例中,当存在多于N个熵分片时,解码线程可以在完成对熵分片的熵解码时就开始对下个熵分片进行熵解码。因此,当线程完成对低复杂度熵分片的熵解码时,线程可以开始对附加熵分片进行解码,而无需等待其他线程完成它们的解码。

[0092] 在可以包容现有标准或方法的本发明的一些实施例中，熵分片可以共享根据该标准或方法的常规分片的大多数分片属性。因此，熵分片可以需要较小首部。在本发明的一些实施例中，熵分片首部可以允许解码器识别熵分片的开始并开始熵解码。在一些实施例中，在画面或重构分片的开始处，熵分片首部可以是常规首部或者重构分片首部。

[0093] 在包括H.264/AVC编解码器的本发明的一些实施例中，可以通过向现有分片首部添加新比特“entropy_slice_flag”来通知熵分片。表1列出了根据本发明实施例的熵分片首部的语法，其中C指示类别，描述符u(1)、ue(v)指示一些固定长度或可变长度编码方法。包括“entropy_slice_flag”的本发明的一些实施例可以实现提高的编码效率。

[0094] “first_mb_in_slice”指定了熵分片中的与熵分片首部相关联的第一宏块的地址。在一些实施例中，熵分片可以包括宏块序列。

[0095] “cabac_init_idc”指定了用于对在针对上下文模式的初始化处理中使用的初始化表加以确定的索引。

[0096]

slice_header() {	C	描述符
entropy_slice_flag	2	u(1)
if (entropy_slice_flag) {		
first_mb_in_slice	2	ue(v)
if (entropy_coding_mode_flag && slice_type != I && slice_type != SI)		
cabac_init_idc	2	ue(v)
}		
}		
else {		
a regular slice header …		
}		
}		

[0097] 表1:针对熵分片首部的示例性语法表

[0098] 在本发明的一些实施例中，可以向熵分片分配与常规分片不同的网络抽象层(NAL)单元类型。在这些实施例中，解码器可以基于NAL单元类型在常规分片与熵分片之间进行区分。在这些实施例中，不需要比特字段“entropy_slice_flag”。

[0099] 在本发明的一些实施例中，可以不在所有简档中传输比特字段“entropy_slice_flag”。在本发明的一些实施例中，可以不在基线简档中传输比特字段“entropy_slice_flag”，但是可以在较高简档(例如，主、扩展或专业简档)中传输比特字段“entropy_slice_flag”。在本发明的一些实施例中，可以仅在与比固定特性值大的特性相关联的比特流中传输比字段“entropy_slice_flag”。示例性特性可以包括空间分辨率、帧速率、比特深度、比特率和其他比特流特性。在本发明的一些实施例中，可以仅在与比交织1920x1080大的空间分辨率相关联的比特流中传输比特字段“entropy_slice_flag”。在本发明的一些实施例中，可以仅在比逐行1920x1080更大的空间分辨率相关联的比特流中传输比特字段“entropy_slice_flag”。在本发明的一些实施例中，如果没有传输比特字段“entropy_

slice_flag”,则可以使用缺省值。

[0100] 在本发明的一些实施例中,可以通过改变数据复用来构造熵分片。在本发明的一些实施例中,熵分片中包含的符号组可以在宏块级上复用。在本发明的备选实施例中,熵分片中包含的符号组可以在画面级上复用。在本发明的其他备选实施例中,熵分片中包含的符号中可以按照数据类型来复用。在本发明的又一备选实施例中,熵分片中包含的符号组可以按照以上的组合来复用。

[0101] 可以关于图8和图9理解包括基于画面级复用的熵分片构造的本发明的一些实施例。在图8中示出的本发明的一些实施例中,预测编码器194和残差编码器196可以分别对预测数据190和残差数据192进行熵编码,并且画面级复用器198可以在画面级对已编码的预测数据和已编码的残差数据进行复用。在本发明的一些实施例中,针对画面190的预测数据可以与第一熵分片相关联,并且针对画面192的残差数据可以与第二熵分片相关联。可以并行地解码已编码预测数据和已编码熵数据。在本发明的一些实施例中,可以将包括预测数据或残差数据的每个分区划分成可以被并行解码的熵分片。

[0102] 在图9中示出的本发明的一些实施例中,可以通过Y编码器206、U编码器208、和V编码器210分别对每个颜色平面的残差,例如亮度残差200和两个色度残差202、204,进行熵编码,并且可以通过画面级复用器212在画面级对熵编码的残差进行复用。在本发明的一些实施例中,针对画面200的亮度残差可以与第一熵分片相关联,针对画面202的第一色度残差可以与第二熵分片相关联,并且针对画面204的第二残差可以与第三熵分片相关联。可以并行地解码针对三个颜色平面的已编码残差数据。在本发明的一些实施例中,可以将包括颜色平面残差数据的每个分区划分成可以被并行解码的熵分片。在本发明的一些实施例中,亮度残差200与色度残差202、204相比可以具有相对更多的熵分片。

[0103] 在本发明的一些实施例中,可以对压缩的视频比特流进行代码转换以包括熵分片,从而允许上述本发明实施例所适应的并行熵解码。可以关于图10描述本发明的一些实施例。可以根据图10逐画面地处理没有熵分片的输入比特流。在本发明的这些实施例中,可以对来自输入比特流的画面进行熵解码220。可以获得已编码的数据,例如,模式数据、运动信息、残差信息和其他数据。可以根据数据一次一个地构造222熵分片。可以在新比特流中插入224与熵分片相对应的熵分片首部。可以重置编码器状态,并且定义226邻域信息。可以对熵分片进行熵编码228,并且写到新比特流。如果存在尚未被构造的熵分片消耗232的画面数据,则可以构造222另一熵分片,并且处理224-230继续,直到构造的熵分片消耗了所有画面数据,并且然后可以处理下个画面。

[0104] 在本发明的一些实施例中,编码器可以将重构分片划分成多个熵分片,其中,每个熵分片的大小可以小于或可以不超过固定数目的bin。在一些实施例中,编码器可以约束每个熵分片的大小,并且可以在比特流中通知bin的最大数目。在备选实施例中,编码器可以约束每个熵分片的大小,bin的最大数目可以由编码器的简档和级别符合点来限定。例如,可以扩展H.264/AVC视频编码规范的附录A,以包括对熵分片中允许的bin的最大数目的限定。

[0105] 在本发明的一些实施例中,可以根据表(例如,如表2所示)为编码器的每个级别符合点指定熵分片中允许的bin的最大数目,在表2中,$M_{m.n}$表示对于级别m.n符合点,熵分片中允许的bin的最大数目。

[0106]

级别	每个熵分片中bin的最大数目
1.1	$M_{1.1}$
1.2	$M_{1.2}$
:	:
m.n	$M_{m.n}$
:	:
5.1	$M_{5.1}$

[0107] 表2:对于每个级别每个熵分片的bin的最大数目

[0108] 熵分片中允许的bin的示例性最大数目是$M_{1.1}$=1,000个bin,$M_{1.2}$=2,000个bin,...,和$M_{5.1}$=40,000个bin。熵分片中允许的bin的其他示例性最大数目是$M_{1.1}$=2,500个bin,$M_{1.2}$=4,200个bin,…,和$M_{5.1}$=150,000个bin。

[0109] 在一些实施例中,可以基于比特率、图像大小、宏块数目和其他编码参数,针对所有级别确定熵分片中允许的bin的最大数目集合。在本发明的一些实施例中,对于所有级别可以将熵分片中允许的bin的最大数目设置为相同数目。示例性值是38,000个bin和120,000个bin。

[0110] 在本发明的一些实施例中,编码器可以确定与宏块相关联的bin最差情况数目,并且编码器可以向每个熵分片写入与$\frac{ESLICE_MaxNumberBins}{BinsPerMB}$个宏块相关联的bin,其中,ESLICE_MaxNumberBins可以表示熵分片中允许的bin的最大数目,BinsPerMB可以表示与宏块相关联的bin最差情况数目。在一些实施例中,可以按照光栅扫描顺序选择宏块。在备选实施例中,可以按照另一预定顺序选择宏块。在一些实施例中,与宏块相关联的bin最差情况数目可以是固定数目。在备选实施例中,编码器可以基于对先前处理的宏块的大小的测量来更新最差情况数目。

[0111] 可以关于图11描述本发明的一些实施例。在这些实施例中,编码器可以针对重构分片将重构分片划分成多个熵分片,其中熵分片的大小不可以大于预定数目的bin。编码器可以将与当前熵分片中bin的数目相关联的计数器设置为零。为了示意性目的,在关于图11描述的本发明的实施例的描述的其余部分中,计数器值可以表示为A。可以获得242下个宏块的语法元素。可以根据预定宏块处理顺序来确定下个宏块。在一些实施例中,宏块处理顺序可以对应于光栅扫描顺序。可以将宏块中的非二进制语法元素转换成224bin串。二进制语法元素不需要转换。可以确定246与宏块相关联的bin的数目。除了二进制语法元素以外,与宏块相关联的bin的数目可以包括与非二进制语法元素相关联的bin串中的bin,并且为了示意性目的,在关于图11描述的本发明实施例的说明的其余部分中,与宏块相关联的bin的数目可以表示为num。

[0112] 如果可以将与宏块相关联的bin的数目和与当前熵分片相关联的已累积的bin的数目相加,而没有249超过熵分片允许的bin的最大数目,则可以更新250与当前熵分片相关

联的累积bin的数目，以包括与宏块相关联的bin，并且与宏块相关联的bin可以由熵编码器写入252比特流，并与当前熵分片相关联。可以获得242下个宏块的语法元素，并且划分处理可以继续。

[0113] 如果248与宏块相关联的bin的数目和与当前熵分片相关联的累积的bin的数目之和超过253熵分片允许的bin的最大数目，则编码器可以开始254与当前重构分片相关联的新熵分片，并且可以终止当前熵分片。然后，可以将与新的当前熵分片中bin的数目相关联的计数器初始化256为零。可以更新250与当前熵分片相关联的累积bin的数目，以包括与宏块相关联的bin，并且与宏块相关联的bin可以通过熵编码器写入252比特流，并与当前熵分片相关联。可以获得242下个宏块的语法元素，并且划分处理可以继续。

[0114] 可以关于图12描述本发明的一些实施例。在这些实施例中，编码器对于重构分片可以将重构分片划分成多个熵分片，其中，熵分片的大小不大于预定最大数目的bin。在这些实施例中，编码器可以将宏块语法元素与熵分片相关联，直到熵分片的大小达到与熵分片中允许的bin的最大预定数目相关联的阈值为止。在一些实施例中，阈值可以是熵分片中允许的bin的最大数目的百分比。在一个示例性实施例中，假定宏块中期望的bin的最大数目比bin的最大数目小10％，阈值可以是熵分片中允许的bin的最大数目的90％。在另一示例性实施例中，阈值可以是熵分片中允许的bin的最大数目的百分比，其中百分比可以基于宏块中期望的bin的最大数目。在这些实施例中，一旦熵分片的大小超过阈值大小，则可以创建另一熵分片。可以选择阈值大小来确保熵分片不超过熵分片中允许的bin的最大数目。在一些实施例中，阈值大小可以依据熵分片中允许的bin的最大数目以及针对宏块预期的bin的最大数目的估计。

[0115] 编码器可以将与当前熵分片中bin的数目相关联的计数器初始化为零。为了示意性目的，在关于图12描述的本发明实施例的描述的其余部分中，计数器值可以表示为A。可以获得272下个宏块的语法元素。可以根据预定宏块处理顺序确定下个宏块。在一些实施例中，宏块处理顺序可以对应于光栅扫描顺序。可以将宏块中的非二进制语法元素转换274成bin串。二进制语法元素不需要转换。与宏块相关联的bin可以通过熵编码器写入到比特流中，并且与当前熵分片相关联。可以确定278与宏块相关联的bin的数目，并且可以更新280与当前熵分片相关联的累积bin的数目，以包括与宏块相关联的bin。如果282与当前熵分片相关联的累积bin的数目大于284基于熵分片中允许的bin的最大数目的阈值，阈值可以表示为TH(MaxNumBins)，则编码器可以开始286新熵分片，并且可以终止当前熵分片。然后编码器可以将与新的当前熵分片中的bin的数目相关联的计数器初始化为零。可以获得272下个宏块的语法元素，并且划分处理可以继续。如果与当前熵分片相关联的累积bin的数目不大于283基于熵分片中允许的bin的最大数目的阈值，则可以获得272下个宏块的语法元素，并且划分处理可以继续。

[0116] 在本发明的一些实施例中，编码器可以在已经将预定数目的宏块分配到当前重构分片时终止当前重构分片并且开始新的重构分片。

[0117] 可以关于图13描述本发明的一些实施例。在这些实施例中，编码器可以在已经将预定数目的宏块分配到当前重构分片时终止当前重构分片并且开始新的重构分片。编码器可以将与当前重构分片中宏块的数目相关联的计数器初始化300为零。为了示意性目的，在关于图13描述的本发明实施例的描述的其余部分中，计数器值可以表示为AMB。编码器可以

将与当前熵分片中bin的数目相关联的计数器初始化为零。为了示意性目的，在关于图13描述的本发明实施例的描述的其余部分中，计数器值可以表示为ABin。如果312与当前重构分片中的宏块的数目相关联的计数器值不小于331重构分片中允许的宏块的预定最大数目，则可以开始332新的熵分片，并且可以开始334新的重构分片，终止当前重构分片和当前熵分片。为了示意性目的，在关于图13描述的本发明实施例的描述的其余部分中，重构分片中允许的宏块的最大数目可以表示为MaxMBperRSlice。

[0118] 如果与当前重构分片中宏块的数目相关联的计数器的计数器值小于313重构分片中允许的宏块的预定最大数目，则可以获得314下个宏块的语法元素。可以根据预定的宏块处理顺序来确定下个宏块。在一些实施例中，宏块处理顺序可以对应于光栅扫描顺序。可以将宏块中的非二进制语法元素转换成316bin串。二进制语法元素不需要转换。可以确定318与宏块相关联的bin的数目。除了二进制语法元素以外，与宏块相关联的bin的数目还可以包括与非二进制语法元素相关联的bin串中的bin，并且为了示意性目的，在关于图13描述的本发明实施例的描述的其余部分中，与宏块相关联的bin的数目可以表示为num。

[0119] 如果与宏块相关联的bin的数目可以和与当前熵分片相关联的累积bin的数目相加，而不超过321熵分片允许的bin的最大数目，则可以更新322与当前熵分片相关联的累积bin的数目，以包括与宏块相关联的bin，与宏块相关联的bin可以通过熵编码器写入324比特流，并且与当前熵分片相关联，并且可以递增326与当前重构分片相关联的宏块的数目。可以将与当前重构分片相关联的宏块的数目与重构分片中允许的宏块的预定最大数目相比较，并且划分处理可以继续。

[0120] 如果320与宏块相关联的bin的数目和与当前熵分片相关联的累积bin的数目之和超过327熵分片允许的bin的最大数目，则编码器可以开始328与当前重构分片相关联的新的当前熵分片，并且将与当前熵分片中bin的数目相关联的计数器初始化330为零。可以更新322与当前熵分片相关联的累积bin的数目，以包括与宏块相关联的bin，与宏块相关联的bin可以通过熵编码器写入324比特流，并且与当前熵分片相关联，并且可以递增326与当前重构分片相关联的宏块的数目。可以将与当前重构分片相关联的宏块的数目与重构分片中允许的宏块的预定最大数目相比较312，并且划分处理可以继续。

[0121] 可以关于图14描述本发明的一些实施例。在这些实施例中，编码器可以在已经将宏块的预定数目分配到当前重构分片时开始新的重构分片。在这些实施例中，编码器可以将宏块语法元素与熵分片相关联，直到熵分片的大小达到与熵分片中允许的bin的预定最大数目相关联的阈值。在一些实施例中，阈值可以是熵分片中允许的bin的最大数目的百分比。在一个示例性实施例中，假定宏块中期望的bin的最大数目比bin的最大数目小10％，阈值可以是熵分片中允许的bin的最大数目的90％。在另一示例性实施例中，阈值可以是熵分片中允许的bin的最大数目的百分比，其中，百分比可以基于宏块中期望的bin的最大数目。在这些实施例中，一旦熵分片的大小超过阈值大小，则可以创建另一熵分片。可以选择阈值大小，以确保熵分片不会超过熵分片中允许的bin的最大数目。在一些实施例中，阈值大小可以依据熵分片中允许的bin的最大数目和针对宏块期望的bin的最大数目的估计。

[0122] 编码器可以将与当前重构分片中宏块的数目相关联的计数器初始化350为零。为了示意性目的，在关于图14描述的本发明实施例的描述的其余部分中，计数器值可以表示为AMB。编码器可以将与当前熵分片中bin的数目相关联的计数器初始化352为零。为了示意

性目的，在关于图14描述的本发明实施例的描述的其余部分中，计数器值可以表示为ABin。如果与当前重构分片中宏块的数目相关联的计数器的计数器值不小于373重构分片中允许的宏块的预定最大数目，则开始374新的熵分片，并且可以开始376新的重构分片。为了示意性目的，在关于图14描述的本发明实施例的描述的其余部分中，重构分片中允许的宏块的最大数目可以表示为MaxMBperRSlice。

[0123] 如果与当前重构分片中宏块的数目相关联计数器的计数器值小于355重构分片中允许的宏块的预定最大数目，则可以获得356下个宏块的语法元素。可以根据预定的宏块处理顺序确定下个宏块。在一些实施例中，宏块处理顺序可以对应于光栅扫描顺序。可以将宏块中的非二进制语法元素转换成358bin串。二进制语法元素不需要转换。与宏块相关联的bin可以通过熵编码器写入360比特流，并且与当前熵分片相关联。可以确定362与宏块相关联的bin的数目，并且可以更新364与当前熵分片相关联的累积bin的数目，以包括与宏块相关联的bin。如果366与当前熵分片相关联的累积bin的数目大于369基于熵分片中允许的bin的最大数目的阈值，阈值可以表示为TH(MaxNumBins)，则编码器可以开始370新熵分片，并且将与当前熵分片中的bin的数目相关联计数器初始化372为零。可以递增368与当前重构分片相关联的宏块的数目。可以将与当前重构分片相关联的宏块的数目与重构分片中允许的宏块的预定最大数目相比较354，并且划分处理可以继续。如果与当前熵分片相关联的累积bin的数目不大于367基于熵分片中允许的bin的最大数目的阈值，则可以递增368与当前重构分片相关联的宏块的数目，并且可以将与当前重构分片相关联的宏块的数目与重构分片中允许的宏块的预定最大数目相比较354，并且划分处理可以继续。

[0124] 在本发明的备选实施例中，编码器可以将重构分片划分成多个熵分片，其中每个熵分片可以与不多于预定数目的比特相关联。

[0125] 可以关于图15描述本发明的一些实施例。在这些实施例中，编码器对于重构分片可以将重构分片划分成多个熵分片，其中，熵分片的大小不大于预定数目的比特。编码器可以将与当前熵分片中的比特数目相关联的计数器初始化400为零。为了示意性目的，在关于图15描述的本发明实施例的描述的其余部分中，计数器值可以表示为A。可以获得402下个宏块的语法元素。可以根据预定宏块处理顺序确定下个宏块。在一些实施例中，宏块处理顺序可以对应于光栅扫描顺序。可以将宏块中的非二进制语法元素转换404成bin串。二进制语法元素不需要转换。可以向熵编码器提供与宏块相关联的bin、经转换的非二进制元素和二进制元素，并且可以对bin进行熵编码406。可以确定408与宏块相关联的比特数目。为了示意性目的，在关于图15描述的本发明实施例的描述的其余部分中，与宏块相关联的比特数目可以表示为num。

[0126] 如果将与宏块相关联的比特数目和与当前熵分片相关联的累积比特的数目相加410，而没有超过411熵分片允许的最大比特数目，则可以更新412与当前熵分片相关联的累积比特数目，以包括与宏块相关联的比特数目，并且可以将与宏块相关联的比特写入比特流，并且与当前熵分片相关联。获得402下个宏块的语法元素，并且划分处理可以继续。

[0127] 如果410与宏块相关联的比特的数目和与当前熵分片相关联的累积比特的数目之和超过415熵分片允许的比特的最大数目，则编码器可以开始416与当前重构分片相关联的新熵分片，可以将与当前熵分片中的比特的数目相关联的计数器初始化418为零。可以更新412与当前熵分片相关联的累积比特的数目，以包括与宏块相关联的比特，并且可以将与宏

块相关联的比特写入414比特流，并与当前熵分片相关联。可以获得402下个宏块的语法元素，并且划分处理可以继续。

[0128] 可以关于图16描述本发明的一些实施例。在这些实施例中，编码器对于重构分片可以将重构分片划分成多个熵分片，其中，熵分片的大小不大于预定最大数目的比特。在这些实施例中，编码器可以将宏块语法元素与熵分片相关联，直到熵分片的大小达到与熵分片中允许的比特的最大预定数目相关联的阈值为止。在一些实施例中，阈值可以是熵分片中允许的比特的最大数目的百分比。在一个示例性实施例中，假定宏块中期望的比特的最大数目比比特的最大数目小10％，阈值可以是熵分片中允许的比特的最大数目的90％。在另一示例性实施例中，阈值可以是熵分片中允许的比特的最大数目的百分比，其中百分比可以基于宏块中期望的比特的最大数目。在这些实施例中，一旦熵分片的大小超过阈值大小，则可以创建另一熵分片。可以选择阈值大小来确保熵分片不超过熵分片中允许的比特的最大数目。在一些实施例中，阈值大小可以依据熵分片中允许的比特的最大数目以及针对宏块预期的比特的最大数目的估计。

[0129] 编码器可以将与当前熵分片中比特的数目相关联的计数器初始化为零。为了示意性目的，在关于图16描述的本发明实施例的描述的其余部分中，计数器值可以表示为A。可以获得442下个宏块的语法元素。可以根据预定宏块处理顺序确定下个宏块。在一些实施例中，宏块处理顺序可以对应于光栅扫描顺序。可以将宏块中的非二进制语法元素转换成444bin串。二进制语法元素不需要转换。可以对与宏块相对应的bin进行熵编码446，并且可以确定448与宏块相关联的bin的数目，并且可以更新450与当前熵分片相关联的累积bin的数目，以包括与宏块相关联的bin。并且可以将与宏块相关联的熵编码的bin写入452比特流。如果454与当前熵分片相关联的累积比特的数目大于456基于熵分片中允许的比特的最大数目的阈值，则编码器可以开始458新熵分片，并且可以将与当前熵分片中的比特的数目相关联的计数器初始化为零。可以获得442下个宏块的语法元素，并且划分处理可以继续。如果与当前熵分片相关联的累积比特的数目不大于455基于熵分片中允许的比特的最大数目的阈值，则可以获得442下个宏块的语法元素，并且划分处理可以继续。

[0130] 在本发明的备选实施例中，编码器可以将重构分片划分成多个熵分片，其中每个熵分片可以与不多于预定数目的宏块相关联。

[0131] 在本发明的一些实施例帧，除了对熵分片大小的约束以外，还可以施加对重构分片中宏块的最大数目的约束。

[0132] 在本发明的一些实施例中，编码器可以将重构分片划分成多个熵分片，其中每个熵分片的大小可以被约束为小于预定数目的宏块，并且小于预定数目的bin。

[0133] 在本发明的一些实施例中，编码器可以将重构分片划分成多个熵分片，其中每个熵分片的大小可以被约束为小于预定数目的宏块，并且小于预定数目的比特。

[0134] 在本发明的一些实施例中，编码器可以将重构分片划分成多个熵分片，其中每个熵分片的大小可以被约束为小于预定数目的宏块，小于预定数目的bin，并且小于预定数目的比特。

[0135] 在本发明的一些实施例帧，熵编码器内的bin编码可以并行化，允许多于一个bin的并行编码，这可以减少编码时间。可以关于图17中示出的示例性熵编码器理解本发明的这些实施例。在这些实施例中，熵编码器480可以包括上下文适应单元482、基于状态的bin

编码器选择器484以及多个bin编码器，也被视为可以并行操作的bin编码器单元486、488、500(示出了三个)。可以从二进制化器504使bin 502对于熵编码器480可用，二进制化器504可以根据输入的符号506产生bin 502。使bin 502可用于上下文适应单元482以及基于状态的bin编码器选择器484。上下文适应单元482可以执行上下文适应并且产生模型状态(也被称作状态)508，模型状态508可以用于在bin编码器486、488、500之中选择bin502所指向的bin编码器。基于状态的bin编码器选择器484可以在bin编码器486、488、500之中选择与产生的模型状态508相关联的bin编码器，来编码bin 502。在一些实施例(未示出)中，使产生的状态508可用于所选的bin编码器。输出比特510、512、514可以由bin编码器486、488、500产生，并且输出比特510、512、514可以合并成比特流。在本发明的一些实施例中，可以对输出比特510、512、514进行缓冲并且通过级联合并成比特流。在备选实施例中，可以对输出比特510、512、514进行缓冲，并且根据交织方案合并成比特流。

[0136] 关于图17描述本发明的实施例，可以响应于关于第一bin产生的第一模型状态向第一bin编码器发送第一bin。上下文适应单元482在完成对第一bin的处理时可以开始处理第二bin，响应于关于第二bin产生的第二模型状态向第二bin编码器发送第二bin，从而允许实质上并行地处理多于一个bin。

[0137] 在本发明的备选实施例中，熵编码器可以包括可以并行操作的多个上下文适应单元，以及单个bin编码器。在上下文适应单元需要比bin编码器更长处理时间的系统中，多个上下文适应单元并行操作可以减少编码时间。可以关于图18中示出的示例性熵编码器理解本发明的这些实施例中的一些。在这些实施例中，熵编码器530可以包括多个(示出了三个)上下文适应单元532、534、536，上下文适应单元选择器535，状态选择器540和bin编码器542。可以从二进制化器546使bin544对于熵编码器530可用，二进制化器546可以根据输入的符号548产生bin 544。可以使bin 544可用于上下文适应单元选择器538、状态选择器540和bin编码器542。上下文适应单元选择器538可以用于选择或调度bin 544所指向的并且从中可以产生状态值550、552、554的上下文适应单元532、534、536。在一些示例性实施例中，上下文适应单元选择器538可以基于与bin相关联的语法在上下文适应单元532、532、536之中选择上下文适应单元，例如，上下文适应单元标识符可以与对bin所指向以被处理的上下文适应单元加以标识的bin相关联。在备选实施例中，上下文适应单元选择器538可以基于与上下文适应单元532、534、536相关联的调度协议或负载平衡约束来在上下文适应单元532、534、536之中选择上下文适应单元。在一些实施例中，可以根据在上下文适应单元选择器538处使用的准则，在适当定时处通过状态选择器540选择产生的状态值以传递给bin编码器542。bin编码器542可以使用状态选择器540传递的状态值556来编码bin 544。在本发明的备选实施例(未示出)中，bin编码器可以不需要状态值，因此使状态值对于bin编码器不可用。输出比特588可以由bin编码器542产生，并且输出比特588可以合并成比特流。在本发明的一些实施例中，可以对输出比特588进行缓冲并且通过级联合并成比特流。在备选实施例中，可以对输出比特588进行缓冲，并且根据交织方案合并成比特流。

