(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 120075441 A
(43)申请公布日 2025.05.30

(21)申请号 202510148620.8

(22)申请日 2020.06.16

(30)优先权数据
19305799.9 2019.06.20 EP

(62)分案原申请数据
202080056767.X 2020.06.16

(71)申请人 交互数字CE专利控股公司
地址 法国巴黎

(72)发明人 T·波伊里尔 F·莱林内克
K·纳瑟 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 19/12(2014.01)
H04N 19/154(2014.01)
H04N 19/176(2014.01)
H04N 19/70(2014.01)

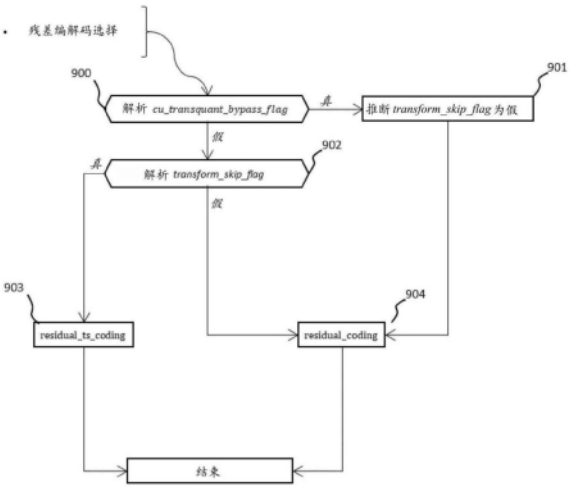
权利要求书2页 说明书30页 附图18页

(54)发明名称

用于通用视频编解码的无损模式

(57)摘要

在包括多个编解码工具的视频编解码系统中提出了一种无损编解码模式,其中一些编解码工具通过设计是有损的,一些编解码工具可以适于成为无损或近无损的。为了在这种系统中启用无损模式,提出禁用通过设计有损的工具并且仅使用无损工具,以适配一些工具以启用无损编解码,并且适配一些工具以启用近无损编解码,使得可以在残差编解码之后应用二次无损编解码。在特定实施例中,提出了通过以下方式来确定残差编解码的类型:当信息指示使用变换跳过残差编解码时,获得表示特殊模式的标志,并且当该标志为真时,确定必须使用常规残差编解码,而不是应该使用的变换跳过残差编解码。



CN 120075441 A

1.一种方法,包括:

获得指示图像或视频的像素的块是变换跳过块的第一标志,其中变换跳过残差编解码应当被用于对与所述图像或视频的所述像素的块相关联的变换块进行编解码;

获得指示残差编解码被用于变换跳过块的第二标志;

使用残差编解码对所述变换块的系数进行解码;以及

使用所解码的系数对所述块进行解码,

其中所述第二标志是从高于所述变换块的级别获得的。

2.一种用于对图像或视频的像素的块的系数进行解码的装置,包括处理器,所述处理器被配置为:

获得指示图像或视频的像素的块是变换跳过块的第一标志,其中变换跳过残差编解码应当被用于对与所述图像或视频的所述像素的块相关联的变换块进行编解码;

获得指示残差编解码被用于变换跳过块的第二标志;

使用残差编解码对所述变换块的系数进行解码;以及

使用所解码的系数对所述块进行解码,

其中所述第二标志是从高于所述变换块的级别获得的。

3.一种方法,包括:

基于速率失真优化,对于图像或视频的像素的块,确定变换跳过和残差编解码应当被使用;

使用常规残差编解码对所述块的系数进行编码;以及

对指示图像或视频的像素的块是变换跳过块的第一标志和指示残差编解码被用于变换跳过块的第二标志进行编码,其中所述第二标志在比变换块更高的级别被编码。

4.一种装置,包括

速率失真优化模块,被配置为确定变换跳过和残差编解码应当被使用;以及

处理器,被配置为在所述速率失真优化模块确定变换跳过和残差编解码应当被使用的情况下:

使用常规残差编解码对所述块的系数进行编码;以及

对指示图像或视频的像素的块是变换跳过块的第一标志和指示残差编解码被用于变换跳过块的第二标志进行编码,其中所述第二标志在比变换块更高的级别被编码。

5.非暂时性计算机可读介质,包括程序代码指令,所述程序代码指令在由处理器执行时用于实现根据权利要求1所述的方法的步骤。

6.非暂时性计算机可读介质,包括程序代码指令,所述程序代码指令在由处理器执行时用于实现根据权利要求3所述的方法的步骤。

7.一种用于确定图像或视频的块的残差编解码的类型的方法,所述方法包括:

获得指示变换跳跃被使用的第一信息;

获得指示常规残差编解码被用于变换跳过块的第二信息;以及

为所述块选择常规残差编解码而不是变换跳过残差编解码。

8.一种用于对图像或视频的像素的块的系数进行编码的方法,其中残差编解码的类型是通过根据权利要求7所述的方法来确定的。

9.一种用于对图像或视频的像素的块的系数进行解码的方法,其中残差编解码的类型

是通过根据权利要求7所述的方法来确定的。

10.一种用于对图像或视频的像素的块的系数进行编码的装置,其中残差编解码的类型是通过根据权利要求7所述的方法来确定的。

11.一种用于对图像或视频的像素的块的系数进行解码的装置,其中残差编解码的类型是通过根据权利要求7所述的方法来确定的。

用于通用视频编解码的无损模式

[0001] 本分案申请是申请日为2020年06月16日、申请号为202080056767.X、发明名称为“用于通用视频编解码的无损模式”的分案申请。

技术领域

[0002] 本公开属于视频压缩领域,并且至少一个实施例更具体地涉及用于通用视频编解码(Versatile Video Coding,VVC)的无损模式。

背景技术

[0003] 为了实现高压缩效率,图像和视频编解码方案通常采用预测和变换以利用(leverage)视频内容中的空间和时间冗余。通常,使用帧内或帧间预测来利用帧内或帧间相关性,然后原始图像块和预测图像块之间的差(通常表示为预测误差或预测残差)被变换、量化和熵编解码。在编码期间,原始图像块通常被分割/划分为子块(例如可能使用四叉树分割)。为了重构视频,通过与预测、变换、量化和熵编解码对应的逆过程来解码经压缩的数据。

发明内容

[0004] 在包括多个编解码工具的视频编解码系统中提出了无损编解码模式,其中一些编解码工具通过设计是有损的,一些编解码工具可以适于成为无损或近无损的。为了在这种视频编解码系统中启用无损模式,提出禁用通过设计有损的工具并且仅使用无损工具,以适配一些工具以启用无损编解码,并且适配一些工具以启用近无损编解码,使得可以在残差编解码之后应用二次无损编解码,从而提供无损编解码。

[0005] 在特定实施例中,用于确定残差编解码类型的方法包括,在信息指示使用变换跳过残差编解码(transform skip residual coding)的情况下,获得表示特殊模式的标志,并且当表示特殊模式的标志为真时,确定必须使用常规残差编解码而不是应该使用的变换跳过残差编解码。

[0006] 根据第一方面,一种用于确定残差编解码类型的方法包括,在信息指示使用变换跳过残差编解码的情况下,获得表示特殊模式的标志,并且当表示特殊模式的标志为真时,确定必须使用常规残差编解码而不是应该使用的变换跳过残差编解码。

[0007] 根据第二方面,一种视频编码方法包括,对于视频块,根据第一方面的方法确定残差编解码的类型。

[0008] 根据第三方面,一种视频解码方法包括,对于视频块,根据第一方面的方法确定残差编解码的类型。

[0009] 根据第四方面,一种视频编码装置包括,编码器,该编码器被配置为根据第一方面的方法来确定残差编解码的类型。

[0010] 根据第五方面,一种视频解码装置包括,解码器,该解码器被配置为根据第一方面的方法来确定残差编解码的类型。

[0011] 本实施例的一个或多个实施例还提供了一种非暂时性计算机可读存储介质,其上存储有用于根据上述任何方法中的至少一部分对视频数据进行编码或解码的指令。一个或多个实施例还提供了一种计算机程序产品,包括用于执行上述任何方法中的至少一部分的指令。

附图说明

[0012] 图1A示出了根据一实施例的视频编码器的框图。

[0013] 图1B示出了根据一实施例的视频解码器的框图。

[0014] 图2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。

[0015] 图3示出了用于PCM、无损和变换跳过模式的解码处理的简化框图的示例。

[0016] 图4示出了每个SBT位置的水平和竖直变换。

[0017] 图5从解码器的角度示出了LMCS架构。

[0018] 图6示出了二次变换的应用。

[0019] 图7示出了简化二次变换(RST)。

[0020] 图8示出了正向和反向简化变换。

[0021] 图9示出了利用16x48矩阵的正向RST8x8处理的示例。

[0022] 图10A和10B示出了根据至少一个实施例的示例流程图。

[0023] 图11示出了用于亮度整形(reshaping)的正向然后反向函数的应用。

[0024] 图12示出了包括对cu_transquant_bypass_flag、transform_skip_flag的解析以及当变换跳过对于无损编解码块被推断为真时的残差编解码的实施例的示例流程图。

[0025] 图13示出了包括对cu_transquant_bypass_flag、transform_skip_flag的解析以及当变换跳过对于无损编解码块被推断为假时的残差编解码的实施例的示例流程图。

[0026] 图14示出了包括对cu_transquant_bypass_flag、transform_skip_flag的解析以及当对于无损编解码块总是解析变换跳过时的残差编解码的实施例的示例流程图。

[0027] 图15示出了二次无损编解码的简化的框图的示例。

[0028] 图16示出了在使用区域级信令时对region_transquant_bypass_flag和split_cu_flag的解析处理的示例流程图。

[0029] 图17示出了在双树情况下对亮度和色度的不同分割。

具体实施方式

[0030] 各种实施例涉及一种用于图像块的采样的预测值的后处理方法,该值是根据帧内预测角度预测的,其中采样的值在该预测之后被修改,使得其基于左参考采样的值和所获得的采样的预测值之间的差的加权来确定,其中左参考采样是基于帧内预测角度确定的。提出了基于该后处理方法的编码方法、解码方法、编码装置、解码装置。

[0031] 此外,尽管描述了与VVC(通用视频编解码)或HEVC(高效视频编解码)规范的特定草案相关的原理,但是本方面不限于VVC或HEVC,并且可以应用于例如其他标准和推荐,无论是现有的还是未来开发的,以及任何这种标准和推荐(包括VVC和HEVC)的扩展。除非另有说明,或者技术上被排除,否则本申请中描述的各方面可以单独使用或者组合使用。

[0032] 图1A示出了视频编码器100。预期了该编码器100的变型,但是为了清楚起见,下面

描述了编码器100而未描述所有预期的变型。在经编码之前,视频序列可以经过预编码处理(101),例如,对输入颜色画面应用颜色变换(例如,从RGB 4:4:4转换为YCbCr 4:2:0),或者执行输入画面分量的重新映射,以便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)。可将元数据与预处理相关联,并将元数据附接到比特流。

[0033] 在编码器100中,由编码器元件对画面进行编码,如下所述。例如以CU为单位对要编码的画面进行分割(102)和处理。使用例如帧内或者帧间模式对每个单元进行编码。当以帧内模式对单元进行编码时,其执行帧内预测(160)。在帧间模式中,执行运动估计(175)和补偿(170)。编码器判断(105)使用帧内模式或帧间模式中的哪一个用于对单元进行编码,并通过例如预测模式标志来指示该帧内/帧间判断。例如,通过从原始图像块中减去(110)预测块,来计算预测残差。

[0034] 然后对预测残差进行变换(125)和量化(130)。对经量化的变换系数、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。编码器可以跳过变换,并对未变换的残差信号直接应用量化。编码器可以绕过变换和量化,即,直接对残差进行编解码,而不应用变换或量化处理。

[0035] 编码器对已编码的块进行解码,以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)并进行逆变换(150)以解码预测残差。在组合(155)已解码的预测残差和已预测块的情况下,重构图像块。环内滤波器(165)被应用于重构的画面,以执行例如去块/SAO(采样自适应偏移)、自适应环路滤波器(ALF)滤波以减少编码伪影。滤波后的图像存储在参考画面缓冲器(180)中。

[0036] 图1B示出了视频解码器200的框图。在解码器200中,由解码器元件对比特流进行解码,如下所述。视频解码器200通常执行与图1A中描述的编码遍历相反(reciprocal)的解码遍历。编码器100通常还执行视频解码作为编码视频数据的一部分。特别地,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。首先对该比特流进行熵解码(230)以获得变换系数、运动矢量和其他编解码信息。画面分割信息指示如何对画面进行分割。因此,解码器可以根据经解码的画面分割信息来对画面进行划分(235)。对变换系数进行去量化(240)和逆变换(250)以解码预测残差。通过组合(255)经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)中获得(270)预测块。对经重构的图像应用环内滤波器(265)。将经滤波的图像存储在参考画面缓冲器(280)中。

[0037] 经解码的画面可以进一步经过后解码处理(285),例如逆颜色变换(例如,从YCbCr 4:2:0转换为RGB 4:4:4)、或执行在预编码处理(101)中所执行的重新映射处理的逆的逆重新映射。后解码处理可以使用从预编码处理中导出并在比特流中用信号通知的元数据。

[0038] 图2示出了其中实现了各个方面和实施例的系统的示例的框图。系统1000可以被实施为包括以下描述的各种组件的设备,并且被配置为执行本文档中描述的各方面中的一个或多个。这种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、数字电视接收机、个人视频记录系统、连接的家用电器、以及服务器。系统1000的元件可以单独或组合地实施在单个集成电路(IC)、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/解码器元件被分布在多个IC和/或分立组件上。在各种实施例中,系统1000经由例如通信总线或通过专用输入和/或输出端口来通信地耦合到一个或多个其他系统或其他电子设备。在各种实施

例中,系统1000被配置为实现本文档中描述的各方面中的一个或多个。

[0039] 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用于实现例如本文档中描述的各个方面的指令。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包括至少一个存储器1020(例如,易失性存储器件和/或非易失性存储器件)。系统1000包括存储设备1040,该存储设备1040可以包括非易失性存储器和/或易失性存储器,包括但不限于电可擦除可编程只读存储器(EEPROM)、只读存储器(ROM)、可编程只读存储器(PROM)、随机存取存储器(RAM)、动态随机存取存储器(DRAM)、静态随机存取存储器(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、附接存储设备(包括可拆卸和不可拆卸存储设备)和/或网络可访问存储设备。

[0040] 系统1000包括编码器/解码器模块1030,其被配置为例如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表示可以被包括在设备中以执行编码和/或解码功能的(多个)模块。如已知的,设备可以包括编码和解码模块中的一个或两者。此外,编码器/解码器模块1030可以被实现为系统1000的分离元件,或者可以作为本领域技术人员已知的硬件和软件的组合被合并在处理器1010内。

[0041] 要加载到处理器1010或编码器/解码器1030上以执行本文档中描述的各个方面的程序代码可以被存储在存储设备1040中,并随后加载到存储器1020上以供处理器1010运行。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档中描述的处理期间存储各种项目中的一个或多个。这种存储的项目可以包括但不限于输入视频、解码的视频或解码的视频中的部分、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。

[0042] 在一些实施例中,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,可以使用处理设备(例如,处理设备可以是处理器1010或编码器/解码器模块1030)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。在几个实施例中,使用外部非易失性闪存来存储例如电视的操作系统。在至少一个实施例中,诸如RAM的快速外部动态易失性存储器被用作用于视频编解码和解码操作的工作存储器,诸如用于MPEG-2(MPEG是指动态画面专家组,MPEG-2也被称为ISO/IEC 13818,而13818-1也被称为H.222,并且13818-2也被称为H.262)、HEVC(HEVC是指高效视频编解码,也被称为H.265和MPEG-H第2部分)、或VVC(通用视频编解码,联合视频专家小组JVET正在开发的新标准)。

[0043] 如块1130中所示,可以通过各种输入设备提供到系统1000的元件的输入。这种输入设备包括但不限于:(i)接收例如由广播者通过空中发送的射频(RF)信号的RF部分,(ii)组件(COMP)输入端(或COMP输入端集合),(iii)通用串行总线(USB)输入端,和/或(iv)高清晰度多媒体接口(HDMI)输入端。图2中未示出的其他示例包括复合视频。

[0044] 在各种实施例中,块1130的输入设备具有相关联的相应的本领域已知的输入处理元件。例如,RF部分可以与适用于以下操作的元件相关联:(i)选择期望的频率(也称为选择信号,或者将信号频带限制到频带),(ii)下变换所选择的信号,(iii)再次频带限制到较窄

的频带,以选择(例如)信号频带(在某些实施例中可以称为信道),(iv)解调经下变换和频带限制的信号,(v)执行纠错,以及(vi)解复用以选择期望的数据分组的流。各种实施例的RF部分包括执行这些功能的一个或多个元件,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、下变换器、解调器、误差校正器、和解复用器。RF部分可以包括执行这些功能中的各种功能的调谐器,这些功能包括例如将接收的信号下变换至较低频率(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、下变换和再次滤波至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/或添加执行类似或不同的功能的其他元件。添加元件可以包括在现有元件之间插入元件,诸如,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。

[0045] 此外,USB和/或HDMI终端可以包括相应的接口处理器,以用于通过USB和/或HDMI连接将系统1000连接至其他电子设备。应当理解,可以根据需要在例如单独的输入处理IC内或处理器1010内实现输入处理的各个方面,例如里德-所罗门纠错。类似地,可以根据需要在单独的接口IC内或处理器1010内实现USB或HDMI接口处理的各方面。将经解调、纠错和解复用的流提供至各种处理元件,包括例如处理器1010、和编码器/解码器1030,其与存储器和存储元件组合操作以根据需要处理数据流以用于在输出设备上呈现。

[0046] 可以将系统1000的各种元件提供在集成外壳内。在集成外壳内,各种元件可以使用合适的连接布置1140(例如,本领域已知的内部总线,包括IC间(I2C)总线、布线、和印刷电路板)互连并在其间发送数据。

[0047] 系统1000包括实现经由通信信道1060与其他设备通信的通信接口1050。通信接口1050可以包括但不限于被配置为通过通信信道1060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可以在例如有线和/或无线介质内实现。

[0048] 在各种实施例中,使用诸如Wi-Fi网络的无线网络(例如IEEE 802.11(IEEE指电气和电子工程师协会)),将数据流式传输或以其他方式提供至系统1000。这些实施例的Wi-Fi信号是通过适于Wi-Fi通信的通信信道1060和通信接口1050来接收的。这些实施例的通信信道1060通常被连接至提供对包括互联网的外部网络的接入的接入点或路由器,以用于允许流式传输的应用和其他过顶通信(over-the-top communication)。其他实施例使用机顶盒向系统1000提供流式传输的数据,机顶盒通过输入块1130的HDMI连接传送数据。还有其他实施例使用输入块1130的RF连接向系统1000提供流式传输的数据。如上所述,各种实施例以非流式传输的方式提供数据。此外,各种实施例使用除Wi-Fi外的无线网络,例如蜂窝网络或蓝牙网络。

