



(12) 发明专利申请

(10) 申请公布号 CN 118945360 A

(43) 申请公布日 2024. 11. 12

(21) 申请号 202411204760.4

(22) 申请日 2020.03.06

(30) 优先权数据

19305292.5 2019.03.12 EP

19305645.4 2019.05.21 EP

19305649.6 2019.05.23 EP

(62) 分案原申请数据

202080020023.2 2020.03.06

(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 A·罗伯特 陈娅

(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 19/13 (2014.01)

H04N 19/124 (2014.01)

H04N 19/127 (2014.01)

H04N 19/132 (2014.01)

H04N 19/156 (2014.01)

H04N 19/157 (2014.01)

H04N 19/172 (2014.01)

H04N 19/176 (2014.01)

H04N 19/18 (2014.01)

H04N 19/42 (2014.01)

H04N 19/44 (2014.01)

H04N 19/60 (2014.01)

H04N 19/70 (2014.01)

H04N 19/91 (2014.01)

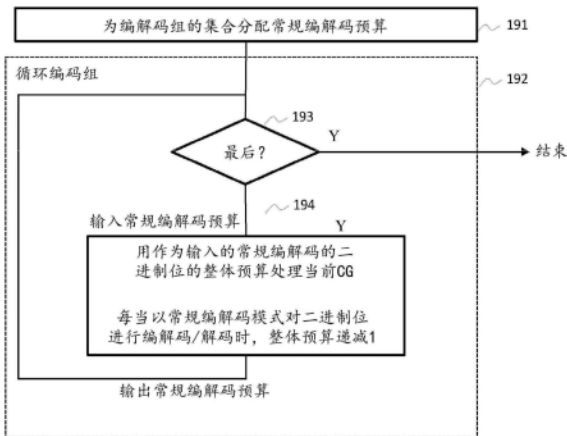
权利要求书1页 说明书16页 附图28页

(54) 发明名称

用于视频编解码的方法和装置

(57) 摘要

在至少一个实施例中,一种用于对视频进行编码/解码的方法和装置,其中,视频编码方法包括残差编码过程,残差编码过程使用有限数量的常规二进制位对表示包括编解码组的图片区域的语法元素进行编解码,编解码组表示变换块的量化系数的子块,其中,编解码是CABAC编码,其中,常规二进制位的数量是基于根据多个编解码组的样本数量为多个编解码组分配的预算来确定的。



1. 一种视频编码方法,包括残差编码过程,所述残差编码过程使用有限数量的常规二进制位对表示包括编解码组的图片区域的语法元素进行编解码,所述编解码组表示变换块的量化系数的子块,其中,所述编解码是CABAC编码,其中,常规二进制位的数量是基于根据多个编解码组的样本数量为所述多个编解码组分配的预算来确定的。

2. 根据权利要求1所述的方法,其中,对于正在被编解码的块,每当使用常规CABAC编解码模式对二进制元素进行编解码时,在多个编解码组中分配的预算就递减。

3. 一种视频解码方法,包括残差解码过程,所述残差解码过程使用常规二进制位来解析表示包括编解码组的图片区域的比特流,所述编解码组表示变换块的量化系数的子块,其中,所述残差解码过程使用CABAC解码来完成,其中,常规二进制位的数量是基于根据多个编解码组的样本数量为所述多个编解码组分配的预算来确定的。

4. 根据权利要求3所述的方法,其中,对于正在被解码的块,每当使用常规CABAC解码模式对二进制元素进行解码时,在多个编解码组中分配的预算就递减。

5. 一种视频编码装置,包括残差编码过程,所述残差编码过程使用有限数量的常规二进制位对表示包括编解码组的图片区域的语法元素进行编解码,所述编解码组表示变换块的量化系数的子块,其中,所述编解码是CABAC编码,其中,常规二进制位的数量是基于根据多个编解码组的样本数量为所述多个编解码组分配的预算来确定的。

6. 根据权利要求5所述的视频编码装置,其中,对于正在被编解码的块,每当使用常规CABAC编解码模式对二进制元素进行编解码时,在多个编解码组中分配的预算就递减。

7. 一种视频解码装置,包括残差解码过程,所述残差解码过程使用常规二进制位来解析表示包括编解码组的图片区域的比特流,所述编解码组表示变换块的量化系数的子块,其中,所述残差解码过程使用CABAC解码来完成,其中,常规二进制位的数量是基于根据多个编解码组的样本数量为所述多个编解码组分配的预算来确定的。

8. 根据权利要求7所述的视频解码装置,其中,对于正在被解码的块,每当使用常规CABAC解码模式对二进制元素进行解码时,在多个编码组中分配的预算就递减。

用于视频编解码的方法和装置

[0001] 本申请是申请号为202080020023.2,申请日为2020年3月6日,进入国家日期为2021年9月9日,发明名称为“用于视频编解码的残差编解码中常规二进制位的灵活分配”的申请的分案申请。

技术领域

[0002] 本实施例中的至少一个通常涉及用于视频编码或解码的残差编解码中的常规二进制位(bin)的分配。

背景技术

[0003] 为了实现高压压缩效率,图像和视频编解码方案通常采用预测和变换来利用视频内容中的空域和时域冗余。通常,帧内或帧间预测用于利用帧内或帧间的帧相关性,然后原始块与经预测的块之间的差(通常被表示为预测误差或预测残差)被变换、量化和熵编解码。为了重建视频,通过对应于熵编解码、量化、变换和预测的逆过程对压缩的数据进行解码。

发明内容

[0004] 本实施例中的一个或多个实施例在块及其系数组的残差编解码过程中处理对二进制位的常规CABAC编解码的最大使用的高级别约束,使得高级别约束得到遵守,并且使得与当前方法相比,压缩效率得到提高。

[0005] 根据至少一个实施例的第一方面,一种视频编码方法,包括:残差编码过程,该残差编码过程使用有限数量的常规二进制位对表示包括编解码组的图片区域的语法元素进行编解码,其中该编解码是CABAC编码,以及其中基于在多个编解码组之间分配的预算来确定常规二进制位的数量。

[0006] 根据至少一个实施例的第二方面,一种视频解码方法,包括:残差解码过程,该残差解码过程使用常规二进制位对表示包括编解码组的图片区域的比特流进行解析,其中使用CABAC解码来完成解码,以及其中基于在多个编解码组之间分配的预算来确定常规二进制位的数量。

[0007] 根据至少一个实施例的第三方面,一种装置,包括:用于对图片或视频中的至少一个块的图片数据进行编码的编码器,其中该编码器被配置为使用有限数量的常规二进制位来执行残差编码过程,以对表示包括编解码组的图片区域的语法元素进行编解码,其中该编解码是CABAC编码,以及其中基于在多个编解码组之间分配的预算来确定常规二进制位的数量。

[0008] 根据至少一个实施例的第四方面,一种装置,包括:用于对图片或视频中的至少一个块的图片数据进行解码的解码器,其中该解码器被配置为使用常规二进制位来执行残差解码过程,以对表示包括编解码组的图片区域的比特流进行解析,其中使用CABAC解码来完成解码,以及其中基于在多个编解码组之间分配的预算来确定常规二进制位的数量。

[0009] 根据至少一个实施例的第五方面,呈现了一种包括可由处理器执行的程序代码指

令的计算机程序,该计算机程序实现根据至少第一或第二方面的方法的步骤。

[0010] 根据至少一个实施例的第六方面,呈现了一种存储在非暂时性计算机可读介质上并且包括可由处理器执行的程序代码指令的计算机程序产品,该计算机程序产品实现了根据至少第一或第二方面的方法的步骤。

附图说明

[0011] 图1示出了视频编码器100(例如高效视频编解码(HEVC)编码器)的示例的框图。

[0012] 图2示出了视频解码器200(例如HEVC解码器)的示例的框图。

[0013] 图3示出了其中实现了各种方面和实施例的系统的示例的框图。

[0014] 图4A示出了压缩域中的编解码树单元和编解码树的示例。

[0015] 图4B示出了将CTU划分成编解码单元、预测单元和变换单元的示例。

[0016] 图5示出了两个标量量化器在依赖标量量化中的使用。

[0017] 图6A和图6B示出了在标量量化器之间切换的示例机制。

[0018] 图6C示出了如在 8×8 变换块中在VVC中使用的CG和系数之间的扫描顺序的示例。

[0019] 图7A示出了用于变换块的编解码/解析的语法元素的示例。

[0020] 图7B示出了CG级别残差编解码语法的示例。

[0021] 图8A示出了CU级别语法的示例。

[0022] 图8B示出了transform_tree语法结构的示例。

[0023] 图8C示出了变换单元级别语法布置。

[0024] 图9A和图9B分别示出了CABAC解码和编码过程。

[0025] 图10示出了根据本原理的实施例的包括CABAC优化器的CABAC编码过程。

[0026] 图11示出了根据本原理的实施例的编码过程中使用的CABAC优化器的示例流程图。

[0027] 图12示出了使用CABAC优化器对编解码组进行编码/解码的修改过程的示例。

[0028] 图13A示出了其中在变换块级别确定常规二进制位的预算的示例实施例。

[0029] 图13B示出了其中根据最后有效系数(last significant coefficient)的位置在变换块级别确定常规二进制位的预算的示例实施例。

[0030] 图14A示出了其中在变换单元级别确定常规二进制位的预算的示例实施例。

[0031] 图14B示出了示例实施例,其中根据不同变换块之间的相对表面并且根据在所考虑的变换单元中已经被编解码/解析的变换块中使用的常规二进制位,在变换单元级别确定并且在变换块之间分配常规二进制位。

[0032] 图14C示出了用于TB级别残差编解码/解析的过程的示例实施例。

[0033] 图15A示出了其中在编解码单元级别分配常规二进制位的预算的示例实施例。

[0034] 图15B示出了基于CU级别分配的常规二进制位预算对与当前CU相关联的变换树进行编解码/解析的示例实施例。

[0035] 图15C示出了当预算在CU级别是固定的时编解码树编解码/解析过程的变型实施例。

[0036] 图15D和图15E示出了变换单元编解码/解析过程,当常规二进制位预算在CU级别是固定的时,该过程适用于本实施例。

[0037] 图16A和图16B示出了示例实施例,其中常规二进制位的预算在比CG级别更高的级别被分配,并且将使用两种类型的CG编解码/解析过程。

具体实施方式

[0038] 各种实施例涉及量化的变换系数的熵编解码。视频编解码器的该阶段也被称为残差编解码步骤。至少一个实施例目的在于在每单位面积的常规编解码的二进制位的最大数量的约束下,优化视频编解码器的编解码效率。

[0039] 本申请中描述的各种方法和其他方面可以用于修改如图1和图2所示的视频编码器100和解码器200的至少熵编解码和/或熵解码模块(145、230)。此外,尽管描述了涉及VVC(多功能视频编解码)或HEVC(高效视频编解码)规范的特定草案的原理,但是本申请各方面不限于VVC或HEVC,并且可以应用于例如其他标准和建议(无论是现有的还是未来开发的)以及任何这样的标准和建议(包括VVC和HEVC)的扩展。除非另有说明,或者技术上排除,本申请中描述的各方面可以被单独使用或者组合使用。

[0040] 图1示出了视频编码器100(例如HEVC编码器)的示例的框图。图1还可以示出其中对HEVC标准做出改进的编码器或者采用类似于HEVC的技术的编码器,例如由JVET(联合视频探索小组)为VVC开发的JEM(联合探索模型)或VTM(VVC测试模型)编码器。

[0041] 在被编码之前,视频序列可以经历预编码处理(101)。这例如通过对输入颜色图片应用颜色变换(例如,从RGB 4:4:4到YCbCr 4:2:0的转换)或者执行输入图片分量的重映射来执行,以便(例如,使用颜色分量之一的直方图均衡化)得到对压缩更有弹性的信号分布。元数据可以与预处理相关联并被附加到比特流。

[0042] 在HEVC中,为了对具有一个或多个图片的视频序列进行编码,图片被分割(102)成一个或多个条带,其中每个条带可以包括一个或多个条带段。条带段被组织成编解码单元、预测单元和变换单元。HEVC规范区分了“块”和“单元”,其中“块”寻址样点阵列中的特定区域(例如,亮度、Y),而“单元”包括所有经编码的颜色分量(Y、Cb、Cr或单色)的同位(collocated)块、语法元素以及与块相关联的预测数据(例如,运动矢量)。

[0043] 针对HEVC中的编解码,图片被分割成具有可配置大小的方形编解码树块(CTB),并且连续的一组编解码树块被分组为一个条带。编解码树单元(CTU)包含经编码的颜色分量的CTB。CTB是分割成编解码块(CB)的四叉树的根,编解码块可以被分割成一个或多个预测块(PB),并形成分割成变换块(TB)的四叉树的根。对应于编解码块、预测块和变换块,编解码单元(CU)包括预测单元(PU)和树结构的变换单元(TU)组,PU包括所有颜色分量的预测信息,而TU包括每个颜色分量的残差编解码语法结构。亮度分量的CB、PB和TB的大小适用于对应的CU、PU和TU。在本申请中,术语“块”可以用来指,例如,CTU、CU、PU、TU、CB、PB和TB中的任何一种。此外,“块”还可以用来指H.264/AVC或其他视频编解码标准中指定的宏块和分区,更一般地指各种大小的数据阵列。

