

DECODING ORDER OF DIFFERENT SEI MESSAGES

CROSS-REFERENCE TO RELATED APPLICATION

[001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of U.S. Provisional Patent Application No. US 62/959,108 filed on January 9, 2020. For all purposes under the law, the entire disclosure of the aforementioned application is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[002] This patent document relates to image and video coding and decoding.

BACKGROUND

[003] Digital video accounts for the largest bandwidth use on the internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, it is expected that the bandwidth demand for digital video usage will continue to grow.

SUMMARY

[004] The present document discloses techniques that can be used by video encoders and decoders for video encoding or decoding, and includes constraints, restrictions and signaling for subpictures, slices, and tiles.

[005] In one example aspect, a video processing method is disclosed. The method includes performing a conversion between a video and a bitstream of the video, wherein the bitstream comprises one or more access units according to a format rule, and wherein the format rule specifies an order in which a first message and a second message that apply to an operation point (OP) are present within an access unit (AU) such that the first message precedes the second message in a decoding order.

[006] In another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video and a bitstream of the video, wherein the bitstream comprises one or more access units according to a format rule, and wherein the format rule

specifies an order in which a plurality of messages that apply to an operation point (OP) are present within an access unit such that a first message of the plurality of messages precedes a second message of the plurality of messages in a decoding order.

[007] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising a picture and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule specifies that an indication of whether a first flag is signaled at a beginning of a picture header associated with the picture, wherein the first flag is indicative of whether the picture is an intra random access point (IRAP) picture or a gradual decoding refresh (GDR) picture.

[008] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising one or more pictures and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule disallows coding of a picture of the one or more pictures to include a coded slice network abstraction layer (NAL) unit having a gradual decoding refresh type and to associate with a flag indicating that the picture contains mixed types of NAL units.

[009] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising one or more pictures and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule allows coding of a picture of the one or more pictures to include a coded slice network access layer (NAL) unit having a gradual decoding refresh type and to associate with a flag indicating that the picture does not contain mixed types of NAL units.

[0010] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a picture of a video and a bitstream of the video, wherein the bitstream conforms to a format rule that specifies whether a first syntax element is signaled in a picture parameter set (PPS) associated with the picture, wherein the picture comprises one or more slices with a slice type, wherein the first syntax element indicates that the slice type is signaled in the picture header due to the first syntax element being equal to zero, and otherwise indicates that the slice type is signaled in a slice header.

[0011] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a picture of a video and a bitstream of the video

according to a rule, wherein the conversion comprises an in-loop filtering process, and wherein the rule specifies that a total number of vertical virtual boundaries and a total number of horizontal virtual boundaries related to the in-loop filtering process are signaled at a picture-level or a sequence-level.

[0012] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising one or more pictures and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule conditionally allows coding of a picture in one layer using reference pictures from other layers based on a first syntax element indicating whether the reference pictures from the other layers are present in the bitstream, and wherein the first syntax element is conditionally signaled in the bitstream based on a second syntax element that indicates whether an identifier of a parameter set associated with the picture is not equal to zero.

[0013] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a picture of a video and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule defines a first syntax element for enabling (a) a synchronization process for context variables before decoding a coding tree unit (CTU) in the picture and (b) a storage process for the context variables after decoding the CTU, wherein the first syntax element is signaled in a sequence parameter set (SPS) associated with the picture.

[0014] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a picture of a video and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule defines a syntax element for indicating whether signaling for entry point offsets for tiles or tile-specific CTU rows are present in a slice header of the picture, and wherein the syntax element is signaled in a sequence parameter set (SPS) associated with the picture.

[0015] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that a first syntax element, which indicates a number of parameters for an output layer set (OLS) hypothetical reference decoder (HRD) in a video parameter set (VPS) associated with the video, is less than a first predetermined threshold.

[0016] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that a syntax element, which indicates a number of profile/tier/level (PTL) syntax structures in a video parameter set (VPS) associated with the video, is less than a predetermined threshold.

[0017] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that a first syntax element, which indicates a number of decoded picture buffer parameters syntax structures in a video parameters set (VPS), must be less than or equal to a second syntax element, which indicates a number of layers specified by the VPS.

[0018] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule allows a terminating network abstraction layer (NAL) unit to be made available to a decoder by signaling in the bitstream or providing through external means.

[0019] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video and a bitstream of the video, wherein the bitstream conforms to a format rule, and wherein the format rule restricts each layer in the bitstream to contain only one subpicture due to a syntax element being equal to zero, which indicates that the each layer is configured to use inter-layer prediction.

[0020] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that a sub-bitstream extraction process is implemented to generate a sub-bitstream for decoding, wherein the sub-bitstream extraction process is configured to extract, from the bitstream, a sub-bitstream with a target highest temporal identifier, and wherein, the rule specifies that, during the extracting, upon removing a video coding layer (VCL) network abstraction layer (NAL) unit, filler data units and filler supplemental enhancement information (SEI) messages in SEI NAL units that are associated with the VCL NAL unit are also removed.

[0021] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video unit of a video and a bitstream of the video,

wherein the bitstream conforms to a format rule, wherein the format rule specifies that the bitstream includes a first syntax element, which indicates whether the video unit is coded in a lossless mode or in a lossy mode, and wherein signaling a second syntax element, which indicates an escape sample in a palette mode applied to the video unit, is selectively included based on a value of the first syntax element.

[0022] In yet another example aspect, a video encoder apparatus is disclosed. The video encoder comprises a processor configured to implement above-described methods.

[0023] In yet another example aspect, a video decoder apparatus is disclosed. The video decoder comprises a processor configured to implement above-described methods.

[0024] In yet another example aspect, a computer readable medium having code stored thereon is disclosed. The code embodies one of the methods described herein in the form of processor-executable code.

[0025] These, and other, features are described throughout the present document.

BRIEF DESCRIPTION OF DRAWINGS

[0026] FIG. 1 shows an example of partitioning a picture with luma coding tree units (CTUs).

[0027] FIG. 2 shows another example of partitioning a picture with luma CTUs.

[0028] FIG. 3 shows an example partitioning of a picture.

[0029] FIG. 4 shows another example partitioning of a picture.

[0030] FIG. 5 is a block diagram of an example video processing system in which disclosed techniques may be implemented.

[0031] FIG. 6 is a block diagram of an example hardware platform used for video processing.

[0032] FIG. 7 is a block diagram that illustrates a video coding system in accordance with some embodiments of the present disclosure.

[0033] FIG. 8 is a block diagram that illustrates an encoder in accordance with some embodiments of the present disclosure.

[0034] FIG. 9 is a block diagram that illustrates a decoder in accordance with some embodiments of the present disclosure.

[0035] FIGS. 10-26 show flowcharts for example methods of video processing.

DETAILED DESCRIPTION

[0036] Section headings are used in the present document for ease of understanding and do not limit the applicability of techniques and embodiments disclosed in each section only to that section. Furthermore, H.266 terminology is used in some description only for ease of understanding and not for limiting scope of the disclosed techniques. As such, the techniques described herein are applicable to other video codec protocols and designs also.

1. Introduction

[0037] This document is related to video coding technologies. Specifically, it is about signaling of subpictures, tiles, and slices. The ideas may be applied individually or in various combination, to any video coding standard or non-standard video codec that supports multi-layer video coding, e.g., the being-developed Versatile Video Coding (VVC).

2. Abbreviations

APS	Adaptation Parameter Set
AU	Access Unit
AUD	Access Unit Delimiter
AVC	Advanced Video Coding
CLVS	Coded Layer Video Sequence
CPB	Coded Picture Buffer
CRA	Clean Random Access
CTU	Coding Tree Unit
CVS	Coded Video Sequence
DPB	Decoded Picture Buffer
DPS	Decoding Parameter Set
EOB	End Of Bitstream
EOS	End Of Sequence
GDR	Gradual Decoding Refresh
HEVC	High Efficiency Video Coding
HRD	Hypothetical Reference Decoder
IDR	Instantaneous Decoding Refresh
JEM	Joint Exploration Model

MCTS	Motion-Constrained Tile Sets
NAL	Network Abstraction Layer
OLS	Output Layer Set
PH	Picture Header
PPS	Picture Parameter Set
PTL	Profile, Tier and Level
PU	Picture Unit
RBSP	Raw Byte Sequence Payload
SEI	Supplemental Enhancement Information
SPS	Sequence Parameter Set
SVC	Scalable Video Coding
VCL	Video Coding Layer
VPS	Video Parameter Set
VTM	VVC Test Model
VUI	Video Usability Information
VVC	Versatile Video Coding

3. Initial Discussion

[0038] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. To explore the future video coding technologies beyond HEVC, the Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM). The JVET meeting is concurrently held once every quarter, and the new coding standard is targeting at 50% bitrate reduction as compared to HEVC. The new video coding standard was officially named as Versatile Video Coding (VVC) in the April 2018 JVET meeting, and the first version of VVC test model (VTM) was released at that time. As there are continuous effort contributing

to VVC standardization, new coding techniques are being adopted to the VVC standard in every JVET meeting. The VVC working draft and test model VTM are then updated after every meeting. The VVC project is now aiming for technical completion (FDIS) at the July 2020 meeting.

3.1. Picture partitioning schemes in HEVC

[0039] HEVC includes four different picture partitioning schemes, namely regular slices, dependent slices, tiles, and Wavefront Parallel Processing (WPP), which may be applied for Maximum Transfer Unit (MTU) size matching, parallel processing, and reduced end-to-end delay.

[0040] Regular slices are similar as in H.264/AVC. Each regular slice is encapsulated in its own NAL unit, and in-picture prediction (intra sample prediction, motion information prediction, coding mode prediction) and entropy coding dependency across slice boundaries are disabled. Thus a regular slice can be reconstructed independently from other regular slices within the same picture (though there may still have interdependencies due to loop filtering operations).

[0041] The regular slice is the only tool that can be used for parallelization that is also available, in virtually identical form, in H.264/AVC. Regular slices based parallelization does not require much inter-processor or inter-core communication (except for inter-processor or inter-core data sharing for motion compensation when decoding a predictively coded picture, which is typically much heavier than inter-processor or inter-core data sharing due to in-picture prediction). However, for the same reason, the use of regular slices can incur substantial coding overhead due to the bit cost of the slice header and due to the lack of prediction across the slice boundaries. Further, regular slices (in contrast to the other tools mentioned below) also serve as the key mechanism for bitstream partitioning to match MTU size requirements, due to the in-picture independence of regular slices and that each regular slice is encapsulated in its own NAL unit. In many cases, the goal of parallelization and the goal of MTU size matching place contradicting demands to the slice layout in a picture. The realization of this situation led to the development of the parallelization tools mentioned below.

[0042] Dependent slices have short slice headers and allow partitioning of the bitstream at treeblock boundaries without breaking any in-picture prediction. Basically, dependent slices provide fragmentation of regular slices into multiple NAL units, to provide reduced end-to-end

delay by allowing a part of a regular slice to be sent out before the encoding of the entire regular slice is finished.

[0043] In WPP, the picture is partitioned into single rows of coding tree blocks (CTBs). Entropy decoding and prediction are allowed to use data from CTBs in other partitions. Parallel processing is possible through parallel decoding of CTB rows, where the start of the decoding of a CTB row is delayed by two CTBs, so to ensure that data related to a CTB above and to the right of the subject CTB is available before the subject CTB is being decoded. Using this staggered start (which appears like a wavefront when represented graphically), parallelization is possible with up to as many processors/cores as the picture contains CTB rows. Because in-picture prediction between neighboring treeblock rows within a picture is permitted, the required inter-processor/inter-core communication to enable in-picture prediction can be substantial. The WPP partitioning does not result in the production of additional NAL units compared to when it is not applied, thus WPP is not a tool for MTU size matching. However, if MTU size matching is required, regular slices can be used with WPP, with certain coding overhead.

[0044] Tiles define horizontal and vertical boundaries that partition a picture into tile columns and rows. Tile column runs from the top of a picture to the bottom of the picture. Likewise, tile row runs from the left of the picture to the right of the picture. The number of tiles in a picture can be derived simply as number of tile columns multiply by number of tile rows.

[0045] The scan order of CTBs is changed to be local within a tile (in the order of a CTB raster scan of a tile), before decoding the top-left CTB of the next tile in the order of tile raster scan of a picture. Similar to regular slices, tiles break in-picture prediction dependencies as well as entropy decoding dependencies. However, they do not need to be included into individual NAL units (same as WPP in this regard); hence tiles cannot be used for MTU size matching. Each tile can be processed by one processor/core, and the inter-processor/inter-core communication required for in-picture prediction between processing units decoding neighboring tiles is limited to conveying the shared slice header in cases a slice is spanning more than one tile, and loop filtering related sharing of reconstructed samples and metadata. When more than one tile or WPP segment is included in a slice, the entry point byte offset for each tile or WPP segment other than the first one in the slice is signaled in the slice header.

[0046] For simplicity, restrictions on the application of the four different picture partitioning schemes have been specified in HEVC. A given coded video sequence cannot include both tiles and wavefronts for most of the profiles specified in HEVC. For each slice and tile, either or both of the following conditions must be fulfilled: 1) all coded treeblocks in a slice belong to the same tile; 2) all coded treeblocks in a tile belong to the same slice. Finally, a wavefront segment contains exactly one CTB row, and when WPP is in use, if a slice starts within a CTB row, it must end in the same CTB row.

[0047] A recent amendment to HEVC is specified in the JCT-VC output document JCTVC-AC1005, J. Boyce, A. Ramasubramanian, R. Skupin, G. J. Sullivan, A. Tourapis, Y.-K. Wang (editors), "HEVC Additional Supplemental Enhancement Information (Draft 4)," Oct. 24, 2017, publicly available herein: http://phenix.int-evry.fr/jct/doc_end_user/documents/29_Macau/wg11/JCTVC-AC1005-v2.zip. With this amendment included, HEVC specifies three MCTS-related SEI messages, namely temporal MCTSs SEI message, MCTSs extraction information set SEI message, and MCTSs extraction information nesting SEI message.

[0048] The temporal MCTSs SEI message indicates existence of MCTSs in the bitstream and signals the MCTSs. For each MCTS, motion vectors are restricted to point to full-sample locations inside the MCTS and to fractional-sample locations that require only full-sample locations inside the MCTS for interpolation, and the usage of motion vector candidates for temporal motion vector prediction derived from blocks outside the MCTS is disallowed. This way, each MCTS may be independently decoded without the existence of tiles not included in the MCTS.

