



(51) International Patent Classification:  
*H04N 19/174* (2014.01)

(21) International Application Number:  
PCT/US2021/034259

(22) International Filing Date:  
26 May 2021 (26.05.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:  
63/030,726 27 May 2020 (27.05.2020) US

(71) Applicant: **BYTEDANCE INC.** [US/US]; 12655 West  
Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Ange-  
les, California 90066 (US).

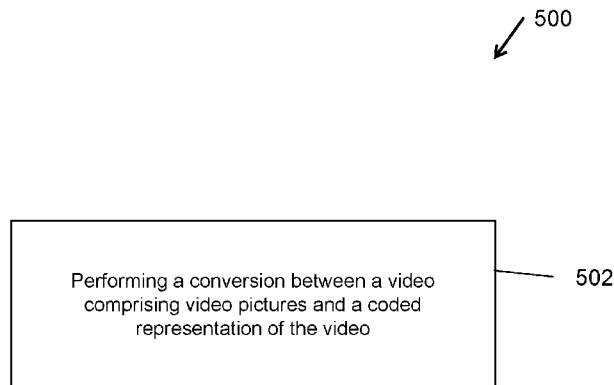
(72) Inventors: **WANG, Ye-kui**; 12655 West Jefferson Boule-  
vard, Sixth Floor, Suite No. 137, Los Angeles, California  
90066 (US). **ZHANG, Li**; 12655 West Jefferson Boule-  
vard, Sixth Floor, Suite No. 137, Los Angeles, California

90066 (US). **XU, Jizheng**; 12655 West Jefferson Boule-  
vard, Sixth Floor, Suite No. 137, Los Angeles, California  
90066 (US). **ZHANG, Kai**; 12655 West Jefferson Boule-  
vard, Sixth Floor, Suite No. 137, Los Angeles, California  
90066 (US).

(74) Agent: **SATHE, Vinay**; PERKINS COIE LLP, P.O. Box  
1247, Seattle, Washington 98111-1247 (US).

(81) Designated States (*unless otherwise indicated, for every  
kind of national protection available*): AE, AG, AL, AM,  
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,  
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO,  
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,  
HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN,  
KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD,  
ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO,  
NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW,  
SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN,  
TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(54) Title: USING SUBPICTURES AND TILES IN VIDEO CODING



**FIG. 5**

(57) Abstract: Several techniques for video encoding and video decoding are described. One example method includes performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that a quantization parameter table used in the conversion is included in a sequence parameter set, and wherein the rule specifies whether or how a first syntax element indicating a number of points in the quantization parameter table is included in the sequence parameter set is controlled according to a second syntax element in the sequence parameter set.



**(84) Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Declarations under Rule 4.17:**

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

**Published:**

— *with international search report (Art. 21(3))*

## USING SUBPICTURES AND TILES IN VIDEO CODING

### CROSS REFERENCE TO RELATED APPLICATIONS

[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 benefit of U.S. Provisional Patent Application No. 63/030,726, filed on May 27, 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 processing coded representation of video using control information useful for decoding of the coded representation.

[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 according to a rule, wherein the rule specifies that a quantization parameter table used in the conversion is included in a sequence parameter set, and wherein the rule specifies whether or how a first syntax element indicating a number of points in the quantization parameter table is included in the sequence parameter set is controlled according to a second syntax element in the sequence parameter set.

[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 according to a rule, wherein the rule specifies that one of the following two conditions is true when a number of subpictures minus 1 indicated in a sequence parameter set is greater than zero and when a syntax element in

the sequence parameter set indicates that an  $i$ -th subpicture of a first video picture of the video in a coded layer video sequence is treated as a second video picture of the video in a decoding process that excludes an in-loop filtering operation: (1) the second video picture referred to by each active entry in a first reference picture list or a second reference picture list of the first video picture and the first video picture have a same value for each of the following: a picture width in units of luma samples, a picture height in units of luma samples, the number of subpictures minus 1, and a horizontal position of a top-left coding tree unit of the  $i$ -th subpicture, a vertical position of the top-left CTU of the  $i$ -th subpicture, a width of the  $i$ -th subpicture minus 1, a height of the  $i$ -th subpicture minus 1, the syntax element for the  $i$ -th subpicture, respectively, for each value of  $i$  in a range of 0 to the number of subpictures minus 1, inclusive; and (2) the second video picture referred to by each active entry in the first reference picture list or the second reference picture list is an inter-layer reference picture for which a value of the number of subpictures minus 1 is equal to zero.

**[007]** In another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising a video block and a bitstream of the video according to a rule, wherein the rule specifies that a maximum number of subblock merge candidates is disallowed from being a number  $N$  when affine coding tool is enabled and sub-block based temporal motion vector prediction coding tool is disabled.

**[008]** In another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising one or more video pictures and a bitstream of the video, wherein the bitstream conforms to a format rule, and wherein the format rule specifies whether a syntax element that indicates whether affine merge mode is enabled is in the bitstream.

**[009]** In another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising one or more video pictures comprising one or more subpictures and a bitstream of the video according to a rule, wherein the rule specifies to indicate or omit one or more of the following subpicture related syntax elements from a sequence parameter set: a first syntax element indicating whether all subpictures referring to a same sequence parameter set have a same dimension, a second syntax element that indicates a common width and a common height of all subpictures having the same dimension, or one or more syntax elements according to a second rule are omitted when all subpictures do not have the same dimension.

**[0010]** In another example aspect, a video processing method is disclosed. The method includes determining, for a conversion between a video comprising one or more video pictures and a bitstream of the video, that a first syntax element in a picture parameter set that indicates a width of an  $i$ -th tile column in units of coding tree blocks minus 1 is equal to a width of a video picture in units of coding tree blocks minus 1 in response to a width of each video picture referring to the picture parameter set in units of luma samples being less than or equal to the luma coding tree block size; and performing the conversion based on the determining.

**[0011]** In another example aspect, a video processing method is disclosed. The method includes determining, for a conversion between a video comprising one or more video pictures and a bitstream of the video, that a first syntax element in a picture parameter set that indicates a height of an  $i$ -th tile row in units of coding tree blocks minus 1 is equal to a height of a video picture in units of coding tree blocks minus 1 in response to a height of each video picture referring to the picture parameter set in units of luma samples being less than or equal to a luma coding tree block size; and performing the conversion based on the determining.

**[0012]** In another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising video pictures and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies whether and where one or more syntax elements indicative of one or more initial quantization parameter (QP) values used during the conversion are included in the coded representation. In yet another example aspect, a video encoder apparatus is disclosed. The video encoder comprises a processor configured to implement above-described methods.

**[0013]** In another example aspect, another video processing method is disclosed. The method includes determining to use, for a conversion between a video block of a video and a coded representation of the video, an initial quantization parameter (QP) value; and performing a conversion based on the determining, wherein a syntax element in the coded representation is indicative of the initial QP value using a offset relationship comprising an offset value  $K$  that is (a) between 1 and 25 or (b) greater than 26 or (c) a function of maximum allowed QP value for the conversion or (d) a function of bit depth used for the conversion.

**[0014]** In another example aspect, another video processing method is disclosed. The method includes determining, for a conversion between a chroma block of a video and a coded representation of the video, one or more quantization parameter (QP) tables based on a type of a

picture or a slice that contains the chroma block according to a rule; and performing the conversion based on the determining.

**[0015]** In another example aspect, another video processing method is disclosed. The method includes determining, for a conversion between a chroma block of a video and a coded representation of the video, one or more quantization parameter (QP) tables; and performing the conversion based on the determining; wherein the conversion conforms to a first rule that specifies one or more syntax elements corresponding to the one or more QP tables or a second rule that specifies a relationship between the one or more QP tables and a coding characteristic of the chroma block.

**[0016]** In another example aspect, another video processing method is disclosed. The method includes performing a conversion between a chroma block of a video and a coded representation of the video; wherein the coded representation conforms to a format rule, wherein the format rule specifies a characteristic of a syntax element associated with one or more chroma quantization parameter (QP) tables used for the conversion.

**[0017]** In another example aspect, another video processing method is disclosed. The method includes determining, a signaling scheme used for signaling a quantization parameter table in a coded representation of a video based on a format rule related to one or more syntax elements in the coded representation, and performing, based on the determining, a conversion between the video and the coded representation.

**[0018]** In another example aspect, another video processing method is disclosed. The method includes performing a conversion between a video block of a video and a coded representation of the video according to a rule, wherein the rule specifies a number  $N$  that, in case that use of affine coding is enabled and use of sub-block based temporal motion vector prediction disabled, is used to limit a maximum number of subblock merge candidates.

**[0019]** In another example aspect, another video processing method is disclosed. The method includes performing a conversion between a video comprising one or more video pictures comprising one or more video subpictures and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to include or omit one or more of the following subpicture related syntax elements: a first syntax element indicating whether all subpictures referring to a same sequence parameter set have a same dimension, a second syntax element that signals a common width and a common height of all

subpictures having the same dimension, or a third syntax element according to a second format rule.

[0020] In yet another example aspect, a video decoder apparatus is disclosed. The video decoder comprises a processor configured to implement above-described methods.

[0021] In yet another example aspect, a computer readable medium having code stored thereon is disclose. The code embodies one of the methods described herein in the form of processor-executable code.

[0022] These, and other, features are described throughout the present document.

### BRIEF DESCRIPTION OF DRAWINGS

[0023] FIG. 1 is an illustration of the two scalar quantizers used in dependent quantization.

[0024] FIG.2 shows an example of state transition and quantizer selection for dependent quantization.

[0025] FIG. 3 is a block diagram of an example video processing system.

[0026] FIG. 4 is a block diagram of a video processing apparatus.

[0027] FIG. 5 is a flowchart for an example method of video processing.

[0028] FIG. 6 is a block diagram that illustrates a video coding system in accordance with some embodiments of the present disclosure.

[0029] FIG. 7 is a block diagram that illustrates an encoder in accordance with some embodiments of the present disclosure.

[0030] FIG. 8 is a block diagram that illustrates a decoder in accordance with some embodiments of the present disclosure.

[0031] FIGS. 9 to 15 are flowcharts for example methods of video processing.

### DETAILED DESCRIPTION

[0032] 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

[0033] This invention is related to video coding technologies. Specifically, it is related to initial quantization parameter (QP) and chroma QP table in video coding. It may be applied to the existing video coding standard like HEVC, or the standard (Versatile Video Coding) to be finalized. It may be also applicable to future video coding standards or video codec.

## 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
QP	Quantization Parameter
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. Video coding introduction

[0034] 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, 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). In April 2018, the Joint Video Expert Team (JVET) between VCEG (Q6/16) and ISO/IEC JTC1 SC29/WG11

(MPEG) was created to work on the VVC standard targeting at 50% bitrate reduction compared to HEVC.

[0035] The latest version of VVC draft, i.e., Versatile Video Coding (Draft 8) can be found at:

[0036] [http://phenix.int-evry.fr/jvet/doc\\_end\\_user/documents/17\\_Brussels/wg11/JVET-Q2001-v13.zip](http://phenix.int-evry.fr/jvet/doc_end_user/documents/17_Brussels/wg11/JVET-Q2001-v13.zip)

[0037] And the latest test model software can be found at:

[0038] [https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware\\_VTM/-/archive/VTM-8.0/VVCSoftware\\_VTM-VTM-8.0.zip](https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/archive/VTM-8.0/VVCSoftware_VTM-VTM-8.0.zip)

### 3.1. Quantization parameter control

[0039] In VVC, Maximum QP was extended from 51 to 63, and the signaling of initial QP was changed accordingly. The initial value of SliceQpY is modified at the slice segment layer when a non-zero value of slice\_qp\_delta is coded. Specifically, the value of init\_qp\_minus26 is modified to be in the range of  $(-26 + QpBdOffsetY)$  to +37. When the size of a transform block is not a power of 4, the transform coefficients are processed along with a modification to the QP or QP levelScale table rather than by multiplication by 181/256 (or 181/128), to compensate for an implicit scaling by the transform process. For transform skip block, minimum allowed Quantization Parameter (QP) is defined as 4 because quantization step size becomes 1 when QP is equal to 4.

[0040] In HEVC (and also in H.264), a fixed look-up table is used to convert the luma quantization parameter QPY to chroma quantization parameter QPC. In VVC, a more flexible luma-to-chroma QP mapping is used. Instead of having a fixed table, the luma-to-chroma QP mapping relationship is signalled in the SPS using a flexible piecewise linear model, with the only constraint on the linear model being that the slope of each piece cannot be negative (i.e., as luma QP increases, chroma QP must stay flat or increase, but cannot decrease). The piecewise linear model is defined by: 1) the number of pieces in the model; 2) input (luma) and output (chroma) delta QPs for that piece. The input range of the piecewise linear model is  $[-QpBdOffsetY, 63]$  and the output range of the piecewise linear model is  $[-QpBdOffsetC, 63]$ . The QP mapping relationship can be signalled separately for Cb, Cr and joint Cb/Cr coding, or signalled jointly for all three types of residual coding.

[0041] Same as in HEVC, CU-level QP adaptation is allowed in VVC. Delta QP values for luma and chroma components can be signalled separately. For the chroma components, the allowed chroma QP offset values are signalled in the form of offset lists in the PPS in a similar manner as in HEVC. The lists are defined separately for Cb, Cr and joint Cb/Cr coding. Up to 6 offset values are allowed for each of Cb, Cr, and joint Cb/Cr lists. At the CU-level, an index is signalled to indicate which one of the offset values in the offset list is used to adjust the chroma QP for that CU. CU chroma QP offset signalling is also consistent with the VPDU CU QP delta availability, and for CU larger than 64x64, send the chroma QP offset with the first transform unit regardless of whether it has non-zero CBF or not.

### 3.2. Dependent quantization

[0042] In addition, the same HEVC scalar quantization is used with a new concept called dependent scalar quantization. Dependent scalar quantization refers to an approach in which the set of admissible reconstruction values for a transform coefficient depends on the values of the transform coefficient levels that precede the current transform coefficient level in reconstruction order. The main effect of this approach is that, in comparison to conventional independent scalar quantization as used in HEVC, the admissible reconstruction vectors are packed denser in the N-dimensional vector space (N represents the number of transform coefficients in a transform block). That means, for a given average number of admissible reconstruction vectors per N-dimensional unit volume, the average distortion between an input vector and the closest reconstruction vector is reduced. The approach of dependent scalar quantization is realized by: (a) defining two scalar quantizers with different reconstruction levels and (b) defining a process for switching between the two scalar quantizers.

[0043] The two scalar quantizers used, denoted by Q0 and Q1, are illustrated in FIG. 1. The location of the available reconstruction levels is uniquely specified by a quantization step size  $\Delta$ . The scalar quantizer used (Q0 or Q1) is not explicitly signalled in the bitstream. Instead, the quantizer used for a current transform coefficient is determined by the parities of the transform coefficient levels that precede the current transform coefficient in coding/reconstruction order.

[0044] As illustrated in FIG. 2, the switching between the two scalar quantizers (Q0 and Q1) is realized via a state machine with four states. The state can take four different values: 0, 1, 2, 3. It is uniquely determined by the parities of the transform coefficient levels preceding the current transform coefficient in coding/reconstruction order. At the start of the inverse quantization for a transform block, the state is set equal to 0. The transform coefficients are reconstructed in scanning order (i.e., in the same order they are entropy decoded). After a current transform coefficient is reconstructed, the state is updated as shown in Figure 18, where k denotes the value of the transform coefficient level.

### 3.3. Scaling matrices

[0045] VVC supports to use the default scaling matrices or signal user-defined scaling matrices. The DEFAULT mode scaling matrices are all flat, with elements equal to 16 for all TB sizes. IBC and intra coding modes currently share the same scaling matrices. Thus, for the case of USER\_DEFINED matrices, the number of MatrixType and MatrixType\_DC are updated as follows:

- **MatrixType: 30 = 2** (2 for intra&IBC/inter)  $\times$  **3** (Y/Cb/Cr components)  $\times$  **5** (square TB size: from 4 $\times$ 4 to 64 $\times$ 64 for luma, from 4 $\times$ 4 to 32 $\times$ 32 for chroma)
- **MatrixType\_DC: 14 = 2** (2 for intra&IBC/inter  $\times$  1 for Y component)  $\times$  **3** (TB size: 16 $\times$ 16, 32 $\times$ 32, 64 $\times$ 64) + **4** (2 for intra&IBC/inter  $\times$  2 for Cb/Cr components)  $\times$  **2** (TB size: 16 $\times$ 16, 32 $\times$ 32)

[0046] The DC values are separately coded for following scaling matrices: 16 $\times$ 16, 32 $\times$ 32, and 64 $\times$ 64. For TBs of size smaller than 8 $\times$ 8, all elements in one scaling matrix are signalled. If the TBs have size greater than or equal to 8 $\times$ 8, only 64 elements in one 8 $\times$ 8 scaling matrix are signalled as a base scaling matrix. For obtaining square matrices of size greater than 8 $\times$ 8, the 8 $\times$ 8 base scaling matrix is up-sampled (by duplication of elements) to the corresponding square size (i.e. 16 $\times$ 16, 32 $\times$ 32, 64 $\times$ 64). when the zeroing-out of the high frequency coefficients for 64-point transform is applied, corresponding high frequencies of the scaling matrices are also zeroed out. That is, if the width or height of the TB is greater than or equal to 32, only left or top half of the coefficients is kept, and the remaining coefficients are assigned to zero. Moreover, the number of elements signalled for the 64 $\times$ 64 scaling matrix is also reduced from 8 $\times$ 8 to three 4 $\times$ 4 submatrices, since the bottom-right 4 $\times$ 4 elements are never used. In VVC, 2x2, 2x4, and 4x2

chroma intra coding blocks (CBs) do not exist, and the smallest intra block size is equal to 2x8 and 8x2, as well as the smallest chroma intra block copy (IBC) block size. Furthermore, inter-prediction is disabled for 4x4 luma CBs. Therefore, small 2x2 chroma blocks can be created only by applying a subblock transform (SBT). Considering these essences, 2x2 intra chroma quantization matrices (QMs) are removed from the default QM list, and not code user-defined intra QMs for this size.

**[0047]** In order to improve coding efficiency for user defined quantization matrixes (QM), following approaches are considered.

- Allow referencing a previously coded QM with the same base size as the current QM.
- Allow coding element-to-element differences between the current QM and the reference QM.
- Keep the original DPCM coding of elements within the current QM.
- Use a single matrix identifier `scalingListId` which combines `matrixId` and `sizeId`.

### 3.4. Joint coding of chroma residuals

**[0048]** VVC supports a mode where the chroma residuals are coded jointly; this is named as JCCR (Joint Coding of Chroma Residuals). The usage (activation) of a joint chroma coding mode is indicated by a TU-level flag `tu_joint_cbr_residual_flag` and the selected mode is implicitly indicated by the chroma CBFs. The flag `tu_joint_cbr_residual_flag` is present if either or both chroma CBFs for a TU are equal to 1. In the PPS and slice header, chroma QP offset values are signalled for the joint chroma residual coding mode to differentiate from the usual chroma QP offset values signalled for regular chroma residual coding mode. These chroma QP offset values are used to derive the chroma QP values for those blocks coded using the joint chroma residual coding mode. When a corresponding joint chroma coding mode (modes 2 in the following table) is active in a TU, this chroma QP offset is added to the applied luma-derived chroma QP during quantization and decoding of that TU. For the other modes (modes 1 and 3 in the following table), the chroma QPs are derived in the same way as for conventional Cb or Cr blocks. The reconstruction process of the chroma residuals (`resCb` and `resCr`) from the transmitted transform blocks is depicted in Table 1. When this mode is activated, one single joint chroma residual block (`resJointC[x][y]` in the following table) is signalled, and residual block for

Cb (resCb) and residual block for Cr (resCr) are derived considering information such as tu\_cbf\_cb, tu\_cbf\_cr, and CSign, which is a sign value specified in the slice header.

At the encoder side, the joint chroma components are derived as explained in the following. Depending on the mode (listed in the tables above), resJointC{1,2} are generated by the encoder as follows:

- If mode is equal to 2 (single residual with reconstruction Cb = C, Cr = CSign \* C), the joint residual is determined according to

$$\text{resJointC}[x][y] = (\text{resCb}[x][y] + \text{CSign} * \text{resCr}[x][y]) / 2$$

- Otherwise, if mode is equal to 1 (single residual with reconstruction Cb = C, Cr = (CSign \* C) / 2), the joint residual is determined according to

$$\text{resJointC}[x][y] = (4 * \text{resCb}[x][y] + 2 * \text{CSign} * \text{resCr}[x][y]) / 5$$

- Otherwise (mode is equal to 3, i. e., single residual, reconstruction Cr = C, Cb = (CSign \* C) / 2), the joint residual is determined according to

$$\text{resJointC}[x][y] = (4 * \text{resCr}[x][y] + 2 * \text{CSign} * \text{resCb}[x][y]) / 5$$

Table 1. Reconstruction of chroma residuals. The value CSign is a sign value (+1 or -1), which is specified in the slice header, resJointC[ ][ ] is the transmitted residual.

tu_cbf_c b	tu_cbf_cr	reconstruction of Cb and Cr residuals	mode
1	0	resCb[x][y] = resJointC[x][y] resCr[x][y] = (CSign * resJointC[x][y]) >> 1	1
1	1	resCb[x][y] = resJointC[x][y] resCr[x][y] = CSign * resJointC[x][y]	2
0	1	resCb[x][y] = (CSign * resJointC[x][y]) >> 1 resCr[x][y] = resJointC[x][y]	3

[0049] The three joint chroma coding modes described above are only supported in I slices. In P and B slices, only mode 2 is supported. Hence, in P and B slices, the syntax element

*tu\_joint\_cbr\_residual\_flag* is only present if both chroma cbfs are 1. Note that transform depth is removed in the context modeling of *tu\_cbf\_luma* and *tu\_cbf\_cb*.

### 3.5. Chroma QP table in SPS

[0050] In clause 7.3.2.3 of JVET-Q2001-vC, the SPS includes a structure named chroma QP table, shown as follows:

	<b>Descriptor</b>
seq_parameter_set_rbsp() {	
.....	
if( ChromaArrayType != 0 ) {	
<b>sps_joint_cbr_enabled_flag</b>	u(1)
<b>same_qp_table_for_chroma</b>	u(1)
numQpTables = same_qp_table_for_chroma ? 1 : (	
sps_joint_cbr_enabled_flag ? 3 : 2 )	
for( i = 0; i < numQpTables; i++ ) {	
<b>qp_table_start_minus26[ i ]</b>	se(v)
<b>num_points_in_qp_table_minus1[ i ]</b>	ue(v)
for( j = 0; j <= num_points_in_qp_table_minus1[ i ]; j++ ) {	
<b>delta_qp_in_val_minus1[ i ][ j ]</b>	ue(v)
<b>delta_qp_diff_val[ i ][ j ]</b>	ue(v)
}	
}	
}	
}	
.....	

They are with the following semantics and QP table derivation:

**sps\_joint\_cbr\_enabled\_flag** equal to 0 specifies that the joint coding of chroma residuals is disabled. **sps\_joint\_cbr\_enabled\_flag** equal to 1 specifies that the joint coding of chroma

residuals is enabled. When not present, the value of `sps_joint_cbr_enabled_flag` is inferred to be equal to 0.

`same_qp_table_for_chroma` equal to 1 specifies that only one chroma QP mapping table is signalled and this table applies to Cb and Cr residuals and additionally to joint Cb-Cr residuals when `sps_joint_cbr_enabled_flag` is equal to 1. `same_qp_table_for_chroma` equal to 0 specifies that chroma QP mapping tables, two for Cb and Cr, and one additional for joint Cb-Cr when `sps_joint_cbr_enabled_flag` is equal to 1, are signalled in the SPS. When `same_qp_table_for_chroma` is not present in the bitstream, the value of `same_qp_table_for_chroma` is inferred to be equal to 1.

`qp_table_start_minus26[ i ]` plus 26 specifies the starting luma and chroma QP used to describe the *i*-th chroma QP mapping table. The value of `qp_table_start_minus26[ i ]` shall be in the range of  $-26 - \text{QpBdOffset}$  to 36 inclusive. When `qp_table_start_minus26[ i ]` is not present in the bitstream, the value of `qp_table_start_minus26[ i ]` is inferred to be equal to 0.

`num_points_in_qp_table_minus1[ i ]` plus 1 specifies the number of points used to describe the *i*-th chroma QP mapping table. The value of `num_points_in_qp_table_minus1[ i ]` shall be in the range of 0 to  $63 + \text{QpBdOffset}$ , inclusive. When `num_points_in_qp_table_minus1[ 0 ]` is not present in the bitstream, the value of `num_points_in_qp_table_minus1[ 0 ]` is inferred to be equal to 0.

`delta_qp_in_val_minus1[ i ][ j ]` specifies a delta value used to derive the input coordinate of the *j*-th pivot point of the *i*-th chroma QP mapping table. When `delta_qp_in_val_minus1[ 0 ][ j ]` is not present in the bitstream, the value of `delta_qp_in_val_minus1[ 0 ][ j ]` is inferred to be equal to 0.

`delta_qp_diff_val[ i ][ j ]` specifies a delta value used to derive the output coordinate of the *j*-th pivot point of the *i*-th chroma QP mapping table.

The *i*-th chroma QP mapping table `ChromaQpTable[ i ]` for  $i = 0..numQpTables - 1$  is derived as follows:

$$\begin{aligned} \text{qpInVal}[ i ][ 0 ] &= \text{qp\_table\_start\_minus26}[ i ] + 26 \\ \text{qpOutVal}[ i ][ 0 ] &= \text{qpInVal}[ i ][ 0 ] \end{aligned}$$

```

for(j = 0; j <= num_points_in_qp_table_minus1[ i ]; j++) {
    qpInVal[ i ][ j + 1 ] = qpInVal[ i ][ j ] + delta_qp_in_val_minus1[ i ][ j ] + 1
    qpOutVal[ i ][ j + 1 ] = qpOutVal[ i ][ j ] +
    ( delta_qp_in_val_minus1[ i ][ j ] ^ delta_qp_diff_val[ i ][ j ] )
}
ChromaQpTable[ i ][ qpInVal[ i ][ 0 ] ] = qpOutVal[ i ][ 0 ]
for( k = qpInVal[ i ][ 0 ] - 1; k >= -QpBdOffset; k -- )
    ChromaQpTable[ i ][ k ] = Clip3( -QpBdOffset, 63, ChromaQpTable[ i ][ k + 1 ] - 1 )
for(j = 0; j <= num_points_in_qp_table_minus1[ i ]; j++) {
    sh = ( delta_qp_in_val_minus1[ i ][ j ] + 1 ) >> 1
    for( k = qpInVal[ i ][ j ] + 1, m = 1; k <= qpInVal[ i ][ j + 1 ]; k++, m++)
        ChromaQpTable[ i ][ k ] = ChromaQpTable[ i ][ qpInVal[ i ][ j ] ] +
        ( ( qpOutVal[ i ][ j + 1 ] - qpOutVal[ i ][ j ] ) * m + sh ) /
    ( delta_qp_in_val_minus1[ i ][ j ] + 1 )
}
for( k = qpInVal[ i ][ num_points_in_qp_table_minus1[ i ] + 1 ] + 1; k <= 63; k++)
    ChromaQpTable[ i ][ k ] = Clip3( -QpBdOffset, 63, ChromaQpTable[ i ][ k - 1 ] + 1 )

```

When `same_qp_table_for_chroma` is equal to 1, `ChromaQpTable[ 1 ][ k ]` and `ChromaQpTable[ 2 ][ k ]` are set equal to `ChromaQpTable[ 0 ][ k ]` for `k` in the range of `-QpBdOffset` to 63, inclusive.