[0044] 如下所述,在编码器100的示例中,图片由编码器元件来进行编码。要被编码的图片是以CU为单位被处理的。使用帧内或帧间模式对每个CU进行编码。当CU以帧内模式被编码时,它执行帧内预测(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)使用帧内模式或帧间模式中的哪一个来对CU进行编码,并通过预测模式标志来指示帧内/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。

[0045] 帧内模式中的CU是从同一条带内重建的相邻样点中预测的。HEVC中有一组35种帧内预测模式是可用的,包括DC、平面和33种角度预测模式。从与当前块相邻的行和列重建帧内预测参考。使用来自先前重建的块的可用样点,该参考在水平和垂直方向上延伸两倍的块大小。当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点。

[0046] 可以使用两种不同的选项来对当前块的适用亮度帧内预测模式进行编解码。如果适用模式包括在由三种最可能模式(MPM)的构建列表中,则通过MPM列表中的索引对该模式进行信令通知。否则,通过模式索引的固定长度二值化来对该模式进行信令通知。三种最可能的模式是从顶部和左侧相邻块的帧内预测模式中导出的。

[0047] 对于帧间CU,对应的编解码块被进一步分割成一个或多个预测块。在PB级别上执行帧间预测,并且对应的PU包含关于如何执行帧间预测的信息。可以用两种方法信令通知运动信息(例如,运动矢量和参考图片索引),即“合并模式”和“高级运动矢量预测(AMVP)”。

[0048] 在合并模式中,视频编码器或解码器基于已经被编解码的块来组装候选列表,并且视频编码器信令通知候选列表中的候选之一的索引。在解码器侧,基于信令通知的候选来重建运动矢量(MV)和参考图片索引。

[0049] 在AMVP中,视频编码器或解码器基于从已经被编解码的块确定的运动矢量来组装候选列表。然后,视频编码器信令通知候选列表中的索引,以识别运动矢量预测器(MVP),并信令通知运动矢量差(MVD)。在解码器侧,运动矢量(MV)被重建为MVP+MVD。适用的参考图片索引也在AMVP的PU语法中被显式地编解码。

[0050] 然后,对预测残差进行变换(125)和量化(130),其包括至少一个用于调节下面描述的色度量化的实施例。变换通常基于可分离的变换。例如,首先在水平方向上应用DCT变换,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,两个方向上使用的变换可以不同(例如,一个方向上DCT变换,另一个方向上DST变换),这导致了各种各样的2D变换,而在以前的编解码器中,用于给定的块大小的2D变换的种类通常是有限的。

[0051] 对量化的变换系数以及运动矢量和其他语法元素进行熵编解码(145)以输出比特流。编码器也可以跳过变换,并在 4×4 TU的基础上直接对未变换的残差信号应用量化。编码器也可以旁路变换和量化两者,也就是说,直接对残差进行编解码,而无需应用变换或量化过程。在直接PCM编解码中,不应用预测,并且直接将编解码单元样点编解码到比特流中。

[0052] 编码器对经编码的块进行解码,为进一步的预测提供参考。对量化的变换系数进行去量化(140)和逆变换(150)以解码预测残差。组合(155)经解码的预测残差和预测块,重建图像块。将环路内滤波器(165)应用于重建的图片,例如,以执行去方块/样点自适应偏移(SAO)滤波以减少编码伪影。经滤波的图片存储在参考图片缓冲器(180)中。

[0053] 图2示出了视频解码器200(例如HEVC解码器)的示例的框图。如下所述,在解码器200的示例中,由解码器元件对比特流进行解码。视频解码器200通常执行与图1中描述的编码过程相反的解码过程,其中编码过程执行视频解码作为编码视频数据的一部分。图2还可以示出对HEVC标准做出改进的解码器或采用类似于HEVC技术的解码器,例如JEM或VVC解码器。

[0054] 具体地,解码器的输入包括视频比特流,其可以由视频编码器100生成。首先对比特流进行熵解码(230)以获取变换系数、运动矢量、图片分割信息和其他编解码的信息。图

片分割信息指示CTU的大小,以及CTU被划分成CU的方式,如果适用,还可能指示CTU被划分成PU的方式。因此,解码器可以根据经解码的图片分割信息,将图片划分(235)成CTU,并将每个CTU划分成CU。对变换系数进行去量化(240),其包括至少一个用于调节下面描述的色度量化参数的实施例,并且对变换系数进行逆变换(250)以解码预测残差。

[0055] 组合(255)经解码的预测残差和预测块,重建图像块。可以从帧内预测(260)或运动补偿预测(即帧间预测)(275)中获得(270)预测块。如上所述,AMVP和合并模式技术可用于导出用于运动补偿的运动矢量,其可使用插值滤波器来计算参考块的子整数样点的插值。将环路内滤波器(265)应用于重建的图像。经滤波的图片存储在参考图片缓冲器(280)中。

[0056] 经解码的图片可以进一步经历解码后处理(285),例如逆颜色变换(例如从YCbCr 4:2:0到RGB 4:4:4的转换)或者执行与预编码处理(101)中执行的重映射过程相反的逆重映射。解码后处理可以使用在预编码处理中导出的并在比特流中信令通知的元数据。

[0057] 图3示出了其中实现各种方面和实施例的系统的示例的框图。系统300可以体现为包括以下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。这种设备的示例包括但不限于各种电子设备,例如个人计算机、膝上型计算机、智能手机、平板电脑、数字多媒体机顶盒、数字电视接收机、个人视频记录系统、连接的家用电器、编码器、代码转换器和服务器。系统300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300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300的元件通过内部总线310通信地耦合。在各种实施例中,系统300经由例如通信总线或通过专用输入和/或输出端口通信地耦合到其他类似系统或其他电子设备。在各种实施例中,系统300被配置成实现本文档中描述的一个或多个方面,例如上面描述的以及下面描述的修改的视频编码器100和视频解码器200。

[0058] 系统300包括至少一个处理器301,处理器301被配置为执行加载在其中的指令,用于实现例如本文档中描述的各种方面。处理器301可以包括嵌入式存储器、输入输出接口以及本领域已知的各种其他电路。系统300包括至少一个存储器302(例如,易失性存储器设备和/或非易失性存储器设备)。系统300包括存储设备304,存储设备304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304可以包括内部存储设备、附接的存储设备和/或网络可访问存储设备。

[0059] 系统300包括编码器/解码器模块303,其被配置为例如处理数据以提供经编码的视频或经解码的视频,并且编码器/解码器模块303可以包括其自己的处理器和存储器。编码器/解码器模块303表示可以被包括在设备中以执行编码和/或解码功能的(多个)模块。众所周知,设备可以包括编码和解码模块中的一者或两者。此外,如本领域技术人员所知,编码器/解码器模块303可以被实现为系统300的单独的元件,或者可以被合并处理器301中作为硬件和软件的组合。

[0060] 要加载到处理器301或编码器/解码器303上以执行本文档中描述的各种方面的程序代码可以被存储在存储设备304中,并且随后被加载到存储器302上以供处理器301执行。根据各种实施例,处理器301、存储器302、存储设备304和编码器/解码器模块303中的一个

或多个可以在执行本文档中描述的处理期间存储各种项目中的一个或多个。这种被存储的项目可以包括但不限于输入视频、经解码的视频或经解码的视频的部分、比特流、矩阵、变量以及来自等式、公式、运算和运算逻辑的处理的中间或最终结果。

[0061] 在几个实施例中,处理器301和/或编码器/解码器模块303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(例如,处理设备可以是处理器301或编码器/解码器模块303)外部的存储器用于这些功能中的一个或多个。外部存储器可以是存储器302和/或存储设备304,例如动态易失性存储器和/或非易失性闪存。在几个实施例中,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,例如针对MPEG-2、HEVC或VVC,诸如RAM的快速外部动态易失性存储器被用作视频编码和解码操作的工作存储器。

[0062] 如框309所示,可以通过各种输入设备向系统300的元件提供输入。这种输入设备包括但不限于(i)接收例如由广播公司通过空中发送的RF信号的RF部分、(ii)复合输入终端,(iii)USB输入终端和/或(iv)HDMI输入终端。

[0063] 在各种实施例中,如本领域中已知的,框309的输入设备具有相关联的相应输入处理元件。例如,RF部分可以与以下所需的元件相关联:(i)选择期望的频率(也被称为选择信号,或者将信号带限到频带)、(ii)下变频所选择的信号、(iii)再次带限到较窄的频带,以选择(例如)在某些实施例中可以被称为信道的信号频带、(iv)解调经下变频和经带限的信号、(v)执行误差校正以及(vi)解复用以选择期望的数据分组流。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、带限器、信道选择器、滤波器、下变频器、解调器、误差校正器和解复用器。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变频到较低频率(例如,中频或近基带频率)或基带。在一个机顶盒实施例中,RF部分及其相关联的输入处理元件接收通过有线(例如,电缆)介质发送的RF信号,并通过滤波、下变频和再次滤波到期望的频带来执行频率选择。各种实施例重新排列上述(和其他)元件的顺序,移除这些元件中的一些,和/或添加执行相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插入元件,例如插入放大器和模数转换器。在各种实施例中,RF部分包括天线。

[0064] 另外,USB和/或HDMI终端可以包括相应的接口处理器,用于通过通用USB和/或HDMI连接将系统300连接到其他电子设备。应当理解,输入处理的各种方面,例如里德-所罗门(Reed-Solomon)误差校正,可以根据需要在例如单独的输入处理IC内或处理器301内实现。类似地,可以根据需要在单独的接口IC内或处理器301内实现USB或HDMI接口处理的各方面。经解调、误差纠正和解复用的流被提供给各种处理元件,包括例如处理器301和编码器/解码器303,编码器/解码器303与存储器和存储元件结合操作,以根据需要处理数据流以供在输出设备上呈现。

[0065] 系统300的各种元件可以被装备在集成的外壳内。在集成的外壳内,各种元件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、布线和印刷电路板)来相互连接并在它们之间发送数据。

[0066] 系统300包括经由通信信道320使能与其他设备进行通信的通信接口305。通信接口305可以包括但不限于被配置为通过通信信道320发送和接收数据的收发器。通信接口305可以包括但不限于调制解调器或网卡,并且通信信道320可以例如在有线和/或无线介

质内实现。

[0067] 在各种实施例中,使用诸如IEEE 802.11的Wi-Fi网络将数据流式传输到系统300。这些实施例的Wi-Fi信号是通过适合于Wi-Fi通信的通信信道320和通信接口305接收的。这些实施例的通信信道320通常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在内的外部网络的访问,以允许流式传输应用和其他过顶(over-the-top)通信。其他实施例使用机顶盒向系统300提供流式数据,机顶盒通过输入框309的HDMI连接输送数据。还有其他实施例使用输入框309的RF连接向系统300提供流式数据。

[0068] 系统300可以向各种输出设备提供输出信号,输出设备包括显示器330、扬声器340和其他外围设备350。在实施例的各种示例中,其他外围设备350包括独立的DVR、磁盘播放器、立体声系统、照明系统和基于系统300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,使用信令(诸如AV链路、CEC或者在无论是否有用户干扰的情况下都使能设备到设备控制的其他通信协议)在系统300和显示器330、扬声器340或其他外围设备350之间传送控制信号。输出设备可以通过相应接口306、307和308经由专用连接通信地耦合到系统300。可选地,输出设备可以经由通信接口305使用通信信道320连接到系统300。显示器330和扬声器340可以与电子设备(例如,电视机)中的系统300的其他组件集成在单个单元中。在各种实施例中,显示接口306包括显示驱动器,例如定时控制器(T Con)芯片。

[0069] 例如,如果输入309的RF部分是单独的机顶盒的一部分,则可选择地显示器330和扬声器340也可以与一个或多个其他组件分离。在显示器330和扬声器340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)来提供输出信号。本文描述的实现可以以例如方法或过程、装置、软件程序、数据流或信号来实现。即使仅在单一实现形式的上下文中讨论(例如,仅作为方法讨论),所讨论的特征的实现也可以以其他形式实现(例如,装置或程序)。装置可以以例如适当的硬件、软件和固件来实现。方法可以在例如诸如处理器的装置中实现,处理器通常指处理设备,包括例如计算机、微处理器、集成电路或可编程逻辑设备。处理器还包括通信设备,例如计算机、蜂窝电话、便携式/个人数字助理(“PDA”)以及促进终端用户之间信息通信的其他设备。

[0070] 图4A示出了压缩域中的编解码树单元和编解码树的示例。在HEVC视频压缩标准中,采用运动补偿的时域预测来利用视频的连续图片之间存在的冗余。为此,图片被分割成所谓的编解码树单元(CTU),其大小通常为 64×64 、 128×128 或 256×256 像素。在压缩域中(例如在CTU的四叉树划分中)每个CTU由编解码树表示。每个叶子被称为编解码单元(CU)。

