(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 119135887 A
(43)申请公布日 2024.12.13

(21)申请号 202411377004.1

(22)申请日 2019.05.28

(30)优先权数据
18305693.6 2018.06.07 EP
18305849.4 2018.07.02 EP

(62)分案原申请数据
201980045597.2 2019.05.28

(71)申请人 交互数字VC控股公司
地址 美国特拉华州

(72)发明人 F·莱林内克 F·加尔平
T·波伊里尔 E·弗朗索瓦

(74)专利代理机构 北京市柳沈律师事务所 11105
专利代理师 赵碧洋

(51)Int.Cl.
H04N 19/105(2014.01)
H04N 19/176(2014.01)
H04N 19/70(2014.01)
H04N 19/119(2014.01)
H04N 19/96(2014.01)
H04N 19/11(2014.01)
H04N 19/117(2014.01)
H04N 19/82(2014.01)
H04N 19/577(2014.01)

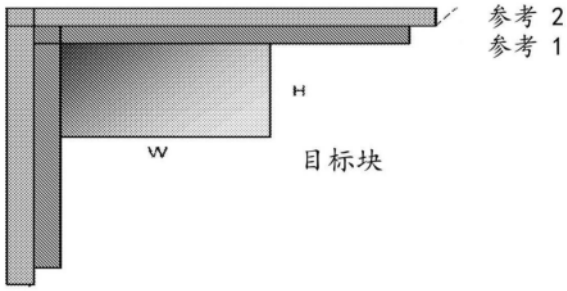
权利要求书1页 说明书24页 附图6页

(54)发明名称

用于解码视频的画面数据的方法、设备和存储介质

(57)摘要

本申请涉及用于解码视频的画面数据的方法、设备和存储介质。一种用于解码视频的画面数据的方法,所述方法包括,针对所述视频的块:获得编码的画面数据;基于多个参考线获得指示帧内预测的使用的标志,参考线包括位于所述块的左侧的左参考线和位于所述块上方的顶部参考线;基于所述标志的值使用在所述多个参考线中选择的一个或多个参考线来预测所述块;以及基于所预测的块来解码画面数据。



CN 119135887 A

1.一种用于解码视频的画面数据的方法,所述方法包括,针对所述视频的块:

获得编码的画面数据;

基于多个参考线获得指示帧内预测的使用的标志,参考线包括位于所述块的左侧的左参考线和位于所述块上方的顶部参考线;

基于所述标志的值使用在所述多个参考线中选择的一个或多个参考线来预测所述块;以及

基于所预测的块来解码画面数据。

2.根据权利要求1所述的方法,其中,第一参考线包括围绕所述块的第一线和第一列,所述第一线位于所述块的上方并且第二列位于所述块的左侧,并且第二参考线包括围绕所述块的第二线和第二列,所述第二线位于所述第一线的上方并且所述第二列位于所述第一列的左侧。

3.根据权利要求2所述的方法,其中,另外的参考线包括位于先前的线上方的另外的线和位于先前的列的左侧的另外的列。

4.根据权利要求2所述的方法,其中,所述预测作为来自所述第一参考线和所述第二参考线的加权平均值来执行。

5.根据权利要求4所述的方法,其中,来自最近参考线的预测相比来自最远参考线的预测被赋予更高的权重。

6.根据权利要求4所述的方法,其中,所使用的权重是3和1。

7.一种用于解码视频的画面数据的设备,所述设备包括处理器,所述处理器被配置为,针对所述视频的块:

获得编码的画面数据;

基于多个参考线获得指示帧内预测的使用的标志,参考线包括位于所述块的左侧的左参考线和位于所述块上方的顶部参考线;

基于所述标志的值使用在所述多个参考线中选择的一个或多个参考线来预测所述块;以及

基于所预测的块来解码画面数据。

8.一种计算机可读存储介质,包括指令,所述指令在由计算机执行时使得所述计算机执行根据权利要求1所述的方法。

用于解码视频的画面数据的方法、设备和存储介质

[0001] 本申请是申请号为201980045597.2、申请日为2019年05月28日、发明名称为“用于视频编码或解码的语法元素”的发明专利申请的分案申请。

技术领域

[0002] 本实施例中的至少一个一般涉及用于视频编码或解码的语法元素。

背景技术

[0003] 为了实现高压缩效率,图像和视频编码方案通常采用预测和变换来利用视频内容中的空间和时间冗余。一般地,使用帧内或帧间预测来利用帧内或帧间相关性,然后对原始块和预测块之间的差(经常表示为预测误差或预测残差)进行变换、量化和熵编码。为了重构视频,通过与熵编码、量化、变换和预测对应的逆处理来解码压缩数据。

发明内容

[0004] 根据至少一个实施例的第一方面,提出了一种用于解码视频的画面数据的方法,所述方法包括,针对所述视频的块:获得编码的画面数据;基于多个参考线获得指示帧内预测的使用的标志,参考线包括位于所述块的左侧的左参考线和位于所述块上方的顶部参考线;基于所述标志的值使用在所述多个参考线中选择的一个或多个参考线来预测所述块;以及基于所预测的块来解码画面数据。

[0005] 根据至少一个实施例的第二方面,提出了一种用于解码视频的画面数据的设备,所述设备包括处理器,所述处理器被配置为,针对所述视频的块:获得编码的画面数据;基于多个参考线获得指示帧内预测的使用的标志,参考线包括位于所述块的左侧的左参考线和位于所述块上方的顶部参考线;基于所述标志的值使用在所述多个参考线中选择的一个或多个参考线来预测所述块;以及基于所预测的块来解码画面数据。

[0006] 根据至少一个实施例的第三方面,提出了一种计算机可读存储介质,包括指令,所述指令在由计算机执行时使得所述计算机执行上述用于解码视频的画面数据的方法。

附图说明

[0007] 图1例示视频编码器100(诸如,高效视频编码(HEVC)编码器)的示例。

[0008] 图2例示视频解码器200(例如,HEVC解码器)的示例的框图。

[0009] 图3例示压缩域中的编码树单元和编码树的示例。

[0010] 图4例示将CTU划分为编码单元、预测单元和变换单元的示例。

[0011] 图5例示四叉树加二叉树(QTBT)CTU表示的示例。

[0012] 图6例示编码单元分区的示例性扩展集合。

[0013] 图7例示具有两个参考层的帧内预测模式的示例。

[0014] 图8例示用于双向照度补偿的补偿模式的示例实施例。

[0015] 图9例示根据示例实施例的bt-split-flag的解释。

[0016] 图10例示其中实现各个方面和实施例的系统的示例的框图。

[0017] 图11例示根据实施例的使用新编码工具的编码方法的示例的流程图。

[0018] 图12例示根据实施例的使用新编码工具的解码方法的一部分的示例的流程图。

具体实施方式

[0019] 在至少一个实施例中,以下描述的新编码工具的使用导致编码效率的改进。在示例实施例中,这些编码工具的有效信令正在将表示用于编码的编码工具的信息例如从编码器设备传送到接收器设备(例如,解码器或显示器),使得适当的工具用于解码阶段。这些工具包括新的分区模式,新的帧内预测模式,样本自适应偏移的改进的灵活性以及双向预测块的新的照度补偿模式。因此,提出的聚集多个编码工具的新语法提供更有效的视频编码。

[0020] 图1例示视频编码器100(诸如,高效视频编码(HEVC)编码器)的示例。图1还可以例示其中对HEVC标准作出改进的编码器或采用类似于HEVC的技术的编码器,诸如由JVET(联合视频探索小组)开发的JEM(联合探索模型)编码器。

[0021] 在本申请中,术语“重构”和“解码”可以互换使用,术语“编码(encoded)”或“编码(coded)”可以互换使用,并且术语“图像”,“画面”和“帧”可以互换使用。通常,但是不一定,在编码器侧使用术语“重构”,而在解码器侧使用“解码”。

[0022] 在编码之前,视频序列可以经历预编码处理(101)。例如,对输入颜色画面应用颜色变换(例如,从RGB 4:4:4到YCbCr 4:2:0的转换),或者对输入画面分量执行重新映射,以获得对压缩更有弹性的信号分布(例如,使用颜色分量之一的直方图均衡化)来执行。元数据可以与预处理相关联,并附加到比特流。

[0023] 在HEVC中,为了编码具有一个或多个画面的视频序列,画面被分区(102)成一个或多个条带,其中每个条带可包括一个或多个条带片段。条带片段被组织为编码单元、预测单元和变换单元。HEVC规范区分“块”和“单元”,其中“块”针对样本阵列中的特定区域(例如,亮度,Y),而“单元”包括所有编码的颜色分量的共位块(Y,Cb,Cr或单色)、语法元素以及与块相关联的预测数据(例如,运动矢量)。

[0024] 对于HEVC中的编码,将画面分区为具有可配置尺寸的正方形形状的编码树块(CTB),并且将连续的一组编码树块分组为条带。编码树单元(CTU)包含编码的颜色分量的CTB。CTB是分区成编码块(CB)的四叉树的根,并且编码块可以被分区为一个或多个预测块(PB)并形成分区成变换块(TB)的四叉树的根。对应于编码块、预测块和变换块,编码单元(CU)包括预测单元(PU)和树形结构的一组变换单元(TU),PU包括所有颜色分量的预测信息,以及TU包括每个颜色分量的残差编码语法结构。亮度分量的CB,PB和TB的尺寸应用于对应的CU,PU和TU。在本申请中,术语“块”可以用于例如指代CTU,CU,PU,TU,CB,PB和TB中的任何一个。另外,“块”也可以用于指代H.264/AVC或其他视频编码标准中规定的宏块和分区,并且更一般地,指代各种尺寸的数据的阵列。

[0025] 在编码器100的示例中,画面由编码器元件编码,如下所述。要编码的画面以CU为单元处理。使用帧内或帧间模式编码每个CU。当CU以帧内模式编码时,其执行帧内预测(160)。在帧间模式中,执行运动估计(175)和运动补偿(170)。编码器决定(105)帧内模式或帧间模式中的哪一个用于编码CU,并且通过预测模式标志指示帧内/帧间决定。通过从原始图像块中减去(110)预测块来计算预测残差。