It is a requirement of bitstream conformance that the values of `qpInVal[ i ][ j ]` and `qpOutVal[ i ][ j ]` shall be in the range of `-QpBdOffset` to 63, inclusive for `i` in the range of 0 to `numQpTables - 1`, inclusive, and `j` in the range of 0 to `num_points_in_qp_table_minus1[ i ] + 1`, inclusive.

In the above description, `QpBdOffset` is derived as:

**bit\_depth\_minus8** specifies the bit depth of the samples of the luma and chroma arrays, `BitDepth`, and the value of the luma and chroma quantization parameter range offset, `QpBdOffset`, as follows:

$$\text{BitDepth} = 8 + \text{bit\_depth\_minus8}$$

$$\text{QpBdOffset} = 6 * \text{bit\_depth\_minus8}$$

bit\_depth\_minus8 shall be in the range of 0 to 8, inclusive.

### 3.6. Initial QP in PPS

[0051] There is a syntax element, named **init\_qp\_minus26**, in PPS. The semantics is as follows: **init\_qp\_minus26** plus 26 specifies the initial value of SliceQ<sub>PY</sub> for each slice referring to the PPS. The initial value of SliceQ<sub>PY</sub> is modified at the picture level when a non-zero value of ph\_qp\_delta is decoded or at the slice level when a non-zero value of slice\_qp\_delta is decoded. The value of init\_qp\_minus26 shall be in the range of -( 26 + QpBdOffset ) to +37, inclusive.

When qp\_delta\_info\_in\_ph\_flag is equal to 1, the initial value of the Q<sub>PY</sub> quantization parameter for all slices of the picture, SliceQ<sub>PY</sub>, is derived as follows:

$$\text{SliceQ}_{PY} = 26 + \text{init\_qp\_minus26} + \text{ph\_qp\_delta}$$

When qp\_delta\_info\_in\_ph\_flag is equal to 0, the initial value of the Q<sub>PY</sub> quantization parameter for the slice, SliceQ<sub>PY</sub>, is derived as follows:

$$\text{SliceQ}_{PY} = 26 + \text{init\_qp\_minus26} + \text{slice\_qp\_delta}$$

The value of SliceQ<sub>PY</sub> shall be in the range of -QpBdOffset to +63, inclusive.

## 4. Technical problems solved by disclosed technical solutions and embodiments

1. The signalling of the initial QP value (i.e., **init\_qp\_minus26**) may not be reasonable.
2. I and B/P pictures or slices may need different chroma QP tables, however, they share the same in the current VVC draft text.
3. Transform skip may need another chroma QP table.
4. Chroma QP table start point should cover the whole QP range which is not the case in the current design.
5. Chroma QP table may have a biased start point.
6. The number of points in a chroma QP table should be able to be 0, but this is disallowed in the current design since the **num\_points\_in\_qp\_table\_minus1** is coded and the

minimum allowed value of the syntax element is equal to 0 and consequently the minimum number of points is 1.

7. Specifying a QP clipping in the derivation of chroma QP may be better than specifying a bitstream conformance constraint for limiting the derived chroma QP value, as the former would prevent a violation from happening.
8. In the latest VVC draft text, the SH syntax element **slice\_ts\_residual\_coding\_disabled\_flag** is used to specify whether transform skip based residual coding (TSRC) or regular residual coding (RRC) is used for a transform block. However, there might be higher-level (SPS/PPS) and/or lower-level (CU/TU) level control flags for whether the current block is using TSRC or RRC. Moreover, the interactions between different level control flags, and between the control flag and the transform skip flags would be further specified.
9. In current VVC, chroma ALF and CC-ALF are disabled implicitly when luma ALF is disabled at SPS/PH/SH. However, such a restriction is not applied at CTU level. When luma ALF is disabled for a CTU, the chroma ALF and CC-ALF may still be applied for the CTU. Such a design is conflicting with the intention of controlling chroma ALF/CC-ALF based on luma ALF at higher levels
10. Chroma QP tables may have a default set of values, dependent on colour format.
11. In the sub-bitstream extraction process specified in clause C.6 of the latest VVC draft in JVET-Q2001-vE, one step is specified as follows:
  - Remove from outBitstream all NAL units for which all of the following conditions are true:
    - nal\_unit\_type is not equal to IDR\_W\_RADL, IDR\_N\_LP, or CRA\_NUT.
    - nuh\_layer\_id is equal to LayerIdInOls[ targetOlsIdx ][ j ] for a value of j in the range of 0 to NumLayersInOls[ targetOlsIdx ] – 1 inclusive.
    - TemporalId is greater than or equal to NumSubLayersInLayerInOLS[ targetOlsIdx ][ j ].

However, that means parameter sets etc. non-VCL NAL units that satisfy the all the three conditions would also be removed. That is not desirable, and parameter sets, e.g., PPSs and

APSs can be shared by pictures with different TemporalId values, and can be placed at the beginning of the bitstream.

12. The current spec allows 8 ALF\_APS, 8 SCALING\_APS, and 3 LMCS APS, respectively. However, the number of APS NAL units allowed for pictures may be not enough, e.g., a typical use case is that a picture contains 96 subpictures and each subpicture may require different ALF\_APS. In such case, current eight ALF\_APS NAL units are not sufficient to achieve a decent coding efficiency. Therefore, there is a need to support a larger number of APSs.
13. JVET-S0071 proposes to add a shortcut for signalling of subpictures layout to the VVC design in JVET-R2001-vA/v10, as follows:
  - a. Add `sps_subpic_same_res_flag`, equal to 1 specifies that all subpictures have the same width and height, respectively.
  - b. When `sps_subpic_same_res_flag` is equal to 1, `sps_subpic_width_minus1[ i ]` and `sps_subpic_height_minus1[ i ]` are only signalled when `i` is equal to 0, and `sps_subpic_ctu_top_left_x[ i ]` and `sps_subpic_ctu_top_left_y[ i ]` are skipped for all `i` values.

However, the syntax design is somewhat messy, and when all subpictures have the same width and height, respectively, the number of subpictures can be derived thus should not be signalled.

## 5. Example listing of embodiments and solutions

**[0052]** To solve the above problems, and others, methods as summarized below are disclosed.

The items should be considered as examples to explain the general concepts and should not be interpreted in a narrow way. Furthermore, these items can be applied individually or combined in any manner.

**[0053]** In the following,  $\text{floor}(x)$  denotes a function that returns the greatest integer that is less than or equal to  $x$ .

1. The initial QP value syntax element, which is used to specify the initial QP values, e.g., currently denoted as `init_qp_minus26` in JVET-Q2001-vC, in PPS may be moved to PH.

- a. Alternatively, the initial QP value syntax element may be repeated at PH.
  - b. Alternatively, the initial QP value syntax element may be signalled in both PPS and PH, and the initial QP value syntax element signalled in PH may override or update (by adding the signalled delta) the initial QP value syntax element signalled in PPS.
  - c. Alternatively, the initial QP value syntax element may be signalled in the SPS, possibly also signalled in one or more of the PPS, PH, and SH, and when present, the value at a lower level overrides or updates (by adding the signalled delta) the value signalled at a higher level. In the case of updating, the initial QP value signalled at the highest level is  $ue(v)$  coded, and the delta values signalled in the lower levels are  $se(v)$  coded.
2. Multiple syntax elements for indications of initial QP values may be signalled in SPS/PPS/PH/SH according to specific types.
- a. In one example, each of them may be corresponding to a specific type.
  - b. In one example, the specific type may include a picture/slice type (e.g., I/P/B; Intra/Inter).
  - c. In one example, the specific type may include a video content type (e.g., screen content or camera captured content).
  - d. In one example, the specific type may include an index of a subpicture or other subpicture identification information, i.e., different subpictures may be associated with different syntax elements for initial QPs derivation.
  - e. In one example, the specific type may include an index of a slice or other slice identification information, i.e., different slices may be associated with different syntax elements for initial QPs derivation.
  - f. In one example, the specific type may include an index of a tile or other tile identification information, i.e., different tiles may be associated with different syntax elements for initial QPs derivation.
  - g. In one example, the specific type may include a transform type. (e.g., transform skip mode or not transform skip mode).
3. The initial QP value syntax element (e.g., in SH/PH/PPS/SPS) may be offsetted by a number  $K$  not equal to 26.
- a. In one example,  $K$  is smaller than 26.

- b. In one example, the syntax element may be replaced by `init_qp_minusK` and/or the value of the syntax element shall be in the range of  $-(K+QpBdOffset)$  to  $(63-K)$ , inclusive, where  $K$  is smaller than 26.
    - i. In one example,  $K$  is equal to 20. Alternatively, furthermore, the syntax element may be replaced by `init_qp_minus20` and/or the value of the syntax element shall be in the range of  $-(20+QpBdOffset)$  to 43 (i.e.,  $63-20$ ), inclusive.
  - c. In one example,  $K$  is larger than 26.
    - i. In one example, the syntax element may be replaced by `init_qp_minusK`, and the value of the syntax element shall be in the range of  $-(K+QpBdOffset)$  to  $(63-K)$ , inclusive, where  $K$  is a constant larger than 26.
    - ii. In one example,  $K$  is equal to 32. Alternatively, furthermore, the syntax element may be replaced by `init_qp_minus32` and the value of the syntax element shall be in the range of  $-(32+QpBdOffset)$  to 31 (i.e.,  $63-32$ ), inclusive.
  - d. Alternatively,  $K$  may be set to a function of the maximum allowed QP and/or the the minimum allowed QP. For example,  $K$  may be set equal to  $(\text{the maximum allowed QP} - \text{the minimum allowed QP})/2$  or  $(\text{the maximum allowed QP} + 1 - \text{the minimum allowed QP})/2$ .
  - e. In above example,  $K$  may be dependent on picture/slice/block types; and/or prediction modes and/or bitdepth.
    - i. In one example, the value of  $K$  for a I slice/picture is no greater than that for a P/B slices/pictures.
4. The initial QP value syntax element (e.g., in SH/PH/PPS) may be offseted by a number dependent on the internal bit depth.
- a. In one example, the initial QP value syntax element may be offseted by  $\text{floor}((-QpBdOffset+M)/2)$ .
    - i. Alternatively, furthermore, the value of the syntax element plus  $\text{floor}((-QpBdOffset+M)/2)$  shall be in the range of  $-QpBdOffset$  to 63, inclusive.

- b. In one example, the initial QP value syntax element may be offsetted by  $(\text{floor}((-QpBdOffset+M)/2) + K)$ , where K is a constant number.
    - i. Alternatively, furthermore, the value of the syntax element plus  $(\text{floor}((-QpBdOffset+M)/2) + K)$  shall be in the range of  $-QpBdOffset$  to N, inclusive.
  - c. In above examples, N may be set to the maximum allowed QP value (e.g., 63).
  - d. In above examples, M may be set to the maximum allowed QP value (e.g., 63) or maximum allowed QP value plus/minus 1.
5. Chroma QP tables may be determined according to picture/slice types.
- a. In one example, different picture/slice types may have different sets of chroma QP tables.
  - b. In one example, I pictures/slices may have their own chroma QP tables.
  - c. In one example, I and B and P pictures/slices may each have their own chroma QP tables.
  - d. In one example, B and P pictures/slices may share the same set of chroma QP tables.
  - e. In one example, the picture/slice types may be classified to M cases (M=2 for I only, BP share; M=3 for I/B/P separate). the number of chroma QP tables to be signalled may be dependent on M.
    - i. In one example, number of chroma QP tables to be signalled may be set to  $M * \text{numQpTables}$  wherein  $(\text{numQpTables} = \text{same\_qp\_table\_for\_chroma} ? 1 : (\text{sps\_joint\_cbcr\_enabled\_flag} ? 3 : 2))$  wherein M is greater than 1.
      - 1) Alternatively, furthermore, the semantics of `same_qp_table_for_chroma` may be further modified to ‘`same_qp_table_for_chroma` equal to 1 specifies that only one chroma QP mapping table is signalled for a picture/slice type’.
    - ii. In one example, furthermore, number of chroma QP tables to be signalled may be set to  $(\text{same\_qp\_table\_for\_chroma} ? 1 : (M * (\text{sps\_joint\_cbcr\_enabled\_flag} ? 3 : 2)))$  wherein M is greater than 1.
    - iii. In one example, for each category, an indication of whether all blocks share the same chroma QP tables is firstly signalled, and number of QP tables

may be set to (same qp table for chroma ? 1 : ( sps\_joint\_cbr\_enabled\_flag ? 3 : 2 )), followed by detailed information of chroma QP tables.

- 1) Alternatively, furthermore, for all categories, indications of the starting luma and chroma QP used to describe a chroma QP mapping table may be further signalled, such as using the actual value minus K (e.g., K = 26).
  - 2) Alternatively, furthermore, indications of the starting luma and chroma QP used to describe a chroma QP mapping table may be further signalled, such as using the actual value minus K (e.g., K = 1 for intra slices/pictures or K=32 for inter slices/pictures) and K depending on category index.
6. An indication, e.g., `sps_non_intra_present_flag`, may be added into SPS to indicate if the current sequence may contain B/P slices.
- a. In one example, when the flag is 1, it indicates that all slices in the current sequences are of intra slices.
  - b. In one example, when the flag is 0, it indicates that there may be B/P slices in the current sequence.
7. Either one or two sets of chroma QP tables may be signalled in the SPS.
- a. A flag is added to the SPS, e.g., named `sps_one_set_of_chroma_qp_tables_flag`.
  - b. `sps_one_set_of_chroma_qp_tables_flag` equal to 1 specifies the presence of one set of chroma QP tables in the SPS (i.e., as in the current VVC draft text), and this set of chroma QP tables only applies to intra coded entities (pictures, slices, CTUS, CUs, or coded blocks).
    - i. In one example, the only set of chroma QP tables applies only to intra pictures (for which all slices are intra slices, i.e., I slices) in CLVSs referring to the SPS.
    - ii. In one example, the only set of chroma QP tables applies only to intra slices in CLVSs referring to the SPS.
    - iii. In one example, the only set of chroma QP tables applies only to intra CTUs in CLVSs referring to the SPS.

- iv. In one example, the only set of chroma QP tables applies only to intra CUs in CLVSs referring to the SPS.
- v. In one example, the only set of chroma QP tables applies only to intra coded blocks in CLVSs referring to the SPS.
- c. `sps_one_set_of_chroma_qp_tables_flag` equal to 1 specifies the presence of one set of chroma QP tables in the SPS (i.e., as in the current VVC draft text), and this set of chroma QP tables applies to both intra and inter coded entities (pictures, slices, CTUS, CUs, or coded blocks).
- d. `sps_one_set_of_chroma_qp_tables_flag` equal to 0 specifies the presence of two set of chroma QP tables in the SPS (i.e., adding one more set of chroma QP tables), the 0-th set of chroma QP tables only applies to intra coded entities (pictures, slices, CTUS, CUs, or coded blocks), and the 1-th set of chroma QP tables only applies to inter coded entities (pictures, slices, CTUS, CUs, or coded blocks).
  - i. In one example, the 0-th set of chroma QP tables applies only to intra pictures (for which all slices are intra slices) in CLVSs referring to the SPS, and the 1-th set of chroma QP tables applies only to inter pictures (for which all slices are inter slices, i.e., B or P slices) in CLVSs referring to the SPS.
  - ii. In one example, the 0-th set of chroma QP tables applies only to intra slices in CLVSs referring to the SPS, and the 1-th set of chroma QP tables applies only to inter slices in CLVSs referring to the SPS.
  - iii. In one example, the 0-th set of chroma QP tables applies only to intra CTUs in CLVSs referring to the SPS, and the 1-th set of chroma QP tables applies only to inter CTUs in CLVSs referring to the SPS.
  - iv. In one example, the 0-th set of chroma QP tables applies only to intra CUs in CLVSs referring to the SPS, and the 1-th set of chroma QP tables applies only to inter CUs in CLVSs referring to the SPS.
  - v. In one example, the 0-th set of chroma QP tables applies only to intra coded blocks in CLVSs referring to the SPS, and the 1-th set of chroma QP tables applies only to inter coded blocks in CLVSs referring to the SPS.
- e. The flag may be sent only when `ChromaArrayType` is not equal to 0.
- f. An exemplar implementation is shown in embodiment 4.

- g. Another exemplar implementation is shown in embodiment 5.
8. Chroma QP tables may be determined according to prediction modes.
- a. In one example, intra CUs and other CUs may have a different set of chroma QP tables.
    - i. In one example, it may only apply on dual tree and/or local dual tree.
  - b. Alternatively, intra/palette CUs and other CUs may have a different set of chroma QP tables.
  - c. Alternatively, intra/IBC/palette CUs and other CUs may have a different set of chroma QP tables.
  - d. Alternatively, intra/IBC CUs and other CUs may have a different set of chroma QP table.
  - e. In above examples, the number of chroma QP tables to be signalled may be dependent on number of the classified sets of prediction modes.
  - f. In above examples, the prediction mode may mean the prediction mode of the luma CB.
9. Transform skip blocks may have a different set of chroma QP tables.
- a. In one example, it may only apply to luma blocks coded in transform skip mode.
10. The maximum allowed value of the chroma QP table start point syntax element, i.e., currently denoted as **qp\_table\_start\_minus26** in JVET-Q2001-vC, may be 37.
11. The chroma QP table start point syntax element, i.e., currently denoted as **qp\_table\_start\_minus26** in JVET-Q2001-vC, may be offsetted by a number K smaller than 26.
- a. In one example, the syntax element may be replaced by **qp\_table\_start\_minusK** and/or the value of the syntax element plus K shall be in the range of  $-(K+QpBdOffset)$  to  $(M-K)$ , inclusive, where K is smaller than 26.
  - b. In one example, the syntax element may be replaced by **qp\_table\_start\_minusK** and/or the value of the syntax element shall be in the range of  $-(K+QpBdOffset)$  to  $(M-1-K)$ , inclusive, where K is smaller than 26.
    - i. In one example, K is equal to 20. Alternatively, furthermore, the syntax element may be replaced by **init\_qp\_minus20** and/or the value of the syntax

- element plus 20 shall be in the range of  $-(20+QpBdOffset)$  to  $(M-20)$ , inclusive.
- ii. In one example,  $K$  is equal to 20. Alternatively, furthermore, the syntax element may be replaced by `init_qp_minus20` and/or the value of the syntax element plus 20 shall be in the range of  $-(20+QpBdOffset)$  to  $(M-1-20)$ , inclusive.
- c. Alternatively, the initial QP value syntax element may be offsetted by a number  $K$  larger than 26.
- i. In one example, the syntax element may be replaced by `init_qp_minusK`, and the value of the syntax element plus  $K$  shall be in the range of  $-(K+QpBdOffset)$  to  $(M-K)$ , inclusive, where  $K$  is a constant larger than 26.
  - ii. In one example, the syntax element may be replaced by `init_qp_minusK`, and the value of the syntax element plus  $K$  shall be in the range of  $-(K+QpBdOffset)$  to  $(M-1-K)$ , inclusive, where  $K$  is a constant larger than 26.
    - 1) In one example,  $K$  is equal to 32. Alternatively, furthermore, the syntax element may be replaced by `init_qp_minus32` and the value of the syntax element plus 32 shall be in the range of  $-(32+QpBdOffset)$  to  $(M-32)$ , inclusive.
    - 2) In one example,  $K$  is equal to 32. Alternatively, furthermore, the syntax element may be replaced by `init_qp_minus32` and the value of the syntax element plus 32 shall be in the range of  $-(32+QpBdOffset)$  to  $(M-1-32)$ , inclusive.
- d. In above example,  $K$  may be dependent on picture/slice types; and/or prediction modes and/or bit depth.
- i. In one example, for Intra picture/slices,  $K$  is set to 1.
  - ii. In one example, for P/B picture/slices,  $K$  is set to 32.
- e. In above example,  $M$  may be the maximum allowed QP value, e.g., 63.
- f. In one example,  $K$  is 0.

- i. Alternatively, furthermore, the syntax element may be binarized with  $uv(e)$  instead of  $sv(e)$ .
- 12. The chroma QP table start point syntax element, i.e., currently denoted as **qp\_table\_start\_minus26** in JVET-Q2001-vC, may be offsetted by a value depend on if the current picture is intra only picture.
  - a. Alternatively, the offset may depend on **intra\_only\_constraint\_flag**.
- 13. The syntax element related to the number of pivot points in a chroma QP table, i.e., currently denoted as **num\_points\_in\_qp\_table\_minus1**, should be able to present 0 number of points.
  - a. In one example, the syntax element **num\_points\_in\_qp\_table\_minus1**, may be replaced by **num\_points\_in\_qp\_table** that is used to specify the number of points in a chroma QP table and the value is a non-negative integer.
    - i. In one example, the value of the syntax element that denotes the number of pivot points shall be in the range of 0 to  $(63+QpBdOffset)$ .
  - b. Alternatively, furthermore, when number of pivot points in a chroma QP table is zero, the i-th entry of a chroma QP table is set equal to the i-th entry of a luma QP table.
  - c. Alternatively, furthermore, when number of pivot points in a chroma QP table is zero, the i-th entry of a chroma QP table is set equal to (the i-th entry of a luma QP table plus an offset).
    - i. In one example, the offset may be dependent on the coded method (e.g., JCCR on or off)
- 14. Parsing of the chroma QP table start point syntax element, i.e., currently denoted as **qp\_table\_start\_minus26** in JVET-Q2001-vC, may be conditional on if the number of pivot points is 0 or not.
  - a. In one example, when the number of pivot points is 0, parsing of the syntax element may be skipped.
- 15. In the derivation process of the chroma QP table, the XOR operator should be performed between  $(\text{delta\_qp\_in\_val\_minus1}[i][j] + 1)$  and  $\text{delta\_qp\_diff\_val}[i][j]$ .
  - a. An example is shown in embodiment 3.
- 16. A QP clipping may be always applied to the chroma QP table index.

- a. In one example, the clipping range is from  $-QpBdOffset$  to 63, inclusive.
17. A QP clipping may be always applied to the mapped chroma QPs in a chroma QP table.
- a. In one example, the clipping range is from  $-QpBdOffset$  to 63, inclusive.
18. Chroma QP tables may be signalled in both SPS and PPS.
- a. In one example, chroma QP table in PPS may overwrite the corresponding table in SPS.
19. Chroma QP tables may be signalled in PH or SH.
- a. In one example, chroma QP table in PH or SH may overwrite the corresponding table in SPS or PPS.
20. The default chroma QP tables may depend on ChromaArrayType.
- a. In one example, when ChromaArrayType is equal to 3 (i.e. 4:4:4 colour format and `separate_colour_plane_flag` is equal to 0), identity QP chroma QP tables (i.e. luma QP = chroma QP) may be set as default.
  - b. In one example, when ChromaArrayType is equal to 2 (i.e. 4:2:2 colour format), identity QP chroma QP tables (i.e. luma QP = chroma QP) may be set as default.
21. Regarding the control of RRC and TSRC for solving the eighth problem, one or more of the following approaches are disclosed, e.g., as in the seventh set of embodiments:
- a. In one example, whether to allow TSRC for a video unit (CLVS/group of pictures/picture/slice/tile/CTU row/CTU/CU/PU/TU) may be dependent on syntax elements (e.g., one or more flags) signalled in the SPS/PPS/PH/SH/block/CU/TU-level.
    - i. For example, the TSRC enabling/disabling flag may be signalled either in the PH or the SH, but not both,
      - 1) Additionally, whether to signalled the TSRC enabling/disabling flag in the PH or the SH may be dependent on a syntax flag signalled in the PPS/SPS.
      - 2) Additionally, when the TSRC enabling/disabling flag is signalled in the PH, then the TSRC enabling/disabling flag in the SH is not signalled.

- 3) Additionally, when the TSRC enabling/disabling flag in the SH is not present, it is inferred to be equal to the TSRC enabling/disabling flag in the PH.
- ii. Alternatively, the TSRC enabling/disabling flags may be signalled in both the PH and SH,
    - 1) Additionally, furthermore, the TSRC enabling/disabling flags in the SH may be not signalled when the TSRC enabling/disabling flags in the PH specifying that the TSRC is disabled for all slices referring to the PH.
  - iii. For example, the block/CU/TU-level TSRC enabling/disabling flag is coded with context-adaptive arithmetic entropy-coded syntax element, such as  $ae(v)$  coding.
  - iv. For example, the SPS/PPS/PH/SH-level TSRC enabling/disabling flag is coded with unsigned integer using  $n$  bits (e.g.,  $u(1)$  coding), such as  $n=1$ .
- b. Additionally, the signalling of the TSRC enabling/disabling flag for a video unit may be dependent on whether the transform skip is enabled/used for a video unit.
    - i. For example, if the transform skip is disabled at the SPS-level (e.g.,  $sps\_transform\_skip\_enabled\_flag$  is equal to 0), then the TSRC enabling/disabling flag in the PPS-level is required be equal to a certain value that specifying the TSRC is disabled for pictures referring to the current PPS.
    - ii. Additionally, if the transform skip is disabled at a higher-level (e.g.,  $sps\_transform\_skip\_enabled\_flag$  is equal to 0), then the TSRC enabling/disabling flags at the lower-level and/or same-layer (e.g., SPS/PH/SH/block/CU/TU-level) are not signalled.
    - iii. Alternatively, furthermore, if the transform skip is enabled at a higher-level (e.g., SPS) but not used for a video unit (e.g., the TU-level

transform\_skip\_flag is equal to 0), then the TSRC enabling/disabling flag of the current video unit (e.g., TU) is not signalled.

