(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112740682 B
(45)授权公告日 2024.08.27

(21)申请号 201980061637.2
(22)申请日 2019.09.13
(65)同一申请的已公布的文献号
申请公布号 CN 112740682 A
(43)申请公布日 2021.04.30
(30)优先权数据
18290103.3 2018.09.21 EP
19305641.3 2019.05.20 EP
(85)PCT国际申请进入国家阶段日
2021.03.19
(86)PCT国际申请的申请数据
PCT/US2019/051089 2019.09.13
(87)PCT国际申请的公布数据
WO2020/060867 EN 2020.03.26
(73)专利权人 交互数字VC控股公司
地址 美国特拉华州
(72)发明人 Y.陈 F.莱林内克 T.波伊里尔 P.德拉格兰治
(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 19/124(2006.01)
H04N 19/136(2006.01)
H04N 19/18(2006.01)
H04N 19/192(2006.01)
H04N 19/60(2006.01)
H04N 19/91(2006.01)
(56)对比文件
Heiko Schwarz.CE7: Transform Coefficient Coding and Dependent Quantization (Tests 7.1.2, 7.2.1).Joint Video Experts Team (JVET) of ITU-T SG 16WP 3 and ISO/IEC JTC 1/SC 29/WG 11 11th Meeting: Ljubljana, SI, 10-18 July, 2018 Document: JVET-K0071.2018,全文. (续)
审查员 刘喆

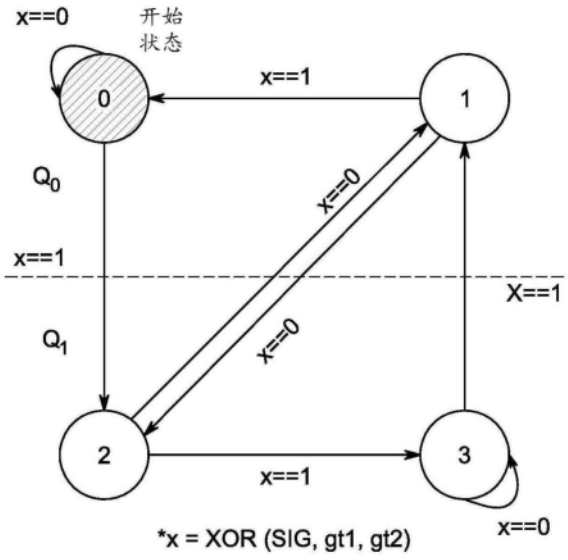
权利要求书3页 说明书24页 附图11页

(54)发明名称

用于依赖性标量量化的标量量化器决策方案

(57)摘要

当使用依赖性的标量量化时,量化器的选择取决于先前变换系数的解码,而变换系数的熵解码取决于量化器选择。为了在用于变换系数熵编码的硬件实现中维持高吞吐量,提出了标量量化器的几种决策方案。在一种实现中,状态转换和上下文模型选择仅基于常规编码的位元(二进制数)。例如,状态转换可以基于SIG、gt1和gt2标志之和,SIG、gt1和gt2标志的异或函数,或者仅基于gt1或gt2标志。当对变换系数块进行编码时,可以先在一个或者多个扫描遍中对常规模式位元进行编码,然后在另一个或者多个扫描遍中将其余的旁路编码的位元分组在一起。



CN 112740682 B

[转续页]

[接上页]

(56)对比文件

Jianle Chen.Algorithm description for Versatile Video Coding and Test Model 4 (VTM 4).JVET-M1002-v2.2019,全文.

1.一种视频解码方法,包括:

对与图片的块中的第一变换系数相关联的第一组参数进行熵解码,所述第一变换系数在解码顺序中在所述图片的所述块中的第二变换系数之前;

对与所述第二变换系数相关联的第二组参数进行熵解码,所述第一和第二组参数在算术编码引擎的常规模式中被熵编码,

其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数的解码,

其中,在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵解码,

其中,所述第一扫描遍是在用于对所述块的变换系数进行熵解码的一个或多个其它扫描遍之前执行的,

其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括至少SIG标志、第一标志和第二标志,所述SIG标志指示相对应的变换系数是否为零,所述第一标志指示所述相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2,并且

其中,基于与所述第一变换系数相关联的所述第一组参数的所述SIG标志、所述第一标志和第二标志的求和或者异或函数,在两个或更多个解量化器之间选择用于解量化所述第二变换系数的解量化器;以及

响应于经解码的变换系数重构所述块。

2.根据权利要求1所述的视频解码方法,其中,基于所述第一变换系数在两个或者更多量化器之间选择用于对所述第二变换系数进行解量化的解量化器。

3.根据权利要求2所述的视频解码方法,其中,基于用于所述第一变换系数的所述第一组参数选择所述解量化器。

4.根据权利要求2所述的视频解码方法,其中,基于所述第一标志、所述第二标志或者所述第一组参数的另一个标志选择所述解量化器,所述另一个标志指示所述相对应的变换系数的所述绝对值是否大于x。

5.根据权利要求4所述的视频解码方法,其中,所述SIG、第一、第二标志或者另一个标志的上下文建模基于所述解量化器或者所述解量化器选择中使用的状态。

6.一种用于视频解码的装置,包括一个或多个处理器,其中所述一个或多个处理器被配置为:

对与图片的块中的第一变换系数相关联的第一组参数进行熵解码,所述第一变换系数在解码顺序中在所述图片的所述块中的第二变换系数之前;

对与所述第二变换系数相关联的第二组参数进行熵解码,所述第一和第二组参数在算术编码引擎的常规模式中被熵编码,

其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数的解码,

其中,在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵解码,

其中,所述第一扫描遍是在用于对所述块的变换系数进行熵解码的一个或多个其它扫描遍之前执行的,

其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括至少SIG标志、

第一标志和第二标志,所述SIG标志指示相对应的变换系数是否为零,所述第一标志指示所述相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2,并且

其中,基于与所述第一变换系数相关联的所述第一组参数的所述SIG标志、所述第一标志和第二标志的求和或者异或函数,在两个或更多个解量化器之间选择用于解量化所述第二变换系数的解量化器;以及

响应于经解码的变换系数重构所述块。

7.根据权利要求6所述的用于视频解码的装置,其中,基于所述第一变换系数在两个或更多量化器之间选择用于对所述第二变换系数进行解量化的解量化器。

8.根据权利要求7所述的用于视频解码的装置,其中,基于所述第一变换系数的所述第一组参数选择所述解量化器。

9.根据权利要求7所述的用于视频解码的装置,其中,基于所述第一标志、所述第二标志或者所述第一组参数的另一个标志选择所述解量化器,所述另一个标志指示所述相对应的变换系数的所述绝对值是否大于x。

10.根据权利要求7所述的用于视频解码的装置,其中,所述SIG、第一、第二标志或者另一个标志的上下文建模基于所述解量化器或者在所述解量化器选择中使用的状态。

11.一种视频编码方法,包括:

访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在编码顺序中在所述图片的所述块中的第二变换系数之前;

访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在算术编码引擎的常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及

在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵编码,

其中,所述第一扫描遍是在用于对所述块的变换系数进行熵编码的一个或多个其它扫描遍之前执行的,

其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括至少SIG标志、第一标志和第二标志,所述SIG标志指示相对应的变换系数是否为零,所述第一标志指示所述相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2,并且

其中,基于与所述第一变换系数相关联的所述第一组参数的所述SIG标志、所述第一标志和第二标志的求和或者异或函数,在两个或更多个解量化器之间选择用于解量化所述第二变换系数的解量化器。

12.根据权利要求11所述的视频编码方法,其中,用于对所述第二变换系数进行量化的量化器基于所述第一变换系数。

13.根据权利要求11所述的视频编码方法,其中,所述SIG、第一或第二标志的上下文建模基于所述量化器或者在所述量化器选择中使用的状态。

14.一种用于视频编码的装置,包括一个或多个处理器,其中,所述一个或多个处理器被配置为:

访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在编码顺

序中在所述图片的所述块中的第二变换系数之前;

访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在算术编码引擎的常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及

在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵编码,

其中,所述第一扫描遍是在用于对所述块的变换系数进行熵编码的一个或多个其它扫描遍之前执行的,

其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括至少SIG标志、第一标志和第二标志,所述SIG标志指示相对应的变换系数是否为零,所述第一标志指示所述相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2,并且

其中,基于与所述第一变换系数相关联的所述第一组参数的所述SIG标志、所述第一标志和第二标志的求和或者异或函数,在两个或更多个解量化器之间选择用于解量化所述第二变换系数的解量化器。

15.根据权利要求14所述的用于视频编码的装置,其中,用于对所述第二变换系数进行量化的量化器基于所述第一变换系数。

16.根据权利要求14所述的用于视频编码的装置,其中,所述SIG、第一或第二标志的上下文建模基于所述量化器或者在所述量化器选择中使用的状态。

用于依赖性标量量化的标量量化器决策方案

技术领域

[0001] 本发明实施例通常涉及视频编码或者解码的方法和装置。

背景技术

[0002] 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。通常,帧内或帧间预测被用来利用帧内或帧间图片相关性,然后,对原始块和预测块之差(经常表示为预测误差或者预测残差)进行变换、量化和熵编码。为了重构该视频,压缩的数据由与该熵编码、量化、变换和预测相对应的逆过程来解码。

发明内容

[0003] 根据一个实施例,提供了一种视频解码方法,包括:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在解码顺序中在所述图片的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数的解码;在针对所述块的所述第一扫描遍中对所述第一和第二组参数进行熵解码,其中,所述第一扫描遍是在用于对所述块的变换系数进行熵解码的一个或多个其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括第一标志和第二标志中的至少一个,所述第一标志指示相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2;以及响应于所述经解码的变换系数重构所述块。

[0004] 根据一个实施例,提供了一种视频编码的方法,包括:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在编码顺序中在所述图片的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及在针对所述块的所述第一扫描遍中对所述第一和第二组参数进行熵编码,其中,所述第一扫描遍是在用于对所述块的变换系数进行熵编码的一个或多个其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括第一标志和第二标志中的至少一个,所述第一标志指示相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2。

[0005] 根据另一个实施例,提供了一种用于视频解码的装置,包括一个或多个处理器,其中,所述一个或多个处理器被配置为:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在解码顺序中在所述图片的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数的解码;在针对所述块的所述第一扫描遍中对所述第一和第二组参数

进行熵解码,其中,所述第一扫描遍是在用于对所述块的变换系数进行熵解码的一个或多个其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括第一标志和第二标志中的至少一个,所述第一标志指示相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2;以及响应于所述经解码的变换系数重构所述块。该装置还可以包括一个或多个存储器,该存储器耦合到所述一个或多个处理器。

[0006] 根据另一个实施例,提供了一种用于视频编码的装置,包括一个或多个处理器,其中,所述一个或多个处理器被配置为:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在编码顺序中在所述图片的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及在针对所述块的所述第一扫描遍中对所述第一和第二组参数进行熵编码,其中,所述第一扫描遍是在用于对所述块的变换系数进行熵编码的一个或多个其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括第一标志和第二标志中的至少一个,所述第一标志指示相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2。

[0007] 根据另一个实施例,提供了一种视频解码的装置,包括:用于访问与图片的块中的第一变换系数相关联的第一组参数的部件,所述第一变换系数在解码顺序中在所述图片的所述块中的第二变换系数之前;用于访问与所述第二变换系数相关联的第二组参数的部件,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数的解码;用于在针对所述块的所述第一扫描遍中对所述第一和第二组参数进行熵解码的部件,其中,所述第一扫描遍是在用于对所述块的变换系数进行熵解码的一个或多个其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括第一标志和第二标志中的至少一个,所述第一标志指示相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2;以及用于响应于所述经解码的变换系数重构所述块的部件。

[0008] 根据另一个实施例,提供了一种视频编码的装置,包括:用于访问与图片的块中的第一变换系数相关联的第一组参数的部件,所述第一变换系数在编码顺序中在所述图片的所述块中的第二变换系数之前;用于访问与所述第二变换系数相关联的第二组参数的部件,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及用于在针对所述块的所述第一扫描遍中对所述第一和第二组参数进行熵编码的部件,其中,所述第一扫描遍是在用于对所述块的变换系数进行熵编码的一个或多个其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括第一标志和第二标志中的至少一个,所述第一标志指示相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2。

[0009] 根据另一个实施例,通过执行以下操作行程包括经编码视频的信号:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在编码顺序中在所述图片

的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及在针对所述块的所述第一扫描遍中对所述第一和第二组参数进行熵编码,其中,所述第一扫描遍是在用于对所述块的变换系数进行熵编码的一个或多个其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括第一标志和第二标志中的至少一个,所述第一标志指示相对应的变换系数的绝对值是否大于1,并且所述第二标志指示所述相对应的变换系数的所述绝对值是否大于2。