[0026] 帧内模式下的CU是根据相同条带内的重构邻近样本预测的。HEVC中可用一组35个帧内预测模式,包括DC、平面和33个角度预测模式。从与当前块相邻的行和列重构帧内预测参考。使用来自先前重构的块的可用样本,参考在水平和垂直方向上延伸超过块尺寸的两倍。当将角度预测模式用于帧内预测时,可以沿着由角度预测模式指示的方向复制参考样本。

[0027] 可以使用两个不同选项编码当前块的可应用亮度帧内预测模式。如果可应用模式被包括在三个最可能模式(MPM)的构造列表中,则由MPM列表中的索引用信号通知该模式。否则,通过模式索引的固定长度二元化来用信号通知该模式。从顶部和左侧邻近块的帧内预测模式推导出三个最可能模式。

[0028] 对于帧间CU,对应编码块还被分区为一个或多个预测块。在PB级上执行帧间预测,并且对应PU包含关于如何执行帧间预测的信息。可以以两种方法,即“合并模式”和“高级运动矢量预测(AMVP)”来用信号通知运动信息(即,运动矢量和参考画面索引)。

[0029] 在合并模式中,视频编码器或解码器基于已经编码的块来组装候选列表,并且视频编码器用信号通知用于候选列表中的候选之一的索引。在解码器侧,基于用信号通知的候选重构运动矢量(MV)和参考画面索引。

[0030] 在AMVP中,视频编码器或解码器基于从已经编码的块确定的运动矢量来组装候选列表。然后,视频编码器用信号通知候选列表中的索引,以标识运动矢量预测量(MVP)并且用信号通知运动矢量差(MVD)。在解码器侧,运动矢量(MV)被重构为MVP+MVD。可应用参考画面索引也在用于AMVP的PU语法中显式编码。

[0031] 然后,变换(125)和量化(130)预测残差,包括至少一个用于适配色度量化参数的实施例,如下所述。变换一般基于可分离的变换。例如,首先在水平方向上然后在垂直方向上应用DCT变换。在最近的编解码器(诸如,JEM)中,在两个方向上使用的变换可能不同(例如,一个方向上的DCT,另一方向上的DST),这导致各种各样的2D变换,而在以前的编解码器中,通常限制给定块尺寸的各种2D转换。

[0032] 量化的变换系数以及运动矢量和其他语法元素被熵编码(145)以输出比特流。编码器还可以跳过该变换,并以4x4 TU为基础将量化直接应用于未变换的残差信号。编码器还可以绕过变换和量化两者,即,在不应用变换或量化处理的情况下直接编码残差。在直接PCM编码中,不应用任何预测,并且将编码单元样本直接编码到比特流中。

[0033] 编码器对编码块进行解码以提供进一步预测的参考。量化的变换系数被去量化(140)和逆变换(150)以解码预测残差。组合(155)解码的预测残差和预测块,重构画面块。例如,将环内滤波器(165)应用于重构画面,以执行去块/SAO(样本自适应偏移)滤波以减少编码伪像。经滤波的图像存储在参考画面缓冲器(180)处。

[0034] 图2例示诸如HEVC解码器之类的视频解码器200的示例的块图。在解码器200的示例中,比特流由解码器元件解码,如下所述。视频解码器200一般执行与图1中描述的编码通道相反的解码通道,其执行视频解码作为编码视频数据的一部分。图2还可以例示其中对HEVC标准作出改进的解码器或采用类似于HEVC的技术的解码器,例如JEM解码器。

[0035] 具体地,解码器的输入包括可由视频编码器100生成的视频比特流。首先熵解码(230)该比特流,以获得变换系数、运动矢量、画面分区信息和其他编码信息。画面分区信息指示CTU的尺寸,以及将CTU分割为CU的方式,并且在适用时可能分割为PU。因此,解码器可

以根据解码的画面分区信息将画面划分(235)成CTU,并且将每个CTU划分成CU。去量化(240)(包括用于适配下面描述的色度量化参数的至少一个实施例)和逆变换(250)变换系数以解码预测残差。

[0036] 组合(255)解码的预测残差和预测块,重构图像块。可以从帧内预测(260)或运动补偿预测(即,帧间预测)(275)获得(270)预测块。如上所述,AMVP和合并模式技术可以用于推导出运动矢量用于运动补偿,其可以使用插值滤波器来计算参考块的亚整数(sub-integer)样本的插值。将环内滤波器(265)应用于重构的图像。经滤波的图像存储在参考画面缓冲器(280)处。

[0037] 解码的画面可以进一步经历后解码处理(285),例如,逆颜色变换(例如,从YCbCr 4:2:0到RGB 4:4:4的转换)或执行对预编码处理(101)中执行的重新映射处理的逆过程的逆重新映射。后解码处理可以使用在预编码处理中推导出并在比特流中用信号通知的元数据。

[0038] 图3例示压缩域中的编码树单元和编码树的示例。在HEVC视频压缩标准中,画面被划分为所谓的编码树单元(CTU),其尺寸典型为64x64、128x128或256x256像素。每个CTU由压缩域中的编码树表示。这是CTU的四叉树划分,其中每个叶片称为编码单元(CU)。

[0039] 图4例示将CTU划分为编码单元、预测单元和变换单元的示例。然后,为每个CU给予一些帧内或帧间预测参数(“预测信息”)。为此,将其在空间上分区为一个或多个预测单元(PU),每个PU均分配有一些预测信息。帧内或帧间编码模式在CU级别上分配。

[0040] 新兴的视频压缩工具包括提出的压缩域中的编码树单元表示以更灵活的方式表示画面数据。编码树的这种更灵活表示的优点是,与HEVC标准的CU/PU/TU布置相比,它提供增加的压缩效率。

[0041] 图5例示四叉树加二叉树(QTBT)CTU表示的示例。四叉树加二叉树(QTBT)编码工具提供这种增加的灵活性。QTBT在于其中编码单元可以以四叉树和二叉树的方式被分割的编码树。编码单元的分割通过率失真优化过程(确定具有最小率失真成本的CTU的QTBT表示)在编码器侧决定。在QTBT技术中,CU具有正方形或矩形形状。编码单元的尺寸始终是2的幂,典型从4到128。除了用于编码单元的各种矩形形状外,这种CTU表示与HEVC相比还具有以下不同特征。CTU的QTBT分解分为两个阶段:首先以四叉树的方式分割CTU,然后可以以二元方式进一步划分每个四叉树的叶片。这在图的右侧例示,其中实线表示四叉树分解阶段,并且虚线表示空间上嵌入四叉树叶片的二元分解。在帧内条带中,亮度和色度块分区结构被分离,并且被独立决定。不再采用CU被分区成预测单元或变换单元。换句话说,每个编码单元系统地由单个预测单元(2N×2N预测单元分区类型)和单个变换单元(不划分为变换树)组成。

[0042] 图6例示编码单元分区的示例性扩展集合。在非对称二元和树分割模式(ABT)中,将通过非对称二元分割模式之一被分割的尺寸(w,h)(宽度和高度)的矩形编码单元(例如HOR_UP(水平向上))将导致2个子编码单元,分别具有矩形尺寸$\left(w,\frac{h}{4}\right)$和$\left(w,\frac{3h}{4}\right)$。另外,可以使用CU的所谓的三叉树分区,从而导致图5中给出的可能分区的集合。三叉树在于在考虑的取向上(例如用于水平分割模式的HOR_TRIPLE)相对于父CU将CU分割为尺寸(1/4,1/2,1/4)的树的子CU。

[0043] 使用上述新拓扑的一个或多个实施例带来显著编码效率改进。

[0044] 图7例示具有两个参考层的帧内预测模式的示例。实际上,在改进的语法中考虑了新的帧内预测模式。第一个新的帧内预测模式被称为多参考帧内预测(MRIP)。该工具有助于将多个参考层用于块的帧内预测。典型地,将2个参考层用于帧内预测,其中每个参考层由左侧参考阵列和顶部参考阵列组成。每个参考层都是通过参考样本替换并且然后被预滤波而构建的。

[0045] 使用每个参考层,构建用于块的预测,典型如在HEVC或JEM中进行的那样。最终预测形成为由两个参考层得出的预测的加权平均。来自最接近参考层的预测比来自最远层的预测具有更高的权重。典型使用的权重是3和1。

[0046] 在预测处理的最后部分,可以使用多个参考层,以使用模式相关处理来平滑某些预测模式的边界样本。

[0047] 在示例实施例中,视频压缩工具还包括用于样本自适应偏移(SAO)的自适应块尺寸。该工具是HEVC中规定的环内滤波器。在HEVC中,SAO处理将一个块的重构样本分类为若干类别,并且属于某些类别的样本利用偏移来校正。SAO参数是按块编码的,或者只能从左侧或上面邻近块继承。

[0048] 通过定义SAO调色板模式,提出了进一步的改进。SAO调色板按块保留相同的SAO参数,但是这些块可以从所有其他块继承这些参数。通过拓宽一个块可能的SAO参数范围,这为SAO带来了更大的灵活性。SAO调色板由一组不同的SAO参数组成。对于每个块,索引被编码以指示要使用哪个SAO参数。

[0049] 图8例示用于双向照度补偿的补偿模式的示例实施例。照度补偿(IC)允许通过可能考虑空间或时间局部照度变化来校正经由运动补偿获得的块预测样本(SMC)。

[0050] $S_{IC}=a_i.S_{MC}+b_i$

[0051] 在双向预测的情况下,使用两个参考CU样本的样本估计IC参数,如图8描绘。首先,在两个参考模块之间估计IC参数(a,b),如下推导参考块和当前块之间的下一IC参数(ai,bi)i=0,1:

[0052] $$\begin{cases} a_0 = a.\alpha + (1-\alpha) \\ b_0 = \alpha.b \\ a_1 = \alpha + (1-\alpha)/a \\ b_1 = -b(1-\alpha)/a \end{cases}$$ 其中:$\alpha = \frac{poc_{cur}-poc_0}{poc_1-poc_0}$

[0053] 如果CU尺寸在宽度或高度上小于或等于8,则使用单向处理估计IC参数。当CU尺寸大于8时,通过选择使IC补偿参考块的平均值的差最小的IC参数,做出在使用从双向或单向处理推导出的IC参数之间的选择。使用双向照度补偿工具启用双向光流(BIO)。

[0054] 上面介绍的多个工具由视频编码器100选择和使用,并且需要由视频解码器200获得和使用。为此,表示这些工具使用的信息承载在由视频编码器生成的编码比特流中并且由视频解码器200以高级语法信息的形式获得。为此,在下面定义并描述对应的语法。聚集多个工具的该语法允许改进的编码效率。

[0055] 此语法结构使用HEVC语法结构作为基础,并包含一些附加语法元素。下表中以斜体粗体用信号通知并以灰色突出显示的语法元素对应于根据示例实施例的附加语法元素。应当注意,语法元素可以采用语法表中未示出的其他形式或名称,同时仍处理相同的功能。

语法元素可以驻留在不同的级别,例如,一些语法元素可以放置在序列参数集(SPS)中,而某些语法元素可以放置在画面参数集(PPS)中。

[0056] 下表1例示序列参数集(SPS),并且介绍了根据至少一个示例实施例插入该参数集中的新语法元素,更确切地:multi_type_tree_enabled_primary,log2_min_cu_size_minus2, log2_max_cu_size_minus4, log2_max_tu_size_minus2,sep_tree_mode_intra, multi_type_tree_enabled_secondary, sps_bdip_enabled_flag,sps_mrip_enabled_flag,use_erp_aqp_flag,use_high_perf_chroma_qp_table,abt_one_third_flag,sps_num_intra_mode_ratio.

[0057]

seq_parameter_set_rbsp( ) {	描述符
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_temporal_id_nesting_flag	u(1)
profile_tier_level(1, sps_max_sub_layers_minus1)	
sps_seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	u(1)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
conformance_window_flag	u(1)
if(conformance_window_flag) {	
conf_win_left_offset	ue(v)
conf_win_right_offset	ue(v)
conf_win_top_offset	ue(v)
conf_win_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
log2_max_pic_order_cnt_lsb_minus4	ue(v)
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1); i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_minus1[i]	ue(v)
sps_max_num_reorder_pics[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
multi_type_tree_enabled_primary	ue(v)
log2_min_cu_size_minus2	ue(v)
log2_max_cu_size_minus4	ue(v)
log2_max_tu_size_minus2	ue(v)
sep_tree_mode_intra	u(1)
if (sep_tree_mode_intra)	
multi_type_tree_enabled_secondary	ue(v)
sample_adaptive_offset_enabled_flag	u(1)
pcm_enabled_flag	u(1)

[0058]

if(pcm_enabled_flag) {	
pcm_sample_bit_depth_luma_minus1	u(4)
pcm_sample_bit_depth_chroma_minus1	u(4)

[0059]

log2_min_pcm_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_luma_coding_block_size	ue(v)
pcm_loop_filter_disabled_flag	u(1)
}	
num_short_term_ref_pic_sets	ue(v)
for(i = 0; i < num_short_term_ref_pic_sets; i++)	
st_ref_pic_set(i)	
long_term_ref_pics_present_flag	u(1)
if(long_term_ref_pics_present_flag) {	
num_long_term_ref_pics_sps	ue(v)
for(i = 0; i < num_long_term_ref_pics_sps; i++) {	
lt_ref_pic_poc_lsb_sps[i]	u(v)
used_by_curr_pic_lt_sps_flag[i]	u(1)
}	
}	
sps_temporal_mvp_enabled_flag	u(1)
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters( )	
use_imv	u(1)
fruc_merge_mode	u(1)
if(fruc_merge_mode) {	
idx_num_template	ue(v)
idx_num_template_ic	ue(v)
fruc_template_affine	u(1)
}	
mode_bilateral_TM	ue(v)
optical_flow_filtering	ue(v)
atmvp_flag	u(1)
sps_lic_enabled_flag	u(1)
sps_bidir_ic_present_flag	u(1)
sps_use_intra_emt	u(1)
sps_use_inter_emt	u(1)
sps_nsst_enabled_flag	u(1)
sps_cross_component_prediction_enabled_flag	u(1)
sps_intra_4tap_filter_enabled_flag	u(1)
sps_intra_boundary_filter_enabled_flag	u(1)
sps_obmc_flag	u(1)
if(sps_obmc_flag)	
obmc_blk_size	ue(v)
obmc_for_sub_block	u(1)
sps_affine_enabled_flag	u(1)
use_NL_Bil_flag	u(1)
sps_bdip_enabled_flag	u(1)
sps_mrip_enabled_flag	u(1)

[0060]

num_predicted_coef_signs	u(4)
mc_frame_pad	u(1)
use_erp_aqp_flag	u(1)
use_chroma_qp_table	
if(use_chroma_qp_table)	
use_high_perf_chroma_qp_table	u(1)
log2_intra131modes_min_area_minus6	ue(v)
log2_intra65modes_min_area_minus4	ue(v)
abt_one_third_flag	u(1)
sps_num_intra_mode_ratio	u(1)
if(sps_range_extension_flag)	
sps_range_extension( )	
if(sps_multilayer_extension_flag)	
sps_multilayer_extension( ) /* specified in Annex F */	
if(sps_3d_extension_flag)	
sps_3d_extension( ) /* specified in Annex I */	
if(sps_extension_5bits)	
while(more_rbsp_data( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

[0061] 表1:修改后的序列参数集

[0062] 新的语法元素定义如下:

[0063] -use_high_perf_chroma_qp_table:此语法元素规定色度QP表,用于推导与条带相关联的色度分量的解码中使用的QP,作为与所考虑的条带的色度分量相关联的基本色度QP的函数。来导出用于解码与条带相关的色度分量的QP。

[0064] -multi_type_tree_enabled_primary:此语法元素指示编码条带中允许的分区类型。在至少一个实现方式中,该语法元素在序列级别上(在SPS中)用信号通知,并且因此应用于使用该SPS的所有编码条带。例如,此语法元素的第一值允许四叉树和二叉树(QTBT)分区(图5的NO-SPLIT,QT-SPLIT,HOR和VER),第二值允许QTBT加三叉树(TT)分区(图5的NO-SPLIT,QT-SPLIT,HOR,VER,HOR_TRIPLE和VER_TRIPLE),第三值允许QTBT加非对称二叉树(ABT)分区(图5的NO-SPLIT,QT-SPLIT,HOR,VER,HOR-UP,HOR_DOWN,VER_LEFT和VER_RIGHT),第四值允许QTBT+TT+ABT分区(图5的所有分割情况)。

[0065] -sep_tree_mode_intra:此语法元素指示是否将单独的编码树用于亮度块和色度块。当sep_tree_mode_intra等于1时,亮度块和色度块使用独立的编码树,并且因此亮度块和色度块的分区是独立的。在至少一个实现方式中,该语法元素在序列级别上(在SPS中)用信号通知,并且因此应用于使用该SPS的所有编码条带。

[0066] -multi_type_tree_enabled_secondary:此语法元素指示分区的类型被允许用于编码条带中的色度块。该语法元素可以采用的值典型与语法元素multi_type_tree_enabled_primary的值相同。在至少一个实现方式中,该语法元素在序列级别上(在SPS中)用信号通知,并且因此应用于使用该SPS的所有编码条带。

[0067] -sps_bdip_enabled_flag:此语法元素指示在考虑的编码视频序列中包含的编码条带中允许JVET-J0022中描述的双向帧内预测。

[0068] -sps_mrip_enabled_flag:此语法元素指示多参考帧内预测工具用于包含在考虑的编码视频比特流中的编码条带的解码。

[0069] -use_erp_aqp_flag:此语法元素指示是否在编码条带中激活JVET-J0022中所述的VR360 ERP内容的空间自适应量化。

[0070] -abt_one_third_flag:此语法元素指示是否在编码条带中激活将非对称分区分为水平或垂直尺寸分别为初始CU的水平或垂直尺寸的1/3和2/3,或2/3和1/3的两个分区。

[0071] -sps_num_intra_mode_ratio:此语法元素规定如何推导用于块尺寸在宽度或高度上等于3的倍数的帧内预测的数量。

[0072] 另外,以下三个语法元素用于控制编码单元(CU)和变换单元(TU)的尺寸。在至少一个实现方式中,这些语法元素在序列级别上(在SPS中)用信号通知,并且因此应用于所有使用此SPS的编码条带。

[0073] -log2_min_cu_size_minus2规定最小CU尺寸。

[0074] -log2_max_cu_size_minus4规定最大CU尺寸。

[0075] -log2_max_tu_size_minus2规定最大TU尺寸。

[0076] 在变型实施例中,上面介绍的新语法元素没有引入到序列参数集中,而是在画面参数集中(PPS)。

[0077] 下表2例示序列参数集(SPS),并介绍根据至少一个示例实施例插入到此参数集中的新语法元素,更准确地:

[0078] -slice_sao_size_id:此语法元素规定对编码条带在其上应用SAO的块的尺寸。

[0079] -slice_bidir_ic_enable_flag:指示是否为编码条带激活双向照度补偿。

[0080]

slice_segment_header( ) {	描述符
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
if(!dependent_slice_segment_flag) {	
for(i = 0; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
slice_pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
st_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
if(num_long_term_ref_pics_sps > 0)	
num_long_term_sps	ue(v)
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_sps + num_long_term_pics; i++) {	
if(i < num_long_term_sps) {	
if(num_long_term_ref_pics_sps > 1)	
lt_idx_sps[i]	u(v)
} else {	
poc_lsb_lt[i]	u(v)
used_by_curr_pic_lt_flag[i]	u(1)