- iv. Additionally, when the TSRC enabling/disabling flag is not signalled at a video unit level, the value of the TSRC enabling/disabling flag is inferred to be a certain value that specifying the TSRC is disabled for the video unit.
- c. Alternatively, furthermore, the signalling of the TSRC enabling/disabling flag at low-level may be dependent on the TSRC enabling/disabling flag at higher-level.
  - i. For example, the signalling of the TSRC enabling/disabling flag at picture/slice level may be dependent on whether the TSRC is enabled at SPS/PPS-level.
  - ii. Additionally, the signalling of the TSRC enabling/disabling flag at block/CU/TU level may be dependent on whether the TSRC is enabled at SPS/PPS/picture/slice-level.
    - 1) For example, when the TSRC is disabled at a higher-level (e.g., SPS/PPS/picture/slice-level), then the TSRC enabling/disabling flag at block/CU/TU level is not signalled.
    - 2) Additionally, furthermore, when the TSRC enabling/disabling flag is not present, it is inferred to a certain value (such as a value specifying the TSRC is disabled for the current video unit).
  - iii. Additionally, the value of the TSRC enabling/disabling flag at PPS-level may be dependent on whether the TSRC is enabled at SPS-level.
    - 1) For example, when the SPS-level TSRC enabling/disabling flag specifies that the TSRC is disabled for the CLVS, then the value of the PPS-level TSRC enabling/disabling flag is required to be equal to a certain value that specifying the TSRC is disabled for pictures referring to the current PPS.

Regarding ALF and CC-ALF

22. it is proposed that indication of usage of chroma ALF/CC-ALF for a current CTU are not signaled when the luma ALF is disabled for the current CTU.
- a. Alternatively, furthermore, the usage is inferred to be false when luma ALF is disabled for a CTU. In other words, chroma ALF/CC-ALF are disabled when luma ALF is disabled in a CTU

Regarding sub-bitstream extraction

23. It is proposed that the information carried in the array NumSubLayersInLayerInOLS[ ][ ] may be used in the sub-bitstream extraction process in VVC only for removing of VCL NAL units.

- a. In one example, the following step in the sub-bitstream extraction process specified in clause C.6 of VVC:
- Remove from outBitstream all NAL units for which all of the following conditions are true:
    - nal\_unit\_type is not equal to IDR\_W\_RADL, IDR\_N\_LP, or CRA\_NUT.
    - nuh\_layer\_id is equal to LayerIdInOls[ targetOlsIdx ][ j ] for a value of j in the range of 0 to NumLayersInOls[ targetOlsIdx ] – 1 inclusive.
    - TemporalId is greater than or equal to NumSubLayersInLayerInOLS[ targetOlsIdx ][ j ].

is changed to be the following:

- Remove from outBitstream all VCL NAL units for which all of the following conditions are true:
  - nal\_unit\_type is not equal to IDR\_W\_RADL, IDR\_N\_LP, or CRA\_NUT.
  - nuh\_layer\_id is equal to LayerIdInOls[ targetOlsIdx ][ j ] for a value of j in the range of 0 to NumLayersInOls[ targetOlsIdx ] – 1 inclusive.
  - TemporalId is greater than or equal to NumSubLayersInLayerInOLS[ targetOlsIdx ][ j ].

### Regarding the number of APSs

In the following discussion,

“The maximum allowed number of filters” may refer to “the maximum allowed number of filters that can be signaled by one or multiple APSs for a sequence/picture/slice.

“The maximum allowed number of APS NAL units” may refer to “the maximum allowed number of APS NAL units that can be signaled for a sequence/picture/slice.

“The maximum allowed number of filter coefficients” may refer to “the maximum allowed number of filter coefficients that can be signaled by one or multiple APSs for a sequence/picture/slice.

24. Regarding the number of APS NAL units to solve the twelfth problem:

- a. The kinds of APS NAL units below may be one or more of ALF APS, LMCS APS, and SCALING APS.
- b. In one example, APS containing information related to different color components may be signalled separately.
  - i. For example, an APS NAL unit may contain luma-related syntax elements (e.g., luma filters) only.
  - ii. Additionally, an APS NAL unit may contain chroma (such as Cb and/or Cr)-related syntax elements (e.g., chroma filters) only.
    - 1) For example, an APS NAL unit may contain chroma ALF syntax elements only.
    - 2) For example, an APS NAL unit may contain CCALF syntax elements only.
    - 3) Alternatively, an APS NAL unit may contain CCALF Cb syntax elements only.
    - 4) Alternatively, an APS NAL unit may contain CCALF Cr syntax elements only.
  - iii. For example, luma and chroma filters of ALF (and/or LMCS and/or Scaling list) may have different APS types (e.g., **aps\_params\_type**).
  - iv. For example, one or multiple SEs are signaled in an APS to indicate which color component(s) is contained in the APS.

- c. In one example, the maximum allowed number of APS NAL units that can be signaled for a sequence/picture/slice may be dependent on a predefined value (such as a pre-defined memory size or a predefined number of filters).
- d. In one example, it is required that number of filters signaled in APSs for a sequence/picture/slice shall be no greater than a maximum allowed number of filters signaled in APSs.
- e. In one example, it is required that number of filter coefficients signaled in APSs for a sequence/picture/slice shall be no greater than a maximum allowed number of filter coefficients signaled in APSs.
  - i. The maximum allowed number of filter coefficients signaled in APSs may be a predefined number.
  - ii. The maximum allowed number of filter coefficients signaled in APSs may be different for different APS types.
  - iii. The maximum allowed number of filter coefficients signaled in APSs may be signaled from the encoder to the decoder, such as in VPS/SPS/PPS/PH/SH/sequence header.
- f. In one example, it is required that number of filters signaled in APSs for a sequence/picture/slice shall be no greater than a maximum allowed number of filters signaled in APSs.
- g. In one example, the maximum allowed numbers of filters signaled in APSs for different color components may be different.
  - i. In one example, the maximum allowed number of luma filters signaled in APSs and the maximum number of chroma filters signaled in APSs may be different.
  - ii. For example, the maximum allowed number of luma filters signaled in APSs of a particular APS type (e.g., ALF/LMCS/SCALING) is equal to a first certain value.
  - iii. Additionally, the maximum allowed number of chroma (such as Cb and/or Cr) filters signaled in APSs of a particular APS type (e.g., ALF/LMCS/SCALING) is equal to a second certain value.

- iv. For example, the maximum allowed number of luma filters signaled in APSs of all APS types (e.g., ALF/LMCS/SCALING) is equal to a first certain value.
- v. Additionally, the maximum allowed number of chroma (such as Cb and/or Cr) filters signaled in APSs of all APS types (e.g., ALF/LMCS/SCALING) is equal to a second certain value.
- h. In one example, the maximum allowed number of ALF luma filters, the maximum allowed number of ALF chroma filters and the maximum allowed number of CCALF filters may be different.
  - i. For example, the maximum allowed number of ALF luma filters is equal to  $X1$  (e.g.,  $X1 = 25 \times 8 = 200$ ).
  - ii. For example, for an ALF APS unit, the maximum allowed number of ALF chroma filters is equal to  $X2$  (e.g.,  $X2 = 8 \times 8 = 64$ ).
  - iii. For example, for an ALF APS unit, the maximum allowed number of CCALF filters is equal to  $X3$  (e.g.,  $X3 = 4 \times 8 \times 2 = 64$ ).
    - 1) Alternatively, the maximum allowed number of CCALF Cb filters is  $Y1$  (e.g.,  $Y1 = 4 \times 8 = 32$ ).
    - 2) Additionally, the maximum allowed number of CCALF Cr filters is  $Y2$  (e.g.,  $Y2 = 4 \times 8 = 32$ ).
  - iv. In one example,  $X1$ ,  $X2$ ,  $X3$ ,  $Y1$ ,  $Y2$  are pre-defined values.
- i. In one example, each of the maximum allowed numbers of filters within an APS NAL unit is equal to  $K_i$ , wherein  $K$  indicates a certain value, and  $i$  indicates the filter type (e.g., luma filter, chroma filter, ccalf filter, and etc).
  - i. For example,  $K_i$  for ALF luma filters is equal to  $25 \times 8 = 200$ .
  - ii. For example,  $K_i$  for ALF chroma filters is equal to  $8 \times 8 = 64$ .
  - iii. For example,  $K_i$  for CCALF filters is equal to  $4 \times 8 \times 2 = 64$ .
    - 1) Alternatively,  $K_i$  for CCALF Cb filters is equal to  $4 \times 8 = 32$ .
    - 2) Alternatively,  $K_i$  for CCALF Cr filters is equal to  $4 \times 8 = 32$ .
- j. In one example, the maximum allowed numbers may be signalled from the encoder to the decoder, such as in the VPS/SPS/PPS/PH/SH/sequence header.

- i. For example, the “maximum allowed number” in this claim may refer to the maximum allowed number of APS NAL units, or the maximum allowed number of filters, or the maximum allowed number of filter coefficients.
- ii. For example, the maximum allowed numbers for a particular type of APS NAL unit (e.g., `aps_params_type` equal to ALF APS) may be signalled.
  - 1) For example, the maximum allowed numbers may be signalled at SPS level.
  - 2) For example, the maximum allowed numbers may be signalled at PH/SH level.
- iii. For example, the signalling of the maximum allowed numbers may be dependent on whether a coding tool (ALF/CCALF/LMCS/scaling list) is enabled or not.
  - 1) In one example, when the ALF/CCALF is disabled, the maximum allowed numbers for ALF/CCALF APS may not be signalled.
    - a. Additionally, the number is inferred to be equal to a certain value (e.g. 0) when not present.
  - 2) In one example, when the CCALF is disabled, the maximum allowed number for ALF APS having the APS ID (e.g., `adaptation_parameter_set_id`) equal to CCALF APS ID (e.g., `ph_cc_alf_cb_aps_id`, `ph_cc_alf_cr_aps_id`, `slice_cc_alf_cb_aps_id`, `slice_cc_alf_cr_aps_id`) may not be signalled.
    - a. Additionally, the number is inferred to be equal to a certain value (e.g. 0) when not present.
  - 3) How to signal the maximum allowed number for a smaller video unit (e.g., picture, slice) may depend on the maximum allowed number for a higher level (e.g., SPS).
    - a. For example, the maximum allowed number of ALF luma APSs is dependent on the maximum allowed number of ALF APSs.

- iv. For example, a variable derived from the maximum allowed numbers (denoted as N) (e.g.  $N - M$  or  $M - N$ , wherein M is a certain value smaller than N) may be signalled.
- k. In one example, the maximum allowed numbers depend on chroma format.
  - i. For example, the maximum allowed number for luma/chroma may depend on whether the ChromaArrayType is equal to 0 (e.g., 4:0:0 and 4:0:0 with separate color plane coding) or not.
  - ii. For example, the maximum allowed number of ALF/SCALING/LMCS APSs depend on whether the ChromaArrayType is equal to 0 or not.
- l. In one example, the maximum allowed numbers above may depend on whether a coding tool is enabled.
  - i. For example, the maximum allowed numbers above depend on whether the ALF is enabled, and/or whether CCALF is enabled at SPS/PPS/PH/SH level.
  - ii. For example, the maximum allowed number of ALF APSs depend on whether the ALF is enabled, and/or whether CCALF is enabled at SPS/PPS/PH/SH level.
  - iii. For example, the maximum allowed number of LMCS APSs depend on whether the LMCS is enabled, and/or whether chroma residual scaling (CRS) is enabled at SPS/PPS/PH/SH level.
  - iv. For example, the maximum allowed number of SCALING APSs depend on whether the explicit scaling list is enabled at SPS/PPS/PH/SH level.
- m. In one example, how to code an APS ID may be dependent on a restricted number.
  - i. In one example, the APS ID may be  $ue(v)$  coded, in the range of 0 to K (e.g., 127, or 255), inclusive.
  - ii. Alternatively, the APS ID may be  $u(X)$  coded, such as  $X=7$  or 8.
    - 1) X may be signaled before APS ID is coded.
    - 2) X may be derived before APS ID is coded.
    - 3) X may be derived based on a maximum allowed number.
  - iii. Alternatively, the max number of APS ID is signalled.
    - 1) For example, the max number of APS ID may be  $u(X)$  coded, such as  $X=7$  or 8.

- 2) Additionally, the APS ID may be u(v) coded.
- iv. For example, The APS ID mentioned above may be one or more of follows:
- 1) adaptation\_parameter\_set\_id in the APS syntax structure.
  - 2) ph\_alf\_aps\_id\_luma[ i ], ph\_alf\_aps\_id\_chroma, ph\_cc\_alf\_cb\_aps\_id, ph\_cc\_alf\_cr\_aps\_id in the PH syntax structure.
  - 3) slice\_alf\_aps\_id\_luma[ i ], slice\_alf\_aps\_id\_chroma, slice\_cc\_alf\_cb\_aps\_id, slice\_cc\_alf\_cr\_aps\_id in the SH syntax structure.
  - 4) ph\_lmcs\_aps\_id in the PH syntax structure.
  - 5) ph\_scaling\_list\_aps\_id in the PH syntax structure.
- n. In one example, the constraints above may be represented by bitstream conformance, bitstream constraints, or explicitly signalling in the syntax table.
- o. In one example, the limitations/restrictions/constraints above may depend on profile/level/general constraint flags and etc.
25. It is proposed that the signaling and/or the range and/or the inference of a syntax element specifying the number of points in QP table is dependent on other syntax elements.
- a. It is proposed to set the maximum value of num\_points\_in\_qp\_table\_minus1[ i ] to (max QP value - the starting luma and chroma QP used to describe the i-th chroma QP mapping table).
    - i. In one example, the maximum value is set to  $(63 - (\text{qp\_table\_start\_minus26}[ i ] + 26) )$ .
    - ii. In one example, the maximum value is set to  $(64 - (\text{qp\_table\_start\_minus26}[ i ] + 26) )$ .
26. It is proposed to disallow the maximum number of subblock merge candidate to be 0 (MaxNumSubblockMergeCand) when affine is enabled and SbTMVP is disabled.
- a. In one example, the maximum number of subblock merge candidate is restricted to greater than 0 when affine is enabled and SbTMVP is disabled.

- b. In one example, the range of `sps_five_minus_max_num_subblock_merge_cand` is from [0, 4] regardless the value of SPS SbTMVP enabling flag (e.g., `sps_sbtmvp_enabled_flag`).
  - c. A conformance bitstream shall satisfy that the value of `MaxNumSubblockMergeCand` shall be in the range of 1 to 5, inclusive, when affine is enabled.
    - i. Alternatively, furthermore, the value of `MaxNumSubblockMergeCand` shall be in the range of 0 to 1, inclusive, when affine is disabled.
  - d. Alternatively, furthermore, whether to signal **`merge_subblock_flag`** is dependent on whether affine is enabled, instead of checking the maximum number of subblock merge candidate.
27. It is proposed to add a first syntax element to indicate whether affine merge mode is allowed.
- a. Alternatively, furthermore, the first syntax element may be conditionally signaled, e.g., according to SPS affine flag is enabled.
  - b. Alternatively, furthermore, indication of the maximum number of subblock merge candidate may be signaled when the first syntax element indicates affine merge mode is enabled.
28. To solve problem 13, the subpictures layout is signalled as follows:
- a. A flag, e.g., named `sps_subpic_same_res_flag`, is added to specify whether all subpictures referring to the SPS have the same width and height, respectively.
  - b. When `sps_subpic_same_res_flag` is equal to 1, the common width and height for all subpictures are signalled, e.g., by the two new syntax elements `sps_uniform_subpic_width_minus1` and `sps_uniform_subpic_height_minus1`, respectively, in units of CTUs.
    - i. In one example, furthermore, the number of subpicture columns, denoted as `numSubpicCols`, is derived to be equal to  $(\text{sps\_pic\_width\_max\_in\_luma\_samples} / ((\text{sps\_uniform\_subpic\_width\_minus1} + 1) * \text{CtbSizeY}))$ , the number of subpicture row, denoted as `numSubpicRows`, is derived to be equal to  $(\text{sps\_pic\_height\_max\_in\_luma\_samples} /$

- (( (sps\_uniform\_subpic\_height\_minus1 + 1) \* CtbSizeY)), and the number of subpictures, denoted as sumSubpics, is derived to be equal to numSubpicCols \* numSubpicRows.
- ii. In one example, furthermore, sps\_num\_subpics\_minus1 is skipped, and the value is inferred to equal to numSubpics - 1.
  - iii. In one example, for each value of i in the range of 0 to sps\_num\_subpics\_minus1, inclusive, the value of sps\_subpic\_ctu\_top\_left\_x[ i ], sps\_subpic\_ctu\_top\_left\_y[ i ], sps\_subpic\_width\_minus1[ i ], and sps\_subpic\_height\_minus1[ i ] are inferred to be equal to
 
$$(sps\_uniform\_subpic\_width\_minus1 + 1) * (i \% numSubpicCols),$$

$$(sps\_uniform\_subpic\_height\_minus1 + 1) * (i / numSubpicCols),$$
 sps\_uniform\_subpic\_width\_minus1, and sps\_uniform\_subpic\_height\_minus1, respectively.
  - iv. In one example, for each value of i in the range of 0 to sps\_num\_subpics\_minus1, inclusive, when sps\_independent\_subpics\_flag is equal to 0, sps\_subpic\_treated\_as\_pic\_flag[ i ] and sps\_loop\_filter\_across\_subpic\_enabled\_flag[ i ] are signalled.
- c. When sps\_subpic\_same\_res\_flag is equal to 0, the current subpictures layout signalling in JVET-R2001-vA/v10 applies.
  - d. In one example, when sps\_pic\_width\_max\_in\_luma\_samples is less than or equal to CtbSizeY, sps\_uniform\_subpic\_width\_minus1 is skipped and inferred to be equal to 0.
  - e. In one example, when sps\_pic\_height\_max\_in\_luma\_samples is less than or equal to CtbSizeY, sps\_uniform\_subpic\_height\_minus1 is skipped and inferred to be equal to 0.
  - f. In one example, sps\_uniform\_subpic\_width\_minus1 and sps\_uniform\_subpic\_height\_minus1 are ue(v)-coded, and sps\_subpic\_ctu\_top\_left\_x[ i ], sps\_subpic\_ctu\_top\_left\_y[ i ], sps\_subpic\_width\_minus1[ i ], and sps\_subpic\_height\_minus1[ i ] are also ue(v)-coded.

- g. In one example, `sps_uniform_subpic_width_minus1` and `sps_uniform_subpic_height_minus1` are u(v)-coded, with the same length as `sps_subpic_width_minus1[ i ]` and `sps_subpic_height_minus1[ i ]`, respectively, in JVET-R2001-vA/v10.
- h. In one example, the SPS syntax is changed as follows, where ***bold italicized underlined*** text indicate newly added text and where open and close double brackets (e.g., `[[ ]]`) indicate deleted text in between the double brackets:

<code>seq_parameter_set_rbsp( ) {</code>	Descript or
<code>...</code>	
<b><i><u>sps_subpic_same_res_flag</u></i></b>	<b><i><u>ue(v)</u></i></b>
<b><i><u>if( sps_subpic_same_res_flag ) {</u></i></b>	
<b><i><u>if( sps_pic_width_max_in_luma_samples &gt; CtbSizeY )</u></i></b>	
<b><i><u>sps_uniform_subpic_width_minus1</u></i></b>	<b><i><u>u(v)</u></i></b>
<b><i><u>if( sps_pic_height_max_in_luma_samples &gt; CtbSizeY )</u></i></b>	
<b><i><u>sps_uniform_subpic_height_minus1</u></i></b>	<b><i><u>u(v)</u></i></b>
<b><i><u>} else</u></i></b>	
<code>sps_num_subpics_minus1</code>	ue(v)
<code>if( sps_num_subpics_minus1 &gt; 0 )</code>	
<code>sps_independent_subpics_flag</code>	u(1)
<code>for( i = 0; <b><i><u>sps_subpic_same_res_flag &amp;&amp;</u></i></b> sps_num_subpics_minus1 &gt; 0 &amp;&amp; i &lt;= sps_num_subpics_minus1; i++ ) {</code>	
<code>if( i &gt; 0 &amp;&amp; sps_pic_width_max_in_luma_samples &gt; CtbSizeY )</code>	
<code>sps_subpic_ctu_top_left_x[ i ]</code>	u(v)
<code>if( i &gt; 0 &amp;&amp; sps_pic_height_max_in_luma_samples &gt; CtbSizeY ) {</code>	

<b>sps_subpic_ctu_top_left_y[ i ]</b>	u(v)
if( i < sps_num_subpics_minus1 && sps_pic_width_max_in_luma_samples > CtbSizeY )	
<b>sps_subpic_width_minus1[ i ]</b>	u(v)
if( i < sps_num_subpics_minus1 && sps_pic_height_max_in_luma_samples > CtbSizeY )	
<b>sps_subpic_height_minus1[ i ]</b>	u(v)
<b>}</b>	
<b><u>for( i = 0; sps_num_subpics_minus1 &gt; 0 &amp;&amp; i &lt;=</u></b> <b><u>sps_num_subpics_minus1; i++ )</u></b>	
if( !sps_independent_subpics_flag ) {	
<b>sps_subpic_treated_as_pic_flag[ i ]</b>	u(1)
<b>sps_loop_filter_across_subpic_enabled_flag[ i ]</b>	u(1)
<b>}</b>	
<b>[[ ]]</b>	
<b>...</b>	

i. In one example, the SPS syntax is changed as follows:

seq_parameter_set_rbsp( ) {	Descript or
...	
<b><u>sps_subpic_same_res_flag</u></b>	<b><u>ue(v)</u></b>
<b><u>if( sps_subpic_same_res_flag ) {</u></b>	
<b><u>sps_uniform_subpic_width_minus1</u></b>	<b><u>ue(v)</u></b>
<b><u>sps_uniform_subpic_height_minus1</u></b>	<b><u>ue(v)</u></b>

<i><u>} else</u></i>	
<b>sps_num_subpics_minus1</b>	ue(v)
if( sps_num_subpics_minus1 > 0 )	
<b>sps_independent_subpics_flag</b>	u(1)
for( i = 0; <i><u>sps_subpic_same_res_flag &amp;&amp;</u></i> sps_num_subpics_minus1 > 0 && i <= sps_num_subpics_minus1; i++ ) {	
if( i > 0 [[ && sps_pic_width_max_in_luma_samples > CtbSizeY]] )	
<b>sps_subpic_ctu_top_left_x[ i ]</b>	ue(v)
if( i > 0 [[&& sps_pic_height_max_in_luma_samples > CtbSizeY]] ) {	
<b>sps_subpic_ctu_top_left_y[ i ]</b>	ue(v)
if( i < sps_num_subpics_minus1 [[&& sps_pic_width_max_in_luma_samples > CtbSizeY ]])	
<b>sps_subpic_width_minus1[ i ]</b>	ue(v)
if( i < sps_num_subpics_minus1 [[ && sps_pic_height_max_in_luma_samples > CtbSizeY ]])	
<b>sps_subpic_height_minus1[ i ]</b>	ue(v)
<i><u>}</u></i>	
<i><u>for( i = 0; sps_num_subpics_minus1 &gt; 0 &amp;&amp; i &lt;=</u></i> <i><u>sps_num_subpics_minus1; i++ )</u></i>	
if( !sps_independent_subpics_flag ) {	
<b>sps_subpic_treated_as_pic_flag[ i ]</b>	u(1)
<b>sps_loop_filter_across_subpic_enabled_flag[ i ]</b>	u(1)
}	
[[ ]]	
...	

29. The following changes to the latest VVC spec text in JVET-R2001-vA/v10 are proposed:

**SPS semantics:**

...

**sps\_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. **sps\_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 **sps\_subpic\_treated\_as\_pic\_flag**[ *i* ] is inferred to be equal to 1.

[[When **sps\_num\_subpics\_minus1** is greater than 0 and **sps\_subpic\_treated\_as\_pic\_flag**[ *i* ] is equal to 1, for each CLVS of a current layer referring to the SPS, let **targetAuSet** be all the AUs starting from the AU containing the first picture of the CLVS in decoding order, to the AU containing the last picture of the CLVS in decoding order, inclusive, it is a requirement of bitstream conformance that all of the following conditions are true for the **targetLayerSet** that consists of the current layer and all the layers that have the current layer as a reference layer:]]

[[– For each AU in **targetAuSet**, all pictures of the layers in **targetLayerSet** shall have the same value of **pps\_pic\_width\_in\_luma\_samples** and the same value of **pps\_pic\_height\_in\_luma\_samples**.]]

[[– All the SPSs referred to by the layers in **targetLayerSet** shall have the same value of **sps\_num\_subpics\_minus1** and shall have the same values of **sps\_subpic\_ctu\_top\_left\_x**[ *j* ], **sps\_subpic\_ctu\_top\_left\_y**[ *j* ], **sps\_subpic\_width\_minus1**[ *j* ], **sps\_subpic\_height\_minus1**[ *j* ], and **sps\_subpic\_treated\_as\_pic\_flag**[ *j* ], respectively, for each value of *j* in the range of 0 to **sps\_num\_subpics\_minus1**, inclusive.]]

[[– For each AU in **targetAuSet**, all pictures of the layers in **targetLayerSet** shall have the same value of **SubpicIdVal**[ *j* ] for each value of *j* in the range of 0 to **sps\_num\_subpics\_minus1**, inclusive.]]

...

**Decoding process for reference picture lists construction**

...

- [[When `vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]]` is equal to 0 and `sps_num_subpics_minus1` is greater than 0, either of the following two conditions (but not both) shall be true:]] *When `sps_num_subpics_minus1` is greater than 0 and the current subpicture with subpicture index `subPicIdx` has `sps_subpic_treated_as_pic_flag[subPicIdx]` equal to 1, either of the following two conditions (but not both) shall be true:*
- [[The picture referred to by each active entry in `RefPicList[0]` or `RefPicList[1]` has the same subpicture layout as the current picture (i.e., the SPSs referred to by that picture and the current picture have the same value of `sps_num_subpics_minus1` and the same values of `sps_subpic_ctu_top_left_x[j]`, `sps_subpic_ctu_top_left_y[j]`, `sps_subpic_width_minus1[j]`, and `sps_subpic_height_minus1[j]`, respectively, for each value of `j` in the range of 0 to `sps_num_subpics_minus1`, inclusive).]]
  - *The picture referred to by each active entry in `RefPicList[0]` or `RefPicList[1]` and the current picture have the same value for each of the following:*
    - *`pps_pic_width_in_luma_samples`*
    - *`pps_pic_height_in_luma_samples`*
    - *`sps_num_subpics_minus1`*
    - *`sps_subpic_ctu_top_left_x[i]`, `sps_subpic_ctu_top_left_y[i]`, `sps_subpic_width_minus1[i]`, `sps_subpic_height_minus1[i]`, `sps_subpic_treated_as_pic_flag[i]`, respectively, for each value of `i` in the range of 0 to `sps_num_subpics_minus1`, inclusive*
  - The picture referred to by each active entry in `RefPicList[0]` or `RefPicList[1]` is an ILRP for which the value of `sps_num_subpics_minus1` is equal to 0.