附图说明

[0010] 图1示出了可以在其中实现本实施例的各方面的系统的框图。

[0011] 图2示出了视频编码器的实施例的框图。

[0012] 图3示出了视频解码器的实施例的框图。

[0013] 图4是示出在JVET-J0014中提出的依赖性量化中使用的两个标量量化器的图示示例。

[0014] 图5是示出在JVET-J0014中提出的用于依赖性量化的状态转换和量化器选择的图示示例。

[0015] 图6是示出如在JVET-J0014中提出的,在CG中系数位元的顺序的图示示例。

[0016] 图7是示出如在JVET-K0319中提出的,基于SIG的状态转换的图示示例。

[0017] 图8是示出如在JVET-K0319中提出的,在CG中系数位元的顺序的图示示例。

[0018] 图9是示出根据一个实施例的CG中系数位元的顺序的图示示例。

[0019] 图10是示出根据一个实施例的基于SUM(SIG,gt1,gt2)的状态转换的图示示例。

[0020] 图11是示出根据一个实施例的基于XOR(SIG,gt1,gt2)的状态转换的图示示例。

[0021] 图12是示出根据一个实施例的基于gt1的状态转换的图示示例。

[0022] 图13示出了根据一个实施例的对当前块进行编码的过程。

[0023] 图14示出了根据一个实施例的对当前块进行解码的过程。

具体实施方式

[0024] 图1示出了可以在其中实现各方面和实施例的系统的示例的框图。系统100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、数字电视接收机、个人视频记录系统、连接的家用电器和服务器。系统100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100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。在各个实施例中,系统100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它类似系统或其它电子设备。在各个实施例中,系统100被配置为实现本申请中描述的一个或多个方面。

[0025] 系统100包括至少一个处理器110,其被配置为执行其中加载的指令,以实现例如本申请中描述的各方面。处理器110可以包括嵌入式存储器、输入输出接口和本领域已知的

各种其它电路。系统100包括至少一个存储器120(例如,易失性存储设备和/或非易失性存储设备)。系统100包括存储设备140,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140可以包括内部存储设备、附接的存储设备和/或网络可访问的存储设备。

[0026] 系统100包括编码器/解码器模块130,其被配置为例如处理数据以提供经编码视频或经解码视频,并且编码器/解码器模块130可以包括其自己的处理器和存储器。编码器/解码器模块130表示可以被包括在设备中以执行编码和/或解码功能的模块。众所周知,设备可以包括编码和解码模块之一或两者。另外,编码器/解码器模块130可以被实现为系统100的单独的元件,或者可以作为硬件和软件的组合合并到处理器110中,这是本领域技术人员所公知的。

[0027] 要被加载到处理器110或编码器/解码器130上以执行本申请中描述的各方面的程序代码可以被存储在存储设备140中,并且随后被加载到存储器120上以由处理器110执行。根据各个实施例,处理器110、存储器120、存储设备140和编码器/解码器模块130中的一个或多个可以在执行本申请中描述的过程期间存储各种项目中的一项或多项。这样存储的项目可以包括但不限于输入视频、经解码视频或经解码视频的一部分、比特流或信号、矩阵、变量和来自方程、公式、运算和运算逻辑的中间或最终结果。

[0028] 在几个实施例中,处理器110和/或编码器/解码器模块130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,在其它实施例中,处理设备外部的存储器(例如,处理设备可以是处理器110或编码器/解码器模块130)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120和/或存储设备140,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,诸如RAM之类的快速外部动态易失性存储器被用作工作存储器,用于针对诸如MPEG-2、HEVC或VVC的视频编码和解码操作。

[0029] 可以通过如框105中指示的各种输入设备来提供对系统100的元件的输入。这种输入设备包括但不限于:(i)接收例如通过广播器无线发送的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。

[0030] 在各个实施例中,框105的输入设备具有相关联的本领域中公知的相应输入处理元件。例如,RF部分可以与适用于以下的元件相关联:(i)选择所需的频率(也称为选择信号,或将信号频带限制到一个频带),(ii)下变频所选信号,(iii)再次频带限制到更窄的频带,以选择(例如)在某些实施例中可以称为信道的信号频带,(iv)对下变频和频带限制后的信号解调制,(v)执行纠错,和(vi)解复用以选择所需的数据包流。各个实施例的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、下变频器、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变频为较低频率(例如,中间频率或近基带频率)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在有线(例如缆线)介质上发送的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。各个实施例重新排列上述(和其它)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它元件。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。在各个实施例中,RF部分包括天线。

[0031] 另外,USB和/或HDMI端子可以包括相应的接口处理器,用于通过USB和/或HDMI连接将系统100连接到其它电子设备。应该理解,输入处理的各个方面,例如Reed-Solomon纠错,可以根据需要例如在单独的输入处理IC中或在处理器110中实现。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中或在处理器110中实现。经解调、经纠错和经解复用的流被提供给各种处理元件,包括例如处理器110和与存储器和存储元件组合操作以根据需要处理用于呈现在输出设备上的数据流的编码器/解码器130。

[0032] 可以在集成外壳内提供系统100的各种元件。在集成外壳内,可以使用合适的连接装置115(例如,本领域中公知的内部总线,包括I2C总线、接线和印刷电路板)互连各种元件并在它们之间发送数据。

[0033] 系统100包括使得能够经由通信信道190与其它设备进行通信的通信接口150。通信接口150可以包括但不限于配置为在通信信道190上发送和接收数据的收发机。通信接口150可以包括但不限于调制解调器或网卡,并且通信信道190可以例如在有线和/或无线介质中实现。

[0034] 在各个实施例中,使用诸如IEEE 802.11之类的Wi-Fi网络将数据流传输到系统100。这些实施例的Wi-Fi信号在适用于Wi-Fi通信的通信信道190和通信接口150上被接收。这些实施例的通信信道190典型地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,以允许流式应用和其它过顶(over-the-top)通信。其它实施例使用机顶盒向系统100提供流传输的数据,该机顶盒通过输入块105的HDMI连接来交付数据。其它实施例也使用输入块105的RF连接向系统100提供流传输的数据。

[0035] 系统100可以向包括显示器165、扬声器175和其它外围设备185的各种输出设备提供输出信号。在实施例的各种示例中,其它外围设备185包括独立DVR、硬盘播放器、立体声系统、照明系统和其它基于系统100的输出提供功能的设备中的一个或多个。在各个实施例中,使用诸如AV.Link、CEC或能够在有或没有用户干预的情况下实现设备到设备控制的其它通信协议之类的信令,在系统100与显示器165、扬声器175或其它外围设备185之间传递控制信号。输出设备可以经由通过各自的接口160、170和180的专用连接通信地耦合至系统100。可替代地,输出设备可以经由通信接口150使用通信信道190连接至系统100。显示器165和扬声器176可以与电子设备(例如,电视)中的系统100的其它组件集成在单个单元中。在各个实施例中,显示接口160包括显示驱动器,例如,时序控制器(T Con)芯片。

[0036] 例如,如果输入105的RF部分是单独的机顶盒的一部分,则显示器165和扬声器175可以替代地与一个或多个其它组件分开。在显示器165和扬声器175是外部组件的各个实施例中,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。

[0037] 图2示出了示例视频编码器200,诸如高效视频编码(HEVC)编码器。图2还示出在其中对HEVC标准做出改进的编码器或者采用类似于HEVC技术(诸如由JVET(联合视频探索小组)开发中的VVC(通用视频编码))的编码器。

[0038] 在本申请中,术语“重构的”和“经解码的”可以互换使用,术语“经编码的”或者“编码的”可以互换使用,并且术语“图像”、“图片”和“帧”可以互换使用。通常但并非必须,术语“重构的”在编码器侧使用,而“经解码的”在解码器侧使用。

[0039] 在编码之前,视频序列可能会经过预编码处理(201),例如,对输入的彩色图片进

行颜色转换(例如,从RGB 4:4:4转换为YCbCr 4:2:0),或者对输入图片分量执行重新映射,以便获得对压缩更具弹性的信号分布(例如,使用颜色分量之一的直方图均衡化)。元数据可以与预处理相关联,并附加到比特流。

[0040] 为了对具有一个或多个图片的视频序列进行编码,一个图片被分割(202)为例如一个或多个切片,其中每个切片可以包括一个或多个片段。在HEVC中,一个片段被组织成编码单元、预测单元和变换单元。HEVC规范区分"块"和"单元",其中"块"寻址样本阵列中的特定区域(例如亮度,Y),而"单元"包括所有经编码颜色分量(Y,Cb,Cr或单色)的并置块,语法元素以及与这些块相关联的预测数据(例如,运动矢量)。

[0041] 针对根据HEVC进行编码,将图片分割为具有可配置尺寸(典型地为64x64、128x128或者256x256像素)的方形编码树块(CTB),并将一组连续的编码树块分组为一个切片。编码树单元(CTU)包含经编码颜色分量的CTB。CTB是分割为编码块(CB)的四叉树的根,并且编码块可以被划分成一个或多个预测块(PB),并且形成分割为变换块(TB)的四叉树的根。大于4x4的变换块(TB)被划分为经量化系数的4x4子块,称为系数分组(CG)。与编码块、预测块和变换块相对应,编码单元(CU)包括预测单元(PU)和树状结构的变换单元(TU)集,PU包括所有颜色分量的预测信息,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的尺寸适用于相对应的CU、PU和TU。在本申请中,术语"块"可以用于指代例如CTU、CU、PU、TU、CG、CB、PB和TB中的任何一个。另外,术语"块"还可用于指代H.264/AVC或其它视频编码标准中指定的宏块和分割,并且更一般地指代各种尺寸的数据阵列。

[0042] 在编码器200中,如下所述,通过编码器元件对图片进行编码。以例如CU为单位处理要编码的图片。使用帧内或帧间模式对每个编码单元进行编码。当编码单元以帧内模式编码时,其执行帧内预测(260)。在帧间模式中,执行运动估计(275)和补偿(270)。编码器决定(205)使用帧内模式或者帧间模式中的哪一个来对编码单元进行编码,并且通过预测模式标志指示帧内/帧间决策。通过从原始图像块中减去(210)预测块来计算预测残差。

[0043] 然后,对预测残差进行变换(225)和量化(230)。对经量化的变换系数以及运动矢量和其它语法元素进行熵编码(245)以输出比特流。作为非限制性示例,可以使用基于上下文的自适应二进制算术编码(CABAC)来将语法元素编码到比特流中。

[0044] 为了用CABAC进行编码,通过二进制化过程将非二进制语法元素值映射到被称为位元(bin)串的二进制序列。针对位元,选择一个上下文模型。"上下文模型"是用于一个或多个位元的概率模型,并且根据最近编码的符号的统计从可用模型的选择中选择。每个位元的上下文模型由上下文模型索引(也称为"上下文索引")识别,并且不同的上下文索引对应于不同的上下文模型。上下文模型存储每个位元的概率为'1'或者'0',并且可以是自适应的或静态的。静态模型以针对位元的1'和'0'等概率来触发编码引擎。在自适应编码引擎中,基于位元的实际编码值来更新上下文模型。与自适应模型和静态模型相对应的操作模式分别被称为常规模式和旁路模式。基于上下文,二进制算术编码引擎根据相对应的概率模型对位元进行编码或解码。

[0045] 扫描模式将2D块转换为1D数组,并且定义样本或系数的处理顺序。扫描遍是对块中的变换系数的迭代(根据所选的扫描模式),以便对特定语法元素进行编码。

[0046] 在HEVC中,一个TB上的扫描遍在于根据扫描模式(对角线、水平、垂直)依次处理每个CG,并且还根据经过考虑的扫描顺序扫描每个CG内的16个系数。扫描开始于TB中的上一

个有效系数,并处理所有系数直到DC系数。CG被顺序扫描。最多可以对CG应用五次扫描。每个扫描遍对CG内的系数的语法元素如下进行编码:

[0047] • 重要系数标志(SIG,significant_coeff_flag):系数的重要性(零/非零)。

[0048] • 系数绝对水平大于1标志(gt1,coeff_abs_level_greater1_flag):指示系数水平的绝对值是否大于1。

[0049] • 系数绝对水平大于2标志(gt2,coeff_abs_level_greater2_flag):指示系数水平的绝对值是否大于2。

[0050] • 系数符号标志(coeff_sign_flag):有效系数的符号(0:正,1:负)。

[0051] • 系数剩余绝对水平(coeff_abs_level_remaining):系数水平的绝对值的剩余值(如果值比前一遍中编码后更大)。一个给定CG中可以被编码最多8个coeff_abs_level_greater1_flags,并且只针对CG中具有大于1的幅值的第一系数编码coeff_abs_level_greater2_flag

[0052] 在每个扫描遍中,仅在由先前的扫描遍确定必要时对语法进行编码。例如,如果系数不重要,则剩余的扫描遍对于该系数是不必要的。前三个扫描遍中的位元以常规模式编码,其中上下文模型索引取决于TB中特定系数的位置和本地模板覆盖的相邻域中先前编码系数的值。扫描遍4和5中的位元以旁路模式编码,使CG中的所有旁路位元被分组到一起。

[0053] 编码器也可以跳过该变换,并且例如以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二者,即,在不应用变换或量化过程的情况下直接对残差进行编码。在直接PCM编码中,不应用任何预测,并且将编码单位样本直接编码到比特流中。

[0054] 编码器对经编码块进行解码,以为进一步的预测提供参考。经量化的变换系数被解量化(240)并且被逆变换(250)以解码预测残差。组合(255)经解码的预测残差和预测块,图像块得以重构。环路滤波器(265)被应用于重构的图片,例如,以执行解块/SAO(样本自适应偏移)滤波以减少编码伪像。滤波后的图像被存储在参考图片缓冲器中(280)。

[0055] 图3示出了示例视频解码器300(诸如HEVC解码器)的框图。在解码器300中,如下所述,由解码器元件对比特流进行解码。如图2中所述,视频解码器300通常执行与编码遍相对应的解码遍,其执行视频解码作为对视频数据进行编码的一部分。图3也可以示出在其中对HEVC标准做出改进的解码器或者采用类似于HEVC的技术的解码器(诸如VVC解码器)。