[0049] The MCTSs extraction information sets SEI message provides supplemental information that can be used in the MCTS sub-bitstream extraction (specified as part of the semantics of the SEI message) to generate a conforming bitstream for an MCTS set. The information consists of a number of extraction information sets, each defining a number of MCTS sets and containing RBSP bytes of the replacement VPSs, SPSs, and PPSs to be used during the MCTS sub-bitstream extraction process. When extracting a sub-bitstream according to the MCTS sub-bitstream extraction process, parameter sets (VPSs, SPSs, and PPSs) need to be rewritten or replaced, slice headers need to be slightly updated because one or all of the slice address related

syntax elements (including `first_slice_segment_in_pic_flag` and `slice_segment_address`) typically would need to have different values.

3.2. Partitioning of pictures in VVC

[0050] In VVC, A picture is divided into one or more tile rows and one or more tile columns. A tile is a sequence of CTUs that covers a rectangular region of a picture. The CTUs in a tile are scanned in raster scan order within that tile.

[0051] A slice consists of an integer number of complete tiles or an integer number of consecutive complete CTU rows within a tile of a picture.

[0052] Two modes of slices are supported, namely the raster-scan slice mode and the rectangular slice mode. In the raster-scan slice mode, a slice contains a sequence of complete tiles in a tile raster scan of a picture. In the rectangular slice mode, a slice contains either a number of complete tiles that collectively form a rectangular region of the picture or a number of consecutive complete CTU rows of one tile that collectively form a rectangular region of the picture. Tiles within a rectangular slice are scanned in tile raster scan order within the rectangular region corresponding to that slice.

[0053] A subpicture contains one or more slices that collectively cover a rectangular region of a picture.

[0054] FIG. 1 shows an example of raster-scan slice partitioning of a picture, where the picture is divided into 12 tiles and 3 raster-scan slices.

[0055] FIG. 2 shows an example of rectangular slice partitioning of a picture, where the picture is divided into 24 tiles (6 tile columns and 4 tile rows) and 9 rectangular slices.

[0056] FIG. 3 shows an example of a picture partitioned into tiles and rectangular slices, where the picture is divided into 4 tiles (2 tile columns and 2 tile rows) and 4 rectangular slices.

[0057] FIG. 4 shows an example of subpicture partitioning of a picture, where a picture is partitioned into 18 tiles, 12 on the left-hand side each covering one slice of 4 by 4 CTUs and 6 tiles on the right-hand side each covering 2 vertically-stacked slices of 2 by 2 CTUs, altogether resulting in 24 slices and 24 subpictures of varying dimensions (each slice is a subpicture).

3.3. Signaling of subpictures, tiles, and slices in VVC

[0058] In the latest VVC draft text, information of subpictures, includes subpicture layout (i.e., the number of subpictures for each picture and the position and size of each picture) and other

sequence-level subpicture information, is signaled in the SPS. The order of subpictures signaled in the SPS defines the subpicture index. A list of subpicture IDs, one for each subpicture, may be explicitly signaled, e.g., in the SPS or in the PPS.

[0059] Tiles in VVC are conceptually the same as in HEVC, i.e., each picture is partitioned into tile columns and tile rows, but with different syntax in the PPS for signaling of tiles.

[0060] In VVC, the slice mode is also signaled in the PPS. When the slice mode is the rectangular slice mode, the slice layout (i.e., the number of slices for each picture and the position and size of each slice) for each picture is signaled in the PPS. The order of the rectangular slices within a picture signaled in the PPS defines the picture-level slice index. The subpicture-level slice index is defined as the order of the slices within a subpicture in increasing order of the picture-level slice indices. The positions and sizes of the rectangular slices are signaled/derived based on either the subpicture positions and sizes that are signaled in the SPS (when each subpicture contains only one slice), or based on the tile positions and sizes that are signaled in the PPS (when a subpicture may contain more than one slice). When the slice mode is the raster-scan slice mode, similarly as in HEVC, the layout of slices within a picture is signaled in the slices themselves, with different details.

[0061] The SPS, PPS, and slice header syntax and semantics in the latest VVC draft text that are most relevant to the inventions herein are as follows.

7.3.2.3 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	Descriptor
...	
subpics_present_flag	u(1)
if(subpics_present_flag) {	
sps_num_subpics_minus1	u(8)
for(i = 0; i <= sps_num_subpics_minus1; i++) {	
subpic_ctu_top_left_x[i]	u(v)
subpic_ctu_top_left_y[i]	u(v)
subpic_width_minus1[i]	u(v)
subpic_height_minus1[i]	u(v)
subpic_treated_as_pic_flag[i]	u(1)
loop_filter_across_subpic_enabled_flag[i]	u(1)

}	
}	
sps_subpic_id_present_flag	u(1)
if(sps_subpics_id_present_flag) {	
sps_subpic_id_signalling_present_flag	u(1)
if(sps_subpics_id_signalling_present_flag) {	
sps_subpic_id_len_minus1	ue(v)
for(i = 0; i <= sps_num_subpics_minus1; i++)	
sps_subpic_id[i]	u(v)
}	
}	
...	
}	

7.4.3.3 Sequence parameter set RBSP semantics

...

subpics_present_flag equal to 1 specifies that subpicture parameters are present in the SPS RBSP syntax. **subpics_present_flag** equal to 0 specifies that subpicture parameters are not present in the SPS RBSP syntax.

NOTE 2 – When a bitstream is the result of a sub-bitstream extraction process and contains only a subset of the subpictures of the input bitstream to the sub-bitstream extraction process, it might be required to set the value of **subpics_present_flag** equal to 1 in the RBSP of the SPSs.

sps_num_subpics_minus1 plus 1 specifies the number of subpictures. **sps_num_subpics_minus1** shall be in the range of 0 to 254. When not present, the value of **sps_num_subpics_minus1** is inferred to be equal to 0.

subpic_ctu_top_left_x[i] specifies horizontal position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of **subpic_ctu_top_left_x[i]** is inferred to be equal to 0.

subpic_ctu_top_left_y[i] specifies vertical position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of **subpic_ctu_top_left_y[i]** is inferred to be equal to 0.

subpic_width_minus1[*i*] plus 1 specifies the width of the *i*-th subpicture in units of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of **subpic_width_minus1**[*i*] is inferred to be equal to $\text{Ceil}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}) - 1$.

subpic_height_minus1[*i*] plus 1 specifies the height of the *i*-th subpicture in units of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of **subpic_height_minus1**[*i*] is inferred to be equal to $\text{Ceil}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}) - 1$.

subpic_treated_as_pic_flag[*i*] equal to 1 specifies that the *i*-th subpicture of each coded picture in the CLVS is treated as a picture in the decoding process excluding in-loop filtering operations.

subpic_treated_as_pic_flag[*i*] equal to 0 specifies that the *i*-th subpicture of each coded picture in the CLVS is not treated as a picture in the decoding process excluding in-loop filtering operations. When not present, the value of **subpic_treated_as_pic_flag**[*i*] is inferred to be equal to 0.

loop_filter_across_subpic_enabled_flag[*i*] equal to 1 specifies that in-loop filtering operations may be performed across the boundaries of the *i*-th subpicture in each coded picture in the CLVS.

loop_filter_across_subpic_enabled_flag[*i*] equal to 0 specifies that in-loop filtering operations are not performed across the boundaries of the *i*-th subpicture in each coded picture in the CLVS. When not present, the value of **loop_filter_across_subpic_enabled_flag**[*i*] is inferred to be equal to 1.

It is a requirement of bitstream conformance that the following constraints apply:

- For any two subpictures subpicA and subpicB, when the subpicture index of subpicA is less than that of subpicB, any coded slice NAL unit of subPicA shall precede any coded slice NAL unit of subPicB in decoding order.
- The shapes of the subpictures shall be such that each subpicture, when decoded, shall have its entire left boundary and entire top boundary consisting of picture boundaries or consisting of boundaries of previously decoded subpictures.

sps_subpic_id_present_flag equal to 1 specifies that subpicture ID mapping is present in the SPS.

sps_subpic_id_present_flag equal to 0 specifies that subpicture ID mapping is not present in the SPS.

sps_subpic_id_signalling_present_flag equal to 1 specifies that subpicture ID mapping is signalled in the SPS. **sps_subpic_id_signalling_present_flag** equal to 0 specifies that subpicture ID mapping is not signalled in the SPS. When not present, the value of **sps_subpic_id_signalling_present_flag** is inferred to be equal to 0.

sps_subpic_id_len_minus1 plus 1 specifies the number of bits used to represent the syntax element **sps_subpic_id**[*i*]. The value of **sps_subpic_id_len_minus1** shall be in the range of 0 to 15, inclusive.

sps_subpic_id[i] specifies that subpicture ID of the i-th subpicture. The length of the **sps_subpic_id[i]** syntax element is **sps_subpic_id_len_minus1 + 1** bits. When not present, and when **sps_subpic_id_present_flag** equal to 0, the value of **sps_subpic_id[i]** is inferred to be equal to i, for each i in the range of 0 to **sps_num_subpics_minus1**, inclusive

...

7.3.2.4 Picture parameter set RBSP syntax

pic_parameter_set_rbsp() {	Descriptor
...	
mixed_nalu_types_in_pic_flag	u(1)
pps_subpic_id_signalling_present_flag	u(1)
if(pps_subpics_id_signalling_present_flag) {	
pps_num_subpics_minus1	ue(v)
pps_subpic_id_len_minus1	ue(v)
for(i = 0; i <= pps_num_subpic_minus1; i++)	
pps_subpic_id[i]	u(v)
}	
no_pic_partition_flag	u(1)
if(!no_pic_partition_flag) {	
pps_log2_ctu_size_minus5	u(2)
num_exp_tile_columns_minus1	ue(v)
num_exp_tile_rows_minus1	ue(v)
for(i = 0; i <= num_exp_tile_columns_minus1; i++)	
tile_column_width_minus1[i]	ue(v)
for(i = 0; i <= num_exp_tile_rows_minus1; i++)	
tile_row_height_minus1[i]	ue(v)
rect_slice_flag	u(1)
if(rect_slice_flag)	
single_slice_per_subpic_flag	u(1)
if(rect_slice_flag && !single_slice_per_subpic_flag) {	
num_slices_in_pic_minus1	ue(v)
tile_idx_delta_present_flag	u(1)
for(i = 0; i < num_slices_in_pic_minus1; i++) {	

slice_width_in_tiles_minus1[i]	ue(v)
slice_height_in_tiles_minus1[i]	ue(v)
if(slice_width_in_tiles_minus1[i] == 0 && slice_height_in_tiles_minus1[i] == 0) {	
num_slices_in_tile_minus1[i]	ue(v)
numSlicesInTileMinus1 = num_slices_in_tile_minus1[i]	
for(j = 0; j < numSlicesInTileMinus1; j++)	
slice_height_in_ctu_minus1[i++]	ue(v)
}	
if(tile_idx_delta_present_flag && i < num_slices_in_pic_minus1)	
tile_idx_delta[i]	se(v)
}	
}	
loop_filter_across_tiles_enabled_flag	u(1)
loop_filter_across_slices_enabled_flag	u(1)
}	
...	u(1)
}	

7.4.3.4 Picture parameter set RBSP semantics

...

pps_subpic_id_signalling_present_flag equal to 1 specifies that subpicture ID mapping is signaled in the PPS. **pps_subpic_id_signalling_present_flag** equal to 0 specifies that subpicture ID mapping is not signalled in the PPS. When **sps_subpic_id_present_flag** is 0 or **sps_subpic_id_signalling_present_flag** is equal to 1, **pps_subpic_id_signalling_present_flag** shall be equal to 0.

pps_num_subpics_minus1 plus 1 specifies the number of subpictures in the coded pictures referring to the PPS.

It is a requirement of bitstream conformance that the value of **pps_num_subpic_minus1** shall be equal to **sps_num_subpics_minus1**.

pps_subpic_id_len_minus1 plus 1 specifies the number of bits used to represent the syntax element **pps_subpic_id[i]**. The value of **pps_subpic_id_len_minus1** shall be in the range of 0 to 15, inclusive.

It is a requirement of bitstream conformance that the value of **pps_subpic_id_len_minus1** shall be the same for all PPSs that are referred to by coded pictures in a CLVS.

pps_subpic_id[i] specifies the subpicture ID of the i-th subpicture. The length of the pps_subpic_id[i] syntax element is pps_subpic_id_len_minus1 + 1 bits.

no_pic_partition_flag equal to 1 specifies that no picture partitioning applied to each picture referring to the PPS. no_pic_partition_flag equal to 0 specifies each picture referring to the PPS may be partitioned into more than one tile or slice.

It is a requirement of bitstream conformance that the value of no_pic_partition_flag shall be the same for all PPSs that are referred to by coded pictures within a CLVS.

It is a requirement of bitstream conformance that the value of no_pic_partition_flag shall not be equal to 1 when the value of sps_num_subpics_minus1 + 1 is greater than 1.

pps_log2_ctu_size_minus5 plus 5 specifies the luma coding tree block size of each CTU.

pps_log2_ctu_size_minus5 shall be equal to sps_log2_ctu_size_minus5.

num_exp_tile_columns_minus1 plus 1 specifies the number of explicitly provided tile column widths.

The value of num_exp_tile_columns_minus1 shall be in the range of 0 to PicWidthInCtbsY - 1, inclusive. When no_pic_partition_flag is equal to 1, the value of num_exp_tile_columns_minus1 is inferred to be equal to 0.

num_exp_tile_rows_minus1 plus 1 specifies the number of explicitly provided tile row heights. The value of num_exp_tile_rows_minus1 shall be in the range of 0 to PicHeightInCtbsY - 1, inclusive. When no_pic_partition_flag is equal to 1, the value of num_exp_tile_rows_minus1 is inferred to be equal to 0.

tile_column_width_minus1[i] plus 1 specifies the width of the i-th tile column in units of CTBs for i in the range of 0 to num_exp_tile_columns_minus1 - 1, inclusive.

tile_column_width_minus1[num_exp_tile_columns_minus1] is used to derive the width of the tile columns with index greater than or equal to num_exp_tile_columns_minus1 as specified in clause 6.5.1.