30. The following changes to the previous item are proposed:

**PPS semantics:**

...

**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 variable **SubpicIdVal[ i ]**, for each value of 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( sps_subpic_id_mapping_explicitly_signalled_flag )
        SubpicIdVal[ i ] = pps_subpic_id_mapping_present_flag ? pps_subpic_id[ i ] :
sps_subpic_id[ i ]
    else
        SubpicIdVal[ i ] = i

```

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

- For any two different values of i and j in the range of 0 to **sps\_num\_subpics\_minus1**, inclusive, **SubpicIdVal[ i ]** shall not be equal to **SubpicIdVal[ j ]**.
- [[For each value of i in the range of 0 to **sps\_num\_subpics\_minus1**, inclusive, when the value of **SubpicIdVal[ i ]** of a current picture, with **nuh\_layer\_id** equal to a particular value **layerId**, is not equal to the value of **SubpicIdVal[ i ]** of a reference picture that has **nuh\_layer\_id** equal to **layerId**, the active entries of the RPLs of the coded slices in the i-th subpicture of the current picture shall not include that reference picture.]]

...

### **Decoding process for reference picture lists construction**

...

- When **sps\_num\_subpics\_minus1** is greater than 0 and the current subpicture with subpicture index **subPicIdx** has **sps\_subpic\_treated\_as\_pic\_flag[ subPicIdx ]** equal to 1, either of the following two conditions (but not both) shall be true:
  - The picture referred to by each active entry in **RefPicList[ 0 ]** or **RefPicList[ 1 ]** and the current picture have the same value for each of the following:

- pps\_pic\_width\_in\_luma\_samples
  - pps\_pic\_height\_in\_luma\_samples
  - sps\_num\_subpics\_minus1
  - sps\_subpic\_ctu\_top\_left\_x[ i ], sps\_subpic\_ctu\_top\_left\_y[ i ], sps\_subpic\_width\_minus1[ i ], sps\_subpic\_height\_minus1[ i ], sps\_subpic\_treated\_as\_pic\_flag[ i ], *SubpicIdVal[ i ]*, respectively, for each value of i in the range of 0 to sps\_num\_subpics\_minus1, inclusive
    - The picture referred to by each active entry in RefPicList[ 0 ] or RefPicList[ 1 ] is an ILRP for which the value of sps\_num\_subpics\_minus1 is equal to 0.
31. When pps\_pic\_width\_in\_luma\_samples is less than or equal to CtbSizeY, pps\_tile\_column\_width\_minus1[ i ] is required to be equal to PicWidthInCtbsY – 1.
32. When pps\_pic\_height\_in\_luma\_samples is less than or equal to CtbSizeY, pps\_tile\_row\_height\_minus1[ i ] is required to be equal to PicHeight – 1.

**6. Embodiments**

**6.1. Embodiment 1: chroma QP tables according to slice type**

The changes, marked in **bold italicized underlined** text, are based on JVET-Q2001-vC. And the deleted texts are marked in open and close double brackets (e.g., [[ ]]) indicating deleted text in between the double brackets.

**7.3.2.3 Sequence parameter set RBSP syntax**

seq_parameter_set_rbsp( ) {	<b>Descript or</b>
.....	
<u><b>sps_non_intra_present_flag</b></u>	<u><b>u(1)</b></u>
if( ChromaArrayType != 0 ) {	
<b>sps_joint_cbr_enabled_flag</b>	u(1)

<b><i>intra</i></b> _same_qp_table_for_chroma	u(1)
numQpTables <b><i>Intra</i></b> = <b><i>intra</i></b> _same_qp_table_for_chroma ? 1 : (sps_joint_cbr_enabled_flag ? 3 : 2)	
for( i = 0; i < numQpTables <b><i>Intra</i></b> ; i++ ) {	
<b><i>intra</i></b> _qp_table_start_minus26[ i ]	se(v)
<b><i>intra</i></b> _num_points_in_qp_table_minus1[ i ]	ue(v)
for( j = 0; j <= <b><i>intra</i></b> _num_points_in_qp_table_minus1[ i ]; j++ ) {	
<b><i>intra</i></b> _delta_qp_in_val_minus1[ i ][ j ]	ue(v)
<b><i>intra</i></b> _delta_qp_diff_val[ i ][ j ]	ue(v)
}	
}	
if( sps_non_intra_present_flag ) {	
<b><i>inter</i></b> _same_qp_table_for_chroma	u(1)
numQpTables <b><i>Inter</i></b> = same_qp_table_for_chroma_inter ? 1 : (sps_joint_cbr_enabled_flag ? 3 : 2)	
for( i = 0; i < numQpTables <b><i>Inter</i></b> ; i++ ) {	
<b><i>inter</i></b> _qp_table_start_minus26[ i ]	se(v)
<b><i>inter</i></b> _num_points_in_qp_table_minus1[ i ]	ue(v)
for( j = 0; j <= <b><i>inter</i></b> _num_points_in_qp_table_minus1[ i ]; j++ ) {	
<b><i>inter</i></b> _delta_qp_in_val_minus1[ i ][ j ]	ue(v)
<b><i>inter</i></b> _delta_qp_diff_val[ i ][ j ]	ue(v)
}	
}	
}	
}	

### 7.3.2.3 Sequence parameter set RBSP syntax

.....

**[[same\_qp\_table\_for\_chroma** equal to 1 specifies that only one chroma QP mapping table is signalled and this table applies to Cb and Cr residuals and additionally to joint Cb-Cr residuals when `sps_joint_cbr_enabled_flag` is equal to 1. `same_qp_table_for_chroma` equal to 0 specifies that chroma QP mapping tables, two for Cb and Cr, and one additional for joint Cb-Cr when `sps_joint_cbr_enabled_flag` is equal to 1, are signalled in the SPS. When `same_qp_table_for_chroma` is not present in the bitstream, the value of `same_qp_table_for_chroma` is inferred to be equal to 1.]]

**[[qp\_table\_start\_minus26[ i ]** plus 26 specifies the starting luma and chroma QP used to describe the i-th chroma QP mapping table. The value of `qp_table_start_minus26[ i ]` shall be in the range of  $-26 - QpBdOffset$  to 36 inclusive. When `qp_table_start_minus26[ i ]` is not present in the bitstream, the value of `qp_table_start_minus26[ i ]` is inferred to be equal to 0.]]

**[[num\_points\_in\_qp\_table\_minus1[ i ]** plus 1 specifies the number of points used to describe the i-th chroma QP mapping table. The value of `num_points_in_qp_table_minus1[ i ]` shall be in the range of 0 to  $63 + QpBdOffset$ , inclusive. When `num_points_in_qp_table_minus1[ 0 ]` is not present in the bitstream, the value of `num_points_in_qp_table_minus1[ 0 ]` is inferred to be equal to 0.]]

**[[delta\_qp\_in\_val\_minus1[ i ][ j ]** specifies a delta value used to derive the input coordinate of the j-th pivot point of the i-th chroma QP mapping table. When `delta_qp_in_val_minus1[ 0 ][ j ]` is not present in the bitstream, the value of `delta_qp_in_val_minus1[ 0 ][ j ]` is inferred to be equal to 0.]]

**[[delta\_qp\_diff\_val[ i ][ j ]** specifies a delta value used to derive the output coordinate of the j-th pivot point of the i-th chroma QP mapping table. ]]

[[The i-th chroma QP mapping table `ChromaQpTable[ i ]` for  $i = 0..numQpTables - 1$  is derived as follows:]]

***sps\_non\_intra\_present\_flag** equal to 0 specifies that all slices are of intra slice in the sequence.  
sps\_non\_intra\_present\_flag equal to 1 specific that there may be non intra slices present in the sequence.*

*intra\_same\_qp\_table\_for\_chroma* equal to 1 specifies that only one chroma QP mapping table is signalled and this table applies to Cb and Cr residuals and additionally to joint Cb-Cr residuals when *sps\_joint\_cbr\_enabled\_flag* is equal to 1 for intra slices. *intra\_same\_qp\_table\_for\_chroma* equal to 0 specifies that chroma QP mapping tables, two for Cb and Cr, and one additional for joint Cb-Cr when *sps\_joint\_cbr\_enabled\_flag* is equal to 1, for intra slices are signalled in the SPS. When *same\_qp\_table\_for\_chroma* is not present in the bitstream, the value of *intra\_same\_qp\_table\_for\_chroma* is inferred to be equal to 1.

*intra\_qp\_table\_start\_minus26[i]* plus 26 specifies the starting luma and chroma QP used to describe the *i*-th chroma QP mapping table for intra slices. The value of *intra\_qp\_table\_start\_minus26[i]* shall be in the range of  $-26 - QpBdOffset$  to 36 inclusive. When *intra\_qp\_table\_start\_minus26[i]* is not present in the bitstream, the value of *intra\_qp\_table\_start\_minus26[i]* is inferred to be equal to 0.

*intra\_num\_points\_in\_qp\_table\_minus1[i]* plus 1 specifies the number of points used to describe the *i*-th chroma QP mapping table for intra slices. The value of *intra\_num\_points\_in\_qp\_table\_minus1[i]* shall be in the range of 0 to  $63 + QpBdOffset$ , inclusive. When *intra\_num\_points\_in\_qp\_table\_minus1[0]* is not present in the bitstream, the value of *intra\_num\_points\_in\_qp\_table\_minus1[0]* is inferred to be equal to 0.

*intra\_delta\_qp\_in\_val\_minus1[i][j]* specifies a delta value used to derive the input coordinate of the *j*-th pivot point of the *i*-th chroma QP mapping table for intra slices. When *intra\_delta\_qp\_in\_val\_minus1[0][j]* is not present in the bitstream, the value of *intra\_delta\_qp\_in\_val\_minus1[0][j]* is inferred to be equal to 0.

*intra\_delta\_qp\_diff\_val[i][j]* specifies a delta value used to derive the output coordinate of the *j*-th pivot point of the *i*-th chroma QP mapping table for intra slices.

*inter\_same\_qp\_table\_for\_chroma* equal to 1 specifies that only one chroma QP mapping table is signalled and this table applies to Cb and Cr residuals and additionally to joint Cb-Cr residuals when *sps\_joint\_cbr\_enabled\_flag* is equal to 1 for B or P slices. *inter\_same\_qp\_table\_for\_chroma* equal to 0 specifies that chroma QP mapping tables, two for Cb and Cr, and one additional for joint Cb-Cr when *sps\_joint\_cbr\_enabled\_flag* is equal to 1,

for B or P slices are signalled in the SPS. When *same\_qp\_table\_for\_chroma* is not present in the bitstream, the value of *inter\_same\_qp\_table\_for\_chroma* is inferred to be equal to 1.

*inter\_qp\_table\_start\_minus26[i]* plus 26 specifies the starting luma and chroma QP used to describe the *i*-th chroma QP mapping table for B or P slices. The value of *intra\_qp\_table\_start\_minus26[i]* shall be in the range of  $-26 - QpBdOffset$  to 36 inclusive. When *inter\_qp\_table\_start\_minus26[i]* is not present in the bitstream, the value of *inter\_qp\_table\_start\_minus26[i]* is inferred to be equal to 0.

*inter\_num\_points\_in\_qp\_table\_minus1[i]* plus 1 specifies the number of points used to describe the *i*-th chroma QP mapping table for B or P slices. The value of *inter\_num\_points\_in\_qp\_table\_minus1[i]* shall be in the range of 0 to  $63 + QpBdOffset$ , inclusive. When *inter\_num\_points\_in\_qp\_table\_minus1[0]* is not present in the bitstream, the value of *inter\_num\_points\_in\_qp\_table\_minus1[0]* is inferred to be equal to 0.

*inter\_delta\_qp\_in\_val\_minus1[i][j]* specifies a delta value used to derive the input coordinate of the *j*-th pivot point of the *i*-th chroma QP mapping table for B or P slices. When *inter\_delta\_qp\_in\_val\_minus1[0][j]* is not present in the bitstream, the value of *inter\_delta\_qp\_in\_val\_minus1[0][j]* is inferred to be equal to 0.

*inter\_delta\_qp\_diff\_val[i][j]* specifies a delta value used to derive the output coordinate of the *j*-th pivot point of the *i*-th chroma QP mapping table for B or P slices.

The *i*-th chroma QP mapping table *IntraChromaQpTable[i]* for  $i = 0..numQpTablesIntra - 1$  and the *i*-th chroma QP mapping table *InterChromaQpTable[i]* for  $i = 0..numQpTablesInter - 1$  are derived as follows:

When *intra\_same\_qp\_table\_for\_chroma* is equal to 1, *IntraChromaQpTable[1][k]* and *IntraChromaQpTable[2][k]* are set equal to *IntraChromaQpTable[0][k]* for *k* in the range of  $-QpBdOffset$  to 63, inclusive.

When *inter\_same\_qp\_table\_for\_chroma* is equal to 1, *InterChromaQpTable[1][k]* and *InterChromaQpTable[2][k]* are set equal to *InterChromaQpTable[0][k]* for *k* in the range of  $-QpBdOffset$  to 63, inclusive.

Let  $ChromaQpTable[i]$  denote the  $IntraChromaQpTable[i]$  for I-slices for  $i = 0..numQpTablesIntra - 1$  and  $InterChromaQpTable[i]$  for B- or P-slices for  $i = 0..numQpTablesInter - 1$ , the following applies

```

qpInVal[ i ][ 0 ] = qp_table_start_minus26[ i ] + 26
qpOutVal[ i ][ 0 ] = qpInVal[ i ][ 0 ]
for( j = 0; j <= num_points_in_qp_table_minus1[ i ]; j++ ) {
    qpInVal[ i ][ j + 1 ] = qpInVal[ i ][ j ] + delta_qp_in_val_minus1[ i ][ j ] + 1
    qpOutVal[ i ][ j + 1 ] = qpOutVal[ i ][ j ] +
( delta_qp_in_val_minus1[ i ][ j ] ^ delta_qp_diff_val[ i ][ j ] )
}
ChromaQpTable[ i ][ qpInVal[ i ][ 0 ] ] = qpOutVal[ i ][ 0 ]
for( k = qpInVal[ i ][ 0 ] - 1; k >= -QpBdOffset; k -- )
    ChromaQpTable[ i ][ k ] = Clip3( -QpBdOffset, 63, ChromaQpTable[ i ][ k + 1 ] - 1 )
(62)
for( j = 0; j <= num_points_in_qp_table_minus1[ i ]; j++ ) {
    sh = ( delta_qp_in_val_minus1[ i ][ j ] + 1 ) >> 1
    for( k = qpInVal[ i ][ j ] + 1, m = 1; k <= qpInVal[ i ][ j + 1 ]; k++, m++ )
        ChromaQpTable[ i ][ k ] = ChromaQpTable[ i ][ qpInVal[ i ][ j ] ] +
            ( ( qpOutVal[ i ][ j + 1 ] - qpOutVal[ i ][ j ] ) * m + sh ) /
( delta_qp_in_val_minus1[ i ][ j ] + 1 )
}
for( k = qpInVal[ i ][ num_points_in_qp_table_minus1[ i ] + 1 ] + 1; k <= 63; k++ )
    ChromaQpTable[ i ][ k ] = Clip3( -QpBdOffset, 63, ChromaQpTable[ i ][ k - 1 ] + 1 )

```

When  $same\_qp\_table\_for\_chroma$  is equal to 1,  $ChromaQpTable[1][k]$  and  $ChromaQpTable[2][k]$  are set equal to  $ChromaQpTable[0][k]$  for  $k$  in the range of  $-QpBdOffset$  to 63, inclusive.

It is a requirement of bitstream conformance that the values of  $qpInVal[i][j]$  and  $qpOutVal[i][j]$  shall be in the range of  $-QpBdOffset$  to 63, inclusive for  $i$  in the range of 0 to  $numQpTables - 1$ , inclusive, and  $j$  in the range of 0 to  $num\_points\_in\_qp\_table\_minus1[i] + 1$ , inclusive.

.....

**8.7.1 Derivation process for quantization parameters**

.....

*When the current slice is an I-slice, ChromaQpTable[ i ] is set equal to IntraChromaQpTable[ i ] for i = 0..2. When the current slices is a B or P slice, ChromaQpTable[ i ] is set equal to InterChromaQpTable[ i ] for i = 0..2.*

When ChromaArrayType is not equal to 0 and treeType is equal to SINGLE\_TREE or DUAL\_TREE\_CHROMA, the following applies:

- When treeType is equal to DUAL\_TREE\_CHROMA, the variable Qp<sub>Y</sub> is set equal to the luma quantization parameter Qp<sub>Y</sub> of the luma coding unit that covers the luma location ( xCb + cbWidth / 2, yCb + cbHeight / 2 ).
- The variables qP<sub>Cb</sub>, qP<sub>Cr</sub> and qP<sub>CbCr</sub> are derived as follows:

$$qP_{\text{Chroma}} = \text{Clip3}( -QpBdOffset, 63, Qp_Y ) \tag{1143}$$

$$qP_{\text{Cb}} = \text{ChromaQpTable}[ 0 ][ qP_{\text{Chroma}} ] \tag{1144}$$

$$qP_{\text{Cr}} = \text{ChromaQpTable}[ 1 ][ qP_{\text{Chroma}} ] \tag{1145}$$

$$qP_{\text{CbCr}} = \text{ChromaQpTable}[ 2 ][ qP_{\text{Chroma}} ] \tag{1146}$$

.....

**6.2. Embodiment 2: number of pivot points and clipping of mapped chroma QP**

The changes, marked in *bold faced italics*, are based on JVET-Q2001-vC.

**7.3.2.3 Sequence parameter set RBSP syntax**

seq_parameter_set_rbsp() {	<b>Descript or</b>
.....	
if( ChromaArrayType != 0 ) {	

<b>sps_joint_cbr_enabled_flag</b>	u(1)
<b>same_qp_table_for_chroma</b>	u(1)
numQpTables = same_qp_table_for_chroma ? 1 : ( sps_joint_cbr_enabled_flag ? 3 : 2 )	
for( i = 0; i < numQpTables; i++ ) {	
<b>[[qp_table_start_minus26[ i ]]]</b>	[[se(v)]]
<b>num_points_in_qp_table[[_minus1]][ i ]</b>	ue(v)
if( num_points_in_qp_table )	
<u><b>qp_table_start_minus26[ i ]</b></u>	<u><b>se(v)</b></u>
for( j = 0; j <[[=]] num_points_in_qp_table[[_minus1]][ i ]; j++ ) {	
<b>delta_qp_in_val_minus1[ i ][ j ]</b>	ue(v)
<b>delta_qp_diff_val[ i ][ j ]</b>	ue(v)
}	
}	
}	
}	
.....	

**num\_points\_in\_qp\_table[[\_minus1]][ i ]** [[plus 1]] specifies the number of points used to describe the i-th chroma QP mapping table. The value of num\_points\_in\_qp\_table[[\_minus1]][ i ] shall be in the range of 0 to 63 + QpBdOffset, inclusive. [[When num\_points\_in\_qp\_table\_minus1[ 0 ] is not present in the bitstream, the value of num\_points\_in\_qp\_table\_minus1[ 0 ] is inferred to be equal to 0.]]

**delta\_qp\_in\_val\_minus1[ i ][ j ]** specifies a delta value used to derive the input coordinate of the j-th pivot point of the i-th chroma QP mapping table. When delta\_qp\_in\_val\_minus1[ 0 ][ j ] is not present in the bitstream, the value of delta\_qp\_in\_val\_minus1[ 0 ][ j ] is inferred to be equal to 0.

**delta\_qp\_diff\_val[ i ][ j ]** specifies a delta value used to derive the output coordinate of the j-th pivot point of the i-th chroma QP mapping table.

The  $i$ -th chroma QP mapping table  $\text{ChromaQpTable}[i]$  for  $i = 0..numQpTables - 1$  is derived as follows:

When `same_qp_table_for_chroma` is equal to 1,  $\text{ChromaQpTable}[1][k]$  and  $\text{ChromaQpTable}[2][k]$  are set equal to  $\text{ChromaQpTable}[0][k]$  for  $k$  in the range of  $-QpBdOffset$  to 63, inclusive.

```

[[qpInVal[i][0] = qp_table_start_minus26[i] + 26
qpOutVal[i][0] = qpInVal[i][0]
for(j = 0; j <= num_points_in_qp_table_minus1[i]; j++) {
    qpInVal[i][j + 1] = qpInVal[i][j] + delta_qp_in_val_minus1[i][j] + 1
    qpOutVal[i][j + 1] = qpOutVal[i][j] +
(delta_qp_in_val_minus1[i][j] ^ delta_qp_diff_val[i][j])
}
ChromaQpTable[i][qpInVal[i][0]] = qpOutVal[i][0]
for(k = qpInVal[i][0] - 1 63; k >= -QpBdOffset; k--)
    ChromaQpTable[i][k] = Clip3(-QpBdOffset, 63, ChromaQpTable[i][k + 1] - 1)
for(j = 0; j <= num_points_in_qp_table_minus1[i]; j++) {
    sh = (delta_qp_in_val_minus1[i][j] + 1) >> 1
    for(k = qpInVal[i][j] + 1, m = 1; k <= qpInVal[i][j + 1]; k++, m++)
        ChromaQpTable[i][k] = ChromaQpTable[i][qpInVal[i][j]] +
((qpOutVal[i][j + 1] - qpOutVal[i][j]) * m + sh) /
(delta_qp_in_val_minus1[i][j] + 1)
}
for(k = qpInVal[i][num_points_in_qp_table_minus1[i] + 1] + 1; k <= 63; k++)
    ChromaQpTable[i][k] = Clip3(-QpBdOffset, 63, ChromaQpTable[i][k - 1] + 1)]]
for(k = 63; k >= -QpBdOffset; k--)
    ChromaQpTable[i][k] = Clip3(-QpBdOffset, 63, ChromaQpTable[i][k + 1] - 1)
if(num_points_in_qp_table[i] > 0) {
    qpInVal[i][0] = qp_table_start_minus26[i] + 26
    qpOutVal[i][0] = qpInVal[i][0]
}

```

```

for(j = 0; j < num_points_in_qp_table[i]; j++) {
    qpInVal[i][j + 1] = qpInVal[i][j] + delta_qp_in_val_minus1[i][j] + 1
    qpOutVal[i][j + 1] = qpOutVal[i][j] +
    ( delta_qp_in_val_minus1[i][j] ^ delta_qp_diff_val[i][j] )                sh =
    ( delta_qp_in_val_minus1[i][j] + 1 ) >> 1
    for(k = qpInVal[i][j] + 1, m = 1; k <= qpInVal[i][j + 1]; k++, m++)
        ChromaQpTable[i][k] = Clip3(-QpBdOffset, 63,
    ChromaQpTable[i][qpInVal[i][j]] +
        ((qpOutVal[i][j + 1] - qpOutVal[i][j]) * m + sh) /
    ( delta_qp_in_val_minus1[i][j] + 1 ))
}
if(num_points_in_qp_table[i] > 0)
    for(k = qpInVal[i][num_points_in_qp_table[i]] + 1; k <= 63; k++)
        ChromaQpTable[i][k] = Clip3(-QpBdOffset, 63, ChromaQpTable[i][k - 1] +
    1)

```

It is a requirement of bitstream conformance that the values of qpInVal[i][j] [[and qpOutVal[i][j]]] shall be in the range of -QpBdOffset to 63, inclusive for i in the range of 0 to numQpTables - 1, inclusive, and j in the range of 0 to num\_points\_in\_qp\_table\_minus1[i] + 1, inclusive.

### 6.3. Embodiment 3

The changes, marked in bold italicized underlined text, are based on JVET-Q2001-vE.

The i-th chroma QP mapping table ChromaQpTable[i] for i = 0..numQpTables - 1 is derived as follows:

```

qpInVal[i][0] = qp_table_start_minus26[i] + 26
qpOutVal[i][0] = qpInVal[i][0]
for(j = 0; j <= num_points_in_qp_table_minus1[i]; j++) {
    qpInVal[i][j + 1] = qpInVal[i][j] + delta_qp_in_val_minus1[i][j] + 1
    qpOutVal[i][j + 1] = qpOutVal[i][j] + ( ( delta_qp_in_val_minus1[i][j] + 1 )
        ^ delta_qp_diff_val[i][j] )

```