[0056] 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200生成。首先对比特流进行熵解码(330)以获得变换系数、运动矢量、图片分割信息和其它经编码信息。如果将CABAC用于熵编码,则以与编码器上下文模型相同的方式初始化该上下文模型,并基于上下文模型从比特流中解码语法元素。

[0057] 图片分割信息指示图片如何分割,例如,CTU的尺寸,以及将CTU拆分为CU并且在适用时可能地拆分为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(335)为例如CTU,并且将每个CTU划分为CU。对变换系数进行解量化(340)并对其进行逆变换(350),以对预测残差进行解码。

[0058] 组合(355)经解码的预测残差和预测块,图像块得以重构。可以从帧内预测(360)或者运动补偿预测(即帧间预测)(375)获得(370)预测块。环路滤波器(365)被应用于重构的图像。滤波后的图像被存储在参考图片缓冲器中(380)。

[0059] 经解码的图片可以进一步经历解码后处理(385),例如,逆颜色变换(例如,从YCbCr 4:2:0转换为RGB 4:4:4)或者执行在预编码处理(201)中执行的重新映射过程的逆过程的逆重新映射。解码后处理可以使用在预编码处理中推导出并在比特流中发信号通知的元数据。

[0060] 在文档JVET-J0014,第十次会议:美国圣迭戈,2018年4月10日-20日(本文中简称"JVET-J0014")的题为"Description of SDR,HDR and 360°video coding technology proposal by Fraunhofer HHI(由Fraunhofer HHI提出的SDR、HDR和360度视频编码技术的描述)"的文章中提出了依赖性标量量化,其中针对量化切换具有不同重构水平的两个标量量化器。相比于传统的独立标量量化(如在HEVC和VTM-1中使用的),用于变换系数的可容许重构值的集合取决于在重构顺序中早于当前变换系数水平的变换系数水平的值。

[0061] 通过以下方式实现依赖性标量量化:(a)定义两个具有不同重构水平的标量量化器,以及(b)定义在两个标量量化器之间切换的过程。

[0062] 图4中示出了所使用的两个标量量化器,由Q 0和Q1表示。可用重构水平的位置由量化步长Δ唯一指定。如果我们忽略了变换系数的实际重构使用整数算法这一事实,则两个标量量化器Q0和Q1的特征如下:

[0063] Q0:第一量化器Q0的重构水平由量化步长Δ的偶数整数倍给出。当使用该量化器时,根据以下公式计算重构的变换系数t'

[0064] $t' = 2 \cdot k \cdot \Delta$

[0065] 其中k表示相关联的变换系数水平。应当注意,术语"变换系数水平"(k)是指经量化的变换系数值,例如,其对应于在下文的residual_coding语法结构中描述的TransCoeffLevel。术语"重构的变换系数"(t')是指解量化的变换系数值。

[0066] Q1:第二量化器Q1的重构水平由量化步长Δ的奇数整数倍给出,另外,重构水平等于零。变换系数水平k到重构变换系数t'的映射由下式指定:

[0067] $t' = (2 \cdot k - sgn(k)) \cdot \Delta$

[0068] 其中sgn(•)表示符号函数,sgn(x)=(k==0?0:(k<0?-1:1))。

[0069] 未在比特流中明确地发信号通知所使用的标量量化器(Q0或Q1)。相反,用于当前变换系数的量化器由按照编码/重构顺序在当前变换系数之前的变换系数水平的奇偶校验确定。

[0070] 如图5中所示,两个标量量化器(Q0和Q1)之间的切换是经由具有四个状态的状态机执行的。状态可以采用四个不同的值:0、1、2、3。该状态唯一地由在编码/重构顺序上在当前变换系数之前的变换系数水平的奇偶校验来确定。在变换块的逆量化开始时,将状态设置为等于0。变换系数以扫描顺序重构(即,以它们被熵编码/解码的相同顺序)。重构当前的变换系数之后,如表1中示出的更新状态,其中k表示变换系数水平的值。注意,下一个状态仅取决于当前状态和当前变换系数水平k的奇偶校验(k&1)。用k表示当前变换系数水平的值,状态更新可以写成

[0071] state=stateTransTable[state][k&1]

[0072] 其中stateTransTable表示图5和表1中示出的表,运算符&在二进制补码算术中指定按位"和"运算符。

[0073] 状态唯一指定使用的标量量化器。如果当前变换系数的状态等于0或1,则使用标

量量化器Q0。否则(状态等于2或3),使用标量量化器Q1。

[0074] 表1

[0075]

当前状态	针对...的下一个状态	
	(k&1)==0	(k&1)==1
0	0	2
1	2	0
2	1	3
3	3	1

[0076] 更一般地,可以在多于两个标量量化器中选择变换系数的量化器,并且状态机可以具有多于四个状态。或者可以经由其它可能的机制来处理量化器切换。

[0077] 在JVET-J0014中,还提出了与依赖性标量量化耦合的系数编码方案,通过该方案,用于经量化系数的上下文建模取决于所使用的量化器。具体而言,重要性标志(SIG)和大于1标志(gt1)中的每一个都有两组上下文模型,并且针对特定SIG或gt1选择的一组模型取决于用于相关联系数的量化器。因此,JVET-J0014中提出的系数编码要求在移动到下一个扫描位置之前完整重构经量化系数的绝对水平(absLevel),以便知道用于确定量化器以及因此的用于下一个系数的上下文集合的奇偶校验。也就是,为了获得上下文模型以便对系数(n+1)进行熵解码,需要完成系数n(SIG,gt1,...,gt4,符号标志,绝对剩余水平)的熵解码。因此,系数(n+1)的一些常规编码位元需要等待系数n的一些旁路经编码位元的解码,因此,不同系数的旁路经编码位元与常规编码位元交织,如图6中所示。

[0078] 如图6中所示,JVET-J0014中的系数编码的设计,与HEVC或者VTM-1中的设计相比降低了吞吐量,说明如下:

[0079] 1.增加了常规编码位元。由于上下文选择和区间细分计算,常规编码位元的处理比旁路经编码位元要慢。使用JVET-J0014中的系数编码时,系数组(CG)中的常规编码位元的数量多达80,而VTM-1中为25(SIG中为16,gt1中为8,gt2中为1)。

[0080] 2.旁路经编码位元未分组。将旁路位元分组为更长的链条会增加每个周期处理的位元的数量,从而减少处理单个旁路位元所需的循环数。但是,使用JVET-J0014中的系数编码时,CG中的旁路经编码位元未分组;相反,它们与每个系数的常规编码位元交织。

[0081] 本申请针对标量量化器的决策方案,以实现与HEVC和VTM-1中的系数编码设计几乎相同水平的吞吐量,同时保持依赖性标量量化提供的大部分增益。

[0082] 在JVET-K0319文稿中(见“CE7-Related:TCQ with High Throughput Coefficient Coding(CE7相关:具有高吞吐量系数编码的TCQ)”,文档JVET-K0319,JVET第十一次会议:Ljubljana,SI,2018年7月10日-18日,以下简称“JVET-K0319”),在JVET-J0014中提出的基于奇偶校验的状态转换被基于SIG的状态转换所代替,如图7和表2中所示,而JVET-J0014中的其它相关标量量化设计保持不变。通过这样做,用于量化当前变换系数的标量量化器由在扫描顺序中位于当前变换系数之前的经量化系数的SIG确定。

[0083] 表2

[0084]

当前状态	针对...的下一状态	
	sig == 0	sig==1
0	0	2
1	2	0
2	1	3
3	3	1

[0085] 在JVET-K0319中提出的系数编码基于HEVC和VTM-1系数编码。不同之处在于SIG和gt1的每一个都有两组上下文模型,并且熵编码器根据相关联系数使用的量化器针对特定SIG或gt1选择上下文集合。因此,将依赖性标量量化的标量量化器从基于奇偶校验改为基于SIG可以实现类似于HEVC和VTM-1系数编码的高吞吐量设计。CG中系数位元的建议顺序在图8中示出。也就是,使用JVET-K0319中提出的系数编码,每个CG的常规编码位元多达25个,与HEVC和VTM-1系数编码保持相同;CG中所有旁路经编码位元都分组在一起。

[0086] 在CE7的测试7.2.1软件中测试了JVET-J0014中的依赖性量化方法,仿真结果显示与VTM-1.0锚点相比降低了4.99%的AI(所有Intra)、3.40%的RA(随机访问)和2.70%的LDB(低延迟B)BD速率。但是,与VTM-1.0锚点相比,JVET-K0319中的模拟结果显示降低了3.98%的AI、2.58%的RA、1.80%的LDB BD速率。也就是,相比于JVET-J0014(基于经量化系数的完整absLevel切换)中提出的,用于如JVET-K0319(仅基于SIG切换)量化变换系数的标量量化器可能降低编码效率。

[0087] 本申请提出了用于依赖性标量量化的标量量化器的一些替代决策方案,以实现高吞吐量和编码效率之间的良好折衷。取代使用绝对水平的奇偶校验或者SIG值,提出了基于常规编码位元的状态转换和上下文模型选择。在下文中,描述了决定用于依赖性标量量化的标量量化器的若干实施例。

[0088] 针对在JVET-J0014中提出的依赖性标量量化,其需要在移动到下一个扫描位置之前完成经量化系数的绝对水平(absLevel)的重构,以便知道用于确定针对下一个系数的量化器的奇偶校验。因此,CG中的旁路经编码位元未分组,并且它们每一系数与常规编码位元交织。此外,与HEVC和VTM-1相反,每个变换系数水平的常规编码位元的最大数量增加了(在JVET-J0014中提出的方法中,每个变换系数水平最多可以存在5个常规编码位元),如在下面的语法表中所示。相对于VTM-1的变化是在斜体。熵编码器根据“状态”选择用于特定SIG或者gt1的上下文集合,这取决于该变化系数水平的信息并且被用于决定量化器。在以下语法中说明了位元的编码顺序,其中,函数getSigCtxId(XC,YC,state)被用于基于当前系数扫描位置(xC,yC)和状态推导语法sig_coeff_flag的上下文,decodeSigCoeffFlag(sigCtxId)用于对具有相关联的上下文sigCtxId的语法sig_coeff_flag进行解码,getGreater1CtxId(xC,yC,state)被用于基于当前系数扫描位置(xC,yC)和状态推导语法abs_level_gt1_flag的上下文,并且decodeAbsLevelGt1Flag(greater1CtxId)用于对具有相关联的上下文greater1CtxId的语法abs_level_gt1_flag进行解码。

[0089]

residual_coding(x0, y0, log2TrafoSizeX, log2TrafoSizeY, cIdx) {	描述符
if(transform_skip_enabled_flag && !cu_transquant_bypass_flag && (log2TrafoSizeX + log2TrafoSizeY <= (Log2MaxTransformSkipSize << 1)))	
transform_skip_flag[x0][y0][cIdx]	ae(v)
if(CuPredMode[x0][y0] == MODE_INTER && explicit_rdpcm_enabled_flag && (transform_skip_flag[x0][y0][cIdx] || cu_transquant_bypass_flag)) {	
explicit_rdpcm_flag[x0][y0][cIdx]	ae(v)
if(explicit_rdpcm_flag[x0][y0][cIdx])	
explicit_rdpcm_dir_flag[x0][y0][cIdx]	ae(v)
}	
last_sig_coeff_x_prefix	ae(v)
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
log2SbSize = (min(log2TrafoSizeX, log2TrafoSizeY) < 2 ? 1 : 2)	
numSbCoeff = 1 << (log2SbSize << 1)	
lastScanPos = numSbCoeff	
lastSubBlock = (1 << (log2TrafoSizeX + log2TrafoSizeY – 2 * log2SbSize)) – 1	
do {	
if(lastScanPos == 0) {	
lastScanPos = numSbCoeff	
lastSubBlock– –	
}	
lastScanPos– –	
xS = ScanOrder[log2TrafoSizeX – log2SbSize][log2TrafoSizeY – log2SbSize] [scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSizeX – log2SbSize][log2TrafoSizeY – log2SbSize] [scanIdx][lastSubBlock][1]	
xC = (xS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][lastScanPos][0]	
yC = (yS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
for(i = lastSubBlock; i >= 0; i– –) {	
xS = ScanOrder[log2TrafoSizeX – log2SbSize][log2TrafoSizeY – log2SbSize] [scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSizeX – log2SbSize][log2TrafoSizeY – log2SbSize] [scanIdx][lastSubBlock][1]	
inferSbDcSigCoeffFlag = 0	
state = 0	
if((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	