[0138] 在本发明又一些备选实施例中，熵编码器可以包括可以并行操作的多个上下文适应单元以及可以并行操作的多个bin编码器。可以关于图19中示出的示例性熵编码器理解本发明的这些实施例。在这些实施例中，熵编码器570可以包括多个(示出了三个)上下文适应单元572、574、576，上下文适应单元选择器578，状态选择器580、基于状态的bin编码器选

择器582，以及多个(示出了三个)bin编码器584、856、588。可以从二进制化器592使bin 590对于熵编码器570可用，二进制化器592可以根据输入的符号594产生bin 590。可以使bin 590可用于上下文适应单元选择器578、状态选择器580和bin编码器选择器582。上下文适应单元选择器578可以用于选择或调度bin 590所指向的并且从中可以产生状态值596、598、600的上下文适应单元572、574、576。在适当定时处通过状态选择器580选择产生的状态值以传递给基于状态的bin编码器选择器582。基于状态的bin编码器选择器582可以使用状态选择器580传递的状态值602来在bin编码器584、586、588之中选择bin 590所指向的bin编码器。在备选实施例(未示出)中，可以使状态值602对于所选的bin编码器可用。所选的bin编码器可以使用状态值602来编码bin 590。在本发明的备选实施例(未示出)中，bin编码器可以不需要状态值，因此可以使状态值对于bin编码器不可用。输出比特604、606、608可以由bin编码器584、586、588产生，并且输出比特604、606、608可以合并成比特流。在本发明的一些实施例中，可以对输出比特604、606、608进行缓冲并且通过级联合并成比特流。在备选实施例中，可以对输出比特604、606、608进行缓冲，并且根据交织方案合并成比特流。

[0139] 本发明的示例性实施例可以包括可以并行操作的多个可变长度编码编解码器。

[0140] 在本发明的一个示例性实施例中，bin编码器可以包括二进制算术编码。在本发明的另一示例性实施例中，bin编码器可以包括可变长度编码。在本发明的又一示例性实施例中，bin编码器可以包括固定长度编码。

[0141] 通常，熵编码器可以包括N_{ca}个上下文适应单元和N_{bc}个bin编码器单元，其中，N_{ca}是大于或等于1的整数，并且N_{bc}是大于或等于1的整数。

[0142] 在本发明的一些实施例中，编码器可以将重构分片划分成多个熵分片，其中，可以约束每个熵分片的大小，使得在处理熵分片期间N_{ca}个上下文适应单元和N_{bc}个bin编码器单元中的一个或更多个可以各自对不多于限制数目的bin进行操作。具有这种约束的上下文适应单元和bin编码器单元可以被称作受约束的熵编码器单元。

[0143] 在本发明的一些实施例中，编码器可以将重构分片划分成多个熵分片，其中，可以约束每个熵分片的大小，使得在处理熵分片期间N_{ca}个上下文适应单元之中没有任何上下文适应单元可以对多于B_{ca}个bin进行操作。在本发明的一些实施例中，例如，可以在比特流、简档约束、级别约束或其他标准化机制中通知B_{ca}的值。

[0144] 在本发明的备选实施例中，编码器可以将重构分片划分成多个熵分片，其中，可以约束每个熵分片的大小，使得在处理熵分片期间N_{bc}个bin编码器单元之中没有任何bin编码器单元可以对多于B_{bc}个bin进行操作。在本发明的一些实施例中，例如，可以在比特流、简档约束、级别约束或其他规范性机制中发信号通知B_{bc}的值。

[0145] 在本发明的又一些备选实施例中，编码器可以将重构分片划分成多个熵分片，其中，可以约束每个熵分片的大小，使得在处理熵分片期间N_{ca}个上下文适应单元之中没有任何上下文适应单元可以对多于B_{ca}个bin进行操作，并且N_{bc}个bin编码器单元之中没有任何bin编码器单元可以对多于B_{bc}个bin进行操作。在本发明的一些实施例中，例如，可以在比特流、简档约束、级别约束或其他规范性机制中通知B_{ca}的值和B_{bc}的值。

[0146] 在本发明的另一些备选实施例中，编码器可以将重构分片划分成多个熵分片，其中，可以约束每个熵分片的大小，使得在处理熵分片期间第i个N_{ca}上下文适应单元(表示为$N_{ca}(i)$，$i=1,\cdots,N_{ca}$)对不多于$B_{ca}(i)$个bin进行操作，并且第i个N_{bc}bin编码器单元(表示为

$N_{bc}(i)$,i=1,…,N_{bc})可以对不多于$B_{bc}(i)$个bin进行操作。在本发明的一些实施例中,例如,可以在比特流、简档约束、级别约束或其他规范性机制中通知$B_{ca}(i)$的值和$B_{bc}(i)$的值。

[0147] 可以关于图20描述本发明的一些示例性实施例。在这些实施例中,编码器对于重构分片可以将重构分片划分成多个熵分片,其中,可以约束每个熵分片的大小,使得N_{ca}个上下文适应单元中的一个或更多个以及N_{bc}个bin编码器单元中的一个或更多个可以对不多于限制数目的bin进行操作。编码器可以针对每个受约束的熵编码器单元,将与当前熵分片中处理的bin的数目相关联的计数器设置为零。为了示意性目的,在关于图21描述的本发明实施例的描述的其余部分中,计数器值可以表示为A,其中A表示向量,向量中的每一项对应于针对当前熵分片受约束的熵编码器单元所处理的bin的累加数目。可以获得652下个宏块的语法元素。可以根据预定宏块处理顺序来确定下个宏块。在一些实施例中,宏块处理顺序可以对应于光栅扫描顺序。可以将宏块中的非二进制语法元素转换成654bin串。二进制语法元素可以不需要转换。可以确定656由每个受约束的熵编码器单元处理的、与宏块相关联的bin的数目。除了二进制语法元素以外,与宏块相关联的bin的数目还可以包括与非二进制语法元素相关联的bin串中的bin,并且为了示意性目的,在关于图20描述的本发明实施例的说明的其余部分中,由每个受约束的熵编码器单元处理的、与宏块相关联的bin的数目可以表示为num,其中,num表示向量,向量中的每一项对应于针对当前熵分片,受约束的熵编码器单元所处理的bin的数目。

[0148] 如果将与针对每个受约束的熵编码器单元的宏块相关联的bin的数目和针对每个受约束的熵编码器单元与当前熵分片相关联的累积bin的数目相加658,而不会超过659任何受约束的熵编码器单元允许的bin的最大数目,则可以更新660与当前熵分片相关联的累积bin的数目,以包括与宏块相关联的bin,并且与宏块相关联的bin可以由熵编码器写入662比特流,并与当前熵分片相关联。可以获得652下个宏块的语法元素,并且划分处理可以继续。

[0149] 如果658与宏块相关联的bin的数目和与当前熵分片相关联的累积bin的数目之和超过663任何受约束的熵编码器单元允许的bin的最大数目,则编码器可以开始664与当前重构分片相关联的新熵分片,并且可以将与当前熵分片中的bin的数目相关联的计数器初始化666为零。可以更新660与当前熵分片相关联的累积bin的数目,以包括与宏块相关联的bin,并且与宏块相关联的bin可以通过熵编码器写入662比特流,并与当前熵分片相关联。可以获得652下个宏块的语法元素,并且划分处理可以继续。

[0150] 可以关于图21描述本发明的一些实施例。在这些实施例中,编码器对于重构分片可以将重构分片划分成多个熵分片,其中,使得在处理熵分片期间N_{ca}个上下文适应单元和N_{bc}个bin编码器单元中的一个或更多个可以对不多于限制数目的bin进行操作。具有这种约束的上下文适应单元和bin编码器单元可以被称作受约束的熵编码器单元。编码器可以针对每个受约束的熵编码器单元,将与当前熵分片中由受约束的熵编码器单元处理的bin的数目相关联的计数器初始化700为零。为了示意性目的,在关于图21描述的本发明实施例的描述的其余部分中,计数器值可以表示为A,其中A表示向量,向量中的每一项对应于针对当前熵分片,受约束的熵编码器单元所处理的bin的累加数目。在这些实施例中,编码器可以将宏块语法元素与熵分片相关联,直到熵分片的大小达到与熵分片中允许被受约束的熵编码器单元处理的bin的最大预定数目相关联的阈值为止。在一些实施例中,阈值可以是熵分

片中允许被受约束的熵编码器单元处理的bin的最大数目的百分比。在一个示例性实施例中，假定宏块中期望的要被受约束的熵编码器单元处理的bin的最大数目比bin的最大数目小10％，阈值可以是熵分片中允许被受约束的熵编码器单元处理的bin的最大数目的90％。在另一示例性实施例中，阈值可以是熵分片中允许被受约束的熵编码器单元处理的bin的最大数目的百分比，其中百分比可以基于宏块中期望的要被受约束的熵编码器单元处理的bin的最大数目。在这些实施例中，一旦熵分片的大小超过阈值大小，则可以创建另一熵分片。可以选择阈值大小来确保熵分片不超过熵分片中允许被受约束的熵编码器单元处理的bin的最大数目。在一些实施例中，阈值大小可以依据熵分片中允许的bin的最大数目以及针对宏块预期的bin的最大数目的估计。

[0151] 可以获得702下个宏块的语法元素。可以根据预定宏块处理顺序来确定下个宏块。在一些实施例中，宏块处理顺序可以对应于光栅扫描顺序。可以将宏块中的非二进制语法元素转换成704bin串。二进制语法元素可以不需要转换。与宏块相关联的bin可以由熵编码器写入706比特流，并与当前熵分片相关联。可以确定708由每个受约束的熵编码器单元处理的与宏块相关联的bin的数目。除了二进制语法元素以外，与宏块相关联的bin的数目还可以包括与非二进制语法元素相关联的bin串中的bin，并且为了示意性目的，在关于图21描述的本发明实施例的说明的其余部分中，由每个受约束的熵编码器单元处理的、与宏块相关联的bin的数目可以表示为num，其中，num表示向量，向量中的每一项对应于针对当前熵分片受约束的熵编码器单元所处理的bin的数目。可以更新710由受约束的熵编码器单元处理的与当前熵分片相关联的累积bin的数目，以包括与宏块相关联的bin。如果712由受约束的熵编码器单元处理的与当前熵分片相关联的累积bin的数目大于714阈值，阈值可以表示为针对受约束的熵编码器单元的TH(MaxNumBins)(i)，则编码器可以开始716新熵分片，并且可以将与当前熵分片中由每个受约束的熵编码器单元处理的bin的数目相关联的计数器初始化718为零。可以获得702下个宏块的语法元素，并且划分处理可以继续。如果由受约束的熵编码器单元的处理的与当前熵分片相关联的累积bin的数目不大于713阈值，则可以获得702下个宏块的语法元素，并且划分处理可以继续。

[0152] 本发明的一些实施例可以包括用于熵分片划分的上述准则的组合。

[0153] 应当理解，尽管本发明的一些实施例可以将熵分片的大小约束到小于第一预定大小，但是可以同样将熵分片的大小约束到不超过第二预定大小。本文描述的实施例是本发明的示例性实施例，并且本领域普通技术人员应当认识到，存在用于约束熵分片的大小的本发明的等同实施例。

[0154] 在本发明的一些实施例中，开始新熵分片可以包括终止当前分片，并且将新熵分片视为当前熵分片。

[0155] 在本发明的一些实施例中，可以在包括多个bin解码器的熵解码器内并行地对熵分片内的多个比特进行解码，这可以减少解码时间。可以关于图22中示出的示例性熵解码器750理解本发明的示例性实施例，示例性熵解码器750包括多个(示出了三个)bin解码器762、764、766。可以使熵分片内的比特752以及先前已解码符号754对于熵解码器750可用。可以使比特752对于bin解码器选择器756可用，bin解码器选择器756可以基于从上下文适应单元760产生的上下文状态758在bin解码器762、764、766之中选择bin解码器。上下文适应单元760可以基于对于上下文适应单元760可用的先前已解码符号754来产生上下文状态

758。bin解码器选择器756可以基于上下文状态758来指定bin解码器762、764、766。要解码的比特752可以由bin解码器选择器756传递到所选的bin解码器。bin解码器762、764、766产生可以由复用器774复用的已解码bin 768、770、772，并且向可以产生与bin 776相关联的符号754的符号化器778发送经复用的bin 776。

[0156] 在本发明的一些实施例中，可以在包括多个上下文适应单元的熵解码器内并行地对熵分片内的多个比特进行解码，这可以减少解码时间。可以关于图23中示出的示例性熵解码器800理解本发明的示例性实施例，示例性熵解码器800包括多个(示出了三个)上下文适应单元814、816、818。使熵分片内的比特802以及先前已解码符号810对于熵解码器800可用。可以使比特802对于上下文适应单元选择器812可用，上下文适应单元选择器812可以从多个上下文适应单元814、816、816之中选择用于输入比特的解码处理的上下文适应单元。在本发明的一些实施例中，上下文适应单元选择器812可以在接收每第N个比特时选择第N个上下文适应单元。所选的上下文适应单元可以基于对于所选上下文适应单元可用的先前已解码符号810来产生上下文状态820、822、824。状态选择器826在适当定时处可以选择与输入比特相关联的产生的上下文状态。在本发明的一些实施例中，状态选择器826可以根据与上下文适应单元选择器812相同的过程在接收每第N比特时选择第N个上下文适应单元。可以使所选状态828对于bin解码器804可用。bin解码器804可以对比特802进行解码，并且向可以产生与已解码bin 806相关联的符号810的符号化器808发送已解码bin 806。

[0157] 在本发明的一些实施例中，可以在包括多个上下文适应单元和多个bin解码器的熵解码器内并行地对熵分片内的多个比特进行解码，这可以减少解码时间。可以关于图24中示出的示例性熵解码器850理解本发明的示例性实施例，示例性熵解码器850包括多个(示出了三个)上下文适应单元852、854、856以及多个(示出了三个)bin解码器858、860、862。可以使熵分片内的比特864以及先前已解码符号866对于熵解码器850可用。可以使比特864对于上下文适应单元选择器868可用，上下文适应单元选择器868可以从多个上下文适应单元852、854、856之中选择用于输入比特的解码处理的上下文适应单元。在本发明的一些实施例中，上下文适应单元选择器868可以在接收每第N个比特时选择第N个上下文适应单元。所选的上下文适应单元可以基于对于所选上下文适应单元可用的先前已解码符号866来产生上下文状态870、872、874。状态选择器876在适当定时处可以选择与输入比特相关联的产生的上下文状态。在本发明的一些实施例中，状态选择器876可以根据与上下文适应单元选择器868相同的过程在接收每第N比特时选择第N个上下文适应单元。可以使所选状态878对于bin解码器选择器880可用。bin解码器选择器880可以基于所选上下文状态878选择bin解码器858、860、862。bin解码器选择器880可以基于上下文状态878指定bin解码器858、860、862。要解码的比特864可以通过bin解码器选择器880传递到所选bin解码器。bin解码器858、860、862可以产生已解码bin 882、884、886，已解码bin 882、884、886可以由复用器888复用，并且向可以产生与bin 890相关联的符号866的符号化器892发送经复用的bin 890。

[0158] 在本发明的一些实施例中，编码器可以将重构分片划分成多个熵分片，其中，熵分片内的宏块是连续的。图25示出了被划分成以下三个分片的示例性重构分片950：以十字影线952示出的熵分片0、以白色956示出的熵分片1以及以点影线956示出的熵分片2。在该示例性重构分片950中每个熵分片952、954、956内的宏块是连续的。

[0159] 在本发明的备选实施例中，编码器可以将重构分片划分成多个熵分片，其中，熵分片内的宏块是不连续的。图26示出了被划分成以下三个分片的示例性重构分片960：以十字影线962示出的熵分片0、以白色964示出的熵分片1以及以点影线966示出的熵分片2。在该示例性重构分片960中每个熵分片962、964、966内的宏块是不连续的。其中熵分片内的宏块不连续的重构分片的分区可以被称作交织分区。

[0160] 在本发明的一些实施例中，在对熵分片内的当前块进行熵解码期间，解码器可以使用来自相同熵分片中其他块来预测与当前块的熵解码有关的预测信息。在本发明的一些实施例中，在重构分片内的当前块的重构期间，来自相同重构分片中其他块可以用于预测与当前块的重构有关的预测信息。

[0161] 在重构分片包括交织分区的本发明的一些实施例中，在对熵分片内的当前块解码中使用的熵分片内的相邻块可以不直接相邻或连续。图27针对图26中示出的示例性交织分区示出了这种情况。

[0162] 在图27中，对于熵分片964内的当前块970，用于熵解码当前块970的左相邻块是熵分片964内连续的左相邻块972。用于熵解码当前块970的上相邻块是同一熵分片964内连续的上相邻块974。为了重构当前块970，左相邻块是重构分片960内连续的左相邻块972，上相邻块是重构分片960内连续的上相邻块976。

[0163] 在重构分片包括交织分区的本发明的一些实施例中，在熵分片中可能不存在要在解码熵分片内的当前块中使用的适当相邻块。图28针对图26中示出的示例性交织分区示出了这种情况。

[0164] 在图28中，对于熵分片964内的当前块980，在熵分片964内不存在要用于熵解码当前块980的左相邻块。用于熵解码当前块980的上相邻块是同一熵分片964内非连续的上相邻块982。为了重构当前块980，左相邻块是重构分片960内连续的左相邻块984，并且上相邻块是重构分片960内连续的上相邻块986。

[0165] 在本发明的一些实施例中，解码器可以对完整的输入比特流进行预处理，以识别熵分片的位置。在本发明的一些实施例中，解码器可以对整个重构分片进行预处理，以识别重构分片内熵分片的位置。在一些实施例中，可以通过识别熵分片首部的位置来确定熵分片的位置。在这些实施例中，解码器可以读取比特流中的比特，并且预定义的开始码值可以被识别。

[0166] 在备选实施例中，可以将熵分片首部约束到位于输入比特流内预定义位置处的比特范围中。在备选实施例中，可以将熵分片首部约束到位于输入比特流内预定义位置处的字节范围中。在这些实施例中，无论是比特对准的还是字节对准的，解码器都不需要预处理输入比特流的相当大部分来定位熵分片。

[0167] 在本发明的一些实施例中，编码器可以在比特流中通知可以约束熵分片首部的位置的熵分片位置信息，也被称作熵分片位置参数，例如，偏移和范围信息。在备选实施例中，熵分片位置信息可以不在比特流中通知，而是可以根据熵分片参数来确定，例如任何给定熵分片中允许的bin的固定数目、任何给定熵分片中允许的比特的固定数目以及其他熵分片参数。在本发明的另一些备选实施例中，熵分片位置信息可以由其他规范性手段来定义，例如，可以在简档约束、级别约束、应用约束或其他约束中指定该信息，或者该信息可以作为辅助信息被通知，或者由其他非限制手段来通知。

[0168] 在本发明的一些实施例中，一个熵分片位置参数值集合可以用于比特流内的所有熵分片。在备选实施例中，熵分片位置参数值可以针对一部分序列所表示的一组像素来定义。在备选实施例中，熵分片位置参数值可以针对比特流内的每个画面来定义，并且可以用于关联画面内的所有熵分片。在备选实施例中，熵分片位置参数值可以针对比特流内的每个重构分片来定义，并且用于关联重构分片内的所有熵分片。在又一些备选实施例中，解码器可以使用多个熵分片位置参数值集合。在又一些备选实施例中，可以向熵分片标识符分配熵分片位置参数值，例如，第一熵分片首部可以使用第一熵分片位置参数集合，第二熵分片首部可以使用第二熵分片位置参数集合，一般性的，第N个熵分片首部可以使用第N个熵分片位置参数集合。在本发明的一些实施例中，第一画面可以使用第一熵分片参数值集合，第二画面可以使用第二熵分片参数值集合，一般性的，第N个画面可以使用第N个熵分片位置参数集合。在另一示例性实施例中，第一类型的画面可以使用第一熵分片位置参数值集合，并且第二类型的画面可以使用第二熵分片位置参数值集合。画面的示例性类型是帧内画面、预测画面和其他类型的画面。

[0169] 在包括H.264/AVC编解码器的本发明的一些实施例中，可以通过向序列参数集(原始字节序列有效载荷(RBSP))添加“entropy_slice_offset”参数和“entropy_slice_range”来在该序列参数集中通知熵分片偏移和熵分片范围。表3列出了根据本发明实施例的示例性序列参数集RBSP语法。

[0170] 在包括H.264/AVC的本发明的一些实施例中，可以通过向画面参数集(原始字节序列有效载荷(RBSP))添加“entropy_slice_offset”参数和“entropy_slice_range”来在该画面参数集中通知熵分片偏移和熵分片范围。表4列出了根据本发明实施例的示例性画面参数集RBSP语法。

[0171] 在包括H.264/AVC的本发明的一些实施例中，可以通过向分片首部添加“entropy_slice_offset”参数和“entropy_slice_range”来在该分片首部中发信号通知熵分片偏移和熵分片范围。表5列出了根据本发明实施例的示例性分片首部语法。

[0172] 在本发明的一些实施例中，可以根据表(例如，如表6所示)为编码器的每个级别符合点指示熵分片偏移和熵分片范围，在表6中，$O_{m.n}$表示级别m.n符合点的熵分片偏移，$R_{m.n}$表示m.n符合点的熵分片范围。

[0173]

seq_parameter_set_rbsp() {	C	描述符
profile_idc	0	u(8)
reserved_zero_8bits /* 等于 0 */	0	u(8)
level_idc	0	u(8)
seq_parameter_set_id	0	ue(v)
bit_depth_luma_minus8	0	ue(v)
bit_depth_chroma_minus8	0	ue(v)
increased_bit_depth_luma	0	ue(v)
increased_bit_depth_chroma	0	ue(v)
log2_max_frame_num_minus4	0	ue(v)
log2_max_pic_order_cnt_lsb_minus4	0	ue(v)
max_num_ref_frames	0	ue(v)
gaps_in_frame_num_value_allowed_flag	0	u(1)
log2_min_coding_unit_size_minus3	0	ue(v)
max_coding_unit_hierarchy_depth	0	ue(v)
log2_min_transform_unit_size_minus2	0	ue(v)
max_transform_unit_hierarchy_depth	0	ue(v)
pic_width_in_luma_samples	0	u(16)
pic_height_in_luma_samples	0	u(16)
entropy_slice_offset	0	ue(v)
entropy_slice_range	0	ue(v)
rbsp_trailing_bits()	0	
}		

[0174] 表3:示例性序列参数集RBSP语法表

[0175]

pic_parameter_set_rbsp() {	C	描述符
pic_parameter_set_id	1	ue(v)
seq_parameter_set_id	1	ue(v)
entropy_coding_mode_flag	1	u(1)
num_ref_idx_l0_default_active_minus1	1	ue(v)
num_ref_idx_l1_default_active_minus1	1	ue(v)

[0176]

pic_init_qp_minus26 /* 相对于 26 */	1	se(v)
constrained_intra_pred_flag	1	u(1)
entropy_slice_offset	0	ue(v)
entropy_slice_range	0	ue(v)
rbsp_trailing_bits()	1	
}		

[0177] 表4:示例性画面参数集RBSP语法表

[0178]

slice_header() {	C	描述符
first_lctb_in_slice	2	ue(v)
entropy_slice_flag		u(1)
if (!entropy_slice_flag){		
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
frame_num	2	u(v)
if(IdrPicFlag)		
idr_pic_id	2	ue(v)
pic_order_cnt_lsb	2	u(v)
if(slice_type == P || slice_type == B) {		
num_ref_idx_active_override_flag	2	u(1)
if(num_ref_idx_active_override_flag) {		
num_ref_idx_l0_active_minus1	2	ue(v)
if(slice_type == B)		
num_ref_idx_l1_active_minus1	2	ue(v)
}		
}		
if(nal_ref_idc != 0)		
dec_ref_pic_marking()	2	
if(entropy_coding_mode_flag&& slice_type != I)		
cabac_init_idc	2	ue(v)
slice_qp_delta	2	se(v)
alf_param()		
if(slice_type == P || slice_type == B) {		
mc_interpolation_idc	2	ue(v)
mv_competition_flag	2	u(1)
if (mv_competition_flag) {		
mv_competition_temporal_flag	2	u(1)
}		

[0179]

}		
if (slice_type = = B && mv_competition_flag)		
collocated_from_l0_flag	2	u(1)
} else		
if (entropy_coding_mode_flag && slice_type != I)		
cabac_init_idc		ue(v)
entropy_slice_offset	0	ue(v)
entropy_slice_range	0	ue(v)
}		

[0180] 表5:针对分片首部的示例性分片表

[0181]

级别	熵分片偏移	熵分片范围
1.1	$O_{1.1}$	$R_{1.1}$
1.2	$O_{1.2}$	$R_{1.2}$
:	:	:
m.n	$O_{m.n}$	$R_{m.n}$
:	:	:
5.1	$O_{5.1}$	$R_{5.1}$

[0182] 表6:针对每个级别的示例性熵分片偏移和熵分片范围

[0183] 在一些实施例中,熵分片位置信息可以包括约束熵分片首部的位置的信息。在一个示例中,熵分片位置信息可以包括偏移值和范围值,偏移也被称作周期或基本偏移,范围也被称作针对周期的偏差或偏移。可以基于偏移值和范围值来约束熵分片首部位置。

[0184] 在本发明的一些实施例中,可以显式地定义偏移值和范围值。在本发明的备选实施例中,可以将偏移值和范围值隐式地定义为最小偏移值和最大偏移值。在本发明的另一些备选实施例中,可以将偏移值和范围值隐式地定义为最大偏移值、以及最大偏移值和最小偏移值之间的差。在本发明的另一些备选实施例中,可以将偏移值和范围值隐式地定义为最小偏移值、以及最小偏移值和最大偏移值之间的差。在备选实施例中,可以将偏移值和范围值隐式地定义为第三值、以及第三值与最大偏移值和最小偏移值之间的差。在又一些备选实施例中,可以通过索引将偏移值和范围值定义到查找表中,查找表包含对应的最小和最大比特值。在一些实施例中,可以使用基于偏移的查找树来定义偏移值和范围值。在一些实施例中,可以使用成本最小化编索引来定义偏移值和范围值。本领域普通技术人员应认识到,存在现有技术中已知的许多方法,来隐式地定义范围值和偏移值,并且确保编码器和解码器利用预定义的偏移和范围值的相同值来操作。

[0185] 在本发明的一些实施例中，通知范围值可以是可选的。在一些实施例中，当没有通知范围值时，那么可以将范围值设置为预定义值。在示例性实施例中，预定义值可以是零。在另一示例性实施例中，预定义值可以是非零整数值。

[0186] 在关于图29描述的示例性实施例中，可以将重构分片内与熵分片相关联的熵分片首部，分片号N约束为在重构分片首部的开始或者重构分片首部内的其他固定位置起Nk-p个比特之后开始，其中k表示偏移值，p表示范围。从其开始可以测量Nk-p个比特的位置可以被称作参考位置。在备选实施例中，参考位置可以与具体重构分片不相关联，并且可以是比特流内针对所有熵分片的相同固定位置。在备选实施例中，熵分片首部可以是字节对准的，并且约束可以与多个字节相关联。尽管在比特方面描述了关于图29示意的示例，但是本领域普通技术人员可以认识到备选的字节对准实施例。