```

}
ChromaQpTable[ i ][ qpInVal[ i ][ 0 ] ] = qpOutVal[ i ][ 0 ]
for( k = qpInVal[ i ][ 0 ] - 1; k >= -QpBdOffset; k -- )
    ChromaQpTable[ i ][ k ] = Clip3( -QpBdOffset, 63, ChromaQpTable[ i ][ k + 1 ] - 1 )
for( j = 0; j <= num_points_in_qp_table_minus1[ i ]; j++ ) {
    sh = ( delta_qp_in_val_minus1[ i ][ j ] + 1 ) >> 1
    for( k = qpInVal[ i ][ j ] + 1, m = 1; k <= qpInval[ i ][ j + 1 ]; k++, m++ )
        ChromaQpTable[ i ][ k ] = ChromaQpTable[ i ][ qpInVal[ i ][ j ] ] +
            ( ( qpOutVal[ i ][ j + 1 ] - qpOutVal[ i ][ j ] ) * m + sh ) /
            ( delta_qp_in_val_minus1[ i ][ j ] + 1 )
    }
for( k = qpInVal[ i ][ num_points_in_qp_table_minus1[ i ] + 1 ] + 1; k <= 63; k++ )
    ChromaQpTable[ i ][ k ] = Clip3( -QpBdOffset, 63, ChromaQpTable[ i ][ k - 1 ] + 1 )

```

When same\_qp\_table\_for\_chroma is equal to 1, ChromaQpTable[ 1 ][ k ] and ChromaQpTable[ 2 ][ k ] are set equal to ChromaQpTable[ 0 ][ k ] for k in the range of -QpBdOffset to 63, inclusive.

It is a requirement of bitstream conformance that the values of qpInVal[ i ][ j ] and qpOutVal[ i ][ j ] shall be in the range of -QpBdOffset to 63, inclusive for i in the range of 0 to numQpTables - 1, inclusive, and j in the range of 0 to num\_points\_in\_qp\_table\_minus1[ i ] + 1, inclusive.

**6.4. Embodiment 4**

The newly added text is marked in ***bold italicized underlined*** text and the deleted texts are marked in open and close double brackets (e.g., [[ ]]) indicating deleted text in between the double brackets. It is based on JVET-Q2001-vE.

**7.3.2.3 Sequence parameter set RBSP syntax**

seq_parameter_set_rbsp() {	<b>Descript or</b>
...	

if( ChromaArrayType != 0 ) {	
<u>sps_one_set_of_chroma_qp_tables_flag</u>	<u>u(1)</u>
sps_joint_cbr_enabled_flag	u(1)
same_qp_table_for_chroma_set0	u(1)
numQpTablesSet0 = same_qp_table_for_chroma_set0 ? 1 : ( sps_joint_cbr_enabled_flag ? 3 : 2 )	
for( i = 0; i < numQpTables; i++ ) {	
[[qp_table_start_minus26[ i ]]]	[[se(v)]]
num_points_in_qp_table[[_minus1]]_set0[ i ]	ue(v)
<u>if( num_points_in_qp_table_set0[ i ] )</u>	
<u>qp_table_start_set0[ i ]</u>	<u>ue(v)</u>
for( j = 0; j <[[=]] num_points_in_qp_table[[_minus1]]_set0[ i ]; j++ ) {	
delta_qp_in_val_minus1_set0[ i ][ j ]	ue(v)
delta_qp_diff_val_set0[ i ][ j ]	ue(v)
}	
}	
<u>if( !sps_one_set_of_chroma_qp_tables_flag ) {</u>	
<u>same_qp_table_for_chroma_set1</u>	<u>u(1)</u>
<u>numQpTablesSet1 = same_qp_table_for_chroma_set1 ? 1 :</u> <u>( sps_joint_cbr_enabled_flag ? 3 : 2 )</u>	
<u>for( i = 0; i &lt; numQpTablesSet1; i++ ) {</u>	
<u>num_points_in_qp_table_set1[ i ]</u>	<u>ue(v)</u>
<u>if( num_points_in_qp_table_set1[ i ] )</u>	
<u>qp_table_start_set1[ i ]</u>	<u>ue(v)</u>
<u>for( j = 0; j &lt; num_points_in_qp_table_set1[ i ]; j++ ) {</u>	
<u>delta_qp_in_val_minus1_set1[ i ][ j ]</u>	<u>ue(v)</u>

<u><i>delta qp diff val set1[i][j]</i></u>	<u><i>ue(v)</i></u>
<i>1</i>	
<i>1</i>	
<i>1</i>	
<i>}</i>	
<i>...</i>	

7.4.3.3 Sequence parameter set RBSP semantics

...

*sps\_one\_set\_of\_chroma\_qp\_tables\_flag equal to 0 specifies that there are two sets of chroma QP mapping tables. sps\_one\_set\_of\_chroma\_qp\_tables\_flag equal to 1 specifies that there are just one set of chroma QP mapping tables.*

**same\_qp\_table\_for\_chroma\_set0** equal to 1 specifies that only one chroma QP mapping table is signalled and this table applies to Cb and Cr residuals and additionally to joint Cb-Cr residuals when **sps\_joint\_cbr\_enabled\_flag** is equal to 1 *for the 1<sup>st</sup> set of chroma QP mapping tables.* **same\_qp\_table\_for\_chroma\_set0** equal to 0 specifies that chroma QP mapping tables, two for Cb and Cr, and one additional for joint Cb-Cr when **sps\_joint\_cbr\_enabled\_flag** is equal to 1, *for the 1<sup>st</sup> set of chroma QP mapping tables,* are signalled in the SPS. When **same\_qp\_table\_for\_chroma\_set0** is not present in the bitstream, the value of **same\_qp\_table\_for\_chroma\_set0** is inferred to be equal to 1.

**num\_points\_in\_qp\_table\_minus1\_set0**[ i ] plus 1 specifies the number of points used to describe the i-th chroma QP mapping table *for the 1<sup>st</sup> set of chroma QP mapping tables.* The value of **num\_points\_in\_qp\_table\_minus1\_set0**[ i ] shall be in the range of 0 to 63 + QpBdOffset, inclusive. When **num\_points\_in\_qp\_table\_minus1\_set0**[ 0 ] is not present in the bitstream, the value of **num\_points\_in\_qp\_table\_minus1\_set0**[ 0 ] is inferred to be equal to 0.

**qp\_table\_start**[ [ **minus26** ] **set0**[ i ] [ [ **plus26** ] ] specifies the starting luma and chroma QP used to describe the i-th chroma QP mapping table *for the 1<sup>st</sup> set of chroma QP mapping tables.* The value of **qp\_table\_start**[ [ **minus26** ] **set0**[ i ] shall be in the range of [ [ **-26** ] ] – QpBdOffset to

[[36]] 62 inclusive. When  $qp\_table\_start[[\_minus26]]\_set0[i]$  is not present in the bitstream, the value of  $qp\_table\_start[[\_minus26]]\_set0[i]$  is inferred to be equal to 0.

$\mathit{delta\_qp\_in\_val\_minus1\_set0}[i][j]$  specifies a delta value used to derive the input coordinate of the  $j$ -th pivot point of the  $i$ -th chroma QP mapping table for the 1<sup>st</sup> set of chroma QP mapping tables. When  $\mathit{delta\_qp\_in\_val\_minus1\_set0}[0][j]$  is not present in the bitstream, the value of  $\mathit{delta\_qp\_in\_val\_minus1\_set0}[0][j]$  is inferred to be equal to 0.

$\mathit{delta\_qp\_diff\_val\_set0}[i][j]$  specifies a delta value used to derive the output coordinate of the  $j$ -th pivot point of the  $i$ -th chroma QP mapping table for the 1<sup>st</sup> set of chroma QP mapping tables.

same qp table for chroma set1 equal to 1 specifies that only one chroma QP mapping table is signalled and this table applies to Cb and Cr residuals and additionally to joint Cb-Cr residuals when sps joint cbr enabled flag is equal to 1 for the 2<sup>nd</sup> set of chroma QP mapping tables. same qp table for chroma set1 equal to 0 specifies that chroma QP mapping tables, two for Cb and Cr, and one additional for joint Cb-Cr when sps joint cbr enabled flag is equal to 1, for the 2<sup>nd</sup> set of chroma QP mapping tables, are signalled in the SPS. When same qp table for chroma set1 is not present in the bitstream, the value of same qp table for chroma set1 is inferred to be equal to 1.

num points in qp table minus1 set1[i] plus 1 specifies the number of points used to describe the  $i$ -th chroma QP mapping table for the 2<sup>nd</sup> set of chroma QP mapping tables. The value of num points in qp table minus1 set1[i] shall be in the range of 0 to  $63 + OpBdOffset$ , inclusive. When num points in qp table minus1 set1[0] is not present in the bitstream, the value of num points in qp table minus1 set1[0] is inferred to be equal to 0.

qp table start set1[i] specifies the starting luma and chroma QP used to describe the  $i$ -th chroma QP mapping table for the 2<sup>nd</sup> set of chroma QP mapping tables. The value of qp table start set1[i] shall be in the range of  $-OpBdOffset$  to 62 inclusive. When qp table start set1[i] is not present in the bitstream, the value of qp table start set1[i] is inferred to be equal to 0.

delta qp in val minus1 set1[i][j] specifies a delta value used to derive the input coordinate of the  $j$ -th pivot point of the  $i$ -th chroma QP mapping table for the 2<sup>nd</sup> set of chroma QP mapping

tables. When  $\text{delta\_qp\_in\_val\_minus1\_set1}[0][j]$  is not present in the bitstream, the value of  $\text{delta\_qp\_in\_val\_minus1\_set1}[0][j]$  is inferred to be equal to 0.

$\text{delta\_qp\_diff\_val\_set1}[i][j]$  specifies a delta value used to derive the output coordinate of the  $j$ -th pivot point of the  $i$ -th chroma OP mapping table for the 2<sup>nd</sup> set of chroma OP mapping tables.

The  $i$ -th chroma OP mapping table  $\text{ChromaOpTableSet0}[i]$  for  $i = 0..numOpTablesSet0 - 1$  and the  $i$ -th chroma OP mapping table  $\text{ChromaOpTableSet1}[i]$  for  $i = 0..numOpTablesSet1 - 1$  are derived as follows:

Let  $\text{ChromaOpTable}[i]$  denotes the  $\text{ChromaOpTableSetX}[i]$ ,  $\text{qp\_table\_start}[i]$  denotes the  $\text{qp\_table\_start\_setX}[i]$ ,  $\text{delta\_qp\_in\_val\_minus1}[i]$  denotes  $\text{delta\_qp\_in\_val\_minus1\_setX}[i]$ ,  $\text{delta\_qp\_diff\_val}[i]$  denotes  $\text{delta\_qp\_diff\_val\_setX}[i]$  for  $X$  being 0 and 1 respectively, the following applies:

```

qpInVal[i][0] = qp_table_start[_minus26][i][+26]
qpOutVal[i][0] = qpInVal[i][0]
for(j = 0; j <=[ ] num_points_in_qp_table[_minus1][i]; j++) {
    qpInVal[i][j+1] = qpInVal[i][j] + delta_qp_in_val_minus1[i][j] + 1
    qpOutVal[i][j+1] = qpOutVal[i][j] +
    ( delta_qp_in_val_minus1[i][j] ^ delta_qp_diff_val[i][j] )
}
ChromaOpTable[i][qpInVal[i][0]] = qpOutVal[i][0]
for(k = qpInVal[i][0] - 1; k >= -QpBdOffset; k--)
    ChromaOpTable[i][k] = Clip3(-QpBdOffset, 63, ChromaOpTable[i][k+1] - 1)
(62)
for(j = 0; j <=[ ] num_points_in_qp_table[_minus1][i]; j++) {
    sh = ( delta_qp_in_val_minus1[i][j] + 1 ) >> 1
    for(k = qpInVal[i][j] + 1, m = 1; k <= qpInVal[i][j+1]; k++, m++)
        ChromaOpTable[i][k] = ChromaOpTable[i][qpInVal[i][j]] +
        (( qpOutVal[i][j+1] - qpOutVal[i][j] ) * m + sh) /
    ( delta_qp_in_val_minus1[i][j] + 1 )

```

```

}
if( num_points_in_qp_table[ i ] == 0 )
    for( k = qpInVal[ i ][ 0 ] + 1; k <= 63; k ++ )
        ChromaOpTable[ i ][ k ] = Clip3( -QpBdOffset, 63,
ChromaOpTable[ i ][ k - 1 ] + 1 )
else
    for( k = qpInVal[ i ][ num_points_in_qp_table[ i ] + 1 ]; k <= 63;
k++ )
        ChromaOpTable[ i ][ k ] = Clip3( -QpBdOffset, 63, ChromaOpTable[ i ][ k - 1 ] +
1 )

```

When `same_qp_table_for_chroma $X$`  is equal to 1, `ChromaOpTable $X$ [ 1 ][ k ]` and `ChromaOpTable $X$ [ 2 ][ k ]` are set equal to `ChromaOpTable $X$ [ 0 ][ k ]`, for  $X$  being 0 and 1 and for  $k$  in the range of `-QpBdOffset` to 63, inclusive.

It is a requirement of bitstream conformance that the values of `qpInVal[ i ][ j ]` and `qpOutVal[ i ][ j ]` shall be in the range of `-QpBdOffset` to 63, inclusive for  $i$  in the range of 0 to `numQpTables - 1`, inclusive, and  $j$  in the range of 0 to `num_points_in_qp_table_minus1[ i ] + 1`, inclusive.

...

### 8.7.1 Derivation process for quantization parameters

...

When the current slice is an I slice, `ChromaOpTable[ i ]` is set equal to `ChromaOpTableSet0[ i ]` for  $i = 0..2$ . When the current slices is a B or P slice, `ChromaOpTable[ i ]` is set equal to `ChromaOpTableSet1[ i ]` for  $i = 0..2$  when `sps_one_set_of_chroma_qp_tables_flag` is equal to 0, otherwise (when `sps_one_set_of_chroma_qp_tables_flag` is equal to 1), `ChromaOpTable[ i ]` is set equal to `ChromaOpTableSet0[ i ]` for  $i = 0..2$ .

When `ChromaArrayType` is not equal to 0 and `treeType` is equal to `SINGLE_TREE` or `DUAL_TREE_CHROMA`, the following applies:

- When treeType is equal to DUAL\_TREE\_CHROMA, the variable QpY is set equal to the luma quantization parameter QpY of the luma coding unit that covers the luma location ( xCb + cbWidth / 2, yCb + cbHeight / 2 ).
- The variables qP<sub>Cb</sub>, qP<sub>Cr</sub> and qP<sub>CbCr</sub> are derived as follows:

$$qP_{\text{Chroma}} = \text{Clip3}( -QpBdOffset, 63, QpY ) \tag{1143}$$

$$qP_{\text{Cb}} = \text{ChromaQpTable}[ 0 ][ qP_{\text{Chroma}} ] \tag{1144}$$

$$qP_{\text{Cr}} = \text{ChromaQpTable}[ 1 ][ qP_{\text{Chroma}} ] \tag{1145}$$

$$qP_{\text{CbCr}} = \text{ChromaQpTable}[ 2 ][ qP_{\text{Chroma}} ] \tag{1146}$$

...

### 6.5. Embodiment 5

The newly added text is marked in ***bold italicized underlined*** and the deleted texts are marked in open and close double brackets (e.g., [[ ]]) indicating deleted text in between the double brackets. It is based on JVET-Q2001-vE.

#### 7.3.2.3 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp( ) {	<b>Descript or</b>
...	
if( ChromaArrayType != 0 ) {	
<b><i><u>sps_one_set_of_chroma_qp_tables_flag</u></i></b>	<b><i><u>u(1)</u></i></b>
sps_joint_cbr_enabled_flag	u(1)
<b><i><u>numTables = sps_one_set_of_chroma_qp_tables_flag ? 1 : 2</u></i></b>	
<b><i><u>for( l = 0; l &lt;= numTables; l++ ) {</u></i></b>	
<b><i><u>same_qp_table_for_chroma[l]</u></i></b>	u(1)

numQpTables $[L]$ = same_qp_table_for_chroma $[L]$ ? 1 :	
( sps_joint_cbr_enabled_flag ? 3 : 2 )	
for( i = 0; i < numQpTables $[L]$ ; i++ ) {	
qp_table_start_minus26 $[L][ i ]$	se(v)
num_points_in_qp_table_minus1 $[L][ i ]$	ue(v)
for( j = 0; j <= num_points_in_qp_table_minus1 $[L][ i ]$ ; j++ ) {	
delta_qp_in_val_minus1 $[L][ i ][ j ]$	ue(v)
delta_qp_diff_val $[L][ i ][ j ]$	ue(v)
}	
}	
}	
}	
...	

7.4.3.3 Sequence parameter set RBSP semantics

...

sps\_one\_set\_of\_chroma\_qp\_tables\_flag equal to 0 specifies that there are two sets of chroma QP mapping tables. sps\_one\_set\_of\_chroma\_qp\_tables\_flag equal to 1 specifies that there are just one set of chroma QP mapping tables. When not present, the value of sps\_one\_set\_of\_chroma\_qp\_tables\_flag is inferred to be equal to 1.

same\_qp\_table\_for\_chroma $[L]$  equal to 1 specifies that only one chroma QP mapping table is signalled and this table applies to Cb and Cr residuals and additionally to joint Cb-Cr residuals when sps\_joint\_cbr\_enabled\_flag is equal to 1 for the l-th set of chroma QP mapping tables. same\_qp\_table\_for\_chroma $[L]$  equal to 0 specifies that chroma QP mapping tables, two for Cb and Cr, and one additional for joint Cb-Cr when sps\_joint\_cbr\_enabled\_flag is equal to 1, for the l-th set of chroma QP mapping tables, are signalled in the SPS. When

same\_qp\_table\_for\_chroma[ $L$ ][ $i$ ] is not present in the bitstream, the value of same\_qp\_table\_for\_chroma[ $L$ ][ $i$ ] is inferred to be equal to 1.

qp\_table\_start\_minus26[ $L$ ][ $i$ ] plus 26 specifies the starting luma and chroma QP used to describe the  $i$ -th chroma QP mapping table for the  $l$ -th set of chroma QP mapping tables. The value of qp\_table\_start\_minus26[ $L$ ][ $i$ ] shall be in the range of  $-26 - \text{QpBdOffset}$  to 36 inclusive. When qp\_table\_start\_minus26[ $L$ ][ $i$ ] is not present in the bitstream, the value of qp\_table\_start\_minus26[ $L$ ][ $i$ ] is inferred to be equal to 0.

num\_points\_in\_qp\_table\_minus1[ $L$ ][ $i$ ] plus 1 specifies the number of points used to describe the  $i$ -th chroma QP mapping table for the  $l$ -th set of chroma QP mapping tables. The value of num\_points\_in\_qp\_table\_minus1[ $L$ ][ $i$ ] shall be in the range of 0 to  $63 + \text{QpBdOffset}$ , inclusive. When num\_points\_in\_qp\_table\_minus1[ $L$ ][0] is not present in the bitstream, the value of num\_points\_in\_qp\_table\_minus1[ $L$ ][0] is inferred to be equal to 0.

delta\_qp\_in\_val\_minus1[ $L$ ][ $i$ ][ $j$ ] specifies a delta value used to derive the input coordinate of the  $j$ -th pivot point of the  $i$ -th chroma QP mapping table for the  $l$ -th set of chroma QP mapping tables. When delta\_qp\_in\_val\_minus1[ $L$ ][0][ $j$ ] is not present in the bitstream, the value of delta\_qp\_in\_val\_minus1[ $L$ ][0][ $j$ ] is inferred to be equal to 0.

delta\_qp\_diff\_val[ $L$ ][ $i$ ][ $j$ ] specifies a delta value used to derive the output coordinate of the  $j$ -th pivot point of the  $i$ -th chroma QP mapping table for the  $l$ -th set of chroma QP mapping tables.

The  $l$ -th chroma QP mapping table ChromaOpTable[ $l$ ][ $i$ ] for  $l = 0..sps\_one\_set\_of\_chroma\_qp\_tables\_flag - 1$  and  $i = 0..numOpTables[ $l$ ] - 1$  are derived as follows:

$$\text{qpInVal}[L][i][0] = \text{qp\_table\_start\_minus26}[L][i] + 26$$

$$\text{qpOutVal}[L][i][0] = \text{qpInVal}[L][i][0]$$

for( $j = 0; j \leq \text{num\_points\_in\_qp\_table\_minus1}[L][i]; j++$ ) {

$$\text{qpInVal}[L][i][j+1] = \text{qpInVal}[L][i][j] + \text{delta\_qp\_in\_val\_minus1}[L][i][j] + 1$$

$$\text{qpOutVal}[L][i][j+1] = \text{qpOutVal}[L][i][j] +$$

$$(\text{delta\_qp\_in\_val\_minus1}[L][i][j] \wedge \text{delta\_qp\_diff\_val}[L][i][j])$$