[0090]

for(n = (i == lastSubBlock) ? lastScanPos − 1 : numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][0]	
yC = (yS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 || !inferSbDcSigCoeffFlag)) {	
sigCtxId = getSigCtxId(xC, yC, state)	
sig_coeff_flag[xC][yC] = decodeSigCoeffFlag(sigCtxId)	ae(v)
}	
absLevel[xC][yC] = 0	
if(sig_coeff_flag[xC][yC]) {	
inferSbDcSigCoeffFlag = 0	
absLevel[xC][yC] = 1	
greater1CtxId = getGreater1CtxId (xC, yC, state)	
***abs_level_gt1_flag**[xC][yC] = decodeAbsLevelGt1Flag(greater1CtxId)*	*ae(v)*
if(abs_level_gt1_flag[xC][yC]) {	
absLevel[xC][yC] = 2	
***abs_level_gt2_flag**[xC][yC]*	*ae(v)*
if(abs_level_gt2_flag[xC][yC]) {	
absLevel[xC][yC] = 3	
***abs_level_gt3_flag**[xC][yC]*	*ae(v)*
if(abs_level_gt3_flag[xC][yC]) {	
absLevel[xC][yC] = 4	
***abs_level_gt4_flag**[xC][yC]*	*ae(v)*
if(abs_level_gt4_flag[xC][yC]) {	
***abs_level_remaining**[xC][yC]*	*ae(v)*
absLevel[xC][yC] = 5 + abs_level_remaining[xC][yC]	
}	
}	
}	
}	
}	
state = stateTransTable[state][absLevel[xC][yC] & 1]	
}	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][0]	
yC = (yS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
coeff_sign_flag[xC][yC]	ae(v)
TransCoeffLevel[x0][y0][cIdx][xC][yC] = absLevel[xC][yC] * (1 − 2 * coeff_sign_flag[xC][yC])	
}	
}	
}	
}	

[0091] 如前所述,由于这些语法的修改,存在关于高吞吐量硬件实现的潜在问题。在我们的建议中,提出了替代方法以实现与HEVC几乎相同的吞吐量水平,同时支持依赖性标量量化。在下文中,以HEVC为例来说明所提出的修改。

[0092] 在一个实施例中,每一变换系数水平的最大常规编码位元数量保持为3而不是5(SIG、gt1和gt2是常规编码的)。针对每个CG,常规编码位元和旁路编码位元按编码顺序分

开;首先,发送用于CG的所有常规编码位元,然后,发送旁路编码位元。CG中系数位元的建议顺序在图9中示出。用于CG的位元在CG中扫描位置上经历多遍编码:

[0093] • 第1遍:以编码顺序对重要性(SIG,sig_coeff_flag)、大于1标志(gt1,abs_level_gt1_flag)和大于2标志(gt2,abs_level_gt2_flag)的编码。大于1标志仅在如果sig_coeff_flag等于1时存在。大于2标志(abs_level_gt2_flag)的编码仅针对具有abs_level_gt1_flag等于1的扫描位置执行。如果在比特流中不存在gt1和gt2,则将其值推断为0。SIG、gt1和gt2标志被在常规模式中编码,并且SIG的上下文建模选择取决于针对相关联的系数选择了哪个状态。

[0094] • 第2遍:针对具有abs_level_gt2_flag等于1的所有扫描位置对语法元素abs_level_remaining的编码。非二进制语法元素被二进制化,并且以算术编码引擎的旁路模式对得到的位元进行编码;

[0095] • 第3遍:针对具有sig_coeff_flag等于1的所有扫描位置的符号(coeff_sign_flag)的编码。符号在旁路模式中进行编码。

[0096] 上述实施例示出所建议的相比于EVC的修改。这些修改也可以基于其它解决方案。例如,如果将JVET-J0014用作基础,则SIG和gt1的上下文建模将取决于量化器的选择,并且其它大于x标志(gtx,x=3和4)可以在第1遍或第2遍中进行编码。如果存在其它常规编码的位元,诸如gt5、gt6、gt7标志,则可以在第1遍或第2遍中对其进行编码。此外,第3遍中的标志(coeff_sign_flag)也可以在常规模式中进行编码。

[0097] 在2019年3月的会议上,JVET为变换跳过残差块采用了新的残差编码过程。当变换跳过(TS)被启用时,针对预测残差跳过变换。系数分组(CG)的残差水平如下在扫描位置上被进行三遍编码:

[0098] 第1遍:发信号通知以下标志

[0099] osig_coeff_flag

[0100] ocoeff_sign_flag

[0101] o大于1标志(abs_level_gtx_flag[0])

[0102] o奇偶校验(par_level_flag)标志

[0103] 第2遍:发信号通知一下标志

[0104] o大于3标志(abs_level_gtx_flag[1])

[0105] o大于5标志(abs_level_gtx_flag[2])

[0106] o大于7标志(abs_level_gtx_flag[3])

[0107] o大于9标志(abs_level_gtx_flag[4])

[0108] 第3遍:使用Golomb-Rice编码绕过剩余绝对水平(abs_remainder)的编码。

[0109] 以上提出的实施例也可以应用于这一新采用的TS残差编码,其中,大于3标志的位置被移动到第一遍,例如如下所示:

[0110] 第1遍:发信号通知以下标志

[0111] osig_coeff_flag

[0112] ocoeff_sign_flag

[0113] o大于1标志(abs_level_gtx_flag[0])

[0114] o奇偶校验(par_level_flag)标志

[0115] o大于3标志(abs_level_gtx_flag[1])

[0116] 第2遍:发信号通知一下标志

[0117] o大于5标志(abs_level_gtx_flag[2])

[0118] o大于7标志(abs_level_gtx_flag[3])

[0119] o大于9标志(abs_level_gtx_flag[4])

[0120] 第3遍:使用Golomb-Rice编码绕过剩余绝对水平(abs_remainder)的编码。

[0121] 实施例1-基于函数SUM(SIG,gt1,gt2)的标量量化器决策方案

[0122] 为了解决高吞吐量硬件实现的问题,将不会执行经量化系数的绝对水平(absLevel)的完全重构以决定状态,并且两个标量量化器之间的切换也不会取决于完全变换系数绝对水平的奇偶校验。如前所述,如JVET-K0319中仅由SIG确定的标量量化器可能会降低编码效率。在一个实施例中,我们建议基于函数SUM(SIG,gt1,gt2)决定标量量化器,该函数将共同考虑当前变换系数的SIG、gt1和gt2值。

[0123] 表3

[0124]

标记值m	gt2	gt1	SIG	可能的变换绝对水平
0	0	0	0	0
1	0	0	1	1
2	0	1	1	2
3	1	1	1	≥3

[0125] 表3中示出了变换系数的SIG、gt1和gt2值的可能组合。存在从这四个不同组合到四个可能的标记水平值的一对一对应映射,其中m表示这四个可能的情况的标记值。从SIG、gt1和gt2值推导出标记值m的函数可以写成:

[0126] m＝SUM(SIG,gt1,gt2)＝SIG+gt1+gt2

[0127] 如图10中所示,这两个标量量化器之间的切换由在编码/重构顺序中在当前变换系数之前的变化系数水平的标记值m的奇偶校验唯一地确定。在针对变换块的逆量化开始时,将状态设置为等于0。在重构当前变换系数的常规编码位元之后,如图10和表4中所示更新状态。注意,下一个状态仅取决于当前状态以及当前变换系数水平的标记值m的奇偶校验(m&1)。该状态更新可以写为:

[0128] state＝stateTransTable[state][m&1]

[0129] 其中stateTransTable表示图10和表4中示出的表,并且运算符&指定在二进制补码算术中的按位“和”运算符。

[0130] 表4

[0131]

当前状态	针对...的下一个状态	
	（m&1）==0	（m&1）==1
0	0	2
1	2	0
2	1	3
3	3	1

[0132] 经过变换和量化之后，大多数变换系数的量级通常处于非常低的值。当变换系数的绝对水平小于3时，如表3和表4中所示，我们提出的方法可以获得与JVET-J0014几乎相同的结果。同时，我们提出的方法不需要经量化系数的绝对水平(absLevel)的完全重构，从而解决了高吞吐量的硬件实现问题。

[0133]

residual_coding(x0, y0, log2TrafoSizeX, log2TrafoSizeY, cIdx) {	描述符
...	
state = 0	
for(i = lastSubBlock; i >= 0; i− −) {	
...	
//====== first pass: regular bins of sig, gt1, gt2 ======	
for(n = (i == lastSubBlock) ? lastScanPos − 1 : numSbCoeff − 1; n >= 0; n− −) {	
m[xC][yC] = 0	
xC = (xS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][0]	
yC = (yS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 || !inferSbDcSigCoeffFlag)) {	
sigCtxId = getSigCtxId(xC, yC, state)	
sig_coeff_flag[xC][yC] = decodeSigCoeffFlag(sigCtxId)	ae(v)
*m[xC][yC] += **sig_coeff_flag**[xC][yC]*	
}	
absLevel[xC][yC] = 0	
if(sig_coeff_flag[xC][yC]) {	
absLevel[xC][yC] = 1	
***abs_level_gt1_flag**[xC][yC]*	ae(v)
*m[xC][yC] += **abs_level_gt1_flag** [xC][yC]*	
if(abs_level_gt1_flag[xC][yC]) {	
absLevel[xC][yC] = 2	
***abs_level_gt2_flag**[xC][yC]*	ae(v)
*m[xC][yC] += **abs_level_gt2_flag** [xC][yC]*	
}	
}	
***state** = stateTransTable[state][m[xC][yC] & 1]*	
}	
//====== end of first pass: regular bins of sig, gt1, gt2 ======	
//====== second pass: bypass bins of remainder ======	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][0]	
yC = (yS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][1]	
if(abs_level_gt2_flag [xC][yC]) {	
abs_level_remaining[xC][yC]	ae(v)
absLevel[xC][yC] = 3 + **abs_level_remaining**[xC][yC]	
}	
}	
//====== end of second pass: bypass bins of remainder ======	
//====== third pass: signs ======	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][0]	
yC = (yS << log2SbSize) + ScanOrder[log2SbSize][log2SbSize][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	

[0134]

coeff_sign_flag[xC][yC]	ae(v)
TransCoeffLevel[x0][y0][cIdx][xC][yC] = absLevel[xC][yC] * (1 – 2 * coeff_sign_flag[xC][yC])	
}	
}	
//===== end of third pass: signs =====	
}	
}	

[0135] 上面的语法表给出了有关编码顺序和位元的存在以及从发送的数据重构变换系数水平的详细信息。为了便于说明,在语法表中注释了扫描位置上的不同遍。相对于HEVC和VTM-1的更改以斜体显示。熵编码器根据“状态”来选择针对特定SIG的上下文集合,其取决于变换系数水平的信息并且用于确定量化器。

[0136] 实施例2-基于函数XOR(SIG,gt1,gt2)的标量量化器决策方案

[0137] 在另一个实施例中,考虑当前变换系数的SIG、gt1和gt2值,并且基于函数XOR(SIG,gt1,gt2)选择标量量化器。从SIG、gt1和gt2值推导出异或值x的函数可以写为:

[0138] x=XOR(SIG,gt1,gt2)=SIG^gt1^gt2

[0139] 在表5中显示变换系数的SIG、gt1和gt2值的可能组合的相对应异或值x。

[0140] 表5

[0141]

gt2	gt1	SIG	异或值x
0	0	0	0
0	0	1	1
0	1	1	0
1	1	1	1

[0142] 与第一实施例相比,这两个标量量化器之间的切换由SIG、gt1和gt2标志的异或值x唯一地确定。状态更新可以写为:

[0143] state=stateTransTable[state][x]

[0144] 其中,stateTransTable表示图11和表6中示出的表。状态机的其它部分保持与以前的方法类似。

[0145] 表6

[0146]

当前状态	针对...的下一个状态	
	x == 0	x == 1
0	0	2
1	2	0
2	1	3
3	3	1

[0147]

[0148] 当变换系数的绝对水平小于3时,如表5和表6中所示,我们提出的方法可以获得与JVET-J0014几乎相同的结果。同时,我们提出的方法不需要经量化系数的绝对水平

(absLevel)的完全重构,从而解决了高吞吐量的硬件实现问题。

[0149] 实施例3-基于常规编码位元之一的标量量化器决策方案

[0150] 根据上述实施例,将考虑当前变换系数(SIG,gt1和gt2)的所有常规编码位元以决定标量量化器。在另一个实施例中,这两个标量量化器之间的切换可以基于常规编码位元中的一个,例如,gt1标志。在该实施例中,先前的状态转换可以由基于gt1的状态转换代替,如图12中所示,同时其它设计保持不变。通过这样做,用于量化当前变换系数的标量量化器由在扫描顺序中位于当前变换系数之前的经量化系数的gt1标志确定。状态更新可以写为:

[0151] state=stateTransTable[state][gt1]

[0152] 其中,stateTransTable表示图12和表7中示出的表。

[0153] 在该实施例中,针对CG的位元在CG中的扫描位置上在3个扫描遍中进行编码:针对sig、gt1和gt2的第一遍,针对剩余绝对水平的第二遍,以及这对签名信息的第三遍。在一个变型中,针对CG的位元在CG中的扫描位置上在4个扫描遍中编码:针对sig和gt1的第一遍,针对gt2的第二遍,针对剩余绝对水平的第三遍,和针对签名信息的第四遍。与先前实施例中提出的3个扫描遍历相比,此变型可以进一步减少位元-到-位元依赖性。

[0154] 表7

[0155]

当前状态	针对...的下一个状态	
	gt1 == 0	gt1 == 1
0	0	2
1	2	0
2	1	3

[0156]

3	3	1

[0157] 替代地,用于量化当前变换系数的标量量化器由在扫描顺序中位于当前变换系数之前的经量化系数的gt2标志确定。更一般而言,用于量化当前变换系数的标量量化器由在扫描顺序中位于当前变换系数之前的经量化系数的一个常规编码位元(例如,gtx标志)确定。

[0158] 在以上示例中,我们示出了基于HEVC的几个实施例,其针对系数使用3个常规编码位元(SIG,gt1,gt2)。当系数的常规编码位元与HEVC中的常规编码位元不同时,可以通过对每个变换系数采用不同数量(大于或少于3个)的常规编码位元来执行所提出的实施例。