[0081]

}	
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)

[0082]

}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(slice_sao_luma_flag || slice_sao_chroma_flag) {	
slice_sao_size_id	u(3)
if(slice_type == P || slice_type == B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type == B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPicTotalCurr > 1)	
ref_pic_lists_modification( )	
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	
cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type == P) || (weighted_bipred_flag && slice_type == B))	
pred_weight_table( )	
if(sps_ic_present_flag && slice_type != I) {	
slice_ic_enable_flag	u(1)
if(sps_bidir_ic_present_flag && slice_ic_enable_flag)	
slice_bidir_ic_enable_flag	u(1)
max_search_depth_region_tree	ue(v)
for (i = 0; i < max_search_depth_region_tree; i++)	
max_search_depth_prediction_tree[i]	ue(v)
if (slice_type == I)	
for (i = 0; i < max_search_depth_region_tree; i++)	
max_search_depth_prediction_tree_chroma[i]	
if (slice_type != I)	
max_num_merge_cand	ue(v)
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)

[0083]

slice_cr_qp_offset	se(v)
}	
if(chroma_qp_offset_list_enabled_flag)	
cu_chroma_qp_offset_enabled_flag	u(1)
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)
if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	
if(pps_loop_filter_across_slices_enabled_flag && (slice_sao_luma_flag || slice_sao_chroma_flag || !slice_deblocking_filter_disabled_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
}	
if(tiles_enabled_flag || entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	
}	
if(pps_clip_adaptive_enable_flag)	
clip_adaptive_flag	u(1)
if(clip_adaptive_flag) {	
clip_adaptive_y_min	ue(v)
clip_adaptive_y_max	ue(v)
clip_adaptive_flag_chroma	u(1)
if(clip_adaptive_flag_chroma) {	
clip_adaptive_c0_min	ue(v)
clip_adaptive_c0_max	ue(v)
clip_adaptive_c1_min	ue(v)
clip_adaptive_c1_max	ue(v)
}	
}	
}	
if(sps_affine_enabled_flag && slice_type != I) {	
affine_control_flag[0]	u(1)
affine_control_flag[1]	u(1)
}	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)

[0084]

for(i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[i]	u(8)
}	
byte_alignment( )	
}	

[0085] 表2:条带首部

[0086] 下面的表3、4、5、6例示根据至少一个示例实施例的经修改以支持附加分区模式的编码树语法。特别是,在coding_binary_tree()中规定用于启用非对称分区的语法。新语法元素被插入编码树语法中,更准确地:

[0087] -asymmetricSplitFlag,指示非对称二元分割是否允许用于当前CU,以及

[0088] -asymmetric_type,指示非对称二元分割的类型。它可以采用两个值,以使水平分割向上或向下,或使垂直分割向左或向右。

[0089] 此外,参数btSplitMode被不同地解释。

[0090]

coding_tree_unit( ) {	描述符
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
if(slice_sao_luma_flag || slice_sao_chroma_flag)	
if(slice_type == I) {	
coding_tree(xCtb, yCtb, CtbSizeX, CtbSizeY, LUMA_TREE, 0, 0)	
coding_tree(xCtb, yCtb, CtbSizeX, CtbSizeY, CHROMA_TREE, 0, 0)	
} else {	
coding_tree(xCtb, yCtb, CtbSizeX, CtbSizeY, LUMA_CHROMA_TREE, 0, 0)	
}	
}	

[0091] 表3:coding_tree_unit

coding_tree(x0, y0, log2CbSize, cqtDepth) {	描述符
if(x0 + (1 << log2CbSize) <= pic_width_in_luma_samples && y0 + (1 << log2CbSize) <= pic_height_in_luma_samples && log2CbSize > MinCbLog2SizeY)	
split_cu_flag[x0][y0]	ae(v)
if(split_cu_flag[x0][y0]) {	
x1 = x0 + (1 << (log2CbSize − 1))	
y1 = y0 + (1 << (log2CbSize − 1))	
coding_tree(x0, y0, log2CbSize − 1, cqtDepth + 1)	
if(x1 < pic_width_in_luma_samples)	
coding_tree(x1, y0, log2CbSize − 1, cqtDepth + 1)	
if(y1 < pic_height_in_luma_samples)	
coding_tree(x0, y1, log2CbSize − 1, cqtDepth + 1)	
if(x1 < pic_width_in_luma_samples && y1 < pic_height_in_luma_samples)	
coding_tree(x1, y1, log2CbSize − 1, cqtDepth + 1)	
} else	
coding_binary_tree(x0, y0, 1<<log2CbSize,1<< log2CbSize,cqtDepth)	
}	

表4:coding_tree

[0092]

coding_binary_tree(x0, y0, width, height, cqtDepth) {	描述符
if(btSplitAllowed(x0,y0,width,height){	
bt_split_mode(x0,y0,width,height,cqtDepth)	
}	
if(btSplitFlag) {	
if(btSplitMode==HOR) {	
x1 = x0	
y1 = y0 + (height >> 1)	
sub_width_1 = sub_width_0 = width;	
sub_height_1 = sub_height_0 = (height >> 1)	
}	
else if(btSplitMode==VER) {	
x1 = x0 + (width >> 1)	
y1 = y0	
sub_width_1 = sub_width_0 = (width >> 1)	
sub_height_1 = sub_height_0 = height	
}	
else if(btSplitMode==HOR_UP) {	
x1 = x0	
y1 = y0 + (height >> 2)	
sub_width_1 = sub_width_0 = width	
sub_height_0 = (height >> 2)	
sub_height_1 = ((height *3) >> 2)	
}	
else if(btSplitMode==HOR_DOWN) {	
x1 = x0	
y1 = y0 + ((height*3) >> 2)	
sub_width_1 = sub_width_0 = width	
sub_height_0 = ((height *3) >> 2)	
sub_height_1 = (height >> 2)	
}	
else if(btSplitMode==VER_LEFT) {	
x1 = x0 + (width >> 2)	
y1 = y0	
sub_width_0 = width >> 2	
sub_width_1 = (width *3) >> 2	
sub_height_1 = sub_height_0 = height	
}	
else if(btSplitMode==VER_RIGHT) {	
x1 = x0 + (width*3) >> 2	
y1 = y0	
sub_width_0 = (width*3) >> 2	
sub_width_1 = width >> 2	
sub_height_1 = sub_height_0 = height	
}	

[0093]

coding_binary_tree(x0, y0, sub_width, sub_height, cqtDepth)	
if(x1 < pic_width_in_luma_samples && y1 < pic_height_in_luma_samples)	
coding_binary_tree(x1, y1, sub_width, sub_height, cqtDepth)	
}	
} else	
coding_unit(x0, y0, width, height)	
}	

[0094] 表5:coding_binary_tree

[0095]

bt_split_mode(x0,y0,width,height,cqtDepth){	描述符
if(btSplitAllowed(x0,y0,width,height){	
btSplitFlag[x0][y0][cbSizeX][cbSizeY]	ae(v)
}	
if(btSplitFlag[x0][y0][cbSizeX][cbSizeY]) {	
if(horizontalSplitAllowed && verticalSplitAllowed){	
btSplitOrientation[x0][y0][cbSizeX][cbSizeY]	ae(v)
}	
if(btSplitOrientation==HOR && horizontal_asymmetric_allowed ||	
btSplitOrientation==VER && vertical_asymmetric_allowed){	
asymmetricSplitFlag[x0][y0][cbSizeX][cbSizeY]	ae(v)
if(asymmetricSplitFlag==true){	
asymmetric_type[x0][y0][cbSizeX][cbSizeY]	ae(v)
}	
}	
}	
}	

[0096] 表6:bt_split_mode

[0097] 图9例示根据至少一个示例实施例的bt-split-flag的解释。两个语法元素asymmetricSplitFlag和asymmetric_type用于规定参数btSplitMode的值。btSplitMode指示应用于当前CU的二元分割模式。btSplitMode与现有技术不同地被推导。在JEM中,它可以采用值HOR,VER。在改进的语法中,它可以附加地采用与如图6例示的分区对应的值HOR_UP,HOR_DOWN,VER_LEFT,VER_RIGHT。

[0098] 下面的表7、8、9和10例示编码单元语法元素,并且介绍了用于双向帧内预测模式的新语法元素。

[0099]

coding_unit(x0, y0, cuWidth, cuHeight, treeMode) {	描述符
if(transquant_bypass_enabled_flag)	
cu_transquant_bypass_flag	ae(v)
if(slice_type != I)	
cu_skip_flag[x0][y0]	ae(v)
if(cu_skip_flag[x0][y0])	
cu_data_merge(x0, y0, cuWidth, cuHeight, treeMode)	
else {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(cuPredMode[x0][y0] == MODE_INTRA)	
cu_data_intra(x0, y0, cuWidth, cuHeight, treeMode)	
else {	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0])	
cu_data_merge(x0, y0, cuWidth, cuHeight, treeMode)	
else	
cu_data_inter(x0, y0, cuWidth, cuHeight, treeMode)	
if(cuPredMode[x0][y0] != MODE_INTRA && !(merge_flag[x0][y0])	
rqt_root_cbf	ae(v)
if(rqt_root_cbf)	
cu_residual_data(x0, y0, cuWidth, cuHeight, treeMode)	
}	
}	
}	

[0100] 表7:coding_unit

[0101]

cu_data_merge(x0, y0, cuWidth, cuHeight, treeMode) {	描述符
if(fruc_merge_enabled_flag)	
fruc_merge_mode[x0][y0]	ae(v)
if(fruc_merge_mode[x0][y0])	
if(lic_enabled_flag && fruc_merge_mode[x0][y0] == 1)	
lic_flag[x0][y0]	ae(v)
if(fruc_merge_mode[x0][y0] == 1 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(lic_enabled_flag)	
lic_flag[x0][y0]	ae(v)
else	
if(MaxNumMergeCand > 1)	
merge_idx[x0][y0]	
}	