[0049] 系统1000可以向各种输出设备提供输出信号,输出设备包括显示器1100、扬声器1110和其他外围设备1120。各种实施例的显示器1100包括例如触摸屏显示器、有机发光二极管(OLED)显示器、弯曲显示器、和/或可折叠显示器中的一个或多个。显示器1100可以用于电视机、平板计算机、膝上型计算机、蜂窝电话(移动电话)、或其他设备。显示器1100也可以与其他组件集成(例如,如在智能电话中),或者分离(例如,用于膝上型计算机的外部监视器)。在实施例的各种示例中,其他外围设备1120包括独立运行的数字视频盘(或数字多功能盘)(对于两个术语均为DVR)、盘播放器、立体声系统、和/或照明系统中的一个或多个。

各种实施例使用基于系统1000的输出提供功能的一个或多个外围设备1120。例如,盘播放器执行播放系统1000的输出的功能。

[0050] 在各种实施例中,使用诸如AV.Link、消费电子控制(CEC)、或在有或没有用户干预的情况下实现设备到设备的控制的其他通信协议的信令来在系统1000和显示器1100、扬声器1110、或其他外围设备1120之间通信控制信号。输出设备可以通过相应的接口1070、1080和1090经由专用的连接通信地耦合至系统1000。可替代地,输出设备可以使用通信信道1060经由通信接口1050连接至系统1000。在电子设备(诸如,例如,电视机)中,显示器1100和扬声器1110可以与系统1000的其他组件集成在单一单元中。在各种实施例中,显示接口1070包括显示驱动器,诸如,例如,时序控制器(T Con)芯片。

[0051] 显示器1100和扬声器1110可以可替代地与其他组件中的一个或多个分离,例如如果输入1130的RF部分是单独的机顶盒的部分。在显示器1100和扬声器1110是外部组件的各种实施例中,可以经由包括例如HDMI端口、USB端口、或COMP输出的专用输出连接来提供输出信号。

[0052] 这些实施例可以通过由处理器1010实现的计算机软件、或通过硬件、或通过硬件和软件的组合来实现。作为非限制性示例,实施例可以由一个或多个集成电路来实现。作为非限制性示例,存储器1020可以是适合于技术环境的任何类型,并且可以使用任何适当的数据存储技术来实现,诸如光存储设备、磁存储设备、基于半导体的存储设备、固定存储器、和可移除存储器。作为非限制性示例,处理器1010可以是适合于技术环境的任何类型,并且可以包括微处理器、通用计算机、专用计算机和基于多核架构的处理器中的一个或多个。

[0053] 无损模式在HEVC是可用的。在这种模式下,变换和量化旁路(bypass)由CU语法结构的开始处的标志在CU级别处指示。如果旁路开启,则变换和量化器缩放操作被跳过,并且残差信号被直接编解码而没有任何降级。因此,这种模式实现了对编解码块的无损表示的完美重构。采样差如同其是量化的变换系数级别那样被编码,即,利用变换子块、系数扫描和最后有效系数信令来重新使用变换块编解码。这种模式可以例如对于图形内容的局部编解码是有用的,其中量化伪像可能是高度可见的或者完全不可容忍的。如果通常变换编解码的速率失真成本(通常在使用低量化参数时)恰好超过旁路编解码的速率成本,编码器也可以切换到该模式。对于以无损模式编解码的CU,后置滤波器被禁用。

[0054] 如果在序列参数集(SPS)中激活,PCM编解码可以在CU级别上被指示。如果对所考虑的CU有效,则不应用预测、量化和变换。相反,对应编解码块中的采样的采样值以在SPS中配置的PCM采样比特深度被直接编解码到比特流中。PCM编解码的应用的粒度可以被配置在高端上的亮度编解码树块尺寸和32×32的最小值与低端上的最小亮度编解码块尺寸之间。如果编解码单元被以PCM模式编解码,则同一编解码树单元中其他编解码单元的尺寸不得小于PCM单元的尺寸。因为根据定义,PCM编解码实现对对应块的无损表示,所以为编解码单元的PCM编解码所花费的比特可以被认为是对CU进行编码所需的比特量的上限。因此,在基于变换的残差编解码的应用将超过该限制的情况下(例如对于异常嘈杂的内容),编码器可以切换到PCM编解码。

[0055] HEVC的无损模式是使用画面参数集(PPS)中编解码的transquant_bypass_enabled_flag标志激活的(见下表1)。该标志实现在CU级别处对cu_transquant_bypass_flag的编解码(表2)。cu_transquant_bypass_flag指定绕过量化、变换处理和环路滤波器。

[0056] 表1:HEVC中用于在画面级用信号通知无损编解码的PPS语法

[0057]

pic_parameter_set_rbsp( ) {	描述符
pps_pic_parameter_set_id	ue(v)
…….	

[0058]

transquant_bypass_enabled_flag	u(1)

[0059] 表2:HEVC中用于无损编解码的编解码单元语法

[0060]

coding_unit(x0, y0, log2CbSize) {	描述符
if(transquant_bypass_enabled_flag)	
cu_transquant_bypass_flag[x0][y0]	ae(v)
if(slice_type != I)	
cu_skip_flag[x0][y0]	ae(v)

[0061] 如果对于所有CU来说cu_transquant_bypass_flag为真,则可以对整个帧使用无损编解码,但也可能仅对一区域进行无损编解码。对于具有重叠的文本和图形的混合的内容与自然视频通常是这种情况。文本和图形区域可以无损编解码以最大化可读性,而自然内容可以以有损方式编解码。

[0062] 此外,由于量化按照定义是有损的,所以对于无损编解码是禁用的。在HEVC中,变换处理(HEVC中使用DCT-2和DST-4)由于舍入运算而是有损的。如果重构的信号是无损的,则如去块滤波器和采样自适应偏移的后置滤波器是无用的。

[0063] 图3示出了用于PCM、无损和变换跳过模式的解码处理的简化框图的示例。解码处理300的输入是比特流。在步骤310中,从比特流中解码数据。特别地,这提供了量化的变换系数和变换类型。在步骤320中,逆量化被应用于量化的变换系数。在步骤330中,逆变换被应用于所得的变换系数。所得的残差在步骤350中被添加到来自步骤340的帧内或帧间预测的预测信号中。结果是重构的块。当在步骤305中选择了I_PCM模式时,采样值被直接解码而无需熵解码。当选择无损编解码模式时,跳过步骤320和330。当选择变换跳过模式时,跳过步骤330。

[0064] 在最近的通用视频编解码(VVC)测试模型4(VTM4)中,启用了高达64×64的大变换尺寸,这主要用于更高分辨率的视频,例如1080p和4K序列。对于尺寸(宽度或高度,或者宽度和高度两者)等于64的变换块,高频变换系数被归零,使得仅保留低频系数,因此丢失该信息。更准确地,该信息由于没有被编码在所得的比特流中而丢失,因此对解码器是不可用的。例如,对于M×N变换块,其中以M为块宽,并且N为块高,当M等于64时,仅保留变换系数的左侧32列。类似地,当N等于64时,仅保留变换系数的顶部32行。在这两个示例中,变换系数的剩余的32列或32行被丢失。当对大的块使用变换跳过模式时,使用整个块而不归零任何值。实际上,这包括设置变换的最大尺寸(设置为5),如下表3中示出的从当前VVC规范摘录的语法所示(相关行被设置为粗体)。因此,使用最大实际变换尺寸,例如在当前VVC规范中设置为32(2^5)。在下面该参数将被命名为“max_actual_transf_size”。

[0065] 表3

[0066]

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	描述符
if((tu_mts_idx[x0][y0] > 0 || (cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6)) && cIdx = = 0 && log2TbWidth > 4)	
log2ZoTbWidth = 4	
else	
log2ZoTbWidth = Min(log2TbWidth, 5)	
if(tu_mts_idx[x0][y0] > 0 || (cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6)) && cIdx = = 0 && log2TbHeight > 4)	
log2ZoTbHeight = 4	
else	
log2ZoTbHeight = Min(log2TbHeight, 5)	

[0067] 除了在HEVC使用的DCT-2之外,VVC还增加了用于帧间和帧内编解码块的残差编解码的多变换选择(MTS)方案。新引入的变换矩阵是DST-7和DCT-8。为了控制MTS方案,在SPS级别分别为帧内和帧间指定了单独的启用标志。当在SPS级别启用MTS时,CU级别的MTS索引被用信号通知以指示在DCT-2、DST-7和DCT-8中用于CU的可分离变换对。MTS仅适用于亮度。当满足以下条件时,CU级MTS索引被用信号通知:宽度和高度都小于或等于32,并且CBF标志等于1。

[0068] 为了降低大尺寸DST-7和DCT-8的复杂性,对于尺寸(宽度或高度,或者宽度和高度两者)等于32的DST-7和DCT-8块,高频变换系数被归零。仅保留16x16低频区域内的系数。

[0069] 对变换跳过的块尺寸限制与MTS的块尺寸限制相同,这规定当块宽度和高度都等于或小于32时,变换跳过适用于CU。

[0070] 对于cu_cbf等于1的帧间预测CU,可以用信号通知cu_sbt_flag来指示是解码整个残差块还是解码残差块的子部分。在前一种情况下,进一步解析帧间MTS信息以确定CU的变换类型。在后一种情况下,残差块的一部分被用推断的自适应变换进行编解码,而残差块的其他部分被归零。

[0071] 图4示出了每个SBT位置的水平和竖直变换。子块变换是应用于亮度变换块的位置相关变换。SBT-H(水平)和SBT-V(竖直)的两个位置与不同的核心变换相关联。更具体地,在图4中指定了每个SBT位置的水平和竖直变换。例如,SBT-V位置0的水平和竖直变换分别是DCT-8和DST-7。当残差TU的一侧大于32时,对应的变换被设置为DCT-2。因此,子块变换联合地指定残差块的TU平铺(tiling)、cbf以及水平和竖直变换,这可以被认为是在块的主要残差在块的一侧的情况下的语法捷径。

[0072] 在VTM4中,在环路滤波器之前添加了称为亮度映射和色度缩放(LMCS)的编解码工具,作为新的处理块。LMCS有两个主要部分:1)亮度分量的基于自适应分段线性模型的环内映射;2)对于色度分量,应用亮度相关的色度残差缩放。

[0073] 图5从解码器的角度示出了LMCS架构。图5中浅灰色阴影块指示处理在映射域中应用的位置;并且这些包括逆量化、逆变换、亮度帧内预测以及亮度预测与亮度残差的相加。图5中的无阴影块指示处理在原始(即,未映射)域中应用的位置;并且这些包括诸如去块、ALF和SAO的环路滤波器、运动补偿预测、色度帧内预测、色度预测与色度残差的相加以及将经解码的画面作为参考画面的存储。图5中的深灰色阴影块是新的LMCS功能块,包括亮度信号的正向和反向映射以及亮度相关的色度缩放处理。像VVC的大多数其他工具一样,可以使用SPS标志在序列级别启用/禁用LMCS。

[0074] 也已知为不可分离二次变换(NSST)或简化二次变换(RST),二次变换被应用于正向主变换和量化之间(在编码器处)以及去量化和反向主变换之间(在解码器侧)。

[0075] 图6示出了二次变换的应用。在JEM中,如图所示,对于每个8×8块,4×4二次变换被应用于小块(即,min(宽度,高度)<8),并且8×8二次变换被应用于较大块(即,min(宽度,高度)>4)。

[0076] 下面以输入为例描述不可分离变换的应用。4x4输入块被表示为以下矩阵:

[0077] $$\begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix}$$

[0078] 为了应用不可分离变换,这个4x4输入块X首先被表示为矢量$\mathbf{X}\vec{}$:

[0079] $$\vec{X} = [X_{00} \quad X_{01} \quad X_{02} \quad X_{03} \quad X_{10} \quad X_{11} \quad X_{12} \quad X_{13} \quad X_{20} \quad X_{21} \quad X_{22} \quad X_{23} \quad X_{30} \quad X_{31} \quad X_{32} \quad X_{33}]^T$$

[0080] 不可分离变换被计算为$\mathbf{F}\vec{}=\mathbf{T}\cdot\mathbf{X}\vec{}$,其中$\mathbf{F}\vec{}$表示变换系数矢量,并且T是16×16变换矩阵。16×1系数矢量$\mathbf{F}\vec{}$随后被使用该块的扫描顺序(水平、竖直或对角)而重新组织为4×4块。具有较小索引的系数将与较小扫描索引一起放置在4x4系数块中。一共存在35个变换集,并且每个变换集使用3个不可分离的变换矩阵(核)。从帧内预测模式到变换集的映射是预先定义的。对于每个变换集,所选择的不可分离的二次变换候选进一步由显式用信号通知的二次变换索引来指定。该索引在每帧内CU在变换系数之后被在比特流中用信号通知一次。

[0081] 图7示出了简化二次变换(RST)。利用这种技术,16x48和16x16矩阵分别被用于8×8和4×4块。为了便于标注,16x48变换被表示为RST8x8,并且16x16变换被表示为RST4x4。简化变换(RT)的主要思想是将N维矢量映射到不同空间中的R维矢量,其中R/N(R<N)是简化因子。

[0082] 图8示出了正向和反向简化变换。RT矩阵是R×N矩阵,如下:

[0083] $$T_{RxN} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1N} \\ t_{21} & t_{22} & t_{23} & & t_{2N} \\ & \vdots & & \ddots & \vdots \\ t_{R1} & t_{R2} & t_{R3} & \cdots & t_{RN} \end{bmatrix}$$

[0084] 其中变换的R行是N维空间的R个基。RT的逆变换矩阵是其正变换的转置。

[0085] 图9示出了利用16x48矩阵的正向RST8x8处理的示例。在所采用的配置中,应用了16x48矩阵,而不是具有相同变换集配置的16x64矩阵,每个矩阵从左上8x8块中的三个4x4块(不包括右下4x4块)获取48个输入数据(图4)。在降维的帮助下,用于存储所有RST矩阵的

内存使用从10KB减少到8KB,同时性能合理下降。

[0086] 此外,在VTM5中,已采用了称为色度残差联合编解码的编解码工具。当该工具被激活时,单个联合残差块被用于描述同一变换单元中Cb和Cr块两者的残差,如等式1所示。

[0087] $res_{joint} = (res_{Cb} - res_{Cr})/2$

[0088] 等式1:根据Cb和Cr残差计算的联合残差

[0089] 然后,通过减去Cb的联合残差并将其与Cr相加来重构Cb和Cr信号,如等式2所示。

[0090] $$\begin{cases} rec_{Cb} = pred_{Cb} + res_{join} \\ rec_{cr} = pred_{Cr} - res_{joint} \end{cases}$$

[0091] 等式2:从联合残差编解码重构Cb和Cr信号

[0092] 在TU级别编解码标志,以启用色度残差的联合编解码,如果该标志被禁用,则使用Cb和Cr残差的单独编解码。

[0093] 下文描述的实施例是已经考虑到前述内容而设计的。图1A的编码器100、图1B的解码器200和图2的系统1000适于实现下述实施例中的至少一个。

[0094] 在至少一个实施例中,本申请针对视频编解码系统中的无损编解码模式,该视频编解码系统包括多个编解码工具,其中一些通过设计是有损的,其中一些可以适于成为无损或近无损的。为了在例如VVC的视频编解码系统中启用无损模式,提出了以下策略:

[0095] -禁用通过设计有损的工具并且仅使用无损工具,

[0096] -适配一些工具以启用无损编解码,

[0097] -适配一些工具以启用近无损编解码(与原始信号的差有限且较小),使得可以在残差编解码之后应用二次无损编解码,从而提供无损编解码。

[0098] 无损编解码可以在帧级或区域级处理。

[0099] 图10A和10B示出了根据至少一实施例的示例流程图。该图示出了通用处理的示例,该通用处理用于决定给定工具是否应该在帧级、块级被禁用,是否应该被适配为无损的,或者二次无损编解码是否应该被执行以使用该被适配为近无损的工具。这种处理可以例如在图1A的编码器设备100中实现。

[0100] 在第一步(图10A的400),评估工具是否通过设计而是有损的。

[0101] ●如果工具是有损的,则第二检查(图10B的402)测试工具是否可以在块级禁用(在这种情况下,只能在帧级进行控制)。

[0102] ○如果该工具不能在块级被禁用,则在图10B的步骤404中检查标志transquant_bypass_enabled_flag(通常在PPS级用信号通知)的值。如果transquant_bypass_enabled_flag为真,则在图10B的步骤410中禁用该工具(这例如适用于LMCS)。如果transquant_bypass_enabled_flag为假,则在图10B的步骤411中启用该工具。一旦应用了步骤410或411,处理结束。

[0103] ○如果该工具可以在块级被禁用,则在图10B的步骤405中检查CU级标志cu_transquant_bypass_flag的值。如果cu_transquant_bypass_flag为真,则在图10B的步骤408中在CU级禁用工具(这适用于例如SBT、MTS、LFNTS)。如果cu_transquant_bypass_flag为假,则在图10B的步骤409中,可以在CU级启用该工具。一旦应用了步骤408或409,处理结束。

[0104] ●如果工具通过设计是无损的,则在图10A的步骤401中执行测试,以检查是否无

损。

[0105] ○如果工具是无损的,则无特别应用,并且流程进行到处理的结束。

[0106] ○如果工具不是无损的,则在图10A的步骤403中检查工具是否近无损的。

[0107] ■如果该工具不是近无损的,由于其也不是通过设计有损的,这意味着它可以适于成为无损的(图10A的步骤406)。这可以适用于例如在MTS变换集中仅使用无损变换的MTS。

[0108] ■如果该工具是近无损的,则应用附加的(二次)无损编解码步骤(图10A的步骤407)。例如,可以应用没有量化的无损变换。

[0109] ○一旦应用了步骤306或307,处理结束。

[0110] 在本公开的下文中,描述了该处理对VVC规范的特定工具的应用。

[0111] 禁用与无损编解码不兼容的工具

[0112] 第一种情况对应于图10A的步骤410(画面级)和408(CU级)。第一元素与具有大CU的归零变换有关。在实施例中,在无损编解码的CU的情况下或者如果使用变换跳过,则不能使用归零变换。表4中的语法规范对此进行了说明,其以斜体文本突出了与当前VVC语法相比所提出的变化。如果块大于max_actual_transf_size x max_actual_transf_siz(实际上在当前的VVC版本中是32x32)并且是无损编解码的,则所有的系数都被编解码,这与只有一部分系数被编解码的有损编解码块的情况相反。

[0113] 实际上,与当前的VVC规范相比,仅当transform_skip_flag为假且cu_transquant_bypass_flag为假时,才应用变换尺寸限制。

[0114] 表4:用于如果对当前CU使用了变换跳过或无损编解码则不使用归零变换的修改的语法

[0115]