[0071] 图4B示出了将CTU划分成编解码单元、预测单元和变换单元的示例。然后给每个单元一些帧内或帧间预测参数(预测信息)。为此,它在空域上被分割成一个或多个预测单元(PU),每个PU被分配一些预测信息,例如运动矢量。在CU级别分配帧内或帧间编解码模式。

[0072] 在本申请中,术语“重建的”和“经解码的”可以互换使用,术语“经编码的”或“经编解码的”可以互换使用,术语“图像”、“图片”和“帧”可以互换使用。通常,但不是必须的,术语“重建的”用于编码器侧,而“经解码的”用于解码器侧。术语“块”或“图片块”可以用来指CTU、CU、PU、TU、CB、PB和TB中的任何一个。此外,术语“块”或“图片块”可以用来指在H.264/AVC或其他视频编解码标准中指定的宏块、分割和子块,并且更一般地指各种大小的样点阵列。

[0073] 图5示出了两个标量量化器在依赖标量量化中的使用。如在JVET(文稿JVET-

J0014)中提出的依赖标量量化使用两个具有不同重建级别的标量量化器进行量化。与传统的标量量化(例如在HEVC和VTM-1中使用的)相比,这种方法的主要效果是,变换系数的可容许重建值的集合依赖于重建顺序中在当前变换系数级别之前的变换系数级别的值。依赖标量量化的方法是通过以下方式实现的:(a)定义两个具有不同重建级别的标量量化器,以及(b)定义两个标量量化器之间的切换过程。如图5所示,示出了由Q0和Q1表示的所使用的两个标量量化器。可用的重建级别的位置由量化步长 Δ 唯一指定。当忽略变换系数的实际重建使用整数算法的事实时,两个标量量化器Q0和Q1的特征如下:

[0074] Q0:第一量化器Q0的重建级别由量化步长 Δ 的偶数整数倍给出。当使用该量化器时,根据下式计算去量化的变换系数 t'

[0075] $t' = 2 \cdot k \cdot \Delta$,

[0076] 其中 k 表示相关联的量化系数(发送的量化指数)。

[0077] Q1:第二量化器Q1的重建级别由量化步长 Δ 的奇数整数倍加上等于零的重建级别给出。根据量化系数 k ,去量化的变换系数 t' 计算如下:

[0078] $t' = (2 \cdot k - \text{sgn}(k)) \cdot \Delta$,

[0079] 其中 $\text{sgn}(\cdot)$ 符号函数,定义如下:

[0080] $\text{sgn}(x) = (k == 0 ? 0 : (k < 0 ? -1 : 1))$ 。

[0081] 所使用的标量量化器(Q0或Q1)没有在比特流中被显式地信令通知。相反,用于当前变换系数的量化器由在编解码/重建顺序中在当前变换系数之前的量化系数的奇偶性以及其在下文中引入的有限状态机的状态来确定。

[0082] 图6A和图6B示出了VVC中的标量量化器之间切换的机制。如图6A所示,两个标量量化器(Q0和Q1)之间的切换经由具有四种状态的有限状态机来实现,这四种状态分别被标记为0、1、2或3。对于给定的量化系数,所考虑的有限状态机的状态由在编解码/重建顺序中在当前量化系数之前的量化系数 k 的奇偶性以及当处理该先前系数时所考虑的有限状态机的状态唯一地确定。在变换块的逆量化开始时,状态被设置为等于0。以扫描顺序(即,以与熵解码相同的顺序)重建变换系数。在当前变换系数被重建之后,状态被更新。 k 是量化系数。下一个状态取决于当前状态和当前量化系数 k 的奇偶性($k \& 1$):

[0083] $\text{state} = \text{stateTransTable}[\text{current_state}][k \& 1]$,

[0084] 其中 stateTransTable 表示图6B所示的状态转换表,运算符 $\&$ 指定二进制补码算法(two's-complement arithmetic)中的逐比特“与”运算。

[0085] 此外,包含在所谓的变换块(TB)中的量化系数可以如下所述被熵编解码和解码。

[0086] 首先,变换块被划分成被称为编解码组(有时也被称为系数组和缩写的CG)的量化系数 4×4 子块。熵编解码/解码由几个扫描遍次(scanning pass)组成,这些扫描遍次根据图6C所示的对角线扫描顺序对TB进行扫描。

[0087] VVC中的变换系数编解码包括五个主要步骤:扫描、最后有效系数(last significant coefficient)编解码、有效性图(significance map)编解码、系数级别余数编解码、绝对级别和符号数据编解码。

[0088] 图6C示出了 8×8 变换块中在VVC中使用的CG与系数之间的扫描顺序。然后,在TB上的扫描遍次包括根据对角线扫描顺序顺序处理每个CG,并且每个CG内的16个系数也根据所考虑的扫描顺序被扫描。扫描遍次从TB中的最后有效系数开始,并且处理所有系数,直到DC

系数。

[0089] 变换系数的熵编解码包括以下列表中多达7个语法元素：

[0090] -coded_sub_block_flag:系数组 (CG) 的有效性

[0091] -sig_flag:系数的有效性 (零/非零)。

[0092] -gt1_flag:指示系数级别的绝对值是否大于1

[0093] -par_flag:指示大于1的系数的奇偶性

[0094] -gt3_flag:指示系数级别的绝对值是否大于3

[0095] -余数:系数级别绝对值的剩余值 (如果该值大于先前遍次中编解码的值)

[0096] -abs_level:系数级别的绝对值的值 (如果对于二进制位预算的最大数量问题而言,没有为当前系数信令通知CABAC二进制位)

[0097] -sign_data:所考虑CG中包含的所有有效系数的符号。它包括一系列二进制位,每个二进制位信令通知每个非零变换系数的符号 (0:正,1:负)。

[0098] 一旦通过解码上述元素的子集 (除符号之外) 知道了量化系数的绝对值,那么就不再针对该系数的绝对值对进一步的语法元素进行编解码。以同样的方式,仅针对非零系数信令通知符号标志。

[0099] 在进入下一个CG之前,对给定的CG的所有必要扫描遍次进行编解码,直到该CG中的所有量化系数都可以被重建。

[0100] 整个解码TB解析过程由以下主要步骤组成:

[0101] 1.解码最后有效系数坐标。这包括以下语法元素:last_sig_coeff_x_prefix、last_sig_coeff_y_prefix、last_sig_coeff_x_suffix和last_sig_coeff_y_suffix。这为解码器提供了整个TB中最后非零系数的空域位置 (x坐标和y坐标)。

[0102] 然后,对于从包含TB中最后有效系数的CG到TB中左上方的CG的每个连续CG,应用以下步骤:

[0103] 2.解码CG有效性标志 (在VVC规范中被称为coded_sub_block_flag)。

[0104] 3.解码所考虑的CG中每个系数的有效系数标志。这对应于语法元素sig_flag。这指示CG中的哪些系数是非零的。

[0105] 在所考虑的CG中,对于已知非零的系数,下一个解析阶段与系数级别有关。这涉及以下语法元素:

[0106] 4.gt1_flag:该标志指示当前系数的绝对值是否大于1。否则,绝对值等于1。

[0107] 5.par_flag:该标志指示当前量化系数是否为偶数。如果当前量化系数的gt1_flag为真,则对其进行编解码。如果par_flag为零,则量化系数为偶数,否则为奇数。在解码器侧解析par_flag之后,部分经解码的量化系数被设置为等于(1+gt1_flag+par_flag)。

[0108] 6.gt3_flag:该标志指示当前系数的绝对值是否大于3。如果不是,则绝对值等于1+gt1_flag+par_flag。如果1+gt1_flag+par_flag大于或等于2,则对gt3_flag进行编解码。一旦gt3_flag被解析,则量化系数值在解码器侧变为1+gt1_flag+par_flag+(2*GT3_flag)。

[0109] 7.余数:该余数对系数的绝对值进行编码。如果部分经解码的绝对值大于或等于4,则适用。注意,在VVC草案3的示例中,对于每个编解码组,常规编解码二进制位预算的最大数量是固定的。因此,对于一些系数,可以仅信令通知sig_flag、gt1_flag和par_flag元

素,而对于其他系数,也可以信令通知gt3_flag。因此,因此根据部分经解码的量化系数,相对于所考虑的系数的已经解码的标志,来计算编解码和解析的余数值。

[0110] 8.abs_level:这指示系数的绝对值,对于常规编解码二进制位的最大数量的问题而言,在所考虑的CG中没有为这些系数编解码任何标志(sig_flag、gt1_flag、papr_flag或gt3_flag当中)。这个语法元素是类似于余数语法元素Rice-Golomb二进制化和旁路编解码的。

[0111] 9.sign_flag:这指示非零系数的符号。这是旁路编解码的。

[0112] 图7A示出了根据VVC草案4的用于变换块的编解码/解析的语法元素。编解码/解析包括4遍过程。可以看出,对于给定的变换块,首先对最后有效系数的位置进行编解码。然后,对于从包含最后有效系数(排除)到第一个CG的CG中的每个CG,CG的有效性被信令通知(coded_sub_block_flag),并且如图7B所示,在CG有效的情况下,发生该CG的残差编解码。

[0113] 图7B示出了VVC草案4中指定的CG级别残差编解码语法。它信令通知先前介绍的语法元素sig_flag、gt1_flag、par_flag、gt3_flag、余数、abs_level和所考虑的CG中的符号数据。该图示出了根据VVC草案4的sig_flag、gt1_flag、par_flag、gt3_flag、余数和abs_level语法元素的信令通知和解析。EP的意思是“等概率”,这意味着相关的二进制位不是算术编解码的,而是以旁路模式编解码的。旁路模式直接写入/解析比特,该比特通常等于要编码或解析的二进制语法元素(二进制位)。

[0114] 此外,VVC草案4指定了从CU级别到残差子块(CG)级别的分层语法布置。

[0115] 图8A示出了在VVC草案4的示例中指定的CU级别语法。它包括信号通知所考虑的CU的编解码模式。如果CU处于合并跳过模式,则cu_skip_flag会信令通知。如果不是,因此如果当前CU的预测模式是通过帧内预测(cuPredMode=MODE_INTRA)或帧间预测(cuPredMode=MODE_INTER)编解码的,则cu_pred_mode_flag指示变量cuPredMode的值。

[0116] 在非跳过和非帧内模式的情况下,下一个标志指示当前CU使用帧内块复制(IBC)模式。该语法之后包括当前CU的帧内或帧间预测数据。语法的结尾专用于与当前CU相关联的变换树的信令通知。这从cu_cbf语法元素开始,它指示为当前CU对一些非零残差数据进行编解码。根据图8B所示的语法布置,在该标志等于真的情况下,信令通知与当前CU相关联的transform_tree。

[0117] 图8B示出了VVC草案4的示例的transform_tree语法结构。如果CU包含一个或多个变换单元(TU),则它基本上包括信令通知。首先,如果CU大小高于最大允许TU大小,则以四叉树方式将CU划分成4个子变换树。如果不是,并且如果没有ISP(帧内子分区)或SBT(子块变换)用于当前CU,则当前变换树不被分割并且仅包含一个TU,这通过图8C的变换单元语法表来信令通知。否则,如果CU是帧内的,并且使用了ISP(帧内子分区)模式,则该CU也被划分成多个(实际上是2个)TU。通过图8C的变换单元语法表,连续地信令通知2个TU中的每一个。否则,如果CU是帧间的并且使用了SBT(子块变换)模式,则根据CU级别SBT相关的解码的语法将CU划分成2个TU。通过图8C的变换单元语法表,连续地信令通知2个TU中的每一个。

[0118] 图8C示出了VVC草案4的示例中的变换单元级别语法布置。它由以下部分组成。首先,tu_cbf_luma、tu_cbf_cb和tu_cbf_cr标志分别指示非零残差数据包含在当前TU中的每个变换块中,对应于每个分量。对于每个分量,如果对应的cbf标志为真,则使用变换块级别residual_tb语法表来编解码对应的残差变换块。transform_unit级别语法还包括一些经

编解码的变换和量化的块相关的语法,即增量QP信息(如果存在)和用于对所考虑的TU进行编解码的变换类型。

[0119] 图9A和图9B分别示出了CABAC解码和编码过程。CABAC代表上下文自适应二进制算术编解码(CABAC),例如因为它以良好的压缩效率来提供无损压缩,所以是一种在HEVC或VVC使用的熵编码形式。图9A的过程的输入包括经编解码的比特流,通常符合HEVC规范或其进一步发展。在解码过程的任何时候,解码器都知道接下来要解码哪个语法元素。这在标准化比特流语法和解码过程中有充分说明。此外,它还要知道要解码的当前语法元素是如何被二进制的(即,被表示为称为二进制位的二进制符号序列,每个二进制位等于“1”或“0”),以及二进制位字符串的每个二进制位是如何被编码的。