[0102] 表8:cu_data_merge

cu_data_intra(x0, y0, cuWidth, cuHeight, treeMode) {	描述符
if(treeMode & LUMA_TREE) {	
prev_intra_luma_pred_flag[x0][y0]	ae(v)
if(prev_intra_luma_pred_flag[x0][y0]) {	
mpm_idx[x0][y0]	ae(v)
} else {	
second_mpm_flag[x0][y0]	ae(v)
if(second_mpm_flag[x0][y0]	
second_mpm[x0][y0]	f(5)
else	
rem_intra_luma_pred_mode[x0][y0]	ae(v)
}	
if(bdip_enable_flag && ((luma_mode >= 2 && luma_mode < 18) || (luma_mode >= 114 && luma_mode < 130)	
bdip_enable_flag[x0][y0]	ae(v)
}	
if(treeMode & CHROMA_TREE) {	
intra_chroma_pred_mode[x0][y0]	ae(v)
}	
}	

表9:cu_data_intra

cu_data_inter(x0, y0, cuWidth, cuHeight, treeMode) {	描述符
if(slice_type == B)	
inter_pred_idc[x0][y0]	ae(v)
if(affine_enabled_flag && cuWidth > 8 && cuHeight > 8)	
affine_flag[x0][y0]	ae(v)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
if(affine_flag[x0][y0]) {	
mvd_gr0(x0, y0, 0, AFFINE_LEFT)	ae(v)
mvd_gr0(x0, y0, 0, AFFINE_RIGHT)	ae(v)
} else	
mvd_ gr0 (x0, y0, 0, AFFINE_OFF)	
mvp_l0_flag[x0][y0]	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0][y0]	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] == PRED_BI) {	
MvdL1[x0][y0][0] = 0	

[0103]

[0104]

MvdL1[x0][y0][1] = 0	
} else if(affine_flag[x0][y0]) {	
mvd_gr0(x0, y0, 1, AFFINE_LEFT)	ae(v)
mvd_gr0(x0, y0, 1, AFFINE_RIGHT)	ae(v)
} else	
mvd_gr0(x0, y0, 1, AFFINE_OFF)	
mvp_l1_flag[x0][y0]	ae(v)
}	
if(imv_enabled_flag && CuHasNonZeroMvd)	
imv_mode[x0][y0]	ae(v)
if(affine[x0][y0] || CuHasNonZeroMvd)	
{	
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(affine_flag[x0][y0]) {	
mvd_remain(x0, y0, 0, AFFINE_LEFT)	ae(v)
mvd_remain(x0, y0, 0, AFFINE_RIGHT)	ae(v)
} else	
mvd_ remain(x0, y0, 0, AFFINE_OFF)	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] == PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
} else if(affine_flag[x0][y0]) {	
mvd_gr0(x0, y0, 1, AFFINE_LEFT)	ae(v)
mvd_gr0(x0, y0, 1, AFFINE_RIGHT)	ae(v)
} else	
mvd_gr0(x0, y0, 1, AFFINE_OFF)	ae(v)
}	
}	
if(obmc_enabled_flag && cuWidth*cuHeight <= 16*16)	
obmc_flag[x0][y0]	ae(v)
if(lic_enabled_flag && !affine_flag[x0][y0])	
lic_flag[x0][y0]	ae(v)
}	

[0105] 表10:cu_data_inter

[0106]

cu_residual_data(x0, y0, cuWidth, cuHeight, treeMode) {	描述符
if(treeMode & CHROMA_TREE) {	
cbf_cb[x0][y0]	ae(v)
cbf_cr[x0][y0]	ae(v)
}	
if(treeMode & LUMA_TREE) {	

[0107]

if(cuPredMode[x0][y0] == MODE_INTRA || cbf_cb[x0][y0] || cbf_cr[x0][y0])	
cbf_luma[x0][y0]	ae(v)
}	
if(treeMode & CHROMA_TREE) {	
if(cbf_cb[x0][y0])	
residual_coding(x0, y0, cuWidth / 2, cuHeight / 2, treeMode, 1)	
if(cbf_cr[x0][y0])	
residual_coding(x0, y0, cuWidth / 2, cuHeight / 2, treeMode, 2)	
}	
if(treeMode & LUMA_TREE) {	
if(emt_enable_flag && cuWidth <= 64 && cuHeight <= 64)	
emt_cu_flag[x0][y0]	ae(v)
residual_coding(x0, y0, cuWidth, cuHeight, treeMode, 0)	
}	

[0108] 表11:cu_residual_data

[0109] 本文件中讨论的若干语法元素被定义为数组。例如,btSplitFlag定义为维度4的数组,由画面中的水平和垂直位置以及编码块的水平和垂直尺寸索引。为了简化符号,在语法元素的语义描述中,索引不被保持(例如,仅将btSplitFlag[x0][y0][cbSizeX][cbSizeY]记为btSplitFlag)。

[0110] 图10例示可以实现各个方面和实施例的示例性系统的框图。系统1000可以体现为包括下面描述的各种组件的设备,并且被配置为执行本申请中描述的一个或多个方面。这样的设备的示例包括但不限于各种电子设备,诸如个人计算机,膝上型计算机,智能电话,平板计算机,数字多媒体机顶盒,数字电视接收器,个人视频记录系统,连接的家用电器、编码器、转码器和服务器。系统1000的元件可以单独或组合地体现在单个集成电路,多个IC和/或分立组件中。例如,在至少一个实施例中,系统1000的处理和编码器/解码器元件分布在多个IC和/或分立组件上。在各种实施例中,系统1000可以通信地耦合到其他类似系统,或者经由例如通信总线或通过专用输入和/或输出端口耦合到其他电子设备。在各种实施例中,系统1000配置为实现本文件中描述的一个或者多个方面。

[0111] 系统1000包括至少一个处理器1010,其被配置为执行加载在其中的指令,用于实现本文件描述的各种方面。处理器1010可以包括嵌入式存储器,输入输出接口和本领域已知的各种其他电路。系统1000还可以包括至少一个存储器1020(例如,易失性存储器设备,非易失性存储器设备)。系统1000可以包括存储设备1040,其可以包括非易失性存储器和/或易失性存储器,包括但不限于EEPROM,ROM,PROM,RAM,DRAM,SRAM,闪存,磁盘驱动器和/或光盘驱动器。作为非限制性示例,存储设备1040可以包括内部存储设备,附接的存储设备和/或网络可存取存储设备。

[0112] 系统1000包括编码器/解码器模块1030,其被配置为处理数据以提供编码的视频或解码的视频,并且编码器/解码器模块1030可以包括其自己的处理器和存储器。编码器/解码器模块1030表示可以包括在设备中以执行编码和/或解码功能的(多个)模块。如所知,设备可以包括编码和解码模块中的一个或两个。另外,如本领域技术人员已知的,编码器/解码器模块1030可以实现为系统1000的单独元件,或者可以并入处理器1010内作为硬件和软件的组合。

[0113] 要加载到处理器1010或者编码器/解码器1030上以执行本文件描述的各种方面的程序代码可以存储在存储设备1040中,并且随后加载到存储器1020上用于由处理器1010执行。根据各种实施例,处理器1010,存储器1020,存储设备1040和编码器/解码器模块1030中的一个或多个可以在执行本文件描述的处理期间存储各种项目中的一个或多个。存储的这种项目包括但不限于输入视频,解码视频或解码视频的一部分,比特流,矩阵,变量以及等式、公式、运算和运算逻辑的处理产生中间或最终结果。

[0114] 在若干个实施例中,处理器1010和/或编码器/解码器模块1030内部的存储器用于存储指令并为编码或解码期间所需的处理提供工作存储器。然而,在其他实施例中,处理设备外部的存储器(例如,处理设备可以是处理器1010或编码器/解码器模块1030)被用于这些功能中的一个或多个。外部存储器可以是存储器1020和/或存储设备1040,例如,动态易失性存储器和/或非易失性闪存。在若干实施例中,外部非易失性闪存用于存储电视的操作系统。在至少一个实施例中,诸如RAM之类的快速外部动态易失性存储器被用作用于诸如MPEG-2,HEVC或VVC(通用视频编码)之类的视频编码和解码操作的工作存储器。

[0115] 如块1130所指示的,可以通过各种输入设备来提供对系统1000的元件的输入。这种输入设备包括但不限于(i)接收例如由广播公司通过无线方式发送的RF信号的RF部分,(ii)复合输入端子,(iii)USB输入端子,和/或(iv)HDMI输入端子

[0116] 在各种实施例中,块1130的输入设备具有相关联的本领域中已知的相应输入处理元件。例如,RF部分可以与适合于以下的元件相关联:(i)选择所需频率(也称为选择信号,或将信号频带限制在频带内),(ii)下转换所选信号,(iii)再次将频带限制到较窄的频带以选择(例如)在某些实施例中可以称为信道的信号频带,(iv)解调下转换并限制频带的信号,(v)执行纠错,以及(vi)解复用以选择所需的数据分组流。各种实施例的RF部分包括一个或多个执行这些功能的元件,例如,频率选择器,信号选择器,频带限制器,信道选择器,滤波器,下转换器,解调器,纠错器和解复用器。RF部分可以包括执行各种这些功能的调谐器,包括例如将接收到的信号下转换为较低频率(例如,中频或近基带频率)或基带。在一个机顶盒实施例中,RF部分及其相关的输入处理元件接收通过有线(例如,电缆)介质传输的RF信号,并通过滤波,下转换和再次滤波到所需频带来执行频率选择。各种实施例重新布置上述(和其他)元件的顺序,移除这些元件中的一些,和/或添加执行类似或不同功能的其他元件。添加元件可以包括在现有元件之间的插入元件,例如,插入放大器和模数转换器。在各个实施例中,RF部分包括天线。