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	描述符
if((tu_mts_idx[x0][y0] > 0 || (cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6)) && cIdx = = 0 && log2TbWidth > 4)	
log2TbWidth = 4	
else if(!transform_skip_flag[x0][y0] && !cu_transquant_bypass_flag[x0][y0]) {	
log2TbWidth = Min(log2TbWidth, 5)	
}	
if(tu_mts_idx[x0][y0] > 0 || (cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6)) && cIdx = = 0 && log2TbHeight > 4)	
log2TbHeight = 4	
else if(!transform_skip_flag[x0][y0] && !cu_transquant_bypass_flag[x0][y0]) {	
log2TbHeight = Min(log2TbHeight, 5)	
}	

[0116] 在变型中,如果画面级transquant_bypass_enabled_flag被启用,并且如果块大于max_actual_transf_size x max_actual_transf_size(例如,在当前VVC中为32x32),则推断或强制四叉树划分。换句话说,当希望无损编解码时,大于64x64的块被系统地划分,使得块尺寸变为32x32,并因此不经受归零。表5示出了修改的语法。

[0117] 表5:在画面级启用无损的情况下四叉树划分推断的语法

[0118]

coding_tree_unit( ) {	描述符
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
…….	
if(slice_type = = I && qtbtt_dual_tree_intra_flag)	
dual_tree_implicit_qt_split (xCtb, yCtb, CtbSizeY, 0)	
else if (transquant_bypass_enabled_flag)	
transquant_implicit_qt_split(xCtb, yCtb, CtbSizeY, CtbSizeY, 0)	
else	
coding_tree(xCtb, yCtb, CtbSizeY, CtbSizeY, 1, 0, 0, 0, 0, 0, SINGLE_TREE)	
}	
transquant_implicit_qt_split(x0, y0, cbSize, cqtDepth) {	
*cbSubdiv = 2 * cqtDepth*	
if(cbSize > 32) {	
x1 = x0 + (cbSize / 2)	
y1 = y0 + (cbSize / 2)	
transquant_implicit_qt_split(x0, y0, cbSize / 2, cqtDepth + 1)	
if(x1 < pic_width_in_luma_samples)	
transquant_implicit_qt_split(x1, y0, cbSize / 2, cqtDepth + 1)	
if(y1 < pic_height_in_luma_samples)	
transquant_implicit_qt_split(x0, y1, cbSize / 2, cqtDepth + 1)	
if(x1 < pic_width_in_luma_samples && y1 < pic_height_in_luma_samples)	
transquant_implicit_qt_split(x1, y1, cbSize / 2, cqtDepth + 1)	
} else if (slice_type = = I && qtbtt_dual_tree_intra_flag) {	
coding_tree(x0, y0, cbSize, cbSize, 1, cbSubdiv, cqtDepth, 0, 0, 0, DUAL_TREE_LUMA)	

[0119]

coding_tree(x0, y0, cbSize, cbSize, 0, cbSubdiv, cqtDepth, 0, 0, 0, DUAL_TREE_CHROMA)	
}	
else {	
coding_tree(x0, y0, cbSize, cbSize, 1, cbSubDiv, cqtDepth, 0, 0, 0, SINGLE_TREE)	
}	
}	

[0120] 在另一变型中,只有当前CU小于或等于max_actual_transf_size x max_actual_transf_size(例如,在当前VVC中为32x32)时,才对cu_transquant_bypass_flag进行编解码。这意味着较大的块不能被无损编解码。表6示出了相关联的语法。

[0121] 表6:用于仅针对小于或等于32x32的块编解码cu_transquant_bypass的所提出的语法

[0122]

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	**描述符**
if(transquant_bypass_enabled_flag *&& cbWidth <= 32 && cbHeight <= 32*)	
cu_transquant_bypass_flag[x0][y0]	ae(v)
if(slice_type != I || sps_ibc_enabled_flag)	
cu_skip_flag[x0][y0]	ae(v)

[0123] 第二元素与子块变换(SBT)相关。由于SBT使用变换树细分,其中变换单元中的一个被推断为没有残差,因此该工具不能保证在无损的情况下重构CU。在实施例中,当无损模式被激活时(即,当transquant_bypass_enabled_flag为真时),SBT在CU级被禁用。对应的语法变化如表7所示。只有当transquant_bypass_enabled_flag为假时,才可能解码SBT相关语法并激活SBT。

[0124] 表7:当对于无损编解码CU禁用SBT时的编解码单元语法

[0125]

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	**描述符**
…	

[0126]

if(CuPredMode[x0][y0] != MODE_INTRA && cu_skip_flag[x0][y0] == 0)	
cu_cbf	ae(v)
if(cu_cbf) {	
if(CuPredMode[x0][y0] != MODE_INTRA && sps_sbt_enable_flag && *! transquant_bypass_enabled_flag*) {	
if(cbWidth <= maxSbtSize && cbHeight <= maxSbtSize) {	
allowSbtVerHalf = cbWidth >= 8	
allowSbtVerQuad = cbWidth >= 16	
allowSbtHorHalf = cbHeight >= 8	
allowSbtHorQuad = cbHeight >= 16	
if(allowSbtVerHalf || allowSbtHorHalf || allowSbtVerQuad || allowSbtHorQuad)	
cu_sbt_flag[x0][y0]	ae(v)
}	
if(cu_sbt_flag[x0][y0]) {	
if((allowSbtVerHalf || allowSbtHorHalf) && (allowSbtVerQuad || allowSbtHorQuad))	
cu_sbt_quad_flag[x0][y0]	ae(v)
if((cu_sbt_quad_flag[x0][y0] && allowSbtVerQuad && allowSbtHorQuad) || (!cu_sbt_quad_flag[x0][y0] && allowSbtVerHalf && allowSbtHorHalf))	
cu_sbt_horizontal_flag[x0][y0]	ae(v)
cu_sbt_pos_flag[x0][y0]	ae(v)
}	
}	
transform_tree(x0, y0, cbWidth, cbHeight, treeType)	
}	
}	

[0127] 在替代实现方式中,SBT只能用作TU平铺(tiling)。在该实施例中,可以针对每个子块编解码残差,而在初始设计中,一些块被强制具有值为0的系数。水平和竖直变换被推断为变换跳过。对应的语法变化如表8所示。

[0128] 表8:当没有为SBT编解码的CU中的两个CU推断残差时的编解码单元语法

[0129]

```
transform_tree( x0, y0, tbWidth, tbHeight , treeType) {
    if( tbWidth > MaxTbSizeY || tbHeight > MaxTbSizeY ) {
        trafoWidth = ( tbWidth > MaxTbSizeY ) ? (tbWidth / 2) : tbWidth
        trafoHeight = ( tbHeight > MaxTbSizeY ) ? (tbHeight / 2) : tbHeight
        transform_tree( x0, y0, trafoWidth, trafoHeight )
        if( tbWidth > MaxTbSizeY )
            transform_tree( x0 + trafoWidth, y0, trafoWidth,
                            trafoHeight, treeType )
        if( tbHeight > MaxTbSizeY )
            transform_tree( x0, y0 + trafoHeight, trafoWidth,
                            trafoHeight, treeType )
        if( tbWidth > MaxTbSizeY && tbHeight > MaxTbSizeY )
            transform_tree( x0 + trafoWidth, y0 + trafoHeight,
                            trafoWidth, trafoHeight, treeType )
    } else if( cu_sbt_flag[ x0 ][ y0 ] ) {
        factorTb0 = cu_sbt_quad_flag[ x0 ][ y0 ] ? 1 : 2
        factorTb0 = cu_sbt_pos_flag[ x0 ][ y0 ] ? ( 4 - factorTb0 ) : factorTb0
        noResiTb0 = (cu_sbt_pos_flag[ x0 ][ y0 ] && !transquant_bypass_enabled_flag )? 1 : 0
        if( !cu_sbt_horizontal_flag[ x0 ][ y0 ] ) {
            trafoWidth = tbWidth * factorTb0 / 4
            transform_unit( x0, y0, trafoWidth, tbHeight, treeType ,
                            noResiTb0 )
```

[0130]

transform_unit(x0+trafoWidth, y0, tbWidth−trafoWidth, tbHeight, treeType , !noResiTb0)	
}	
else {	
trafoHeight = tbHeight * factorTb0 / 4	
transform_unit(x0, y0, tbWidth, trafoHeight, treeType , noResiTb0)	
transform_unit(x0, y0+trafoHeight, tbWidth, tbHeight−trafoHeight, treeType , !noResiTb0)	
}	
} else {	
transform_unit(x0, y0, tbWidth, tbHeight, treeType , 0)	
}	
}	

[0131] 图11示出了用于亮度整形的正向然后反向函数的应用。事实上,第三元素与亮度整形(LMCS)有关。整形是有损变换,因为由于舍入操作,应用正向然后反向函数不保证给出原始信号。在LMCS中,参考画面被存储在原始域中。在帧内情况下,预测处理在“整形”域中实现,并且一旦采样被重构,就在环路滤波步骤之前应用逆整形。在帧间情况下,在运动补偿之后,预测信号被正向整形。然后,一旦采样被重构,就在环路滤波步骤之前应用逆整形。

[0132] 在实施例中,如果在PPS中允许无损编解码,则在条带(slice)级禁用LMCS。对应的语法变化如表9所示。

[0133] 表9:用于如果在PPS中启用了transquant旁路则禁用lmcs的条带头语法修改

[0134]

slice_header( ) {	**描述符**
slice_pic_parameter_set_id	ue(v)
……..	
if(sps_lmcs_enabled_flag && *! transquant_bypass_enabled_flag*) {	
slice_lmcs_enabled_flag	

[0135]

if(slice_lmcs_enabled_flag) {	
slice_lmcs_aps_id	
if(!(qtbtt_dual_tree_intra_flag && slice_type == I))	
slice_chroma_residual_scale_flag	
}	

[0136] 第四元素与多变换选择(MTS)和变换跳过相关,并且实施例涉及推断变换跳过为真。即使量化步长等于1,DCT和DST变换也是有损的,因为舍入误差会导致轻微的损失。在第一实施例中,如果CU被无损编解码(由CU级标志cu_transquant_bypass_flag的值检查),则只能使用变换跳过。transform_skip_flag被推断为1,并且tu_mts_idx未被编解码。对应的语法变化如表10中所示。在transform_skip_flag被推断为1的情况下,对无损块使用用于变换跳过的残差编解码。

[0137] 表10:用于在启用CU无损时禁用MTS和变换跳过的编解码的变换单元语法

[0138]

if(tu_cbf_luma[x0][y0] && treeType != DUAL_TREE_CHROMA
&& (tbWidth <= 32) && (tbHeight <= 32)
&& (IntraSubPartitionsSplit[x0][y0] == ISP_NO_SPLIT)
&& (!cu_sbt_flag) && && *!cu_transquant_bypass_flag*) {

if(transform_skip_enabled_flag && tbWidth <= MaxTsSize && tbHeight <= MaxTsSize)
transform_skip_flag[x0][y0]
if(((CuPredMode[x0][y0] != MODE_INTRA && sps_explicit_mts_inter_enabled_flag) || (CuPredMode[x0][y0] == MODE_INTRA && sps_explicit_mts_intra_enabled_flag)) && (tbWidth <= 32) && (tbHeight <= 32) && (!transform_skip_flag[x0][y0]))
tu_mts_idx[x0][y0]
}

[0139] 在VTM-5.0中,可以使用两种不同的残差编解码处理,第一种用于非变换跳过残差编解码,并且对于编解码自然内容块的残差是有效的,第二种用于变换跳过残差编解码,其对于编解码屏幕内容块是有效的。残差编解码语法选择如表11所示。

[0140] 表11:VTM-5.0中的残差编解码语法

[0141]

```
if( tu_cbf_luma[ x0 ][ y0 ] ) {
  if( !transform_skip_flag[ x0 ][ y0 ] )
    residual_coding( x0, y0, Log2( tbWidth ), Log2( tbHeight ), 0 )
  else
    residual_ts_coding( x0, y0, Log2( tbWidth ), Log2( tbHeight ), 0 )
  }
```

[0142] 图12示出了包括对cu_transquant_bypass_flag、transform_skip_flag的解析以及当变换跳过对于无损编解码块被推断为真时的残差编解码的实施例的示例流程图。在第一步骤800,解析cu_transquant_bypass_flag,如果该标志为真,则在步骤801将transform_skip_flag推断为真,否则在步骤802解析transform_skip_flag。如果transform_skip_flag等于假,则在步骤803解析常规残差,否则如果该标志为真,则在步骤804解析变换跳过残差。

[0143] 图13示出了包括对cu_transquant_bypass_flag、transform_skip_flag的解析以及当变换跳过对于无损编解码块被推断为假时的残差编解码的实施例的示例流程图。在第一步骤900,解析cu_transquant_bypass_flag,如果该标志为真,则在步骤901将transform_skip_flag推断为假,并且在步骤904使用常规残差编解码,否则在步骤902解析transform_skip_flag。如果transform_skip_flag等于假,则在步骤904解析常规残差,否则如果该标志为真,则在步骤903解析变换跳过残差。

[0144] 由于用于变换跳过的残差编解码被设计用于对来自屏幕内容编解码块的残差进行编解码,因此对于针对无损块的自然内容进行编解码可能效率较低。因此,在另一实施例中,如果CU被无损编解码,则transform_skip_flag被推断为0,并且tu_mts_idx不被编解码。用于常规变换的残差编解码被用于无损编解码块。因此,在这种特殊模式下(即,当cu_transquant_bypass_flag为真时),即使在transform_skip_flag为真的情况下,也将使用常规编解码,而通常在transform_skip_flag为真时,应使用变换跳过残差编解码。

[0145] 换句话说,图13中描述的该实施例提出,在信息指示使用变换跳过残差编解码的情况下,通过以下来确定残差编解码的类型:获得表示特殊模式的标志,并且在该标志为真时为残差编解码选择常规残差编解码,而不是应该使用的变换跳过残差编解码。表示特殊模式的标志可以从其他信息中导出,诸如编解码是无损的指示(例如cu_transquant_bypass_flag),或者量化、变换处理和环路滤波器被绕过(未使用)的指示,或者残差编解码被强制为常规残差编解码的指示。

[0146] 图14示出了包括对cu_transquant_bypass_flag、transform_skip_flag的解析以及当对于无损编解码块总是解析变换跳过时的残差编解码的实施例的示例流程图。

[0147] 在这样的实施例中,为了保持无损编解码块的设计接近有损编解码块,可以使用常规残差编解码和变换跳过编解码。在这种情况下,transform_skip_flag针对无损编解码块进行编解码。这允许对残差系数进行编解码的两种不同方式之间的竞争,并使残差编解码更好地适应于内容。

[0148] 在第一步骤1400,解析cu_transquant_bypass_flag,然后在步骤1401解析

transform_skip_flag。如果transform_skip_flag等于假,则在步骤1403解析常规残差,否则如果该标志为真,则在步骤1402解析变换跳过残差。

[0149] 实际上,变换跳过残差编解码方法被优化来编解码计算机生成的内容,其中空间相邻系数的值之间的相关性很强。一般地,在有损编解码中,当内容是计算机生成的并且对这种内容有效时,变换跳过系数编解码是最常用的。

[0150] 但是在该实施例中,提出使用标志在选择常规系数编解码方法和变换跳过系数编解码方法之间进行选择。实际上,变换跳过系数编解码可以用于计算机生成的内容,并且常规系数编解码方法可以用于常规内容。这允许对残差系数进行编解码的两种不同方式之间的竞争,并使残差编解码方法更好地适应于内容。该标志仅改变用于对块的残差进行编解码的系数编解码方法。

[0151] 在VVC,使用2种方法来对残差进行编解码:

[0152] -当使用变换和量化时,使用常规系数编解码方法。该方法继承了HEVC加一些改进,它已被设计成在变换后对残差进行编解码,其中能量被压缩在信号的低频部分。

[0153] -当仅使用量化时,使用变换跳过系数编解码方法。该方法是针对计算机生成的内容设计的,其中,由于系数之间存在大量的空间相关性,所以变换经常被跳过。变换跳过残差编解码方法的主要特性在于它们不用信号通知最后的非零系数,从块的左上角开始,使用先前系数的缩减模板来编解码当前系数,并且系数的符号是Cabac上下文编解码的。

[0154] 图12、13和14示出,与当前的VVC规范不同,残差编解码方法可以与变换是否已经被跳过的事实解耦。实际上,在当前的VVC规范中,transform_skip_flag语法元素等于真意味着不对块的残差应用变换,并且使用变换跳过系数编解码方法。而在所提出的方法中,如果当前块被无损编解码(跳过了变换和量化),则使用常规系数编解码方法,通过将transform_skip_flag推断为假,在编解码自然内容的情况下使用更好的系数编解码方法。在变型中,即使cu_transquant_bypass_flag为真(在这种情况下,跳过了变换和量化),也会对标志(这里transform_skip_flag被重新使用以用于此目的)进行编解码,以仅指示使用哪种残差编解码方法。

[0155] 一实施例涉及低频不可分离变换(LFNTS)。在该实施例中,如果当前CU被无损编解码,则LFNTS被禁用。对应的语法变化如表12所示。

[0156] 表12:用于在CU被无损编解码时禁用lfnts的编解码单元语法

[0157]

```
numZeroOutSigCoeff = 0
transform_tree( x0, y0, cbWidth, cbHeight, treeType )
if(Min( cbWidth, cbHeight ) >= 4 && sps_st_enabled_flag == 1 &&
  CuPredMode[ x0 ][ y0 ] == MODE_INTRA
  && IntraSubPartitionsSplitType == ISP_NO_SPLIT
  && !cu_transquant_bypass_flag ) {
  if( ( numSigCoeff > ( ( treeType == SINGLE_TREE ) ? 2 : 1 ) ) &&
      numZeroOutSigCoeff == 0 ) {
    st_idx[ x0 ][ y0 ]
  }
}
```

[0158] 一实施例涉及联合Cb-Cr编解码。Cb和Cr残差的联合编解码处理是不可逆的。通过使用等式1和等式2,其中resCb＝resCr,我们得到resjoint＝0,并且recCb＝predCb,recCr＝predCr。在一实施例中,如果cu_transquant_bypass_flag为真,则在CU级禁用联合Cb-Cr编解码。对应的语法变化如表13所示。

[0159] 表13:用于如果cu_transquant_bypass_flag为真则禁用联合Cb-Cr编解码的残差编解码语法

[0160]

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	描述符
if(cIdx == 2 && tu_cbf_cb[x0][y0] && !cu_transquant_bypass_flag) {	
tu_cb_cr_joint_residual[x0][y0]	ae(v)
if(tu_cb_cr_joint_residual [x0][y0]) {	
return	
}	
…	

[0161] 当等式1将resjoint_Cb与res Joint_Cr分离时,如等式3所示。

[0162] 等式3:对N0347中提出的联合残差计算的修改

[0163] $$\begin{cases} res_{joint\_Cb} = (res_{Cb} - res_{Cr})/2 \\ res_{joint\_cr} = (res_{Cb} + res_{Cr})/2 \end{cases}$$