[0187] 图29是示出了示例性比特流的示例性部分1000的图形表示。比特流部分1000包括由实黑矩形表示的重构分片首部1002、由实灰矩形表示的四个熵分片首部(与第0熵分片相对应的熵分片首部1003，被称作第0熵分片首部，与第一熵分片相对应的熵分片首部1004，被称作第一熵分片首部，与第二熵分片相对应的熵分片首部1005，被称作第二熵分片首部，以及与第三熵分片相对应的熵分片首部1006，被称作第三熵分片首部)、以及由细黑白条表示的熵分片的其余部分。在该示例中，参考位置可以是重构分片首部1002的开始1001。在本发明的一些实施例中，可以将与第0熵分片1003相对应的熵分片首部约束到位于紧接重构分片首部1002之后。在本发明的一些实施例中，与第0熵分片相对应的熵分片首部可以是重构分片首部的一部分。也就是说，重构分片首部也用作与第0熵分片相对应的熵分片首部。在这些实施例中，重构分片首部可以包括重构部分和熵部分。在图29中示出的本发明的一些实施例中，可以将第一熵分片首部1004约束到位于自参考位置1001起k-p个比特1007之后，可以将第二熵分片首部1005约束到位于自参考位置1001起2k-p个比特1008之后，将第三熵分片首部1006约束到位于自参考位置1001起3k-p个比特1009之后。在这些实施例中，分配来解码熵分片N的熵解码器可以在自参考位置1001起Nk-p个比特之后开始搜索对应的熵分片首部。

[0188] 在本发明的备选实施例中，熵分片位置信息可以不包括范围参数。在这些实施例中，熵解码器可以自参考位置起Nk个比特之后开始搜索第N熵分片首部。

[0189] 在关于图30描述的另一示例性实施例中，可以将与重构分片内熵分片(分片号N)相关联的熵分片首部约束到自重构分片首部的开始或者重构分片首部内的其他固定位置起Nk-p个比特之后开始，其中k表示偏移值，p表示范围，并且还可以将熵分片首部约束到在自受约束开始位置2p比特范围内。从其开始测量Nk-p个比特的位置可以被称作参考位置。在备选实施例中，参考位置可以与具体重构分片不相关联，并且可以是比特流内针对所有熵分片的相同固定位置。在备选实施例中，熵分片首部可以是字节对准的，并且约束可以与多个字节相关联。尽管参照比特描述了关于图30示意的示例，但是本领域普通技术人员可以认识到备选的字节对准实施例。

[0190] 图30是示例性比特流的示例性部分1020的图形表示。比特流部分1020包括由实黑矩形表示的重构分片首部1022、由实灰矩形表示的四个熵分片首部(与第0熵分片相对应的熵分片首部1023，被称作第0熵分片首部，与第一熵分片相对应的熵分片首部1024，被称作第一熵分片首部，与第二熵分片相对应的熵分片首部1025，被称作第二熵分片首部，以及与

第三熵分片相对应的熵分片首部1026，被称作第三熵分片首部)、以及由细黑白条表示的熵分片的其余部分。在该示例中，参考位置可以是重构分片首部1022的开始1021。在本发明的一些实施例中，可以将与第0熵分片1023相对应的熵分片首部约束到位于紧接重构分片首部1022之后。在本发明的一些实施例中，与第0熵分片相对应的熵分片首部可以是重构分片首部的一部分。在这些实施例中，重构分片首部可以包括重构部分和熵部分。在图30中示出的本发明的一些实施例中，可以将第一熵分片首部1024约束到位于自参考位置1021起k-p个比特1027之后的2p个比特1031内，可以将第二熵分片首部1025约束到位于自参考位置1021起2k-p个比特1028之后的2p个比特1032内，将第三熵分片首部1026约束到位于自参考位置1021起3k-p个比特1029之后的2p个比特1033内。在这些实施例中，分配为解码熵分片N的熵解码器可以在自参考位置1021起Nk-p个比特之后开始搜索对应的熵分片首部，并且可以在识别熵分片首部之后或在搜索2p个比特之后终止搜索。

[0191] 可以关于图31描述本发明的一些实施例。在这些实施例中，熵解码器可以接收1050对当前重构块中的熵分片的数目加以指示的熵分片数目，来进行熵解码。熵解码器可以确定1052熵分片位置信息。在本发明的一些实施例中，可以在比特流中通知熵分片位置信息(也被称作熵分片位置参数)，并且解码器可以通过检查比特流来确定1052熵分片信息。在备选实施例中，可以在比特流中不通知熵分片位置信息，但是可以通过解码器根据熵分片参数(例如，任何给定熵分片中允许的bin的固定数目、任何给定熵分片中允许的比特的固定数目以及其他熵分片参数)来确定1052熵分片位置信息。在本发明的另一些备选实施例中，熵分片位置信息可以由其他规范性手段来定义和确定1052，例如，可以在简档约束、级别约束、应用约束或其他约束中指定该信息，或者信息可以作为辅助信息被通知，或者由其他非限制手段来通知。

[0192] 熵解码器可以计算1054比特流中的熵分片搜索开始位置，在该位置之前限制编码器写入熵分片首部。在本发明的一些实施例中，可以使用根据熵分片位置信息确定的偏移值和范围值来计算1054的熵分片搜索开始位置。在本发明的备选实施例中，可以使用根据熵分片位置信息确定的偏移值来计算1054熵分片搜索开始位置。熵解码器可以前进至1056比特流中的熵分片搜索开始位置，并且可以在比特流中检查1058熵分片首部。在本发明的一些实施例中，熵分片首部可以由开始码指示。

[0193] 可以关于图32描述本发明的一些实施例。在这些实施例中，熵解码器可以接收1070对当前重构块中的熵分片的数目加以指示的熵分片数目，来进行熵解码。熵解码器可以确定1072熵分片位置信息。在本发明的一些实施例中，可以在比特流中通知熵分片位置信息(也被称作熵分片位置参数)，并且解码器可以通过检查比特流来确定1072熵分片信息。在备选实施例中，可以在比特流中不通知熵分片位置信息，但是可以通过解码器根据熵分片参数(例如，任何给定熵分片中允许的bin的固定数目、任何给定熵分片中允许的比特的固定数目以及其他熵分片参数)来确定1072熵分片位置信息。在本发明的另一些备选实施例中，熵分片位置信息可以由其他规范性手段来定义和确定1072，例如，可以在简档约束、级别约束、应用约束或其他约束中指定该信息，或者该信息可以作为辅助信息被通知，或者由其他非限制手段来通知。

[0194] 熵解码器可以计算1074比特流中的熵分片搜索开始位置，在该位置之前限制编码器写入熵分片首部。在本发明的一些实施例中，可以使用根据熵分片位置信息确定的偏移

值和范围值来计算1074的熵分片搜索开始位置。在本发明的备选实施例中，可以使用根据熵分片位置信息确定的偏移值来计算1074熵分片搜索开始位置。熵解码器可以前进至1076比特流中的熵分片搜索开始位置，并且可以在比特流中检查1078熵分片首部。在本发明的一些实施例中，熵分片首部可以由开始码指示。

[0195] 可以从所述熵分片搜索开始位置开始顺序地检查1078比特流中的比特。如果1080识别出1081的熵分片首部，则熵解码器可以对与识别出的熵分片首部相关联是熵分片进行熵解码1082。如果1080没有识别出1083熵分片首部，则熵解码器可以终止1084搜索。在一些实施例中，熵解码器可以在没有识别出1083的熵分片首部时指示错误。

[0196] 可以关于图33描述本发明的一些实施例。在这些实施例中，熵解码器可以接收1100对当前重构块中的熵分片的数目加以指示的熵分片数目，来进行熵解码。熵解码器可以确定1102熵分片位置信息。在本发明的一些实施例中，可以在比特流中通知熵分片位置信息(也被称作熵分片位置参数)，并且解码器可以通过检查比特流来确定1102熵分片信息。在备选实施例中，可以在比特流中不通知熵分片位置信息，但是可以通过解码器根据熵分片参数(例如，任何给定熵分片中允许的bin的固定数目、任何给定熵分片中允许的比特的固定数目以及其他熵分片参数)来确定1102熵分片位置信息。在本发明的另一些备选实施例中，熵分片位置信息可以由其他规范性手段来定义和确定1102，例如，可以在简档约束、级别约束、应用约束或其他约束中指定该信息，或者信息可以作为辅助信息被通知，或者由其他非限制手段来信号通知。

[0197] 熵解码器可以计算1104比特流中的熵分片搜索开始位置，在该位置之前限制编码器写入熵分片首部。在本发明的一些实施例中，可以使用根据熵分片位置信息确定的偏移值和范围值来计算1104的熵分片搜索开始位置。在本发明的备选实施例中，可以使用根据熵分片位置信息确定的偏移值来计算1104熵分片搜索开始位置。熵解码器可以前进至1106比特流中的熵分片搜索开始位置，并且可以在比特流中检查1078熵分片首部。在本发明的一些实施例中，熵分片首部可以由开始码指示。

[0198] 可以从所述熵分片搜索开始位置开始顺序地检查1108比特流中的比特。如果1110识别出1111熵分片首部，则熵解码器可以对与识别出的熵分片首部相关联是熵分片进行熵解码1112。如果1110没有识别出1113熵分片首部，则如果1114满足1115搜索准则，则熵解码器可以终止1116。搜索准则可以提供如下标准：按照该标准，确定是否留有要搜索的用于熵分片首部开始的有效位置。在一些实施例(未示出)中，如果留有要检查的有效位置，则可以满足搜索准则。在备选实施例中，如果无要检查1115的有效位置剩下，则可以满足搜索准则，并且搜索可以终止1116。在一些实施例中，熵解码器可以在没有识别出1115的熵分片首部时指示错误。如果1114不满足1117搜索准则，则对比特流的检查1108可以在前进至1118比特流中下个搜索位置后继续。

[0199] 在本发明的一些实施例中，搜索准则可以与范围值有关，例如，可以将熵分片首部的开始位置约束到以Nk为中心的2p个比特的范围，其中k表示偏移值，p表示范围值，N是重构分片内熵分片数目。在这些实施例中，可以将与熵分片N相关联的熵分片首部的开始位置约束到Nk-p到Nk+p的范围。在一些实施例中，搜索准则与针对熵分片大小的一个或更多个约束有关。在一些实施例中，搜索准则可以与约束的组合有关。

[0200] 在本发明的一些实施例中，编码器可以填充熵分片，以便满足对下个熵分片首部

的位置的约束。

[0201] 在本发明的一些实施例中，编码器可以在满足其他熵分片大小约束之前终止熵分片，以便满足对下个熵分片首部的位置的约束。

[0202] 在本发明的一些实施例中，当重构分片内的最末熵分片不包含满足对下个熵分片首部的位置的约束所必需的比特(或在字节对准实施例中，字节)数目时，编码器可以填充重构分片内的最末熵分片，以满足对下个熵分片首部的位置的约束。

[0203] 在备选实施例中，熵分片首部可以包括最末熵分片标志，其中最末熵分片标志可以指示与熵分片首部相关联的熵分片是否是重构分片中的最末熵分片。在一些实施例中，最末熵分片标志值0可以与最末熵分片相关联。在备选实施例中，最末熵分片标志值1可以与最末熵分片相关联。在一些实施例中，当最末熵分片标志的值指示熵分片是重构分片中的最末熵分片时，后续熵分片首部可以位于紧随当前熵分片之后，而无需填充。

[0204] 表7示出了用于通知最末熵分片标志(被称作“next_entropy_slice_flag”)的示例性语法和语义。在包括表7中示出的示例性语法和语义的示例性实施例中，“next_entropy_slice_flag”标志通知是否存在针对当前重构分片的附加熵分片。如果“next_entropy_slice_flag”标志指示不存在当前重构分片的附加熵分片，则比特流中下个熵分片首部的位置可以不由熵分片位置参数约束。

[0205] 在本发明的一些实施例中，可以按照树格式组织熵分片首部的位置，其中根节点指向熵分片首部位置。在一些实施例中，根节点所指向的熵分片首部位置可以是相对的。在备选实施例中，由根节点所指向的熵分片首部位置可以是绝对的。树的其余节点可以包含相对于它们父辈节点的偏移距离。例如，可以根据设计约束来设计树，以减少确定熵分片首部位置所需的平均时间，限制确定熵分片首部位置所需的最差情况时间，通知熵分片解码的优选顺序，最小化树和其他设计约束的存储成本。在一些实施例中，可以基于熵分片首部位置确定中期望的并行度级别来控制树中每个节点的子节点的数目。

[0206]

slice_header() {	C	描述符
entropy_slice_flag	2	u(1)
next_entropy _slice_flag	2	ue(v)
if (entropy_slice_flag) {		
first_mb_in_slice	2	ue(v)
if(entropy_coding_mode_flag && slice_type != I && slice_type != SI)		
cabac_init_idc	2	ue(v)
}		
}		
else {		
a regular slice header ……		
}		
}		

[0207] 表7:针对最末熵分片标志的示例性语法表

[0208] 在本发明的一些实施例中，只要满足上下文模型重置条件，就可以在熵分片内重置上下文模型。在这些实施例中的一些实施例中，重置上下文模型的值可以基于熵分片内相邻基本单元的上下文模型，并且如果相邻基本单元不在熵分片内，则可以使用缺省值。在备选实施例中，可以将上下文模型重置为缺省值。在另一些备选实施例中，可以基于在比特流中通知了其标识符的上下文模型来重置上下文模型，所述标识符指示多个预定上下文模型之一。预定上下文模型可以依赖于比特流中的一个或更多个参数。在示例性实施例中，可以基于比特流内通知的“cabac_init_idc”值来重置上下文模型，“cabac_init_idc”值指示多个预定上下文模型之一。

[0209] 在一些实施例中，上下文表可以用于初始化多个上下文模型，其中，上下文表是指上下文模型集合。在一些实施例中，可以基于比特流中的一个或更多个参数(例如，量化参数、分片类型参数或其他参数)对上下文表中的上下文模式集合进行适配。

[0210] 在图34中示意的一个示例性实施例中，除了在熵分片中的开始宏块处重置上下文模型之外，当当前宏块是行中的第一宏块时，也在熵分片中重置上下文模型。图34示出了包含被划分成以下三个熵分片的48个宏块1208-1255的示例性重构分片1200：熵分片“0”(以十字影线示出)1202、熵分片1(以白色示出)1204以及熵分片“2”(以点影线示出)1206。熵分片“0”1202包含15个宏块1208-1222，熵分片“1”1204包含17个宏块1223-1239，熵分片“2”包含16个宏块1240-1255。在其处可以重置上下文模型的宏块由粗黑边缘1260-1266指示，并且是每个熵分片开始处的那些宏块1208、1223、1240，并且是每一行1216、1224、1232、1240、1248中的第一宏块。

[0211] 熵分片开始处的基本单元(例如，宏块)可以被称作分片开始基本单元。例如，对于图34中示例性重构分片1200中的熵分片1202、1204、1206，各个分片开始基本单元是1208、1223和1240。作为熵分片的行中的第一基本单元的基本单元可以被称作行开始基本单元，例如，图34中的宏块1208、1216、1224、1232、1240和1248。

[0212] 在一些实施例中，如果相邻宏块在熵分片内则可以基于相邻宏块的上下文模型重置上下文模型，并且如果相邻宏块不在熵分片内则将上下文模型重置为缺省值。例如，如果当前宏块之上的宏块在相同熵分片中，则基于在当前宏块之上的宏块的上下文模型重置上下文模型，但是如果当前宏块之上的宏块不在相同熵分片中，则将上下文模型设置为缺省值。

[0213] 在另一示例性实施例中，当当前基本单元是行中的第一基本单元时，可以重置上下文模型。在备选实施例中，上下文模型重置条件可以基于其他准则，例如，熵分片内处理的bin的数目、分片内处理的比特的数目、当前基本单元的空间位置和其他准则。

[0214] 在本发明的一些实施例中，上下文模型重置标志可以用于指示是否只要满足上下文模型重置条件就在熵分片中重置上下文模型。在一些实施例中，上下文模型重置标志可以在熵分片首部中。在备选实施例中，上下文模型重置标志可以在重构分片首部中。在一些实施例中，上下文模型重置标志可以是二进制标志，并且上下文模型重置条件可以是缺省条件。在备选实施例中，上下文模型重置标志可以是进一步指示上下文模型重置条件的多值标志。

[0215] 在包括上下文自适应编码(例如CABAC编码、CAV2V编码和其他上下文自适应编码)的一个示例性实施例中，“lcu_row_cabac_init_flag”标志可以通知在最大编码单元(LUC)

行开始处是否初始化熵解码。在一些实施例中,LCU是在H.264到高效率视频编码(HEVC)中使用的宏块概念的广义化,并且画面被划分成分片,其中,分片由LCU序列组成。在备选实施例中,LCU是可以利用单个传输的模式值来表示的像素位置的最大块。在备选实施例中,LCU是可以利用单个传输的预测模式值来表示的像素位置的最大块。在本发明的一些实施例中,“lcu_row_cabac_init_flag”标志值“1”可以通知重置熵编码上下文。熵编码上下文可以表示与熵编码器相关联的所有上下文模型的集合。在本发明的一些实施例中,“lcu_row_cabac_init_flag”标志值“1”可以通知重置熵编码上下文并且重置自适应扫描。自适应扫描可以指如下处理:编解码器基于先前传输的变换系数值来适配变换系数的扫描顺序。在一个实施例中,通过产生系数显著图来确定扫描顺序,并且可以在与小于或等于预定值的系数显著值相对应的变换系数值之前传输与大于该预定值的系数显著值相对应的变换系数值。在一个实施例中,可以随后增大与大于预定值的变换系数值相对应的系数显著值。在备选实施例中,可以随后减小与小于或等于预定值的变换系数值相对应的系数显著值。可以通过将系数显著图设置为预定值来重置自适应扫描处理。在一些实施例中,当没有发送标志时,针对“lcu_row_cabac_init_flag”标志采用的缺省值可以是“0”。“lcu_row_cabac_init_idc_flag”标志可以通知在每个LCU行的开始处是否传输cabac_init_idc值。在一些实施例中,当“lcu_row_cabac_init_idc_flag”标志的值是“1”时,在每个LCU行的开始处传输所述值。在一些实施例中,当没有发送“lcu_row_cabac_init_idc_flag”标志时,针对该标志采用的缺省值可以为“0”。在一些实施例中,“cabac_init_idc_present_flag”标志可以通知对于LCU是否传输cabac_init_idc值。在一些实施例中,当针对LCU没有传输cabac_init_idc值时,那么使用比特流中针对cabac_init_idc的在前值来重置熵编码上下文。在本发明的一些实施例中,例如当“entropy_slice_flag”的值为“0”时,可以在常规分片首部中通知“lcu_row_cabac_init_flag”和“lcu_row_cabac_init_idc_flag”。表8和表9示出了这些实施例的示例性语法。在本发明的一些实施例中,例如,当“entropy_slice_flag”的值为“1”时,可以在熵分片首部中通知“lcu_row_cabac_init_flag”和“lcu_row_cabac_init_idc_flag”。表8示出了示例性分片首部语法,表9示出了示例性分片数据语法(coding_unit)。

[0216]

slice_header() {	C	描述符
entropy_slice_flag	2	u(1)
if (entropy_slice_flag) {		
first_lcu_in_slice	2	ue(v)
if (entropy_coding_mode_flag) {		
lcu_row_cabac_init_flag	1	u(1)
if(lcu_row_cabac_init_flag){		
lcu_row_cabac_init_idc_flag	1	u(1)
}		
}		
if(entropy_coding_mode_flag && slice_type != I) {		
cabac_init_idc	2	ue(v)
}		
}		
else {		
lcu_row_cabac_init_flag	1	u(1)
if(lcu_row_cabac_init_flag){		
lcu_row_cabac_init_idc_flag	1	u(1)
}		
a regular slice header		
}		
}		

[0217] 表8:通知LCU行开始处熵编码的初始化的示例性语法表

[0218]

coding_unit(x0, y0, currCodingUnitSize) {	C	描述符
if (x0==0 && currCodingUnitSize==MaxCodingUnitSize && lcu_row_cabac_init_idc_flag==true && lcu_id!=first_lcu_in_slice) {		
cabac_init_idc_present_flag	1	u(1)
if(cabac_init_idc_present_flag)		
cabac_init_idc	2	ue(v)
}		
a regular coding unit ...		
}		

[0219] 表9:通知针对LCU的初始上下文的示例性语法表

[0220] 在包括上下文自适应编码(例如CABAC编码、CAV2V编码和其他上下文自适应编码)

的另一示例性实施例中，“mb_row_cabac_init_flag”标志可以在通知行中第一宏块处是否可以初始初始化熵解码。在本发明的一些实施例中，“mb_row_cabac_init_flag”标志值“1”可以通知在每个宏块行的开始处重置熵编码上下文。在本发明的备选实施例中，“mb_row_cabac_init_flag”标志值“1”可以通知重置熵编码上下文并且在每个宏块行的开始处重置自适应扫描。在一些实施例中，当没有发送“mb_row_cabac_init_flag”标志时，针对该标志采用的缺省值可以是“0”。“mb_row_cabac_init_idc_flag”标志可以通知在每个宏块行的开始处是否传输cabac_init_idc值。在一些实施例中，当“mb_row_cabac_init_idc_flag”标志的值是“1”时，在每个宏块行的开始处传输所述值。在一些实施例中，当不发送“mb_row_cabac_init_idc_flag”标志时，针对该标志采用的缺省值可以为“0”。在一些实施例中，“cabac_init_idc_present_flag”标志可以通知对于宏块是否传输cabac_init_idc值。在一些实施例中，当针对宏块不传输cabac_init_idc值时，使用比特流中针对cabac_init_idc的在前值来重置熵编码上下文。在本发明的一些实施例中，例如当“entropy_slice_flag”的值为“0”时，可以在常规分片首部中通知“mb_row_cabac_init_flag”和“mb_row_cabac_init_idc_flag”。在本发明的一些实施例中，例如，当“entropy_slice_flag”的值为“1”时，可以在熵分片首部中通知“mb_row_cabac_init_flag”和“mb_row_cabac_init_idc_flag”。表10和11示出了这些实施例的示例性语法。表10示出了示例性分片首部语法，表11示出了示例性分片数据语法(coding_unit)。

[0221]

slice_header() {	C	描述符
entropy_slice_flag	2	u(1)
if (entropy_slice_flag) {		
first_mb_in_slice	2	ue(v)
if (entropy_coding_mode_flag) {		
mb_row_cabac_init_flag	1	u(1)
if(mb_row_cabac_init_flag){		
mb_row_cabac_init_idc_flag	1	u(1)
}		
}		
if(entropy_coding_mode_flag && slice_type != I) {		
cabac_init_idc	2	ue(v)
}		
}		
else {		
mb_row_cabac_init_flag	1	u(1)
if(mb_row_cabac_init_flag){		
mb_row_cabac_init_idc_flag	1	u(1)
}		
a regular slice header		
}		
}		

[0222] 表10:通知宏块行开始处熵编码的初始化的示例性语法表

[0223]

coding_unit(x0, y0, currCodingUnitSize) {	C	描述符
if (x0==0 && currCodingUnitSize==MaxCodingUnitSize && mb_row_cabac_init_idc_flag==true && mb_id!=first_mb_in_slice) {		
cabac_init_idc_present_flag	1	u(1)
if(cabac_init_idc_present_flag)		
cabac_init_idc	2	ue(v)
}		
a regular coding unit ...		
}		

[0224] 表11:通知宏块的初始上下文的示例性语法表

[0225] 在本发明的一些实施例中，可以在比特流中通知熵分片在比特流中的位置。在一些实施例中，标志可以用于通知：在比特流中将要通知熵分片在比特流中的位置。一些示例性实施例可以包括“entropy_slice_locations_flag”，如果该标志为真，则可以指示在比特流中将要通知熵分片首部在比特流中的位置。在一些实施例中，可以以有区分地编码位置数据。在一些实施例中，可以在每个重构分片中发送位置数据。在备选实施例中，每个画面发送一次位置数据。

[0226] 在本发明的一些实施例中，可以在比特流中通知LCU行在比特流中的位置。在一些实施例中，标志可以用于通知：在比特流中将要通知每一行中的第一LCU在比特流中的位置。一些示例性实施例可以包括“lcu_row_location_flag”，如果该标志为真，则可以指示在比特流中将要通知每一行中第一LCU在比特流中的位置。在一些实施例中，可以有区分地编码位置数据。在一些实施例中，可以在每个熵分片中发送位置数据。在备选实施例中，每个重构分片发送一次位置数据。

[0227] 表12示出了通知LCU行和熵分片在比特流中的位置的示例性语法，对于该示例性语法，语义是：

[0228] • “entropy_slice_locations_flag”通知是否传输熵分片首部位置。如果“entropy_slice_locations_flag”的值设置为“1”，则传输熵分片首部位置，否则不传输。“entropy_slice_locations_flag”的缺省值为“0”。

[0229] • “num_of_entropy_slice_minus1”通知重构分片中熵分片的数目减1。

[0230] • “entropy_slice_offset[i]”指示第i个熵分片从在前熵分片的偏移。

[0231] • “lcu_row_locations_flag”通知是否传输LCU行位置信息。如果“lcu_row_locations_flag”的值是“1”，则传输LCU行位置信息，否则不传输。“lcu_row_locations_flag”的缺省值是“0”。

[0232] • “num_of_lcu_rows_minus1”通知熵分片中LCU行数减1。

[0233] • “lcu_row_offset[i]”指示第i个LCU行从在前LCU行的偏移。

[0234] 在本发明的一些实施例中，表12中的“lcu”由“宏块”代替。例如，表12中的″first_lcu_in_slice″、″lcu_row_cabac_init_flag″、″lcu_row_cabac_init_idc_flag″、″lcu_row_locations_flag″、″lcu_row_locations()″、″num_of_lcu_rows_minus1″和″lcu_row_offset[i]″分别由″first_mb_in_slice″、″mb_row_cabac_init_flag″、″mb_row_cabac_init_idc_flag″、″mb_row_locations_flag″、″mb_row_locations()″、″num_of_mb_rows_minus1″和″mb_row_offset[i]″来代替。

[0235]

slice_header() {	C	描述符
entropy_slice_flag	2	u(1)
if (entropy_slice_flag) {		
first_lcu_in_slice	2	ue(v)
If (entropy_coding_mode_flag) {		
lcu_row_cabac_init_flag	1	u(1)
if(lcu_row_cabac_init_flag){		
lcu_row_cabac_init_idc_flag	1	u(1)
lcu_row_locations_flag	1	u(1)
if (lcu_row_locations_flag) {		
lcu_row_locations ()		
}		
}		
}		
if(entropy_coding_mode_flag && slice_type != I) {		
cabac_init_idc	2	ue(v)
}		
}		
else {		
entropy_slice_locations_flag	1	u(1)
if (entropy_slice_locations_flag) {		
entropy_slice_locations()		
}		
If (entropy_coding_mode_flag) {		
lcu_row_cabac_init_flag	1	u(1)
if(lcu_row_cabac_init_flag){		
lcu_row_cabac_init_idc_flag	1	u(1)
lcu_row_locations_flag	1	u(1)
if (lcu_row_locations_flag) {		
lcu_row_ locations ()		
}		
}		
}		
a regular slice header		
}		
}		