[0120] 因此,CABAC解码过程的第一阶段(图9A的左侧)解码一系列的二进制位。对于每个二进制位,知道其是根据旁路模式还是常规模式被编码的。在旁路模式下,只需从比特流中读取一个比特,并将所获得的值分配给当前二进制位。这种模式的优点是简单,因此快速,并且不需要大量资源。因此,对于具有均匀统计分布(即,等于‘1’或‘0’的相等概率)的二进制位,它通常是有效的。

[0121] 相反,如果没有以旁路模式编解码当前二进制位,则意味着已经以所谓的常规编解码(即通过基于上下文算术编解码)对它进行编解码。这种模式更加耗费资源。

[0122] 在这种情况下,所考虑的二进制位的解码如下进行。首先,获得用于解码当前二进制位的上下文。它由图9A的上下文建模器给出。上下文目标是在给定一些上下文先验或信息X的情况下,获得当前二进制位具有值“0”的条件概率。这里的先验X是在当前二进制位被解码时以同步方式,在编码器和解码器侧都可用的一些已经解码的语法元素的值。

[0123] 通常,在标准中规定了用于解码二进制位的先验X,并且选择该先验X是因为它与要解码的当前二进制位在统计上相关。使用这种上下文信息的好处在于,这样降低了对二进制位进行编解码的速率成本。这是基于这样的事实,即给定X的二进制位的条件熵低,因为二进制位和X是相关的。下列关系在信息论中是众所周知的:

[0124] $H(\text{bin}|X) < H(\text{bin})$

[0125] 这意味着如果二进制位和X是统计相关的,那么二进制位的知道X的条件熵低于二进制位的熵。上下文信息X因此被用于获得二进制位为“0”或“1”的概率。在给定这些条件概率的情况下,图14的常规解码引擎执行二进制值二进制位的算术解码。然后,在知道当前上下文信息X的情况下,二进制位的值被用于更新与当前二进制位相关联的条件概率的值。这被称为图9A上的上下文模型更新步骤。只要二进制位正在被解码(或编解码)就更新每个二进制位的上下文模型,这允许逐步细化每个二进制元素的上下文建模。因此,CABAC解码器逐步学习每个常规编码的二进制位的统计行为。

[0126] 上下文建模器和上下文模型更新步骤在编码器和解码器侧是严格相同的操作。

[0127] 当前二进制位的常规算术解码或其旁路解码(取决于其编解码的方式)会产生一系列经解码的二进制位。

[0128] 如图9A右侧所示,CABAC解码的第二阶段包括将这一系列的二进制符号转换成更高级别的语法元素。语法元素可以采用标志的形式,在这种情况下,它直接采用当前解码的二进制位的值。另一方面,如果当前语法元素的二值化对应于根据所考虑的标准规范的多个二进制位的集合,则发生在图9A上被称为“二进制码字到语法元素”的转换步骤。

[0129] 这进行了与由如图9B所示的编码器完成的二值化步骤的相反的步骤。因此,这里执行的逆转换包括基于这些语法元素各自的经解码的二进制版本获得它们的值。

[0130] 图1的编码器100、图2的解码器200和图3的系统1000适于实现下述实施例中的至少一个。

[0131] 在当前的视频编解码系统(例如VVC草案4)中,对系数组编解码过程施加了一些硬约束,其中一侧的sig_flag、gt1_flag和parity_flag语法元素以及另一侧的gt3_flag语法元素可以采用常规CABAC二进制位的硬编解码的最大数量。更准确地说,4×4编解码组(或系数组,或CG)级别的约束确保每单位面积的CABAC解码器引擎必须解析常规二进制位的最大数量。然而,由于常规二进制位CABAC编解码的有限使用,它可能导致非率失真最佳的视频压缩。实际上,当图片允许有限数量的常规二进制位时,那么在所考虑的要编解码的图片的一部分包含以跳过模式编解码的有效区域(因此没有任何残差)而图片的另一个有效区域将采用残差编解码的情况下,该总体预算可能在很大程度上没有得到充分利用。在这种情况下,对常规二进制位的最大数量的低级别(即4×4CG级别)约束可能导致图片中常规二进制位的总数远低于可接受的图片级别阈值。

[0132] 至少一个实施例涉及在块及其系数组的残差编解码过程中,处理对二进制位的常规CABAC编解码的最大使用的高级约束,使得高级约束得到遵守,并且与当前方法相比,压缩效率得到提高。换句话说,在比CG更大的图片区域(从而覆盖多个CG)上分配常规编解码的二进制位的预算,并且该预算由每单位面积的常规二进制位的平均允许数量确定。例如,可以允许每个样点平均1.75个常规编解码的二进制位。然后,该预算可以被更有效地分配给更小的单位。在不同的实施例中,在变换块或变换单元或编解码单元或编解码树单元或图片级别设置更高级别的约束。这些实施例可以由如图10所示的CABAC优化器来实现。

[0133] 在至少一个实施例中,在变换单元级别设置更高级别的约束。在这样的实施例中,允许整个变换单元进行编码/解码的常规二进制位的数量是根据每单位面积的常规二进制位的平均允许数量来确定的。从所获得的TU的常规二进制位的预算中,导出TU中每个变换块(TB)所允许的常规二进制位的数量。变换块是属于同一TU和同一颜色分量的一组变换系数。接下来,给定变换块中允许的常规二进制位的数量,在对于整个TB的该允许的常规二进制位的数量的约束下应用残差编解码或解码。因此,这里提出了一种改进的残差编解码和解码过程,它考虑了TB级别常规二进制位预算,而不是CG级别常规二进制位预算。为此目的提出了几个实施例。

[0134] 图10示出了根据本原理的实施例的包括CABAC优化器的CABAC编码过程。在至少一个实施例中,CABAC优化器190使用针对编解码组的集合的常规编解码来处理表示经编码(或将被编码)的二进制位的最大数量的预算。该预算例如通过将常规编解码的二进制位的每样点预算乘以所考虑的数据单元的表面(即其中包含的样点数量)来确定。例如,在常规编解码的二进制位的预算在TB级别是固定的情况下,每个样点的常规编解码的二进制位的允许数量乘以所考虑的变换块中包含的样点数量。CABAC优化器190的输出以常规或旁路编解码模式控制编解码,因此极大地影响了编解码的效率。

[0135] 图11示出了根据本原理实施例的编码过程中使用的CABAC优化器的示例流程图。该流程图针对所考虑的每个新的编解码组的集合执行。因此,根据不同的实施例,它可以发生在每个图片、每个CTU、每个CU、每个TU或每个TB上。在步骤191中,处理器301为如上所述

确定的常规编解码分配预算。在步骤192,处理器301循环该编解码组的集合,并在步骤193检查是否到达最后一个编解码组。在步骤194,对于每个编解码组,处理器用常规编解码的二进制位的输入预算来处理编解码组。在所考虑的编解码组的处理过程期间,常规编解码的二进制位的预算正在减少,即,每当以常规模式对二进制位进行编解码时,它就递减。常规编解码的二进制位的预算作为编解码组编解码或解码过程的输出参数被返回。

[0136] 图12示出了使用CABAC优化器对编解码组进行编码/解码的修改过程的示例。如上所述,专用于残差编解码的常规CABAC二进制位的数量的分配是在比 4×4 编解码组更高的级别上执行的。为此,与传统功能相比,通过添加表示在CG正在被编解码或解码时常规二进制位的当前预算的输入参数来修改对编解码组进行编码/解码的过程。因此,根据用于对当前CG进行编码/解码的常规二进制位的数量,通过对当前CG进行编解码或解码来修改该预算。表示常规二进制位的预算的输入/输出参数在图12中被称为numRegBins_in_out。在给定的CG中对残差数据本身进行编解码的过程可以与传统上相同,除了允许的常规CABAC二进制位的数量由外部手段给出。此外,如图12的情况,每当编解码/解码一个常规二进制位,该预算就递减一。

[0137] 因此,本公开的至少一个实施例包括将常规二进制位的预算作为系数组编解码/解析功能的输入/输出参数进行处理。因此,可以在比CG编解码/解析级别更高的级别确定常规二进制位的预算。不同的实施例提出在以下不同的级别确定预算:变换块或变换单元或编解码单元或编解码树单元或图片级别。

[0138] 图13A示出了其中在变换块级别确定常规二进制位的预算的示例实施例。在这样的实施例中,预算被确定为以下两个主要参数的函数:当前变换块的大小和对于给定的单位图片区域固定的常规二进制位的基本预算。这里,所考虑的面积单位是样点。在当前的VVC草案2中,由于针对 4×4 CG允许32个常规二进制位,这意味着允许每个分量样点平均2个常规二进制位。所提出的变换块级别常规二进制位分配基于每单位面积常规二进制位的平均比率。

[0139] 在至少一个实施例中,为所考虑的变换块分配的预算被计算为变换块表面与每样点分配的常规数量的乘积。接下来,对于所考虑的变换块中的每个有效系数组,将预算传递给CG残差编解码/解析过程。

[0140] 图13B示出了其中根据最后有效系数的位置在变换块级别确定常规二进制位的预算的示例实施例。根据该变型实施例,根据所考虑的变换块中的最后有效系数的位置计算当前变换块的允许常规二进制位的数量。这样的方法的优点是,当分配常规二进制位时,更好地拟合所考虑的变换块中包含的能量。典型地,更多的二进制位可以被分配给高能量变换块,而较少的常规二进制位可以被分配给低能量变换块。

[0141] 图14A示出了其中在变换单元级别确定常规二进制位的预算的示例实施例。如图所示,unitary_budget,即每样点允许的常规二进制位的平均比率仍然等于2。TU级别常规二进制位的预算是通过将该比率乘以所考虑的变换单元中包含的样点总数来确定的,即在 $4:2:0$ 颜色格式的示例中,宽度 $\times$ 高度 $\times 3/2$ 。然后该整体预算被用于对包含在所考虑的变换单元中的所有变换块的编解码。它作为输入/输出参数被传递给每个颜色分量的残差变换块编解码/解析过程。换句话说,每当对所考虑的变换单元中的变换块进行编解码/解析时,预算都会减少在对该变换单元中经连续编解码/解析的变换块的编解码/解析中的常规

二进制位的数量。

[0142] 图14B示出了示例实施例,其中根据不同变换块之间的相对表面以及根据在所考虑的变换单元中已经被编解码/解析的变换块中使用的常规二进制位,在变换块单元级别确定并且在变换块之间分配常规二进制位的预算。在该实施例中,分配给亮度TB的常规二进制位是整体TU级别预算的三分之二。接下来,在对亮度TB进行编解码/解析后,Cb TB预算是剩余预算的一半。最后,在对前两个TB进行编解码/解析后,将Cr TB预算设置为剩余的TU级别预算。此外,注意,在另一个实施例中,基于该TU中的一个或多个TB是用空残差被编解码的知识,整个变换单元的预算可以在该TU中的变换块之间共享。例如,通过分析tu_cbf_luma、tu_cbf_cb或tu_cbf_cr标志就可以知道这一点。因此,如图14C所示的修改过程所示,TB级别残差编解码/解析过程适用于该变型实施例。这样的实施例导致比预算在层级中的较低级别被分配时更好的编解码效率。事实上,如果一些CG或TB具有非常低的能量,则与其他CG或TB相比,它可能会采用减少的二进制位的数量。因此,在需要编码/解析更大数量的二进制位的CG或TB中,可以使用更大数量的常规二进制位。

[0143] 图15A示出了其中在编解码单元级别分配常规二进制位的预算的示例实施例。它的计算类似于前面的TU级别固定实施例,但是基于常规二进制位的采样率和CU大小。然后,计算出的常规二进制位预算被传递到图15B的变换树编解码/解析过程,该过程示出了基于分配常规二进制位预算的CU级别对与当前CU相关联的变换树进行的适配编解码/解析。这里,CU级别基本上连续地被传递给包含在所考虑的变换树中的每个变换单元的编解码。此外,在每个TU的编解码中采用的常规二进制位的数量也在减少。图15C示出了当预算在CU级别是固定的时编解码树编解码/解析过程的变型实施例。典型地,当在所考虑的变换树中对当前TU进行编解码时,该过程根据已经被编解码/解析的TU中使用的常规二进制位的数量来更新整体CU级别预算。例如,在帧间CU中的SBT(子块变换)模式的情况下,VVC草案4声明CU被划分成2个TU,并且其中一个具有空残差。在这种情况下,该实施例提出,整个CU级别常规二进制位预算专用于具有非零残差的TU,这使得在SBT模式下对于这样的帧间CU的编码效率更好。此外,在是ISP(帧内子分割)的情况下,帧内CU的这种VVC分割模式将CU划分成多个TU。在这种情况下,TU的编解码/解析试图重用分配给同一CU中先前TU的未使用的常规二进制位预算。图15D和图15E示出了变换单元编解码/解析过程,当常规二进制位预算在CU级别是固定的时,该过程适用于本实施例。它们包括图14A和图14B的过程对本实施例的各自的适应。

[0144] 在至少一个实施例中,在残差编解码中使用的常规CABAC二进制位的预算在CTU级别上是固定的。这允许更好地利用允许的常规二进制位的整体预算,因此与先前的实施例相比提高了编解码效率。典型地,最初专用于跳过CU的二进制位可以有利地用于其他非跳过CU。