When not present, the value of tile_column_width_minus1[0] is inferred to be equal to PicWidthInCtbsY - 1.

tile_row_height_minus1[i] plus 1 specifies the height of the i-th tile row in units of CTBs for i in the range of 0 to num_exp_tile_rows_minus1 - 1, inclusive.

tile_row_height_minus1[num_exp_tile_rows_minus1] is used to derive the height of the tile rows with index greater than or equal to num_exp_tile_rows_minus1 as specified in clause 6.5.1. When not present, the value of tile_row_height_minus1[0] is inferred to be equal to PicHeightInCtbsY - 1.

rect_slice_flag equal to 0 specifies that tiles within each slice are in raster scan order and the slice information is not signaled in PPS. rect_slice_flag equal to 1 specifies that tiles within each slice cover a rectangular region of the picture and the slice information is signaled in the PPS. When not present,

rect_slice_flag is inferred to be equal to 1. When **subpics_present_flag** is equal to 1, the value of **rect_slice_flag** shall be equal to 1.

single_slice_per_subpic_flag equal to 1 specifies that each subpicture consists of one and only one rectangular slice. **single_slice_per_subpic_flag** equal to 0 specifies that each subpicture may consist one or more rectangular slices. When **subpics_present_flag** is equal to 0, **single_slice_per_subpic_flag** shall be equal to 0. When **single_slice_per_subpic_flag** is equal to 1, **num_slices_in_pic_minus1** is inferred to be equal to **sps_num_subpics_minus1**.

num_slices_in_pic_minus1 plus 1 specifies the number of rectangular slices in each picture referring to the PPS. The value of **num_slices_in_pic_minus1** shall be in the range of 0 to **MaxSlicesPerPicture** – 1, inclusive, where **MaxSlicesPerPicture** is specified in Annex A. When **no_pic_partition_flag** is equal to 1, the value of **num_slices_in_pic_minus1** is inferred to be equal to 0.

tile_idx_delta_present_flag equal to 0 specifies that **tile_idx_delta** values are not present in the PPS and that all rectangular slices in pictures referring to the PPS are specified in raster order according to the process defined in clause 6.5.1. **tile_idx_delta_present_flag** equal to 1 specifies that **tile_idx_delta** values may be present in the PPS and that all rectangular slices in pictures referring to the PPS are specified in the order indicated by the values of **tile_idx_delta**.

slice_width_in_tiles_minus1[i] plus 1 specifies the width of the i-th rectangular slice in units of tile columns. The value of **slice_width_in_tiles_minus1[i]** shall be in the range of 0 to **NumTileColumns** – 1, inclusive. When not present, the value of **slice_width_in_tiles_minus1[i]** is inferred as specified in clause 6.5.1.

slice_height_in_tiles_minus1[i] plus 1 specifies the height of the i-th rectangular slice in units of tile rows. The value of **slice_height_in_tiles_minus1[i]** shall be in the range of 0 to **NumTileRows** – 1, inclusive. When not present, the value of **slice_height_in_tiles_minus1[i]** is inferred as specified in clause 6.5.1.

num_slices_in_tile_minus1[i] plus 1 specifies the number of slices in the current tile for the case where the i-th slice contains a subset of CTU rows from a single tile. The value of **num_slices_in_tile_minus1[i]** shall be in the range of 0 to **RowHeight[tileY]** – 1, inclusive, where **tileY** is the tile row index containing the i-th slice. When not present, the value of **num_slices_in_tile_minus1[i]** is inferred to be equal to 0.

slice_height_in_ctu_minus1[i] plus 1 specifies the height of the i-th rectangular slice in units of CTU rows for the case where the i-th slice contains a subset of CTU rows from a single tile. The value of **slice_height_in_ctu_minus1[i]** shall be in the range of 0 to **RowHeight[tileY]** – 1, inclusive, where **tileY** is the tile row index containing the i-th slice.

tile_idx_delta[i] specifies the difference in tile index between the i-th rectangular slice and the (i + 1)-th rectangular slice. The value of **tile_idx_delta[i]** shall be in the range of $-\text{NumTilesInPic} + 1$ to $\text{NumTilesInPic} - 1$, inclusive. When not present, the value of **tile_idx_delta[i]** is inferred to be equal to 0. In all other cases, the value of **tile_idx_delta[i]** shall not be equal to 0.

loop_filter_across_tiles_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across tile boundaries in pictures referring to the PPS. **loop_filter_across_tiles_enabled_flag** equal to 0 specifies that in-loop filtering operations are not performed across tile boundaries in pictures referring to the PPS. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, the value of **loop_filter_across_tiles_enabled_flag** is inferred to be equal to 1.

loop_filter_across_slices_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across slice boundaries in pictures referring to the PPS. **loop_filter_across_slice_enabled_flag** equal to 0 specifies that in-loop filtering operations are not performed across slice boundaries in pictures referring to the PPS. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, the value of **loop_filter_across_slices_enabled_flag** is inferred to be equal to 0.

...

7.3.7.1 General slice header syntax

slice_header() {	Descriptor
...	
if(subpics_present_flag)	
slice_subpic_id	u(v)
if(rect_slice_flag NumTilesInPic > 1)	
slice_address	u(v)
if(!rect_slice_flag && NumTilesInPic > 1)	
num_tiles_in_slice_minus1	ue(v)
slice_type	ue(v)
...	
}	

7.4.8.1 General slice header semantics

...

slice_subpic_id specifies the subpicture identifier of the subpicture that contains the slice. If **slice_subpic_id** is present, the value of the variable **SubPicIdx** is derived to be such that **SubpicIdList[SubPicIdx]** is equal to **slice_subpic_id**. Otherwise (**slice_subpic_id** is not present), the variable **SubPicIdx** is derived to be equal to 0. The length of **slice_subpic_id**, in bits, is derived as follows:

- If **sps_subpic_id_signalling_present_flag** is equal to 1, the length of **slice_subpic_id** is equal to **sps_subpic_id_len_minus1 + 1**.
- Otherwise, if **ph_subpic_id_signalling_present_flag** is equal to 1, the length of **slice_subpic_id** is equal to **ph_subpic_id_len_minus1 + 1**.
- Otherwise, if **pps_subpic_id_signalling_present_flag** is equal to 1, the length of **slice_subpic_id** is equal to **pps_subpic_id_len_minus1 + 1**.
- Otherwise, the length of **slice_subpic_id** is equal to $\text{Ceil}(\text{Log}_2(\text{sps_num_subpics_minus1} + 1))$.

slice_address specifies the slice address of the slice. When not present, the value of **slice_address** is inferred to be equal to 0.

If **rect_slice_flag** is equal to 0, the following applies:

- The slice address is the raster scan tile index.
- The length of **slice_address** is $\text{Ceil}(\text{Log}_2(\text{NumTilesInPic}))$ bits.
- The value of **slice_address** shall be in the range of 0 to **NumTilesInPic** – 1, inclusive.

Otherwise (**rect_slice_flag** is equal to 1), the following applies:

- The slice address is the slice index of the slice within the **SubPicIdx**-th subpicture.
- The length of **slice_address** is $\text{Ceil}(\text{Log}_2(\text{NumSlicesInSubpic[SubPicIdx]}))$ bits.
- The value of **slice_address** shall be in the range of 0 to **NumSlicesInSubpic[SubPicIdx]** – 1, inclusive.

It is a requirement of bitstream conformance that the following constraints apply:

- If **rect_slice_flag** is equal to 0 or **subpics_present_flag** is equal to 0, the value of **slice_address** shall not be equal to the value of **slice_address** of any other coded slice NAL unit of the same coded picture.
- Otherwise, the pair of **slice_subpic_id** and **slice_address** values shall not be equal to the pair of **slice_subpic_id** and **slice_address** values of any other coded slice NAL unit of the same coded picture.
- When **rect_slice_flag** is equal to 0, the slices of a picture shall be in increasing order of their **slice_address** values.

- The shapes of the slices of a picture shall be such that each CTU, when decoded, shall have its entire left boundary and entire top boundary consisting of a picture boundary or consisting of boundaries of previously decoded CTU(s).

num_tiles_in_slice_minus1 plus 1, when present, specifies the number of tiles in the slice. The value of **num_tiles_in_slice_minus1** shall be in the range of 0 to **NumTilesInPic** – 1, inclusive.

The variable **NumCtuInCurrSlice**, which specifies the number of CTUs in the current slice, and the list **CtbAddrInCurrSlice[i]**, for **i** ranging from 0 to **NumCtuInCurrSlice** – 1, inclusive, specifying the picture raster scan address of the **i**-th CTB within the slice, are derived as follows:

```

if( rect_slice_flag ) {
    picLevelSliceIdx = SliceSubpicToPicIdx[ SubPicIdx ][ slice_address ]
    NumCtuInCurrSlice = NumCtuInSlice[ picLevelSliceIdx ]
    for( i = 0; i < NumCtuInCurrSlice; i++ )
        CtbAddrInCurrSlice[ i ] = CtbAddrInSlice[ picLevelSliceIdx ][ i ]
} else {
    NumCtuInCurrSlice = 0
    for( tileIdx = slice_address; tileIdx <= slice_address + num_tiles_in_slice_minus1[ i ];
        tileIdx++ ) {
        tileX = tileIdx % NumTileColumns
        tileY = tileIdx / NumTileColumns
        for( ctbY = tileRowBd[ tileY ]; ctbY < tileRowBd[ tileY + 1 ]; ctbY++ ) {
            for( ctbX = tileColBd[ tileX ]; ctbX < tileColBd[ tileX + 1 ]; ctbX++ ) {
                CtbAddrInCurrSlice[ NumCtuInCurrSlice ] = ctbY * PicWidthInCtb + ctbX
                NumCtuInCurrSlice++
            }
        }
    }
}

```

(115)

The variables **SubPicLeftBoundaryPos**, **SubPicTopBoundaryPos**, **SubPicRightBoundaryPos**, and **SubPicBotBoundaryPos** are derived as follows:

```

if( subpic_treated_as_pic_flag[ SubPicIdx ] ) {
    SubPicLeftBoundaryPos = subpic_ctu_top_left_x[ SubPicIdx ] * CtbSizeY
    SubPicRightBoundaryPos = Min( pic_width_max_in_luma_samples – 1,

```

```

    ( subpic_ctu_top_left_x[ SubPicIdx ] + subpic_width_minus1[ SubPicIdx ] + 1 ) * CtbSizeY
- 1)
    SubPicTopBoundaryPos = subpic_ctu_top_left_y[ SubPicIdx ] * CtbSizeY
    SubPicBotBoundaryPos = Min( pic_height_max_in_luma_samples - 1,
    ( subpic_ctu_top_left_y[ SubPicIdx ] + subpic_height_minus1[ SubPicIdx ] + 1 ) * CtbSizeY
- 1)
}

```

...

3.4. Embodiments from JVET-Q0075

[0062] Escape samples are employed to handle the outlier case in the palette mode.

[0063] The binarization of escape sample is EG3 in the current VTM. However, for a uniform distribution signal, the fixed length binarization could be better than EG3 with both distortion and bitrate measurements.

[0064] JVET-Q0075 proposes to use the fixed length binarization for escape samples and the quantization and dequantization process are also modified accordingly.

[0065] The maximal bit depth of an escape sample in the proposed method depends on the quantization parameter and derived as follows.

[0066] $\max(1, \text{bitDepth} - (\max(\text{QpPrimeTsMin}, \text{Qp}) - 4)/6)$

[0067] Herein, the bitDepth is the internal bit depth, Qp is the current quantization parameter (QP), QpPrimeTsMin is the minimal QP for transform skip blocks and max is an operation to get a larger value between two inputs.

[0068] Also, only a shifting operation is needed in the dequantization process for an escape sample. Let escapeVal be the decoded escape value and recon be the reconstructed value of an escape sample. The value is derived as follows.

[0069] $\text{shift} = \min(\text{bitDepth} - 1, (\max(\text{QpPrimeTsMin}, \text{Qp}) - 4)/6)$

[0070] $\text{recon} = (\text{escapeVal} \ll \text{shift})$

[0071] It is to guarantee that the distortion of the reconstructed value is always smaller or equal to that of the current design, i.e. $((\text{escapeVal} * \text{levelScale}[\text{qP}\%6]) \ll (\text{qP}/6 + 32)) \gg 6$.

[0072] At the encoder, the quantization is implemented as follows:

[0073] $\text{escapeVal} = (\text{p} + (1 \ll (\text{shift} - 1))) \gg \text{shift}$

[0074] $\text{escapeVal} = \text{clip3}(0, (1 \ll \text{bd}) - 1, \text{escapeVal})$

[0075] Compared with the current design using EG3 and quantization with a dequantization table, one addition, one multiplication and two shifting operations, the proposed method is much simpler, which needs just one shifting operation.

3.5. Embodiments from JVET-Q0294

[0076] In order to achieve efficient compression in mixed lossy and lossless coding, JVET-Q0294 proposes to signal a flag at each coding tree unit (CTU) to indicate whether a CTU is coded as either in lossless or in lossy mode. If a CTU is lossless coded, an additional CTU level flag is signaled to specify the residual coding method, either regular residual coding or transform skip residual coding, used for that CTU.

4. Examples of technical problems solved by solutions herein

[0077] The existing designs for signaling of subpictures, tiles, and slices in VVC have the following problems:

- 1) The coding of `sps_num_subpics_minus1` is `u(8)`, which disallows more than 256 subpictures per picture. However, in certain applications, the maximum number of subpictures per picture may need to be greater than 256.
- 2) It is allowed to have `subpics_present_flag` equal to 0 and `sps_subpic_id_present_flag` equal to 1. However, this does not make sense as `subpics_present_flag` equal to 0 means that the CLVS has no information on subpictures at all.
- 3) A list of subpicture IDs may be signaled in picture headers (PHs), one for each of the subpictures. However, when the list of subpicture IDs is signaled in PHs, and when a subset of the subpictures is extracted from the bitstream, all the PHs would all need to be changed. This is undesirable.
- 4) Currently when subpicture IDs are indicated to be explicitly signaled, by `sps_subpic_id_present_flag` (or the name of the syntax element is changed to `subpic_ids_explicitly_signalled_flag`) equal to 1, subpicture IDs may be not signalled anywhere. This is problematic as subpicture IDs need to be explicitly signaled in either the SPS or the PPS when subpicture IDs are indicated to be explicitly signaled.
- 5) When subpicture IDs are not explicitly signaled, the slice header syntax element `slice_subpic_id` still need to be signaled as long as `subpics_present_flag` is equal to 1,

including when `sps_num_subpics_minus1` is equal to 0. However, the length of `slice_subpic_id` is currently specified as $\text{Ceil}(\text{Log2}(\text{sps_num_subpics_minus1} + 1))$ bits, which would be 0 bits when `sps_num_subpics_minus1` is equal to 0. This is problematic, as the length of any present syntax elements cannot be 0 bits.