[0236] 表12:通知行中第一LCU在比特流中的位置的示例性语法表。

[0237] 表13示出了全帧内编码的速率失真性能的比较。第三列的两个子列中示出的第一比较是使用H.264/AVC联合模型(JM)软件(版本13.0)的使用多个分片的编码(其中针对分片的熵解码和宏块重构不依赖于其他分片)与不使用分片的编码之间的比较。平均而言,对于相同比特率,使用多个分片的编码与不使用分片的编码相比,质量劣化了-0.3380dB。平均而言,对于相同质量等级,使用多分片的编码与不使用分片的编码相比比特率提高了7%。

[0238] 第四列的两个子列中示出的第二比较是使用根据本发明实施例的被划分成多个熵分片(每个熵分片两行宏块)的一个重构分片的编码与使用JM 13.0而无分片的编码之间的比较。平均而言,对于相同比特率,使用具有多个熵分片的一个重构分片的编码与不使用分片的编码相比,质量劣化了-0.0860dB。平均而言,对于相同质量等级,使用具有多个熵分片的重构分片的编码与不使用分片的编码相比比特率提高了1.83%。

[0239]

全帧内编码					
序列	分辨率	有分片的JM 13.0与无分片的JM 13.0的比较		有多个熵分片的一个重构分片与无分片的JM 13.0的比较	
		BD SNR [dB]	BD比特率 [%]	BD SNR [dB]	BD 比特率 [%]
BigShip	720p	-0.22	4.54	-0.08	1.61
City	720p	-0.28	4.03	-0.06	0.84
Crew	720p	-0.42	11.67	-0.11	2.98
Night	720p	-0.38	5.64	-0.06	0.91
ShuttleStart	720p	-0.39	9.12	-0.12	2.81
平均		**-0.3380**	**7.00**	**-0.0860**	**1.83**

[0240] 表13:速率失真性能的比较-全帧内编码

[0241] 表14示出了针对IBBP编码的速率失真比特率的比较。第三列的两个子列中示出的第一比较是使用H.264/AVC联合模型(JM)软件(版本13.0)的使用多个分片的编码(其中针对分片的熵解码和宏块重构不依赖于其他分片)与不使用分片的编码之间的比较。平均而言,对于相同比特率,使用多个分片的编码与不使用分片的编码相比,质量劣化了-0.5460dB。平均而言,对于相同质量等级,使用多个分片的编码与不使用分片的编码相比比特率提高了21.41%。

[0242] 第四列的两个子列中示出的第二比较是使用根据本发明实施例的被划分成多个熵分片(每个熵分片两行宏块)的一个重构分片的编码与使用JM 13.0而无分片的编码之间的比较。平均而言,对于相同比特率,使用具有多个熵分片的一个重构分片的编码与不使用分片的编码相比,质量劣化了-0.31dB。平均而言,对于相同质量等级,使用具有多个熵分片的重构分片的编码与不使用分片的编码相比比特率提高了11.45%。

[0243]

IBBP编码					
序列	分辨率	有分片的JM 13.0与无分片的JM 13.0的比较		有多个熵分片的一个重构分片与无分片的JM 13.0的比较	
		BD SNR [dB]	BD比特率 [%]	BD SNR [dB]	BD 比特率 [%]
BigShip	720p	-0.45	19.34	-0.26	10.68
City	720p	-0.48	17.83	-0.22	7.24
Crew	720p	-0.62	30.10	-0.33	14.93
Night	720p	-0.36	11.11	-0.19	5.5
ShuttleStart	720p	-0.82	28.69	-0.55	18.89
平均		**-0.5460**	**21.41**	**-0.31**	**11.45**

[0244] 表14:速率失真性能的比较-IBBP编码

[0245] 比较结果,尽管使用一个重构分片中的多个熵分片的编码与使用分片的编码均允许并行解码,对于全帧内编码和IBBP编码,使用一个重构分片中的多个熵分片的编码与使用分片的编码相比分别提供了5.17%和9.96%的比特率节省,其中,针对分片的熵解码和宏块重构不依赖于其他分片。

[0246] 表15示出了针对全帧内编码和和IBBP编码的速率失真性能比较。在该表中,比较是不使用分片的编码与根据本发明实施例的使用被划分成熵分片的一个重构分片的编码之间的比较,每熵分片的最大尺寸为26k bin。第二列的两个子列中示出的第一比较是使用全帧内编码的比较。平均而言,对于相同比特率,使用具有多个分片的重构分片的编码的质量劣化了-0.062dB。平均而言,对于相同质量等级,使用具有多个熵分片的重构分片的编码的比特率提高了1.86%。因此,对于使用熵分片(每熵分片的最大尺寸为26k bin)的全帧内编码,与两行宏块的固定熵分片尺寸的熵分片相比存在接近0.64%的平均比特率节省。

[0247] 第三列的两个子列中示出的第二比较是使用IBBP编码的比较。平均而言,对于相同比特率,使用具有多个分片的重构分片的编码与不使用分片的编码相比质量劣化了-0.022dB。平均而言,对于相同质量等级,使用具有多个熵分片的重构分片的编码与不使用分片的编码相比比特率提高了0.787%。因此,对于使用熵分片(每熵分片的最大尺寸为26k bin)的IBBP编码,与两行宏块的固定熵分片尺寸的熵分片相比存在接近10.66%的平均比特率节省。

[0248]

熵分片与无分片的JM 15.1的比较实验(1): 每个熵分片的最大尺寸为26k bin				
序列 (720p)	全帧内编码		IBBP编码	
	BD SNR [dB]	BD 比特率 [%]	BD SNR [dB]	BD 比特率 [%]
BigShip	-0.07	1.40	-0.02	0.70
City	-0.07	1.02	-0.02	0.51
Crew	-0.05	1.31	-0.03	1.25
Night	-0.07	1.00	-0.02	0.66
ShuttleStart	-0.05	1.20	-0.03	-0.82
平均	**-0.062**	**1.187**	**-0.022**	**0.787**

[0249] 表15:速率失真性能的比较-使用具有少于每熵分片26k bin的熵分片的全帧内和IBBP编码

[0250] 使用熵分片允许并行编码,并且重构分片到熵分片的编码器划分与固定数目宏块的熵分片相比提供可观的比特率节省,其中,每个熵分片小于最大数目个bin。

[0251] 尽管附图中的图表和图可以示出了执行的特定顺序,但是应理解执行顺序可以与所示的执行顺序不同。例如,块的执行顺序可以相对于所示的顺序而变化。同样,作为另一示例,附图中连续示出的两个或更多块可以并发或者部分并发执行。本领域普通技术人员应理解,本领域技术人员可以创建软件、硬件和/或固件来执行本文描述的各种逻辑功能。

[0252] 本发明的一些实施例可以包括计算机程序产品,计算机程序产品包括其上/其中存储有指令的计算机可读存储介质,指令可以用于编程计算系统来执行本文描述的任何特征和方法。示例性计算机可读存储介质可以包括但不限于,闪速存储设备,盘存储介质(例如,软盘、光盘、磁光盘、数字万能盘(DVD)、压缩盘(CD)、微型驱动器和其他盘存储介质)、只读存储器(ROM)、可编程只读存储器(PROM)、可擦除可编程只读存储器(EPROM)、电可擦除可编程只读存储器(EEPROM)、随机存取存储器(RAM)、视频随机存取存储器(VRAM)、动态随机存取存储器(DRAM)和适合于存储指令和/或数据的任何类型的介质或设备。

[0253] 上述说明书中采用的术语和表述在本文用作描述目的并不用于限制,并且并不意在排除所示或所描述的特征的等同物或其部分的情况下使用这种术语和表述。应认识到本发明的范围仅由所附权利要求来限定和限制。

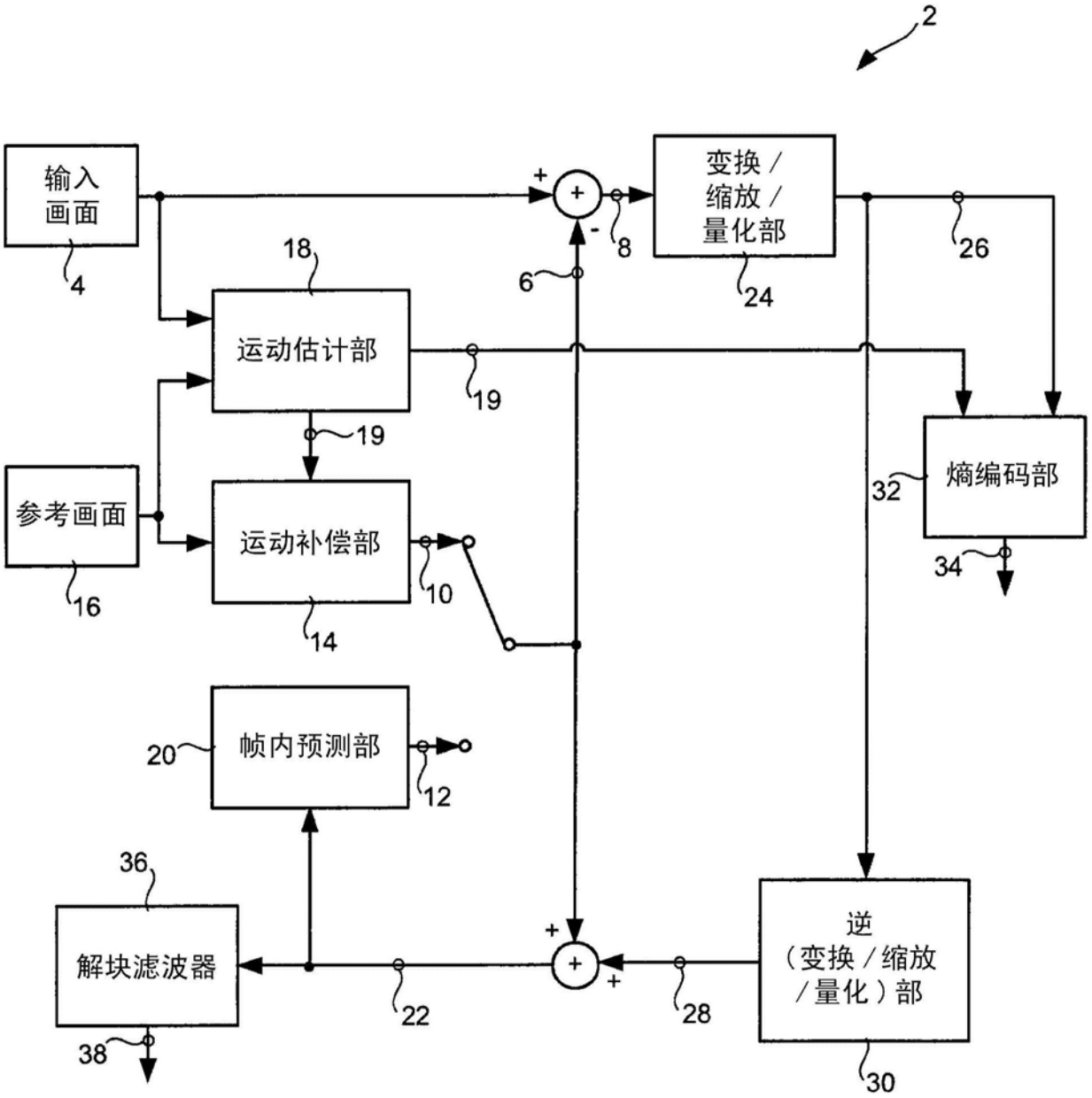

2
输入画面
4
+
+
-
8
变换/缩放/量化部
24
26
18
运动估计部
6
19
19
参考画面
16
运动补偿部
14
10
熵编码部
32
34
帧内预测部
20
12
36
解块滤波器
38
22
+
+
28
逆（变换/缩放/量化）部
30

图1

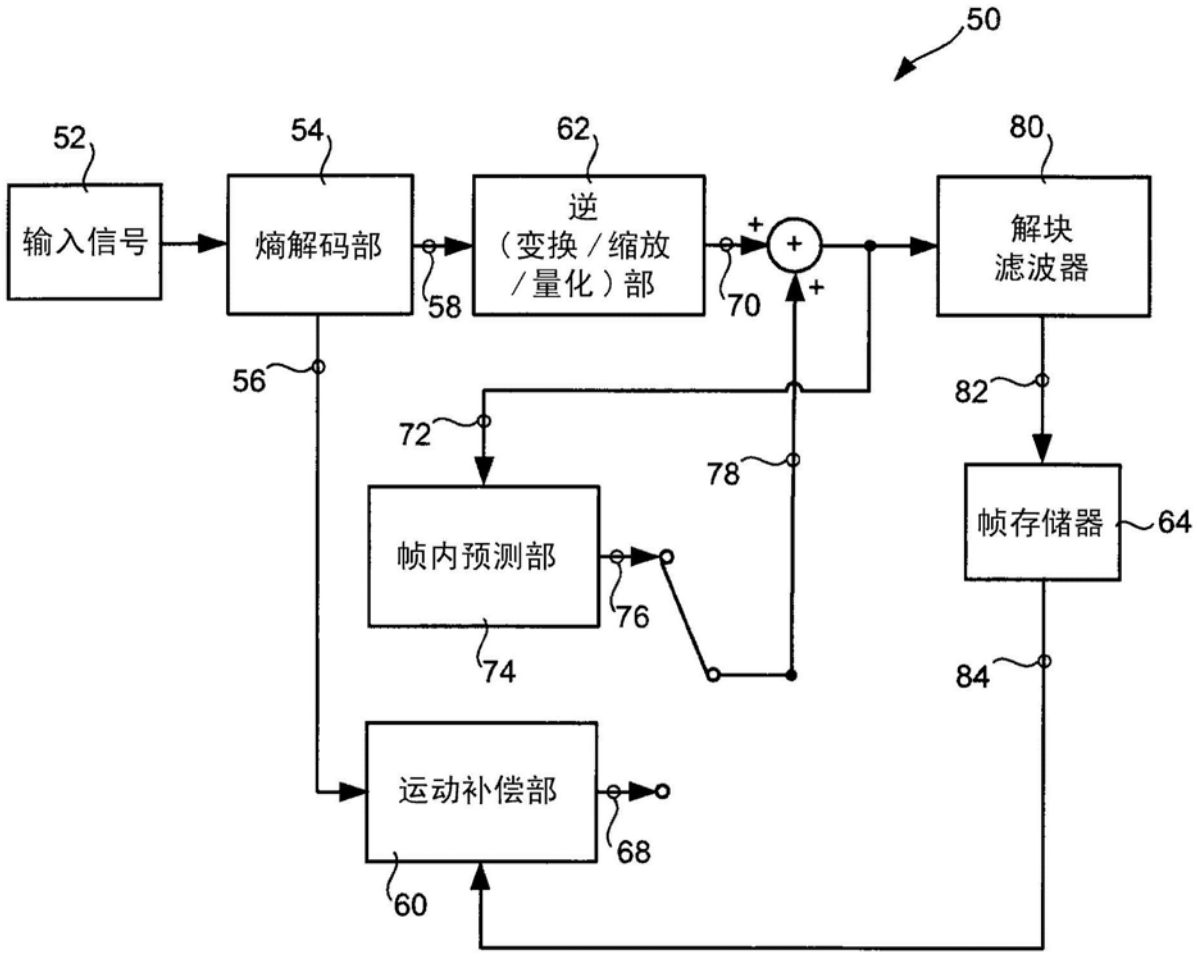

50
52
输入信号
54
熵解码部
58
62
逆
（变换／缩放
／量化）部
70
+
+
+
80
解块
滤波器
56
72
78
82
帧内预测部
76
74
帧存储器
64
84
运动补偿部
68
60

图2

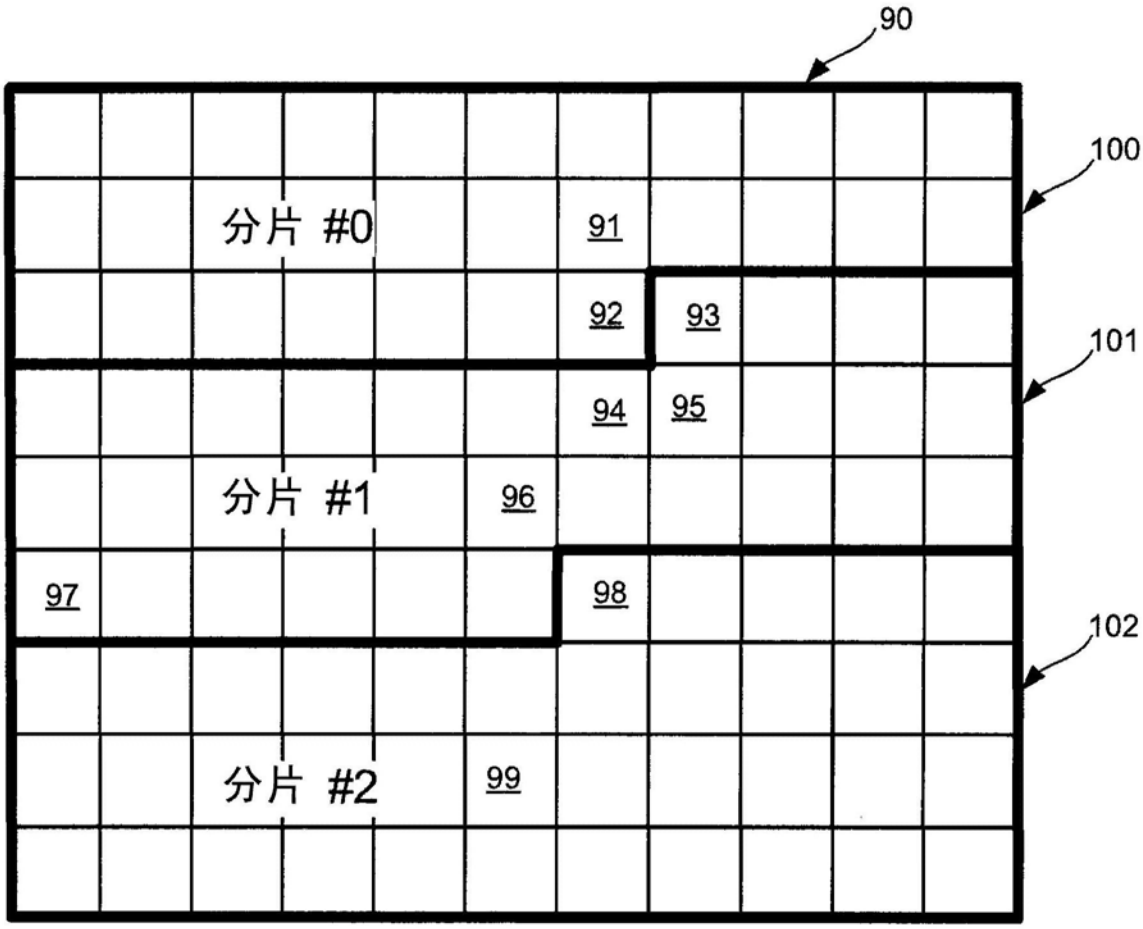


图3

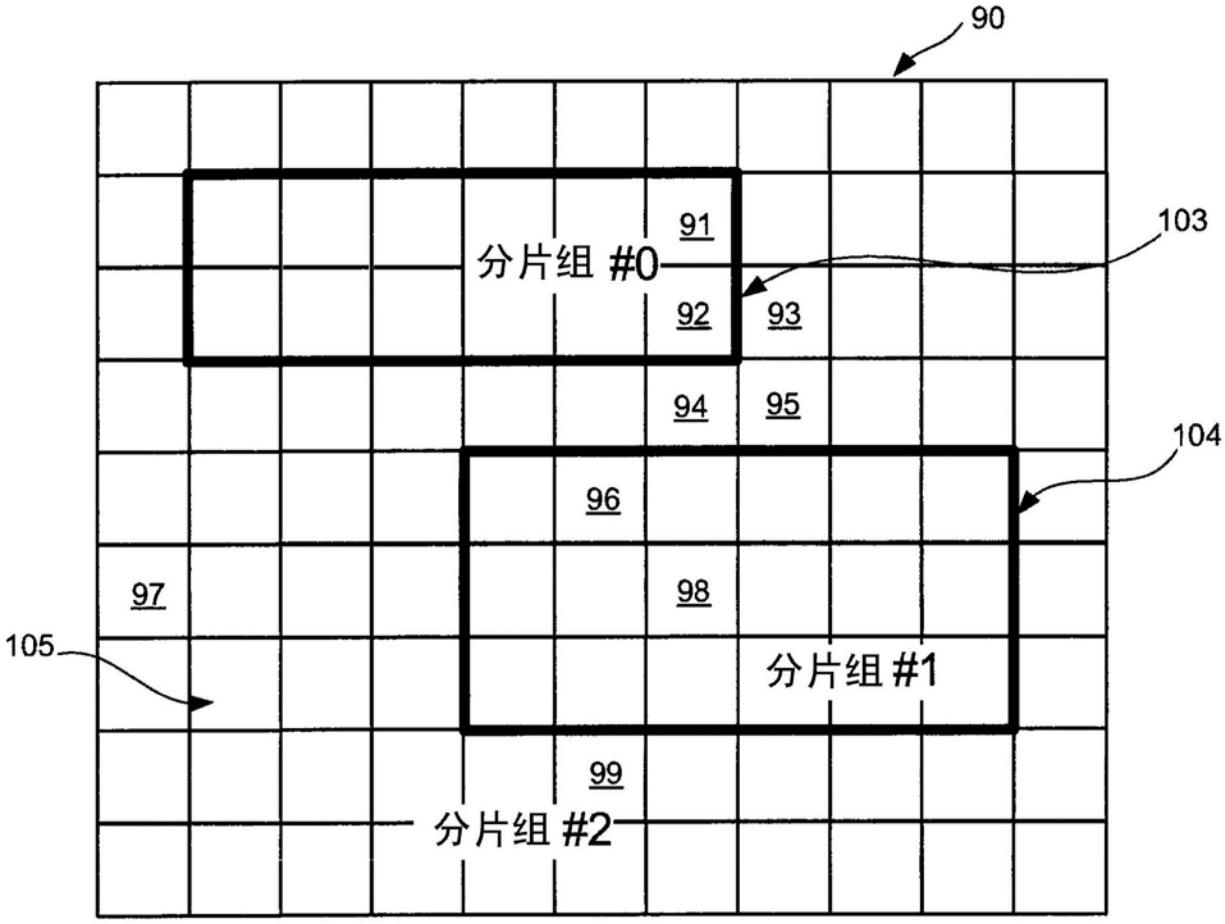


图4

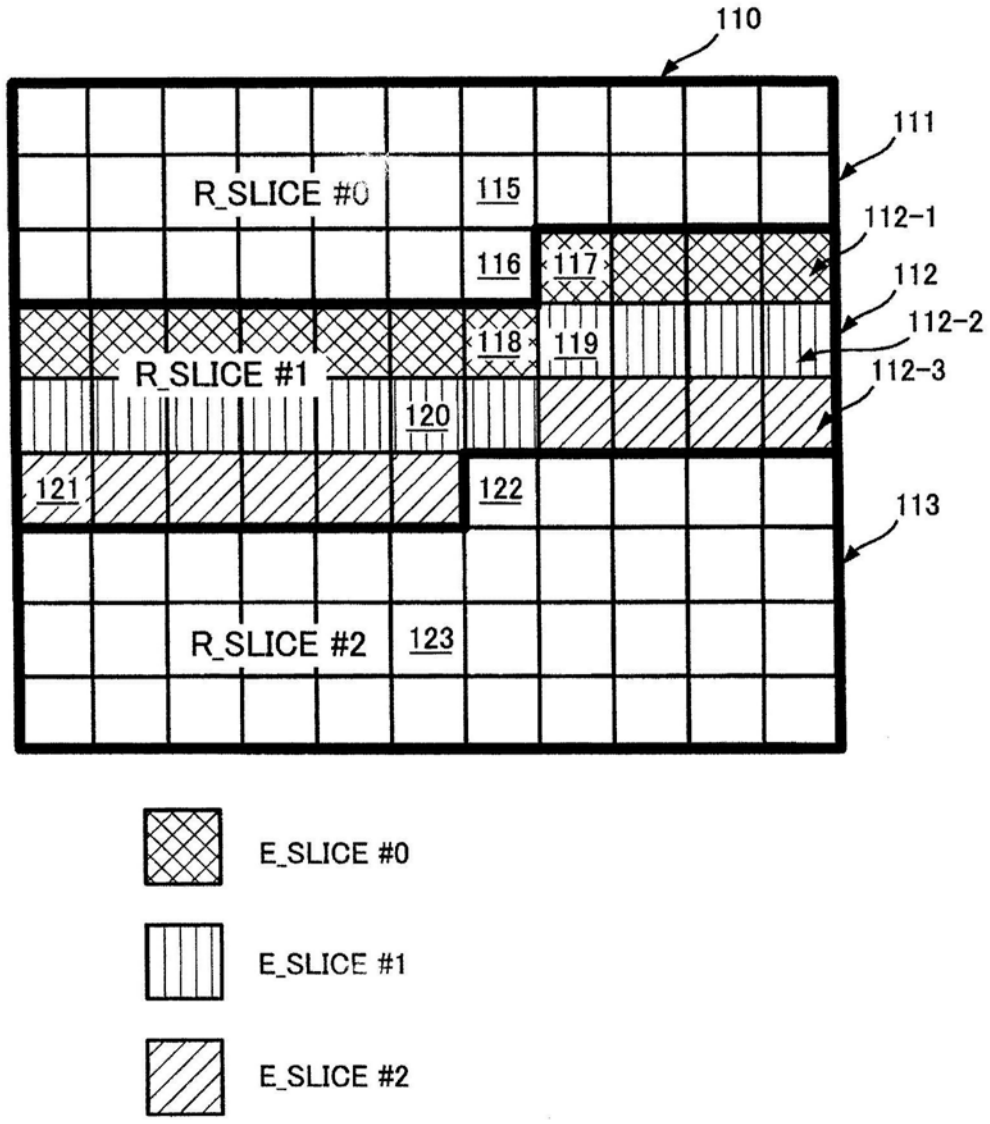

110
111
112-1
112
112-2
112-3
113
R_SLICE #0
115
116
117
118
119
R_SLICE #1
120
121
122
R_SLICE #2
123
E_SLICE #0
E_SLICE #1
E_SLICE #2