[0159] 在上文中,考虑了求和和异或函数,还可以用每个变换系数水平不同于常规编码位元的状态更新推导(1/0)函数来执行所提出的实施例。

[0160] 在上文中,描述主要是关于解量化的。应该注意的是,对量化进行了相应的调整。未在比特流中明确地发信号通知所使用的标量量化器(Q0或Q1)。例如,在编码器侧的量化模块将基于状态选择要针对当前变换系数使用哪个量化器。如果当前变换系数的状态等于0或1,则使用标量量化器Q0。否则(状态等于2或3),则使用标量量化器Q1。该状态是使用如图10-12中描述的方法,由变换系数水平的信息唯一确定的,并且解码器会使用相同的量化器以便相应地解码比特流。

[0161] 图13示出根据实施例的用于对当前编码单元进行编码的方法(1300)。在步骤1305,将初始状态设置为零。为了对编码单元进行编码,扫描编码单元中的系数。用于编码单元的扫描遍根据扫描模式(对角线、水平、垂直)顺序地处理编码单元的每个CG,并且根据所考虑的扫描顺序扫描每个CG内的系数。在编码单元中具有最后一个重要系数在CG处开始扫描(1315),并且处理所有系数直到具有DC系数的第一CG为止。

[0162] 如果CG不包含最后一个重要系数或者DC系数(1320),则对指示CG是否包含任何非零系数的标志(coded_sub_block_flag)进行编码(1325)。对于包含最后的非零水平或者DC系数的CG,coded_sub_block_flag被推断为等于1,并且不出现在比特流中。

[0163] 如果coded_sub_block_flag为真(1330),则将3个扫描遍应用于CG。在第一遍(1335-1360)中,针对系数,对SIG标志(sig_coeff_flag)进行编码(1335)。为了对SIG标志进行编码,使用状态来决定上下文模式索引,例如sigCtxId=getSigCtxId(state)。如果SIG标志为真(1340),则对gt1标志(abs_level_gt1_flag)进行编码(1345)。如果gt1标志为真(1350),则对gt2标志(abs_level_gt2_flag)进行编码(1355)。基于SIG、gt1和gt2标志中的一个或多个,例如,使用如图10-图12所描述的方法来更新状态(1360)。

[0164] 在第二遍扫描中(1365,1370),编码器检查gt2标志是否为真(1365)。如果为真,则对剩余绝对水平(abs_level_remaining)进行编码(1370)。在第三遍扫描中(1375,1380),编码器检查SIG标志是否为真(1375)。如果为真,则对符号标志(coeff_sign_flag)进行编码(1380)。在步骤1385,编码器检查是否还有更多的CG要处理。如果是,则继续进行下一个要处理的CG(1390)。

[0165] 图14示出根据实施例的用于对当前编码单元进行解码的方法(1400)。在步骤1405,将初始状态设置为零。类似于编码器侧,为了对编码单元进行解码,扫描编码单元中的系数位置。在编码单元中具有最后一个重要系数的CG开始扫描(1415),然后处理所有系数直到具有DC系数的第一个CG。

[0166] 如果CG不包含最后的重要系数或者DC系数(1420),对指示CG是否包含任何非零系数的标志(coded_sub_block_flag)进行解码(1425)。对于包含最后的非零水平或者DC系数的CG,coded_sub_block_flag被推断为等于1。

[0167] 如果coded_sub_block_flag为真(1430),则将3个扫描遍应用于CG。在第一遍中(1435-1460),针对系数,对SIG标志(sig_coeff_flag)进行解码(1435)。为了对SIG标志进行解码,使用状态决定上下文模式索引,例如,sigCtxId=getSigCtxId(state)。如果SIG标志为真(1440),则对gt1标志(abs_level_gt1_flag)进行解码(1445)。如果gt1标志为真(1450),则对gt2标志(abs_level_gt2_flag)进行解码(1455)。基于SIG、gt1和gt2标志中的一个或多个,例如,使用图10-图12中所描述的方法来更新状态(1460)。

[0168] 在第二扫描遍中(1470,1475),则解码器检查gt2标志是否为真(1470)。如果为真,则对剩余绝对水平(abs_level_remaining)进行解码(1475)。在第三扫描遍中(1480,1485),解码器检查SIG标志是否为真(1480)。如果为真,则对符号标志(coeff_sign_flag)进行解码(1485)。在步骤1487,解码器基于可用的SIG、gt1、gt2、符号标志和剩余绝对值计算变换系数。

[0169] 解码器检查是否有更多的CG要处理(1490)。如果是,它将继续处理下一个要处理的CG(1495)。如果所有系数都被熵解码,则使用依赖性标量量化来对变换系数进行解量化

(1497)。针对变换系数所使用的标量量化器(Q0或Q1)由状态确定,该状态是使用图10-图12中描述的方法,用经解码变换系数水平的信息推导出的。

[0170] 本文描述了各种方法,并且每个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。除非方法的正确操作需要特定的步骤或动作顺序,否则可以修改或者组合特定步骤和/或动作的顺序和/或使用。另外,诸如“第一”、“第二”等的术语可以在各种实施例中用于修改元素、组件、步骤、操作等,例如,“第一解码”和“第二解码”。除非特别要求,否则使用这些术语并不意味着对修改后的操作进行排序。因此,在该示例中,第一解码不需要在第二解码之前执行,并且可以例如在第二解码之前、期间或在与第二解码重叠的时间段中发生。

[0171] 各种方法和在本申请中描述的其它方面可以被用来修改模块,例如,如图2和图3中示出的视频编码器200和解码器300的熵编码和解码模块(245,330)。此外,当前方面不限于VVC或HEVC,并且可以应用于例如其它标准和建议,以及任何这样的标准和建议的扩展。除非另外指出或在技术上被排除,否则本申请中描述的方面可以单独或组合使用。

[0172] 在本申请中使用各种数值。特定值是出于示例目的,并且所描述的方面不限于这些特定值。

[0173] 根据一个实施例,提供了一种视频解码方法,包括:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在解码顺序中在所述图片的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数的解码;在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵解码,其中,第一扫描遍是在用于对所述块的变换系数进行熵解码的其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括gt1标志和gt2标志中的至少一个,所述gt1标志指示相对应的变换系数的绝对值是否大于1,并且所述gt2标志指示相对应的变换系数的绝对值是否大于2;以及响应于所述经解码的变换系数重构所述块。

[0174] 根据一个实施例,提供了一种视频编码的方法,包括:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在编码顺序中在所述图片的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵编码,其中,第一扫描遍是在用于对所述块的变换系数进行熵编码的其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括gt1标志和gt2标志中的至少一个,所述gt1标志指示相对应的变换系数的绝对值是否大于1,并且所述gt2标志指示相对应的变换系数的绝对值是否大于2。

[0175] 根据另一个实施例,提供了一种用于视频解码的装置,该装置包括一个或多个处理器,其中,所述一个或多个处理器被配置为:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在解码顺序中在所述图片的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建

模取决于所述第一变换系数的解码;在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵解码,其中,第一扫描遍是在用于对所述块的变换系数进行熵解码的其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括gt1标志和gt2标志中的至少一个,所述gt1标志指示相对应的变换系数的绝对值是否大于1,并且所述gt2标志指示相对应的变换系数的绝对值是否大于2;以及响应于所述经解码的变换系数重构所述块。该装置还包括耦合到所述一个或多个处理器的一个或多个存储器。

[0176] 根据另一个实施例,提供了一种用于视频编码的装置,该装置包括一个或多个处理器,其中,所述一个或多个处理器被配置为:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在编码顺序中在所述图片的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵编码,其中,第一扫描遍是在用于对所述块的变换系数进行熵编码的其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括gt1标志和gt2标志中的至少一个,所述gt1标志指示相对应的变换系数的绝对值是否大于1,并且所述gt2标志指示相对应的变换系数的绝对值是否大于2。

[0177] 根据另一个实施例,提供了一种视频解码装置,包括:用于访问与图片的块中的第一变换系数相关联的第一组参数的部件,所述第一变换系数在解码顺序中在所述图片的所述块中的第二变换系数之前;用于访问与所述第二变换系数相关联的第二组参数的部件,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数的解码;用于在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵解码的部件,其中,第一扫描遍是在用于对所述块的变换系数进行熵解码的其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括gt1标志和gt2标志中的至少一个,所述gt1标志指示相对应的变换系数的绝对值是否大于1,并且所述gt2标志指示相对应的变换系数的绝对值是否大于2;以及用于响应于所述经解码的变换系数重构所述块的部件。

[0178] 根据另一个实施例,提供了一种视频编码装置,包括:用于访问与图片的块中的第一变换系数相关联的第一组参数的部件,所述第一变换系数在编码顺序中在所述图片的所述块中的第二变换系数之前;用于访问与所述第二变换系数相关联的第二组参数的部件,所述第一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及用于在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵编码的部件,其中,第一扫描遍是在用于对所述块的变换系数进行熵编码的其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括gt1标志和gt2标志中的至少一个,所述gt1标志指示相对应的变换系数的绝对值是否大于1,并且所述gt2标志指示相对应的变换系数的绝对值是否大于2。

[0179] 根据另一个实施例,通过执行以下步骤形成包括经编码视频的信号:访问与图片的块中的第一变换系数相关联的第一组参数,所述第一变换系数在编码顺序中在所述图片的所述块中的第二变换系数之前;访问与所述第二变换系数相关联的第二组参数,所述第

一和第二组参数在常规模式中被熵编码,其中,用于所述第二变换系数的所述第二组参数中的至少一个参数的上下文建模取决于所述第一变换系数;以及在针对所述块的第一扫描遍中对所述第一和第二组参数进行熵编码,其中,第一扫描遍是在用于对所述块的变换系数进行熵编码的其它扫描遍之前执行的,并且其中,所述第一和第二变换系数的所述第一和第二组参数的每一组包括gt1标志和gt2标志中的至少一个,所述gt1标志指示相对应的变换系数的绝对值是否大于1,并且所述gt2标志指示相对应的变换系数的绝对值是否大于2。

[0180] 根据实施例,用于所述第二变换系数的所述第二组参数中的至少一个参数的所述上下文建模取决于所述第一变换系数的所述第一组参数的解码,并且独立于(1)用于表示所述第一变换系数的参数,和(2)在旁路模式中熵编码的参数。

[0181] 根据实施例,SIG标志也在第一扫描遍中进行编码或者解码,所述SIG标志指示所述相对应的变换系数是否为零。

[0182] 根据实施例,所述gt1标志在第一扫描遍中进行编码或者解码,并且所述gt2标志在第二扫描遍中进行编码或者解码。

[0183] 根据实施例,基于所述第一变换系数在两个或者更多量化器之间选择用于对所述第二变换系数进行解量化的解量化器。

[0184] 根据实施例,基于用于所述第一变换系数的所述第一组参数选择所述解量化器。

[0185] 根据实施例,用于所述第二变换系数的所述第二组参数中至少一个参数的上下文建模取决于所述第一和第二变换系数的解码。

[0186] 根据实施例,基于所述SIG、gt1和gt2标志之和选择所述解量化器。

[0187] 根据实施例,基于所述SIG、gt1和gt2标志的异或函数选择所述解量化器。

[0188] 根据实施例,基于所述gt1标志、所述gt2标志或者gtx标志选择所述解量化器,所述gtx标志指示所述相对应的变换系数的绝对值是否大于x。

[0189] 根据实施例,(1)用于表示所述块中的变换系数的参数,和(2)在旁路模式中编码的参数在(1)用于表示所述块中的变换系数的参数和(2)在所述常规模式中编码的参数之后在一个或多个扫描遍中进行熵编码或者解码。

[0190] 根据实施例,所述SIG、gt1、gt2标志或者gtx标志的上下文建模基于量化器或者量化器选择中使用的状态。

[0191] 实施例提供一种包括指令的计算机程序,当指令由一个或多个处理器执行时,指令使一个或多个处理器执行根据上述任何实施例的编码方法或解码方法。本实施例中的一个或多个还提供了一种计算机可读存储介质,其上存储有用于根据上述方法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一种计算机可读存储介质,其上存储有根据上述方法生成的比特流。一个或多个实施例还提供一种用于发送或接收根据上述方法生成的比特流的方法和装置。

[0192] 各种实现方式涉及解码。如本申请中所使用的,“解码”可以涵盖例如对接收到的经编码序列执行的全部或部分过程,以便产生适合于显示的最终输出。在各种实施例中,这样的过程包括典型地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。短语“解码过程”是旨在专门指代操作的子集还是广义上指代更广泛的解码过程将是显而易见的,并且相信是本领域技术人员充分理解的。

[0193] 各种实现方式涉及编码。以与上述关于“解码”的讨论类似的方式,在本申请中使用的“编码”可以涵盖例如对输入视频序列执行的全部或部分过程,以便产生经编码比特流。

[0194] 注意,本文所使用的语法元素(例如,sig_coeff_flag、abs_level_gt1_flag)是描述性术语。因此,它们不排除使用其它语法元素名称。

[0195] 本文描述的实现方式和方面可以例如以方法或过程、装置、软件程序、数据流或信号来实现。即使仅在单一形式的实现方式的上下文中讨论(例如,仅作为方法讨论),所讨论的特征的实现方式也可以以其它形式(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。方法可以在例如装置(例如处理器)中实现,该处理器通常是指处理设备,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”),以及其它有助于终端用户之间信息通信的设备。