- 6) The subpicture layout, including the number of subpictures and their sizes and positions, keeps unchanging for the entire CLVS. Even when the subpicture IDs are not explicitly signaled in the SPS or the PPS, the subpicture ID length still needs to be signaled, for the subpicture ID syntax element in slice headers.
- 7) Whenever `rect_slice_flag` is equal to 1, the syntax element `slice_address` is signaled in the slice header and specifies the slice index within the subpicture containing the slice, including when the number of slices within the subpicture (i.e., `NumSlicesInSubpic[ SubPicIdx ]`) is equal to 1. However, currently, when `rect_slice_flag` is equal to 1, the length of `slice_address` is specified to be $\text{Ceil}(\text{Log2}(\text{NumSlicesInSubpic[SubPicIdx]}))$ bits, which would be 0 bits when `NumSlicesInSubpic[ SubPicIdx ]` is equal to 1. This is problematic, as the length of any present syntax elements cannot be 0 bits.
- 8) There is redundancy between the syntax elements `no_pic_partition_flag` and `pps_num_subpics_minus1`, although the latest VVC text has the following constraint: When `sps_num_subpics_minus1` is greater than 0, the value of `no_pic_partition_flag` shall be equal to 1.
- 9) Within a CLVS, the subpicture ID value for a particular subpicture position or index may change from picture to picture. When this happens, in principle, the subpicture cannot use inter prediction by referring to a reference picture in the same layer. However, currently, there lacks a constraint to prohibit this in the current VVC specification.
- 10) In current VVC design, a reference picture could be a picture in a different layer to support multiple applications, e.g., scalable video coding and multi-view video coding. If subpicture is present in different layers, whether to allow or disallow the inter-layer prediction needs to be studied.

5. Example techniques and embodiments

[0078] To solve the above problems, and others, methods as summarized below are disclosed. The inventions should be considered as examples to explain the general concepts and should not be interpreted in a narrow way. Furthermore, these inventions can be applied individually or combined in any manner.

[0079] The following abbreviations have the same meaning as they are in JVET-P1001-vE.

[0080] BP (buffering period), BP SEI (supplemental enhancement information),

[0081] PT (picture timing), PT SEI,

[0082] AU (access unit),

[0083] OP (operation point),

[0084] DUI (decoding unit information), DUI SEI,

[0085] NAL (network abstraction layer),

[0086] NUT (NAL unit type),

[0087] GDR (gradual decoding refresh),

[0088] SLI (subpicture level information), SLI SEI

[0089] To solve the first problem, change the coding of `sps_num_subpics_minus1` from `u(8)` to `ue(v)`, to enable more than 256 subpictures per picture.

a. Furthermore, the value of `sps_num_subpics_minus1` is restricted to be in the range of 0 to $\text{Ceil}(\text{pic_width_max_in_luma_samples} \div \text{CtbSizeY}) *$

$\text{Ceil}(\text{pic_height_max_in_luma_samples} \div \text{CtbSizeY}) - 1$, inclusive.

b. Furthermore, the number of subpictures per picture is further restricted in the definition of levels.

2) To solve the second problem, condition the signaling of the syntax element

`sps_subpic_id_present_flag` on "if(`subpics_present_flag`)", i.e., the syntax element

`sps_subpic_id_present_flag` is not signaled when `subpics_present_flag` is equal to 0, and infer the value of `sps_subpic_id_present_flag` to be equal to 0 when it is not present.

a. Alternatively, the syntax element `sps_subpic_id_present_flag` is still signaled when `subpics_present_flag` is equal to 0, but the value is required to be equal to 0 when `subpics_present_flag` is equal to 0.

- b. Furthermore, the names of the syntax elements `subpics_present_flag` and `sps_subpic_id_present_flag` are changed to be `subpic_info_present_flag` and `subpic_ids_explicitly_signalled_flag`, respectively.
- 3) To solve the third problem, the signaling of subpicture IDs in the PH syntax is removed. Consequently, the list `SubpicIdList[ i ]`, for `i` in the range of 0 to `sps_num_subpics_minus1`, inclusive, is derived as follows:


```

      for( i = 0; i <= sps_num_subpics_minus1; i++ )
        if( subpic_ids_explicitly_signalled_flag )
          SubpicIdList[ i ] = subpic_ids_in_pps_flag ? pps_subpic_id[ i ] :
          sps_subpic_id[ i ]
        else
          SubpicIdList[ i ] = i
      
```
- 4) To solve the fourth problem, subpicture IDs are signalled either in the SPS or in the PPS when subpictures are indicated to be explicitly signalled.
 - a. This is realized by add the following constraint: If `subpic_ids_explicitly_signalled_flag` is 0 or `subpic_ids_in_sps_flag` is equal to 1, `subpic_ids_in_pps_flag` shall be equal to 0. Otherwise (`subpic_ids_explicitly_signalled_flag` is 1 and `subpic_ids_in_sps_flag` is equal to 0), `subpic_ids_in_pps_flag` shall be equal to 1.
- 5) To solve the fifth and sixth problems, the length of subpicture IDs is signaled in the SPS regardless of the value of the SPS flag `sps_subpic_id_present_flag` (or renamed to `subpic_ids_explicitly_signalled_flag`), although the length may also be signaled in the PPS when subpicture IDs are explicitly signaled in the PPS to avoid parsing dependency of PPS on SPS. In this case, the length also specifies the length of the subpicture IDs in the slice headers, even subpicture IDs are not explicitly signaled in the SPS or PPS. Consequently, the length of the `slice_subpic_id`, when present, is also specified by the subpicture ID length signaled in the SPS.
- 6) Alternatively, to solve the fifth and sixth problems, a flag is added to the SPS syntax, the value of 1 for which to specify the existence of the subpicture ID length in the SPS syntax. This flag is present is independent of the value of the flag indicating whether

subpicture IDs are explicitly signaled in the SPS or PPS. The value of this flag may be equal to either 1 or 0 when `subpic_ids_explicitly_signalled_flag` is equal to 0, but the value of flag must be equal to 1 when `subpic_ids_explicitly_signalled_flag` is equal to 1. When this flag is equal to 0, i.e., the subpicture length is not present, the length of `slice_subpic_id` is specified to be $\text{Max}(\text{Ceil}(\text{Log}_2(\text{sps_num_subpics_minus1} + 1)), 1)$ bits (as opposed to be $\text{Ceil}(\text{Log}_2(\text{sps_num_subpics_minus1} + 1))$ bits in the latest VVC draft text).

- a. Alternatively, this flag is present only when `subpic_ids_explicitly_signalled_flag` is equal to 0, and when `subpic_ids_explicitly_signalled_flag` is equal to 1 the value of this flag is inferred to be equal to 1.
- 7) To solve the seventh problem, when `rect_slice_flag` is equal to 1, the length of `slice_address` is specified to be $\text{Max}(\text{Ceil}(\text{Log}_2(\text{NumSlicesInSubpic}[\text{SubPicIdx}])), 1)$ bits.
 - a. Alternatively, further more, when `rect_slice_flag` is equal to 0, the length of `slice_address` is specified to $\text{Max}(\text{Ceil}(\text{Log}_2(\text{NumTilesInPic})), 1)$ bits, as opposed to be $\text{Ceil}(\text{Log}_2(\text{NumTilesInPic}))$ bits.
 - 8) To solve the eighth problem, condition the signaling of `no_pic_partition_flag` on "`if( subpic_ids_in_pps_flag && pps_num_subpics_minus1 > 0 )`", and add the following inference: When not present, the value of `no_pic_partition_flag` is inferred to be equal to 1.
 - a. Alternatively, move the subpicture ID syntax (all the four syntax elements) after the tile and slice syntax in the PPS, e.g., immediately before the syntax element `entropy_coding_sync_enabled_flag`, and then condition the signaling of `pps_num_subpics_minus1` on "`if( no_pic_partition_flag )`".
 - 9) To solve the ninth problem, the following constraint is specified: For each particular subpicture index (or equivalently, subpicture position), when the subpicture ID value changes at a picture `picA` compared to that in the previous picture in decoding order in the same layer as `picA`, unless `picA` is the first picture of the CLVS, the subpicture at `picA` shall only contain coded slice NAL units with `nal_unit_type` equal to `IDR_W_RADL`, `IDR_N_LP`, or `CRA_NUT`.

- a. Alternatively, the above constraint applies only for subpicture indices for which the value of the `subpic_treated_as_pic_flag[ i ]` is equal to 1.
 - b. Alternatively, for both item 9 and 9a, change "IDR_W_RADL, IDR_N_LP, or CRA_NUT" to "IDR_W_RADL, IDR_N_LP, CRA_NUT, RSV_IRAP_11, or RSV_IRAP_12".
 - c. Alternatively, the subpicture at `picA` may contain other types of coded slice NAL units, however, these coded slice NAL unit only use one or more of intra prediction, Intra Block Copy (IBC) prediction, and palette mode prediction.
 - d. Alternatively, a first video unit (such as a slice, a tile, a block, etc.) in a subpicture of `picA` may refer to a second video unit in a previous picture. It is constrained that the second video unit and the first video unit may be in subpictures with the same subpicture index, although the subpicture IDs of the two may be different. A subpicture index is a unique number assigned to a subpicture, which cannot be changed in a CLVS.
- 10) For a particular subpicture index (or equivalently, subpicture position), indications of which of subpictures, identified by layer ID values together with subpicture indices or subpicture ID values, are allowed to be used as reference pictures may be signaled in the bitstream.
- 11) For multiple-layer cases, when certain conditions (e.g., which may depend on the number of subpictures, location of subpictures) are satisfied, inter-layer prediction (ILR) from a subpicture in a different layer is allowed, while when the certain conditions are not satisfied, ILR is disabled.
- a. In one example, even when two subpictures in two layers are with the same subpicture index value but different subpicture ID values, the inter-layer prediction may be still allowed when certain conditions are satisfied.
 - i. In one example, the certain conditions are "if the two layers are associated with different view order index/view order ID values".
 - b. It may be constrained that a first subpicture in a first layer and second subpicture in a second layer must be at the collocated positions and/or rational width/height if the two subpictures have the same subpicture index.

- c. It may be constrained that a first subpicture in a first layer and second subpicture in a second layer must be at the collocated positions and/or rational width/height if the first subpicture can refer to the second reference subpicture.
- 12) An indication of whether a current subpicture may use Inter-Layer Prediction (ILP) from sample values and/or other values, e.g., motion information and/or coding modes information, associated with regions or subpictures of reference layers is signaled in the bitstream, such as in VPS/DPS/SPS/PPS/APS/sequence header/picture header.
- a. In one example, the reference regions or subpictures of reference layers are the ones that contains at least one collocated sample of a sample within the current subpicture.
 - b. In one example, the reference regions or subpictures of reference layers are outside the collocated region of the current subpicture.
 - c. In one example, such an indication is signaled in one or more SEI messages.
 - d. In one example, such an indication is signaled regardless of whether the reference layers have multiple subpictures or not, and, when multiple subpictures are present in one or more of the reference layers, regardless of whether the partitioning of the pictures into subpictures aligned with the current picture such that each subpicture in the current picture a corresponding subpicture in a reference picture that covers the collocated region, and furthermore, regardless of whether the corresponding/collocated subpicture has the same subpicture ID value as the current subpicture.
- 13) When a BP SEI message and a PT SEI message that apply to a particular OP are present within an AU, the BP SEI messages shall precede the PT SEI message in decoding order.
- 14) When a BP SEI message and a DUI SEI message that apply to a particular OP are present within an AU, the BP SEI messages shall precede the DUI SEI message in decoding order.
- 15) When a PT SEI message and a DUI SEI message that apply to a particular OP are present within an AU, the PT SEI messages shall precede the DUI SEI message in decoding order.

- 16) An indicator of whether a picture is an IRAP/GDR picture flag is signaled at the beginning of the picture header, and `no_output_of_prior_pics_flag` may be signaled conditioned on the indicator.

An exemplary syntax design is as below:

<code>picture_header_rbsp() {</code>	Descriptor
<code>irap_or_gdr_pic_flag</code>	<code>u(1)</code>
<code>...</code>	
<code>if(irap_or_gdr_pic_flag)</code>	
<code>no_output_of_prior_pics_flag</code>	<code>u(1)</code>
<code>...</code>	

`irap_or_gdr_pic_flag` equal to 1 specifies the picture associated with the PH is an IRAP or GDR picture. `irap_or_gdr_pic_flag` equal to 0 specifies the picture associated with the PH is neither an IRAP picture nor a GDR picture.