[0117] 另外,USB和/或HDMI端子可以包括相应的接口处理器,用于跨越USB和/或HDMI连接将系统1000连接到其他电子设备。要理解,输入处理的各个方面,例如里德-所罗门纠错,可以根据需要例如在单独的输入处理IC内或在处理器1010内实现。类似地,USB或HDMI接口处理的各方面可以根据需要在单独的接口IC内或在处理器1010内实现。解调,纠错和解复用的流被提供给各种处理元件,包括例如与存储器和存储元件结合操作的处理器1010和编码器/解码器1030,以根据需要处理数据流以在输出设备上呈现。

[0118] 系统1000的各种元件可以设置在集成壳体内。在集成壳体内,可以使用合适的连接装置,例如本领域已知的内部总线,包括I2C,总线,接线和印刷电路板,来互连各种元件并在它们之间传输数据。

[0119] 系统1000包括使得能够经由通信信道1060与其他设备进行通信的通信接口1050。

通信接口1050可以包括但不限于配置为在通信信道1060上发送和接收数据的收发器。通信接口1050可以包括但不限于调制解调器或网卡,并且通信信道1060可以例如在有线和/或无线介质内实现。

[0120] 在各种实施例中,使用诸如IEEE 802.11的Wi-Fi网络将数据流传输到系统1000。这些实施例的Wi-Fi信号在适于Wi-Fi通信的通信信道1060和通信接口1050上被接收。这些实施例的通信信道1060典型连接到接入点或路由器,该接入点或路由器提供对包括互联网的外部网络的访问,以允许流传输应用程序和其他空中通信。其他实施例使用通过输入块1130的HDMI连接传递数据的机顶盒将流传输的数据提供给系统1000。其他实施例使用输入块1130的RF连接将流传输的数据提供给系统1000。

[0121] 系统1000可以向包括显示器1100,扬声器1110和其他外围设备1120的各种输出设备提供输出信号。其他外围设备1120在实施例的各种示例中包括独立DVR,盘播放器,立体系统,照明系统和基于系统1000的输出提供功能的其他设备中的一个或多个。在各种实施例中,在系统1000与显示器1100,扬声器1110,或其他外围设备1120之间使用诸如AV.Link,CEC或使得能够在有或没有用户干预的情况下设备到设备的控制成为可能的其他通信协议之类的信令进行控制信号的通信。输出设备可以通过各个接口1070、1080和1090经由专用连接而通信地耦合至系统1000。可替代地,输出设备可以经由通信接口1050使用通信信道1060而连接至系统1000。显示器1100和扬声器1110可以与系统1000的其他组件集成到电子设备(例如电视机)中的单一单元中。在各种实施例中,显示接口1070包括显示驱动器,例如,时序控制器(T Con)芯片。

[0122] 例如,如果输入1130的RF部分是单独的机顶盒的一部分,则显示器1100和扬声器1110可以可替换地与一个或多个其他组件分开。在显示器1100和扬声器1110是外部组件的各种实施例中,可以经由专用输出连接(包括例如HDMI端口,USB端口或COMP输出)提供输出信号。本文描述的实现方式可以在例如方法或处理,装置,软件程序,数据流或信号中实现。即使仅在单个实现形式的上下文中讨论(例如,仅作为方法讨论),讨论的特征的实现方式也可以以其他形式(例如,装置或程序)来实现。装置可以在例如适当的硬件,软件和固件中实现。方法例如可以在例如诸如处理器之类的装置中实现,该处理器一般指代处理设备,包括例如计算机,微处理器,集成电路或可编程逻辑设备。处理器也包括通信设备,诸如例如计算机,蜂窝电话,便携/个人数字助理(“PDA”),以及便于终端用户之间的信息通信的其他设备。

[0123] 图11例示根据实施例的使用新编码工具的编码方法的示例的流程图。这样的编码方法可以由图10中描述的系统1000来执行,并且更确切地可以由处理器1010来实现。在至少一个实施例中,在步骤1190中,处理器1010选择要使用的编码工具。可以使用不同的技术来进行选择。用户可以通过编码配置参数(典型,标志)来进行选择,该编码配置参数指示编码和解码处理期间要使用的工具(以及对应的参数,如果需要)。在示例实施例中,通过在文件中设置一值的值来进行选择,该标志由解释选择使用哪个工具的标志的值得编码设备读出。在示例实施例中,使用处理编码器设备的图形用户界面通过用户选择编码配置参数的手动操作来进行选择。一旦进行该选择,就在步骤1193中利用其他编码工具中的所选工具来执行编码,并且在高级语法元素中(例如,在下一个SPS,PPS甚至是条带首部中)用信号通知对工具的选择。

[0124] 图12例示使用新编码工具的根据实施例的解码方法的一部分的示例的流程图。这样的解码方法可以由图10中描述的系统1000来执行,并且更精确地可以由处理器1010来实现。在至少一个实施例中,在步骤1200中,处理器1010访问信号(例如,在输入界面上接收到的,或从媒体载体中读取的信号),提取并分析高级语法元素,以确定已在编码设备上选择的工具。在步骤1210中,这些工具用于执行解码并产生可以例如提供给设备或显示在设备上的解码图像。

[0125] 对“一个实施例”或“实施例”或“一个实现方式”或“实现方式”的引用以及其其他变型意味着结合实施例描述的具体特征,结构,特性等包括在至少一个实施例中。因此,在整个说明书中出现在各个地方的短语“在一个实施例中”或“在实施例中”或“在一个实现方式中”或“在实现方式中”以及任何其他变型的出现不一定都指代同一个实施例。

[0126] 另外,本申请或其权利要求可以指“确定”各种信息。确定信息可以包括例如估计信息,计算信息,预测信息或从存储器检索信息中的一个或多个。

[0127] 此外,本申请或其权利要求可以指“访问”各种信息。访问信息可以包括例如接收信息,检索信息(例如,从存储器中),存储信息,移动信息,复制信息,计算信息,预测信息或估计信息中的一个或多个。

[0128] 另外,本申请或其权利要求可以指“接收”各种信息。与“访问”一样,接收意图是广义术语。接收信息可以包括例如访问信息或检索信息(例如,从存储器或光学介质存储中)中的一个或多个。此外,“接收”典型在操作期间以一种方式或其他方式,涉及例如,存储信息,处理信息,传送信息,移动信息,复制信息,擦除信息,计算信息,确定信息,预测信息或估计信息。

[0129] 要认识到,例如在“A/B”,“A和/或B”和“A和B中的至少一个”的情况中的以下“/”,“和/或”和“至少一个”的使用意图包括仅对第一所列选项(A)的选择、或仅对第二所列选项(B)的选择、或对两个选项(A和B)的选择。作为进一步的例子,在“A、B和/或C”和“A、B和C中的至少一个”的情况中,这样的措辞意图包括仅对第一所列选项(A)的选择、或仅对第二所列选择(B)的选择、或仅对第三所列选项(C)的选择、或仅对第一和第二所列选项(A和B)的选择、或仅对第一和第三所列选项(A和C)的选择、或仅对第二和第三所列选项(B和C)的选择、或对全部三个选项(A和B和C)的选择。如同对于本领域和相关领域中的普通技术人员来说容易地明显的那样,可以对于所列出的许多项目扩展该措辞。。

[0130] 将对于本领域技术人员显而易见的是,实现方式可以产生被格式化以携带例如可以存储或传送的信息的各种信号。该信息可以包括例如用于执行方法的指令或由描述的实施方式之一产生的数据。例如,可以格式化信号以携带描述的实施例的比特流。这样的信号可以被格式化,例如作为电磁波(例如,使用频谱的射频部分)或者作为基带信号。格式化可以包括,例如编码数据流和用编码数据流调制载波。信号携带的信息可以是例如模拟或数字信息。如已知,信号可以通过各种不同的有线或无线链路传送。信号可以存储在处理器可读介质上。

[0131] 在至少一个实施例的第一,第二,第三,第四,第五,第六,第七和第八方面的变型中,表示分区类型的参数包括用于四叉树和二叉树分区的第一值,用于四叉树和二叉树分区加三叉树分区的第二值,用于四叉树和二叉树分区加非对称二叉树分区的第三值,用于四叉树和二叉树分区加三叉树分区加非对称二叉树分区的第四值。

[0132] 在至少一个实施例的第一,第二,第三,第四,第五,第六,第七和第八方面的变型中,参数还包括针对编码条带在其上应用样本自适应偏移的块尺寸。

[0133] 在至少一个实施例的第一,第二,第三,第四,第五,第六,第七和第八方面的变型中,帧内预测模式的类型包括帧内双向预测模式和多参考帧内预测中的至少一个模式。