[0196] 对“一个实施例”或“实施例”或“一个实现方式”或“实现方式”以及它们的其它变型的引用,意味着在至少一个实施例中包括结合实施例描述的特定特征、结构、特性等。因此,在本申请中各处出现的短语“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”以及任何其它变型的出现不必须全部指代相同的实施例。

[0197] 另外,本申请可能涉及“确定”各种信息。确定信息可以包括例如确定信息、估计信息、计算信息、预测信息或从存储器取回信息中的一个或多个。

[0198] 此外,本申请可能涉及“访问”各种信息。访问信息可以包括例如接收信息、取回信息(例如从存储器)、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的一个或多个。

[0199] 另外,本申请可能涉及“接收”各种信息。接收与“访问”一样,旨在是广义的术语。接收信息可以包括例如访问信息或取回信息(例如,从存储器)中的一个或多个。此外,在例如存储信息、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之类的操作期间,典型地以一种或另一种方式涉及“接收”。

[0200] 应当了解,例如在“A/B”、“A和/或B”和“A和B中的至少一个”的情况下使用以下“/”、“和/或”和“…中的至少一个”中的任何一个,旨在涵盖仅选择第一个列出的选项(A),或者仅选择第二个列出的选项(B),或者同时选择两个选项(A和B)。作为进一步的示例,在“A、B和/或C”和“A、B和C中的至少一个”的情况下,这种措词旨在涵盖仅选择第一个列出的选项(A),或者仅选择第二个列出的选项(B),或者仅选择第三个列出的选项(C),或者仅选择第一个和第二个列出的选项(A和B),或者仅选择第一个和第三个列出的选项(A和C),或者仅选择第二个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)。如对于本领域和相关领域的普通技术人员显而易见的那样,这可以针对如所列出的那样多的项目扩展。

[0201] 如对于本领域普通技术人员将显而易见的,实现方式可以产生被格式化以携带信息的各种不同信号,这些信号例如可以被存储或发送。该信息可以包括例如用于执行方法的指令或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。可以将这种信号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且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。众所周知,该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。

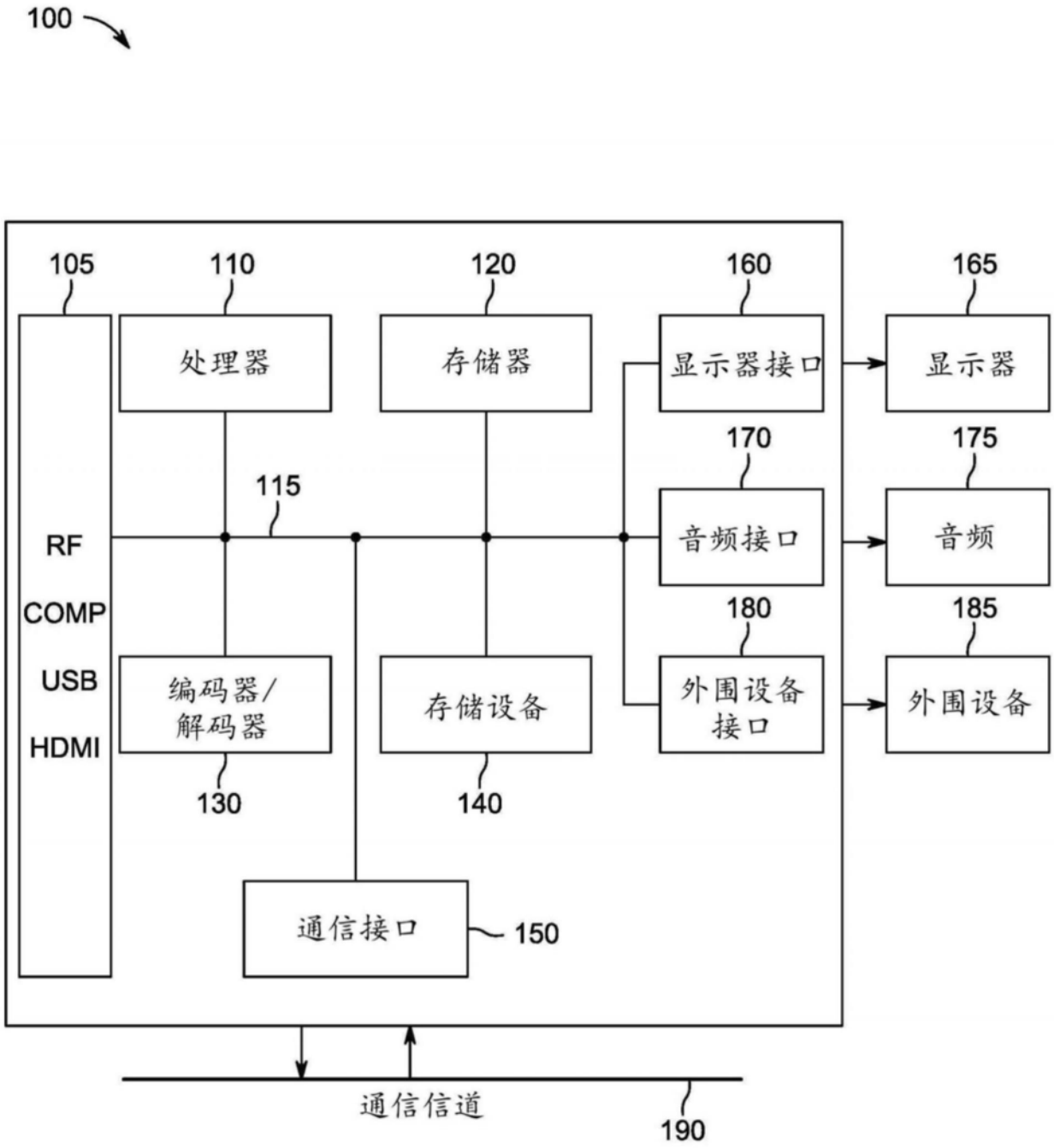

100
105
110
120
160
165
处理器
存储器
显示器接口
显示器
170
175
115
RF
COMP
USB
HDMI
音频接口
音频
180
185
编码器/
解码器
存储设备
外围设备
接口
外围设备
130
140
通信接口
150
通信信道
190

图1

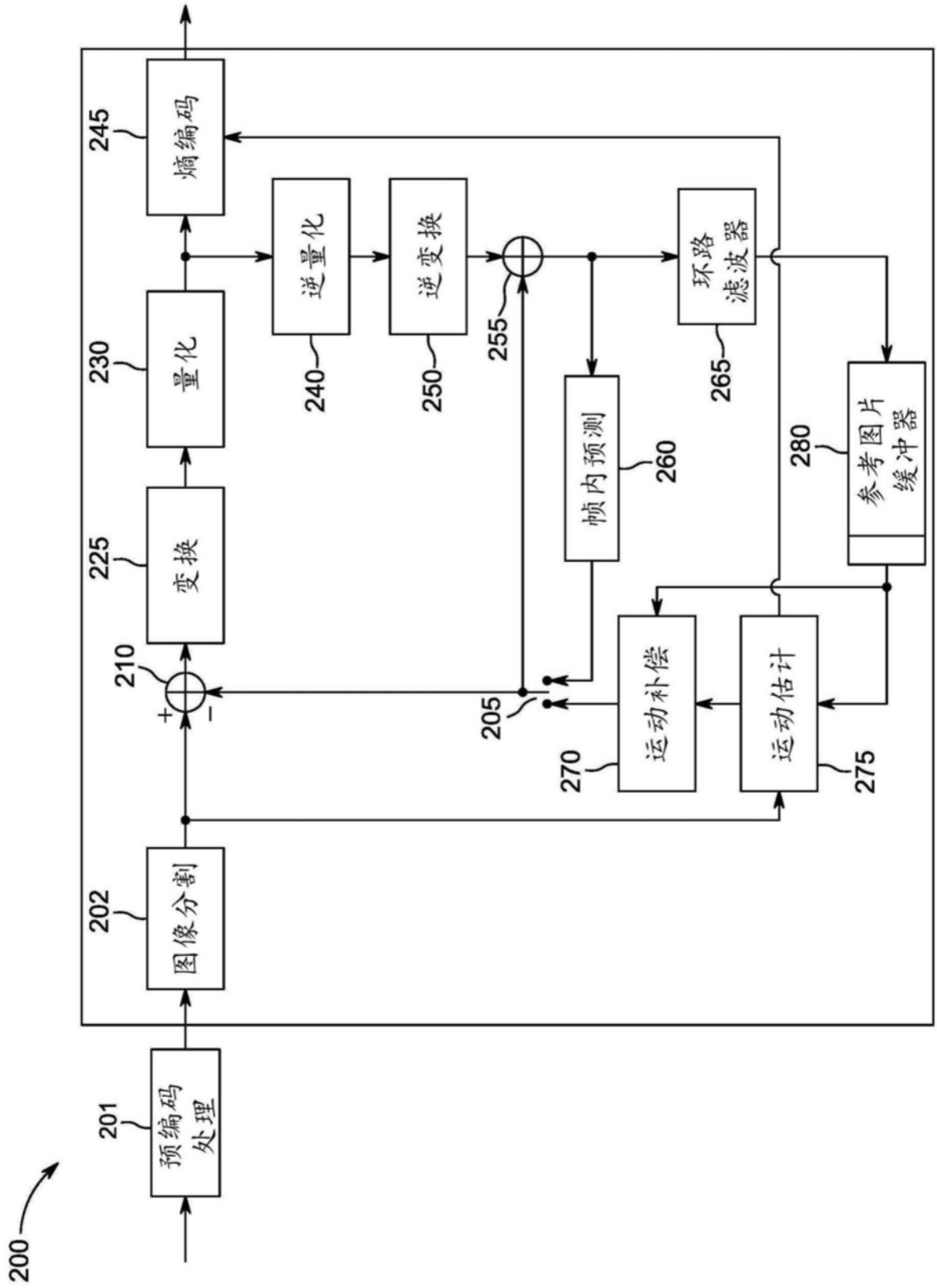

200
201
预编码处理
202
图像分割
210
+
−
225
变换
230
量化
245
熵编码
240
逆量化
250
逆变换
255
205
260
帧内预测
265
环路滤波器
270
运动补偿
275
运动估计
280
参考图片缓冲器

图2

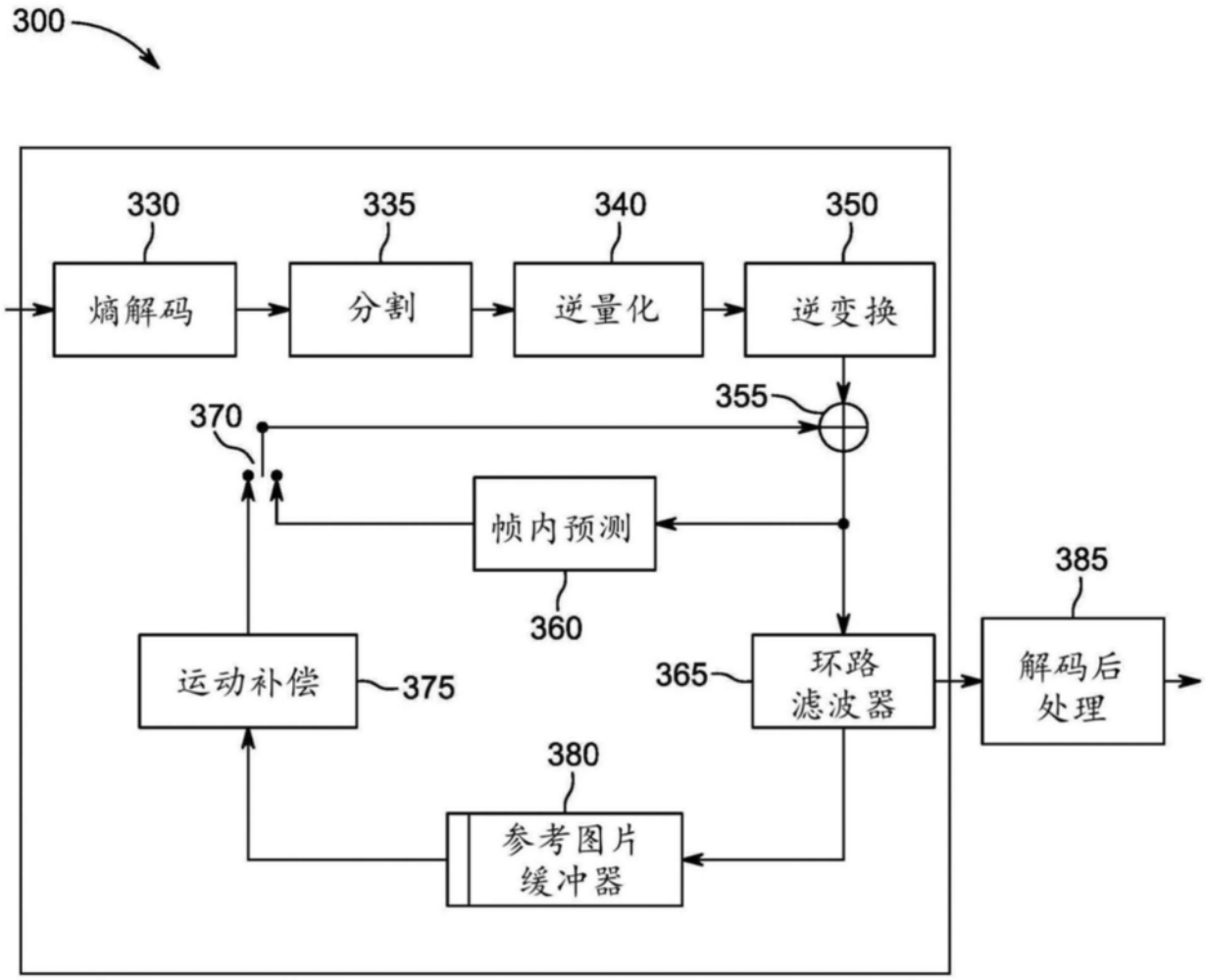

300
330
335
340
350
熵解码
分割
逆量化
逆变换
355
370
帧内预测
360
385
运动补偿
375
365
环路
滤波器
解码后
处理
380
参考图片
缓冲器