[0145] 在至少一个实施例中,在残差编解码中使用的常规CABAC二进制位的预算在图片级别上是固定的。这允许甚至更好地利用允许的常规二进制位的整体预算,因此与先前的实施例相比提高了编解码效率。典型地,最初专用于跳过CU的二进制位可以有利地用于其他非跳过CU。

[0146] 在至少一个实施例中,根据所考虑的图片所属的时间层/深度,给定图片中允许的常规二进制位的平均比率是固定的。例如,在随机访问编解码结构中,一组图片的时域结构

符合层次B图片布置。在这种配置中,图片被组织在时域可缩放层中。来自较高层的图片依赖于来自较低时域层的参考图片。相反,较低层的图片不依赖于任何较高层的图片。较高的时域层通常用比较低层中的图片更高的量化参数进行编解码,因此质量较低。此外,来自较低层的图片严重影响整个序列的整体编解码效率。因此,以最佳编解码效率编码这些图片是有意义的。根据本实施例,提出了向来自较低层的图片分配比来自较高时域层的图片更高的基于样点的常规CABAC二进制位的比率。

[0147] 图16A和图16B示出了示例实施例,其中常规二进制位的预算被分配在比CG级别更高的级别,并且使用两种类型的CG编解码/解析过程。图16A所示的第一种类型被称为“all_bypass”,包括以旁路模式对与CG相关联的所有二进制位进行编解码。这意味着CG中的变换系数的幅度仅通过abs_level语法元素被编解码。图16B所示的第二种类型被称为“all_regular”,包括以常规模式对对应于sig_flag、gt1_flag、par_flag和gt3_flag语法元素的所有二进制位进行编解码。在该实施例中,给定固定在高于CG级别的级别的常规二进制位预算,根据当前考虑的常规二进制位的考虑预算是否已经被完全使用,TB编解码过程可以在“all_regular”CG编解码模式和“all_bypass”CG编解码模式之间切换。

[0148] 各种实现方式包括解码。本申请中使用的“解码”可以涵盖例如对接收到的经编码的序列执行以便产生适合于显示的最终输出的全部或部分过程。在各种实施例中,这些过程包括通常由解码器执行的一个或多个过程,例如熵解码、逆量化、逆变换和差分解码。在各种实施例中,这样的过程也可以包括或者可替换地包括由本申请中描述的各种实现方式的解码器执行的过程,例如图10至图16中呈现的实施例。

[0149] 作为进一步的示例,在一个实施例中,“解码”仅指熵解码,在另一个实施例中,“解码”仅指差分解码,在另一个实施例中,“解码”指熵解码和差分解码的组合。基于具体描述的上下文,短语“解码过程”是旨在具体指操作的子集还是通常指更广泛的解码过程将是清楚的,并且被认为是本领域技术人员很好理解的。

[0150] 各种实现方式包括编码。以类似于上面关于“解码”的讨论的方式,本申请中使用的“编码”可以涵盖例如对输入视频序列执行以便产生编码比特流的全部或部分过程。在各种实施例中,这些过程包括通常由编码器执行的一个或多个过程,例如,分割、差分编码、变换、量化和熵编码。在各种实施例中,这样的过程也包括或者可替换地包括由本申请中描述的各种实现方式的编码器执行的过程,例如,图10至图16的实施例。

[0151] 作为进一步的示例,在一个实施例中,“编码”仅指熵编码,在另一个实施例中,“编码”仅指差分编码,在另一个实施例中,“编码”指差分编码和熵编码的组合。基于具体描述的上下文,短语“编码过程”是旨在具体指操作的子集还是通常指更广泛的编码过程将是清楚的,并且被认为是本领域技术人员很好理解的。

[0152] 注意,本文使用的语法元素是描述性术语。因此,它们不排除使用其他语法元素名称。

[0153] 本申请描述了各种方面,包括工具、特征、实施例、模型、方法等。这些方面中的许多方面被具体地描述,并且至少为了显示单独的特性,通常以听起来可能是限制性的方式来描述。然而,这是出于描述清楚的目的,并不限制那些方面的应用或范围。事实上,所有不同的方面可以被组合和互换以提供进一步的方面。此外,这些方面也可以与早期申请中描述的方面相结合和互换。本申请中描述和预期的方面可以以许多不同的形式实现。上面的

图1、图2和图3提供了一些实施例,但是也预期了其他实施例,并且对图的讨论并不限制实现方式的广度。

[0154] 在本申请中,术语“重建的”和“经解码的”可以互换使用,术语“像素”和“样点”可以互换使用,术语“图像”、“图片”和“帧”可以互换使用。通常,但不是必须的,术语“重建的”用于编码器侧,而“经解码的”用于解码器侧。

[0155] 本文描述了各种方法,并且每种方法包括用于实现所描述的方法的一个或多个步骤或动作。除非该方法的正确操作需要步骤或动作的特定顺序,否则可以修改或组合特定步骤和/或动作的顺序和/或使用。

[0156] 在本申请中使用了各种数值,例如关于块大小。特定值是为了示例的目的,并且所描述的方面不限于这些特定值。

[0157] 涉及“一个实施例”或“实施例”或“一个实现方式”或“实现方式”以及它们的其他变型意味着结合实施例描述的特定特征、结构、特性等被包括在至少一个实施例中。因此,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”的出现,以及在整个说明书的各个地方出现的任何其他变型,不一定都指同一实施例。

[0158] 此外,本申请或其权利要求可能涉及“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。

[0159] 此外,本申请或其权利要求可能涉及“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)、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。

[0160] 此外,本申请或其权利要求可能涉及“接收”各种信息。与“访问”一样,接收旨在是一个宽泛的术语。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)中的一个或多个。此外,在诸如存储信息、处理信息、发送信息、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的操作期间,通常以这样或那样的方式涉及“接收”。

[0161] 应当理解,例如在“A/B”、“A和/或B”和“A和B中的至少一个”的情况下,使用“/”,“和/或”和“至少一个”中的任何一个旨在涵盖仅选择第一个列出的选项(A),或仅选择第二个列出的选项(B),或选择两个选项(A和B)。作为另一个示例,在“A、B和/或C”和“A、B和C中的至少一个”的情况下,这样的措辞旨在涵盖仅选择第一个列出的选项(A),或仅选择第二个列出的选项(B),或仅选择第三个列出的选项(C),或仅选择第一个和第二个列出的选项(A和B),或仅选择第一和第三个列出的选项(A和C),或仅选择第二和第三个列出的选项(B和C),或选择所有三个选项(A和B和C)。如本领域和相关领域的普通技术人员容易明白的,这可以扩展到所列的许多项目。

[0162] 对于本领域技术人员来说显而易见的是,实现方式可以产生各种信号,这些信号被格式化以携带例如可以被存储或发送的信息。信息可以包括例如用于执行方法的指令,或者由所描述的实现方式之一产生的数据。例如,信号可以被格式化以携带所描述的实施例的比特流。这种信号可以被格式化为例如电磁波(例如,使用频谱的射频部分)或基带信号。格式化可以包括,例如,编码数据流和用经编码的数据流调制载波。信号携带的信息可以是例如模拟或数字信息。众所周知,信号可以通过各种不同的有线或无线链路发送。信号可以存储在处理器可读介质上。