[0164] 通过该提出的修改,该处理是可逆的,但由于舍入误差而是有损的,如等式4所示。

[0165] 等式4:对从联合残差编解码重构Cb和Cr信号的修改

[0166] $$\begin{cases} rec_{Cb} = pred_{Cb} + res_{joint\_Cb} + res_{joint\_Cr} \\ rec_{cr} = pred_{Cr} - res_{joint\_Cb} + res_{joint\_Cr} \end{cases}$$

[0167] 当如下所述执行二次无损编解码处理时,该变型可以用于无损编解码。

[0168] 用于无损编解码的VVC工具的适配

[0169] 参考图10A,这种情况对应于步骤406。

[0170] 在一实施例中,无损变换被添加到MTS变换集中。对于无损编解码,应选择那些变换。无损变换的示例是非归一化沃尔什-哈达玛(Walsh-Hadamard)变换、非归一化哈尔(Haar)变换。近无损变换的示例是归一化沃尔什-哈达玛变换、归一化哈尔变换。

[0171] 无损变换可以通过沃尔什-哈达玛变换或哈尔变换获得。沃尔什-哈达玛和哈尔的非归一化变换由±1和零构成,如等式5和等式6所示。例如,非归一化4×4沃尔什-哈达玛矩阵为:

[0172] 等式5:非归一化4x4沃尔什-哈达玛矩阵

[0173] $$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}$$

[0174] 而非归一化4x4哈尔矩阵为:

[0175] 等式6:非归一化4x4哈尔矩阵

[0176] $$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & 0 & 0\\ 0 & 0 & 1 & -1\end{bmatrix}$$

[0177] 利用沃尔什-哈达玛变换或哈尔变换可以实现无损重构。为了解释这一点,考虑2x2变换的示例,它采用残差采样(r0和r1)。哈尔和沃尔什-哈达玛的变换矩阵为:

[0178] 等式7:非归一化2x2沃尔什-哈达玛或哈尔矩阵

[0179] $$\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

[0180] 变换系数(c0和c1)通过下式获得:

[0181] 等式8:利用2x2沃尔什-哈达玛或哈尔非归一化变换计算的变换系数

[0182] $$\begin{bmatrix}c_0\\ c_1\end{bmatrix}=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\times\begin{bmatrix}r_0\\ r_1\end{bmatrix}$$

[0183] 逆变换以这种方式执行:

[0184] 等式9:利用2x2沃尔什变换或哈尔非归一化变换的逆变换

[0185] $$\begin{bmatrix}r_0\\ r_1\end{bmatrix}=\frac{1}{2}\times\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\times\begin{bmatrix}c_0\\ c_1\end{bmatrix}$$

[0186] 在4×4沃尔什-哈达玛变换的另一示例中,为了变换4个残差采样(r0、r1、r2、r3),将矩阵乘以系数矢量以获得变换系数c0、c1、c2和c3:

[0187] 等式10:利用4x4沃尔什-哈达玛非归一化变换计算的变换系数

[0188] $$\begin{bmatrix}c_0\\ c_1\\ c_2\\ c_3\end{bmatrix}=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}\times\begin{bmatrix}r_0\\ r_1\\ r_2\\ r_3\end{bmatrix}$$

[0189] 逆变换以这种方式执行

[0190] 等式11:利用4x4沃尔什变换非归一化变换的逆变换

[0191] $$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \frac{1}{4} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

[0192] 最后,为了使用哈尔变换来变换4个残差采样,使用以下等式等式12:利用4x4哈尔变换计算的变换系数

[0193] $$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

[0194] 逆变换以这种方式执行

[0195] 等式13:利用4x4哈尔非归一化变换的逆变换

[0196] $$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \frac{1}{4} \times \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

[0197] 非归一化哈尔或沃尔什-哈达玛变换有时可以增加比特率,因为系数的动态性增加了。例如,当使用4×4非归一化沃尔什-哈达玛矩阵时,系数能量可以高达残差采样能量的2倍。因此,归一化矩阵是通过将非归一化矩阵除以2来实现的。然而,这样做会导致整数表示的损失。考虑使用“归一化”沃尔什-哈达玛变换来变换4个残差采样的相同示例:

[0198] 等式14:利用4x4沃尔什-哈达玛归一化变换计算的变换系数

[0199] $$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{1}{2} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

[0200] 由于整数表示被用于变换系数,所以舍入会导致+-1/2的误差。因此,需要附加编解码步骤,以在除以2后对提示进行编解码。这可以通过以下方式完成:

[0201] 首先,计算对每个系数的除法的提示:

[0202] 等式15:对使用4x4沃尔什-哈达玛归一化变换计算的每个系数的除法的提示

[0203] $$\begin{cases} rem\_c_0 = 2 \times c_0 - (r_0 + r_1 + r_2 + r_3) \\ rem\_c_1 = 2 \times c_1 - (r_0 + r_1 - r_2 - r_3) \\ rem\_c_2 = 2 \times c_2 - (r_0 - r_1 - r_2 + r_3) \\ rem\_c_3 = 2 \times c_3 - (r_0 - r_1 + r_2 - r_3) \end{cases}$$

[0204] 其中提示(rem_c_0、rem_c_1、rem_c_2和rem_c_3)可以取值(-1、0或1)。

[0205] 然后该提示被编解码到比特流中。首先,对有效提示位进行编码,其指示提示是否不为零,然后,如果有效,则对提示的符号进行编码(负为0,并且正为1)。语法如表15所示。差分脉冲编解码调制(DPCM)也可以用来利用提示之间的相关性。一旦提示被解码,解码器

可以以这种方式计算逆变换：

[0206] 等式16：利用4x4沃尔什-哈达玛归一化变换的逆变换

[0207] $$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \frac{1}{4} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} 2 \times c_0 + rem\_c_0 \\ 2 \times c_1 + rem\_c_1 \\ 2 \times c_2 + rem\_c_2 \\ 2 \times c_3 + rem\_c_3 \end{bmatrix}$$

[0208] 这种技术可以用于其他变换尺寸，但是不能总是实现归一化。例如，尺寸为2x2的沃尔什-哈达玛变换矩阵需要除以2的平方根以进行归一化，这总是会导致损失，因为提示部分不仅仅是+-0.5。然而，像4x4的情况中那样除以2是有益的，因为它降低了变换系数的增加的动态性。

[0209] 该技术也适用于哈尔变换，尽管由于哈尔变换矩阵的非均匀范数，该技术不会导致归一化变换。为了说明，解释了4x4哈尔变换。

[0210] 等式17：利用4x4哈尔归一化变换计算的变换系数

[0211] $$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{1}{2} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

[0212] 提示以这种方式计算：

[0213] 等式18：对利用4x4哈尔归一化变换计算的每个系数的除法的提示

[0214] $$\begin{cases} rem\_c_0 = 2 \times c_0 - (r_0 + r_1 + r_2 + r_3) \\ rem\_c_1 = 2 \times c_1 - (r_0 + r_1 - r_2 - r_3) \\ rem\_c_2 = 2 \times c_2 - (r_0 - r_1) \\ rem\_c_3 = 2 \times c_3 - (r_2 - r_3) \end{cases}$$

[0215] 其中提示取值为(0,-1和1)。在解码器侧，解码后的提示以这种方式用于逆变换处理：

[0216] 等式19：利用4x4哈尔归一化变换的逆变换

[0217] $$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \frac{1}{4} \times \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \times \begin{bmatrix} 2 \times c_0 + rem\_c_0 \\ 2 \times c_1 + rem\_c_1 \\ 4 \times c_2 + rem\_c_2 \\ 4 \times c_3 + rem\_c_3 \end{bmatrix}$$

[0218] 通过哈尔、沃尔什-哈达玛和无变换工具，我们可以有多种方法来以无损方式变换残差数据。总的来说，对于二维残差信号的水平和竖直变换，我们有9种选择。下表提供了它们的细节：

[0219] 表14

[0220]

水平变换	竖直变换
沃尔什-哈达玛	沃尔什-哈达玛
沃尔什-哈达玛	哈尔
沃尔什-哈达玛	无变换
哈尔	沃尔什-哈达玛
哈尔	哈尔
哈尔	无变换

[0221]

无变换	沃尔什-哈达玛
无变换	哈尔
无变换	无变换

[0222] 利用这些变换的直接方式是让编码器选择降低速率的最佳变换对,并对所使用的对的索引进行编码,以便解码器可以推导所使用的对并执行逆变换。

[0223] 在VTM中,当前的变换对要么是用于水平和竖直变换的核心DCT2变换,要么是称为多变换选择(MTS)的DST7和DCT8变换的附加集合。MTS可以通过高级标志开启/关闭。为了符合这种设计,水平和竖直两者的默认无损变换都是无变换,而“多变换”是沃尔什-哈达玛变换和哈尔变换。它们也可以由高级标志“Lossless_MTS_Flag”来控制。因此,变换选择表可以修改为:

[0224] 表15

[0225]

Lossless_MTS_Flag 被启用		Lossless_MTS_Flag 被禁用	
水平变换	竖直变换	水平变换	竖直变换
无变换	无变换	无变换	无变换
哈尔	哈尔	-	-
沃尔什-哈达玛	哈尔	-	-
哈尔	沃尔什-哈达玛	-	-
沃尔什-哈达玛	沃尔什-哈达玛	-	-

[0226] 通常,两个非归一化变换很好地拟合了具有小维度的残差。例如,它们可以使用高达8x8或4x4的尺寸。

[0227] 具有二次无损编解码的块级

[0228] 参考图10A,这种情况对应于步骤407。

[0229] 如果CU是有损编解码的,但误差限制在给定阈值以下(通常是如果误差的绝对值小于或等于1,如等式20所示),则可以引入附加编解码阶段来允许无损编解码。误差被测量为原始像素和重构采样之间的差值。

[0230] 等式20:可对其应用二次无损编解码的有损编解码单元的有限误差

[0231] -1≤error≤1

[0232] 例如,如果使用DCT或DST变换,其中量化步长等于1,则可以在有损编解码之后应用二次无损编解码。

[0233] 图15示出了二次无损编解码的简化框图的示例。该处理的输入是比特流。在步骤700中,从比特流中解码数据。特别地,这提供了量化的变换系数、变换类型和新引入的二次系数。在步骤701中,逆量化被应用于量化的变换系数。在步骤702中,逆变换被应用于所得的变换系数。所得的残差在步骤703中被添加到来自步骤704的帧内或帧间预测的预测信号中。结果是重构的块。当在步骤710中选择I_PCM模式时,采样值被直接解码而不需要熵解码。当选择无损编解码模式时,跳过步骤701和702。当选择变换跳过模式时,跳过步骤702。当选择具有二次变换模式的块级时,跳过步骤701,在步骤720应用逆变换以获得第一残差,将二次无损残差添加到第一残差。然后,在步骤703中,将所得的残差和添加到预测信号。

[0234] 由于误差被限制在给定阈值之下,因此用于编解码这个小残差的语法非常简单,基本上,如果阈值为1,则可能只需要有效标志和符号标志来编解码二次残差,语法如表16所示。

[0235] 表16:对于阈值为1的二次残差语法

[0236]

Secondary_residual_coding(x0, y0, log2TbWidth, log2TbHeight) {	**描述符**
for(i = 0; i < (1<<log2TbHeight); i++) {	
for(j = 0; j < (1<<log2TbWidth); j++) {	
secondary_significant_flag[x0 + j][y0 + i]	ae(v)
if(**secondary_significant_flag**[x0 + j][y0 + i]) {	
secondary_sign_flag[x0 + j][y0 + i]	
}	
}	
}	
}	
…	

[0237] 这种二次无损编解码也可以按区域而不是按块来进行应用。

[0238] 区域级信令

[0239] 在无损编解码的用例中,很可能对整个区域无损地编解码,并且不将一些无损编解码块与有损编解码块混合在一起。为了处理这种情况,实际的语法需要为每个CU编解码cu_transquant_bypass_flag,这可能是高成本的。

[0240] 在第一实施例中,我们提出移动该标志,并将其编解码在split_cu_flag之前,以便来自父CU的所有子CU可以共享region_transquant_bypass_flag,相关联的语法如表17所示。

[0241] 表17:region_transquant_bypass_flag的所提出的语法

[0242]

coding_tree(x0, y0, cbWidth, cbHeight, qgOn, cbSubdiv, cqtDepth, mttDepth, depthOffset, partIdx, treeType, *isRegionTransquantCoded*, *isRegionTransquant*) {	**描述符**
if(transquant_bypass_enabled_flag && !isRegionTransquantCoded) {	
region_transquant_bypass_flag	ae(v)
isRegionTransquantCoded = true	
isRegionTransquant = region_transquant_bypass_flag	
}	
if((allowSplitBtVer || allowSplitBtHor || allowSplitTtVer || allowSplitTtHor || allowSplitQT) &&(x0 + cbWidth <= pic_width_in_luma_samples) && (y0 + cbHeight <= pic_height_in_luma_samples))	
split_cu_flag	ae(v)
if(split_cu_flag) {	
….	
coding_tree(x0, y0, cbWidth / 4, cbHeight, qgOn, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, *isRegionTransquant*)	
coding_tree(x1, y0, cbWidth / 2, cbHeight, qgOn, cbSubdiv + 1, cqtDepth, mttDepth+1, depthOffset, 1, treeType, *isRegionTransquant*)	
…..	
} else {	
coding_unit(x0, y0, cbWidth, cbHeight, treeType, *isRegionTransquant*)	
}	

[0243] 图16示出了在使用区域级信令时对region_transquant_bypass_flag和split_cu_flag的解析处理的示例流程图。在解析编解码树单元的语法时,第一步骤(500)是将两个变量初始化为假,其中isRegionTransquantCoded指示用于指示无损编解码的标志是否已针对当前区域被编解码,并且isRegionTransquant指示当前区域是否被无损编解码。然后步骤501检查无损编解码是否在画面级被启用(Transquant_bypass_enabled_flag为真)并且region_transquant_bypass_flag还未针对当前区域被编解码(isRegionTransquantCoded为假)。如果这些条件为真,则在步骤502中解析标志region_transquant_bypass_flag。否则,步骤504解析split_cu_flag。在步骤502之后,步骤503检查region_transquant_bypass_flag是否为真。如果region_transquant_bypass_flag为真,则在步骤504中将变量isRegionTransquant设置为真。然后在步骤504中解析split_cu_flag。如果split_cu_flag为真,则处理返回到步骤500,否则树不再被划分,并且处理结束(步骤505)。可以解析当前编解码单元的语法。

[0244] 双树情况下的无损编解码

[0245] 图17示出了在双树情况下对亮度和色度的不同分割。在VVC中,称为双树的编解码结构允许对帧内条带的亮度树和色度树单独进行编码。

[0246] 在一实施例中,单独为每个树解析cu_transquant_bypass_flag,这允许更灵活地用信号通知无损编解码。

[0247] 在变型实施例中,cu_transquant_bypass_flag不是为色度树编解码的,而是从亮度树推断的。对于给定的色度CU,如果同位亮度CU中的一个被无损编解码,则当前色度CU也被无损编解码。该图示出了亮度树和色度树的不同分割,例如,如果亮度CU L4被无损编解码,则色度CU C3和C4被无损编解码。在另一示例中,如果亮度CU L3被无损编解码,则色度CU C2也被无损编解码,即使同位的亮度CU L2不被无损编解码。

[0248] 本申请描述了各种方面,包括工具、特征、实施例、模型、方法等。这些方面中的多个方面被具体地描述,并且至少为了显示各自特性而通常被以听起来可能是限制性的方式进行描述。然而,这是出于描述清楚的目的,而并不限制那些方面的应用或范围。事实上,所有不同的方面都可以被组合和互换以提供进一步的方面。此外,这些方面也可以与更早的文件中描述的各方面相结合和互换。

[0249] 本申请中描述和预期的方面可以以多种不同的形式实现。图1A、1B和2提供了一些实施例,但是可以预期其他实施例,并且这些图的讨论并不限制实现方式的广度。这些方面中的至少一个通常涉及视频编码和解码,并且至少一个其他方面通常涉及发送生成或编码的比特流。这些和其他方面可以被实现为方法、装置、其上存储有用于根据所述方法中的任何一个来对视频数据进行编码或解码的指令的计算机可读存储介质、和/或其上存储有根据所述方法中的任何一个生成的比特流的计算机可读存储介质。

[0250] 本文描述了各种方法,并且这些方法中的每一种都包括用于实现所述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,否则特定步骤和/或动作的顺序和/或使用可以被修改或组合。

[0251] 本申请中描述的各种方法和其他方面可用于修改如图1A和图1B所示的视频编码器100和解码器200的模块,例如运动补偿和运动估计模块(170、175、275)。此外,本方面不限于VVC或HEVC,并且可以应用于例如其他标准和推荐,无论是现有的还是未来开发的,以及任何这种标准和推荐(包括VVC和HEVC)的扩展。除非另有说明,或者技术上被排除,否则本申请中描述的各方面可以单独使用或者组合使用。

[0252] 在本申请中使用了各种数值。这些特定值是出于示例的目的,并且所描述的各方面不限于这些特定值。

[0253] 各种实现方式涉及解码。如本申请中使用的“解码”可以包含例如对接收的编码序列执行的全部或部分处理,以便产生适于显示的最终输出。在各种实施例中,这种处理包括通常由解码器执行的处理中的一个或多个。在各种实施例中,这种处理也包括或者可替代地包括由本申请中描述的各种实现方式的解码器执行的处理。

[0254] 作为进一步的示例,在一个实施例中,“解码”仅指代熵解码,在另一实施例中,“解码”仅指代差分解码,而在另一实施例中,“解码”指代熵解码和差分解码的组合。基于具体描述的上下文,短语“解码处理”旨在具体指代操作的子集还是泛指更广泛的解码处理将会是清楚的,并且被认为是由本领域技术人员完全理解的。

[0255] 各种实现方式涉及编码。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”可以包含例如对输入视频序列执行的全部或部分处理,以便产生编码的比特流。在各种实施例中,这种处理包括通常由编码器执行的处理中的一个或多个。在各种实施例中,这种处理也包括或者可替代地包括由本申请中描述的各种实现方式的编码器执行的处理。

[0256] 作为进一步的示例,在一个实施例中,“编码”仅指代熵编码,在另一实施例中,“编码”仅指代差分编码,而在另一实施例中,“编码”指代差分编码和熵编码的组合。基于具体描述的上下文,短语“编码处理”旨在具体指代操作的子集还是泛指更广泛的编码处理将会是清楚的,并且被认为是由本领域技术人员完全理解的。