图3

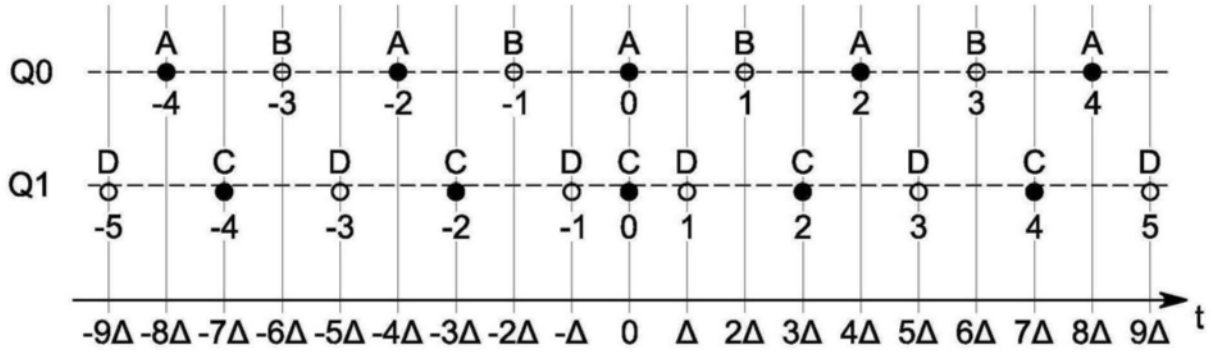

Q0
Q1
A -4
B -3
A -2
B -1
A 0
B 1
A 2
B 3
A 4
D -5
C -4
D -3
C -2
D -1
C 0
D 1
C 2
D 3
C 4
D 5
-9Δ -8Δ -7Δ -6Δ -5Δ -4Δ -3Δ -2Δ -Δ 0 Δ 2Δ 3Δ 4Δ 5Δ 6Δ 7Δ 8Δ 9Δ
t

图4

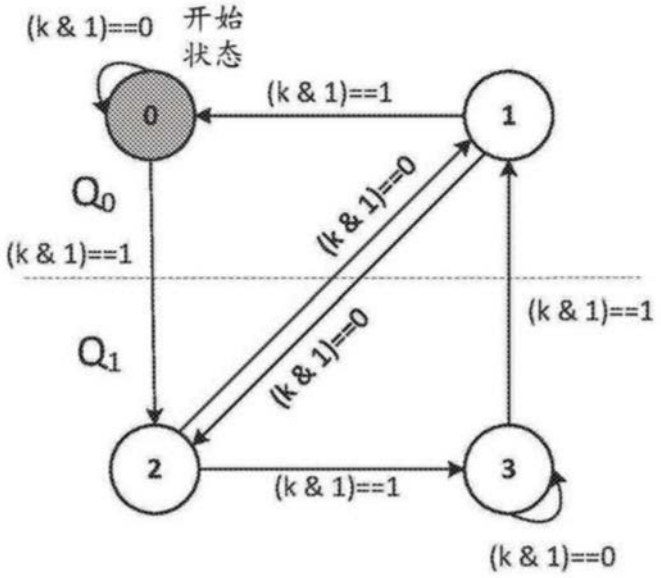

(k & 1)==0
开始
状态
0
(k & 1)==1
1
Q0
(k & 1)==1
(k & 1)==0
(k & 1)==1
Q1
(k & 1)==0
2
(k & 1)==1
3
(k & 1)==0

图5

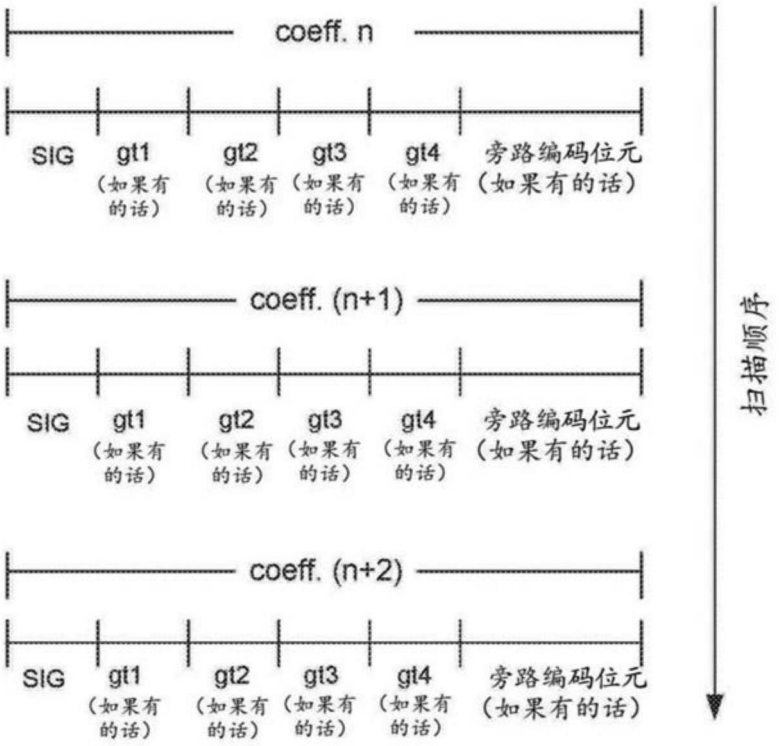

coeff. n
SIG
gt1
(如果有的话)
gt2
(如果有的话)
gt3
(如果有的话)
gt4
(如果有的话)
旁路编码位元
(如果有的话)
coeff. (n+1)
SIG
gt1
(如果有的话)
gt2
(如果有的话)
gt3
(如果有的话)
gt4
(如果有的话)
旁路编码位元
(如果有的话)
coeff. (n+2)
SIG
gt1
(如果有的话)
gt2
(如果有的话)
gt3
(如果有的话)
gt4
(如果有的话)
旁路编码位元
(如果有的话)
扫描顺序

图6

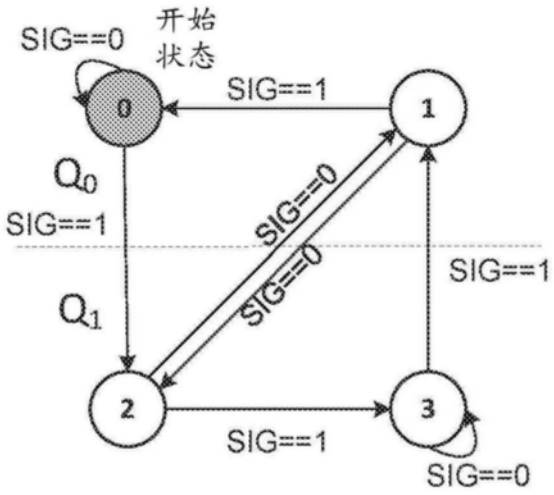


图7

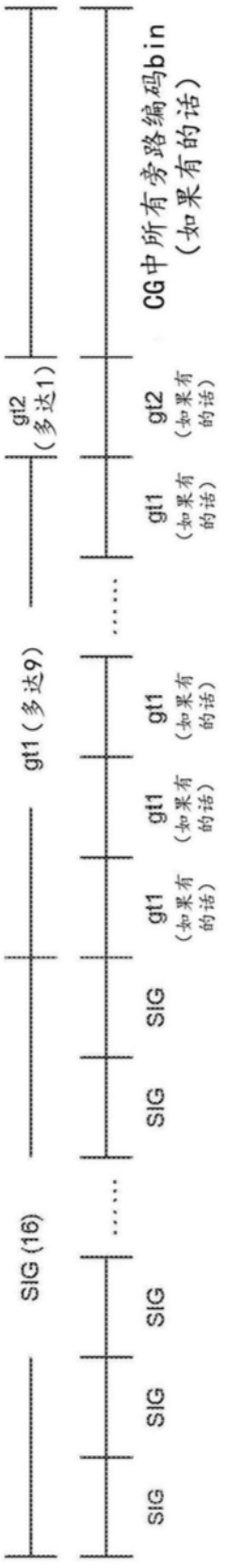

SIG
SIG
SIG
SIG (16)
.....
SIG
SIG
gt1
(如果有的话)
gt1
(如果有的话)
gt1 (多达9)
gt1
(如果有的话)
gt1
(如果有的话)
.....
gt1
(如果有的话)
gt2
(多达1)
gt2
(如果有的话)
CG中所有旁路编码bin
(如果有的话)

图8

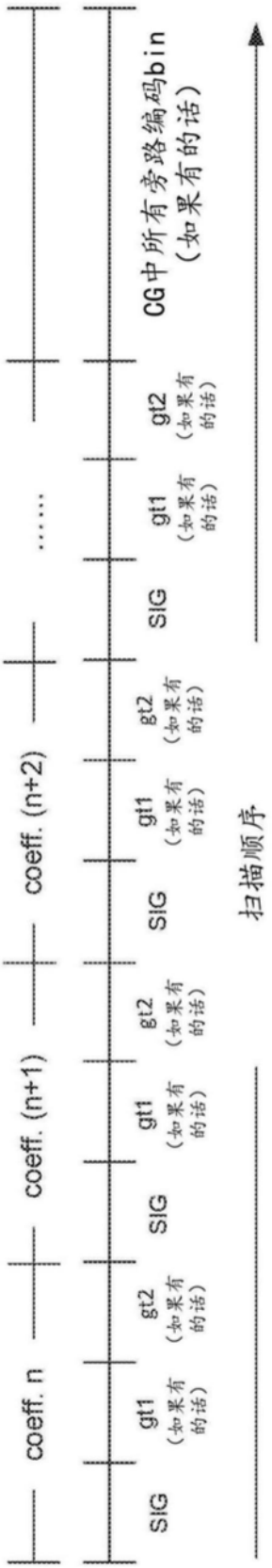

coeff. n
SIG
gt1 (如果有的话)
gt2 (如果有的话)
coeff. (n+1)
SIG
gt1 (如果有的话)
gt2 (如果有的话)
coeff. (n+2)
SIG
gt1 (如果有的话)
gt2 (如果有的话)
.....
SIG
gt1 (如果有的话)
gt2 (如果有的话)
CG中所有旁路编码bin (如果有的话)
扫描顺序

图9

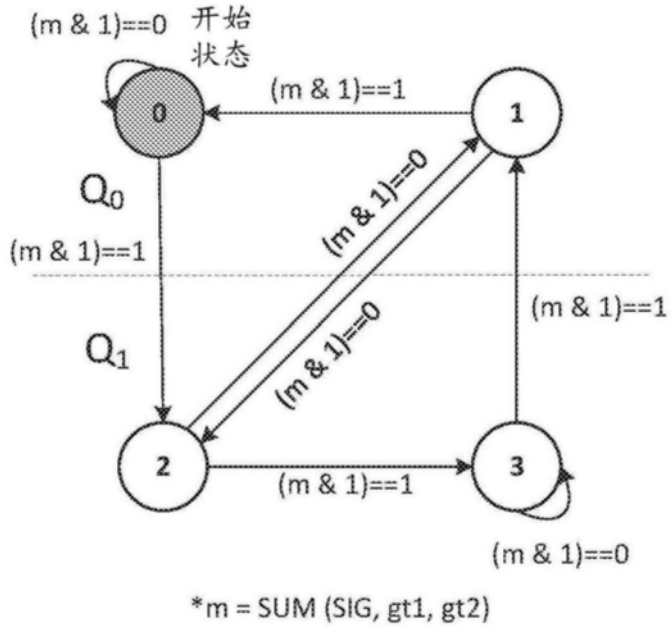

(m & 1)==0
开始
状态
0
(m & 1)==1
1
Q0
(m & 1)==1
(m & 1)==0
(m & 1)==1
Q1
(m & 1)==0
2
(m & 1)==1
3
(m & 1)==0
*m = SUM (SIG, gt1, gt2)