```

}
ChromaQpTable[L][i][ qpInVal[L][i][ 0 ] ] = qpOutVal[L][i][ 0 ]
for( k = qpInVal[L][i][ 0 ] - 1; k >= -QpBdOffset; k -- )
    ChromaQpTable [L][i][ k ] = Clip3( -QpBdOffset, 63,
ChromaQpTable[L][i][ k + 1 ] - 1 )
for( j = 0; j <= num_points_in_qp_table_minus1[L][i]; j++ ) {
    sh = ( delta_qp_in_val_minus1[L][i][ j ] + 1 ) >> 1
    for( k = qpInVal[L][i][ j ] + 1, m = 1; k <= qpInVal[L][i][ j + 1 ]; k++, m++ )
        ChromaQpTable[L][i][ k ] = ChromaQpTable[L][i][ qpInVal[L][i][ j ] ] +
            ( ( qpOutVal[L][i][ j + 1 ] - qpOutVal[L][i][ j ] ) * m + sh ) /
            ( delta_qp_in_val_minus1[L][i][ j ] + 1 )
}
for( k = qpInVal[L][i][ num_points_in_qp_table_minus1[L][i] + 1 ] + 1; k <= 63; k++ )
    ChromaQpTable[L][i][ k ] = Clip3( -QpBdOffset, 63, ChromaQpTable[L][i][ k - 1 ]
+ 1 )

```

When `same_qp_table_for_chroma[L]` is equal to 1, `ChromaQpTable[L][1][k]` and `ChromaQpTable[L][2][k]` are set equal to `ChromaQpTable[L][0][k]` for `k` in the range of `-QpBdOffset` to 63, inclusive.

It is a requirement of bitstream conformance that the values of `qpInVal[L][i][j]` and `qpOutVal[L][i][j]` shall be in the range of `-QpBdOffset` to 63, inclusive for `i` in the range of 0 to `numQpTables[L]-1`, inclusive, and `j` in the range of 0 to `num_points_in_qp_table_minus1[L][i]+1`, inclusive.

...

### 8.7.1 Derivation process for quantization parameters

...

**When the current slice is an I slice, `chromaOpTable[i]` is set equal to `ChromaOpTable[0][i]` for `i = 0..2`. When the current slices is a B or P slice, `chromaOpTable[i]` is set equal to**

*ChromaOpTable[ 1 ][ i ] for i = 0..2 when sps one set of chroma qp tables flag is equal to 0, otherwise (when sps one set of chroma qp tables flag is equal to 1), chromaOpTable[ i ] is set equal to ChromaOpTable[ 0 ][ i ] for i = 0..2.*

When ChromaArrayType is not equal to 0 and treeType is equal to SINGLE\_TREE or DUAL\_TREE\_CHROMA, the following applies:

- When treeType is equal to DUAL\_TREE\_CHROMA, the variable  $Q_{pY}$  is set equal to the luma quantization parameter  $Q_{pY}$  of the luma coding unit that covers the luma location (  $x_{Cb} + cbWidth / 2, y_{Cb} + cbHeight / 2$  ).
- The variables  $qP_{Cb}$ ,  $qP_{Cr}$  and  $qP_{CbCr}$  are derived as follows:

$$qP_{Chroma} = \text{Clip3}( -Q_{pBdOffset}, 63, Q_{pY} ) \quad (1143)$$

$$qP_{Cb} = \text{[[C]]chromaOpTable[ 0 ][ } qP_{Chroma} \text{ ]} \quad (1144)$$

$$qP_{Cr} = \text{[[C]]chromaOpTable[ 1 ][ } qP_{Chroma} \text{ ]} \quad (1145)$$

$$qP_{CbCr} = \text{[[C]]chromaOpTable[ 2 ][ } qP_{Chroma} \text{ ]} \quad (1146)$$

## 6.6. Embodiment 6

The suggested specification changes on top of JVET-Q2001-vE is described as follows. Deleted texts are marked in open and close double brackets (e.g., [[ ]]) indicating deleted text in between the double brackets, and newly added text is highlighted in ***bold italicized underlined*** text.

### 7.3.10.2 Coding tree unit syntax

coding_tree_unit() {	Descript or
xCtb = CtbAddrX << CtbLog2SizeY	
yCtb = CtbAddrY << CtbLog2SizeY	
if( slice_sao_luma_flag    slice_sao_chroma_flag )	
sao( CtbAddrX, CtbAddrY )	
if( slice_alf_enabled_flag ){	
<b>alf_ctb_flag</b> [ 0 ][ CtbAddrX ][ CtbAddrY ]	ae(v)
if( alf_ctb_flag[ 0 ][ CtbAddrX ][ CtbAddrY ] ) {	
if( slice_num_alf_aps_ids_luma > 0 )	
<b>alf_use_aps_flag</b>	ae(v)
if( alf_use_aps_flag ) {	
if( slice_num_alf_aps_ids_luma > 1 )	
<b>alf_luma_prev_filter_idx</b>	ae(v)
} else	
<b>alf_luma_fixed_filter_idx</b>	ae(v)
[[ ]]	
if( slice_alf_chroma_idc == 1    slice_alf_chroma_idc == 3 ) {	
<b>alf_ctb_flag</b> [ 1 ][ CtbAddrX ][ CtbAddrY ]	ae(v)
if( alf_ctb_flag[ 1 ][ CtbAddrX ][ CtbAddrY ] && alf_chroma_num_alt_filters_minus1 > 0 )	
<b>alf_ctb_filter_alt_idx</b> [ 0 ][ CtbAddrX ][ CtbAddrY ]	ae(v)
}	
if( slice_alf_chroma_idc == 2    slice_alf_chroma_idc == 3 ) {	
<b>alf_ctb_flag</b> [ 2 ][ CtbAddrX ][ CtbAddrY ]	ae(v)

if( alf_ctb_flag[ 2 ][ CtbAddrX ][ CtbAddrY ] && alf_chroma_num_alt_filters_minus1 > 0 )	
<b>alf_ctb_filter_alt_idx</b> [ 1 ][ CtbAddrX ][ CtbAddrY ]	ae(v)
}	
[[ ]]	
if( slice_cc_alf_cb_enabled_flag )	
<b>alf_ctb_cc_cb_idc</b> [ CtbAddrX ][ CtbAddrY ]	ae(v)
if( slice_cc_alf_cr_enabled_flag )	
<b>alf_ctb_cc_cr_idc</b> [ CtbAddrX ][ CtbAddrY ]	ae(v)
<u> </u>	
<u> </u>	
if( slice_type == I && qtbt_dual_tree_intra_flag )	
dual_tree_implicit_qt_split( xCtb, yCtb, CtbSizeY, 0 )	
else	
coding_tree( xCtb, yCtb, CtbSizeY, CtbSizeY, 1, 1, 0, 0, 0, 0, 0, SINGLE_TREE, MODE_TYPE_ALL )	
}	

**6.7. Seventh set of embodiments**

This is a set of embodiments for items 20 summarized above in Section 5.

*The changed texts are based on the latest VVC text in JVET-Q2001-vE. Most relevant parts that have been added or modified are highlighted in **bold italicized underlined** text, and some of the deleted texts are marked in open and close double brackets (e.g., [[ ]]) indicating deleted text in between the double brackets.*

**6.7.1. An example embodiment**

In one example, the SPS syntax structure may be changed as follows:

seq_parameter_set_rbsp() {	<b>Descript or</b>
...	u(4)
sps_transform_skip_enabled_flag	u(1)
if( sps_transform_skip_enabled_flag ) {	
<u>sps ts residual coding enabled flag</u>	<u>u(1)</u>
log2_transform_skip_max_size_minus2	ue(v)
sps_bdpcm_enabled_flag	u(1)
}	
...	

sps ts residual coding enabled flag equal to 1 specifies that slice ts residual coding disabled flag may be present in the CLVS.  
sps ts residual coding enabled flag equal to 0 specifies that slice ts residual coding disabled flag is not present in the CLVS. When not present, the value of sps ts residual coding enabled flag is inferred to be equal to 0.

In one example, the PPS syntax structure may be changed as follows:

pic_parameter_set_rbsp() {	<b>Descript or</b>
...	
cabac_init_present_flag	u(1)
<u>pps ts residual coding enabled flag</u>	<u>u(1)</u>
<u>if(pps ts residual coding enabled flag)</u>	
<u>ts residual coding flag in ph</u>	<u>u(1)</u>

pps ts residual coding enabled flag equal to 1 specifies that ph/slice ts residual coding enabled flag may be present in pictures/slices referring to the PPS. pps ts residual coding enabled flag equal to 0 specifies that ph/slice ts residual coding enabled flag is not present in the pictures/slices referring to the PPS.

It is required that, when sps ts residual coding enabled flag is equal to 0, the value of pps ts residual coding enabled flag shall be equal to 0.

It is required that, when sps transform skip enabled flag is equal to 0, the value of pps ts residual coding enabled flag shall be equal to 0.

ts residual coding flag in ph equal to 1 specifies that ph ts residual coding disabled flag may be present in the PH. pps ts residual coding enabled flag equal to 0 specifies that slice ts residual coding disabled flag may be present in the SH.

In one example, the PH syntax structure may be changed as follows:

picture_header_structure( ) {	<b>Descript or</b>
...	
if( output_flag_present_flag )	
<b>pic_output_flag</b>	<b>u(1)</b>
<u>if( pps ts residual coding enabled flag &amp;&amp; ts residual coding flag in ph )</u>	
<u>ph ts residual coding enabled flag</u>	<u><b>u(1)</b></u>
...	

ph ts residual coding enabled flag equal to 0 specifies that the residual\_coding() syntax structure is used to parse the residual samples of a transform skip block for all slices of the

current picture. *ph ts residual coding enabled flag equal to 1 specifies that the residual ts coding() syntax structure may be used to parse the residual samples of a transform skip block for all slices of the current picture. When ph ts residual coding enabled flag is not present, it is inferred to be equal to 0.*

In one example, the PH syntax structure may be changed as follows:

slice_header() {	Descriptor
...	
<u><i>if(pps ts residual coding enabled flag &amp;&amp; !ts residual coding flag in ph )</i></u>	
slice_ts_residual_coding_[[disabled]] <u><i>enabled flag</i></u>	u(1)

slice\_ts\_residual\_coding\_[[disabled]]*enabled flag* equal to [[1]] 0 specifies that the residual\_coding( ) syntax structure is used to parse the residual samples of a transform skip block for the current slice. slice\_ts\_residual\_coding\_[[disabled]]*enabled flag* equal to [[0]] 1 specifies that the residual\_ts\_coding( ) syntax structure is used to parse the residual samples of a transform skip block for the current slice. When slice\_ts\_residual\_coding\_[[disabled]]*enabled flag* is not present, it is inferred to be equal to [[0]] *ph ts residual coding enabled flag*.

In one example, the transform\_unit() syntax structure may be changed as follows:

transform_unit( x0, y0, tbWidth, tbHeight, treeType, subTuIndex, chType ) {	<b>Descriptor</b>
...	
if( tu_y_coded_flag[ x0 ][ y0 ] && treeType != DUAL_TREE_CHROMA ) {	

if( sps_transform_skip_enabled_flag && !BdpcmFlag[ x0 ][ y0 ][ 0 ] && tbWidth <= MaxTsSize && tbHeight <= MaxTsSize && ( IntraSubPartitionsSplitType == ISP_NO_SPLIT ) && !cu_sbt_flag )	
<b>transform_skip_flag[ x0 ][ y0 ][ 0 ]</b>	<b>ae(v)</b>
<u><i>if(slice ts residual coding enabled flag &amp;&amp;                  transform skip flag[ x0 ][ y0 ][ 0 ] )</i></u>	
<u><i>tu ts residual coding flag[ x0 ][ y0 ][ 0 ]</i></u>	<u><i>ae(v)</i></u>
if( !transform_skip_flag[ x0 ][ y0 ][ 0 ]    <u><i>!tu ts residual coding flag[ x0 ][ y0 ][ 0 ]</i></u> [[slice_ts_residual_coding_disabled _flag]] )	
residual_coding( x0, y0, Log2( tbWidth ), Log2( tbHeight ), 0 )	
else	
residual_ts_coding( x0, y0, Log2( tbWidth ), Log2( tbHeight ), 0 )	
}	
if( tu_cb_coded_flag[ xC ][ yC ] && treeType != DUAL_TREE_LUMA ) {	
if( sps_transform_skip_enabled_flag && !BdpcmFlag[ x0 ][ y0 ][ 1 ] && wC <= MaxTsSize && hC <= MaxTsSize && !cu_sbt_flag )	
<b>transform_skip_flag[ xC ][ yC ][ 1 ]</b>	<b>ae(v)</b>
<u><i>if(slice ts residual coding enabled flag &amp;&amp;                  transform skip flag[ xC ][ yC ][ 1 ] )</i></u>	
<u><i>tu ts residual coding flag[ xC ][ yC ][ 1 ]</i></u>	<u><i>ae(v)</i></u>
if( !transform_skip_flag[ xC ][ yC ][ 1 ]    <u><i>!tu ts residual coding flag[ xC ][ yC ][ 1 ]</i></u> [[slice_ts_residual_coding_disabled _flag]] )	
residual_coding( xC, yC, Log2( wC ), Log2( hC ), 1 )	
else	
residual_ts_coding( xC, yC, Log2( wC ), Log2( hC ), 1 )	
}	

<pre> if( tu_cr_coded_flag[ xC ][ yC ] &amp;&amp; treeType != DUAL_TREE_LUMA &amp;&amp;     !( tu_cb_coded_flag[ xC ][ yC ] &amp;&amp;     tu_joint_cbr_residual_flag[ xC ][ yC ] ) ) { </pre>	
<pre> if( sps_transform_skip_enabled_flag &amp;&amp; !BdpcmFlag[ x0 ][ y0 ][ 2 ] &amp;&amp;     wC &lt;= MaxTsSize &amp;&amp; hC &lt;= MaxTsSize &amp;&amp; !cu_sbt_flag ) </pre>	
<pre> transform_skip_flag[ xC ][ yC ][ 2 ] </pre>	ae(v)
<pre> <u>if(slice ts residual coding enabled flag &amp;&amp; transform skip flag[ xC ][ yC ][ 2 ] )</u> </pre>	
<pre> <u>tu ts residual coding flag[ xC ][ yC ][ 2 ]</u> </pre>	<u>ae(v)</u>
<pre> if( !transform_skip_flag[ xC ][ yC ][ 2 ]    <u>!tu ts residual coding flag[ xC ][ yC ][ 2 ]</u> [ [slice_ts_residual_coding_disabled _flag] ] ) </pre>	
<pre> residual_coding( xC, yC, Log2( wC ), Log2( hC ), 2 ) </pre>	
<pre> else </pre>	
<pre> residual_ts_coding( xC, yC, Log2( wC ), Log2( hC ), 2 ) </pre>	
<pre> } </pre>	
<pre> } </pre>	

tu ts residual coding flag[ x0 ][ y0 ][ cIdx ] equal to 1 specifies that for the current coding unit, when slice ts residual coding enabled flag and transform skip flag[ x0 ][ y0 ][ cIdx ] are equal to 1, then transform skip residual coding is used.

tu ts residual coding flag[ x0 ][ y0 ][ cIdx ] equal to 0 specifies that for the current coding unit, when slice ts residual coding enabled flag and transform skip flag[ x0 ][ y0 ][ cIdx ] are equal to 1, the transform residual coding is used. When

tu ts residual coding flag[ x0 ][ y0 ][ cIdx ] is not signaled, it is inferred to be equal to 0.

### 6.7.2. Another example embodiment

Alternatively, the PPS, PH, SH syntax may be changed as follows:

pic_parameter_set_rbsp() {	<b>Descript or</b>
...	
<b>cabac_init_present_flag</b>	<b>u(1)</b>
<b><u>pps_ts_residual_coding_enabled_flag</u></b>	<b><u>u(1)</u></b>

pps\_ts\_residual\_coding\_enabled\_flag equal to 1 specifies that ph/slice\_ts\_residual\_coding\_enabled\_flag may be present in pictures/slices referring to the PPS. pps\_ts\_residual\_coding\_enabled\_flag equal to 0 specifies that ph/slice\_ts\_residual\_coding\_enabled\_flag is not present in the pictures/slices referring to the PPS.

It is required that, when sps\_ts\_residual\_coding\_enabled\_flag is equal to 0, the value of pps\_ts\_residual\_coding\_enabled\_flag shall be equal to 0.

It is required that, when sps\_transform\_skip\_enabled\_flag is equal to 0, the value of pps\_ts\_residual\_coding\_enabled\_flag shall be equal to 0.

picture_header_structure() {	<b>Descript or</b>
...	
if( output_flag_present_flag )	
<b>pic_output_flag</b>	<b>u(1)</b>
<b><u>if(pps_ts_residual_coding_enabled_flag)</u></b>	
<b><u>ph_ts_residual_coding_enabled_flag</u></b>	<b><u>u(1)</u></b>
...	

ph\_ts\_residual\_coding\_enabled\_flag equal to 0 specifies that the residual\_coding() syntax structure is used to parse the residual samples of a transform skip block for all slices of the current picture. ph\_ts\_residual\_coding\_enabled\_flag equal to 1 specifies that the

residual\_ts\_coding() syntax structure may be used to parse the residual samples of a transform skip block for all slices of the current picture. When ph\_ts\_residual\_coding\_enabled flag is not present, it is inferred to be equal to 0.

slice_header() {	Descriptor
...	
<u>if(ph_ts_residual_coding_enabled flag)</u>	
slice_ts_residual_coding_00disabled00enabled_flag	u(1)

slice\_ts\_residual\_coding\_00disabled00enabled\_flag equal to 0 specifies that the residual\_coding() syntax structure is used to parse the residual samples of a transform skip block for the current slice. slice\_ts\_residual\_coding\_00disabled00enabled\_flag equal to 1 specifies that the residual\_ts\_coding() syntax structure is used to parse the residual samples of a transform skip block for the current slice. When slice\_ts\_residual\_coding\_00disabled00enabled\_flag is not present, it is inferred to be equal to 0.

**6.7.3. Another example embodiment**

Alternatively, the PPS, SH syntax may be changed as follows:

pic_parameter_set_rbsp() {	<b>Descriptor</b>
...	
<b>cabac_init_present_flag</b>	<b>u(1)</b>
<u>pps_ts_residual_coding_enabled_flag</u>	<u>u(1)</u>

pps\_ts\_residual\_coding\_enabled\_flag equal to 1 specifies that ph/slice ts\_residual\_coding\_enabled flag may be present in pictures/slices referring to the

PPS. pps ts residual coding enabled flag equal to 0 specifies that ph/slice ts residual coding enabled flag is not present in the pictures/slices referring to the PPS.

It is required that, when sps ts residual coding enabled flag is equal to 0, the value of pps ts residual coding enabled flag shall be equal to 0.

It is required that, when sps transform skip enabled flag is equal to 0, the value of pps ts residual coding enabled flag shall be equal to 0.

slice_header() {	Descriptor
...	
<u>if(pps ts residual coding enabled flag)</u>	
slice_ts_residual_coding_[[disabled]] <u>enabled</u> flag	u(1)

slice\_ts\_residual\_coding\_[[disabled]]enabled flag equal to [[1]] **0** specifies that the residual\_coding( ) syntax structure is used to parse the residual samples of a transform skip block for the current slice. slice\_ts\_residual\_coding\_[[disabled]]enabled flag equal to [[0]] **1** specifies that the residual\_ts\_coding( ) syntax structure is used to parse the residual samples of a transform skip block for the current slice. When slice\_ts\_residual\_coding\_[[disabled]]enabled flag is not present, it is inferred to be equal to 0.

**6.7.4. Another example embodiment**

Alternatively, the SH syntax may be changed as follows:

slice_header() {	Descriptor
...	

<i>if(sps transform skip enabled flag)</i>	
<b>slice_ts_residual_coding_</b> <del>[[disabled]]</del> <b>enabled_flag</b>	u(1)

**slice\_ts\_residual\_coding\_**~~[[disabled]]~~**enabled\_flag** equal to ~~[[1]]~~ **0** specifies that the residual\_coding( ) syntax structure is used to parse the residual samples of a transform skip block for the current slice. **slice\_ts\_residual\_coding\_**~~[[disabled]]~~**enabled\_flag** equal to ~~[[0]]~~ **1** specifies that the residual\_ts\_coding( ) syntax structure is used to parse the residual samples of a transform skip block for the current slice. When **slice\_ts\_residual\_coding\_**~~[[disabled]]~~**enabled\_flag** is not present, it is inferred to be equal to 0.

**[0054]** FIG. 3 is a block diagram showing an example video processing system 1900 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 1900. The system 1900 may include input 1902 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 1902 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.

**[0055]** The system 1900 may include a coding component 1904 that may implement the various coding or encoding methods described in the present document. The coding component 1904 may reduce the average bitrate of video from the input 1902 to the output of the coding component 1904 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 1904 may be either stored, or transmitted via a communication connected, as represented by the component 1906. The stored or communicated bitstream (or coded) representation of the video received at the input 1902 may be used by the component 1908 for generating pixel values or displayable video that is sent to a display interface 1910. 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.

**[0056]** 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.

**[0057]** FIG. 4 is a block diagram of a video processing apparatus 3600. The apparatus 3600 may be used to implement one or more of the methods described herein. The apparatus 3600 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 3600 may include one or more processors 3602, one or more memories 3604 and video processing hardware 3606. The processor(s) 3602 may be configured to implement one or more methods described in the present document. The memory (memories) 3604 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 3606 may be used to implement, in hardware circuitry, some techniques described in the present document.

**[0058]** FIG. 6 is a block diagram that illustrates an example video coding system 100 that may utilize the techniques of this disclosure.

**[0059]** As shown in FIG. 6, 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.

**[0060]** Source device 110 may include a video source 112, a video encoder 114, and an input/output (I/O) interface 116.

**[0061]** 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.

**[0062]** Destination device 120 may include an I/O interface 126, a video decoder 124, and a display device 122.

**[0063]** 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.

**[0064]** 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 (VVC) standard and other current and/or further standards.

**[0065]** FIG. 7 is a block diagram illustrating an example of video encoder 200, which may be video encoder 114 in the system 100 illustrated in FIG. 6.

**[0066]** Video encoder 200 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 7, 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.

**[0067]** 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.

**[0068]** 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.

**[0069]** 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. 7 separately for purposes of explanation.

**[0070]** 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.

**[0071]** 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.

**[0072]** 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.

**[0073]** 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.

**[0074]** 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.

**[0075]** 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.

**[0076]** In some examples, motion estimation unit 204 may output a full set of motion information for decoding processing of a decoder.

**[0077]** 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.

**[0078]** 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.

**[0079]** 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.

**[0080]** 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.

**[0081]** 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.

**[0082]** 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.

**[0083]** 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.

**[0084]** 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.

**[0085]** 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.

**[0086]** 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.

**[0087]** After reconstruction unit 212 reconstructs the video block, loop filtering operation may be performed reduce video blocking artifacts in the video block.

**[0088]** 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.

**[0089]** Some embodiments of the disclosed technology include making a decision or determination to enable a video processing tool or mode. In an example, when the video processing tool or mode is enabled, the encoder will use or implement the tool or mode in the processing of a block of video, but may not necessarily modify the resulting bitstream based on the usage of the tool or mode. That is, a conversion from the block of video to the bitstream (or the bitstream representation) of the video will use the video processing tool or mode when it is enabled based on the decision or determination. In another example, when the video processing tool or mode is enabled, the decoder will process the bitstream with the knowledge that the bitstream has been modified based on the video processing tool or mode. That is, a conversion from the bitstream of the video to the block of video will be performed using the video processing tool or mode that was enabled based on the decision or determination.