[0257] 注意,如本文使用的语法元素是描述性术语。因此,它们不排除对其他语法元素名称的使用。

[0258] 当图被呈现为流程图时,应当理解其也提供了对应的装置的框图。类似地,当图被呈现为框图时,应当理解其也提供了对应的方法/处理的流程图。

[0259] 各种实施例涉及速率失真优化。特别是,在编码处理期间,通常会考虑速率和失真之间的平衡或权衡,通常会给定计算复杂性的限制。速率失真优化通常被公式化为最小化速率失真函数,其是速率和失真的加权和。存在不同的方法来解决速率失真优化问题。例如,这些方法可以基于对所有编码选项的广泛测试,包括所有考虑的模式或编解码参数值,以及对它们的编解码成本与编解码和解码后重构的信号的相关失真的完整评估。还可以使用更快的方法来节省编码复杂性,特别是基于预测或预测残差信号而不是重构信号来计算近似失真。也可以使用这两种方法的混合,诸如通过只对可能的编码选项中的一些使用近似失真,而对其他编码选项使用完全失真。其他方法只评估可能的编码选项的子集。更一般地,许多方法采用多种技术中的任何一种来执行优化,但是优化不一定是对编解码成本和相关失真的完整评估。

[0260] 本申请描述了各种方面,包括工具、特征、实施例、模型、方法等。这些方面中的多个方面被具体地描述,并且至少为了显示各自特性而通常被以听起来可能是限制性的方式进行描述。然而,这是出于描述清楚的目的,而并不限制那些方面的应用或范围。事实上,所有不同的方面都可以被组合和互换以提供进一步的方面。此外,这些方面也可以与更早的文件中描述的各方面相结合和互换。

[0261] 本文描述的实现方式和各方面可以例如以方法或处理、装置、软件程序、数据流、或信号来实现。即使仅在单一形式的实现方式的上下文中进行讨论(例如,仅作为方法讨论),所讨论的特征的实现方式也可以其他形式(例如,装置或程序)来实现。装置可以例如以适当的硬件、软件和固件来实现。所述方法可以在例如处理器中实现,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、或可编程逻辑器件。处理器还包括通信设备,诸如,例如计算机、平板计算机、智能手机、手机、便携式/个人数字助理、和有助于终端用户之间的信息通信的其他设备。

[0262] 对“一个实施例”或“实施例”或“一种实现方式”或“实现方式”、及其其他变型的引用意味着结合实施例所描述的特定特征、结构、特性等等被包括在至少一个实施例中。因此,贯穿本申请在各个地方出现的短语“在一个实施例中”或“在实施例中”或“在一种实现方式中”或“在实现方式中”、以及任何其他变型的出现并不一定全都指代相同的实施例。

[0263] 此外,本申请可指代“确定”各条信息。确定信息可以包括例如估计信息、计算信息、预测信息、或从存储器中检索信息中的一个或多个。

[0264] 此外,本申请可指代“访问”各条信息。访问信息可以包括例如接收信息、(例如,从存储器中)检索信息、存储信息、移动信息、复制信息、计算信息、确定信息、预测信息、或估计信息中的一个或多个。

[0265] 此外,本申请可指代“接收”各条信息。与“访问”一样,接收意欲为广义的术语。接收信息可以包括例如访问信息、或(例如,从存储器中)检索信息中的一个或多个。此外,通常在诸如例如存储信息、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,以一种方式或另一方式涉及“接收”。

[0266] 在本申请中,术语“重构”和“解码”可以互换使用,术语“像素”和“采样”可以互换使用,术语“图像”、“画面”、“帧”、“条带”和“图块”可以互换使用。通常,但不是必须的,术语“重构”被用于编码器侧,而“解码”被用于解码器侧。

[0267] 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,对以下“/”、“和/或”和“…中的至少一个”中的任何一个的使用意欲包含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)。作为另一示例,在“A、B、和/或C”和“A、B、和C中的至少一个”的情况下,这种措辞意欲包含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),或者选择所有三个选项(A和B和C)。正如本领域和相关领域的普通技术人员所清楚的那样,这可以扩展到所列出的尽可能多的项目。

[0268] 此外,如本文所使用的,“信号(signal)”一词除了其他以外是指向对应解码器指示某物。例如,在某些实施例中,编码器用信号通知照明补偿参数中的特定一个。以这种方式,在实施例中,在编码器侧和解码器侧均使用相同的参数。因此,例如,编码器可以向解码器发送(显式信令)特定参数,使得解码器可以使用相同的特定参数。相反,如果解码器已经具有特定参数以及其他参数,则可以使用信令而不发送(隐式信令)来简单地允许解码器知道并选择特定参数。通过避免任何实际功能的传输,在各种实施例中实现了比特节省。应当理解,信令可以以多种方式完成。例如,在各种实施例中,一个或多个语法元素、标志等被用于向对应的解码器用信号通知信息。虽然前面提到了“信号”一词的动词形式,但是“信号”一词在本文也可以用作名词。

[0269] 对于本领域普通技术人员来说将显而易见的是,各实现方式可以产生各种信号,这些信号被格式化以携带可被例如存储或发送的信息。该信息可以包括,例如用于执行方法的指令、或由所述实现方式中的一种所产生的数据。例如,信号可以被格式化以携带所述实施例的比特流。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对数据流进行编码并用经编码的数据流来调制载波。信号携带的信息可以是,例如模拟或数字信息。如已知的,信号可以通过各种不同的有线或无线链路来发送。信号可以被存储在处理器可读介质上。

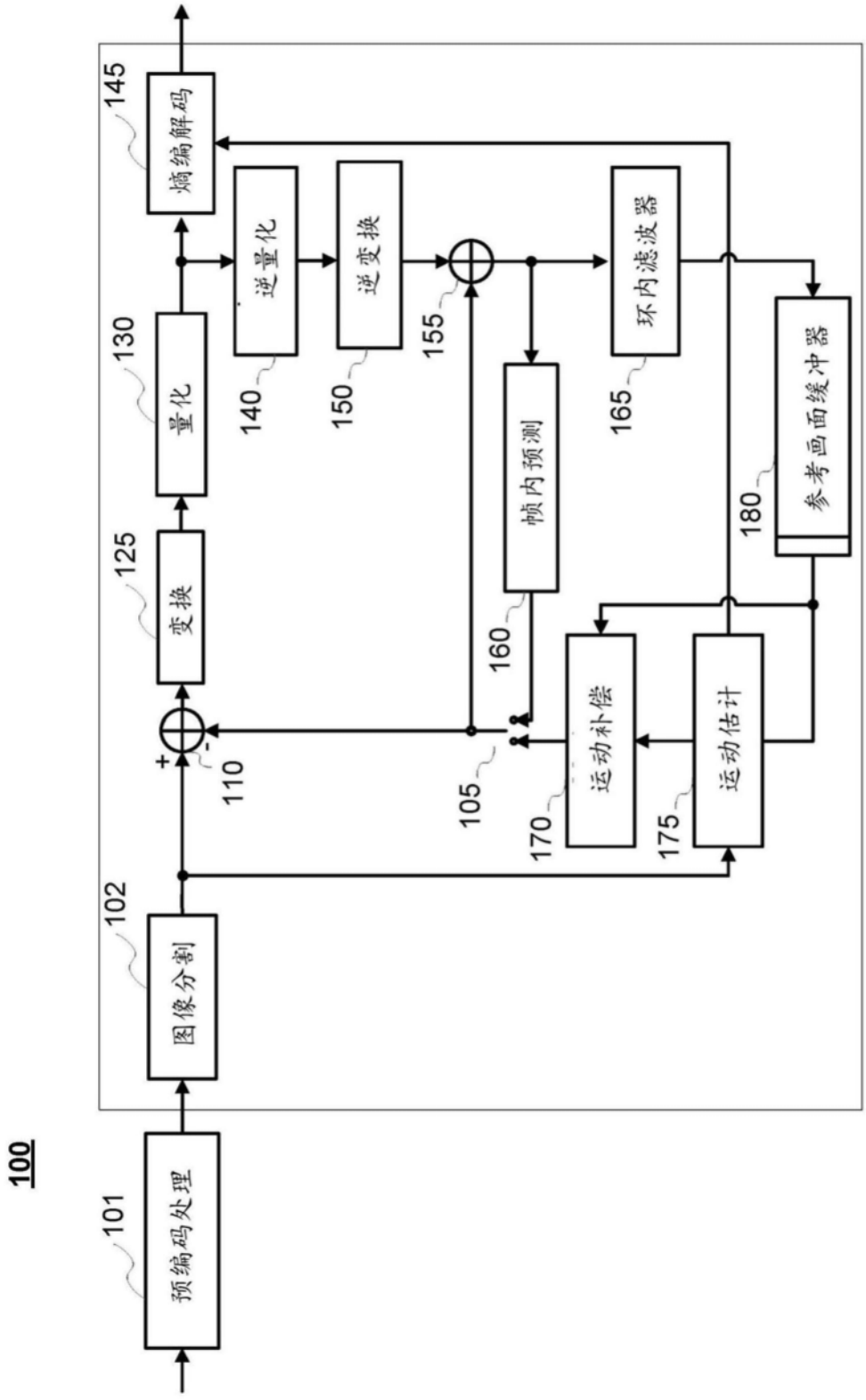

100
101
预编码处理
102
图像分割
110
+
-
125
变换
130
量化
145
熵编解码
140
逆量化
150
逆变换
155
105
160
帧内预测
165
环内滤波器
180
参考画面缓冲器
170
运动补偿
175
运动估计

图1A

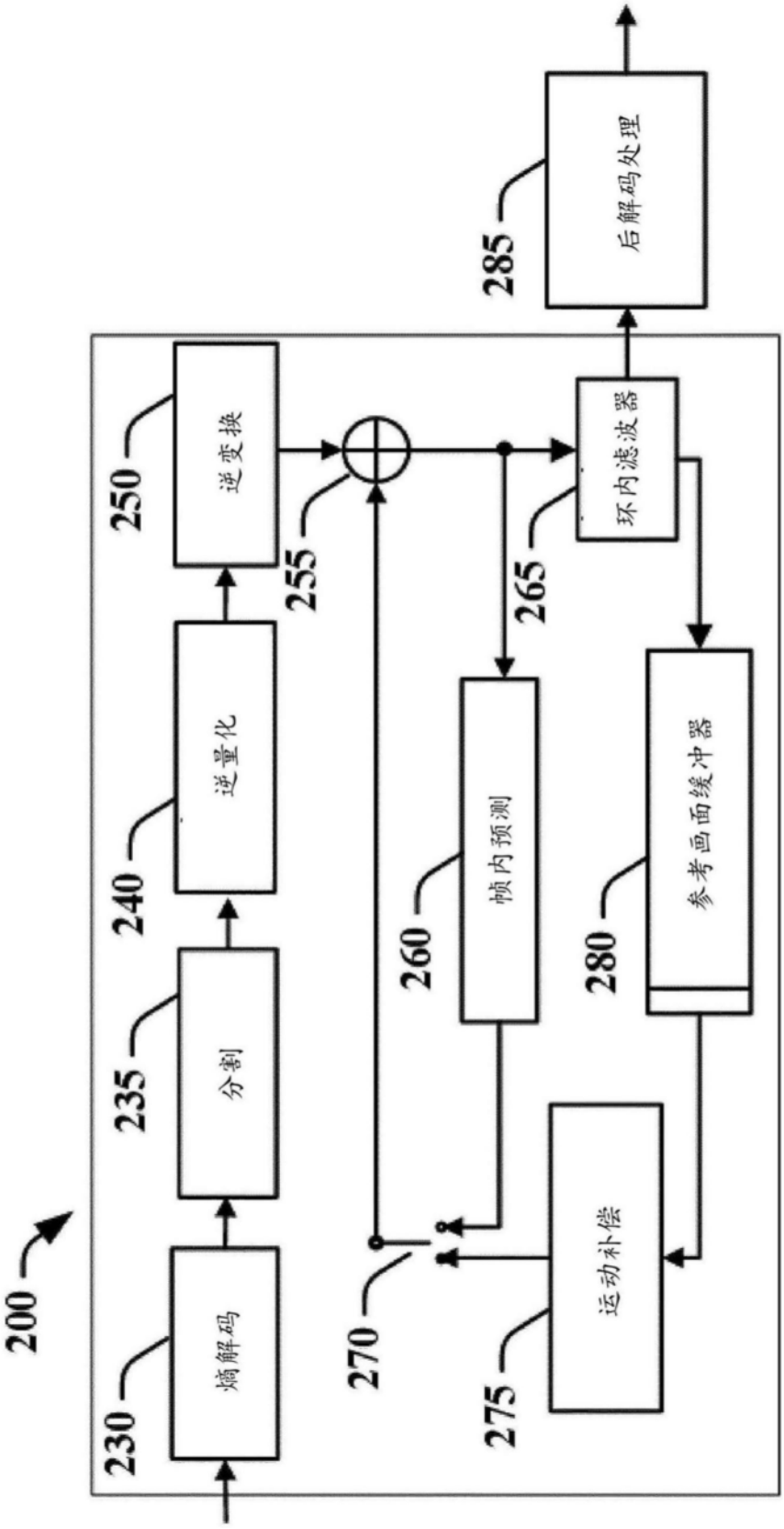

200
230
熵解码
235
分割
240
逆量化
250
逆变换
255
260
帧内预测
265
环内滤波器
270
275
运动补偿
280
参考画面缓冲器
285
后解码处理

图1B

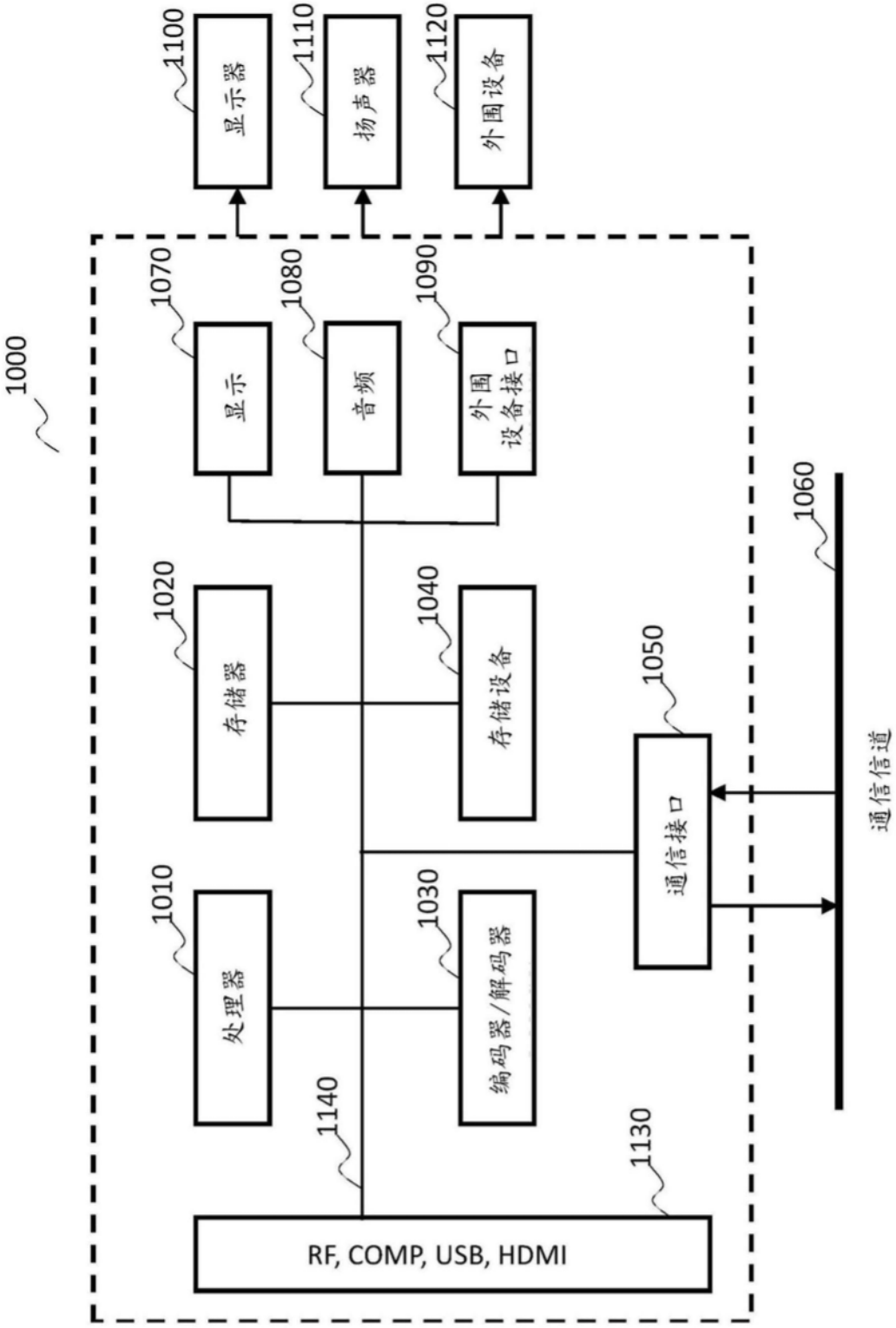

1000
1100
显示器
1110
扬声器
1120
外围设备
1070
显示
1080
音频
1090
外围设备接口
1020
存储器
1040
存储设备
1050
通信接口
1060
通信信道
1010
处理器
1030
编码器/解码器
1140
1130
RF, COMP, USB, HDMI

图2

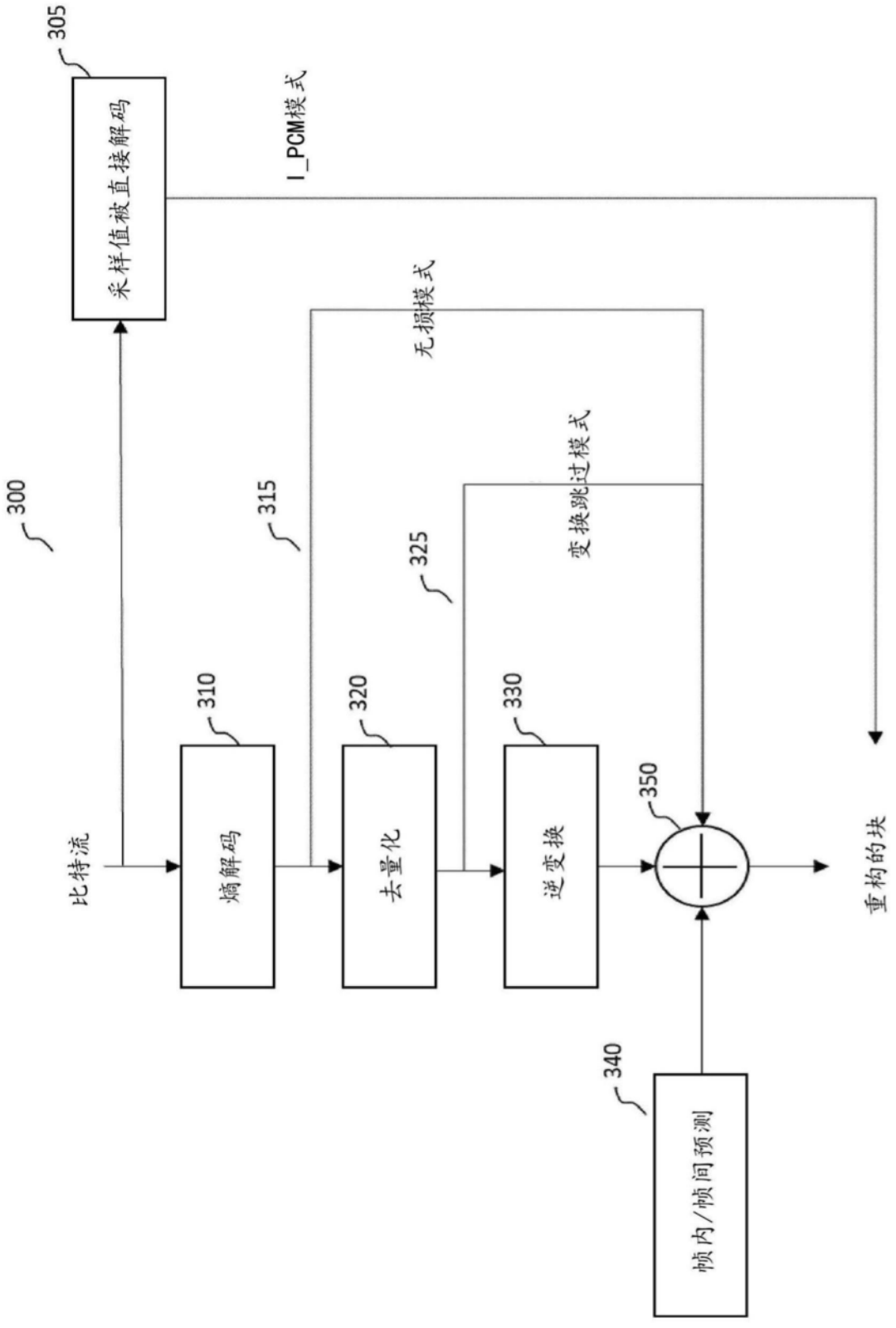

305
采样值被直接解码
I_PCM模式
无损模式
变换跳过模式
300
315
325
310
320
330
350
比特流
熵解码
去量化
逆变换
重构的块
340
帧内/帧间预测