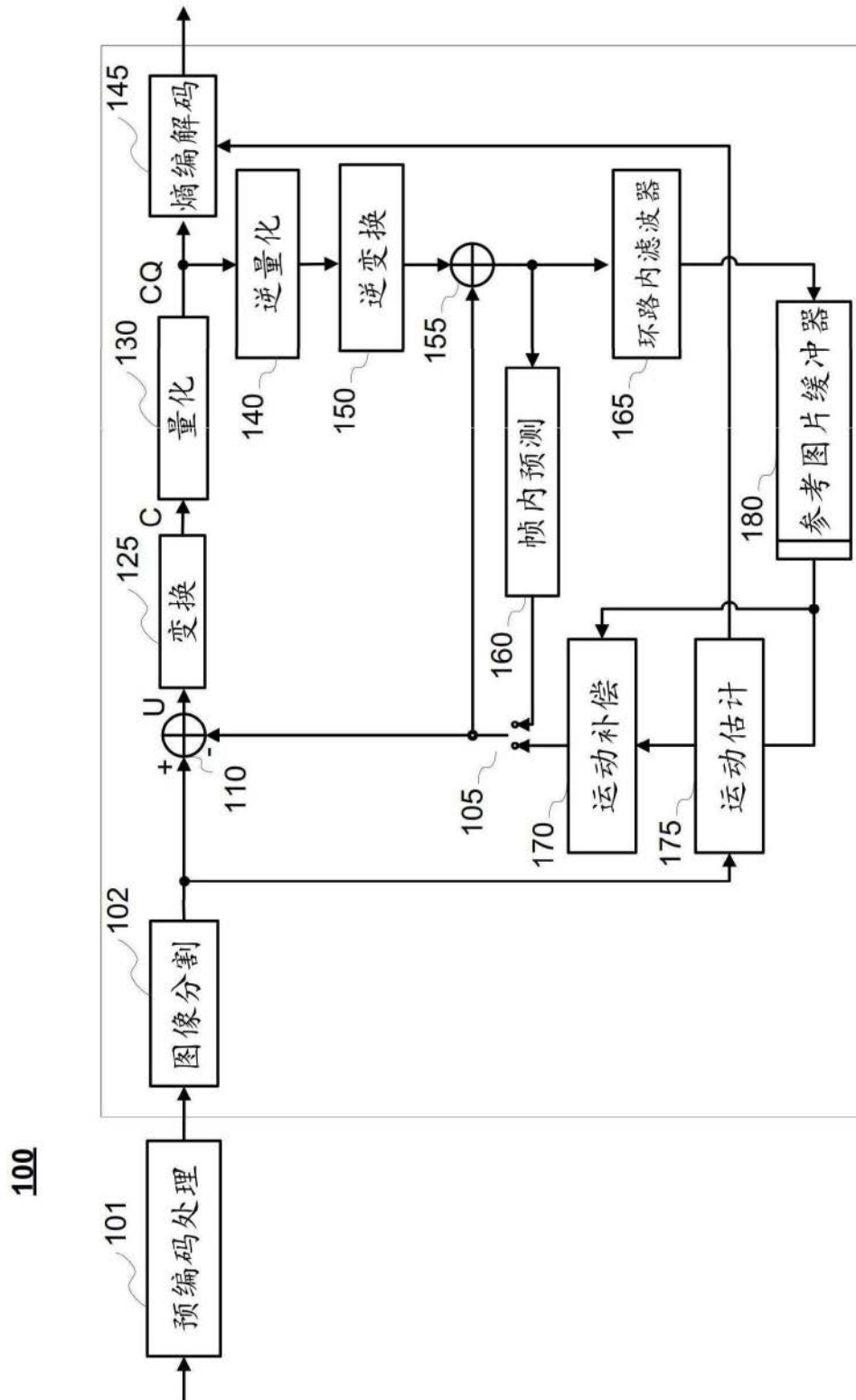


图1

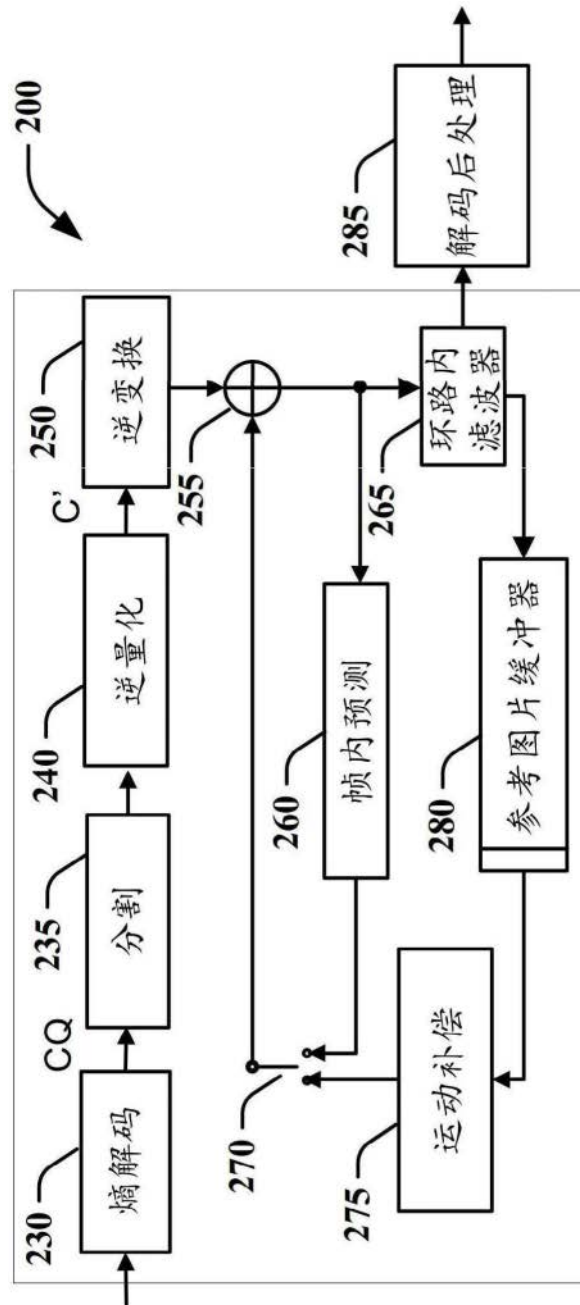


图2

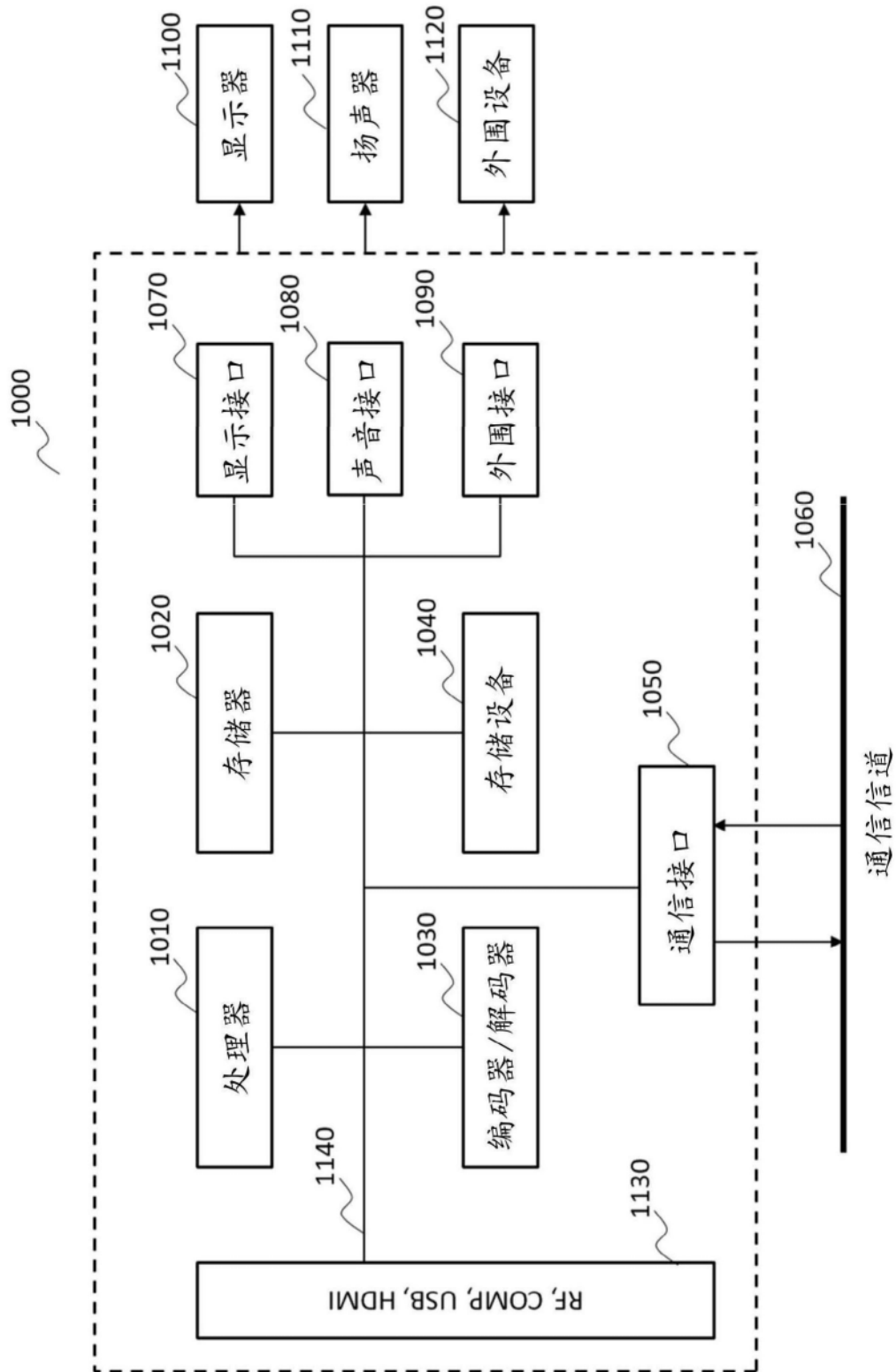


图3

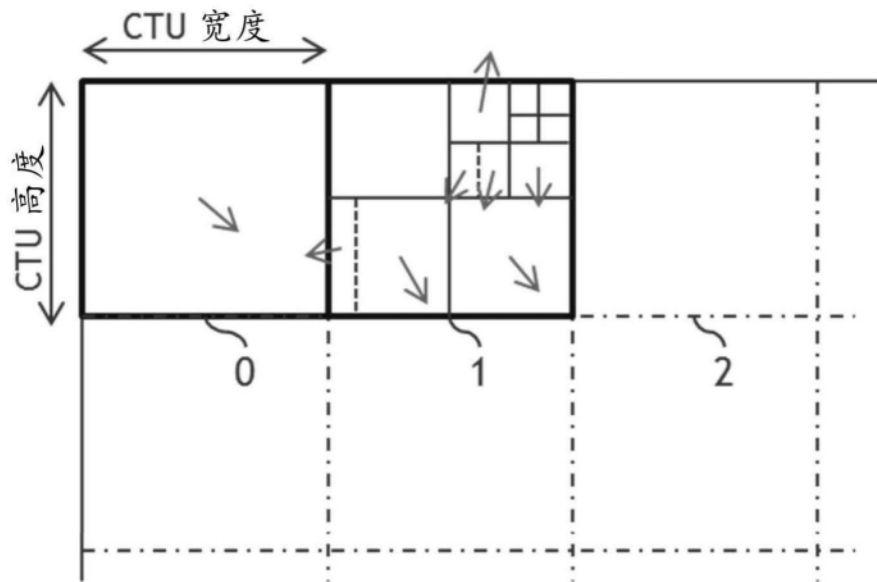


图4A

条带

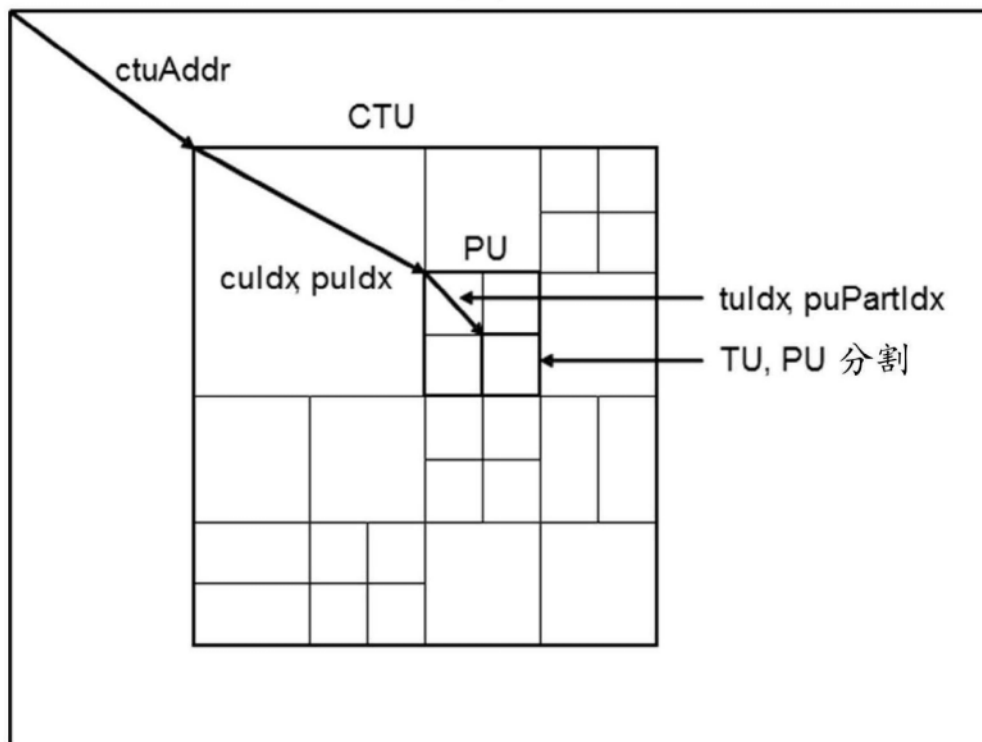


图4B

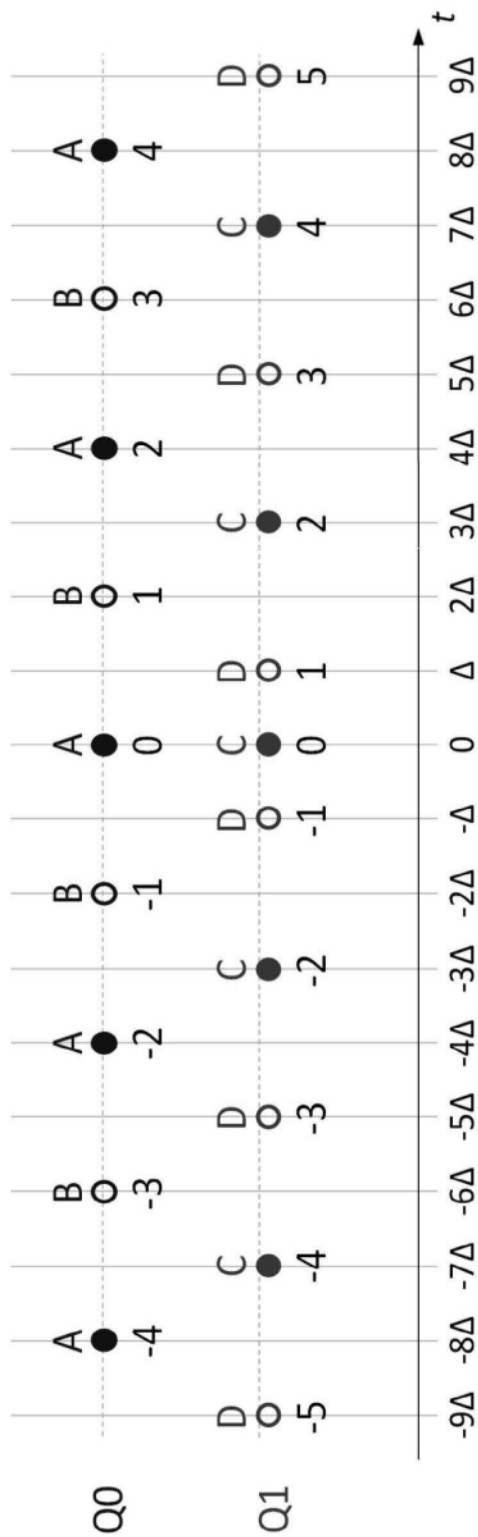


图5

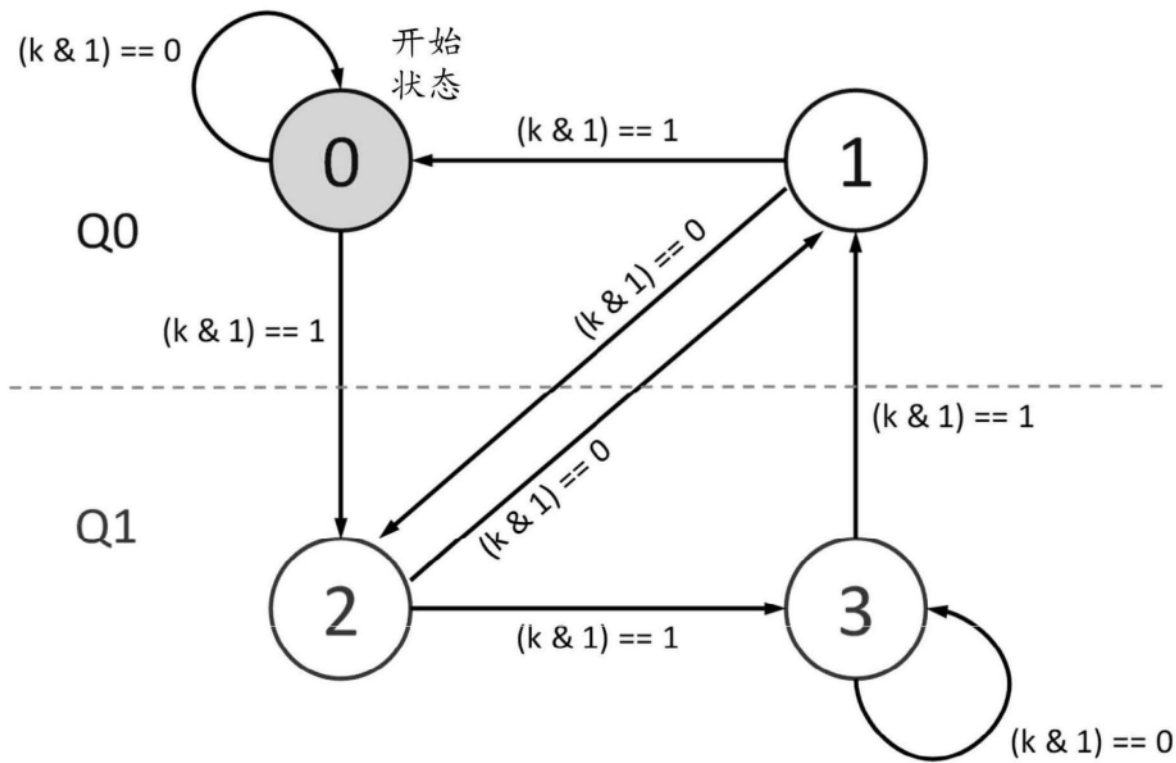


图6A

当前状态	下一状态...	
	$(k \& 1) == 0$	$(k \& 1) == 1$
0	0	2
1	2	0
2	1	3
3	3	1

图6B

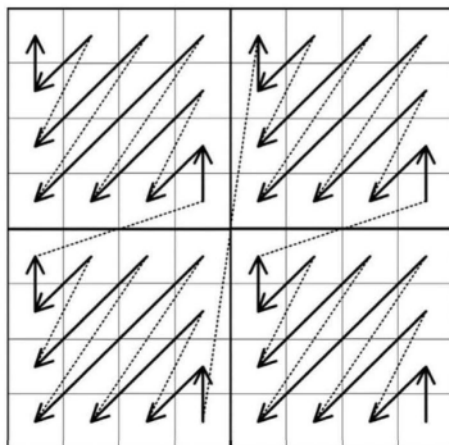


图6C

```

residual_tb( ... )
{
    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    for( i=lastSubBlock ; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( ... )
            }
        }
    }
}