**[0090]** FIG. 8 is a block diagram illustrating an example of video decoder 300 which may be video decoder 114 in the system 100 illustrated in FIG. 6.

**[0091]** 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.

**[0092]** In the example of FIG. 8, 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. 7).

**[0093]** 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.

**[0094]** 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.

**[0095]** Motion compensation unit 302 may use interpolation filters as used by video encoder 200 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.

**[0096]** 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.

**[0097]** 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.

**[0098]** 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.

**[0099]** A listing of solutions preferred by some embodiments is provided next.

**[00100]** The following solutions show example embodiments of techniques discussed in the previous section (e.g., item 1).

**[00101]** 1. A method of video processing (e.g., method 500 in FIG. 5), comprising: performing (502) a conversion between a video comprising video pictures and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies whether and where one or more syntax elements indicative of one or more initial quantization parameter (QP) values used during the conversion are included in the coded representation.

**[00102]** 2. The method of solution 1, wherein the format rule specifies that a syntax element indicating the one or more QP values is included in a picture header without including the syntax element in a picture parameter set.

**[00103]** 3. The method of solution 1, wherein the format rule specifies that a syntax element indicating the one or more QP values is included in a picture parameter set and repeated in one or more picture headers.

**[00104]** 4. The method of solution 1, wherein the format rule specifies that the one or more syntax elements are included in the coded representation at multiple levels comprising two or more of a sequence parameter set, a picture parameter set, a picture header and a sequence header according to a rule.

**[00105]** 5. The method of solution 4, wherein the rule is based on a video picture type of a video picture.

**[00106]** The following solutions show example embodiments of techniques discussed in the previous section (e.g., items 2-3).

**[00107]** 6. A method of video processing, comprising: determining to use, for a conversion between a video block of a video and a coded representation of the video, an initial quantization parameter (QP) value; and performing a conversion based on the determining, wherein a syntax element in the coded representation is indicative of the initial QP value using an offset relationship comprising an offset value  $K$  that is (a) between 1 and 25 or (b) greater than 26 or (c) a function of maximum allowed QP value for the conversion or (d) a function of bit depth used for the conversion.

**[00108]** 7. The method of solution 6, wherein  $K$  is a function of a type of the video block or a type of a video picture or a video slice containing the video block.

**[00109]** The following solutions show example embodiments of techniques discussed in the previous section (e.g., items 4-5).

**[00110]** 8. A video processing method, comprising: determining, for a conversion between a chroma block of a video and a coded representation of the video, one or more quantization parameter (QP) tables based on a type of a picture or a slice that contains the chroma block according to a rule; and performing the conversion based on the determining.

**[00111]** 9. The method of solution 8, wherein the rule specifies that the one or more QP tables are different from I, B and P video pictures.

**[00112]** 10. The method of solution 8, wherein the rule specifies that the one or more QP tables for I coded video pictures are different from the one or more QP tables for B and P coded video pictures.

**[00113]** 11. The method of any of solutions 8-10, wherein the rule is based on a classification of a picture type and a slice type to which the chroma block belongs, wherein a number M represents an index to the classification, where M is an integer greater than 1.

**[00114]** The following solutions show example embodiments of techniques discussed in the previous section (e.g., items 6-8).

**[00115]** 12. A video processing method, comprising: determining, for a conversion between a chroma block of a video and a coded representation of the video, one or more quantization parameter (QP) tables; and performing the conversion based on the determining; wherein the conversion conforms to a first rule that specifies one or more syntax elements corresponding to the one or more QP tables or a second rule that specifies a relationship between the one or more QP tables and a coding characteristic of the chroma block.

**[00116]** 13. The method of solution 12, wherein the first rule specifies that the one or more syntax elements are included in a sequence parameter set associated with the coded representation.

**[00117]** 14. The method of any of solutions 12-13, wherein the second rule specifies that the one or more QP tables are based on a prediction mode used for the conversion.

**[00118]** 15. The method of solution 13, wherein the second rule specifies that the one or more QP tables for intra coded chroma blocks are different from chroma blocks that use other prediction modes.

**[00119]** The following solutions show example embodiments of techniques discussed in the previous section (e.g., item 9-14).

**[00120]** 16. A video processing method, comprising: performing a conversion between a chroma block of a video and a coded representation of the video; wherein the coded representation conforms to a format rule, wherein the format rule specifies a characteristic of a syntax element associated with one or more chroma quantization parameter (QP) tables used for the conversion.

**[00121]** 17. The method of solution 16, wherein the format rule specifies a maximum value of a start point syntax element for the one or more chroma QP tables.

**[00122]** 18. The method of solution 16, wherein the format rule specifies that the start point syntax element has a value offset by a number K smaller than 26.

**[00123]** 19. The method of solution 16, wherein the format rule specifies that the start point syntax element is offset by a value that is dependent on whether a current picture of the chroma block is an intra-only picture.

**[00124]** 20. The method of any of solutions 16-19, wherein the format rule specifies that a number of pivot points in the one or more chroma QP tables is zero.

**[00125]** 21. The method of any of solutions 16-20, due to the number of pivot points being zero, an *i*th entry the one or more chroma QP tables is set equal to an *i*th entry in a corresponding luma QP table.

**[00126]** 22. The method of any of solutions 16-21, wherein the syntax element is included in both a sequence parameter set and a picture parameter set in the coded representation.

**[00127]** 23. The method of any of solutions 16-22, wherein the syntax element is included in either a picture header or a sequence header.

**[00128]** The following solutions show example embodiments of techniques discussed in the previous section (e.g., item 25).

**[00129]** 24. A video processing method, comprising: determining, a signaling scheme used for signaling a quantization parameter table in a coded representation of a video based on a format rule related to one or more syntax elements in the coded representation, and performing, based on the determining, a conversion between the video and the coded representation.

**[00130]** 25. The method of solution 24, wherein the signaling scheme comprises indication of a number of points in the quantization parameter table.

**[00131]** The following solutions show example embodiments of techniques discussed in the previous section (e.g., item 26).

**[00132]** 26. A video processing method, comprising: performing a conversion between a video block of a video and a coded representation of the video according to a rule, wherein the rule specifies a number  $N$  that, in case that use of affine coding is enabled and use of sub-block based temporal motion vector prediction disabled, is used to limit a maximum number of subblock merge candidates.

**[00133]** 27. The method of solution 26, wherein  $N=0$ .

**[00134]** The following solutions show example embodiments of techniques discussed in the previous section (e.g., item 27-29).

**[00135]** 28. A video processing method, comprising: performing a conversion between a video comprising one or more video pictures comprising one or more video subpictures and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to include or omit one or more of the following subpicture related syntax elements: a first syntax element indicating whether all subpictures referring to a same sequence parameter set have a same dimension, a second syntax element that signals a common width and a common height of all subpictures having the same dimension, or a third syntax element according to a second format rule.

**[00136]** 29. The method of solution 28, wherein the second format rule specifies to omit syntax elements indicative of sample width under certain conditions.

**[00137]** 30. The method of any of solutions 1-29, wherein the conversion comprises encoding the video into the coded representation.

**[00138]** 31. The method of any of solutions 1-29, wherein the conversion comprises decoding the coded representation to generate pixel values of the video.

**[00139]** 32. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of solutions 1 to 31.

**[00140]** 33. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of solutions 1 to 31.

**[00141]** 34. 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 1 to 31.

**[00142]** FIG. 9 is a flowchart for an example method 900 of video processing. Operation 902 includes performing a conversion between a video and a bitstream of the video according to a

rule, wherein the rule specifies that a quantization parameter table used in the conversion is included in a sequence parameter set, and wherein the rule specifies whether or how a first syntax element indicating a number of points in the quantization parameter table is included in the sequence parameter set is controlled according to a second syntax element in the sequence parameter set.

**[00143]** In some embodiments of method 900, the quantization parameter table is a chroma quantization parameter mapping table. In some embodiments of method 900, a value of the first syntax element plus 1 is equal to the number of points in the quantization parameter table. In some embodiments of method 900, a value of the second syntax element indicates a starting luma and chroma quantization parameter used to describe the quantization parameter table. In some embodiments of method 900, a maximum value of the number of points in the quantization parameter table is based on the value of the second syntax element. In some embodiments of method 900, the rule specifies that a maximum value of the number of points in the quantization parameter table minus 1 is a difference between a maximum quantization parameter value and a starting luma and chroma quantization parameter used to describe an i-th chroma quantization parameter mapping table.

**[00144]** In some embodiments of method 900, the maximum value is set to  $(63 - (qp\_table\_start\_minus26[i] + 26))$ . In some embodiments of method 900, the maximum value is set to  $(64 - (qp\_table\_start\_minus26[i] + 26))$ .

**[00145]** FIG. 10 is a flowchart for an example method 1000 of video processing. Operation 1002 includes performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that one of the following two conditions is true when a number of subpictures minus 1 indicated in a sequence parameter set is greater than zero and when a syntax element in the sequence parameter set indicates that an i-th subpicture of a first video picture of the video in a coded layer video sequence is treated as a second video picture of the video in a decoding process that excludes an in-loop filtering operation: (1) the second video picture referred to by each active entry in a first reference picture list or a second reference picture list of the first video picture and the first video picture have a same value for each of the following: a picture width in units of luma samples, a picture height in units of luma samples, the number of subpictures minus 1, and a horizontal position of a top-left coding tree unit of the i-th subpicture, a vertical position of the top-left CTU of the i-th subpicture, a width of the i-th

subpicture minus 1, a height of the  $i$ -th subpicture minus 1, the syntax element for the  $i$ -th subpicture, respectively, for each value of  $i$  in a range of 0 to the number of subpictures minus 1, inclusive; and (2) the second video picture referred to by each active entry in the first reference picture list or the second reference picture list is an inter-layer reference picture for which a value of the number of subpictures minus 1 is equal to zero.

**[00146]** FIG. 11 is a flowchart for an example method 1100 of video processing. Operation 1102 includes performing a conversion between a video comprising a video block and a bitstream of the video according to a rule, wherein the rule specifies that a maximum number of subblock merge candidates is disallowed from being a number  $N$  when affine coding tool is enabled and sub-block based temporal motion vector prediction coding tool is disabled.

**[00147]** In some embodiments of method 1100,  $N=0$ . In some embodiments of method 1100,  $N$  is greater than 0. In some embodiments of method 1100, a range of a maximum number of subblock-based merging motion vector prediction candidates supported in a sequence parameter set minus five is from  $[0, 4]$  regardless of a value of a syntax element in the sequence parameter set that indicates whether the SbTMVP coding tool is enabled or disabled. In some embodiments of method 1100, the bitstream is a conformance bitstream that includes a value of a maximum number of subblock-based merging motion vector prediction candidates from within a range of 1 to 5, inclusive, when the affine coding tool is enabled. In some embodiments of method 1100, the bitstream is a conformance bitstream that includes a value of a maximum number of subblock-based merging motion vector prediction candidates from within a range of 0 to 1, inclusive, when the affine coding tool is disabled. In some embodiments of method 1100, whether the bitstream includes a syntax element that indicates whether a merge subblock operation is enabled is based on whether the affine coding tool is enabled and is not based on the maximum number of subblock merge candidates.

**[00148]** FIG. 12 is a flowchart for an example method 1200 of video processing. Operation 1202 includes performing a conversion between a video comprising one or more video pictures and a bitstream of the video, wherein the bitstream conforms to a format rule, and wherein the format rule specifies whether a syntax element that indicates whether affine merge mode is enabled is in the bitstream.

**[00149]** In some embodiments of method 1200, the format rule specifies that the syntax element is selectively indicated in the bitstream based on whether the affine merge mode is enabled in a

sequence parameter set. In some embodiments of method 1200, the format rule specifies that a maximum number of subblock merge candidates is indicated in the bitstream when the syntax element indicates that the affine merge mode is enabled.

**[00150]** FIG. 13 is a flowchart for an example method 1300 of video processing. Operation 1302 includes performing a conversion between a video comprising one or more video pictures comprising one or more subpictures and a bitstream of the video according to a rule, wherein the rule specifies to indicate or omit one or more of the following subpicture related syntax elements from a sequence parameter set: a first syntax element indicating whether all subpictures referring to a same sequence parameter set have a same dimension, a second syntax element that indicates a common width and a common height of all subpictures having the same dimension, or one or more syntax elements according to a second rule are omitted when all subpictures do not have the same dimension.

**[00151]** In some embodiments of method 1300, the rule specifies that: a number of subpicture columns, denoted as numSubpicCols, is derived to be equal to

$(\text{sps\_pic\_width\_max\_in\_luma\_samples} / ((\text{sps\_uniform\_subpic\_width\_minus1} + 1) * \text{CtbSizeY}))$ ,

a number of subpicture row, denoted as numSubpicRows, is derived to be equal

to  $(\text{sps\_pic\_height\_max\_in\_luma\_samples} / ((\text{sps\_uniform\_subpic\_height\_minus1} + 1) * \text{CtbSizeY}))$ ,

and wherein a number of subpictures, denoted as sumSubpics, is derived to be equal to numSubpicCols \* numSubpicRows. In some embodiments of method 1300, the rule

specifies that sps\_num\_subpics\_minus1 is excluded from the sequence parameter set, and

wherein a value of the sps\_num\_subpics\_minus1 is inferred to equal to numSubpics - 1. In

some embodiments of method 1300, the rule specifies that for each value of i in a range of 0 to sps\_num\_subpics\_minus1, inclusive, values for sps\_subpic\_ctu\_top\_left\_x[i],

sps\_subpic\_ctu\_top\_left\_y[i], sps\_subpic\_width\_minus1[i], and

sps\_subpic\_height\_minus1[i] are inferred to be equal to  $(\text{sps\_uniform\_subpic\_width\_minus1} + 1) * (i \% \text{numSubpicCols})$ ,

$(\text{sps\_uniform\_subpic\_height\_minus1} + 1) * (i / \text{numSubpicCols})$ ,

sps\_uniform\_subpic\_width\_minus1, and sps\_uniform\_subpic\_height\_minus1, respectively

**[00152]** In some embodiments of method 1300, the rule specifies that for each value of i in a

range of 0 to sps\_num\_subpics\_minus1, inclusive, when sps\_independent\_subpics\_flag is equal

to 0, sps\_subpic\_treated\_as\_pic\_flag[i] and sps\_loop\_filter\_across\_subpic\_enabled\_flag[i] are

indicated in the sequence parameter set. In some embodiments of method 1300, the second rule

specifies to use a current subpictures layer signaling. In some embodiments of method 1300, the rule specifies that when `sps_pic_width_max_in_luma_samples` is less than or equal to `CtbSizeY`, `sps_uniform_subpic_width_minus1` is excluded from the sequence parameter set and is inferred to be equal to 0. In some embodiments of method 1300, the rule specifies that when `sps_pic_height_max_in_luma_samples` is less than or equal to `CtbSizeY`, `sps_uniform_subpic_height_minus1` is excluded from the sequence parameter set and is inferred to be equal to 0.

**[00153]** In some embodiments of method 1300, the rule specifies that `sps_uniform_subpic_width_minus1`, `sps_uniform_subpic_height_minus1`, `sps_subpic_ctu_top_left_x[i]`, `sps_subpic_ctu_top_left_y[i]`, `sps_subpic_width_minus1[i]`, and `sps_subpic_height_minus1[i]` are `ue(v)`-coded. In some embodiments of method 1300, the rule specifies that `sps_uniform_subpic_width_minus1` and `sps_uniform_subpic_height_minus1` are `u(v)`-coded with a same length as `sps_subpic_width_minus1[i]` and `sps_subpic_height_minus1[i]`, respectively. In some embodiments of method 1300, the sequence parameter set includes: a first syntax element that indicates whether all subpictures of the video have the same width and height, respectively, a second syntax element that indicates a common width for all subpictures, and a third syntax element that indicates a common height for all subpictures. In some embodiments of method 1300, the rule specifies that one of the following two conditions is true when a number of subpictures minus 1 indicated in a sequence parameter set is greater than zero and when a syntax element in the sequence parameter set indicates that an *i*-th subpicture of a first video picture of the video in a coded layer video sequence is treated as a second video picture of the video in a decoding process that excludes an in-loop filtering operation: (1) the second video picture referred to by each active entry in a first reference picture list or a second reference picture list of the first video picture and the first video picture have a same value for each of the following: a picture width in units of luma samples, a picture height in units of luma samples, the number of subpictures minus 1, and a horizontal position of a top-left coding tree unit of the *i*-th subpicture, a vertical position of the top-left CTU of the *i*-th subpicture, a width of the *i*-th subpicture minus 1, a height of the *i*-th subpicture minus 1, the syntax element for the *i*-th subpicture, and an identifier value for the *i*-th subpicture respectively, for each value of *i* in a range of 0 to the number of subpictures minus 1, inclusive; and (2) the second video picture referred to by each active entry in the first reference picture list

or the second reference picture list is an inter-layer reference picture for which a value of the number of subpictures minus 1 is equal to zero