图5

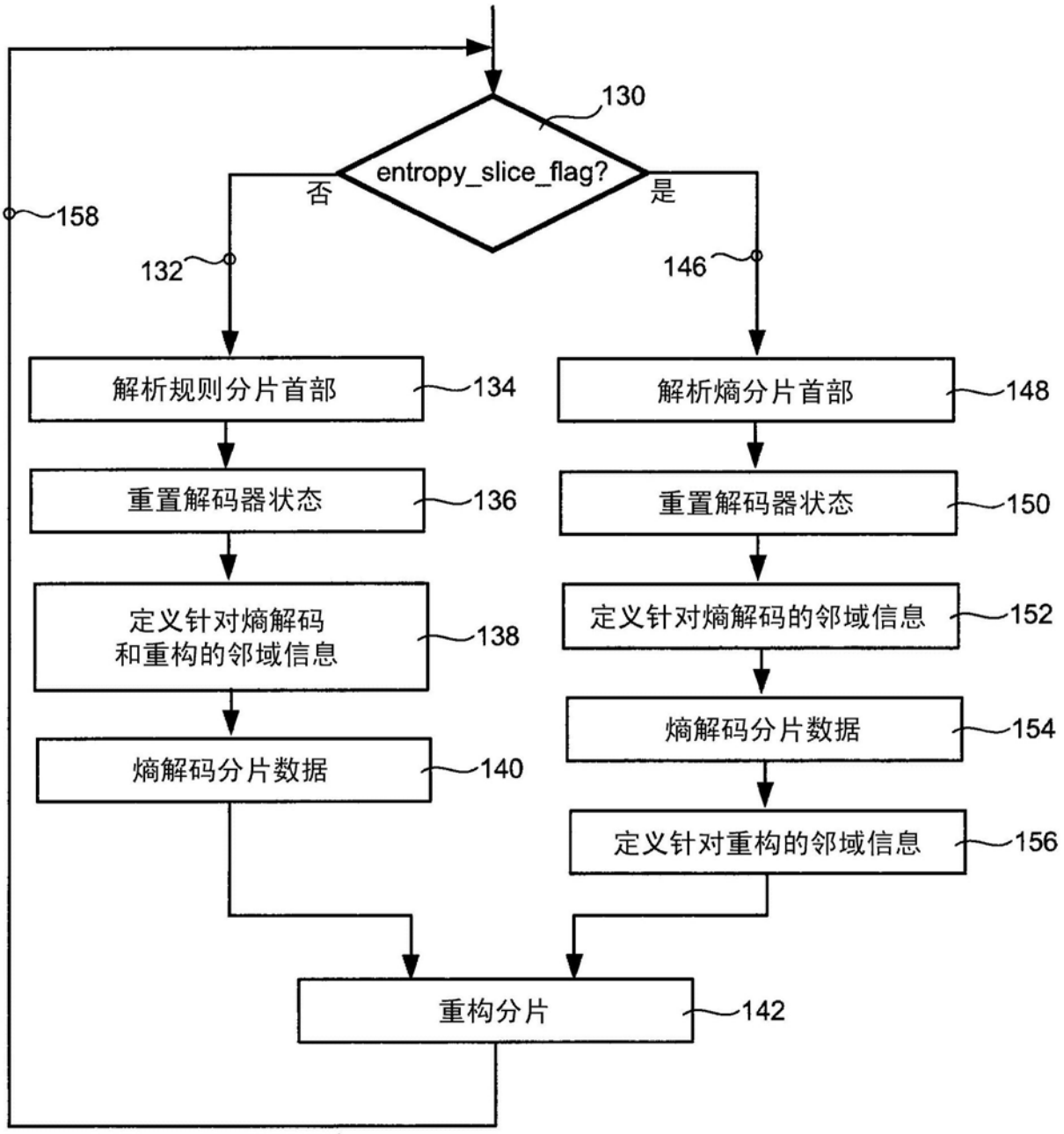

130
entropy_slice_flag?
否
是
158
132
146
解析规则分片首部
134
解析熵分片首部
148
重置解码器状态
136
重置解码器状态
150
定义针对熵解码
和重构的邻域信息
138
定义针对熵解码的邻域信息
152
熵解码分片数据
140
熵解码分片数据
154
定义针对重构的邻域信息
156
重构分片
142

图6

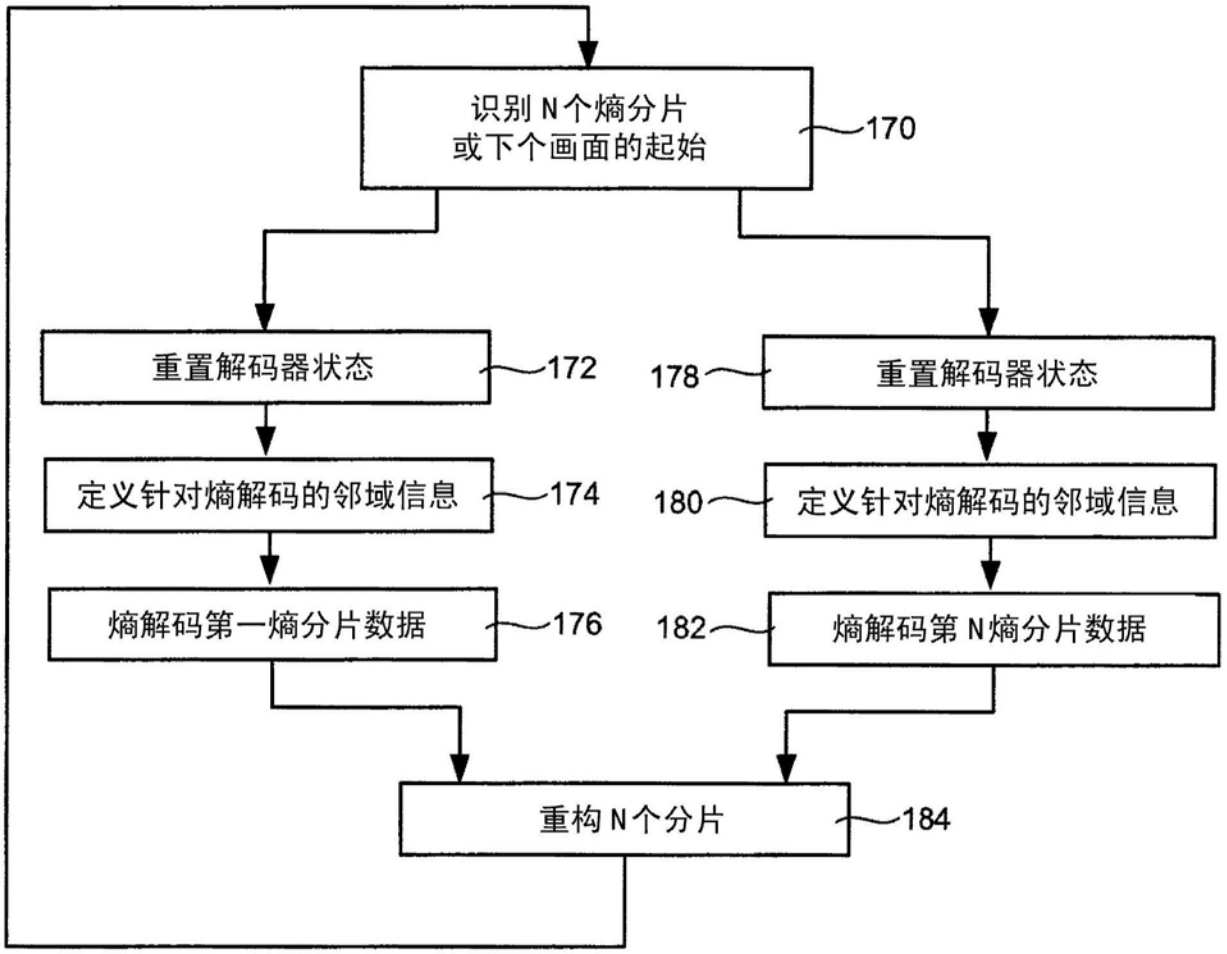

识别 N 个熵分片
或下个画面的起始
170
重置解码器状态
172
定义针对熵解码的邻域信息
174
熵解码第一熵分片数据
176
178
重置解码器状态
180
定义针对熵解码的邻域信息
182
熵解码第 N 熵分片数据
重构 N 个分片
184

图7

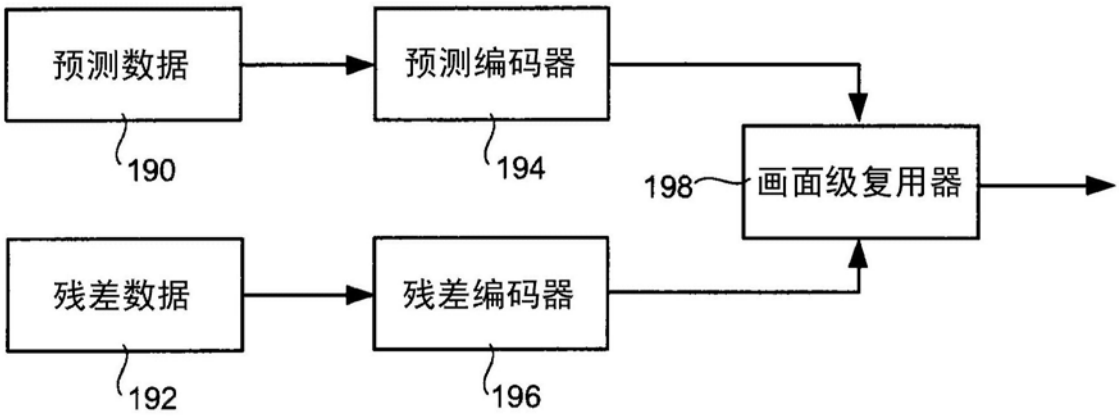

预测数据
190
预测编码器
194
198
画面级复用器
残差数据
192
残差编码器
196

图8

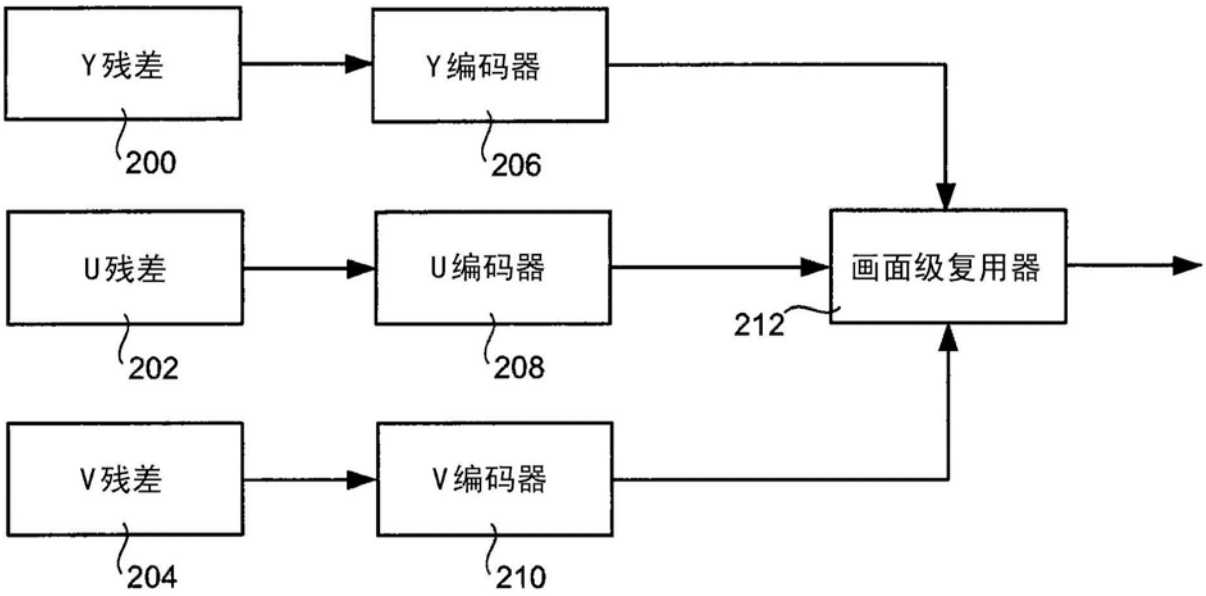


图9

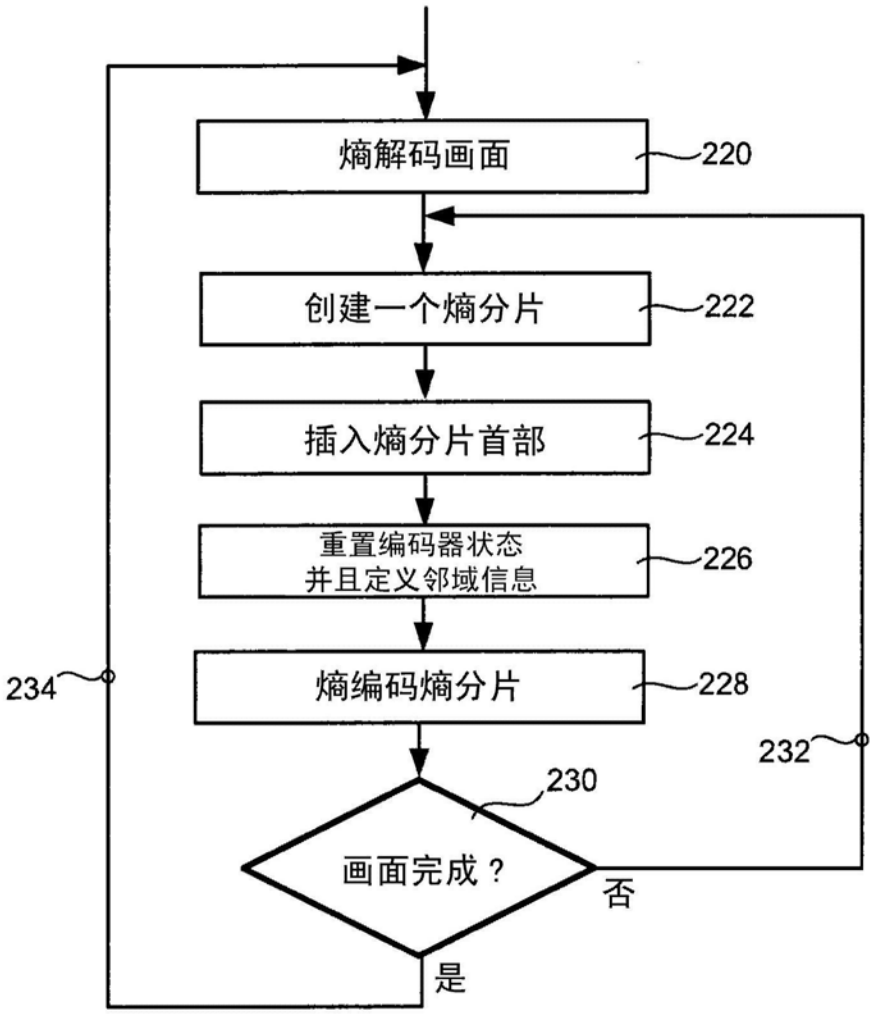

熵解码画面
220
创建一个熵分片
222
插入熵分片首部
224
重置编码器状态
并且定义邻域信息
226
熵编码熵分片
228
230
画面完成？
否
是
232
234

图10

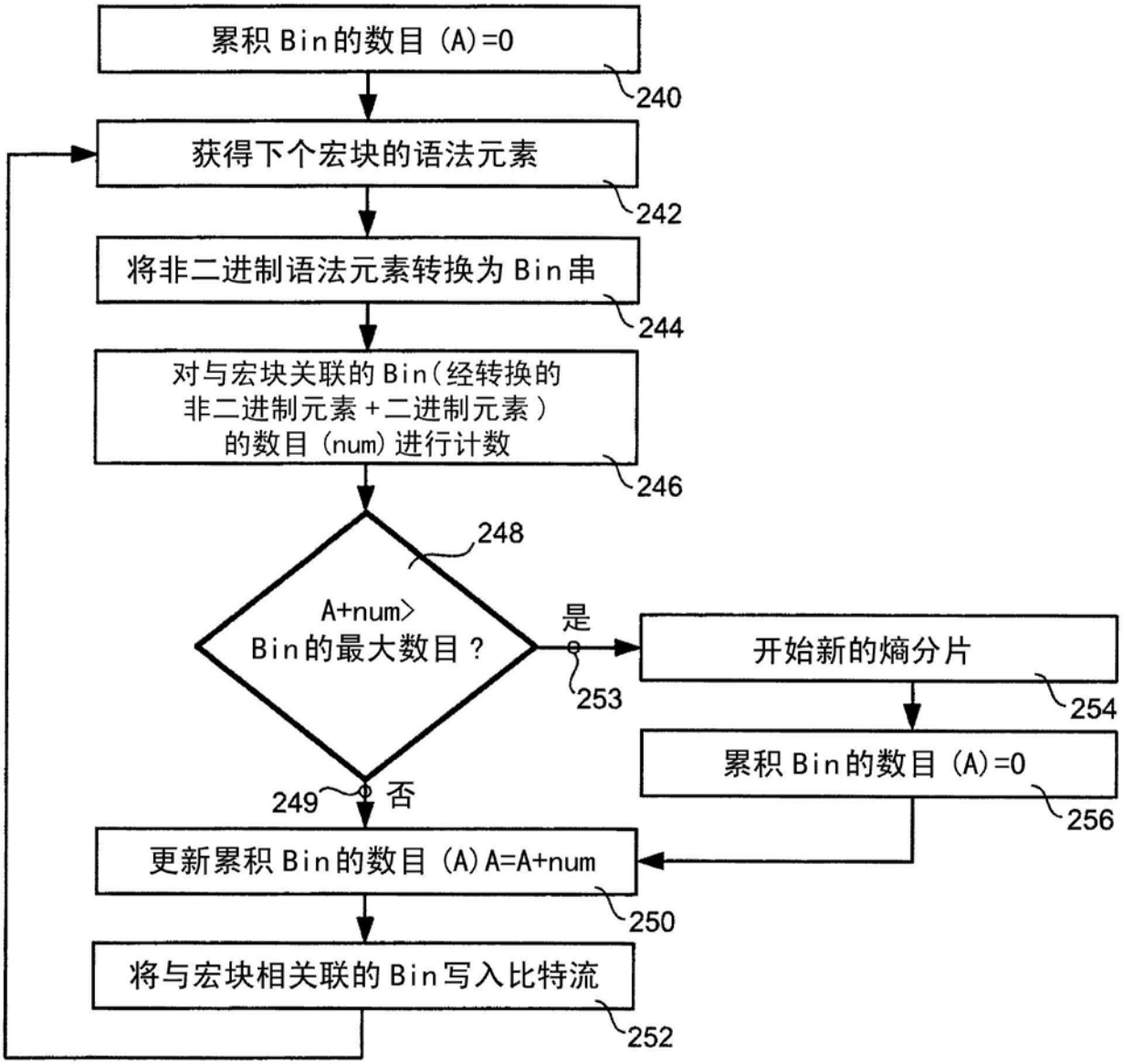

累积 Bin的数目 (A)=0
240
获得下个宏块的语法元素
242
将非二进制语法元素转换为 Bin串
244
对与宏块关联的 Bin(经转换的
非二进制元素 +二进制元素)
的数目 (num) 进行计数
246
248
A+num>
Bin的最大数目 ?
是
253
开始新的熵分片
254
累积 Bin的数目 (A)=0
256
249
否
更新累积 Bin的数目 (A) A=A+num
250
将与宏块相关联的 Bin写入比特流
252

图11

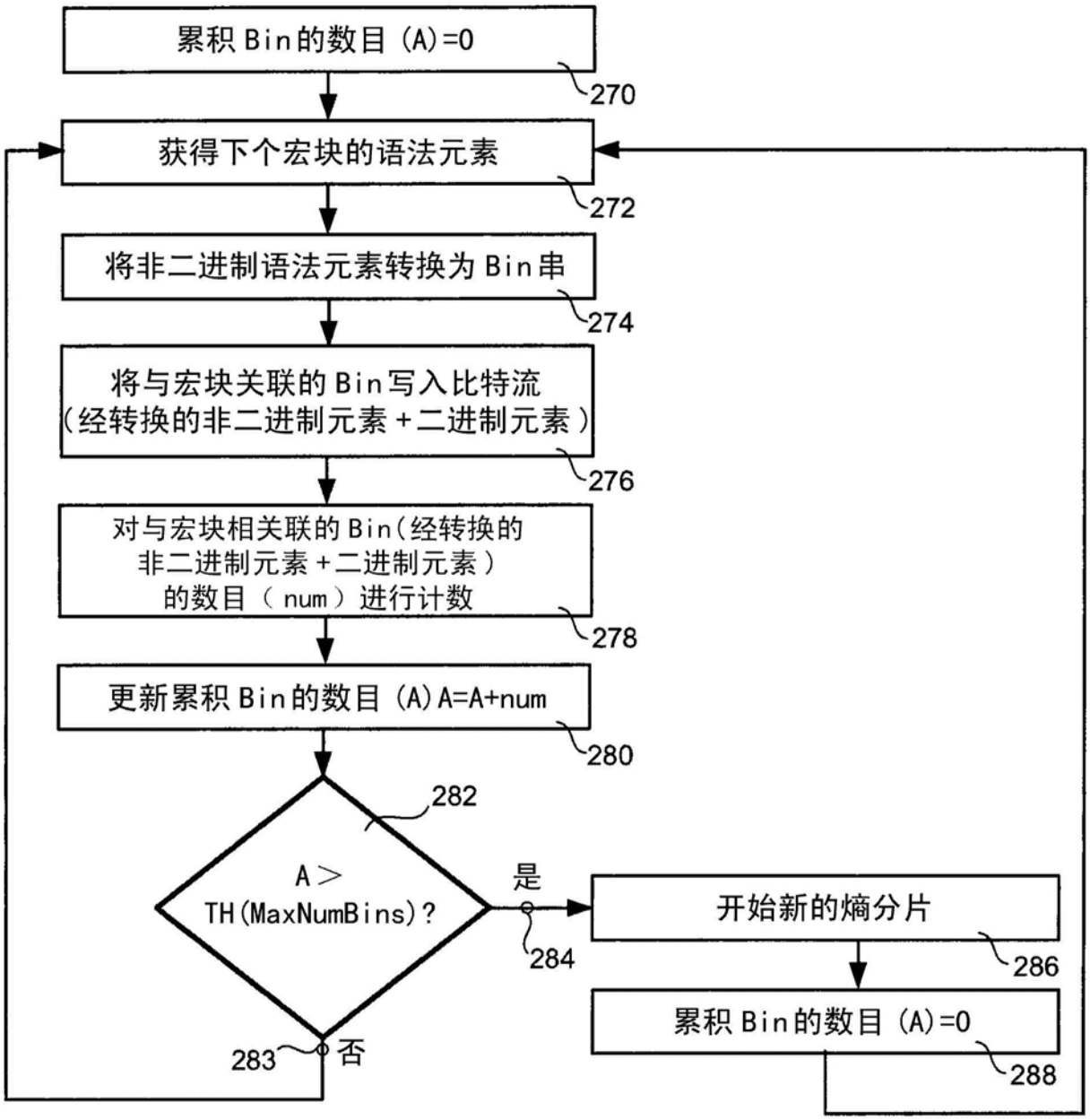

累积 Bin 的数目 (A)=0
270
获得下个宏块的语法元素
272
将非二进制语法元素转换为 Bin 串
274
将与宏块关联的 Bin 写入比特流
(经转换的非二进制元素 + 二进制元素)
276
对与宏块相关联的 Bin(经转换的
非二进制元素 + 二进制元素)
的数目 (num) 进行计数
278
更新累积 Bin 的数目 (A) A=A+num
280
282
A >
TH(MaxNumBins)?
是
284
开始新的熵分片
286
累积 Bin 的数目 (A)=0
288
283
否

图12

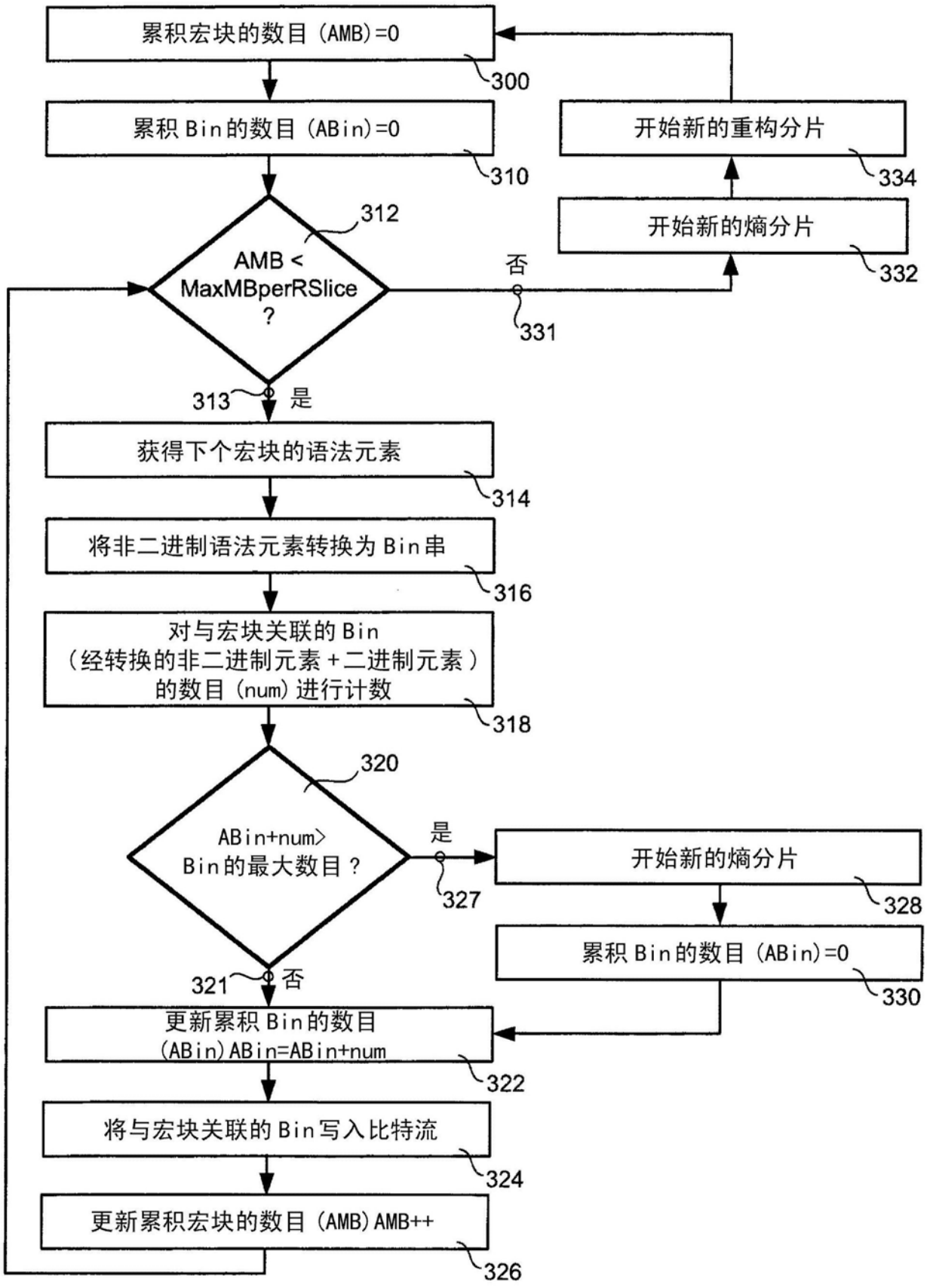

累积宏块的数目（AMB）=0
300
累积 Bin 的数目（ABin）=0
310
312
AMB < MaxMBperRSlice ?
否
331
开始新的重构分片
334
开始新的熵分片
332
313
是
获得下个宏块的语法元素
314
将非二进制语法元素转换为 Bin 串
316
对与宏块关联的 Bin（经转换的非二进制元素 + 二进制元素）的数目（num）进行计数
318
320
ABin+num> Bin 的最大数目？
是
327
开始新的熵分片
328
累积 Bin 的数目（ABin）=0
330
321
否
更新累积 Bin 的数目（ABin）ABin=ABin+num
322
将与宏块关联的 Bin 写入比特流
324
更新累积宏块的数目（AMB）AMB++
326