```

图7A

```
residual_subblock ( ... )
{
    numRegBins = is2x2CG ? 8 : 32

    for( k = firstPos ... ; remRegBins >= 4 ... ; k-- ) { // 第一遍次
        sig_flag[ k ] // ctx取决于prev. par_flag
        remRegBins -- // 递减remRegBins
        if( sig_flag[ k ] ) {
            gt1_flag[ k ]
            if ( gt1_flag[ k ] ) {
                par_flag[ k ] // 基于上下文的编解码 (常规)
                remRegBins -- // 递减remRegBins
                gt3_flag[ k ] // 基于上下文的编解码 (常规)
                remRegBins -- // 递减remRegBins
            }
            coeff[k] = 1 + parFlag + gt1Flag + (gt3Flag << 1)
        }
        firstPosMode2 = k
        for( k=firstPos; k >firstPosMode2; k-- ) { // 第二遍次 (EP二进制位)
            if( coeff[k] >= 4 )
                remainder[ k ]
        }

        for ( k = firstPosMode2 ; k > lastPos ; k-- ) { // 系数旁路 (EP二进制位)
            abs_level[ k ]
        }
        sign_data
    }
}
```

图7B

```
coding_unit( ... )
{
    cu_ckpt_flag
    If ( cu_skip_flag == 0 )
        cu_pred_mode_flag

    if( cu_skip_flag == 0 && cuPredMode != MODE_INTRA ){
        pred_mode_ibc_flag
    }

    If( cuPredMode == IMODE_NTRA ){
        intra_prediction_data
    }
    else { // MODE_INTER或者MODE_IBC
        inter_prediction_data
    }

    If (!pcm_flag){
        if( cuPredMode != MODE_INTRA && merge_flag==0 )
            cu_cbf
            if ( cu_cbf ) {
                // SBT模式相关的信令通知

                transform_tree( ... )
            }
    }
}
```

图8A

```
Transform_tree( ... )
{
    If ( IntraSubPartSize == NO_ISP_FLAG ){
        If ( IntraSubPartSize == NO_ISP_FLAG )
            if (width > MaxTbSizeX || height > MaxTbSizeY )
                四叉树划分成4个子变换树
        }
        else {
            transform_unit( ... )
        }
    }
    Else if ( cu_sbt_fag ) {
        根据SBT语法划分成2个变换单元
        transform_unit( ... )
        transform_unit( ... )
    }
    else {
        根据ISP语法划分成2个变换单元
        transform_unit( ... )
        transform_unit( ... )
    }
}
```

图8B

```
transform_unit( ... )
{
    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    If(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    If( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    If( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        Residual_tb ( ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        Residual_tb ( ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        Residual_tb ( ... , compId=Cr,..)
    }
}
```