图3

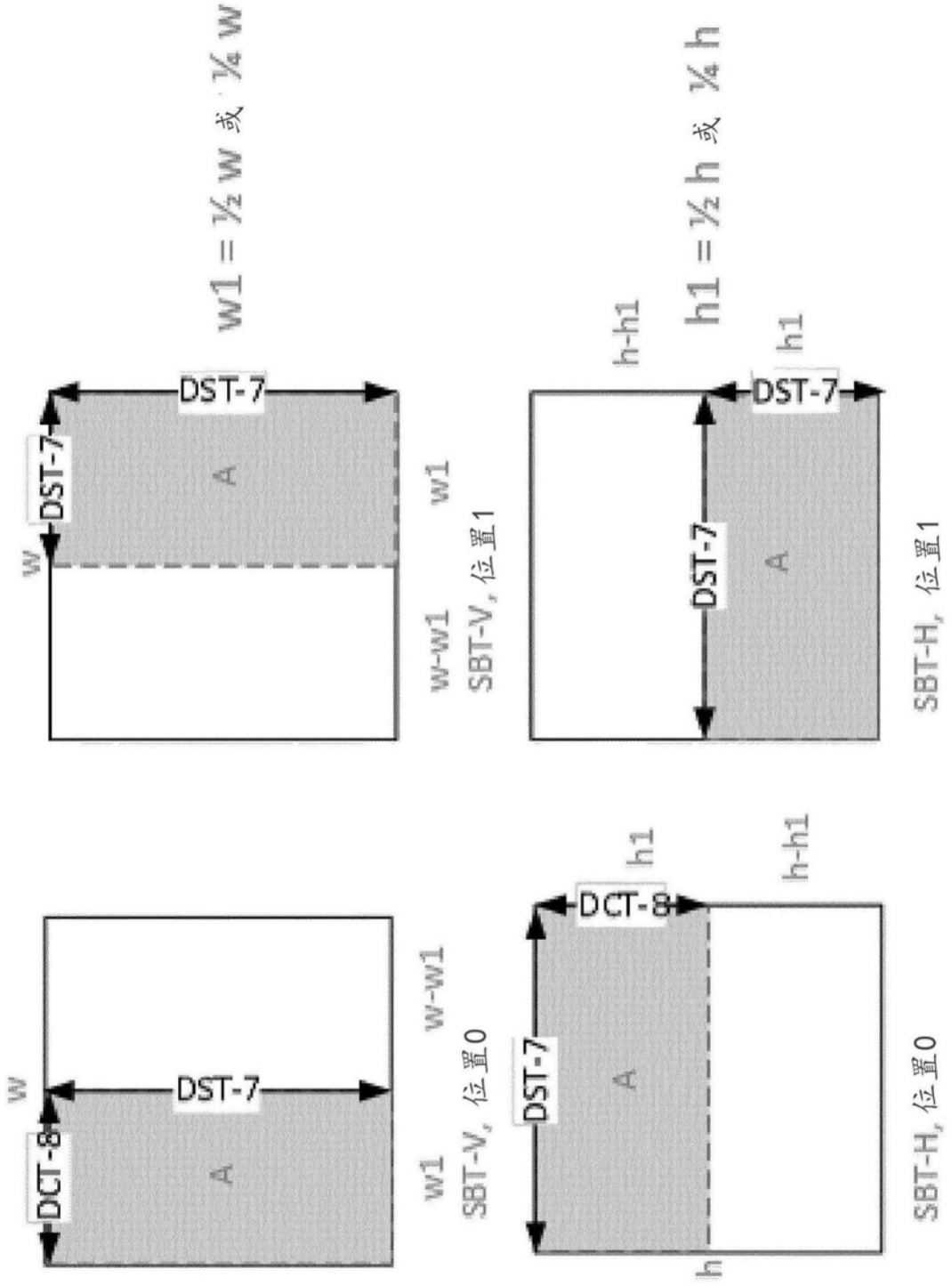

w1 = ½ w 或 ¼ w
DST-7
DST-7
A
w
w1
w-w1
SBT-V，位置1
h-h1
h1 = ½ h 或 ¼ h
h1
DST-7
DST-7
A
SBT-H，位置1
DCT-8
DST-7
A
w
w1
w-w1
SBT-V，位置0
h1
h-h1
DCT-8
DST-7
A
h
SBT-H，位置0

图4

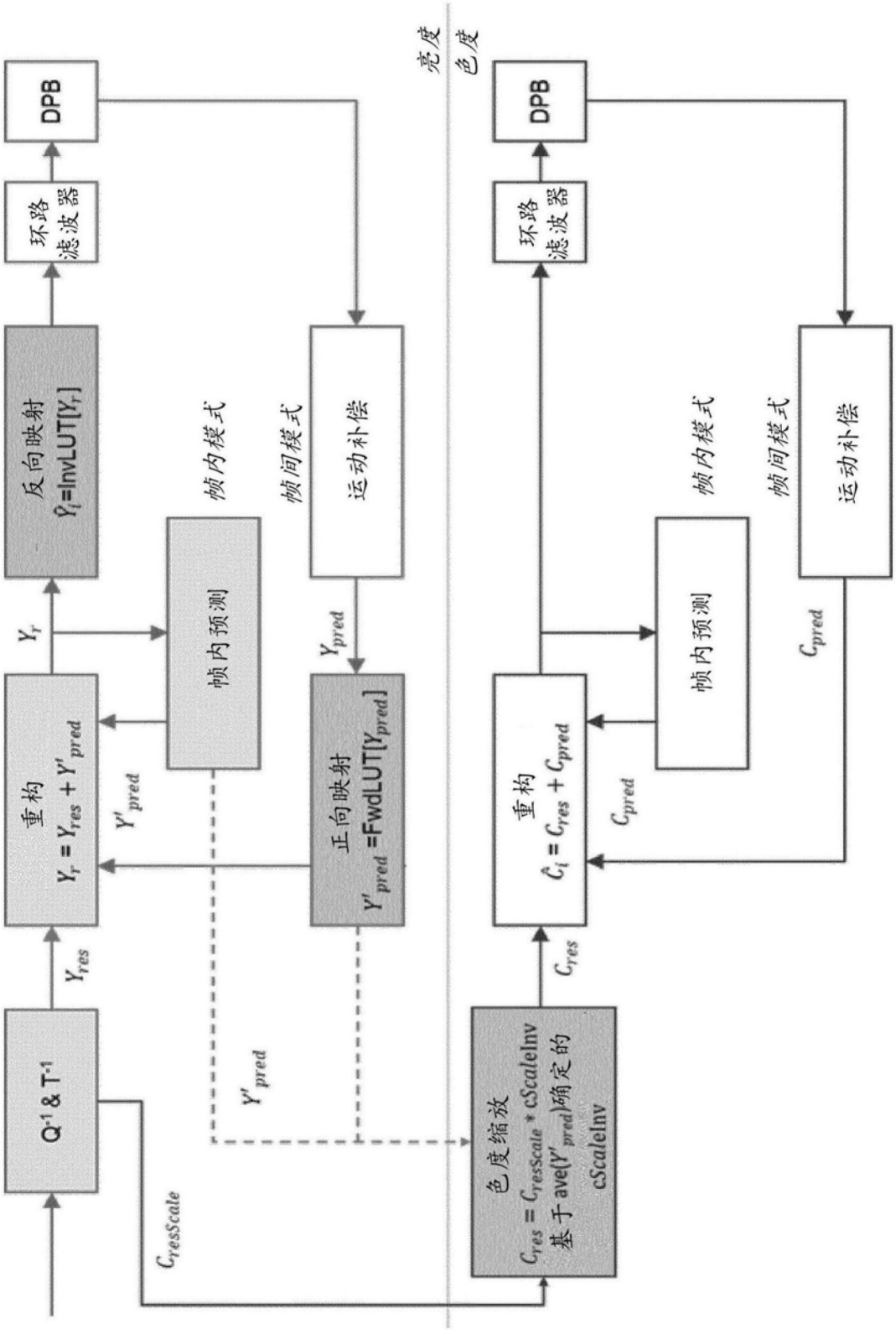

亮度
色度
DPB
环路滤波器
反向映射
$\hat{Y}_I$=InvLUT[Y_I]
重构
$Y_r = Y_{res} + Y'_{pred}$
Y_r
Y_{res}
Q^{-1} & T^{-1}
$C_{resScale}$
帧内模式
帧内预测
Y'_{pred}
帧间模式
运动补偿
Y_{pred}
正向映射
Y'_{pred} =FwdLUT[Y_{pred}]
Y'_{pred}
色度缩放
$C_{res} = C_{resScale} * cScaleInv$
基于 ave(Y'_{pred})确定的
cScaleInv
C_{res}
重构
$\hat{C}_I = C_{res} + C_{pred}$
C_{pred}
帧内预测
帧内模式
帧间模式
运动补偿
C_{pred}
DPB
环路滤波器

图5

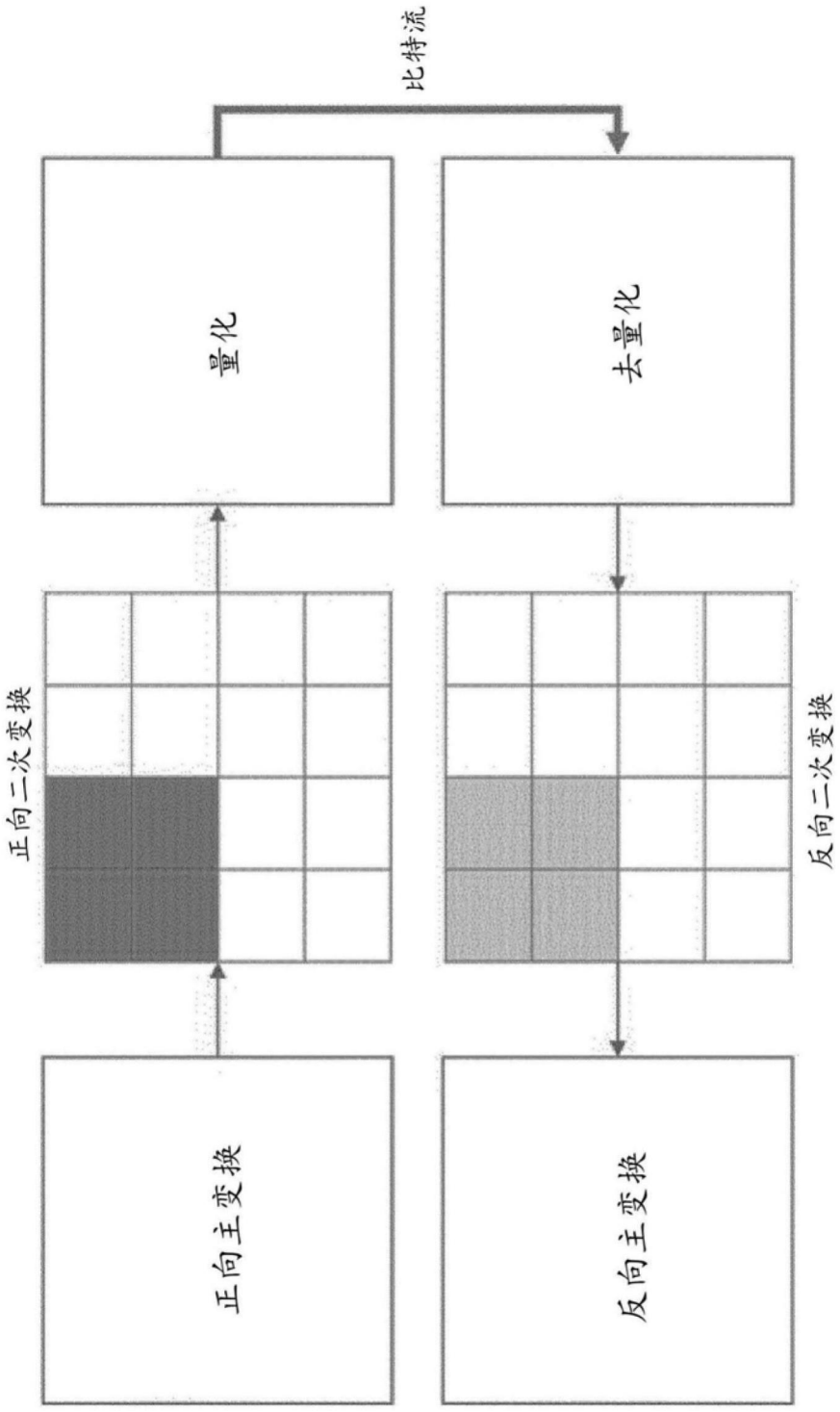


图6

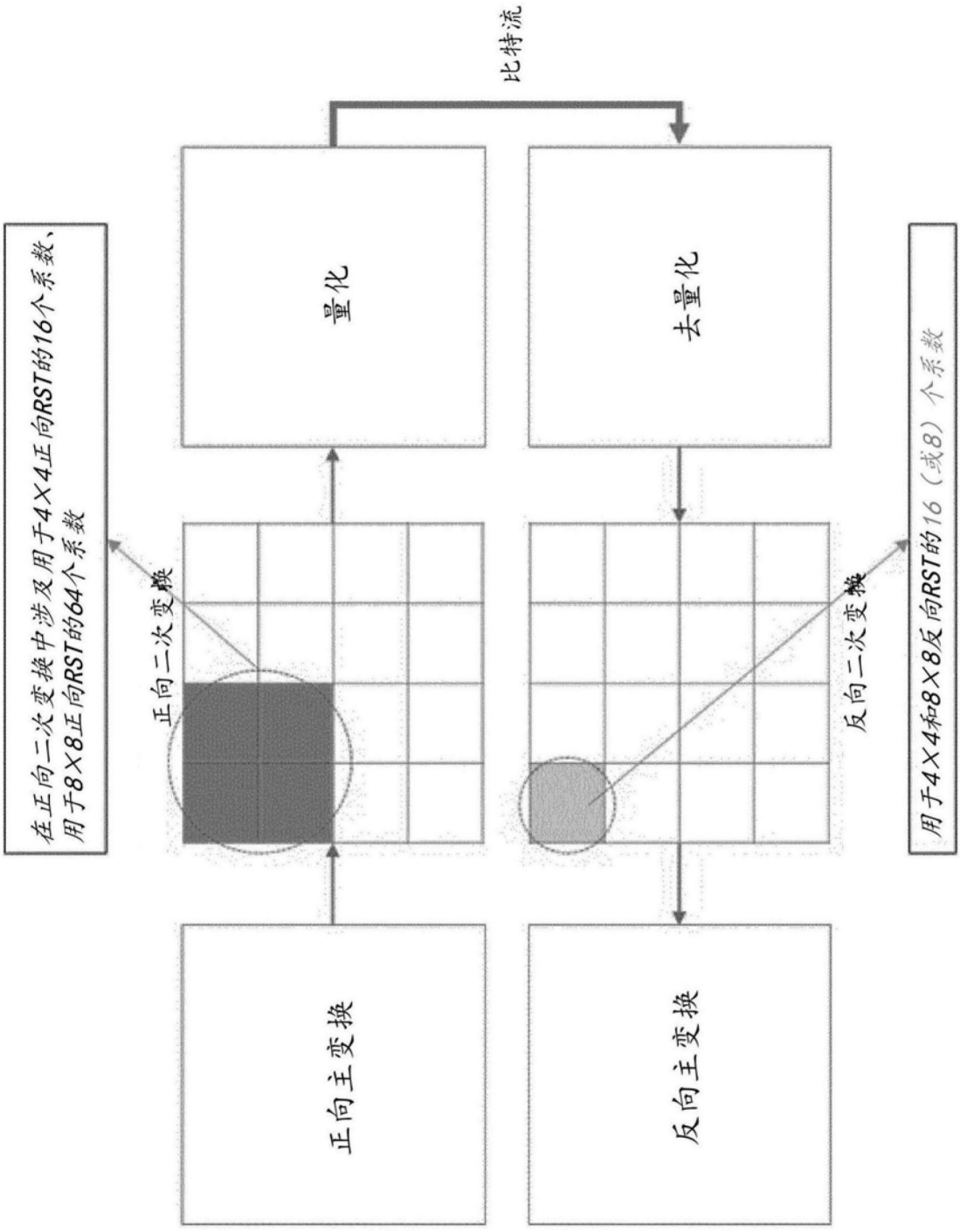

比特流
量化
去量化
正向主变换
反向主变换
正向二次变换
反向二次变换
在正向二次变换中涉及用于4×4正向RST的16个系数、用于8×8正向RST的64个系数
用于4×4和8×8反向RST的16（或8）个系数