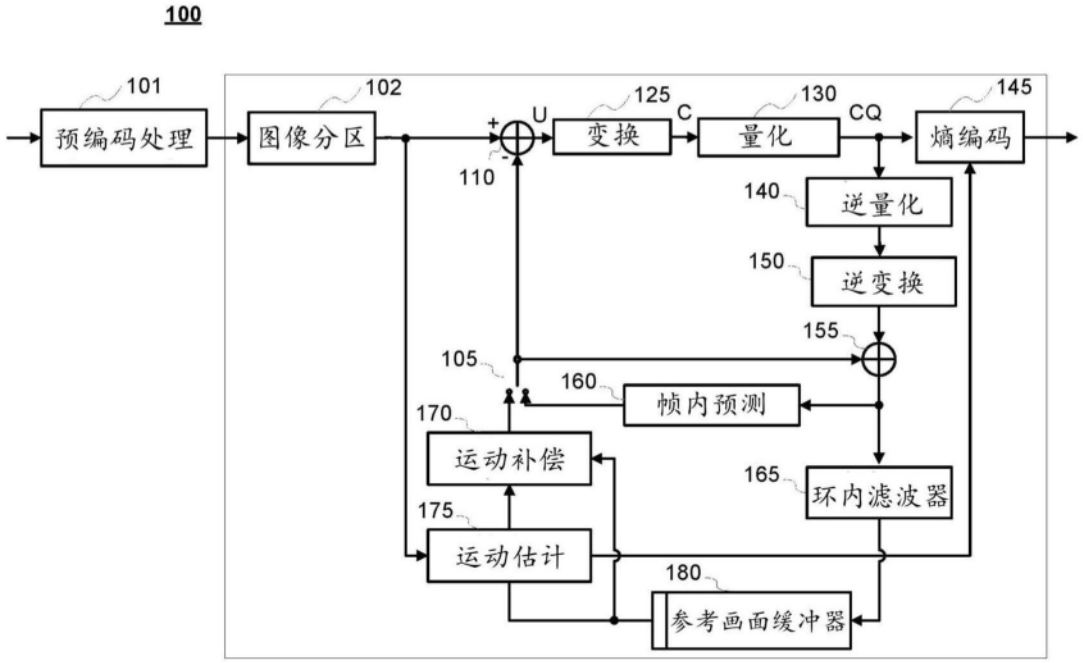

100
101
预编码处理
102
图像分区
+
-
U
110
125
变换
C
130
量化
CQ
145
熵编码
140
逆量化
150
逆变换
155
105
160
帧内预测
170
运动补偿
165
环内滤波器
175
运动估计
180
参考画面缓冲器

图1

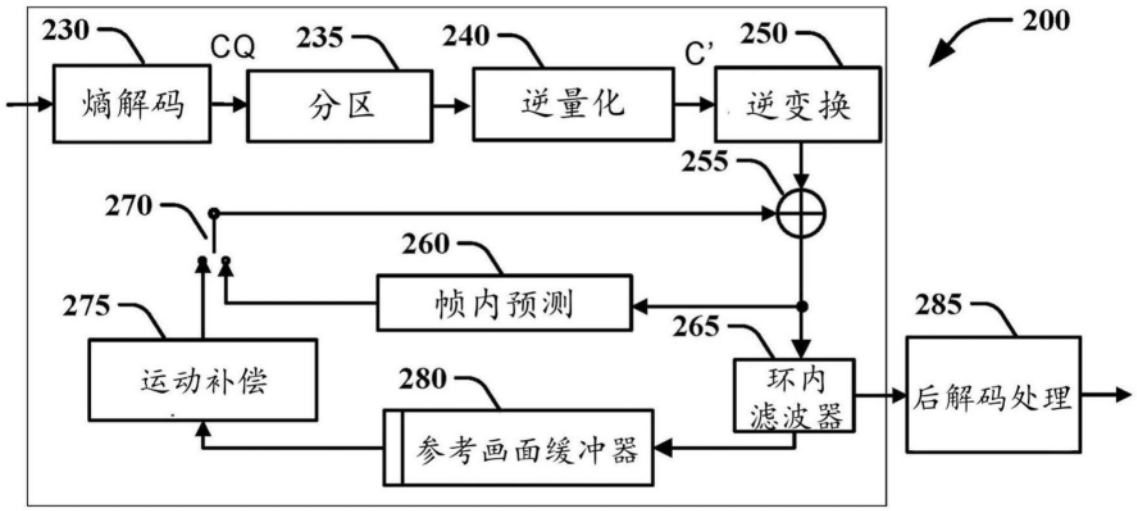

200
230
熵解码
CQ
235
分区
240
逆量化
C'
250
逆变换
255
270
260
帧内预测
275
运动补偿
265
环内
滤波器
285
后解码处理
280
参考画面缓冲器