图8C

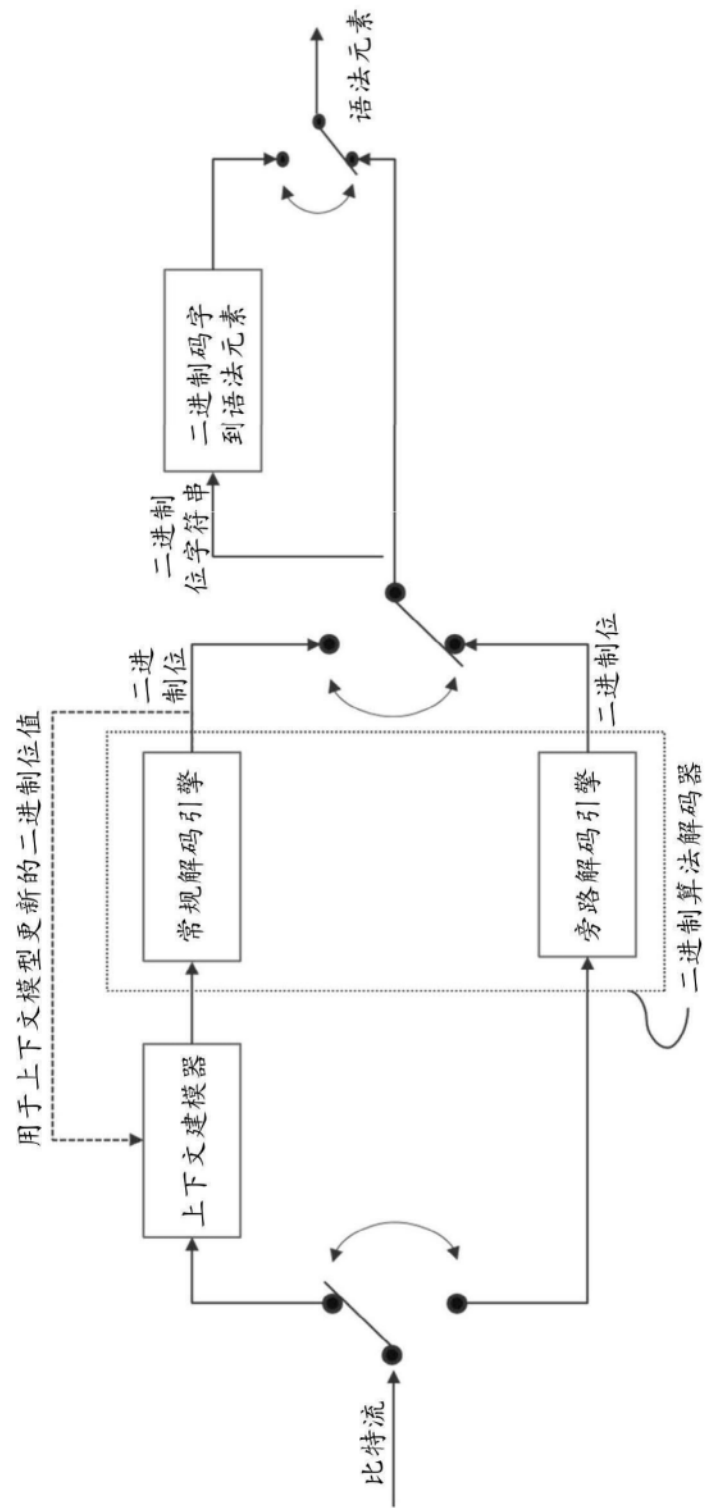


图9A

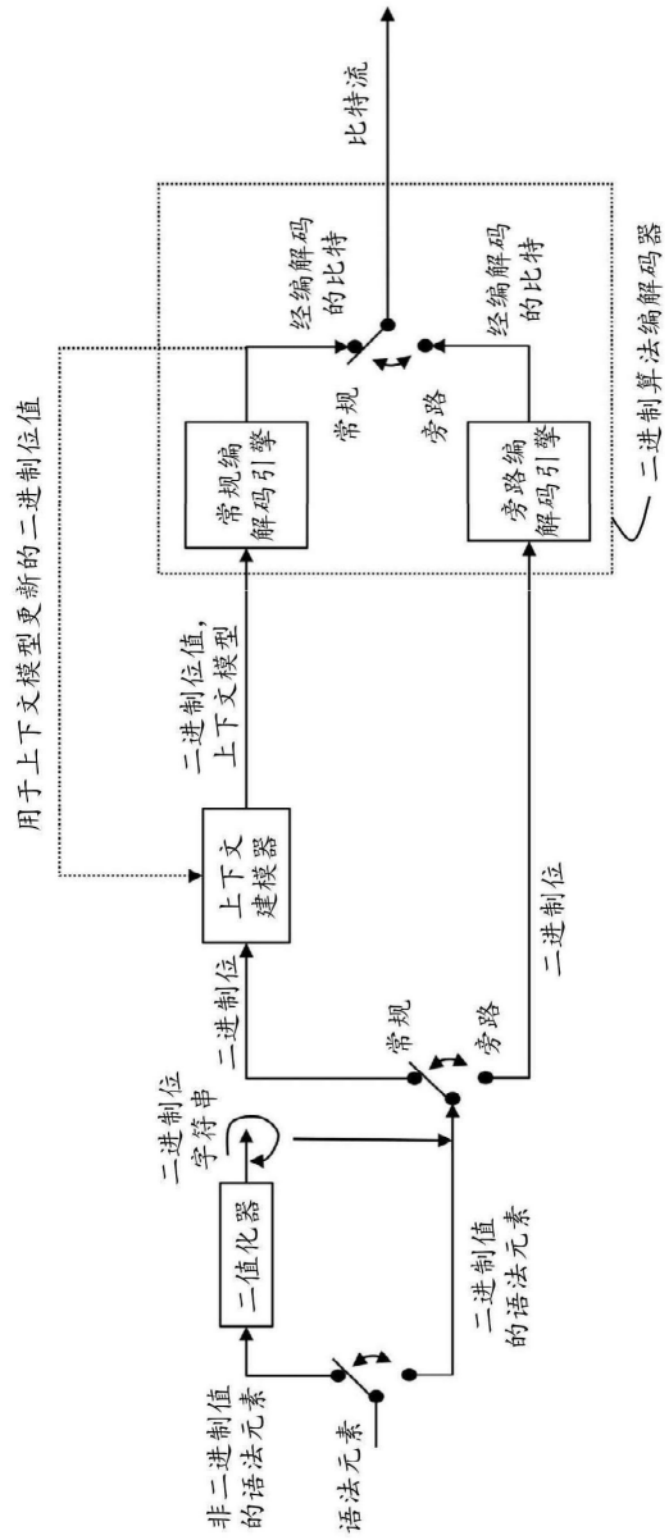


图9B

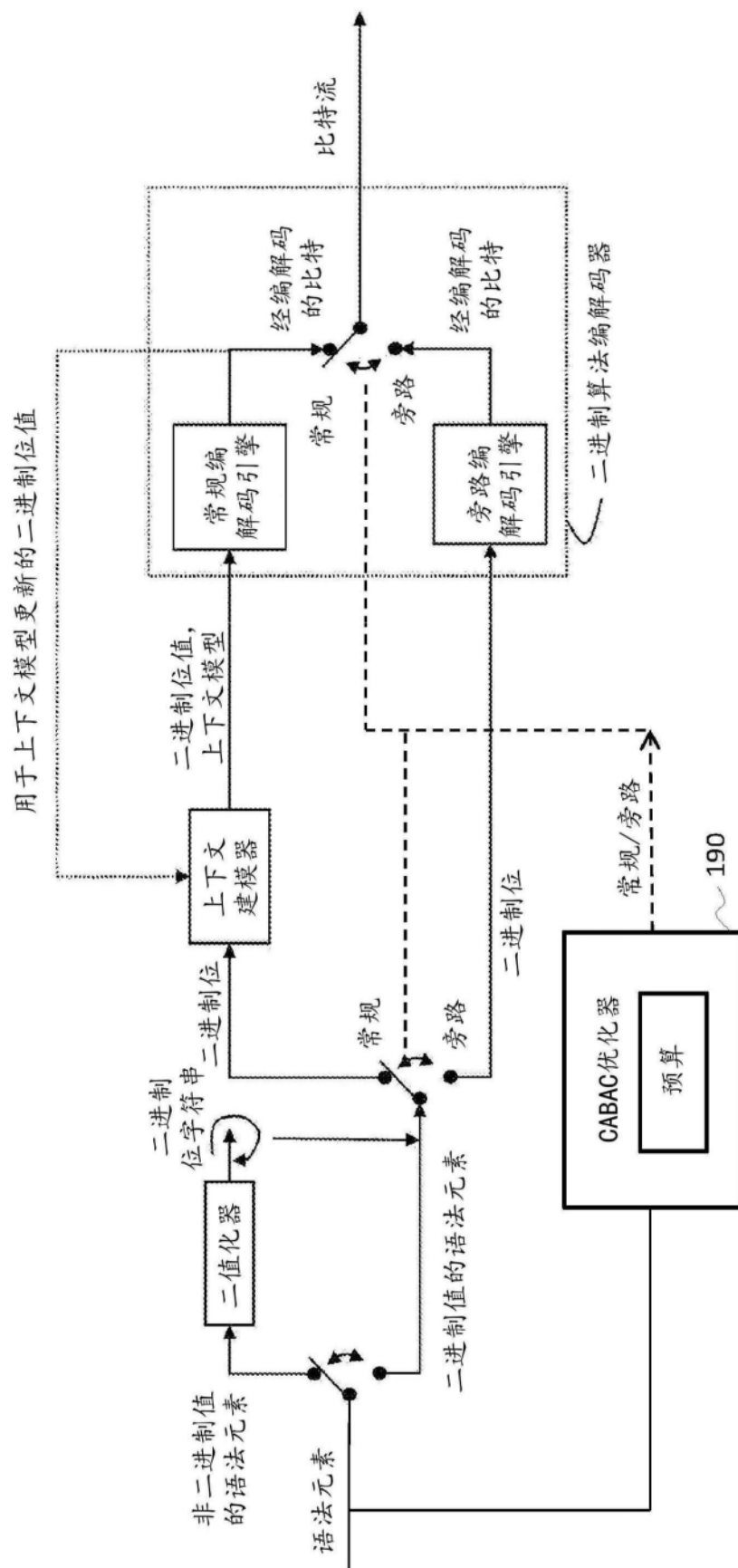


图10

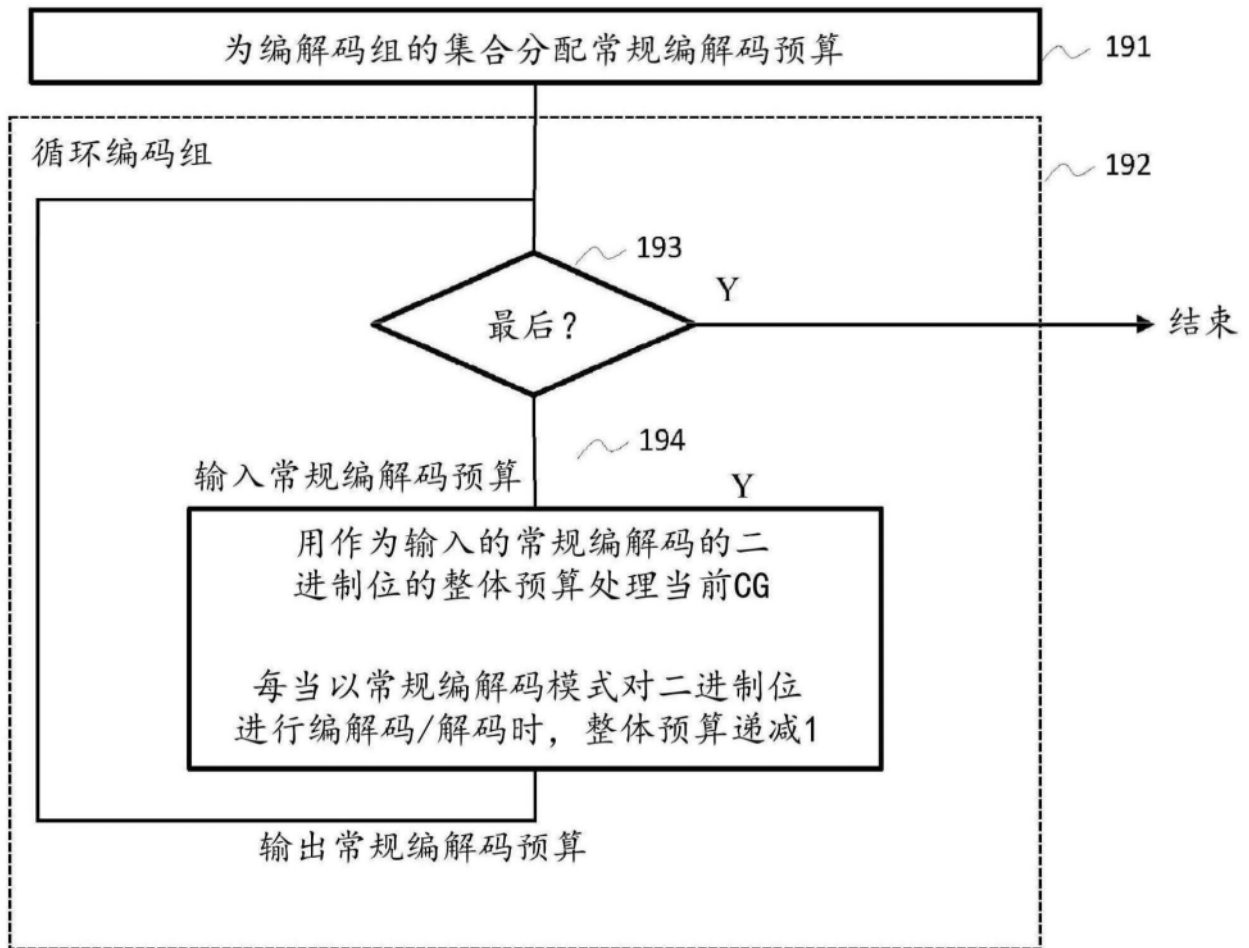


图11

```
residual_subblock ( numRegBins_in_out )
{
    remRegBins = numRegBins_in_out

    for( k = firstPos ... ; remRegBins >= 4 ... ; k-- ) { // 第一遍次
        sig_flag[ k ]      // ctx取决于prev. par_flag
        remRegBins -- // 递减remRegBins
        if( sig_flag[ k ] ) {
            gt1_flag[ k ]
            if ( gt1_flag[k] ) {
                par_flag[ k ] // 基于上下文的编解码 (常规)
                remRegBins -- // 递减remRegBins
                gt3_flag[ k ] // 基于上下文的编解码 (常规)
                remRegBins -- // 递减remRegBins
            }
            coeff[k] = 1 + parFlag + gt1Flag + (gt3Flag << 1)
        }
        firstPosMode2 = k
        for( k=firstPos; k >firstPosMode2; k-- ) { // 第二遍次 (EP二进制位)
            if( coeff[k] >= 4 )
                remainder[ k ]
        }

        for ( k = firstPosMode2 ; k > lastPos ; k-- ) { // 系数旁路 (EP二进制位)
            abs_level[ k ]
        }
        sign_data
        numRegBins_in_out = remRegBins
    }
}
```

图12

```
residual_tb( ... )
{
    unitary_budget = 2;
    TB_level_remRegBins = unitary_budget * width*height

    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    for( i=lastSubBlock; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( TB_level_remRegBins )
            }
        }
    }
}
```

图13A

```
residual_tb( ... )
{
    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    unitary_budget = 2;
    TB_level_remRegBins = unitary_budget * last_pos_x * last_pos_y

    for( i=lastSubBlock; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( TB_level_remRegBins )
            }
        }
    }
}
```

图13B

```
transform_unit( ... )
{
    unitary_budget = 2; // 平均常规二进制位比率是2个二进制位/样点
    TU_level_remRegBins = unitary_budget * width*height *3 / 2

    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    if(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    if( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    if( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        Residual_tb (TU_level_remRegBins, ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        Residual_tb ( TU_level_remRegBins, ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        Residual_tb ( TU_level_remRegBins, ... , compId=Cr,..)
    }
}
```

图14A

```
transform_unit( ... )
{
    unitary_budget = 2; // 平均常规二进制位比率是2个二进制位/样点
    TU_level_RegBins = unitary_budget * width*height *3 / 2

    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    If(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    If( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    If( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        bins_Y = TU_level_RegBins * 2 /3
        Residual_tb (bins_Y, ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        bins_Cb = (TU_level_RegBins - binsY ) /2
        Residual_tb ( bins_Cb, ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        bins_Cr = TU_level_RegBins - binsY - binCb)
        Residual_tb ( bins_Cr, ... , compId=Cr,..)
    }
}
```

图14B

```
residual_tb ( numRegBins_in_out )
{
    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    for( i= lastSubBlock ; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( numRegBins_in_out )
            }
        }
    }
}
```

图14C

```
coding_unit( ... )
{
    unitary_budget = 2; // 平均常规二进制位比率是2个二进制位/样点
    CU_level_remRegBins = unitary_budget * width * height * 3 / 2

    cu_skip_flag
    If ( cu_skip_flag == 0 )
        cu_pred_mode_flag

    if( cu_skip_flag == 0 && cuPredMode != MODE_INTRA ){
        pred_mode_ibc_flag
    }

    If( cuPredMode == IMODE_NTRA ){
        intra_prediction_data
    }
    else { // MODE_INTER或者MODE_IBC
        inter_prediction_data
    }

    If (!pcm_flag){
        if( cuPredMode != MODE_INTRA && merge_flag==0 )
            cu_cbf
            if ( cu_cbf ) {
                // SBT模式相关的信令通知

                transform_tree(CU_level_remRegBins , ... )
            }
    }
}
```

图15A

```
Transform_tree(CU_level_remRegBins , ... )  
{  
    if ( IntraSubPartSize == NO_ISP_FLAG ){  
        if ( IntraSubPartSize == NO_ISP_FLAG )  
            if (width > MaxTbSizeX || height > MaxTbSizeY )  
                四叉树划分成4个子变换树  
            }  
        else {  
            transform_unit(CU_level_remRegBins , ... )  
        }  
    }  
    Else if ( cu_sbt_fag ) {  
        根据SBT语法划分成2个变换单元  
        transform_unit(CU_level_remRegBins , ... )  
        transform_unit(CU_level_remRegBins , ... )  
    }  
    else {  
        根据ISP语法划分成2个变换单元  
        transform_unit(CU_level_remRegBins , ... )  
        transform_unit(CU_level_remRegBins , ... )  
    }  
}
```

图15B

```
Transform_tree(CU_level_remRegBins , ... )  
{  
    If ( IntraSubPartSize == NO_ISP_FLAG ){  
        If ( IntraSubPartSize == NO_ISP_FLAG )  
            if (width > MaxTbSizeX || height > MaxTbSizeY )  
                四叉树划分成4个子变换树  
        }  
        else {  
            transform_unit(CU_level_remRegBins , ... )  
        }  
    }  
    Else if ( cu_sbt_fag ) {  
        根据SBT语法划分成2个变换单元  
        transform_unit(sbt_coded ? CU_level_remRegBins : 0, ... )  
        transform_unit(sbt_coded ? CU_level_remRegBins : 0, ... )  
    }  
    else {  
        根据ISP语法划分成2个变换单元  
        regBins = CU_level_remRegBins/2  
        transform_unit(regBins, ... )  
        transform_unit( (CU_level_remRegBins – regBins) , ... )  
    }  
}
```

图15C

```
transform_unit(TU_regBins , ... )
{
    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    If(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    If( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    If( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        Residual_tb (TU_regBins , ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        Residual_tb (TU_regBins , ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        Residual_tb (TU_regBins , ... , compId=Cr,..)
    }
}
```

图15D

```
transform_unit(TU_RegBins , ... )
{
    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    If(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    If( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    If( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        bins_Y = TU_RegBins * 2 /3
        Residual_tb (bins_Y, ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        bins_Cb = (TU_RegBins - binsY ) /2
        Residual_tb ( bins_Cb, ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        bins_Cr = TU_RegBins - binsY - binCb)
        Residual_tb ( bins_Cr, ... , compId=Cr,..)
    }
}
```

图15E

```
residual_subblock_all_bypass ()  
{  
    remRegBins = numRegBins_in_out  
  
    K=firstPos  
    firstPosMode2 = k  
  
    for ( k = firstPosMode2 ; k > lastPos ; k-- ) {  
        abs_level[ k ]  
    }  
    sign_data  
    numRegBins_in_out = remRegBins  
}
```

图16A

```
residual_subblock_all_regular ()
{
    for( k = firstPos ... ; remRegBins >= 4 ... ; k--) { // 第一遍次
        sig_flag[k] // ctx取决于prev. par_flag
        if( sig_flag[k] ) {
            gt1_flag[k]
            if ( gt1_flag[k] ) {
                par_flag[k] // 基于上下文的编解码 (常规)
                gt3_flag[k] // 基于上下文的编解码 (常规)
            }
            coeff[k] = 1 + parFlag + gt1Flag + (gt3Flag << 1)
        }
        firstPosMode2 = k
    }
    for( k=firstPos; k >lastPos; k-- ) { // 第二遍次 (EP二进制位)
        if( coeff[k] >= 4 )
            remainder[ k ]
    }

    sign_data
    numRegBins_in_out = remRegBins
}
```

图16B