图13

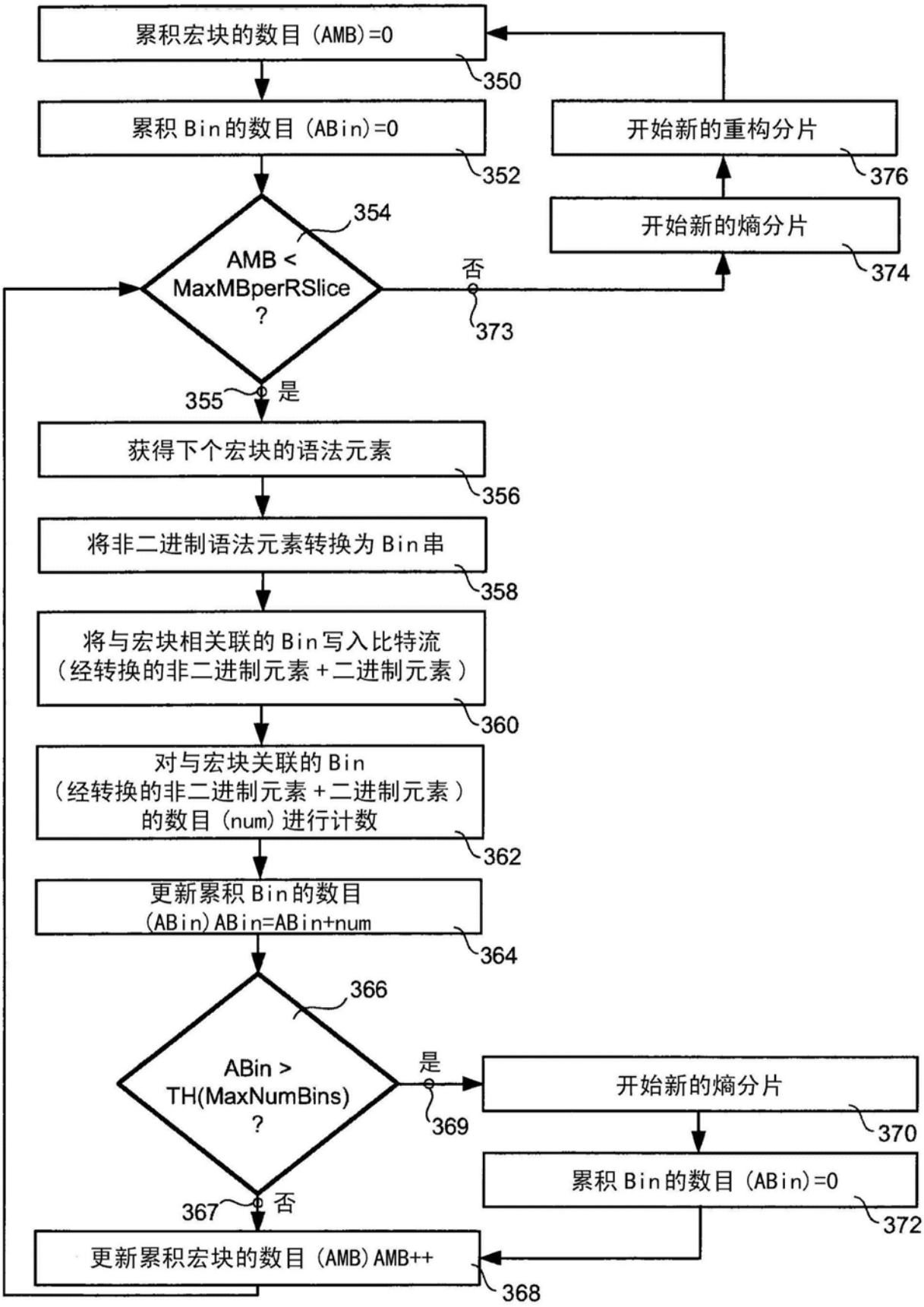

累积宏块的数目 (AMB)=0
350
累积 Bin 的数目 (ABin)=0
352
354
AMB < MaxMBperRSlice ?
否
373
开始新的重构分片
376
开始新的熵分片
374
355
是
获得下个宏块的语法元素
356
将非二进制语法元素转换为 Bin 串
358
将与宏块相关联的 Bin 写入比特流 (经转换的非二进制元素 + 二进制元素)
360
对与宏块关联的 Bin (经转换的非二进制元素 + 二进制元素) 的数目 (num) 进行计数
362
更新累积 Bin 的数目 (ABin) ABin=ABin+num
364
366
ABin > TH(MaxNumBins) ?
是
369
开始新的熵分片
370
累积 Bin 的数目 (ABin)=0
372
367
否
更新累积宏块的数目 (AMB) AMB++
368

图14

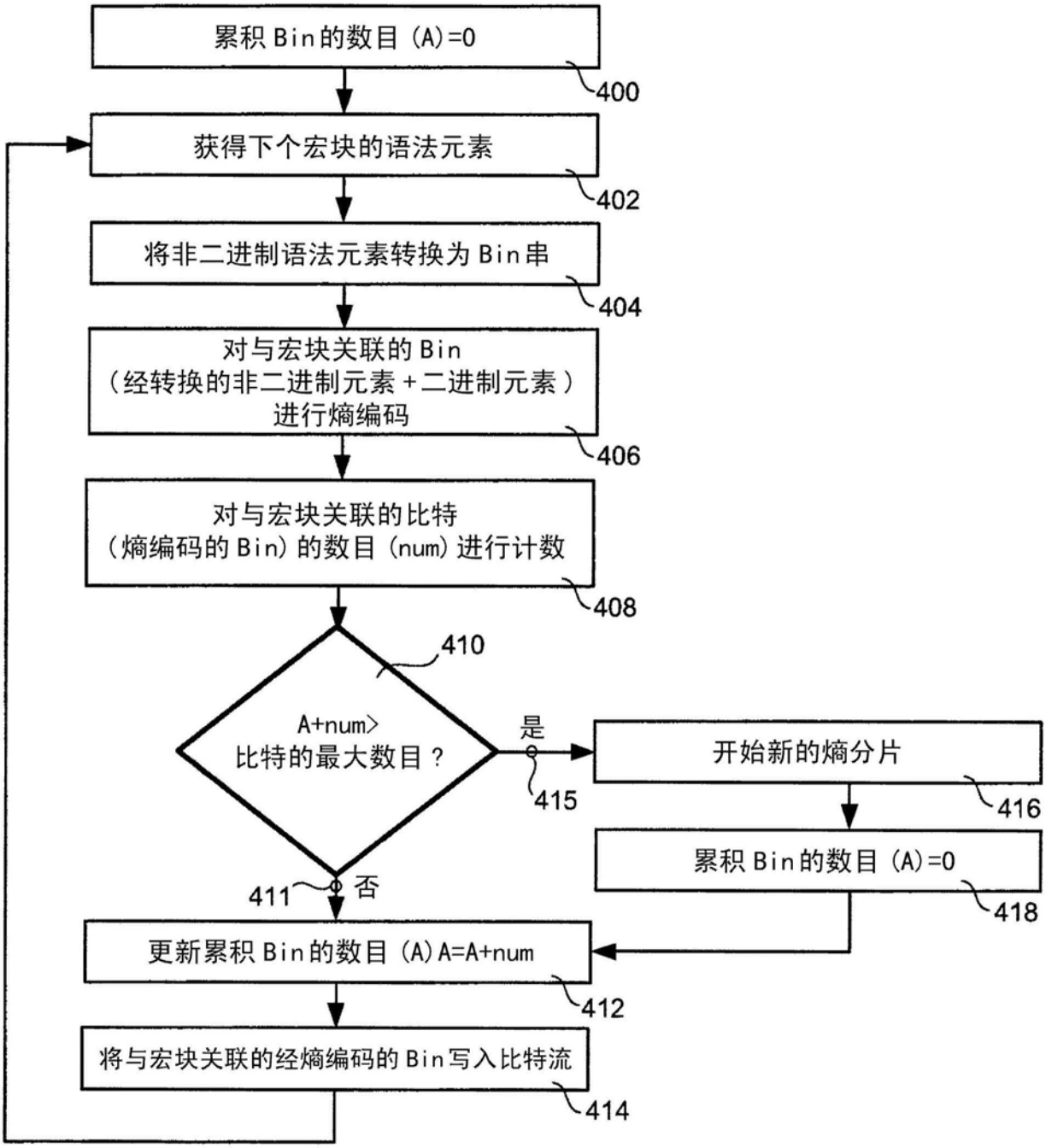

累积 Bin 的数目 (A)=0
400
获得下个宏块的语法元素
402
将非二进制语法元素转换为 Bin 串
404
对与宏块关联的 Bin
（经转换的非二进制元素 + 二进制元素）
进行熵编码
406
对与宏块关联的比特
（熵编码的 Bin）的数目 (num) 进行计数
408
410
A+num>
比特的最大数目？
是
415
开始新的熵分片
416
累积 Bin 的数目 (A)=0
418
411
否
更新累积 Bin 的数目 (A) A=A+num
412
将与宏块关联的经熵编码的 Bin 写入比特流
414

图15

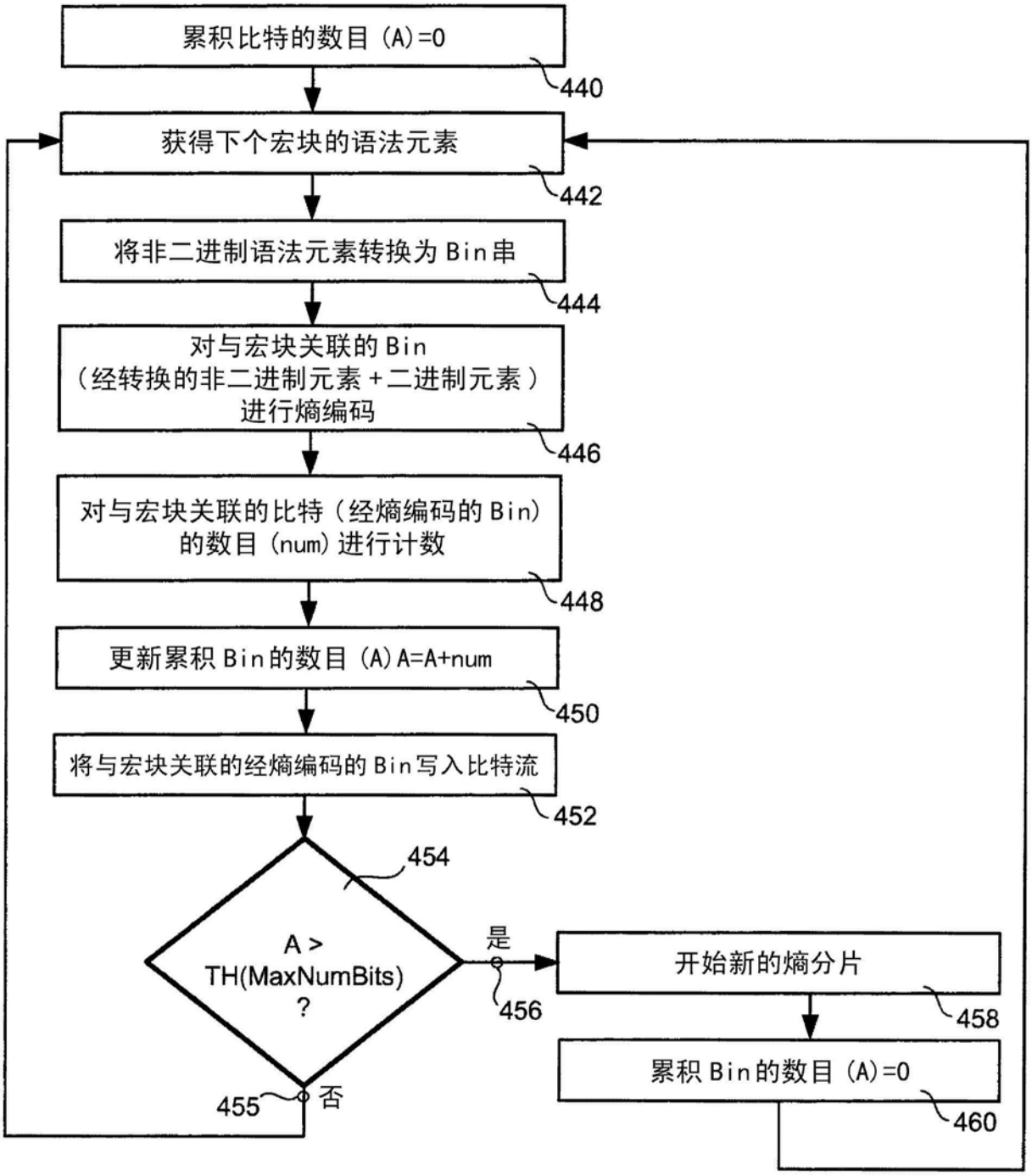

累积比特的数目 (A)=0
440
获得下个宏块的语法元素
442
将非二进制语法元素转换为 Bin 串
444
对与宏块关联的 Bin
（经转换的非二进制元素 + 二进制元素）
进行熵编码
446
对与宏块关联的比特（经熵编码的 Bin）
的数目 (num) 进行计数
448
更新累积 Bin 的数目 (A) A=A+num
450
将与宏块关联的经熵编码的 Bin 写入比特流
452
454
A >
TH(MaxNumBits)
?
是
456
开始新的熵分片
458
累积 Bin 的数目 (A)=0
460
否
455

图16

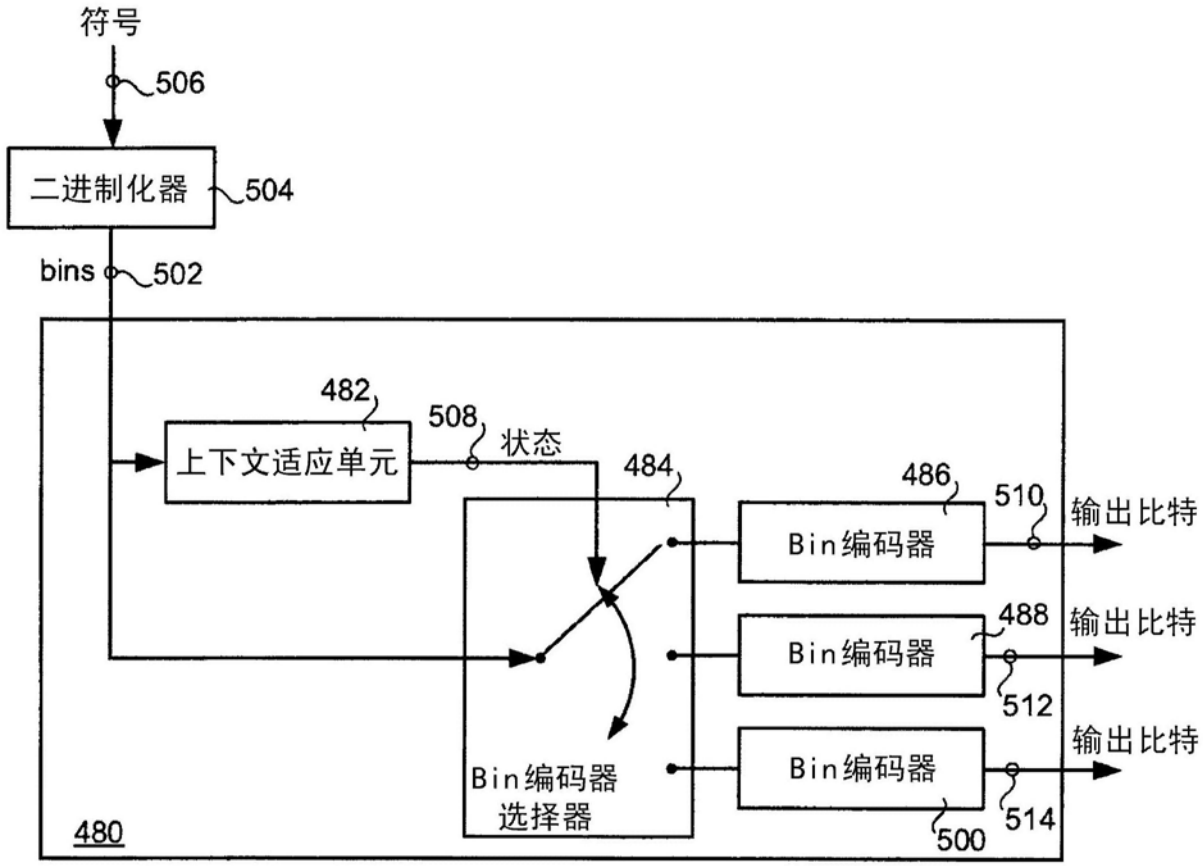

符号
506
二进制化器
504
bins
502
482
上下文适应单元
508
状态
484
486
510
输出比特
Bin编码器
488
输出比特
Bin编码器
512
输出比特
Bin编码器
Bin编码器
选择器
514
500
480

图17

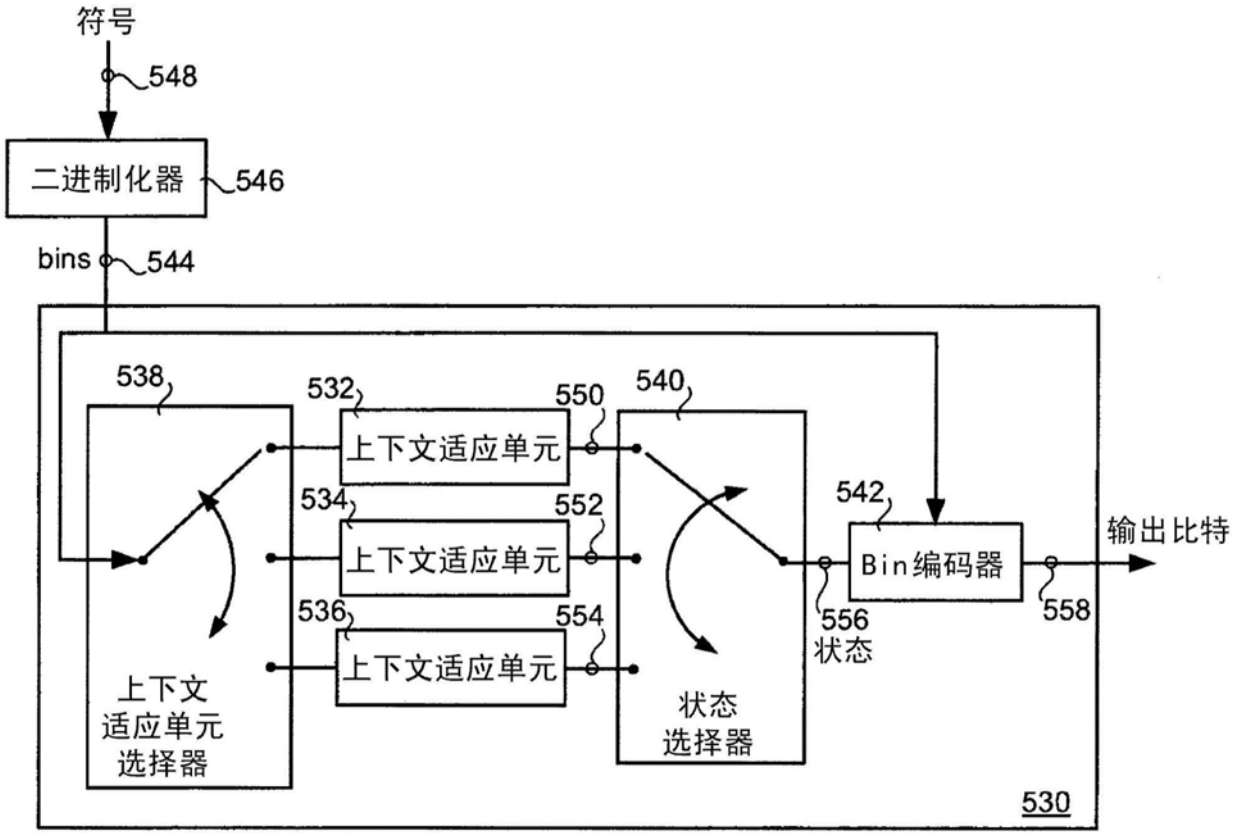

符号
548
二进制化器
546
bins
544
538
532
550
540
上下文适应单元
534
552
542
Bin编码器
输出比特
上下文适应单元
536
554
556
状态
558
上下文
适应单元
选择器
状态
选择器
530

图18

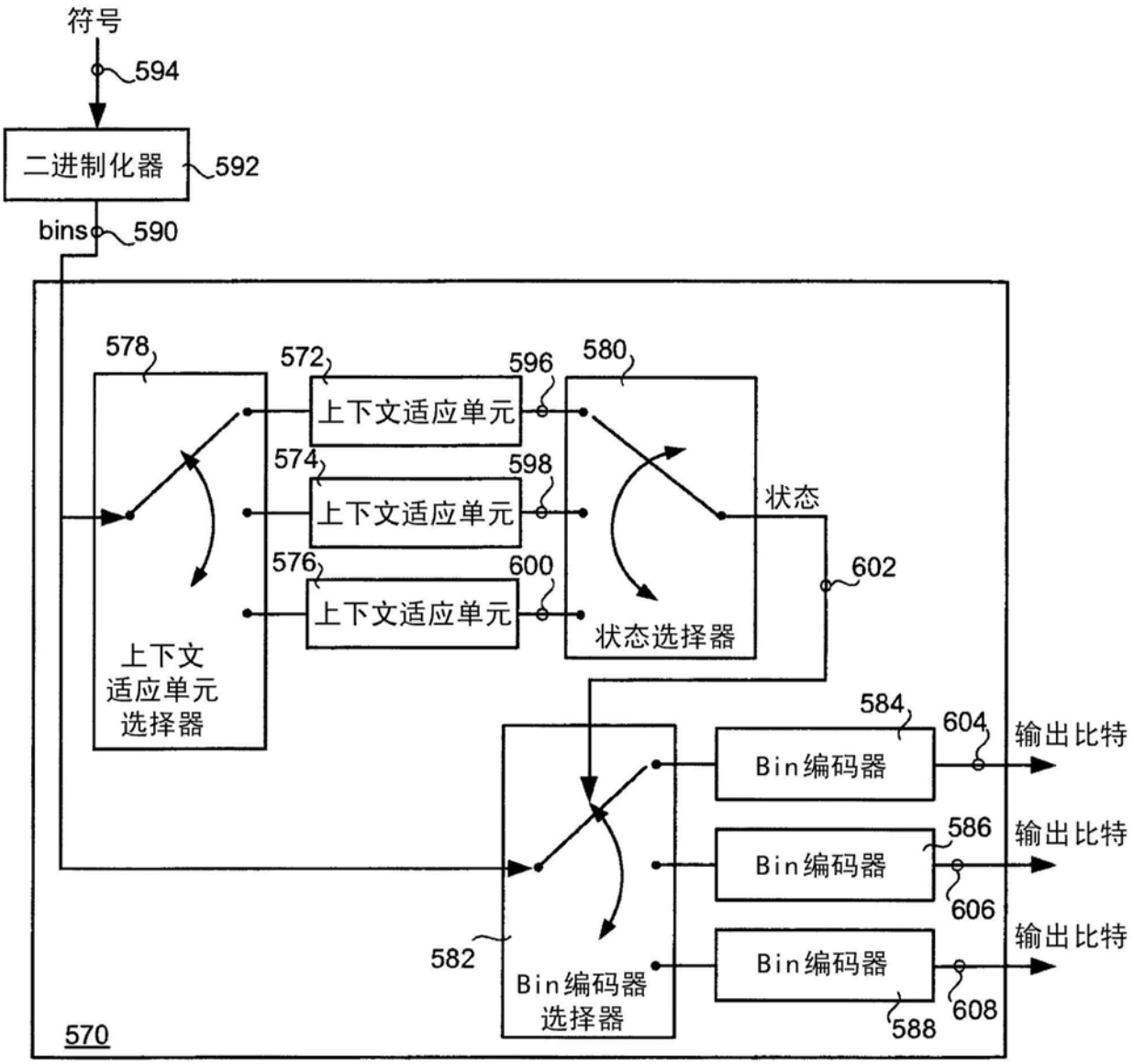

符号
594
二进制化器
592
bins
590
578
572
596
580
上下文适应单元
574
598
上下文适应单元
状态
576
600
602
上下文适应单元
状态选择器
上下文
适应单元
选择器
584
604
输出比特
Bin编码器
586
输出比特
Bin编码器
606
582
输出比特
Bin编码器
Bin编码器
选择器
608
588
570