- 17) It is disallowed that a picture for which the value of `mixed_nalu_types_in_pic_flag` equal to 0 shall not contain a coded slice NAL unit with `nal_unit_type` equal to `GDR_NUT`.
- 18) A syntax element (namely `mixed_slice_types_in_pic_flag`) is signaled in the PPS. If `mixed_slice_types_in_pic_flag` is equal to 0, the slice type (B, P, or I) is coded in the PH. Otherwise, the slice type is coded in SHs. The syntax values related to the unused slice types are further skipped in the picture header. The syntax element `mixed_slice_types_in_pic_flag` is signaled conditionally as follows:

<code>if(!rect_slice_flag num_slices_in_pic_minus1 > 0)</code>	
<code>mixed_slice_types_in_pic_flag</code>	<code>u(1)</code>

- 19) In the SPS or PPS, signal up to N1 (e.g., 3) vertical virtual boundaries and up to N2 (e.g., 3) horizontal virtual boundaries. In the PH, signal up to N3 (e.g., 1) extra vertical boundaries and up to N4 (e.g., 1) extra horizontal virtual boundaries, and it is constrained that the total number of vertical virtual boundaries shall be less than or equal to N1, and the total number of horizontal virtual boundaries shall be less than or equal to N2.
- 20) The syntax element `inter_layer_ref_pics_present_flag` in the SPS may be signaled conditionally. For example as follows:

<code>if(sps_video_parameter_set_id != 0)</code>	
--	--

inter_layer_ref_pics_present_flag	u(1)
--	------

- 21) Signal the syntax elements `entropy_coding_sync_enabled_flag` and `entry_point_offsets_present_flag` in the SPS instead of in the PPS.
- 22) It is required that the values of `vps_num_ptls_minus1` and `num_ols_hrd_params_minus1` must be less than a value T. For example, T may be equal to `TotalNumOlss` specified in JVET-2001-vE.
- The difference between T and `vps_num_ptls_minus1` or $(vps_num_ptls_minus1+1)$ may be signaled.
 - The difference between T and `hrd_params_minus1` or $(hrd_params_minus1+1)$ may be signaled.
 - The above differences may be signaled by unary code or Exponential Golomb code.
- 23) It is required that the value of `vps_num_dpb_params_minus1` must be less than or equal to `vps_max_layers_minus1`.
- The difference between T and `vps_num_ptls_minus1` or $(vps_num_ptls_minus1+1)$ may be signaled.
 - The difference may be signaled by unary code or Exponential Golomb code.
- 24) It is allowed that an EOB NAL unit to be made available to the decoder either by being included in the bitstream, or provided through an external means.
- 25) It is allowed that the EOS NAL unit may be made available to the decoder either by being included in the bitstream, or provided through an external means.
- 26) For each layer with layer index i, when `vps_independent_layer_flag[i]` is equal to 0, each picture in the layer shall contain only one subpicture.
- 27) In a sub-bitstream extraction process, specify that, whenever a VCL NAL unit is removed, also remove the filler data units that are associated with the VCL NAL units and also remove all filler SEI messages in SEI NAL units that are associated with the VCL NAL unit.
- 28) It is constrained that, when an SEI NAL unit contains a filler SEI message, it shall not contain any other SEI message that is not a filler SEI message.

- a. Alternatively, it is constrained that, when an SEI NAL unit contains a filler SEI message, it shall contain any other SEI message.
- 29) It is constrained that a VCL NAL unit shall have at most one associated filler NAL unit.
- 30) It is constrained that a VCL NAL unit shall have at most one associated filler NAL unit.
- 31) It is proposed that a flag is signaled to indicate whether a video unit is coded as either in lossless or in lossy mode, wherein the video unit may be a CU or a CTU, and the signaling of an escape sample in the palette mode may depend on the flag.
- a. In one example, the binarization method of an escape sample in the palette mode may depend on the flag.
- b. In one example, the determination of the coding context in arithmetic coding for an escape sample in the palette mode may depend on the flag.

6. Embodiments

*Below are some example embodiments for all the invention aspects except item 8 summarized above in Section 5, which can be applied to the VVC specification. The changed texts are based on the latest VVC text in JVET-P2001-v14. Most relevant parts that have been added or modified are shown in **underline, bolded and italicized text**, and the most relevant removed parts are highlighted in enclosed in bolded double brackets, e.g., **[[a]]** indicates that “a” has been removed.. There are some other changes that are editorial in nature and thus not highlighted.*

6.1. First embodiment

7.3.2.3 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	Descriptor
...	
sps_log2_ctu_size_minus5	u(2)
<u>subpic_info_present_flag</u>	<u>u(1)</u>
if(<u>subpic_info_present_flag</u>) {	
sps_num_subpics_minus1	<u>ue(v)</u>
for(i = 0; i <= sps_num_subpics_minus1; i++) {	
subpic_ctu_top_left_x[i]	u(v)
subpic_ctu_top_left_y[i]	u(v)
subpic_width_minus1[i]	u(v)

subpic_height_minus1[i]	u(v)
subpic_treated_as_pic_flag[i]	u(1)
loop_filter_across_subpic_enabled_flag[i]	u(1)
}	
<u>sps_subpic_id_len_minus1</u>	<u>ue(v)</u>
<u>subpic_ids_explicitly_signalled_flag</u>	u(1)
if(<u>subpic_ids_explicitly_signalled_flag</u>) {	
<u>subpic_ids_in_sps_flag</u>	u(1)
if(<u>subpic_ids_in_sps_flag</u>)	
for(i = 0; i <= sps_num_subpics_minus1; i++)	
sps_subpic_id[i]	u(v)
}	
}	
...	
}	

7.4.3.3 Sequence parameter set RBSP semantics

...

subpic_info_present_flag equal to 1 specifies that subpicture information is present.

subpic_info_present_flag equal to 0 specifies that subpicture information is not present. The values of sps_ref_pic_resampling_enabled_flag and subpic_info_present_flag shall not be both equal to 1.

NOTE 2 – When a bitstream is the result of a sub-bitstream extraction process and contains only a subset of the subpictures of the input bitstream to the sub-bitstream extraction process, it might be required to set the value of subpic_info_present_flag equal to 1 in the SPSs.

sps_num_subpics_minus1 plus 1 specifies the number of subpictures. The value of

sps_num_subpics_minus1 shall be in the range of 0 to

$\text{Ceil}(\text{pic_width_max_in_luma_samples} \div \text{CtbSizeY}) *$

$\text{Ceil}(\text{pic_height_max_in_luma_samples} \div \text{CtbSizeY}) - 1$, inclusive. Noet that the maximum number of subpictures can be further restricted in the level definitions. When not present, the value of

sps_num_subpics_minus1 is inferred to be equal to 0.

subpic_ctu_top_left_x[i] specifies horizontal position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of $\text{subpic_ctu_top_left_x}[i]$ is inferred to be equal to 0.

subpic_ctu_top_left_y $[i]$ specifies vertical position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of $\text{subpic_ctu_top_left_y}[i]$ is inferred to be equal to 0.

subpic_width_minus1 $[i]$ plus 1 specifies the width of the i-th subpicture in units of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of $\text{subpic_width_minus1}[i]$ is inferred to be equal to

$\text{Ceil}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}) - 1$.

subpic_height_minus1 $[i]$ plus 1 specifies the height of the i-th subpicture in units of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of $\text{subpic_height_minus1}[i]$ is inferred to be equal to

$\text{Ceil}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}) - 1$.

subpic_treated_as_pic_flag $[i]$ equal to 1 specifies that the i-th subpicture of each coded picture in the CLVS is treated as a picture in the decoding process excluding in-loop filtering operations.

$\text{subpic_treated_as_pic_flag}[i]$ equal to 0 specifies that the i-th subpicture of each coded picture in the CLVS is not treated as a picture in the decoding process excluding in-loop filtering operations. When not present, the value of $\text{subpic_treated_as_pic_flag}[i]$ is inferred to be equal to 0.

loop_filter_across_subpic_enabled_flag $[i]$ equal to 1 specifies that in-loop filtering operations may be performed across the boundaries of the i-th subpicture in each coded picture in the CLVS.

$\text{loop_filter_across_subpic_enabled_flag}[i]$ equal to 0 specifies that in-loop filtering operations are not performed across the boundaries of the i-th subpicture in each coded picture in the CLVS. When not present, the value of $\text{loop_filter_across_subpic_enabled_pic_flag}[i]$ is inferred to be equal to 1.

It is a requirement of bitstream conformance that the following constraints apply:

- For any two subpictures subpicA and subpicB, when the subpicture index of subpicA is less than that of subpicB, any coded slice NAL unit of subPicA shall precede any coded slice NAL unit of subPicB in decoding order.
- The shapes of the subpictures shall be such that each subpicture, when decoded, shall have its entire left boundary and entire top boundary consisting of picture boundaries or consisting of boundaries of previously decoded subpictures.

sps_subpic_id_len_minus1 plus 1 specifies the number of bits used to represent the explicitly signalled subpicture IDs as well as the slice_subpic_id syntax element in the slice header. The value of sps_subpic_id_len_minus1 shall be in the range of 0 to 15, inclusive.

subpic_ids_explicitly_signalled_flag equal to 1 specifies that a set of subpicture ID, one for each subpicture, is explicitly signalled for each subpicture either in the SPS or the PPS.

subpic_ids_explicitly_signalled_flag equal to 0 specifies that no subpicture IDs are explicitly signalled in the SPS or the PPS. When not present, the value of subpic_ids_explicitly_signalled_flag is inferred to be equal to 0.

subpic_ids_in_sps_flag equal to 1 specifies that a subpicture ID for each subpicture is explicitly signalled in the SPS. subpic_ids_in_sps_flag equal to 0 specifies that no subpicture IDs are signalled in the SPS. When not present, the value of subpic_ids_in_sps_flag is inferred to be equal to 0.

sps_subpic_id[i] specifies the subpicture ID of the i-th subpicture. The length of the sps_subpic_id[i] syntax element is sps_subpic_id_len_minus1 + 1 bits.

...

7.3.2.4 Picture parameter set RBSP syntax

pic_parameter_set_rbsp() {	Descriptor
...	
mixed_nalu_types_in_pic_flag	u(1)
<u>subpic_ids_in_pps_flag</u>	u(1)
if(<u>subpic_ids_in_pps_flag</u>) {	
pps_num_subpics_minus1	ue(v)
pps_subpic_id_len_minus1	ue(v)
for(i = 0; i <= pps_num_subpic_minus1; i++)	
pps_subpic_id[i]	u(v)
}	
no_pic_partition_flag	u(1)
if(!no_pic_partition_flag) {	
pps_log2_ctu_size_minus5	u(2)
num_exp_tile_columns_minus1	ue(v)
num_exp_tile_rows_minus1	ue(v)
for(i = 0; i <= num_exp_tile_columns_minus1; i++)	
tile_column_width_minus1[i]	ue(v)

for(i = 0; i <= num_exp_tile_rows_minus1; i++)	
tile_row_height_minus1[i]	ue(v)
if(NumTilesInPic > 1)	
rect_slice_flag	u(1)
if(rect_slice_flag)	
single_slice_per_subpic_flag	u(1)
if(rect_slice_flag && !single_slice_per_subpic_flag) {	
num_slices_in_pic_minus1	ue(v)
tile_idx_delta_present_flag	u(1)
for(i = 0; i < num_slices_in_pic_minus1; i++) {	
slice_width_in_tiles_minus1[i]	ue(v)
slice_height_in_tiles_minus1[i]	ue(v)
if(slice_width_in_tiles_minus1[i] == 0 && slice_height_in_tiles_minus1[i] == 0) {	
num_slices_in_tile_minus1[i]	ue(v)
for(j = 0; j < num_slices_in_tile_minus1[i]; j++)	
slice_height_in_ctu_minus1[i++]	ue(v)
}	
if(tile_idx_delta_present_flag && i < num_slices_in_pic_minus1)	
tile_idx_delta[i]	se(v)
}	
}	
loop_filter_across_tiles_enabled_flag	u(1)
loop_filter_across_slices_enabled_flag	u(1)
}	
...	
}	

7.4.3.4 Picture parameter set RBSP semantics

...

subpic_ids_in_pps_flag equal to 1 specifies that a subpicture ID for each subpicture is explicitly signalled in the PPS. subpic_ids_in_pps_flag equal to 0 specifies that no subpicture IDs are signalled in the PPS. If subpic_ids_explicitly_signalled_flag is 0 or subpic_ids_in_sps_flag is equal to 1,

subpic_ids in pps flag shall be equal to 0. Otherwise (subpic_ids_explicitly_signalled_flag is 1 and subpic_ids in sps flag is equal to 0), subpic_ids in pps flag shall be equal to 1.

pps_num_subpics_minus1 shall be equal to sps_num_subpics_minus1.

pps_subpic_id_len_minus1 shall be equal to sps_subpic_id_len_minus1.

pps_subpic_id[i] specifies the subpicture ID of the i-th subpicture. The length of the pps_subpic_id[i] syntax element is pps_subpic_id_len_minus1 + 1 bits.

The list SubpicIdList[i], for i in the range of 0 to sps_num_subpics_minus1, inclusive, is derived as follows:

```

for( i = 0; i <= sps_num_subpics_minus1; i++ )
  if( subpic_ids_explicitly_signalled_flag )
    SubpicIdList[ i ] = subpic_ids in pps flag ? pps_subpic_id[ i ] : sps_subpic_id[ i ]
  (76)
  else
    SubpicIdList[ i ] = i

```

It is a requirement of bitstream conformance that, for any i and j in the range of 0 to sps_num_subpics_minus1, inclusive, when i is less than j, SubpicIdList[i] shall be less than SubpicIdList[j].

...

rect_slice_flag equal to 0 specifies that tiles within each slice are in raster scan order and the slice information is not signalled in PPS. rect_slice_flag equal to 1 specifies that tiles within each slice cover a rectangular region of the picture and the slice information is signalled in the PPS. When not present, rect_slice_flag is inferred to be equal to 1. When subpic_info_present_flag is equal to 1, the value of rect_slice_flag shall be equal to 1.

single_slice_per_subpic_flag equal to 1 specifies that each subpicture consists of one and only one rectangular slice. single_slice_per_subpic_flag equal to 0 specifies that each subpicture may consist one or more rectangular slices. When subpic_info_present_flag is equal to 0, single_slice_per_subpic_flag shall be equal to 0. When single_slice_per_subpic_flag is equal to 1, num_slices_in_pic_minus1 is inferred to be equal to sps_num_subpics_minus1.

...

7.3.7.1 General slice header syntax

slice_header() {	Descriptor
...	
if(<u>subpic_info_present_flag</u>)	

slice_subpic_id	u(v)
if(rect_slice_flag NumTilesInPic > 1)	
slice_address	u(v)
if(!rect_slice_flag && NumTilesInPic > 1)	
num_tiles_in_slice_minus1	ue(v)
slice_type	ue(v)
...	
}	

7.4.8.1 General slice header semantics

...

slice_subpic_id specifies the subpicture ID of the subpicture that contains the slice. The length of slice_subpic_id is $\text{sps_subpic_id_len_minus1} + 1$ bits.

When not present, the value of slice_subpic_id is inferred to be equal to 0.

The variable SubPicIdx is derived to be the value such that SubpicIdList[SubPicIdx] is equal to slice_subpic_id.

slice_address specifies the slice address of the slice. When not present, the value of slice_address is inferred to be equal to 0.