图7

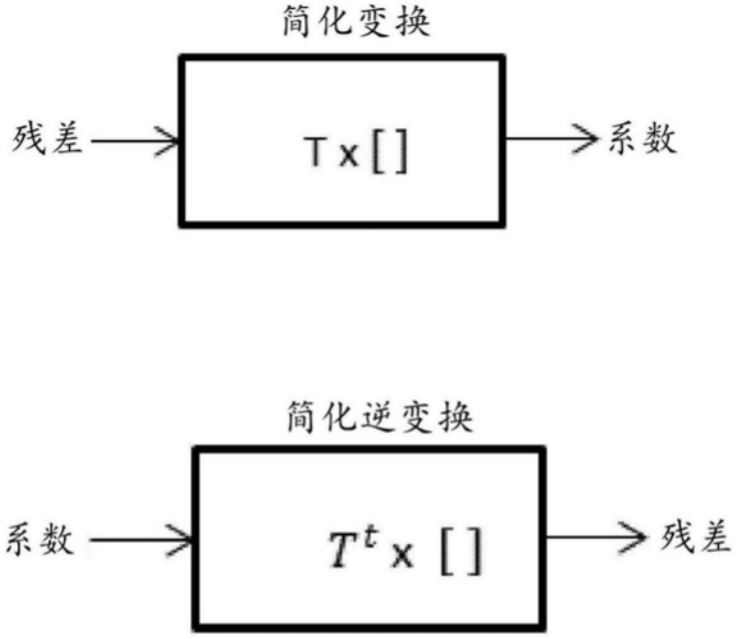


图8

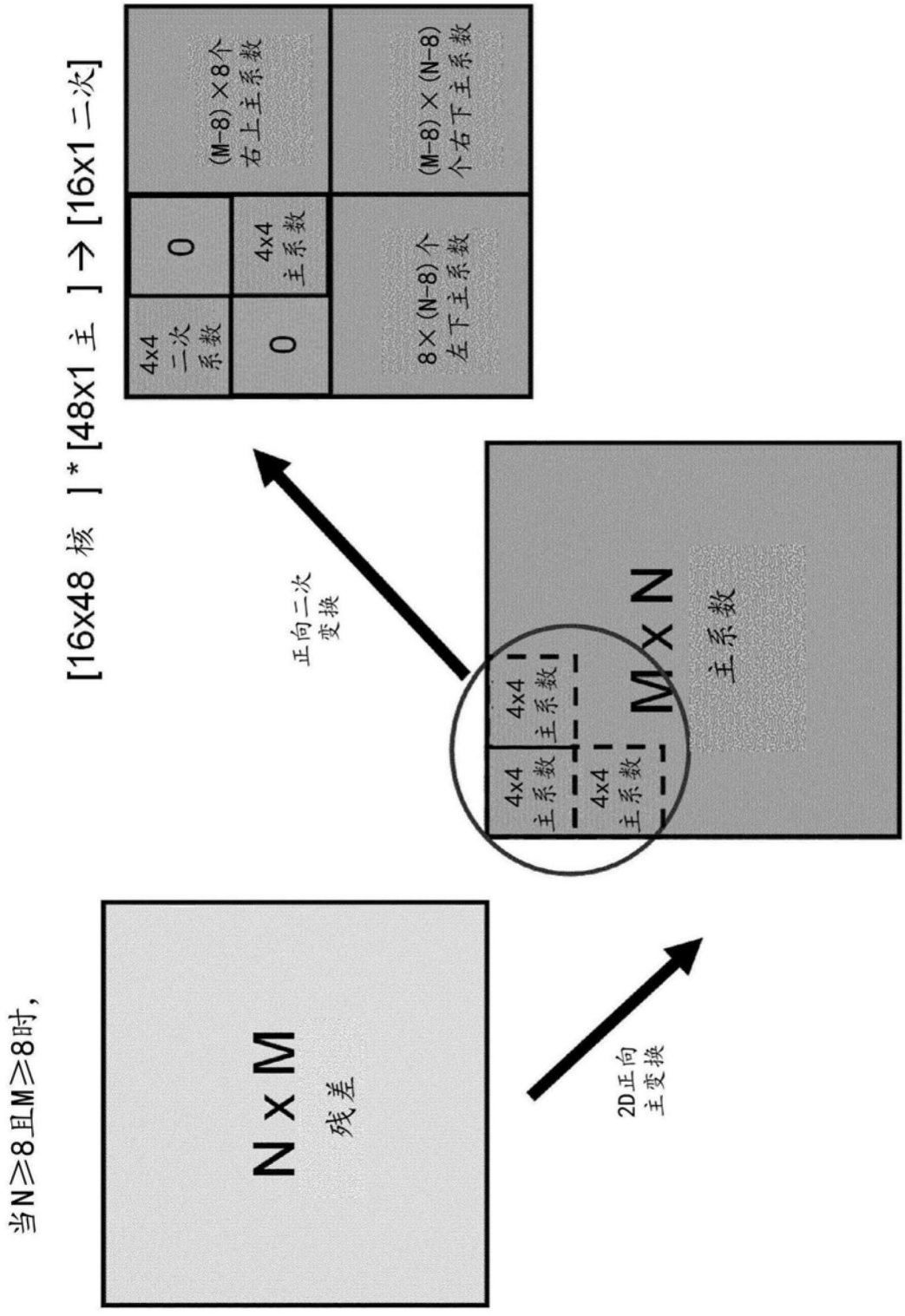

当N≥8且M≥8时，
N x M
残差
2D正向
主变换
M x N
主系数
4x4
主系数
4x4
主系数
4x4
主系数
正向二次
变换
[16x48 核] * [48x1 主] → [16x1 二次]
4x4
二次
系数
0
4x4
主系数
0
8×(N-8)个
左下主系数
(M-8)×8个
右上主系数
(M-8)×(N-8)
个右下主系数

图9

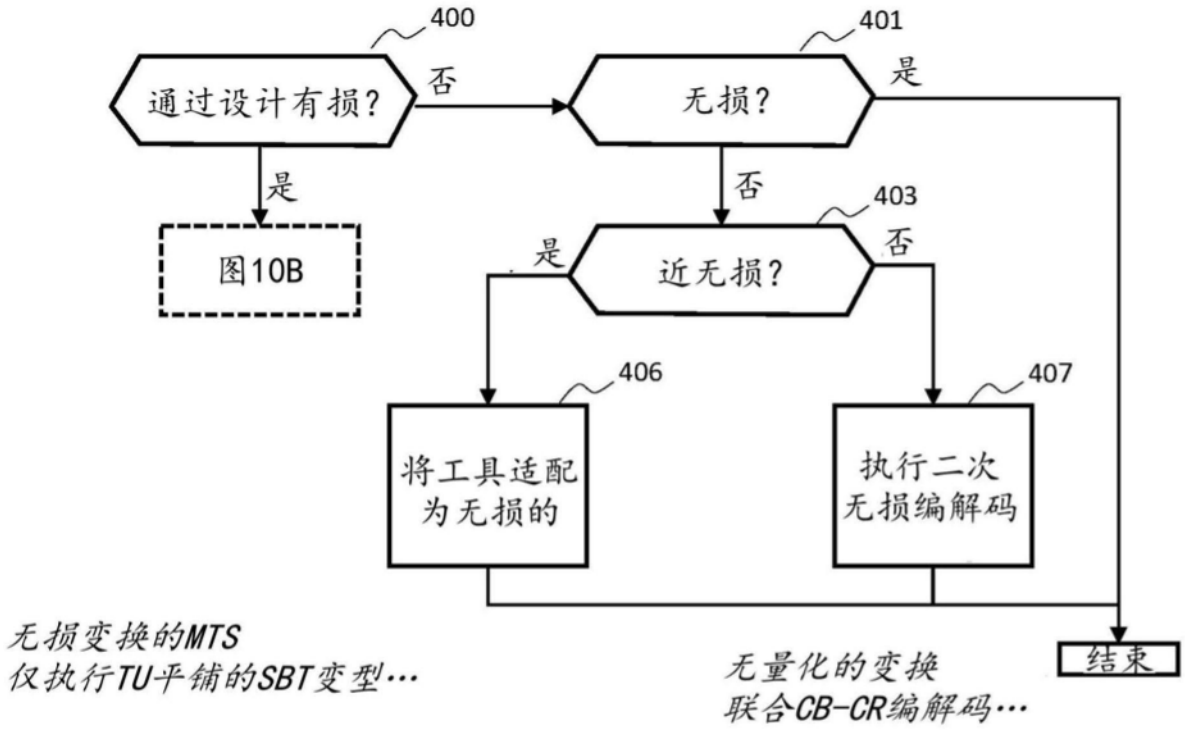

400
通过设计有损?
否
是
401
无损?
是
否
图10B
403
近无损?
是
否
406
将工具适配为无损的
407
执行二次无损编解码
结束
无损变换的MTS
仅执行TU平铺的SBT变型…
无量化的变换
联合CB-CR编解码…

图10A

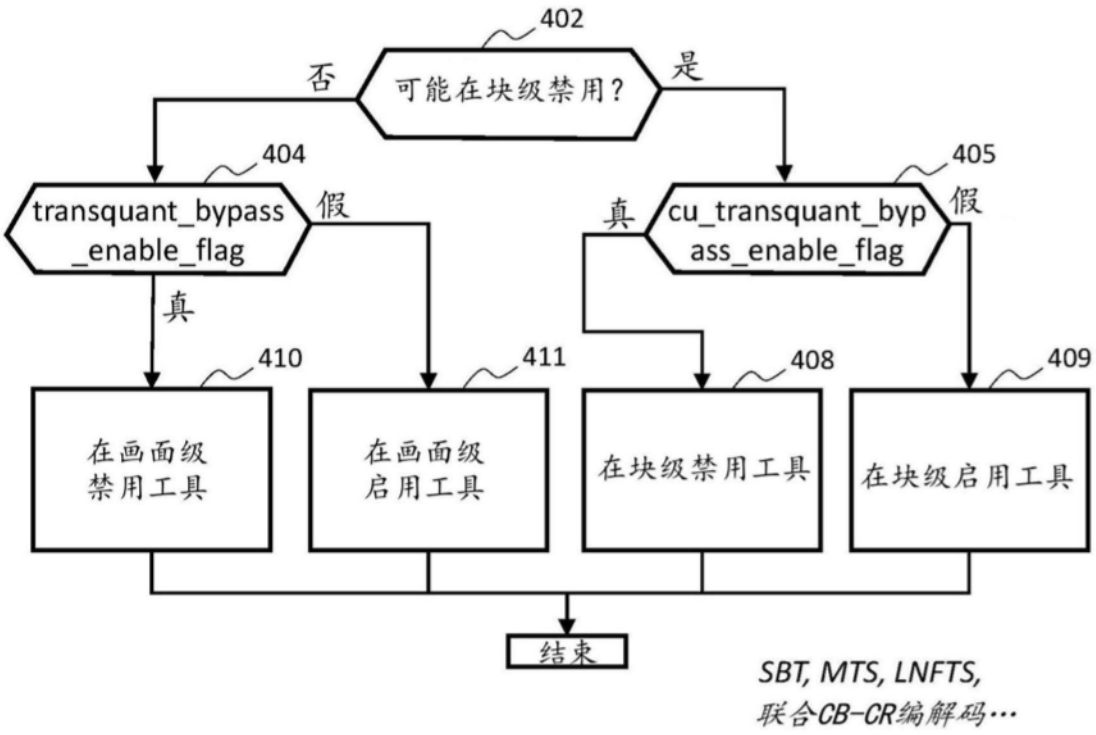

402
可能在块级禁用?
否
是
404
transquant_bypass_enable_flag
假
真
405
cu_transquant_bypass_enable_flag
真
假
410
在画面级禁用工具
411
在画面级启用工具
408
在块级禁用工具
409
在块级启用工具
结束
SBT, MTS, LNFTS,
联合CB-CR编解码…

图10B

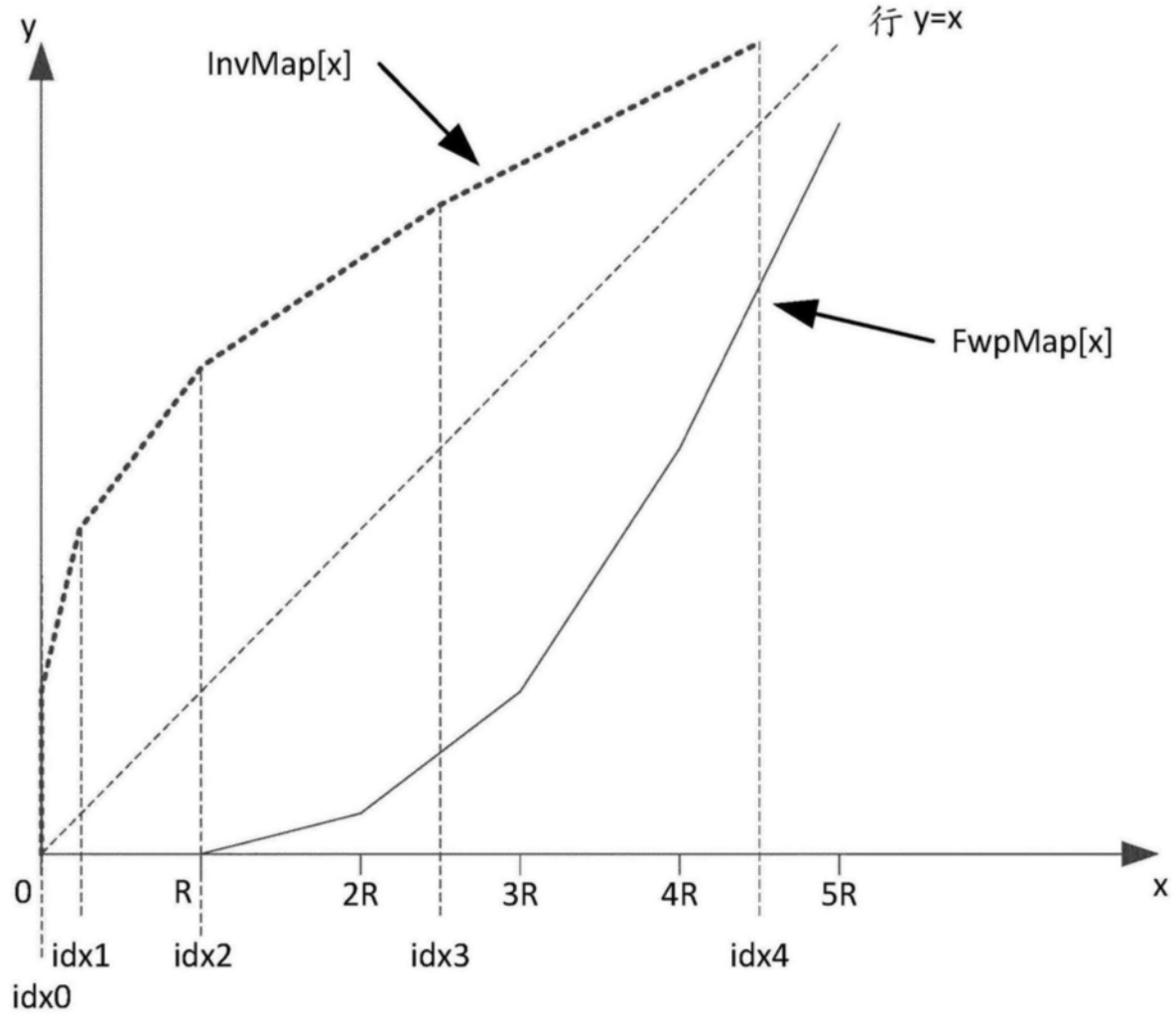


图11

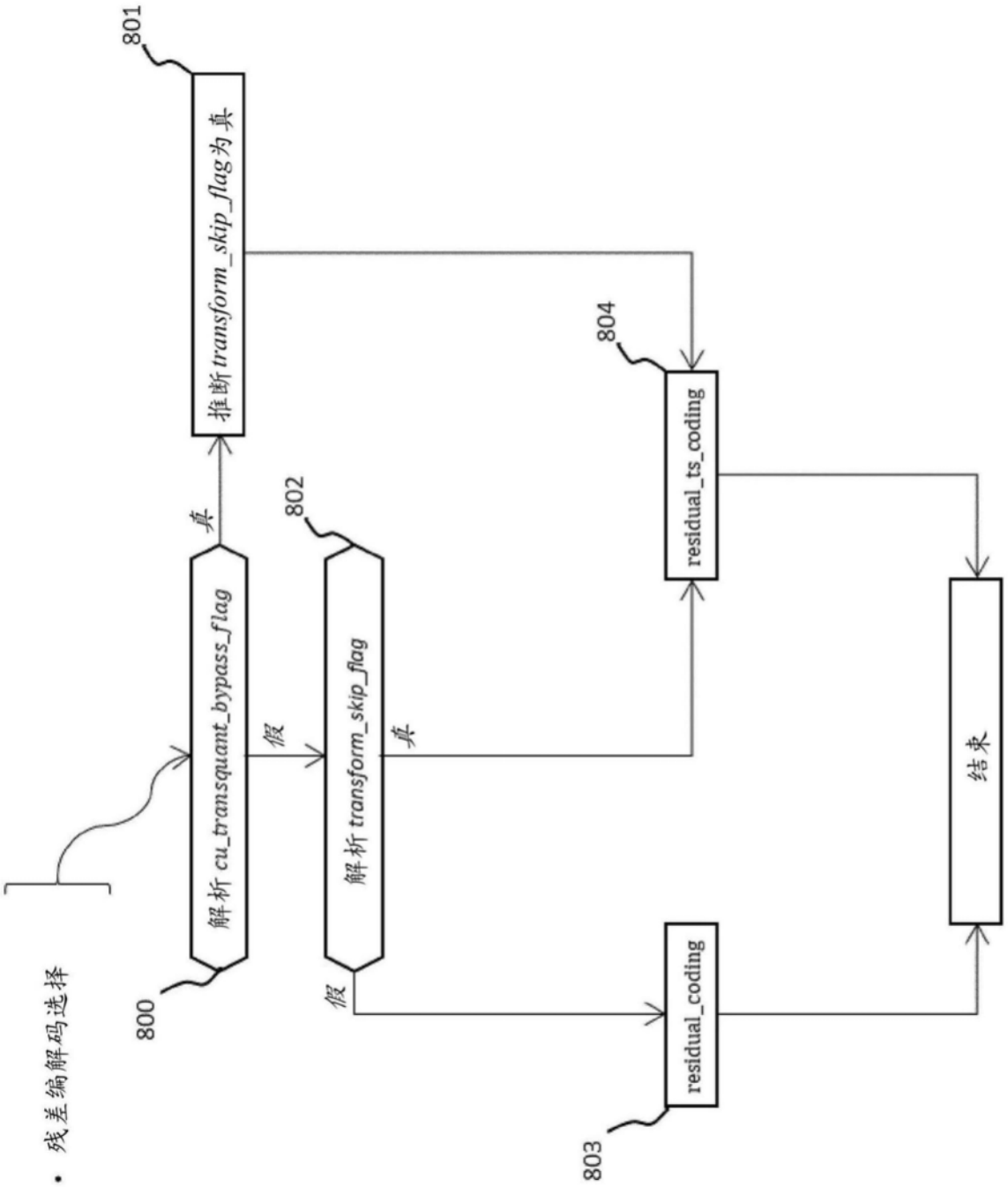

801
推断 transform_skip_flag为 真
804
residual_ts_coding
802
真
解析 cu_transquant_bypass_flag
假
解析 transform_skip_flag
真
结束
假
800
residual_coding
803
• 残差编解码选择