图19

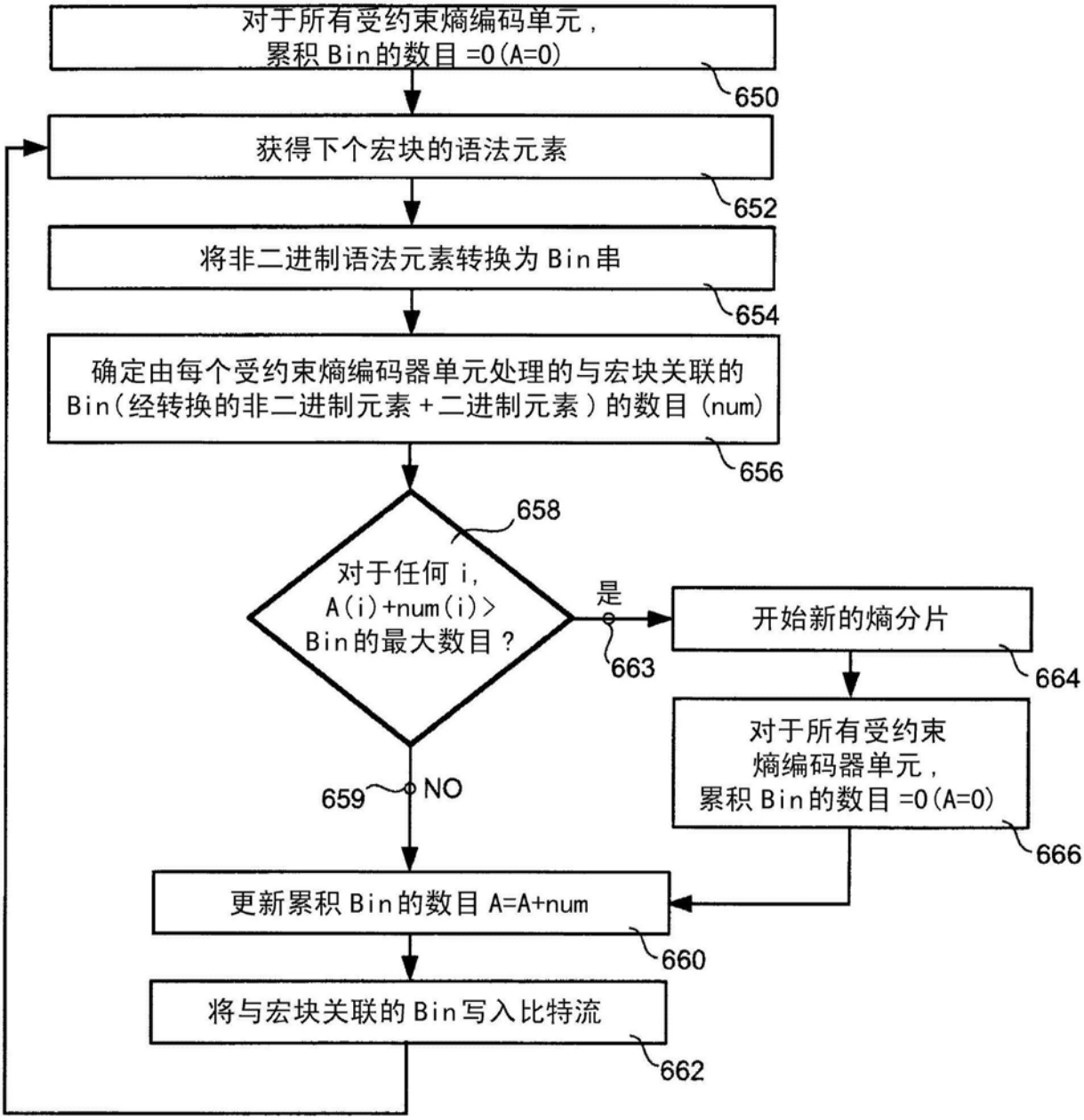

对于所有受约束熵编码单元，
累积 Bin的数目 =0(A=0)
650
获得下个宏块的语法元素
652
将非二进制语法元素转换为 Bin串
654
确定由每个受约束熵编码器单元处理的与宏块关联的
Bin(经转换的非二进制元素 +二进制元素)的数目 (num)
656
658
对于任何 i,
A(i)+num(i)>
Bin的最大数目？
是
663
开始新的熵分片
664
对于所有受约束
熵编码器单元，
累积 Bin的数目 =0(A=0)
666
659
NO
更新累积 Bin的数目 A=A+num
660
将与宏块关联的 Bin写入比特流
662

图20

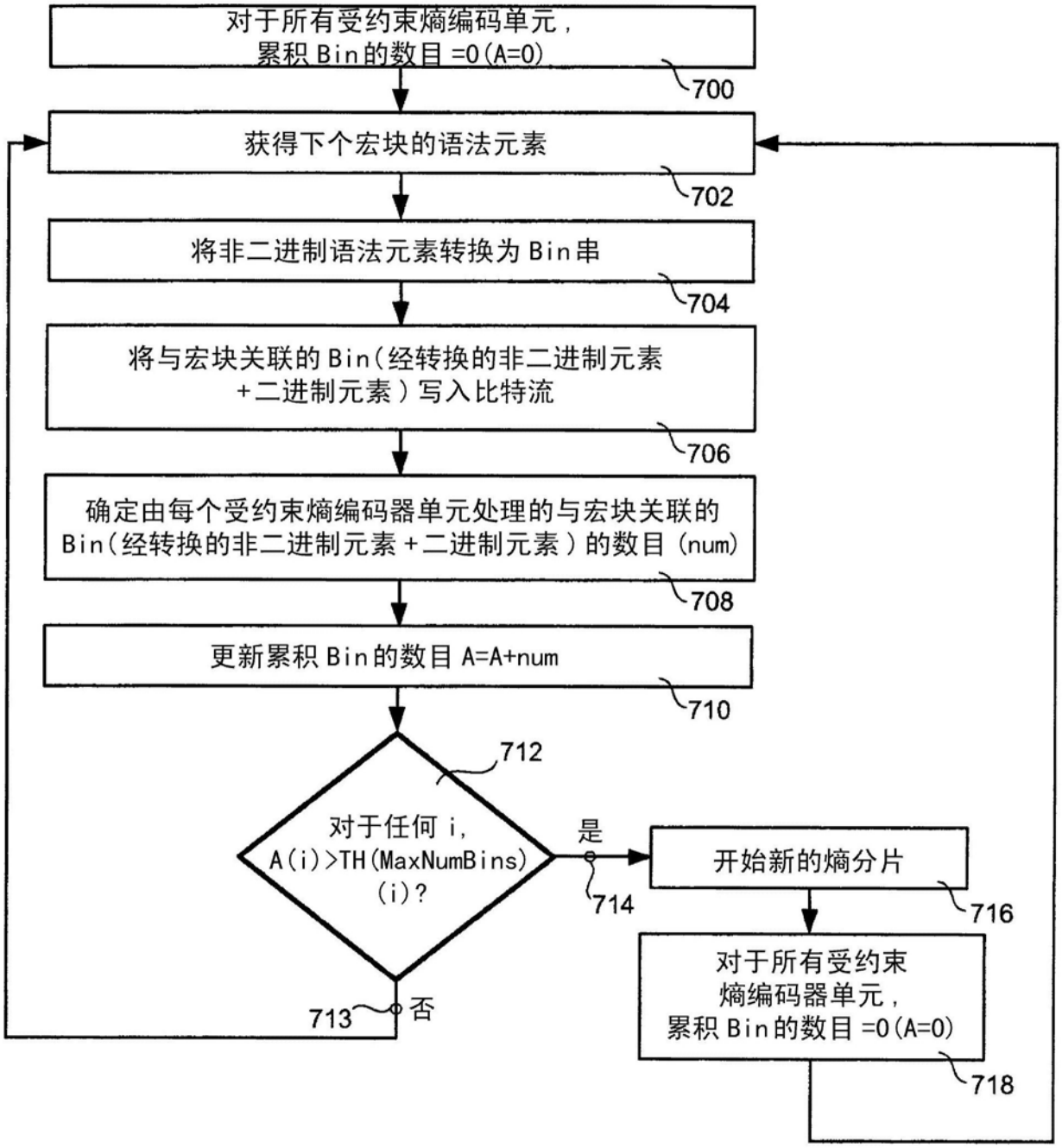

对于所有受约束熵编码单元，
累积 Bin的数目 =0(A=0)
700
获得下个宏块的语法元素
702
将非二进制语法元素转换为 Bin串
704
将与宏块关联的 Bin(经转换的非二进制元素
+二进制元素）写入比特流
706
确定由每个受约束熵编码器单元处理的与宏块关联的
Bin(经转换的非二进制元素 +二进制元素）的数目 (num)
708
更新累积 Bin的数目 A=A+num
710
712
对于任何 i,
A(i)>TH(MaxNumBins)
(i)?
是
714
开始新的熵分片
716
对于所有受约束
熵编码器单元，
累积 Bin的数目 =0(A=0)
718
713
否

图21

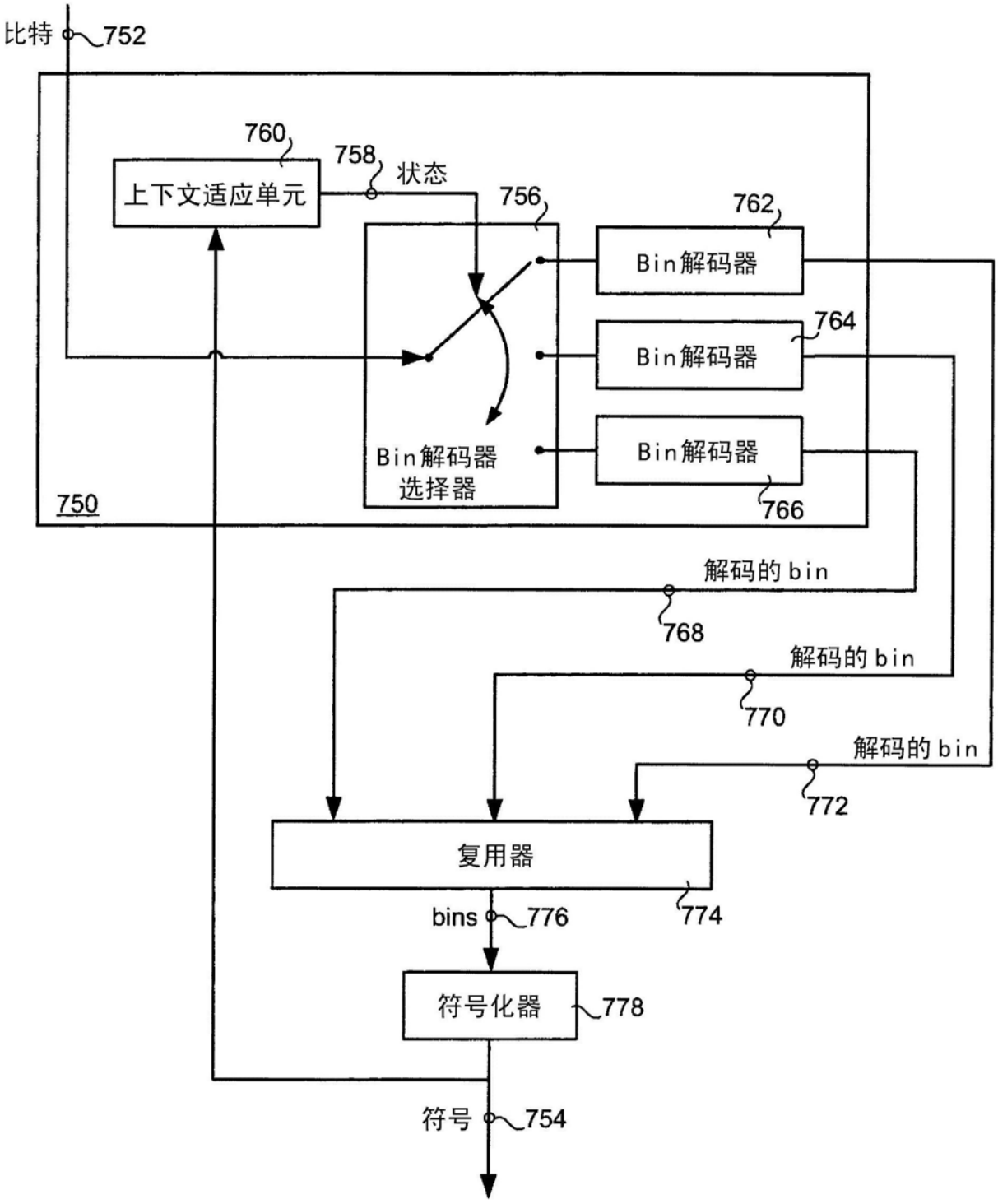

比特
752
760
上下文适应单元
758
状态
756
762
Bin解码器
764
Bin解码器
Bin解码器
选择器
Bin解码器
766
750
解码的bin
768
解码的bin
770
解码的bin
772
复用器
774
bins
776
符号化器
778
符号
754

图22

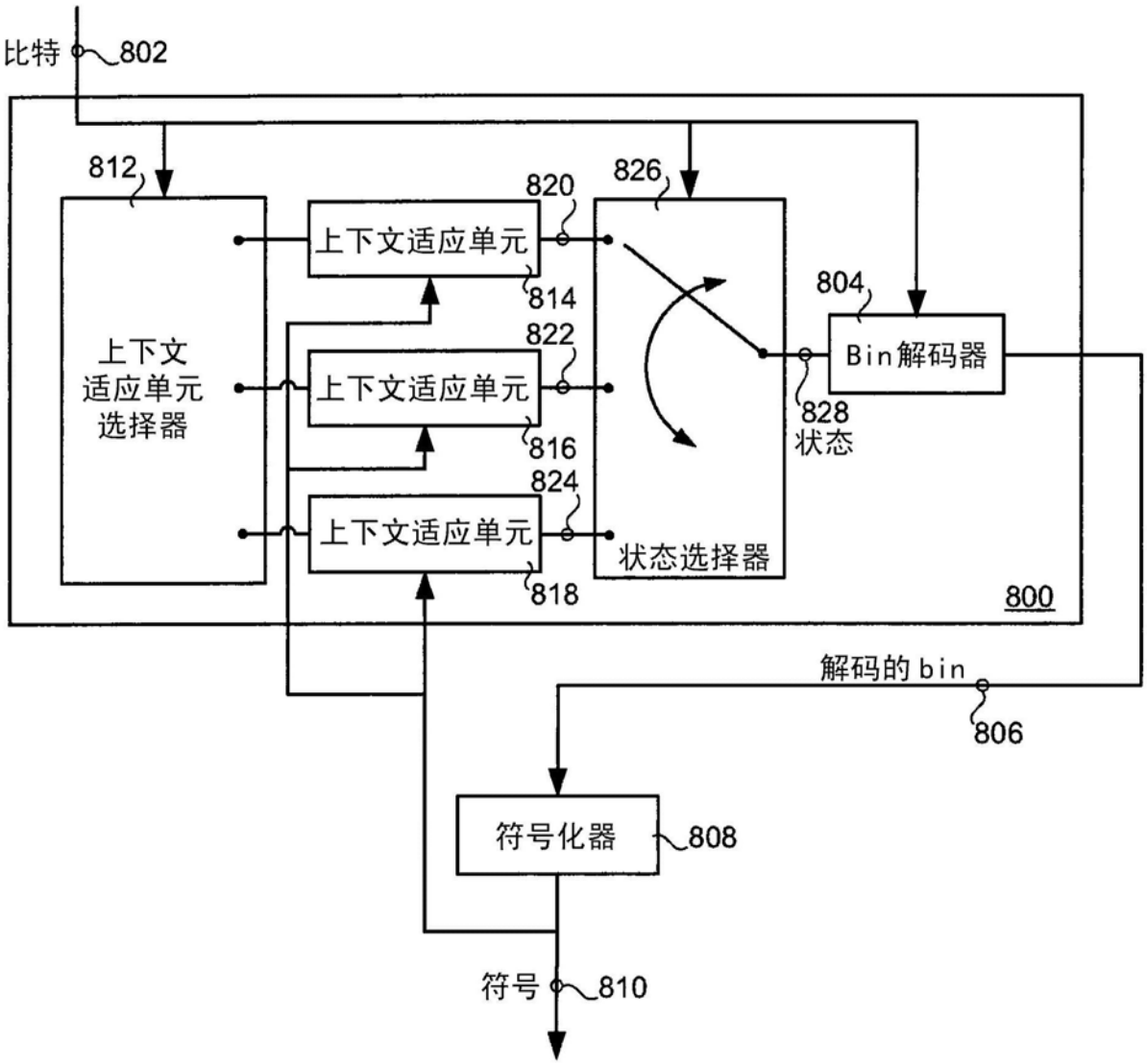

比特
802
812
820
826
上下文适应单元
814
804
Bin解码器
上下文
适应单元
选择器
822
上下文适应单元
816
828
状态
824
上下文适应单元
状态选择器
818
800
解码的bin
806
符号化器
808
符号
810

图23

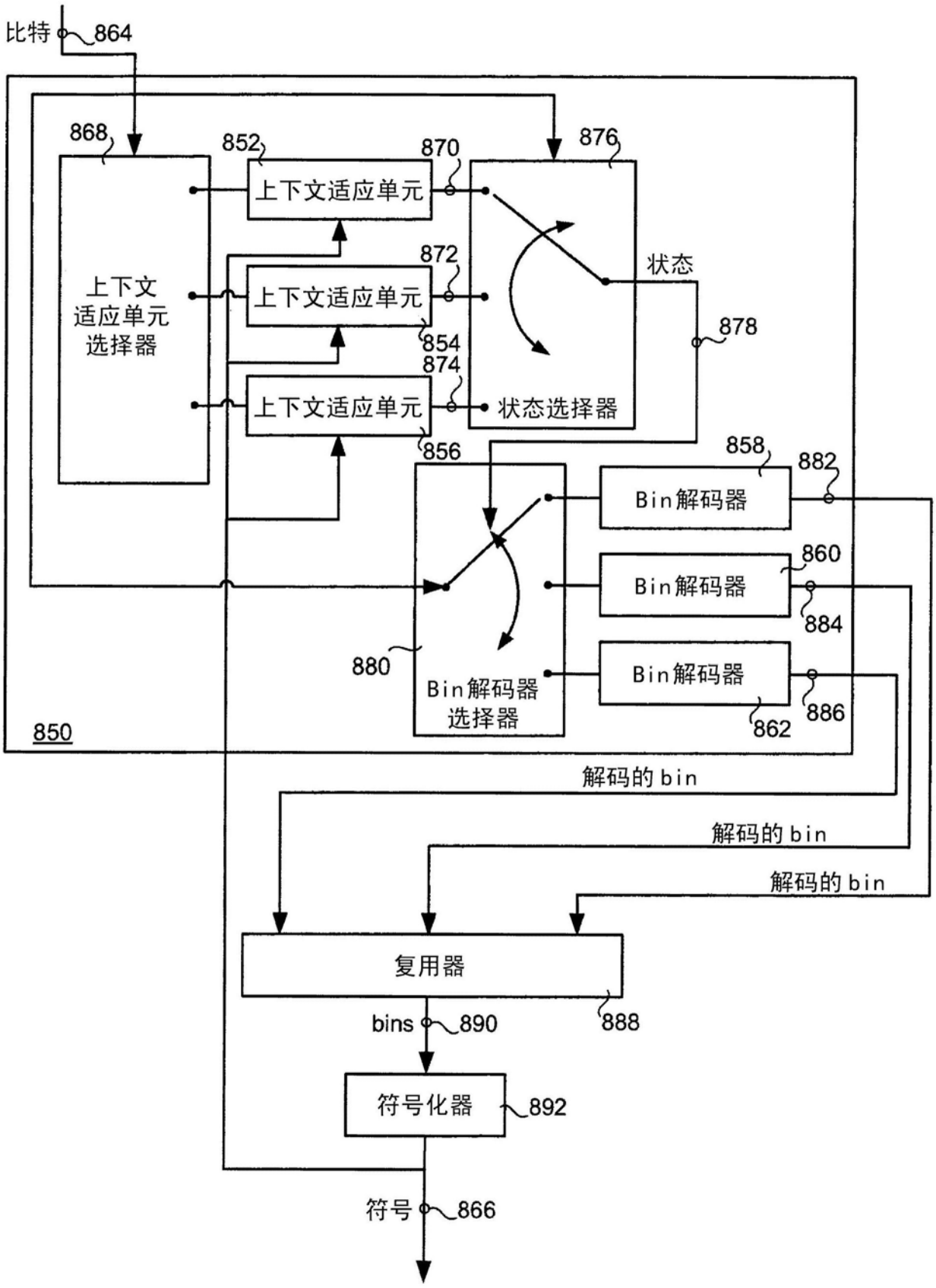

比特
864
868
852
870
876
上下文适应单元
上下文适应单元选择器
872
上下文适应单元
状态
878
854
874
上下文适应单元
状态选择器
856
858
882
Bin解码器
860
Bin解码器
884
880
Bin解码器选择器
Bin解码器
886
862
850
解码的bin
解码的bin
解码的bin
复用器
888
bins
890
符号化器
892
符号
866

图24

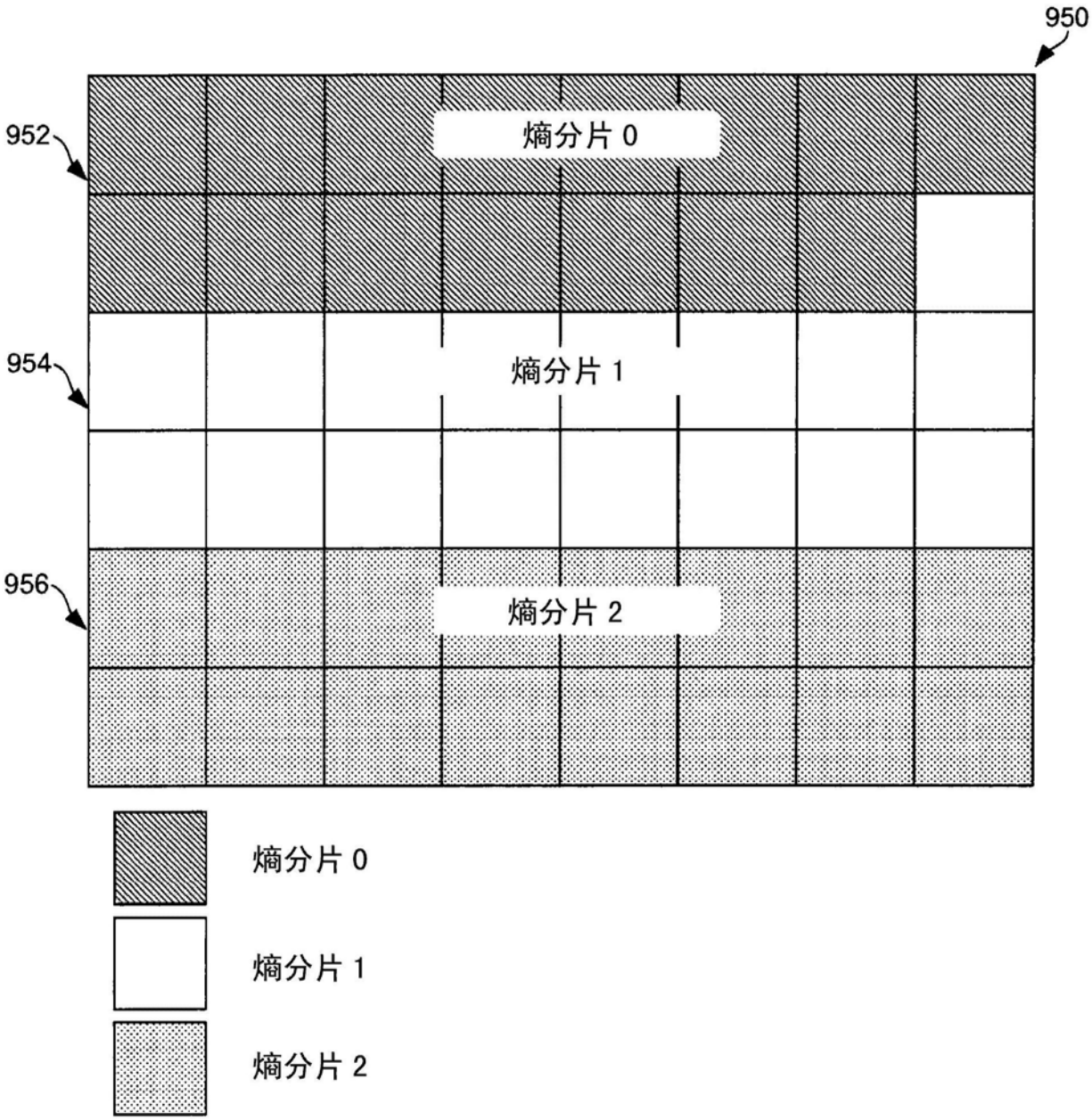


图25

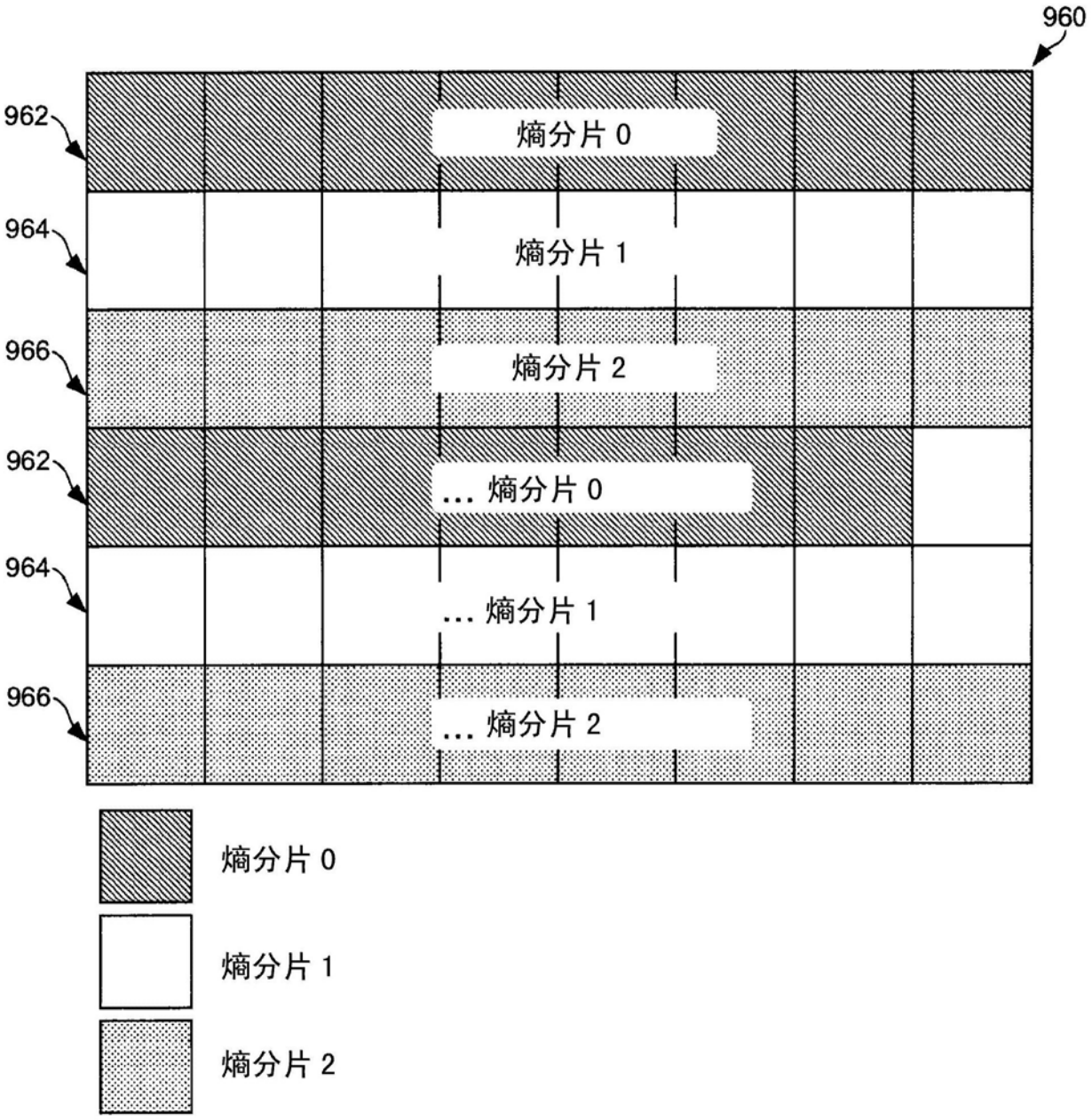

960
962
熵分片 0
964
熵分片 1
966
熵分片 2
962
... 熵分片 0
964
... 熵分片 1
966
... 熵分片 2
熵分片 0
熵分片 1
熵分片 2