If rect_slice_flag is equal to 0, the following applies:

- The slice address is the raster scan tile index.
- The length of slice_address is $\text{Ceil}(\text{Log2}(\text{NumTilesInPic}))$ bits.
- The value of slice_address shall be in the range of 0 to NumTilesInPic – 1, inclusive.

Otherwise (rect_slice_flag is equal to 1), the following applies:

- The slice address is the subpicture-level slice index of the slice.
- The length of slice_address is $\text{Max}(\text{Ceil}(\text{Log2}(\text{NumSlicesInSubpic}[\text{SubPicIdx}]]), 1)$ bits.
- The value of slice_address shall be in the range of 0 to NumSlicesInSubpic[SubPicIdx] – 1, inclusive.

It is a requirement of bitstream conformance that the following constraints apply:

- If rect_slice_flag is equal to 0 or subpic_info_present_flag is equal to 0, the value of slice_address shall not be equal to the value of slice_address of any other coded slice NAL unit of the same coded picture.

- Otherwise, the pair of slice_subpic_id and slice_address values shall not be equal to the pair of slice_subpic_id and slice_address values of any other coded slice NAL unit of the same coded picture.
- When rect_slice_flag is equal to 0, the slices of a picture shall be in increasing order of their slice_address values.
- The shapes of the slices of a picture shall be such that each CTU, when decoded, shall have its entire left boundary and entire top boundary consisting of a picture boundary or consisting of boundaries of previously decoded CTU(s).

...

[0090] FIG. 5 is a block diagram showing an example video processing system 500 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 500. The system 500 may include input 502 for receiving video content. The video content may be received in a raw or uncompressed format, e.g., 8 or 10 bit multi-component pixel values, or may be in a compressed or encoded format. The input 502 may represent a network interface, a peripheral bus interface, or a storage interface. Examples of network interface include wired interfaces such as Ethernet, passive optical network (PON), etc. and wireless interfaces such as Wi-Fi or cellular interfaces.

[0091] The system 500 may include a coding component 504 that may implement the various coding or encoding methods described in the present document. The coding component 504 may reduce the average bitrate of video from the input 502 to the output of the coding component 504 to produce a coded representation of the video. The coding techniques are therefore sometimes called video compression or video transcoding techniques. The output of the coding component 504 may be either stored, or transmitted via a communication connected, as represented by the component 506. The stored or communicated bitstream (or coded) representation of the video received at the input 502 may be used by the component 508 for generating pixel values or displayable video that is sent to a display interface 510. The process of generating user-viewable video from the bitstream representation is sometimes called video decompression. Furthermore, while certain video processing operations are referred to as “coding” operations or tools, it will be appreciated that the coding tools or operations are used at an encoder and corresponding

decoding tools or operations that reverse the results of the coding will be performed by a decoder.

[0092] Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or Displayport, and so on. Examples of storage interfaces include SATA (serial advanced technology attachment), PCI, IDE interface, and the like. The techniques described in the present document may be embodied in various electronic devices such as mobile phones, laptops, smartphones or other devices that are capable of performing digital data processing and/or video display.

[0093] FIG. 6 is a block diagram of a video processing apparatus 600. The apparatus 600 may be used to implement one or more of the methods described herein. The apparatus 600 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 600 may include one or more processors 602, one or more memories 604 and video processing hardware 606. The processor(s) 602 may be configured to implement one or more methods described in the present document. The memory (memories) 604 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 606 may be used to implement, in hardware circuitry, some techniques described in the present document. In some embodiments, the hardware 606 may be partly or entirely in the processors 602, e.g., a graphics processor.

[0094] FIG. 7 is a block diagram that illustrates an example video coding system 100 that may utilize the techniques of this disclosure.

[0095] As shown in FIG. 7, video coding system 100 may include a source device 110 and a destination device 120. Source device 110 generates encoded video data which may be referred to as a video encoding device. Destination device 120 may decode the encoded video data generated by source device 110 which may be referred to as a video decoding device.

[0096] Source device 110 may include a video source 112, a video encoder 114, and an input/output (I/O) interface 116.

[0097] Video source 112 may include a source such as a video capture device, an interface to receive video data from a video content provider, and/or a computer graphics system for generating video data, or a combination of such sources. The video data may comprise one or more pictures. Video encoder 114 encodes the video data from video source 112 to generate a

bitstream. The bitstream may include a sequence of bits that form a coded representation of the video data. The bitstream may include coded pictures and associated data. The coded picture is a coded representation of a picture. The associated data may include sequence parameter sets, picture parameter sets, and other syntax structures. I/O interface 116 may include a modulator/demodulator (modem) and/or a transmitter. The encoded video data may be transmitted directly to destination device 120 via I/O interface 116 through network 130a. The encoded video data may also be stored onto a storage medium/server 130b for access by destination device 120.

[0098] Destination device 120 may include an I/O interface 126, a video decoder 124, and a display device 122.

[0099] I/O interface 126 may include a receiver and/or a modem. I/O interface 126 may acquire encoded video data from the source device 110 or the storage medium/ server 130b. Video decoder 124 may decode the encoded video data. Display device 122 may display the decoded video data to a user. Display device 122 may be integrated with the destination device 120, or may be external to destination device 120 which be configured to interface with an external display device.

[00100] Video encoder 114 and video decoder 124 may operate according to a video compression standard, such as the High Efficiency Video Coding (HEVC) standard, Versatile Video Coding(VVM) standard and other current and/or further standards.

[00101] FIG. 8 is a block diagram illustrating an example of video encoder 200, which may be video encoder 114 in the system 100 illustrated in FIG. 7.

[00102] Video encoder 200 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 8, video encoder 200 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of video encoder 200. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[00103] The functional components of video encoder 200 may include a partition unit 201, a predication unit 202 which may include a mode select unit 203, a motion estimation unit 204, a motion compensation unit 205 and an intra prediction unit 206, a residual generation unit 207, a

transform unit 208, a quantization unit 209, an inverse quantization unit 210, an inverse transform unit 211, a reconstruction unit 212, a buffer 213, and an entropy encoding unit 214.

[00104] In other examples, video encoder 200 may include more, fewer, or different functional components. In an example, predication unit 202 may include an intra block copy(IBC) unit. The IBC unit may perform predication in an IBC mode in which at least one reference picture is a picture where the current video block is located.

[00105] Furthermore, some components, such as motion estimation unit 204 and motion compensation unit 205 may be highly integrated, but are represented in the example of FIG. 8 separately for purposes of explanation.

[00106] Partition unit 201 may partition a picture into one or more video blocks. Video encoder 200 and video decoder 300 may support various video block sizes.

[00107] Mode select unit 203 may select one of the coding modes, intra or inter, e.g., based on error results, and provide the resulting intra- or inter-coded block to a residual generation unit 207 to generate residual block data and to a reconstruction unit 212 to reconstruct the encoded block for use as a reference picture. In some example, Mode select unit 203 may select a combination of intra and inter predication (CIIP) mode in which the predication is based on an inter predication signal and an intra predication signal. Mode select unit 203 may also select a resolution for a motion vector (e.g., a sub-pixel or integer pixel precision) for the block in the case of inter-predication.

[00108] To perform inter prediction on a current video block, motion estimation unit 204 may generate motion information for the current video block by comparing one or more reference frames from buffer 213 to the current video block. Motion compensation unit 205 may determine a predicted video block for the current video block based on the motion information and decoded samples of pictures from buffer 213 other than the picture associated with the current video block.

[00109] Motion estimation unit 204 and motion compensation unit 205 may perform different operations for a current video block, for example, depending on whether the current video block is in an I slice, a P slice, or a B slice.

[00110] In some examples, motion estimation unit 204 may perform uni-directional prediction for the current video block, and motion estimation unit 204 may search reference pictures of list

0 or list 1 for a reference video block for the current video block. Motion estimation unit 204 may then generate a reference index that indicates the reference picture in list 0 or list 1 that contains the reference video block and a motion vector that indicates a spatial displacement between the current video block and the reference video block. Motion estimation unit 204 may output the reference index, a prediction direction indicator, and the motion vector as the motion information of the current video block. Motion compensation unit 205 may generate the predicted video block of the current block based on the reference video block indicated by the motion information of the current video block.

[00111] In other examples, motion estimation unit 204 may perform bi-directional prediction for the current video block, motion estimation unit 204 may search the reference pictures in list 0 for a reference video block for the current video block and may also search the reference pictures in list 1 for another reference video block for the current video block. Motion estimation unit 204 may then generate reference indexes that indicate the reference pictures in list 0 and list 1 containing the reference video blocks and motion vectors that indicate spatial displacements between the reference video blocks and the current video block. Motion estimation unit 204 may output the reference indexes and the motion vectors of the current video block as the motion information of the current video block. Motion compensation unit 205 may generate the predicted video block of the current video block based on the reference video blocks indicated by the motion information of the current video block.

[00112] In some examples, motion estimation unit 204 may output a full set of motion information for decoding processing of a decoder.

[00113] In some examples, motion estimation unit 204 may do not output a full set of motion information for the current video. Rather, motion estimation unit 204 may signal the motion information of the current video block with reference to the motion information of another video block. For example, motion estimation unit 204 may determine that the motion information of the current video block is sufficiently similar to the motion information of a neighboring video block.

[00114] In one example, motion estimation unit 204 may indicate, in a syntax structure associated with the current video block, a value that indicates to the video decoder 300 that the current video block has the same motion information as the another video block.

[00115] In another example, motion estimation unit 204 may identify, in a syntax structure associated with the current video block, another video block and a motion vector difference (MVD). The motion vector difference indicates a difference between the motion vector of the current video block and the motion vector of the indicated video block. The video decoder 300 may use the motion vector of the indicated video block and the motion vector difference to determine the motion vector of the current video block.

[00116] As discussed above, video encoder 200 may predictively signal the motion vector. Two examples of predictive signaling techniques that may be implemented by video encoder 200 include advanced motion vector predication (AMVP) and merge mode signaling.

[00117] Intra prediction unit 206 may perform intra prediction on the current video block. When intra prediction unit 206 performs intra prediction on the current video block, intra prediction unit 206 may generate prediction data for the current video block based on decoded samples of other video blocks in the same picture. The prediction data for the current video block may include a predicted video block and various syntax elements.

[00118] Residual generation unit 207 may generate residual data for the current video block by subtracting (e.g., indicated by the minus sign) the predicted video block(s) of the current video block from the current video block. The residual data of the current video block may include residual video blocks that correspond to different sample components of the samples in the current video block.

[00119] In other examples, there may be no residual data for the current video block for the current video block, for example in a skip mode, and residual generation unit 207 may not perform the subtracting operation.

[00120] Transform processing unit 208 may generate one or more transform coefficient video blocks for the current video block by applying one or more transforms to a residual video block associated with the current video block.

[00121] After transform processing unit 208 generates a transform coefficient video block associated with the current video block, quantization unit 209 may quantize the transform coefficient video block associated with the current video block based on one or more quantization parameter (QP) values associated with the current video block.

[00122] Inverse quantization unit 210 and inverse transform unit 211 may apply inverse quantization and inverse transforms to the transform coefficient video block, respectively, to reconstruct a residual video block from the transform coefficient video block. Reconstruction unit 212 may add the reconstructed residual video block to corresponding samples from one or more predicted video blocks generated by the predication unit 202 to produce a reconstructed video block associated with the current block for storage in the buffer 213.

[00123] After reconstruction unit 212 reconstructs the video block, loop filtering operation may be performed reduce video blocking artifacts in the video block.

[00124] Entropy encoding unit 214 may receive data from other functional components of the video encoder 200. When entropy encoding unit 214 receives the data, entropy encoding unit 214 may perform one or more entropy encoding operations to generate entropy encoded data and output a bitstream that includes the entropy encoded data.

[00125] FIG. 9 is a block diagram illustrating an example of video decoder 300 which may be video decoder 114 in the system 100 illustrated in FIG. 7.

[00126] The video decoder 300 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 8, the video decoder 300 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of the video decoder 300. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[00127] In the example of FIG. 9, video decoder 300 includes an entropy decoding unit 301, a motion compensation unit 302, an intra prediction unit 303, an inverse quantization unit 304, an inverse transformation unit 305, and a reconstruction unit 306 and a buffer 307. Video decoder 300 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 200 (FIG. 8).

[00128] Entropy decoding unit 301 may retrieve an encoded bitstream. The encoded bitstream may include entropy coded video data (e.g., encoded blocks of video data). Entropy decoding unit 301 may decode the entropy coded video data, and from the entropy decoded video data, motion compensation unit 302 may determine motion information including motion vectors, motion vector precision, reference picture list indexes, and other motion information. Motion

compensation unit 302 may, for example, determine such information by performing the AMVP and merge mode.

[00129] Motion compensation unit 302 may produce motion compensated blocks, possibly performing interpolation based on interpolation filters. Identifiers for interpolation filters to be used with sub-pixel precision may be included in the syntax elements.

[00130] Motion compensation unit 302 may use interpolation filters as used by video encoder 20 during encoding of the video block to calculate interpolated values for sub-integer pixels of a reference block. Motion compensation unit 302 may determine the interpolation filters used by video encoder 200 according to received syntax information and use the interpolation filters to produce predictive blocks.

[00131] Motion compensation unit 302 may use some of the syntax information to determine sizes of blocks used to encode frame(s) and/or slice(s) of the encoded video sequence, partition information that describes how each macroblock of a picture of the encoded video sequence is partitioned, modes indicating how each partition is encoded, one or more reference frames (and reference frame lists) for each inter-encoded block, and other information to decode the encoded video sequence.

[00132] Intra prediction unit 303 may use intra prediction modes for example received in the bitstream to form a prediction block from spatially adjacent blocks. Inverse quantization unit 303 inverse quantizes, i.e., de-quantizes, the quantized video block coefficients provided in the bitstream and decoded by entropy decoding unit 301. Inverse transform unit 303 applies an inverse transform.

[00133] Reconstruction unit 306 may sum the residual blocks with the corresponding prediction blocks generated by motion compensation unit 202 or intra-prediction unit 303 to form decoded blocks. If desired, a deblocking filter may also be applied to filter the decoded blocks in order to remove blockiness artifacts. The decoded video blocks are then stored in buffer 307, which provides reference blocks for subsequent motion compensation/intra prediction and also produces decoded video for presentation on a display device.