图2

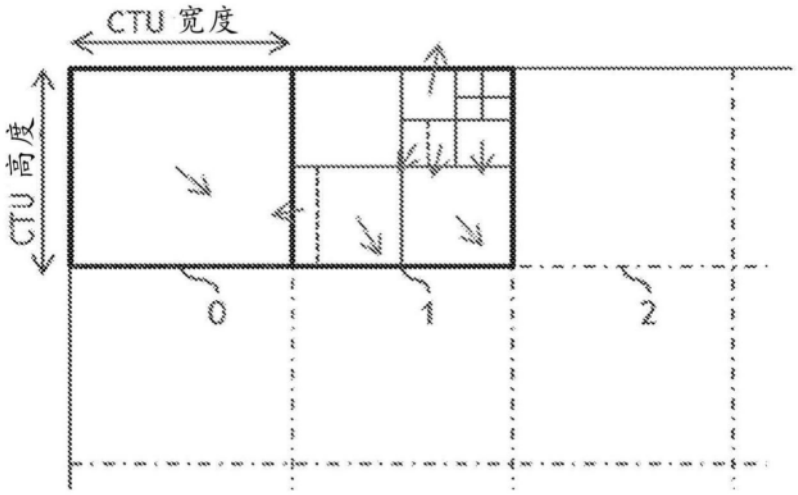

CTU 宽度
CTU 高度
0
1
2

图3

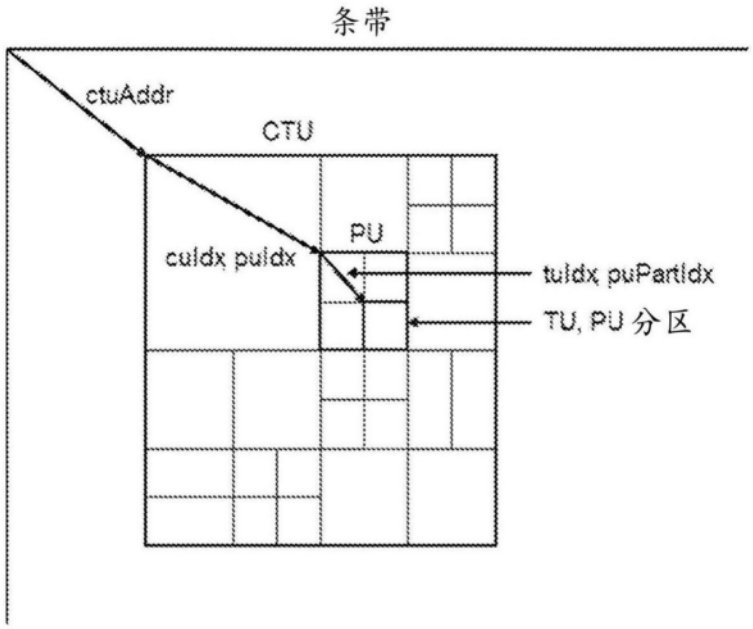

条带
ctuAddr
CTU
PU
cuIdx puIdx
tuIdx puPartIdx
TU, PU 分区

图4

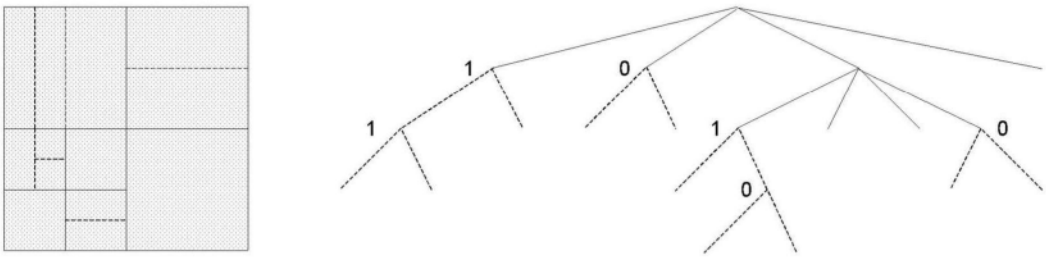

1
0
1
1
0
0

图5

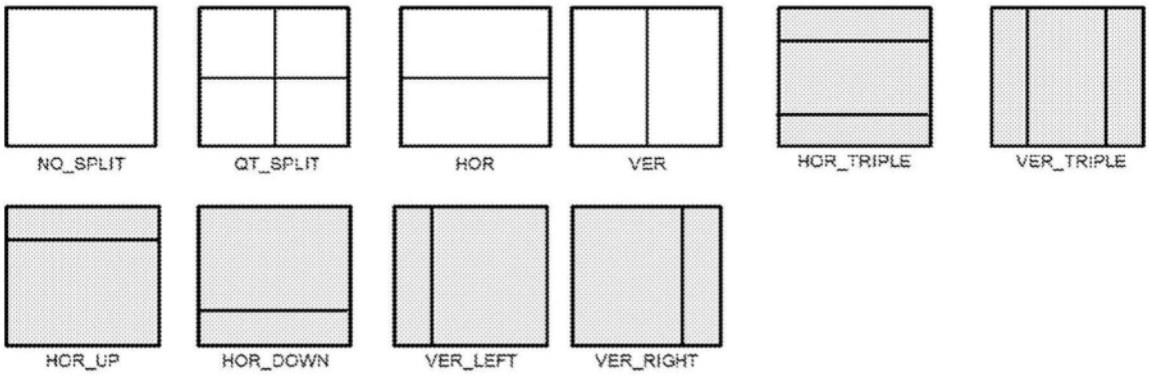

NO_SPLIT
QT_SPLIT
HOR
VER
HOR_TRIPLE
VER_TRIPLE
HOR_UP
HOR_DOWN
VER_LEFT
VER_RIGHT

图6

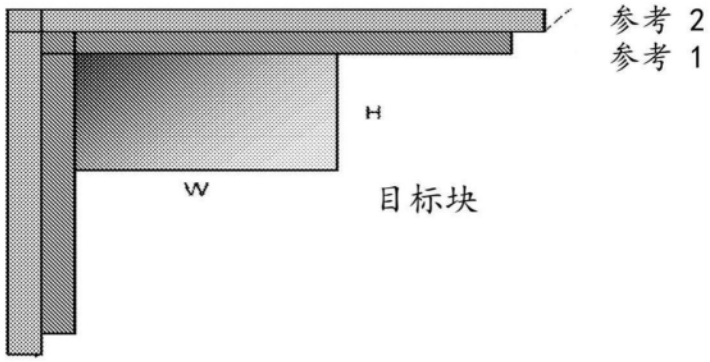

参考 2
参考 1
H
W
目标块

图7

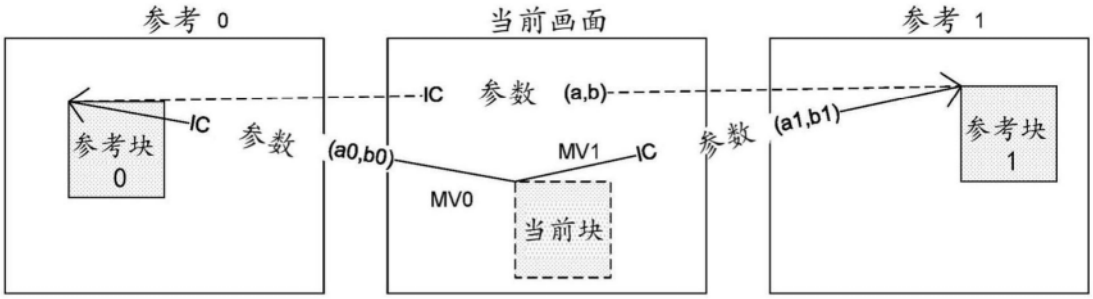

参考 0
当前画面
参考 1
IC
参数
(a,b)
IC
参考块
0
参数
(a0,b0)
MV1
IC
参数
(a1,b1)
参考块
1
MV0
当前块

图8

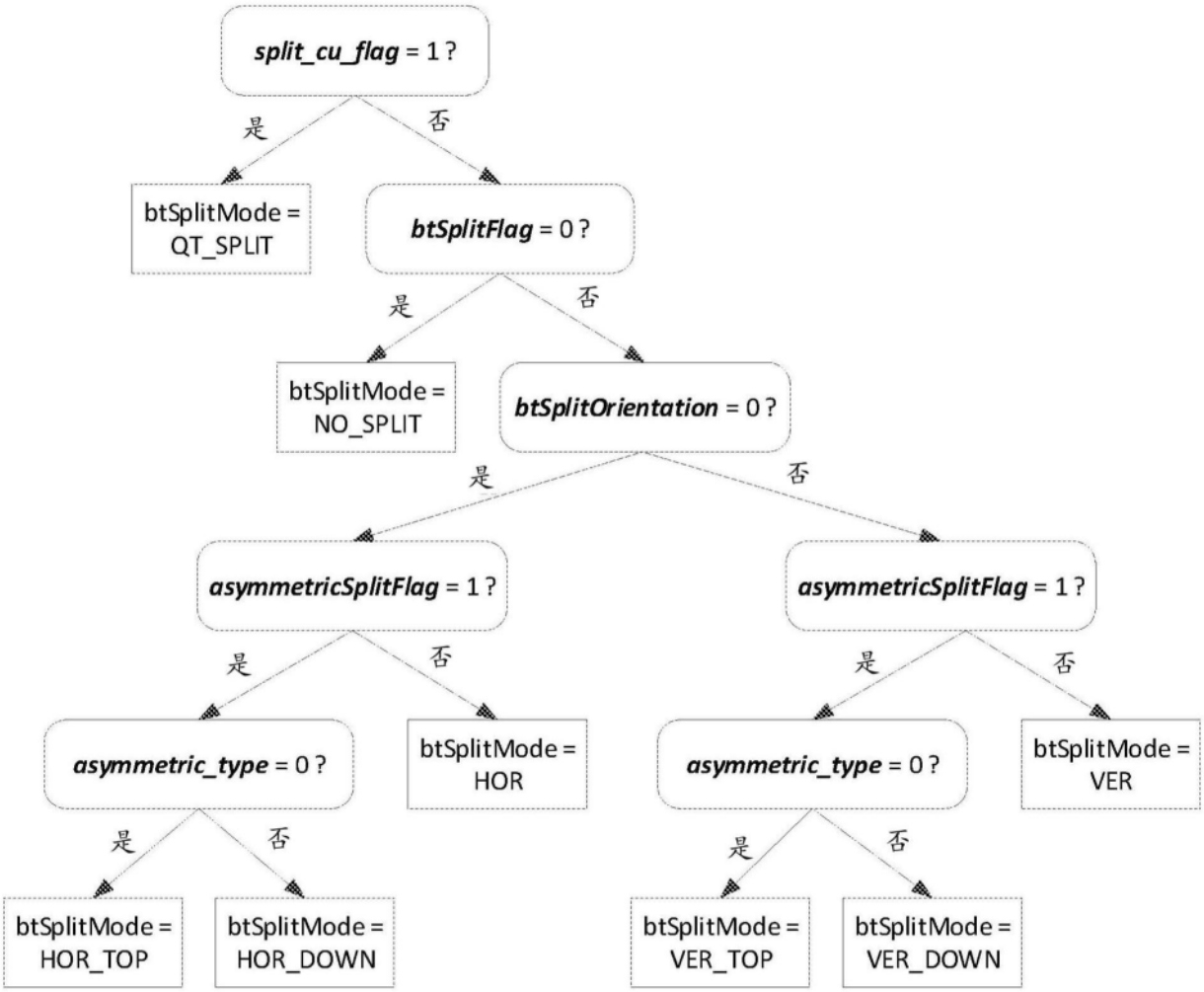

split_cu_flag = 1 ?
是
否
btSplitMode = QT_SPLIT
btSplitFlag = 0 ?
是
否
btSplitMode = NO_SPLIT
btSplitOrientation = 0 ?
是
否
asymmetricSplitFlag = 1 ?
asymmetricSplitFlag = 1 ?
是
否
是
否
asymmetric_type = 0 ?
btSplitMode = HOR
asymmetric_type = 0 ?
btSplitMode = VER
是
否
是
否
btSplitMode = HOR_TOP
btSplitMode = HOR_DOWN
btSplitMode = VER_TOP
btSplitMode = VER_DOWN

图9

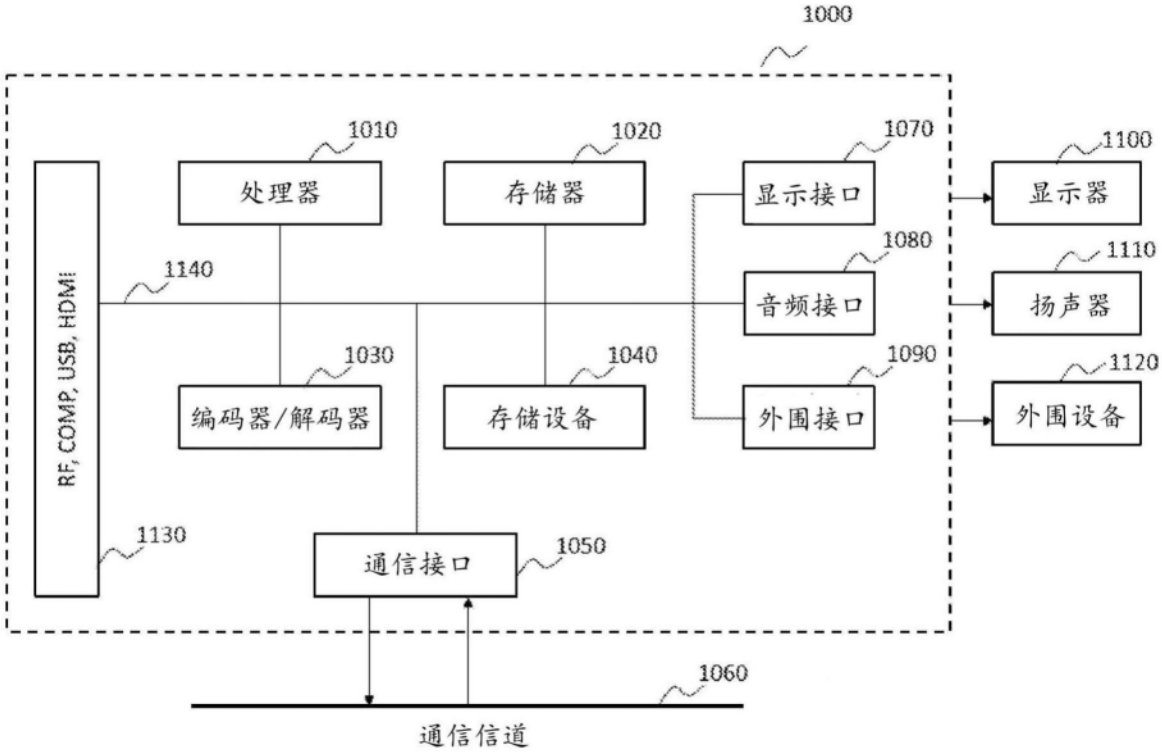

1000
1010
处理器
1020
存储器
1070
显示接口
1100
显示器
1080
音频接口
1110
扬声器
1140
RF, COMP, USB, HDMI
1030
编码器/解码器
1040
存储设备
1090
外围接口
1120
外围设备
1130
通信接口
1050
1060
通信信道

图10

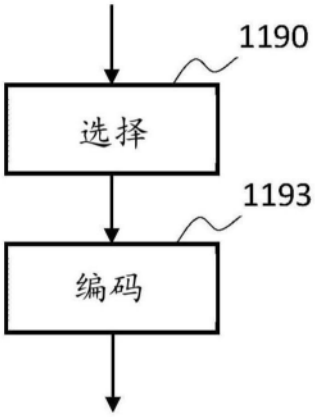

1190
选择
1193
编码

图11

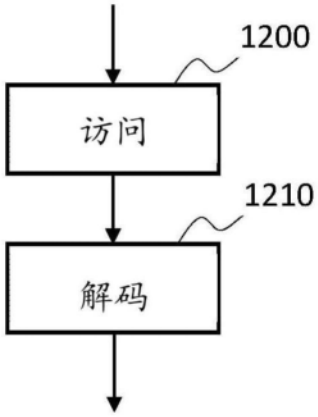


图12