图12

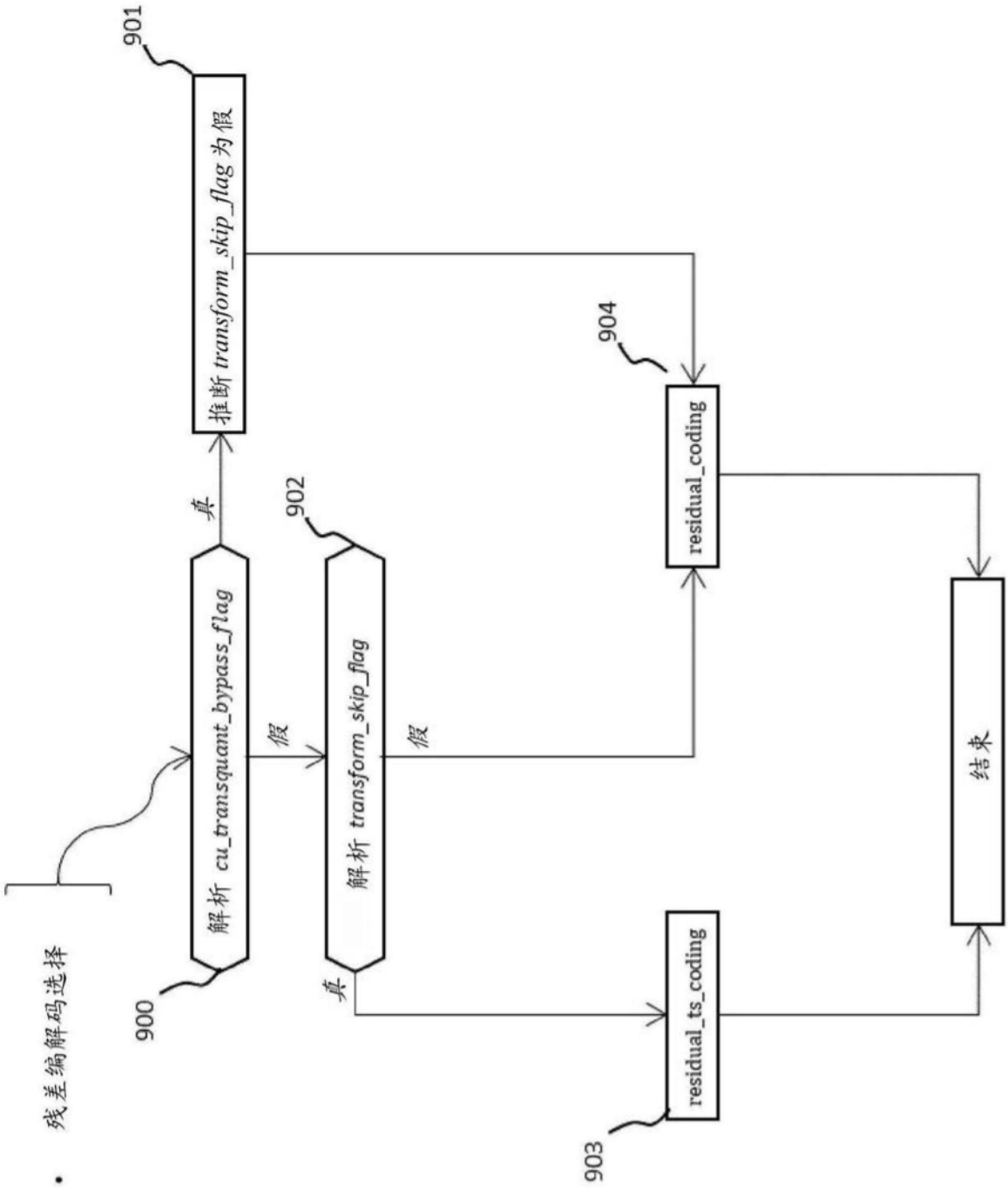

残差编解码选择
解析 cu_transquant_bypass_flag
900
真
假
推断 transform_skip_flag 为假
901
解析 transform_skip_flag
902
真
假
residual_ts_coding
903
residual_coding
904
结束

图13

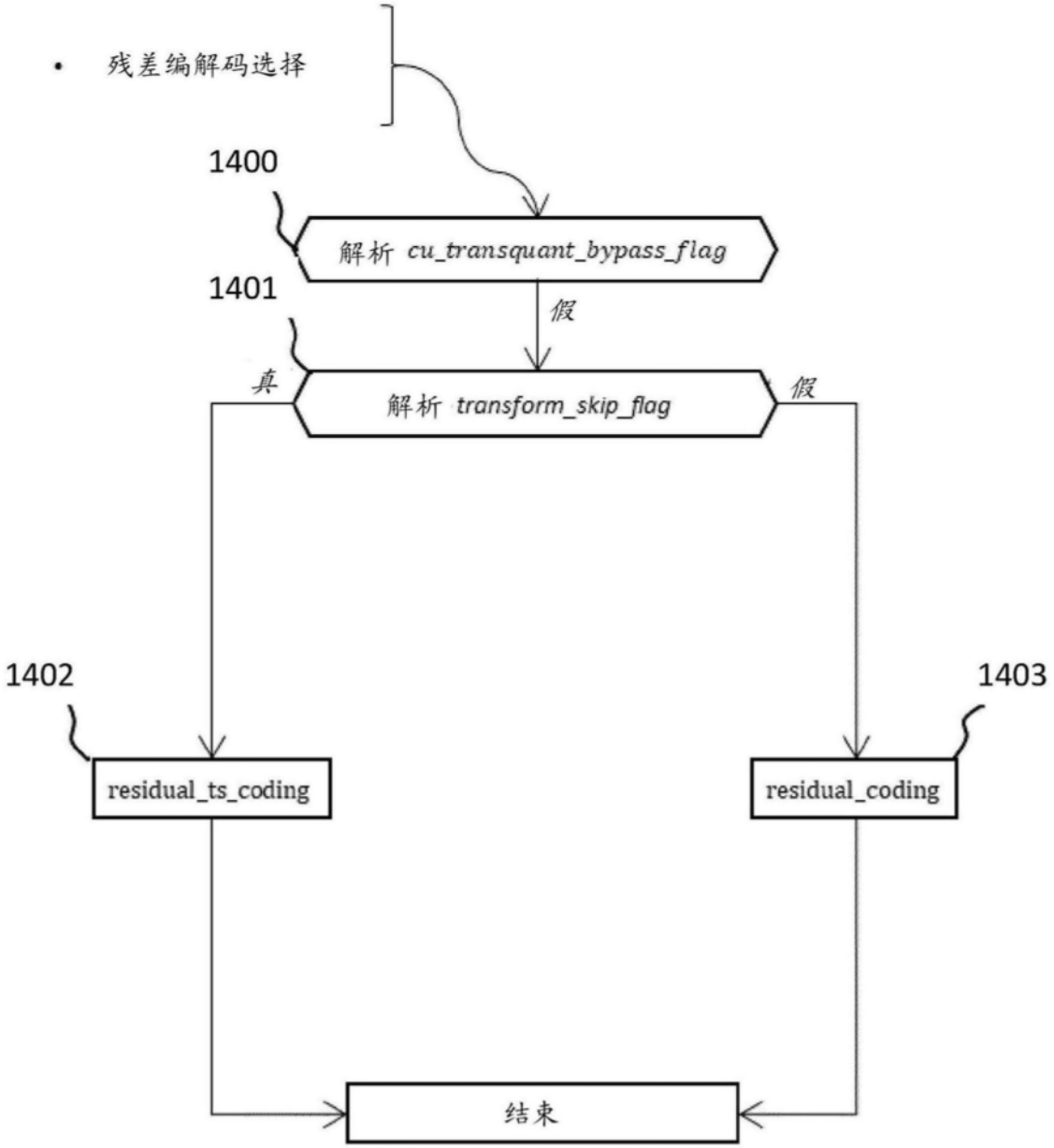

• 残差编解码选择
1400
解析 cu_transquant_bypass_flag
1401
假
真
解析 transform_skip_flag
假
1402
1403
residual_ts_coding
residual_coding
结束

图14

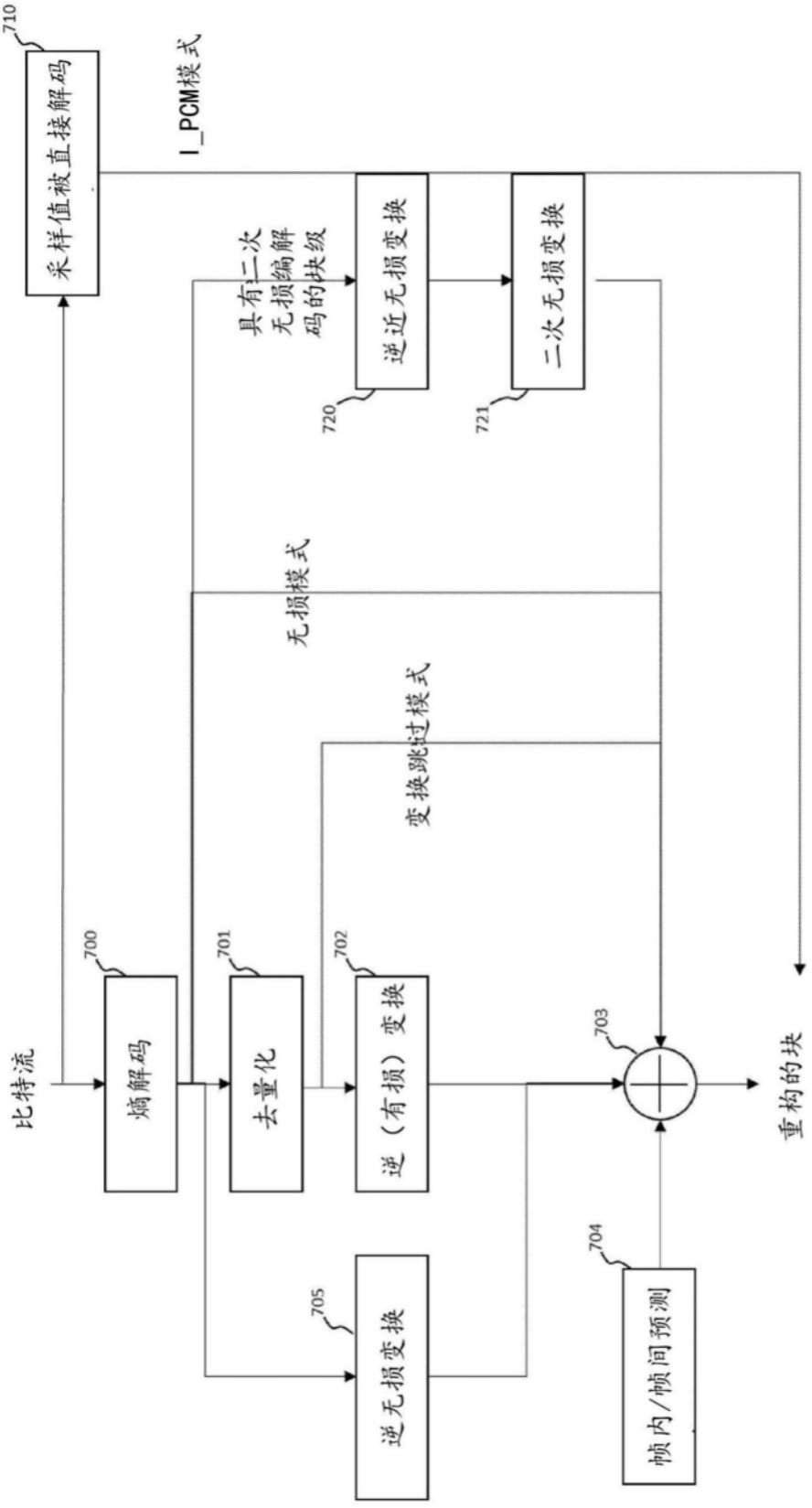

比特流
熵解码
去量化
逆（有损）变换
逆无损变换
帧内/帧间预测
重构的块
采样值被直接解码
I_PCM模式
具有二次无损编码的块级
逆近无损变换
二次无损变换
无损模式
变换跳过模式
700
701
702
703
704
705
710
720
721

图15

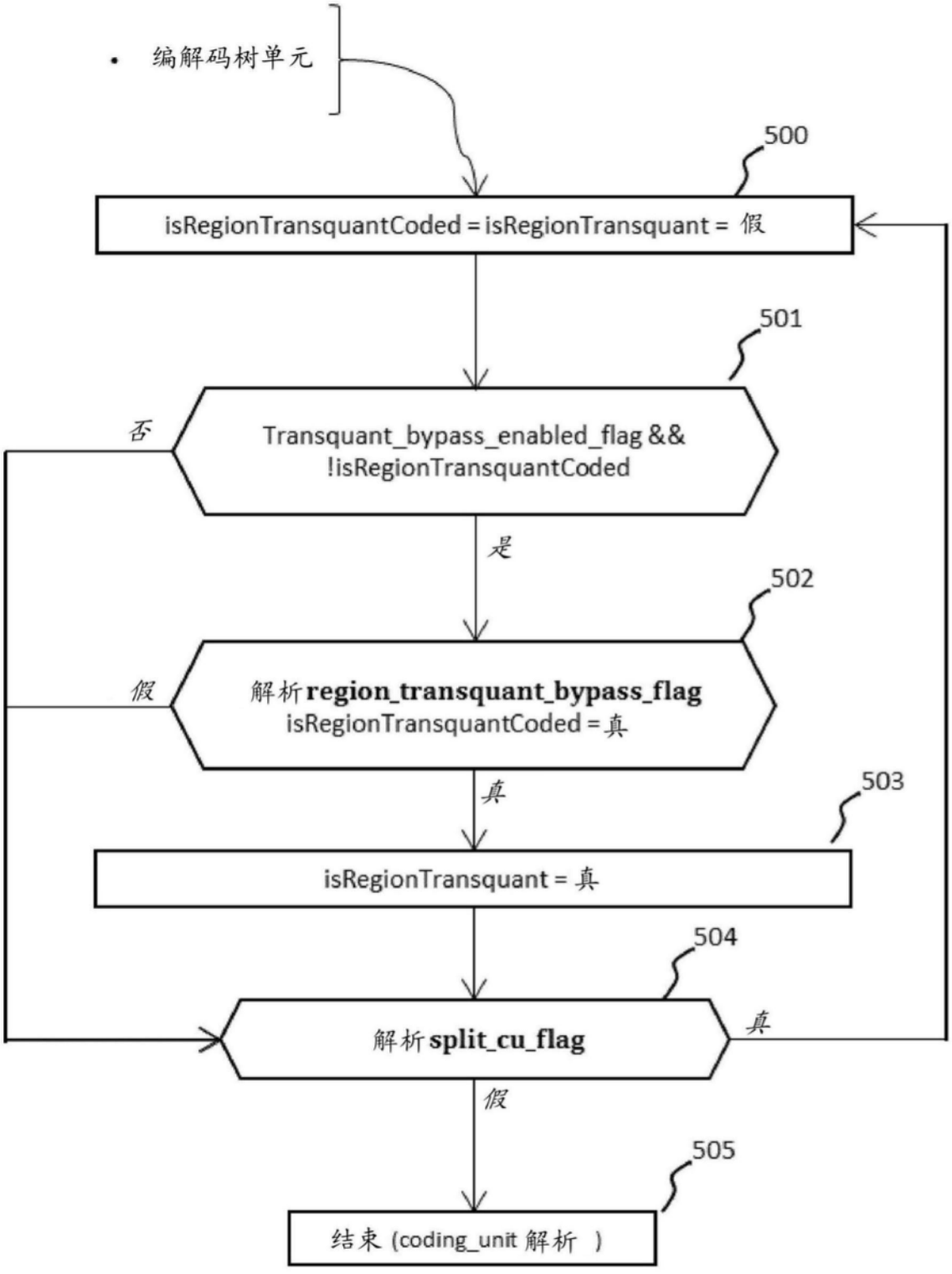

编解码树单元
500
isRegionTransquantCoded = isRegionTransquant = 假
501
否
Transquant_bypass_enabled_flag &&
!isRegionTransquantCoded
是
502
假
解析 region_transquant_bypass_flag
isRegionTransquantCoded = 真
真
503
isRegionTransquant = 真
504
解析 split_cu_flag
真
假
505
结束 (coding_unit 解析)

图16

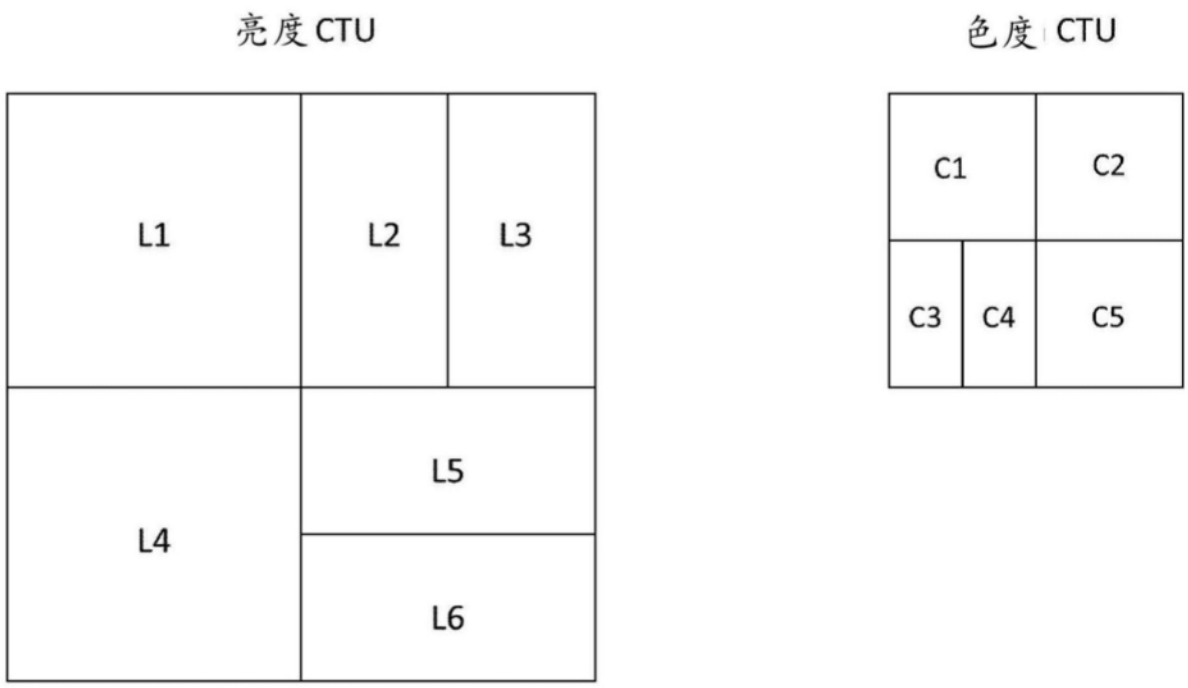


图17