[00134] FIGS. 10-26 show example methods that can implement the technical solution described above in, for example, the embodiments show in FIGS. 5-9.

[00135] FIG. 10 shows a flowchart for an example method 1000 of video processing. The method 1000 includes, at operation 1010, performing a conversion between a video and a bitstream of the video, the bitstream comprising one or more access units according to a format rule, and the format rule specifying that an order in which a first message and a second message that apply to an operation point (OP) are present within an access unit (AU) such that the first message precedes the second message in a decoding order.

[00136] FIG. 11 shows a flowchart for an example method 1100 of video processing. The method 1100 includes, at operation 1110, performing a conversion between a video and a bitstream of the video, the bitstream comprising one or more access units according to a format rule, and the format rule specifying that an order in which a plurality of messages that apply to an operation point (OP) are present within an access unit such that a first message of the plurality of messages precedes a second message of the plurality of messages in a decoding order.

[00137] FIG. 12 shows a flowchart for an example method 1200 of video processing. The method 1200 includes, at operation 1210, performing a conversion between a video comprising a picture and a bitstream of the video, the bitstream conforming to a format rule that specifies that an indication of whether a first flag is signaled at a beginning of a picture header associated with the picture, and the first flag being indicative of whether the picture is an intra random access point (IRAP) picture or a gradual decoding refresh (GDR) picture.

[00138] FIG. 13 shows a flowchart for an example method 1300 of video processing. The method 1300 includes, at operation 1310, performing a conversion between a video comprising one or more pictures and a bitstream of the video, the bitstream conforming to a format rule that disallows coding of a picture of the one or more pictures to include a coded slice network abstraction layer (NAL) unit having a gradual decoding refresh type and to associate with a flag indicating that the picture contains mixed types of NAL units.

[00139] FIG. 14 shows a flowchart for an example method 1400 of video processing. The method 1400 includes, at operation 1410, performing a conversion between a video comprising one or more pictures and a bitstream of the video, the bitstream conforming to a format rule that allows coding of a picture of the one or more pictures to include a coded slice network access layer (NAL) unit having a gradual decoding refresh type and to associate with a flag indicating that the picture does not contain mixed types of NAL units.

[00140] FIG. 15 shows a flowchart for an example method 1500 of video processing. The method 1500 includes, at operation 1510, performing a conversion between a picture of a video and a bitstream of the video, the bitstream conforming to a format rule that specifies whether a first syntax element is signaled in a picture parameter set (PPS) associated with the picture, the picture comprising one or more slices with a slice type, and the first syntax element indicating that the slice type is signaled in the picture header due to the first syntax element being equal to zero, and otherwise indicates that the slice type is signaled in a slice header.

[00141] FIG. 16 shows a flowchart for an example method 1600 of video processing. The method 1600 includes, at operation 1610, performing a conversion between a picture of a video and a bitstream of the video according to a rule, the conversion comprising an in-loop filtering process, and the rule specifying that a total number of vertical virtual boundaries and a total number of horizontal virtual boundaries related to the in-loop filtering process are signaled at a picture-level or a sequence-level.

[00142] FIG. 17 shows a flowchart for an example method 1700 of video processing. The method 1700 includes, at operation 1710, performing a conversion between a video comprising one or more pictures and a bitstream of the video, the bitstream conforming to a format rule that conditionally allows coding of a picture in one layer using reference pictures from other layers based on a first syntax element indicating whether the reference pictures from the other layers are present in the bitstream, and the first syntax element being conditionally signaled in the bitstream based on a second syntax element that indicates whether an identifier of a parameter set associated with the picture is not equal to zero.

[00143] FIG. 18 shows a flowchart for an example method 1800 of video processing. The method 1800 includes, at operation 1810, performing a conversion between a picture of a video and a bitstream of the video, the bitstream conforming to a format rule that defines a first syntax element for enabling (a) a synchronization process for context variables before decoding a coding tree unit (CTU) in the picture and (b) a storage process for the context variables after decoding the CTU, and the first syntax element being signaled in a sequence parameter set (SPS) associated with the picture.

[00144] FIG. 19 shows a flowchart for an example method 1900 of video processing. The method 1900 includes, at operation 1910, performing a conversion between a picture of a video

and a bitstream of the video, the bitstream conforming to a format rule that defines a syntax element for indicating whether signaling for entry point offsets for tiles or tile-specific CTU rows are present in a slice header of the picture, and the syntax element being signaled in a sequence parameter set (SPS) associated with the picture.

[00145] FIG. 20 shows a flowchart for an example method 2000 of video processing. The method 2000 includes, at operation 2010, performing a conversion between a video and a bitstream of the video according to a rule that specifies that a first syntax element, which indicates a number of parameters for an output layer set (OLS) hypothetical reference decoder (HRD) in a video parameter set (VPS) associated with the video, is less than a first predetermined threshold.

[00146] FIG. 21 shows a flowchart for an example method 2100 of video processing. The method 2100 includes, at operation 2110, performing a conversion between a video and a bitstream of the video according to a rule that specifies that a syntax element, which indicates a number of profile/tier/level (PTL) syntax structures in a video parameter set (VPS) associated with the video, is less than a predetermined threshold.

[00147] FIG. 22 shows a flowchart for an example method 2200 of video processing. The method 2200 includes, at operation 2210, performing a conversion between a video and a bitstream of the video according to a rule that specifies that a first syntax element, which indicates a number of decoded picture buffer parameters syntax structures in a video parameters set (VPS), must be less than or equal to a second syntax element, which indicates a number of layers specified by the VPS.

[00148] FIG. 23 shows a flowchart for an example method 2300 of video processing. The method 2300 includes, at operation 2310, performing a conversion between a video and a bitstream of the video according to a rule that allows a terminating network abstraction layer (NAL) unit to be made available to a decoder by signaling in the bitstream or providing through external means.

[00149] FIG. 24 shows a flowchart for an example method 2400 of video processing. The method 2400 includes, at operation 2410, performing a conversion between a video and a bitstream of the video, the bitstream conforming to a format rule that restricts each layer in the

bitstream to contain only one subpicture due to a syntax element being equal to zero, which indicates that the each layer is configured to use inter-layer prediction.

[00150] FIG. 25 shows a flowchart for an example method 2500 of video processing. The method 2500 includes, at operation 2510, performing a conversion between a video and a bitstream of the video according to a rule that specifies that a sub-bitstream extraction process is implemented to generate a sub-bitstream for decoding, wherein the sub-bitstream extraction process is configured to extract, from the bitstream, a sub-bitstream with a target highest temporal identifier, and the rule further specifying that, during the extracting, upon removing a video coding layer (VCL) network abstraction layer (NAL) unit, filler data units and filler supplemental enhancement information (SEI) messages in SEI NAL units that are associated with the VCL NAL unit are also removed.

[00151] FIG. 26 shows a flowchart for an example method 2600 of video processing. The method 2600 includes, at operation 2610, performing a conversion between a video unit of a video and a bitstream of the video, the bitstream conforming to a format rule that specifies that the bitstream includes a first syntax element, which indicates whether the video unit is coded in a lossless mode or in a lossy mode, and signaling a second syntax element, which indicates an escape sample in a palette mode applied to the video unit, being selectively included based on a value of the first syntax element.

[00152] A listing of solutions preferred by some embodiments is provided next.

[00153] A1. A method of video processing, comprising performing a conversion between a video and a bitstream of the video, wherein the bitstream comprises one or more access units according to a format rule, and wherein the format rule specifies an order in which a first message and a second message that apply to an operation point (OP) are present within an access unit (AU) such that the first message precedes the second message in a decoding order.

[00154] A2. The method of solution A1, wherein the first message comprises a buffering period (BP) supplemental enhancement information (SEI) message and the second message comprises a picture timing (PT) SEI message.

[00155] A3. The method of solution A1, wherein the first message comprises a buffering period (BP) supplemental enhancement information (SEI) message and the second message comprises a decoding unit information (DUI) SEI message.

[00156] A4. The method of solution A1, wherein the first message comprises a picture timing (PT) supplemental enhancement information (SEI) message and the second message comprises a decoding unit information (DUI) SEI message.

[00157] A5. A method of video processing, comprising performing a conversion between a video and a bitstream of the video, wherein the bitstream comprises one or more access units according to a format rule, and wherein the format rule specifies an order in which a plurality of messages that apply to an operation point (OP) are present within an access unit such that a first message of the plurality of messages precedes a second message of the plurality of messages in a decoding order.

[00158] A6. The method of solution A5, wherein the plurality of messages comprises a buffering period (BP) SEI message, a decoding unit information (DUI) SEI message, a picture timing (PT) SEI message, and a subpicture level information (SLI) SEI message.

[00159] A7. The method of solution A6, wherein the decoding order is the SLI SEI message, the BP SEI message, the PT SEI message, and the DUI message.

[00160] Another listing of solutions preferred by some embodiments is provided next.

[00161] B1. A method of video processing, comprising performing a conversion between a video comprising a picture and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule specifies that an indication of whether a first flag is signaled at a beginning of a picture header associated with the picture, wherein the first flag is indicative of whether the picture is an intra random access point (IRAP) picture or a gradual decoding refresh (GDR) picture.

[00162] B2. The method of solution B1, wherein the IRAP picture is a picture that, when a decoding of the bitstream starts at the picture, the picture and all subsequent pictures in an output order can be correctly decoded.

[00163] B3. The method of solution B1, wherein the IRAP picture contains only I slices.

[00164] B4. The method of solution B1, wherein the GDR picture is a picture that, when the decoding of the bitstream starts at the picture, a corresponding recovery point picture and all subsequent pictures in both a decoding order and the output order can be correctly decoded.

[00165] B5. The method of solution B1, wherein the format rule further specifies that whether a second flag is signaled in the bitstream is based on the indication.

[00166] B6. The method of solution B5, wherein the first flag is `irap_or_gdr_pic_flag`.

[00167] B7. The method of solution B6, wherein the second flag is `no_output_of_prior_pics_flag`.

[00168] B8. The method of any of solutions B1 to B7, wherein the first flag being equal to one indicates that the picture is an IRAP picture or a GDR picture.

[00169] B9. The method of any of solutions B1 to B7, wherein the first flag being equal to zero indicates that the picture is neither an IRAP picture nor a GDR picture.

[00170] B10. The method of any of solutions B1 to B9, wherein each network access layer (NAL) unit in the GDR picture has a `nal_unit_type` syntax element equal to `GDR_NUT`.

[00171] Yet another listing of solutions preferred by some embodiments is provided next.

[00172] C1. A method of video processing, comprising performing a conversion between a video comprising one or more pictures and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule disallows coding of a picture of the one or more pictures to include a coded slice network abstraction layer (NAL) unit having a gradual decoding refresh type and to associate with a flag indicating that the picture contains mixed types of NAL units.

[00173] C2. A method of video processing, comprising performing a conversion between a video comprising one or more pictures and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule allows coding of a picture of the one or more pictures to include a coded slice network access layer (NAL) unit having a gradual decoding refresh type and to associate with a flag indicating that the picture does not contain mixed types of NAL units.

[00174] C3. The method of solution C1 or C2, wherein the flag is `mixed_nalu_types_in_pic_flag`.

[00175] C4. The method of solution C3, wherein the flag is in a picture parameter set (PPS).

[00176] C5. The method of any of solutions C1 to C4, wherein the picture is a gradual decoding refresh (GDR) picture, and wherein each slice or NAL unit in the picture has a `nal_unit_type` equal to `GDR_NUT`.

[00177] C6. A method of video processing, comprising performing a conversion between a picture of a video and a bitstream of the video, wherein the bitstream conforms to a format rule that specifies whether a first syntax element is signaled in a picture parameter set (PPS)

associated with the picture, wherein the picture comprises one or more slices with a slice type, wherein the first syntax element indicates that the slice type is signaled in the picture header due to the first syntax element being equal to zero, and otherwise indicates that the slice type is signaled in a slice header.

[00178] C7. The method of solution C6, wherein the first syntax element is `mixed_slice_types_in_pic_flag`.

[00179] C8. The method of solution C6 or C7, wherein the first syntax element is signaled due to a second syntax element being equal to zero and/or a third syntax element being greater than zero.

[00180] C9. The method of solution C8, wherein the second syntax element specifies at least one characteristic of one or more tiles within each of the one or more slices.

[00181] C10. The method of solution C9, wherein the second syntax element equaling zero specifies that the one or more tiles are in raster scan order and slice information is excluded from the PPS.

[00182] C11. The method of solution C9, wherein the second syntax element equaling one specifies that the one or more tiles cover a rectangular region of the picture and slice information is signaled in the PPS.

[00183] C12. The method of any of solutions C9 to C11, wherein the second syntax element is `rect_slice_flag`.

[00184] C13. The method of solution C8, wherein the third syntax element specifies a number of rectangular slices in the picture that refer to the PPS.

[00185] C14. The method of solution C13, wherein the third syntax element is `num_slices_in_pic_minus1`.

[00186] C15. A method of video processing, comprising performing a conversion between a picture of a video and a bitstream of the video according to a rule, wherein the conversion comprises an in-loop filtering process, and wherein the rule specifies that a total number of vertical virtual boundaries and a total number of horizontal virtual boundaries related to the in-loop filtering process are signaled at a picture-level or a sequence-level.

[00187] C16. The method of solution C15, wherein the total number of vertical virtual boundaries comprise up to a first number (N1) of vertical virtual boundaries that are signaled in a

picture parameter set (PPS) or a sequence parameter set (SPS) and up to a second number (N3) of extra vertical virtual boundaries that are signaled in a picture header (PH), and wherein the total number of horizontal virtual boundaries comprise up to a third number (N2) of horizontal virtual boundaries that are signaled in the PPS or the SPS and up to a fourth number (N4) of extra horizontal virtual boundaries that are signaled in the PH.

[00188] C17. The method of solution C16, wherein $N1 + N3 \leq N1$ and $N2 + N4 \leq N2$.

[00189] C18. The method of solution C16 or C17, wherein $N1 = 3$, $N2 = 3$, $N3 = 1$, and $N4 = 1$.

[00190] Yet another listing of solutions preferred by some embodiments is provided next.