图26

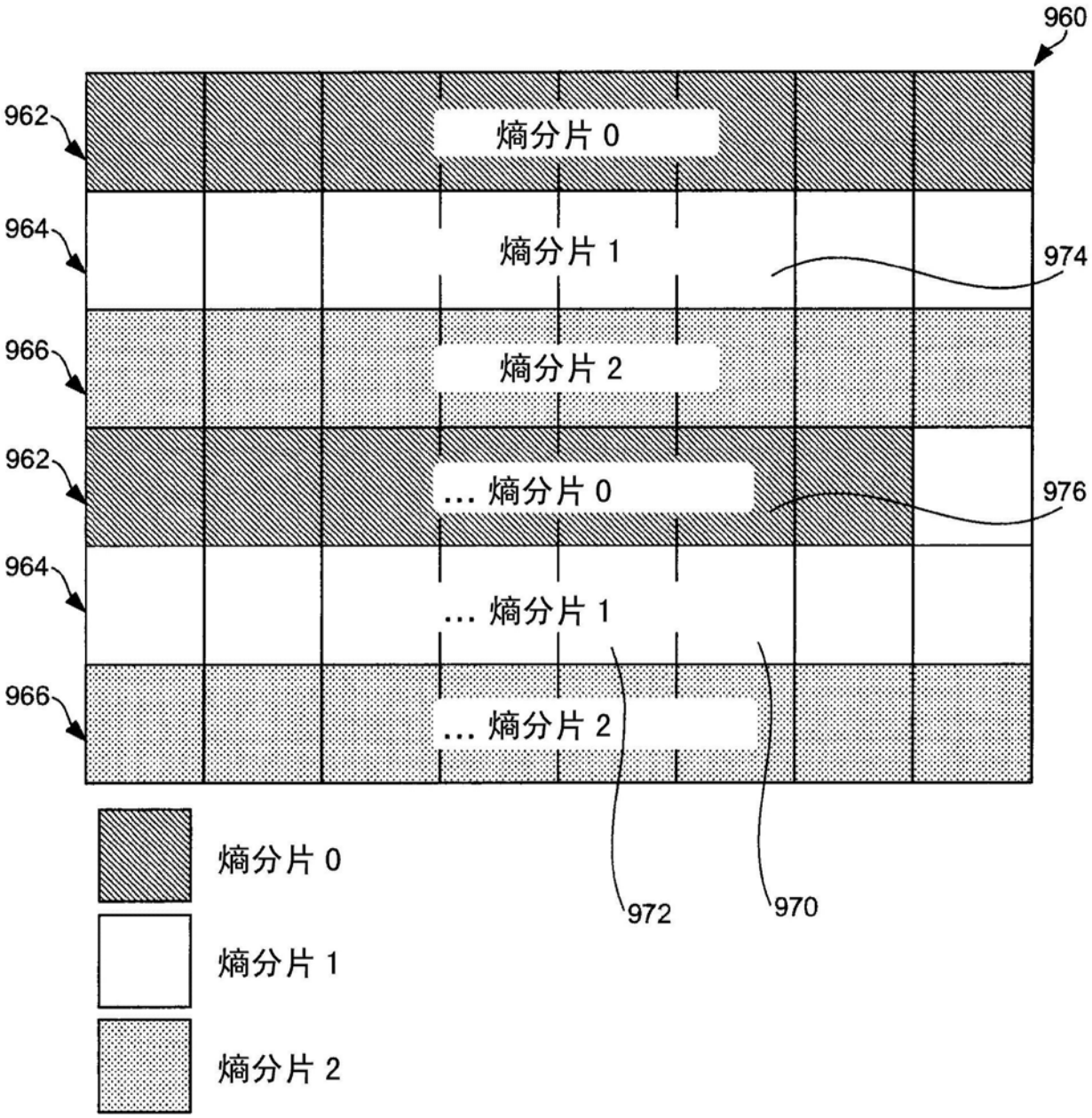

960
962
964
966
962
964
966
熵分片 0
熵分片 1
熵分片 2
... 熵分片 0
... 熵分片 1
... 熵分片 2
974
976
972
970
熵分片 0
熵分片 1
熵分片 2

图27

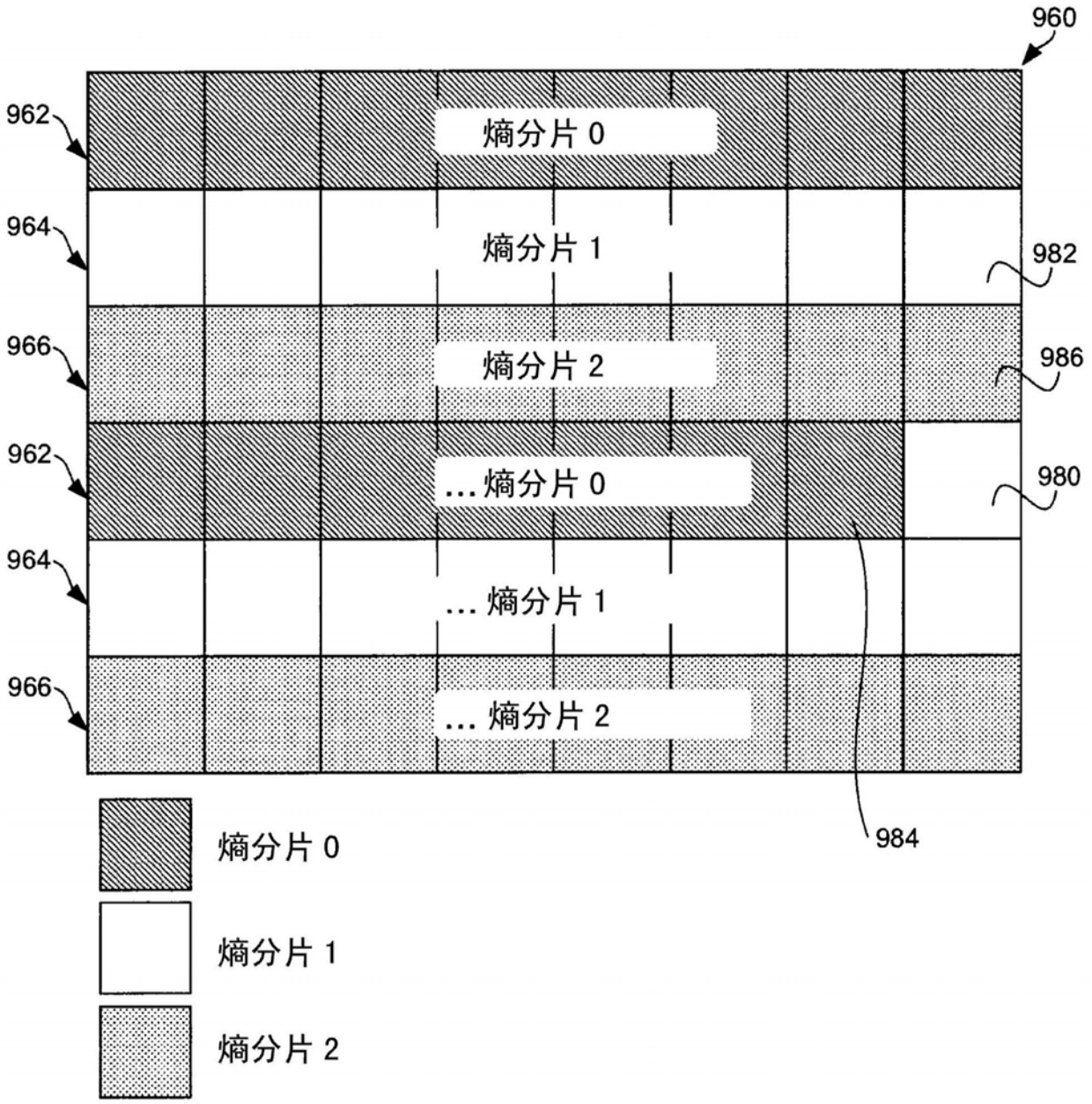

960
962
964
966
962
964
966
熵分片 0
熵分片 1
熵分片 2
... 熵分片 0
... 熵分片 1
... 熵分片 2
982
986
980
984
熵分片 0
熵分片 1
熵分片 2

图28

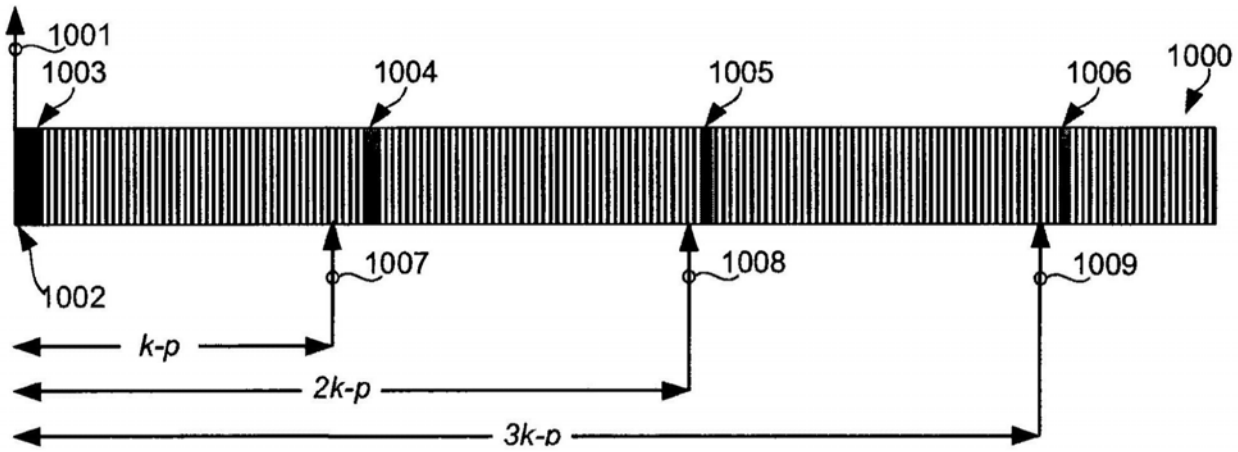

1001
1003
1004
1005
1006
1000
1002
1007
1008
1009
k-p
2k-p
3k-p

图29

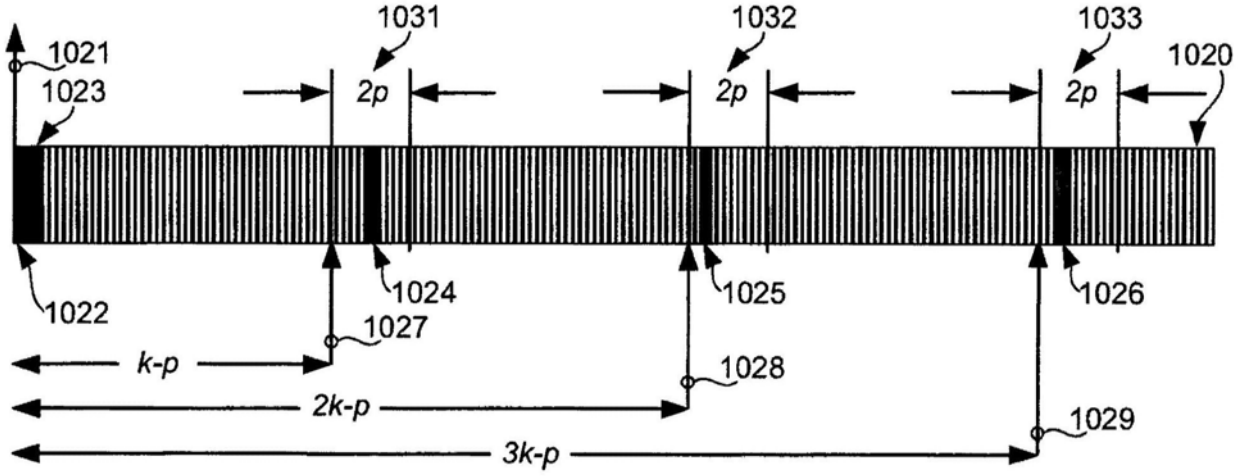

1031
1032
1033
1021
1023
2p
2p
2p
1020
1022
1024
1025
1026
1027
k-p
1028
2k-p
1029
3k-p

图30

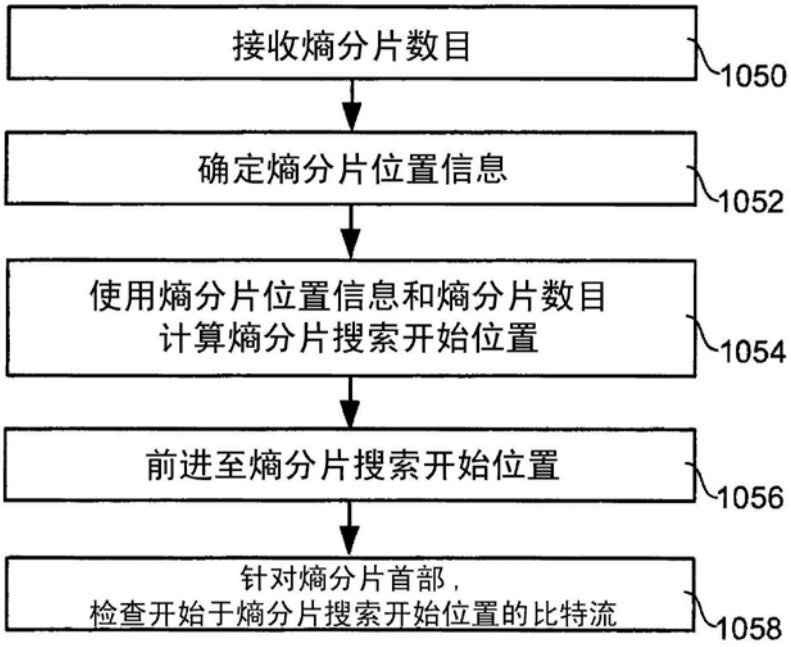

接收熵分片数目
1050
确定熵分片位置信息
1052
使用熵分片位置信息和熵分片数目
计算熵分片搜索开始位置
1054
前进至熵分片搜索开始位置
1056
针对熵分片首部，
检查开始于熵分片搜索开始位置的比特流
1058

图31

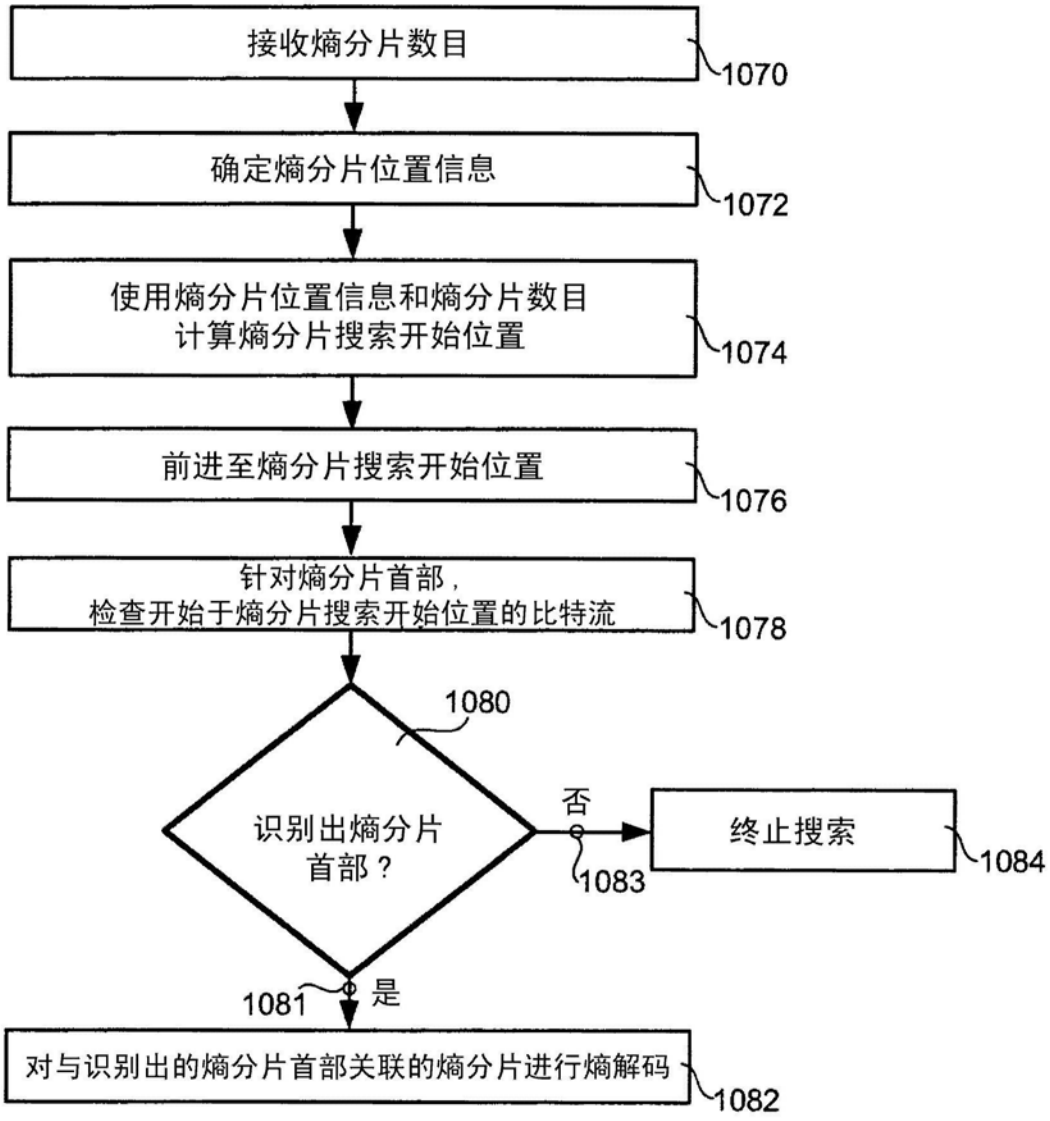

接收熵分片数目
1070
确定熵分片位置信息
1072
使用熵分片位置信息和熵分片数目
计算熵分片搜索开始位置
1074
前进至熵分片搜索开始位置
1076
针对熵分片首部，
检查开始于熵分片搜索开始位置的比特流
1078
1080
识别出熵分片
首部？
否
1083
终止搜索
1084
1081
是
对与识别出的熵分片首部关联的熵分片进行熵解码
1082

图32

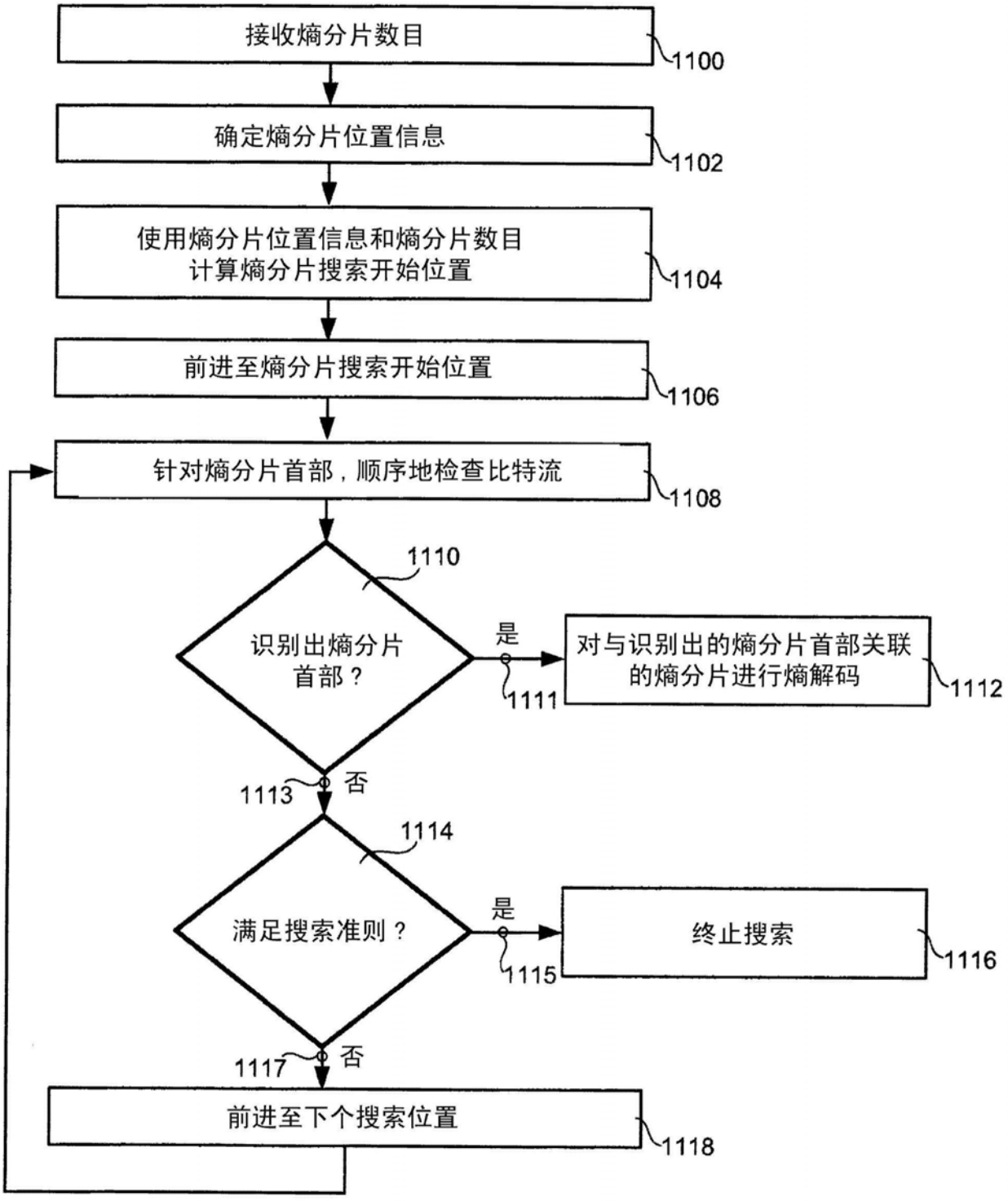

接收熵分片数目
1100
确定熵分片位置信息
1102
使用熵分片位置信息和熵分片数目
计算熵分片搜索开始位置
1104
前进至熵分片搜索开始位置
1106
针对熵分片首部，顺序地检查比特流
1108
1110
识别出熵分片
首部？
是
1111
对与识别出的熵分片首部关联
的熵分片进行熵解码
1112
1113
否
1114
满足搜索准则？
是
1115
终止搜索
1116
1117
否
前进至下个搜索位置
1118

图33

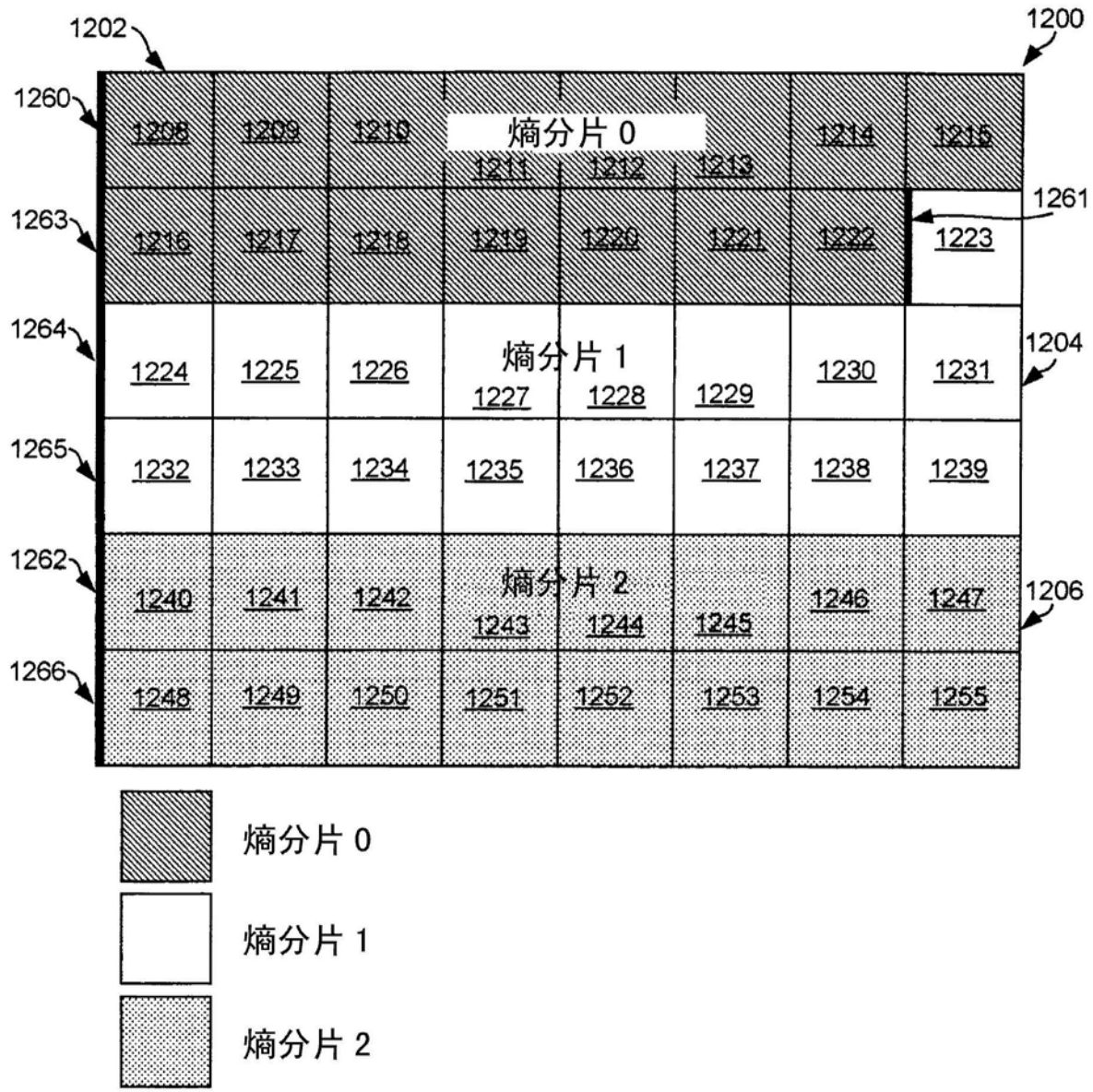

1202
1200
1260
1208
1209
1210
熵分片 0
1211
1212
1213
1214
1215
1263
1216
1217
1218
1219
1220
1221
1222
1261
1223
1264
1224
1225
1226
熵分片 1
1227
1228
1229
1230
1231
1204
1265
1232
1233
1234
1235
1236
1237
1238
1239
1262
1240
1241
1242
熵分片 2
1243
1244
1245
1246
1247
1206
1266
1248
1249
1250
1251
1252
1253
1254
1255
熵分片 0
熵分片 1
熵分片 2

图34