图10

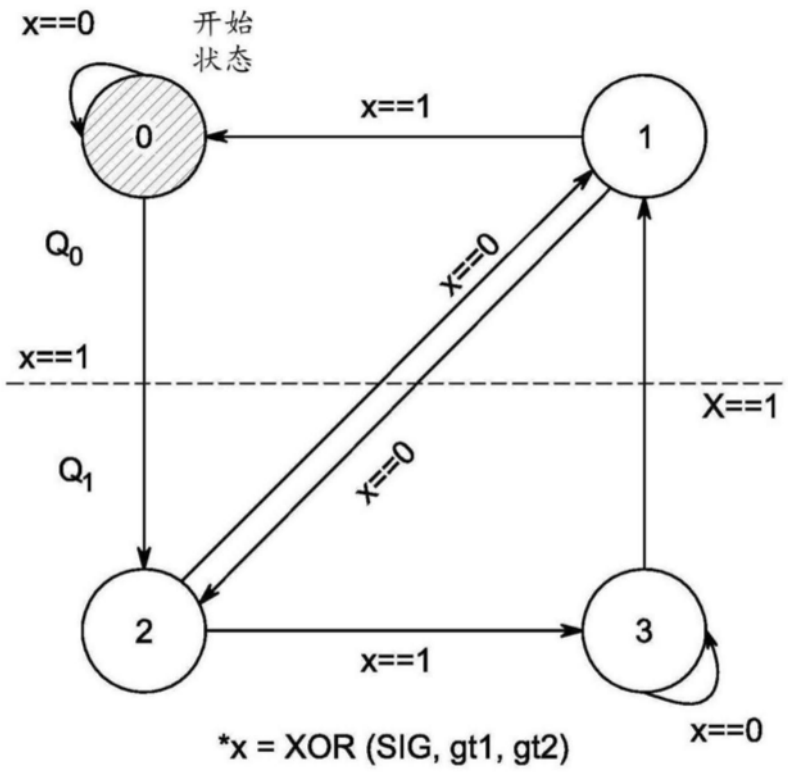

x==0
开始
状态
0
x==1
1
Q0
x==0
x==1
X==1
Q1
x==0
2
x==1
3
x==0
*x = XOR (SIG, gt1, gt2)

图11

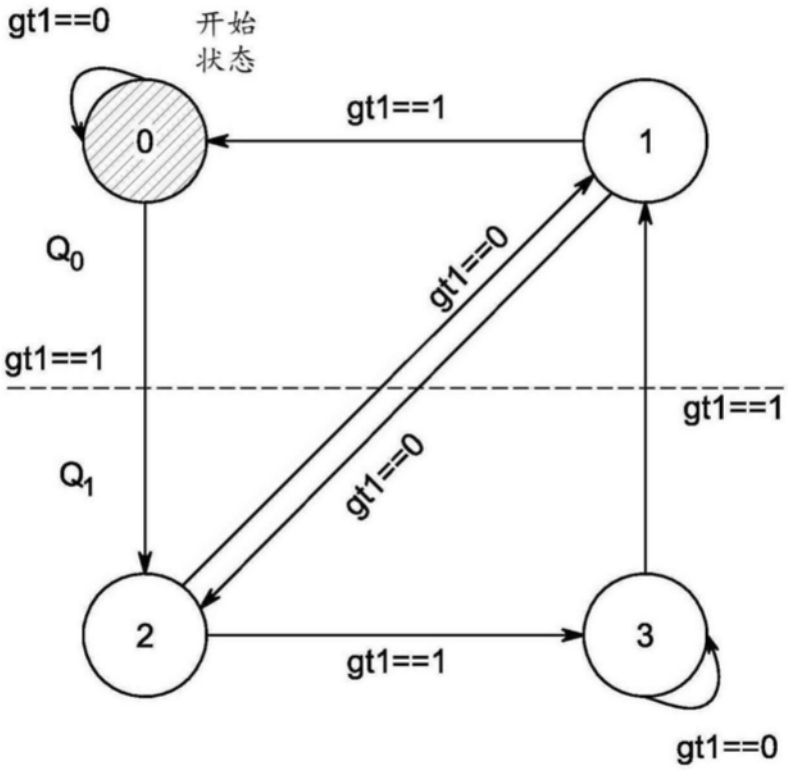


图12

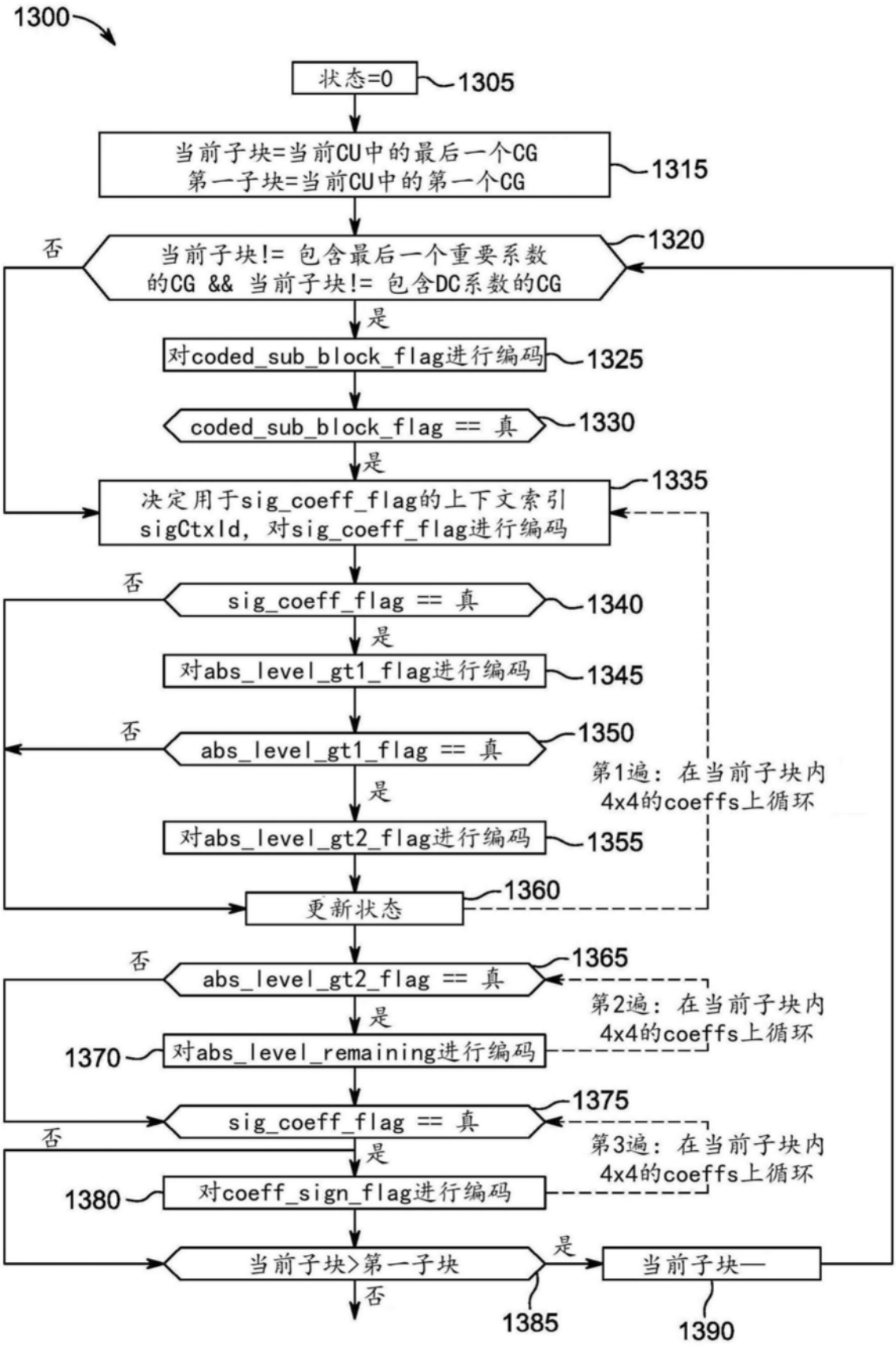

1300
状态=0
1305
当前子块=当前CU中的最后一个CG
第一子块=当前CU中的第一个CG
1315
否
当前子块!= 包含最后一个重要系数的CG && 当前子块!= 包含DC系数的CG
1320
是
对coded_sub_block_flag进行编码
1325
coded_sub_block_flag == 真
1330
是
决定用于sig_coeff_flag的上下文索引sigCtxId，对sig_coeff_flag进行编码
1335
否
sig_coeff_flag == 真
1340
是
对abs_level_gt1_flag进行编码
1345
否
abs_level_gt1_flag == 真
1350
是
对abs_level_gt2_flag进行编码
1355
更新状态
1360
第1遍：在当前子块内4x4的coeffs上循环
否
abs_level_gt2_flag == 真
1365
是
对abs_level_remaining进行编码
1370
第2遍：在当前子块内4x4的coeffs上循环
sig_coeff_flag == 真
1375
否
是
对coeff_sign_flag进行编码
1380
第3遍：在当前子块内4x4的coeffs上循环
当前子块>第一子块
是
当前子块—
否
1385
1390

图13

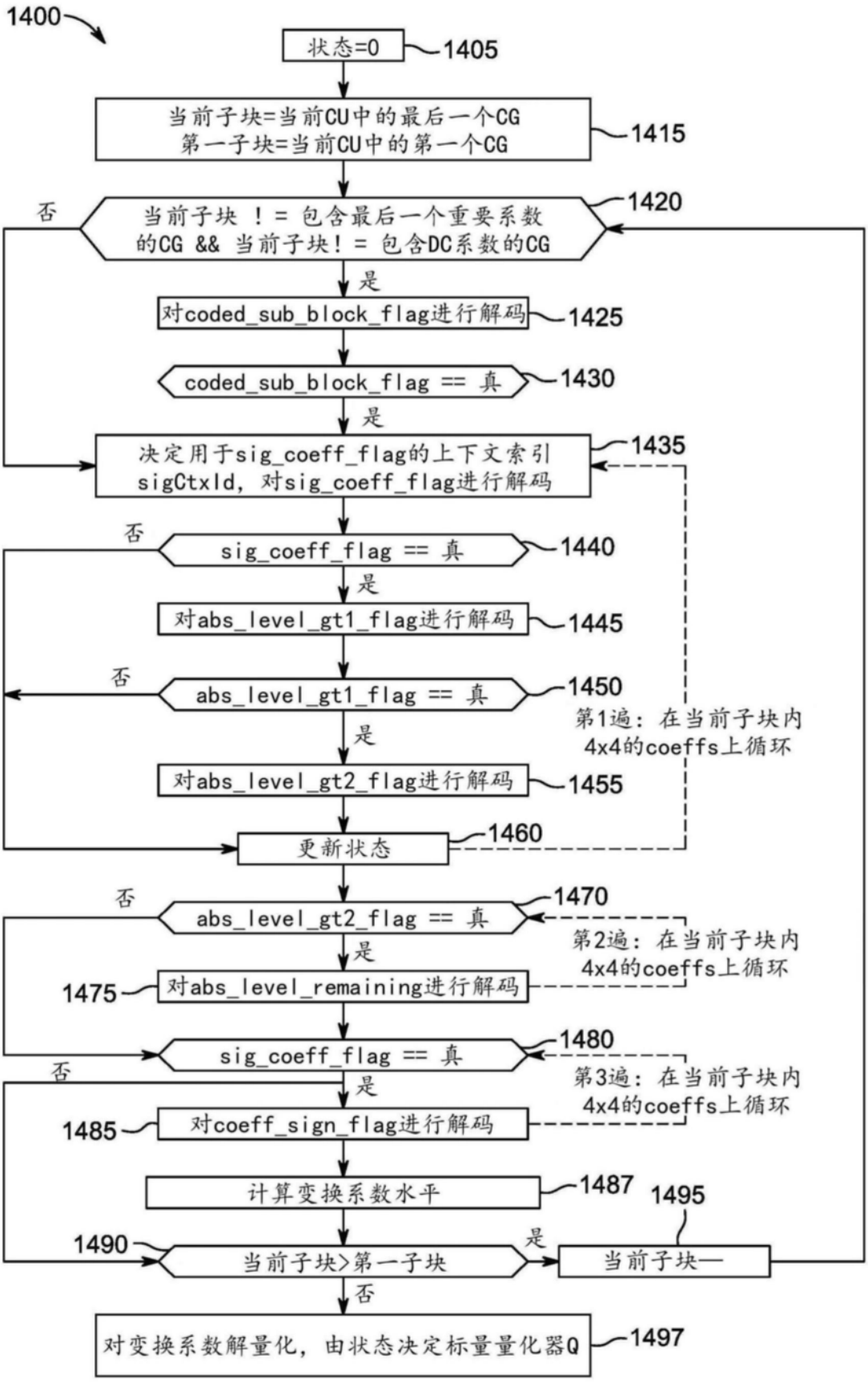

1400
状态=0
1405
当前子块=当前CU中的最后一个CG
第一子块=当前CU中的第一个CG
1415
否
当前子块 ！= 包含最后一个重要系数的CG && 当前子块！= 包含DC系数的CG
1420
是
对coded_sub_block_flag进行解码
1425
coded_sub_block_flag == 真
1430
是
决定用于sig_coeff_flag的上下文索引sigCtxId，对sig_coeff_flag进行解码
1435
否
sig_coeff_flag == 真
1440
是
对abs_level_gt1_flag进行解码
1445
否
abs_level_gt1_flag == 真
1450
第1遍：在当前子块内4x4的coeffs上循环
是
对abs_level_gt2_flag进行解码
1455
更新状态
1460
否
abs_level_gt2_flag == 真
1470
第2遍：在当前子块内4x4的coeffs上循环
是
1475
对abs_level_remaining进行解码
sig_coeff_flag == 真
1480
第3遍：在当前子块内4x4的coeffs上循环
否
是
1485
对coeff_sign_flag进行解码
计算变换系数水平
1487
1495
1490
当前子块>第一子块
是
当前子块—
否
对变换系数解量化，由状态决定标量量化器Q
1497

图14