[00191] D1. A method of video processing, comprising performing a conversion between a video comprising one or more pictures and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule conditionally allows coding of a picture in one layer using reference pictures from other layers based on a first syntax element indicating whether the reference pictures from the other layers are present in the bitstream, and wherein the first syntax element is conditionally signaled in the bitstream based on a second syntax element that indicates whether an identifier of a parameter set associated with the picture is not equal to zero.

[00192] D2. The method of solution D1, wherein the first syntax element is `inter_layer_ref_pics_present_flag`.

[00193] D3. The method of solution D1 or D2, wherein the first syntax element specifies whether an inter-layer prediction is enabled and an inter-layer reference picture can be used.

[00194] D4. The method of solution D1, wherein the first syntax element is `sps_inter_layer_prediction_enabled_flag`.

[00195] D5. The method of any of solutions D1 to D4, wherein the first syntax element is in a sequence parameter set (SPS).

[00196] D6. The method of any of solutions D1 to D4, wherein the parameter set is a video parameter set (VPS).

[00197] D7. The method of solution D6, wherein the second syntax element is `sps_video_parameter_set_id`.

[00198] D8. The method of solution D6 or D7, wherein the parameter set provides an identifier for the VPS for reference by other syntax elements.

[00199] D9. The method of solution D7, wherein the second syntax element is in a sequence parameter set (SPS).

[00200] Yet another listing of solutions preferred by some embodiments is provided next.

[00201] E1. A method of video processing, comprising performing a conversion between a picture of a video and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule defines a first syntax element for enabling (a) a synchronization process for context variables before decoding a coding tree unit (CTU) in the picture and (b) a storage process for the context variables after decoding the CTU, wherein the first syntax element is signaled in a sequence parameter set (SPS) associated with the picture.

[00202] E2. The method of solution E1, wherein the first syntax element is excluded from a picture parameter set (PPS) associated with the picture.

[00203] E3. The method of solution E1 or E2, wherein the CTU comprises a first coding tree block (CTB) of a row of CTBs in each tile in the picture.

[00204] E4. The method of any of solutions E1 to E3, wherein the first syntax element is `entropy_coding_sync_enabled_flag`.

[00205] E5. The method of any of solutions E1 to E3, wherein the first syntax element is `sps_entropy_coding_sync_enabled_flag`.

[00206] E6. The method of any of solutions E1 to E5, wherein the format rule defines a second syntax element for indicating whether signaling for entry point offsets for tiles or tile-specific CTU rows are present in a slice header of the picture, and wherein the second syntax element is signaled in the SPS associated with the picture.

[00207] E7. A method of video processing, comprising performing a conversion between a picture of a video and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule defines a syntax element for indicating whether signaling for entry point offsets for tiles or tile-specific coding tree unit (CTU) rows are present in a slice header of the picture, and wherein the syntax element is signaled in a sequence parameter set (SPS) associated with the picture.

[00208] E8. The method of solution E7, wherein the format rule allows signaling of the entry point offsets based on a value of the syntax element being equal to one.

[00209] E9. The method of solution E7 or E8, wherein the syntax element is excluded from a picture parameter set (PPS) associated with the picture.

[00210] E10. The method of any of solutions E7 to E9, wherein the syntax element is entry_point_offsets_present_flag.

[00211] E11. The method of any of solutions E7 to E9, wherein the syntax element is sps_entry_point_offsets_present_flag.

[00212] Yet another listing of solutions preferred by some embodiments is provided next.

[00213] F1. A method of video processing, comprising performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that a first syntax element, which indicates a number of parameters for an output layer set (OLS) hypothetical reference decoder (HRD) in a video parameter set (VPS) associated with the video, is less than a first predetermined threshold.

[00214] F2. The method of solution F1, wherein the first syntax element is vps_num_ols_timing_hrd_params_minus1.

[00215] F3. The method of solution F1 or F2, wherein the first predetermined threshold is a number of multi-layer output layer sets (denoted NumMultiLayerOlss) minus one.

[00216] F4. The method of any of solutions F1 to F3, wherein the first syntax element is inferred to be zero when the first syntax element is not signaled in the bitstream.

[00217] F5. The method of any of solutions F1 to F4, wherein the rule specifies that a second syntax element, which indicates a number of profile/tier/level (PTL) syntax structures in a video parameter set (VPS) associated with the video, is less than a second predetermined threshold.

[00218] F6. A method of video processing, comprising performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that a syntax element, which indicates a number of profile/tier/level (PTL) syntax structures in a video parameter set (VPS) associated with the video, is less than a predetermined threshold.

[00219] F7. The method of solution F6, wherein the syntax element is vps_num_ptls_minus1.

[00220] F8. The method of solution F6 or F7, wherein the predetermined threshold is a total number of output layer sets (denoted TotalNumOlss).

[00221] F9. The method of any of solutions F1 to F8, wherein a difference between the predetermined threshold and the syntax element is signaled in the bitstream.

[00222] F10. The method of solution F9, wherein the difference is signaled using a unary code.

[00223] F11. The method of solution F9, wherein the difference is signaled using an exponential-Golomb (EG) code.

[00224] F12. A method of video processing, comprising performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that a first syntax element, which indicates a number of decoded picture buffer parameters syntax structures in a video parameters set (VPS), must be less than or equal to a second syntax element, which indicates a number of layers specified by the VPS.

[00225] F13. The method of solution F12, wherein a difference between the first syntax element and a predetermined threshold is signaled in the bitstream.

[00226] F14. The method of solution F12, wherein a difference between the second syntax element and a predetermined threshold is signaled in the bitstream.

[00227] F15. The method of solution F13 or F14, wherein the difference is signaled using a unary code.

[00228] F16. The method of solution F13 or F14, wherein the difference is signaled using an exponential-Golomb (EG) code.

[00229] F17. A method of video processing, comprising performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule allows a terminating network abstraction layer (NAL) unit to be made available to a decoder by signaling in the bitstream or providing through external means.

[00230] F18. The method of solution F17, wherein the terminating NAL unit is an end of bitstream (EOB) NAL unit.

[00231] F19. The method of solution F17, wherein the terminating NAL unit is an end of sequence (EOS) NAL unit.

[00232] F20. The method of any of solutions F17 to F19, wherein the external means include a parameter set.

[00233] F21. A method of video processing, comprising performing a conversion between a video and a bitstream of the video, wherein the bitstream conforms to a format rule, and wherein the format rule restricts each layer in the bitstream to contain only one subpicture due to a syntax

element being equal to zero, which indicates that the each layer is configured to use inter-layer prediction.

[00234] F22. The method of solution F21, wherein the syntax element is `vps_independent_layer_flag`.

[00235] Yet another listing of solutions preferred by some embodiments is provided next.

[00236] G1. A method of video processing, comprising performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that a sub-bitstream extraction process is implemented to generate a sub-bitstream for decoding, wherein the sub-bitstream extraction process is configured to extract, from the bitstream, a sub-bitstream with a target highest temporal identifier, and wherein, the rule specifies that, during the extracting, upon removing a video coding layer (VCL) network abstraction layer (NAL) unit, filler data units and filler supplemental enhancement information (SEI) messages in SEI NAL units that are associated with the VCL NAL unit are also removed.

[00237] G2. The method of solution G1, wherein the VCL NAL unit is removed based on an identifier of a layer that the VCL NAL unit belong to.

[00238] G3. The method of solution G2, wherein the identifier is `nuh_layer_id`.

[00239] G4. The method of solution G1, wherein the bitstream conforms to a format rule that specifies that an SEI NAL unit that comprises an SEI message with a filler payload does not include SEI messages with a payload different from the filler payload.

[00240] G5. The method of solution G1, wherein the bitstream conforms to a format rule that specifies that an SEI NAL unit that comprises an SEI message with a filler payload is configured to exclude another SEI message with any other payload.

[00241] G6. The method of solution G1, wherein the VCL NAL unit has a maximum of one associated filler NAL unit.

[00242] G7. A method of video processing, comprising performing a conversion between a video unit of a video and a bitstream of the video, wherein the bitstream conforms to a format rule, wherein the format rule specifies that the bitstream includes a first syntax element, which indicates whether the video unit is coded in a lossless mode or in a lossy mode, and wherein signaling a second syntax element, which indicates an escape sample in a palette mode applied to the video unit, is selectively included based on a value of the first syntax element.

[00243] G8. The method of solution G7, wherein a binarization method of the escape sample is based on the first syntax element.

[00244] G9. The method of solution G7, wherein a determination of a context coding in an arithmetic coding for the escape sample is based on the flag.

[00245] G10. The method of any of solutions G7 to G9, wherein the video unit is a coding unit (CU) or a coding tree unit (CTU).

[00246] Yet another listing of solutions preferred by some embodiments is provided next.

[00247] P1. A method of video processing, comprising performing a conversion between a picture of a video and a coded representation of the video, wherein a number of subpictures in the picture is included in the coded representation as a field whose bitwidth is dependent on a value of the number of subpictures.

[00248] P2. The method of solution P1, wherein the field represents the number of subpictures using a codeword.

[00249] P3. The method of solution P2, wherein the codeword comprises a Golomb codeword.

[00250] P4. The method of any of solutions P1 to P3, wherein the value of the number of subpictures is restricted to be less than or equal to an integer number of coding tree blocks that fit within the picture.

[00251] P5. The method of any of solutions P1 to P4, wherein the field is dependent on a coding level associated with the coded representation.

[00252] P6. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to omit a syntax element indicative of subpicture identifiers due to the video region not comprising any subpictures.

[00253] P7. The method of solution P6, wherein the coded representation includes a field having a 0 value indicating that the video region is not comprising any subpictures.

[00254] P8. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to omit identifiers of subpictures in the video region at a video region header level in the coded representation.

[00255] P9. The method of solution P8, wherein the coded representation identifies subpictures numerically according to an order in which the subpictures are listed in the video region header.

[00256] P10. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to include identifiers of subpictures and/or a length of the identifiers of subpictures in the video region at a sequence parameter set level or a picture parameter set level.

[00257] P11. The method of solution P10, wherein the length is included in the picture parameter set level.

[00258] P12. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to include a field in the coded representation at a video sequence level to indicate whether a subpicture identifier length field is included in the coded representation at the video sequence level.

[00259] P13. The method of solution P12, wherein the format rule specifies to set the field to “1” in case that another field in the coded representation indicates that a length identifier for the video region is included in the coded representation.

[00260] P14. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, and wherein the format rule specifies to include an indication in the coded representation to indicate whether the video region can be used as a reference picture.

[00261] P15. The method of solution P14, wherein the indication comprises a layer ID and an index or an ID value associated with the video region.

[00262] P16. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, and wherein the format rule specifies to include an indication in the coded representation to indicate whether the video region may use inter-layer prediction (ILP) from a plurality of sample values associated with video regions of reference layers.

[00263] P17. The method of solution P16, wherein the indication is included at a sequence level, a picture level or a video level.

[00264] P18. The method of solution P16, wherein the video regions of the reference layers comprise at least one collocated sample of a sample within the video region.

[00265] P19. The method of solution P16, wherein the indication is included in one or more supplemental enhancement information (SEI) messages.

[00266] P20. A method of video processing, comprising configuring, upon a determination that a first message and a second message that apply to an operation point in an access unit of a video are present in a coded representation of the video, the coded representation such the first message precedes the second message in decoding order; and performing, based on the configuring, a conversion between a video region of the video and the coded representation.

[00267] P21. The method of solution P20, wherein the first message comprises a buffering period (BP) supplemental enhancement information (SEI) message and the second message comprises a picture timing (PT) SEI message.

[00268] P22. The method of solution P20, wherein the first message comprises a buffering period (BP) supplemental enhancement information (SEI) message and the second message comprises a decoding unit information (DUI) SEI message.

[00269] P23. The method of solution P20, wherein the first message comprises a picture timing (PT) supplemental enhancement information (SEI) message and the second message comprises a decoding unit information (DUI) SEI message.

[00270] P24. The method of any of above claims, wherein the video region comprises a subpicture of the video.

[00271] P25. The method of any of above claims, wherein the conversion comprises parsing and decoding the coded representation to generate the video.

[00272] P26. The method of any of above claims, wherein the conversion comprises encoding the video to generate the coded representation.

[00273] P27. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of solutions P1 to P26.

[00274] P28. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of solutions P1 to P26.

[00275] P29. A computer program product having computer code stored thereon, the code, when executed by a processor, causes the processor to implement a method recited in any of solutions P1 to P26.

[00276] The following solutions apply to the technical solutions listed above.

[00277] O1. The method of any of the preceding solutions, wherein the conversion comprises decoding the video from the bitstream representation.

[00278] O2. The method of any of the preceding solutions, wherein the conversion comprises encoding the video into the bitstream representation.

[00279] O3. The method of any of the preceding solutions, wherein the conversion comprises generating the bitstream from the video, and wherein the method further comprises storing the bitstream in a non-transitory computer-readable recording medium.

[00280] O4. A method of storing a bitstream representing a video to a computer-readable recording medium, comprising generating a bitstream from a video according to a method described in any of the preceding solutions; and writing the bitstream to the computer-readable recording medium.

[00281] O5. A video processing apparatus comprising a processor configured to implement a method recited in any of the preceding solutions.

[00282] O6. A computer-readable medium having instructions stored thereon, the instructions, when executed, causing a processor to implement a method recited in any of the preceding solutions.

[00283] O7. A computer readable medium that stores the bitstream representation generated according to any of the preceding solutions.

[00284] O8. A video processing apparatus for storing a bitstream representation, wherein the video processing apparatus is configured to implement a method recited in any of the preceding solutions.

[00285] O9. A bitstream that is generated using a method recited in any of the preceding solutions, the bitstream being stored on a computer-readable medium.

[00286] The disclosed and other solutions, examples, embodiments, modules and the functional operations described in this document can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this document and

their structural equivalents, or in combinations of one or more of them. The disclosed and other embodiments can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more them. The term “data processing apparatus” encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[00287] A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[00288] The processes and logic flows described in this document can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[00289] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random-access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00290] While this patent document contains many specifics, these should not be construed as limitations on the scope of any subject matter or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular techniques. Certain features that are described in this patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[00291] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. Moreover, the separation of various system components in the embodiments described in this patent document should not be understood as requiring such separation in all embodiments.

[00292] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.