**[00154]** FIG. 14 is a flowchart for an example method 1400 of video processing. Operation 1402 includes determining, for a conversion between a video comprising one or more video pictures and a bitstream of the video, that a first syntax element in a picture parameter set that indicates a width of an  $i$ -th tile column in units of coding tree blocks minus 1 is equal to a width of a video picture in units of coding tree blocks minus 1 in response to a width of each video picture referring to the picture parameter set in units of luma samples being less than or equal to the luma coding tree block size. Operation 1404 includes performing the conversion based on the determining.

**[00155]** FIG. 15 is a flowchart for an example method 1500 of video processing. Operation 1502 includes determining, for a conversion between a video comprising one or more video pictures and a bitstream of the video, that a first syntax element in a picture parameter set that indicates a height of an  $i$ -th tile row in units of coding tree blocks minus 1 is equal to a height of a video picture in units of coding tree blocks minus 1 in response to a height of each video picture referring to the picture parameter set in units of luma samples being less than or equal to a luma coding tree block size. Operation 1504 includes performing the conversion based on the determining.

**[00156]** In some embodiments of method(s) 900-1500, the performing the conversion comprising encoding the video into the bitstream. In some embodiments of method(s) 900-1500, the performing the conversion comprises generating the bitstream from the video, and the method further comprises storing the bitstream in a non-transitory computer-readable recording medium. In some embodiments of method(s) 900-1500, the performing the conversion comprises decoding the video from the bitstream.

**[00157]** In some embodiments, a video decoding apparatus comprising a processor configured to implement a method recited in one or more embodiments of method(s) 900-1500. In some embodiments, a video encoding apparatus comprising a processor configured to implement a method recited in one or more embodiments of method(s) 900-1500. In some embodiments, a computer program product having computer instructions stored thereon, the instructions, when executed by a processor, causes the processor to implement a method recited in one or more embodiments of method(s) 900-1500. In some embodiments, a non-transitory computer-

readable storage medium that stores a bitstream generated according to a method recited in one or more embodiments of method(s) 900-1500. In some embodiments, a non-transitory computer-readable storage medium storing instructions that cause a processor to implement a method recited in one or more embodiments of method(s) 900-1500. In some embodiments, a method of bitstream generation, comprising: generating a bitstream of a video according to a method recited in one or more embodiments of method(s) 900-1500, and storing the bitstream on a computer-readable program medium. In some embodiments, a method, an apparatus, a bitstream generated according to a disclosed method or a system described in the present document.

**[00158]** In the present document, the term “video processing” may refer to video encoding, video decoding, video compression or video decompression. For example, video compression algorithms may be applied during conversion from pixel representation of a video to a corresponding bitstream representation or vice versa. The bitstream representation of a current video block may, for example, correspond to bits that are either co-located or spread in different places within the bitstream, as is defined by the syntax. For example, a macroblock may be encoded in terms of transformed and coded error residual values and also using bits in headers and other fields in the bitstream. Furthermore, during conversion, a decoder may parse a bitstream with the knowledge that some fields may be present, or absent, based on the determination, as is described in the above solutions. Similarly, an encoder may determine that certain syntax fields are or are not to be included and generate the coded representation accordingly by including or excluding the syntax fields from the coded representation.

**[00159]** 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.

**[00160]** 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.

**[00161]** 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).

**[00162]** 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.

**[00163]** 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.

**[00164]** 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.

**[00165]** 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.

## CLAIMS

1. A method of processing video data, comprising:  
performing a conversion between a video and a bitstream of the video according to a rule,  
wherein the rule specifies that a quantization parameter table used in the conversion is included in a sequence parameter set, and  
wherein the rule specifies whether or how a first syntax element indicating a number of points in the quantization parameter table is included in the sequence parameter set is controlled according to a second syntax element in the sequence parameter set.
2. The method of claim 1, wherein the quantization parameter table is a chroma quantization parameter mapping table.
3. The method of claim 1, wherein a value of the first syntax element plus 1 is equal to the number of points in the quantization parameter table.
4. The method of claim 1, wherein a value of the second syntax element indicates a starting luma and chroma quantization parameter used to describe the quantization parameter table.
5. The method of claim 1, wherein a maximum value of the number of points in the quantization parameter table is based on the value of the second syntax element.
6. The method of claim 1, wherein the rule specifies that a maximum value of the number of points in the quantization parameter table minus 1 is a difference between a maximum quantization parameter value and a starting luma and chroma quantization parameter used to describe an i-th chroma quantization parameter mapping table.
7. The method of claim 6, wherein the maximum value is set to  $(63 - (qp\_table\_start\_minus26[i] + 26))$ .
8. The method of claim 6, wherein the maximum value is set to  $(64 - (qp\_table\_start\_minus26[i] + 26))$ .

9. A method of processing video data, comprising:  
performing a conversion between a video and a bitstream of the video according to a rule,  
wherein the rule specifies that one of the following two conditions is true when a number of subpictures minus 1 indicated in a sequence parameter set is greater than zero and when a syntax element in the sequence parameter set indicates that an  $i$ -th subpicture of a first video picture of the video in a coded layer video sequence is treated as a second video picture of the video in a decoding process that excludes an in-loop filtering operation:

(1) the second video picture referred to by each active entry in a first reference picture list or a second reference picture list of the first video picture and the first video picture have a same value for each of the following:

a picture width in units of luma samples,  
a picture height in units of luma samples,  
the number of subpictures minus 1, and

a horizontal position of a top-left coding tree unit of the  $i$ -th subpicture, a vertical position of the top-left CTU of the  $i$ -th subpicture, a width of the  $i$ -th subpicture minus 1, a height of the  $i$ -th subpicture minus 1, the syntax element for the  $i$ -th subpicture, respectively, for each value of  $i$  in a range of 0 to the number of subpictures minus 1, inclusive; and

(2) the second video picture referred to by each active entry in the first reference picture list or the second reference picture list is an inter-layer reference picture for which a value of the number of subpictures minus 1 is equal to zero.

10. A method of processing video data, comprising:  
performing a conversion between a video comprising a video block and a bitstream of the video according to a rule,

wherein the rule specifies that a maximum number of subblock merge candidates is disallowed from being a number  $N$  when affine coding tool is enabled and sub-block based temporal motion vector prediction coding tool is disabled.

11. The method of claim 10, wherein  $N=0$ .

12. The method of claim 10, wherein N is greater than 0.
13. The method of claim 10, wherein a range of a maximum number of subblock-based merging motion vector prediction candidates supported in a sequence parameter set minus five is from [0, 4] regardless of a value of a syntax element in the sequence parameter set that indicates whether the SbTMVP coding tool is enabled or disabled.
14. The method of claim 10, wherein the bitstream is a conformance bitstream that includes a value of a maximum number of subblock-based merging motion vector prediction candidates from within a range of 1 to 5, inclusive, when the affine coding tool is enabled.
15. The method of claim 10, wherein the bitstream is a conformance bitstream that includes a value of a maximum number of subblock-based merging motion vector prediction candidates from within a range of 0 to 1, inclusive, when the affine coding tool is disabled.
16. The method of claim 10, wherein whether the bitstream includes a syntax element that indicates whether a merge subblock operation is enabled is based on whether the affine coding tool is enabled and is not based on the maximum number of subblock merge candidates.
17. A method of processing video data, comprising:
  - performing a conversion between a video comprising one or more video pictures and a bitstream of the video,
  - wherein the bitstream conforms to a format rule, and
  - wherein the format rule specifies whether a syntax element that indicates whether affine merge mode is enabled is in the bitstream.
18. The method of claim 17, wherein the format rule specifies that the syntax element is selectively indicated in the bitstream based on whether the affine merge mode is enabled in a sequence parameter set.

19. The method of claim 17, wherein the format rule specifies that a maximum number of subblock merge candidates is indicated in the bitstream when the syntax element indicates that the affine merge mode is enabled.

20. A method of processing video data, comprising:

performing a conversion between a video comprising one or more video pictures comprising one or more subpictures and a bitstream of the video according to a rule,

wherein the rule specifies to indicate or omit one or more of the following subpicture related syntax elements from a sequence parameter set:

a first syntax element indicating whether all subpictures referring to a same sequence parameter set have a same dimension,

a second syntax element that indicates a common width and a common height of all subpictures having the same dimension, or

one or more syntax elements according to a second rule are omitted when all subpictures do not have the same dimension.

21. The method of claim 20, wherein the rule specifies that:

a number of subpicture columns, denoted as numSubpicCols, is derived to be equal to  $( \text{sps\_pic\_width\_max\_in\_luma\_samples} / ( ( \text{sps\_uniform\_subpic\_width\_minus1} + 1 ) * \text{CtbSizeY} ) )$ ,

a number of subpicture row, denoted as numSubpicRows, is derived to be equal to  $( \text{sps\_pic\_height\_max\_in\_luma\_samples} / ( ( \text{sps\_uniform\_subpic\_height\_minus1} + 1 ) * \text{CtbSizeY} ) )$ , and

wherein a number of subpictures, denoted as sumSubpics, is derived to be equal to  $\text{numSubpicCols} * \text{numSubpicRows}$ .

22. The method of claim 20, wherein the rule specifies that  $\text{sps\_num\_subpics\_minus1}$  is excluded from the sequence parameter set, and wherein a value of the  $\text{sps\_num\_subpics\_minus1}$  is inferred to equal to  $\text{numSubpics} - 1$ .

23. The method of claim 20, wherein the rule specifies that for each value of  $i$  in a range of 0 to  $\text{sps\_num\_subpics\_minus1}$ , inclusive, values for  $\text{sps\_subpic\_ctu\_top\_left\_x}[i]$ ,  $\text{sps\_subpic\_ctu\_top\_left\_y}[i]$ ,  $\text{sps\_subpic\_width\_minus1}[i]$ , and  $\text{sps\_subpic\_height\_minus1}[i]$  are inferred to be equal to  $(\text{sps\_uniform\_subpic\_width\_minus1} + 1) * (i \% \text{numSubpicCols})$ ,  $(\text{sps\_uniform\_subpic\_height\_minus1} + 1) * (i / \text{numSubpicCols})$ ,  $\text{sps\_uniform\_subpic\_width\_minus1}$ , and  $\text{sps\_uniform\_subpic\_height\_minus1}$ , respectively.

24. The method of claim 20, wherein the rule specifies that for each value of  $i$  in a range of 0 to  $\text{sps\_num\_subpics\_minus1}$ , inclusive, when  $\text{sps\_independent\_subpics\_flag}$  is equal to 0,  $\text{sps\_subpic\_treated\_as\_pic\_flag}[i]$  and  $\text{sps\_loop\_filter\_across\_subpic\_enabled\_flag}[i]$  are indicated in the sequence parameter set.

25. The method of claim 20, wherein the second rule specifies to use a current subpictures layer signaling.

26. The method of claim 20, wherein the rule specifies that when  $\text{sps\_pic\_width\_max\_in\_luma\_samples}$  is less than or equal to  $\text{CtbSizeY}$ ,  $\text{sps\_uniform\_subpic\_width\_minus1}$  is excluded from the sequence parameter set and is inferred to be equal to 0.

27. The method of claim 20, wherein the rule specifies that when  $\text{sps\_pic\_height\_max\_in\_luma\_samples}$  is less than or equal to  $\text{CtbSizeY}$ ,  $\text{sps\_uniform\_subpic\_height\_minus1}$  is excluded from the sequence parameter set and is inferred to be equal to 0.

28. The method of claim 20, wherein the rule specifies that  $\text{sps\_uniform\_subpic\_width\_minus1}$ ,  $\text{sps\_uniform\_subpic\_height\_minus1}$ ,  $\text{sps\_subpic\_ctu\_top\_left\_x}[i]$ ,  $\text{sps\_subpic\_ctu\_top\_left\_y}[i]$ ,  $\text{sps\_subpic\_width\_minus1}[i]$ , and  $\text{sps\_subpic\_height\_minus1}[i]$  are  $\text{ue}(v)$ -coded.

29. The method of claim 20, wherein the rule specifies that `sps_uniform_subpic_width_minus1` and `sps_uniform_subpic_height_minus1` are u(v)-coded with a same length as `sps_subpic_width_minus1[ i ]` and `sps_subpic_height_minus1[ i ]`, respectively.

30. The method of claim 20, wherein the sequence parameter set includes:  
a first syntax element that indicates whether all subpictures of the video have the same width and height, respectively,  
a second syntax element that indicates a common width for all subpictures, and  
a third syntax element that indicates a common height for all subpictures.

31. The method of claim 20,  
wherein the rule specifies that one of the following two conditions is true when a number of subpictures minus 1 indicated in a sequence parameter set is greater than zero and when a syntax element in the sequence parameter set indicates that an i-th subpicture of a first video picture of the video in a coded layer video sequence is treated as a second video picture of the video in a decoding process that excludes an in-loop filtering operation:

(1) the second video picture referred to by each active entry in a first reference picture list or a second reference picture list of the first video picture and the first video picture have a same value for each of the following:

- a picture width in units of luma samples,
- a picture height in units of luma samples,
- the number of subpictures minus 1, and

a horizontal position of a top-left coding tree unit of the i-th subpicture, a vertical position of the top-left CTU of the i-th subpicture, a width of the i-th subpicture minus 1, a height of the i-th subpicture minus 1, the syntax element for the i-th subpicture, and an identifier value for the i-th subpicture respectively, for each value of i in a range of 0 to the number of subpictures minus 1, inclusive; and

(2) the second video picture referred to by each active entry in the first reference picture list or the second reference picture list is an inter-layer reference picture for which a value of the number of subpictures minus 1 is equal to zero.

32. A method of processing video data, comprising:  
determining, for a conversion between a video comprising one or more video pictures and a bitstream of the video, that a first syntax element in a picture parameter set that indicates a width of an  $i$ -th tile column in units of coding tree blocks minus 1 is equal to a width of a video picture in units of coding tree blocks minus 1 in response to a width of each video picture referring to the picture parameter set in units of luma samples being less than or equal to the luma coding tree block size; and  
performing the conversion based on the determining.
33. A method of processing video data, comprising:  
determining, for a conversion between a video comprising one or more video pictures and a bitstream of the video, that a first syntax element in a picture parameter set that indicates a height of an  $i$ -th tile row in units of coding tree blocks minus 1 is equal to a height of a video picture in units of coding tree blocks minus 1 in response to a height of each video picture referring to the picture parameter set in units of luma samples being less than or equal to a luma coding tree block size; and  
performing the conversion based on the determining.
34. The method of any of claims 1 to 33, wherein the performing the conversion comprising encoding the video into the bitstream.
35. The method of any of claims 1 to 33, wherein the performing the conversion comprises generating the bitstream from the video, and the method further comprises storing the bitstream in a non-transitory computer-readable recording medium.
36. The method of any of claims 1 to 33, wherein the performing the conversion comprises decoding the video from the bitstream.
37. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of claims 1 to 36.

38. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of claims 1 to 36.
39. A computer program product having computer instructions stored thereon, the instructions, when executed by a processor, causes the processor to implement a method recited in any of claims 1 to 36.
40. A non-transitory computer-readable storage medium that stores a bitstream generated according to the method in any one of claims 1 to 36.
41. A non-transitory computer-readable storage medium storing instructions that cause a processor to implement a method recited in any of claims 1 to 36.
42. A method of bitstream generation, comprising:  
generating a bitstream of a video according to a method recited in any of claims 1 to 36, and  
storing the bitstream on a computer-readable program medium.
43. A method, an apparatus, a bitstream generated according to a disclosed method or a system described in the present document.

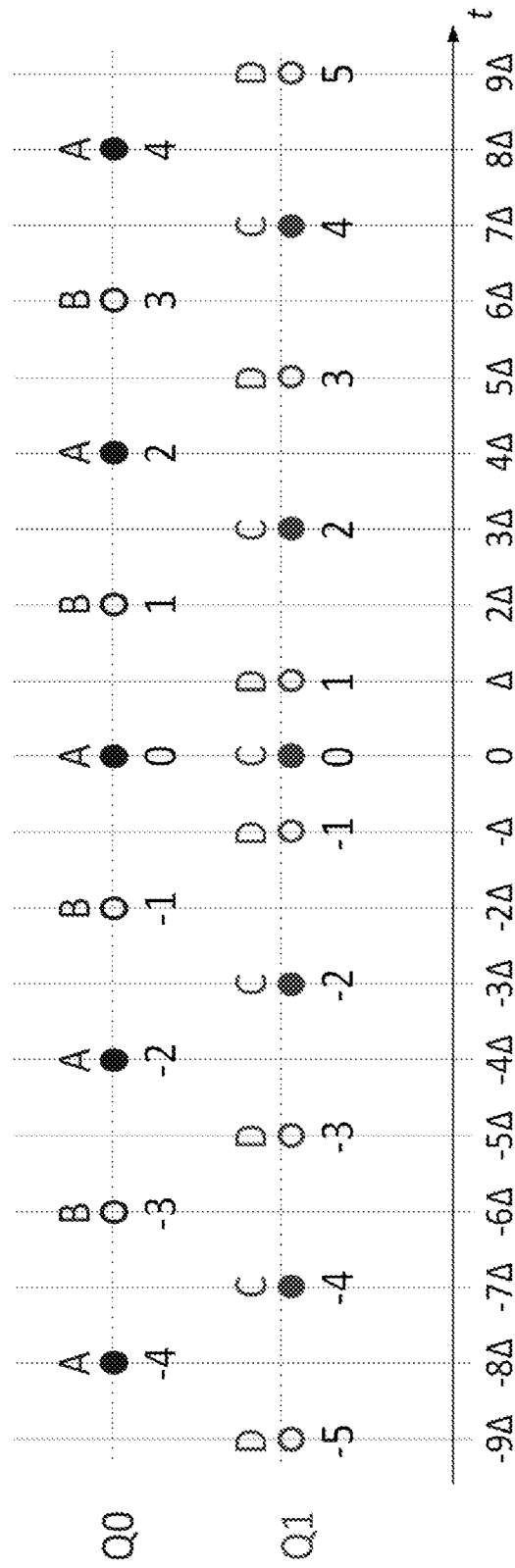


FIG. 1

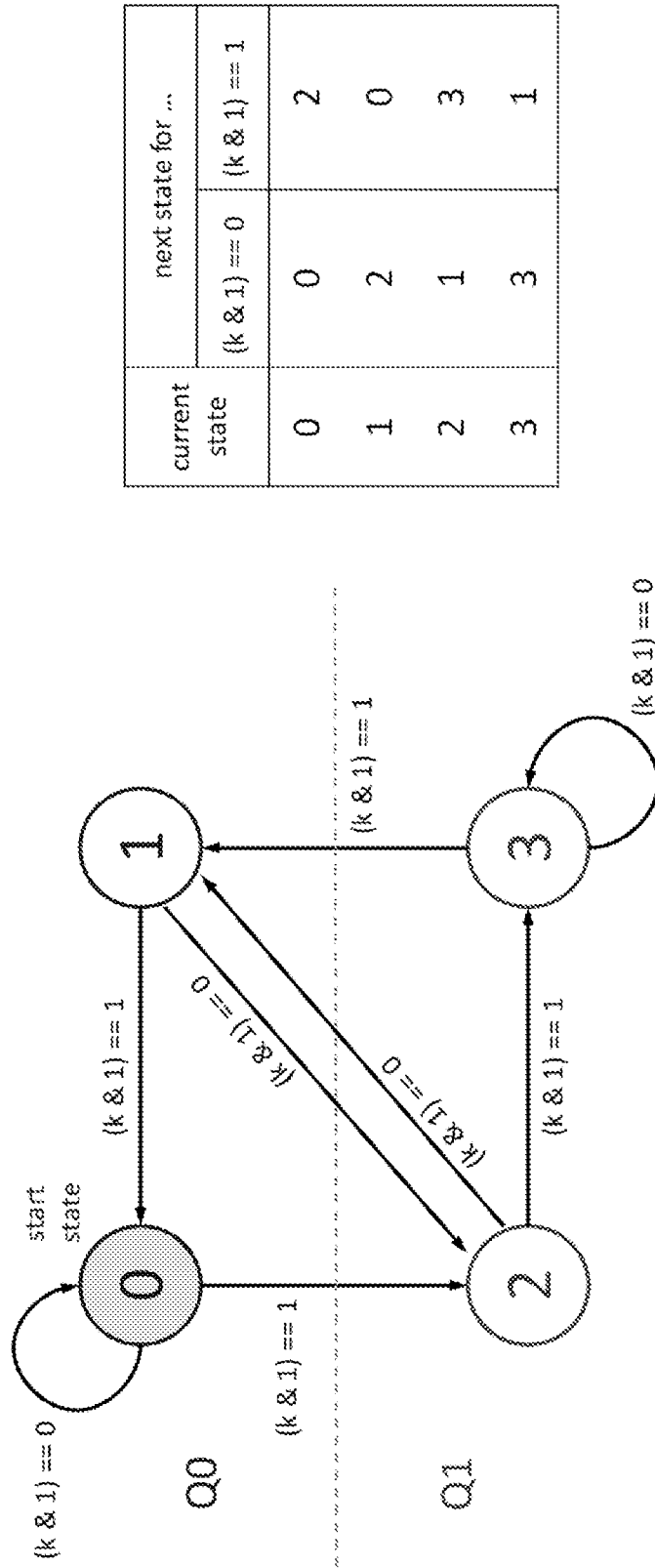
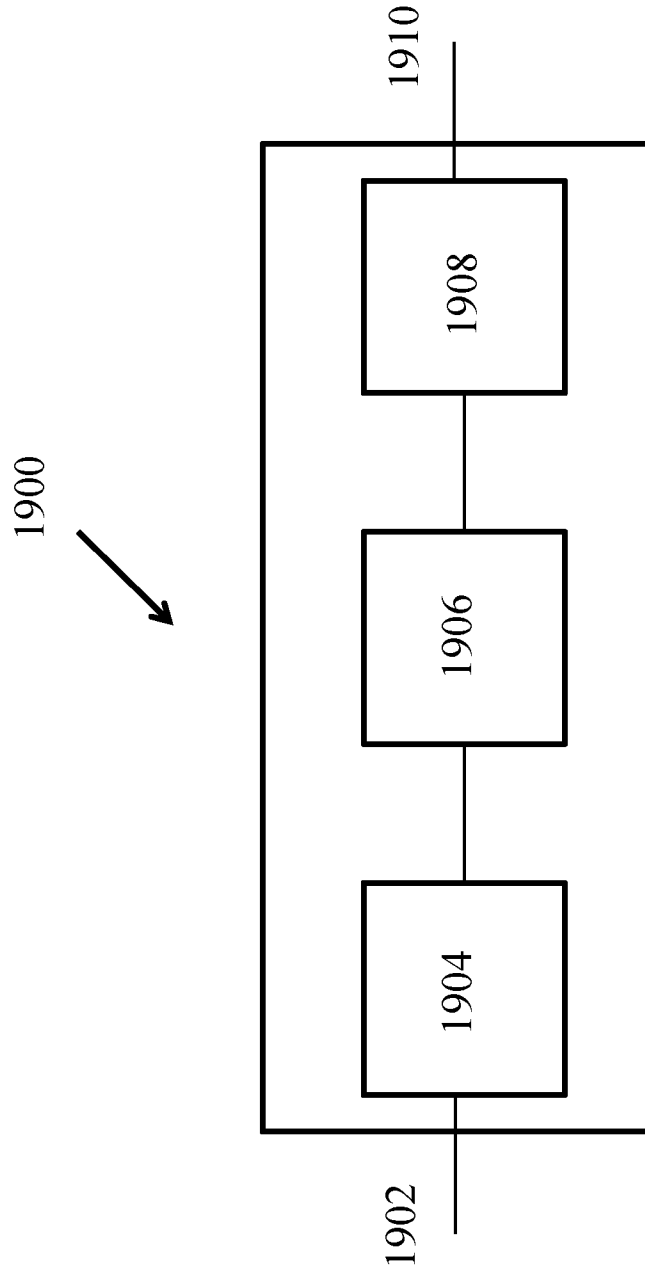
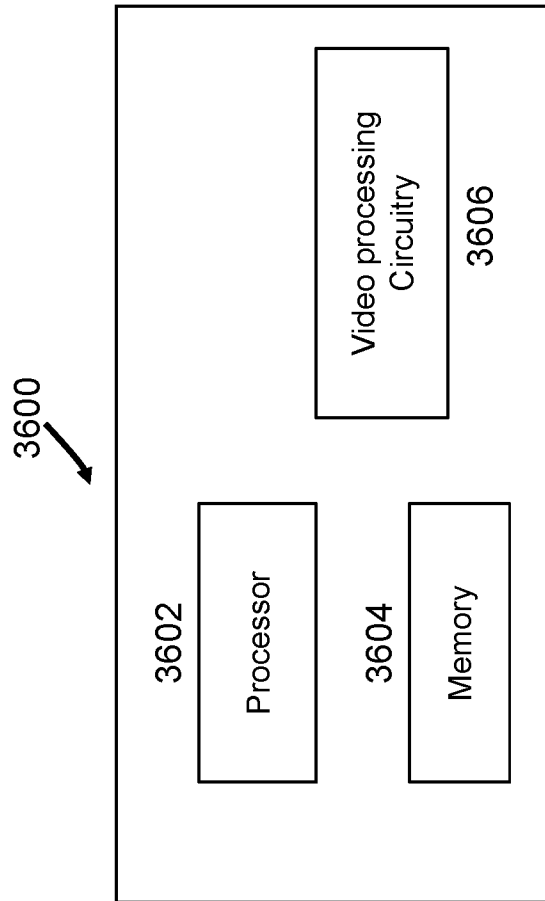


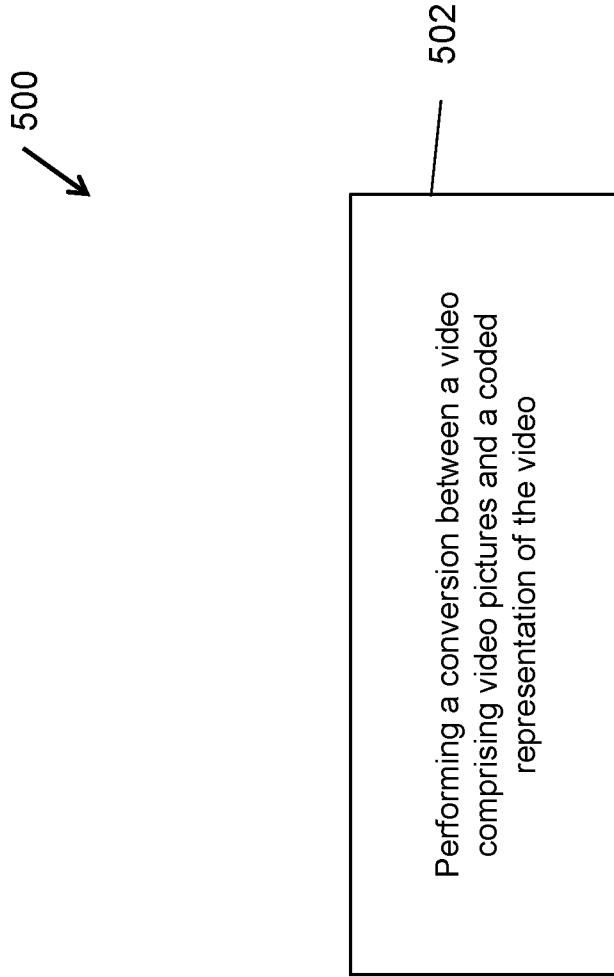
FIG. 2



**FIG. 3**



**FIG. 4**



**FIG. 5**

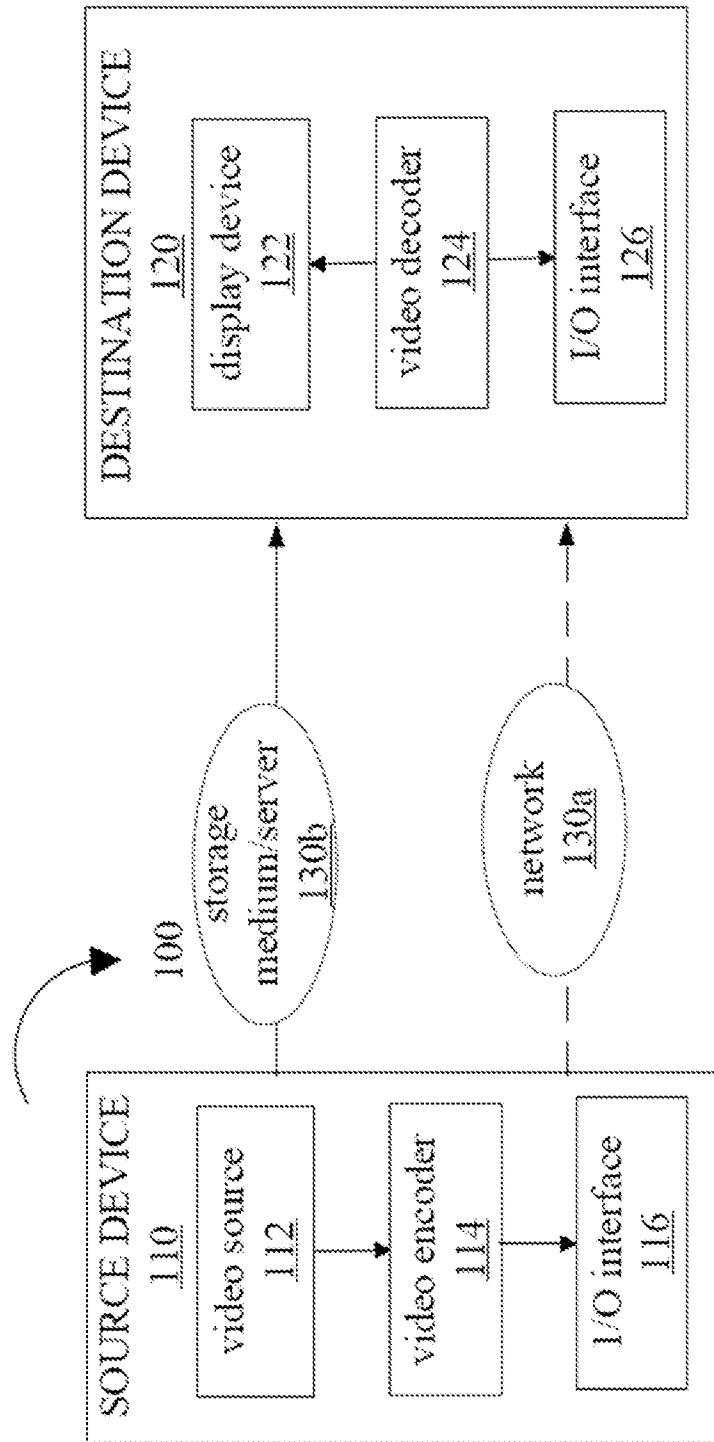


FIG. 6

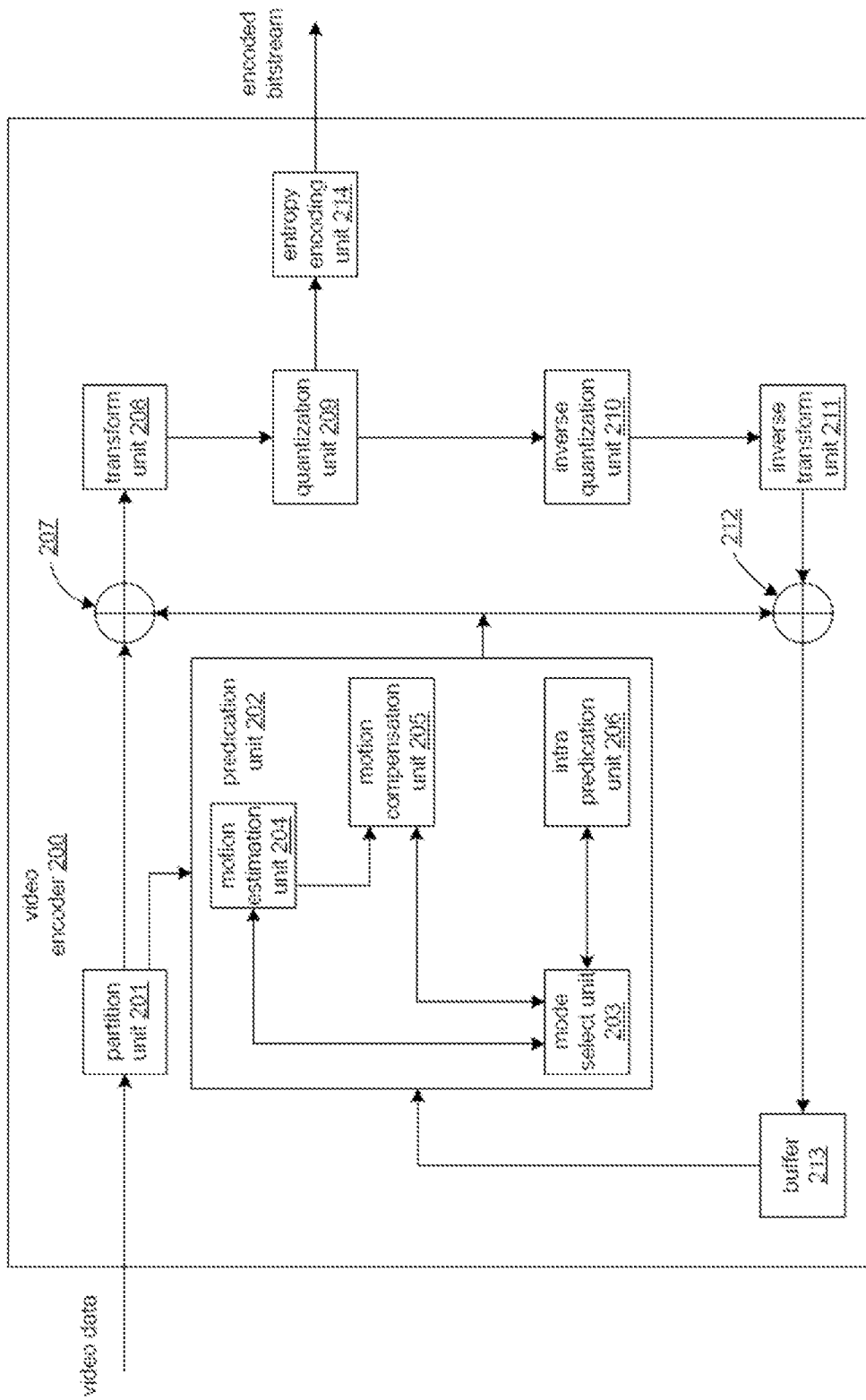


FIG. 7

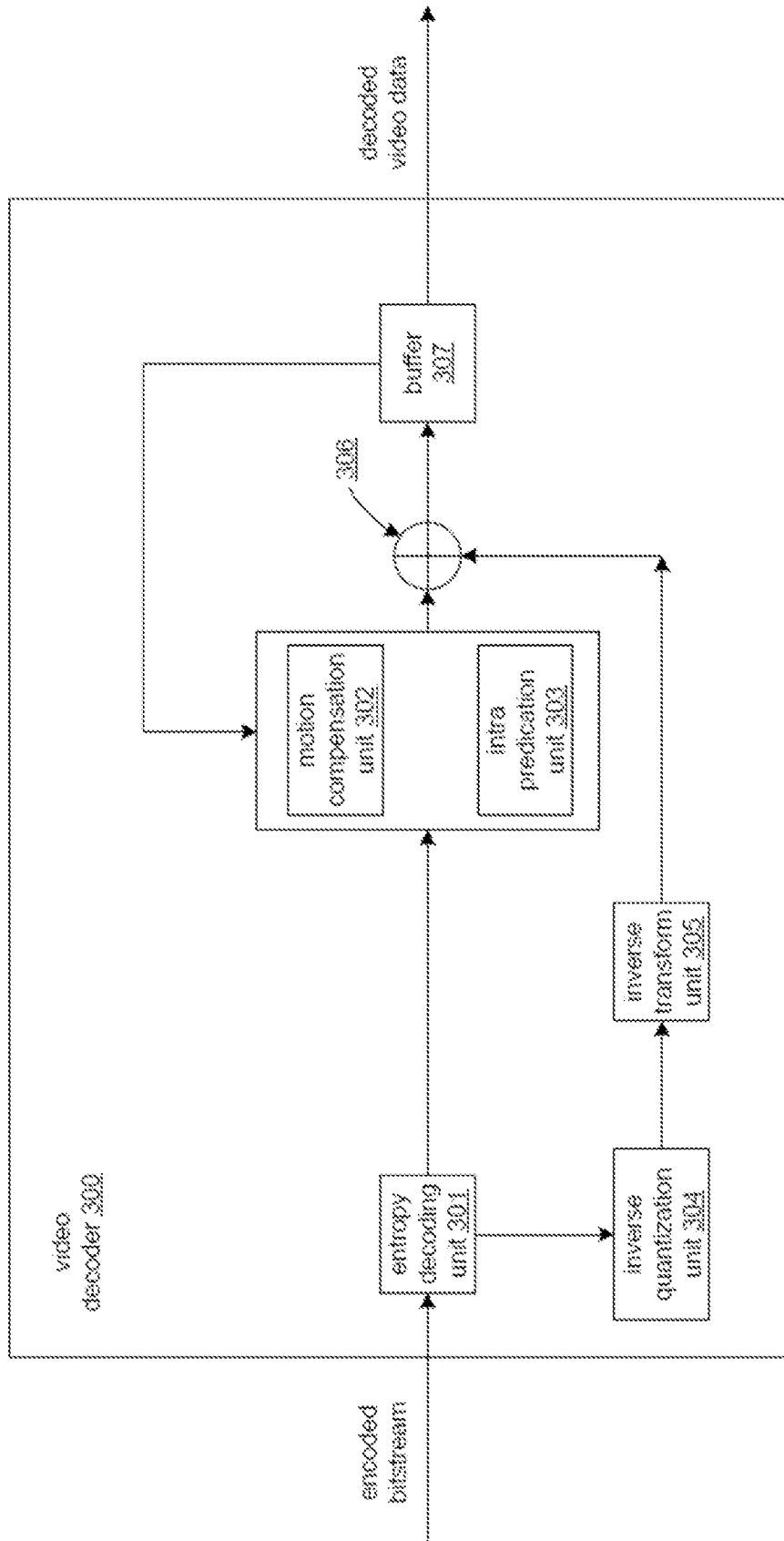
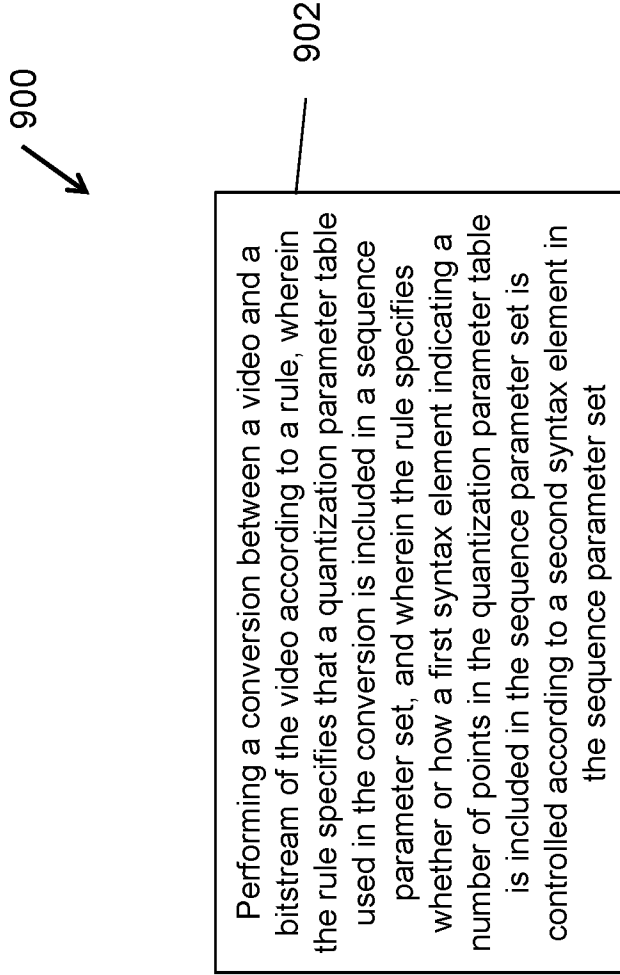


FIG. 8



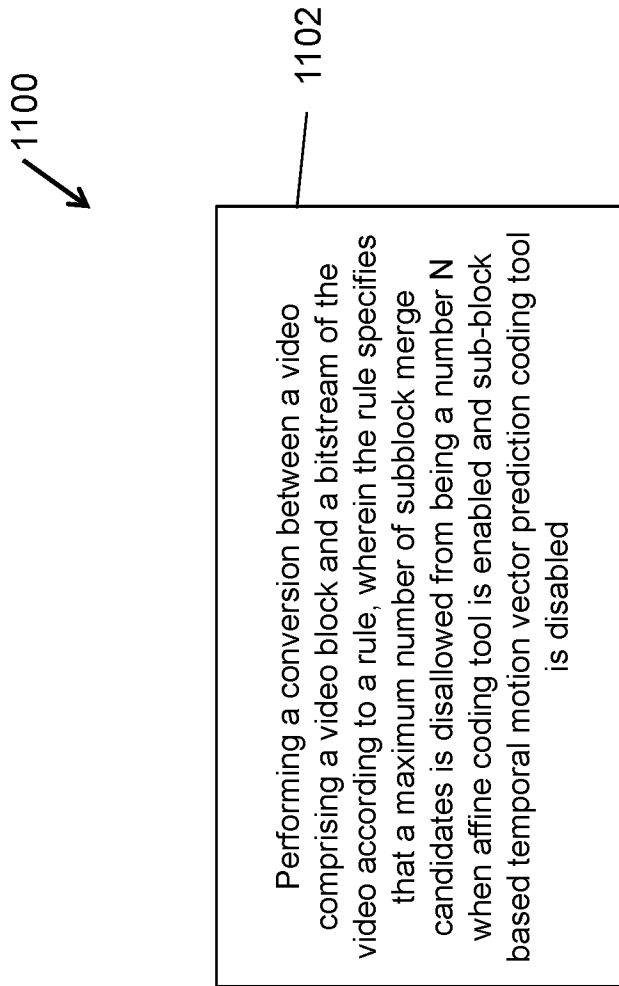
**FIG. 9**

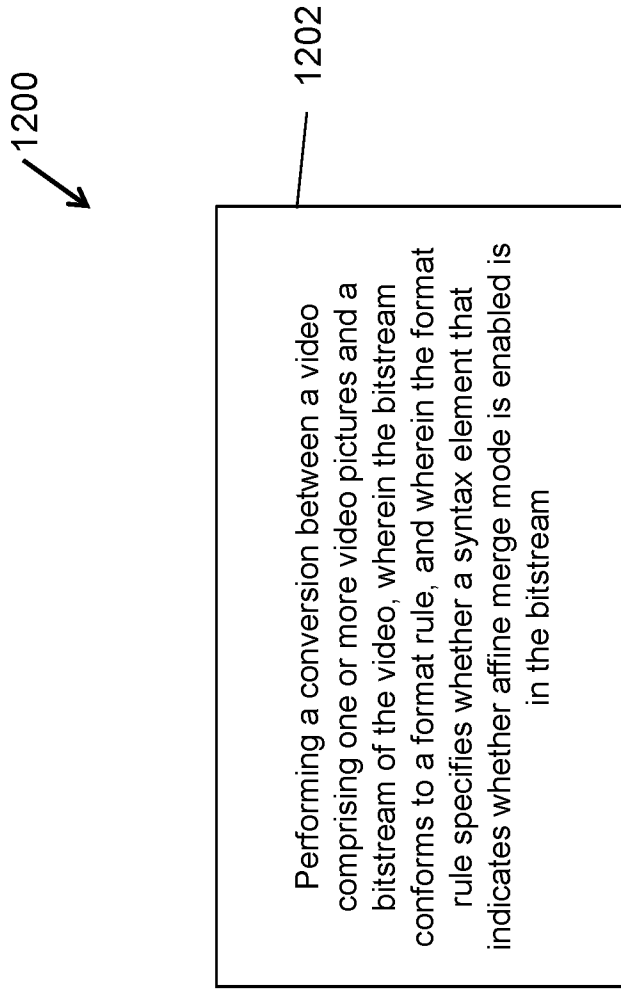
1000

1002

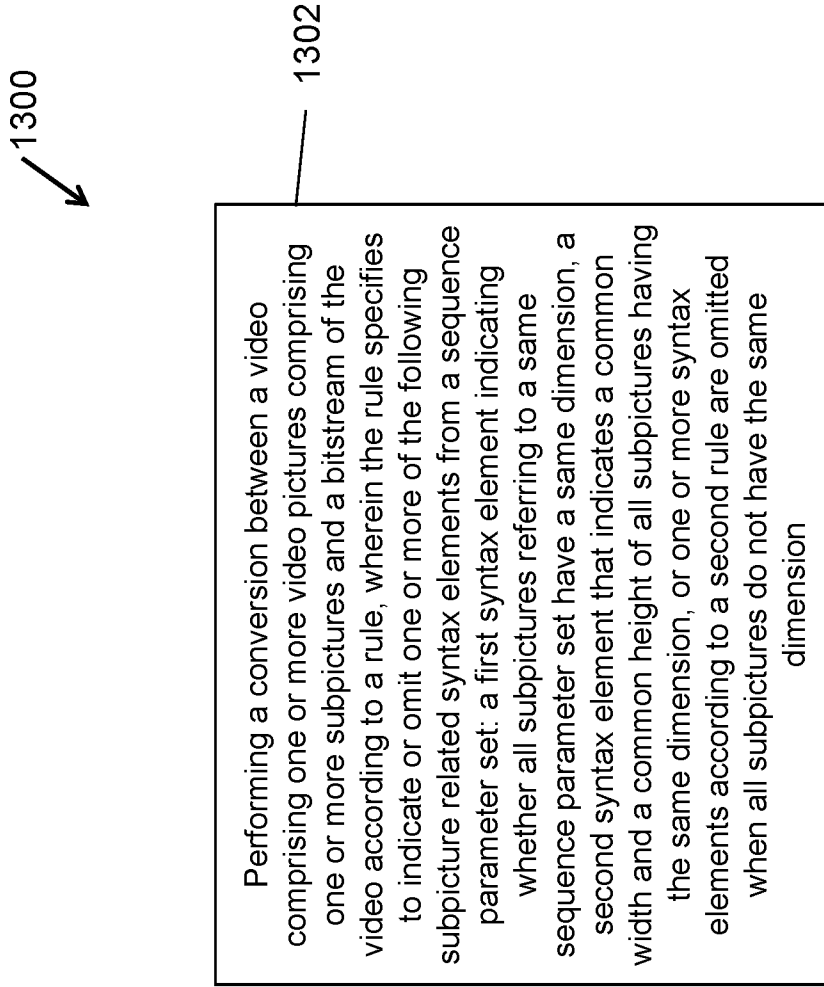
Performing a conversion between a video and a bitstream of the video according to a rule, wherein the rule specifies that one of the following two conditions is true when a number of subpictures minus 1 indicated in a sequence parameter set is greater than zero and when a syntax element in the sequence parameter set indicates that an  $i$ -th subpicture of a first video picture of the video in a coded layer video sequence is treated as a second video picture of the video in a decoding process that excludes an in-loop filtering operation: (1) the second video picture referred to by each active entry in a first reference picture list or a second reference picture list of the first video picture and the first video picture have a same value for each of the following: a picture width in units of luma samples, a picture height in units of luma samples, the number of subpictures minus 1, and a horizontal position of a top-left coding tree unit of the  $i$ -th subpicture, a vertical position of the top-left CTU of the  $i$ -th subpicture, a width of the  $i$ -th subpicture minus 1, a height of the  $i$ -th subpicture minus 1, the syntax element for the  $i$ -th subpicture, respectively, for each value of  $i$  in a range of 0 to the number of subpictures minus 1, inclusive; and (2) the second video picture referred to by each active entry in the first reference picture list or the second reference picture list is an inter-layer reference picture for which a value of the number of subpictures minus 1 is equal to zero

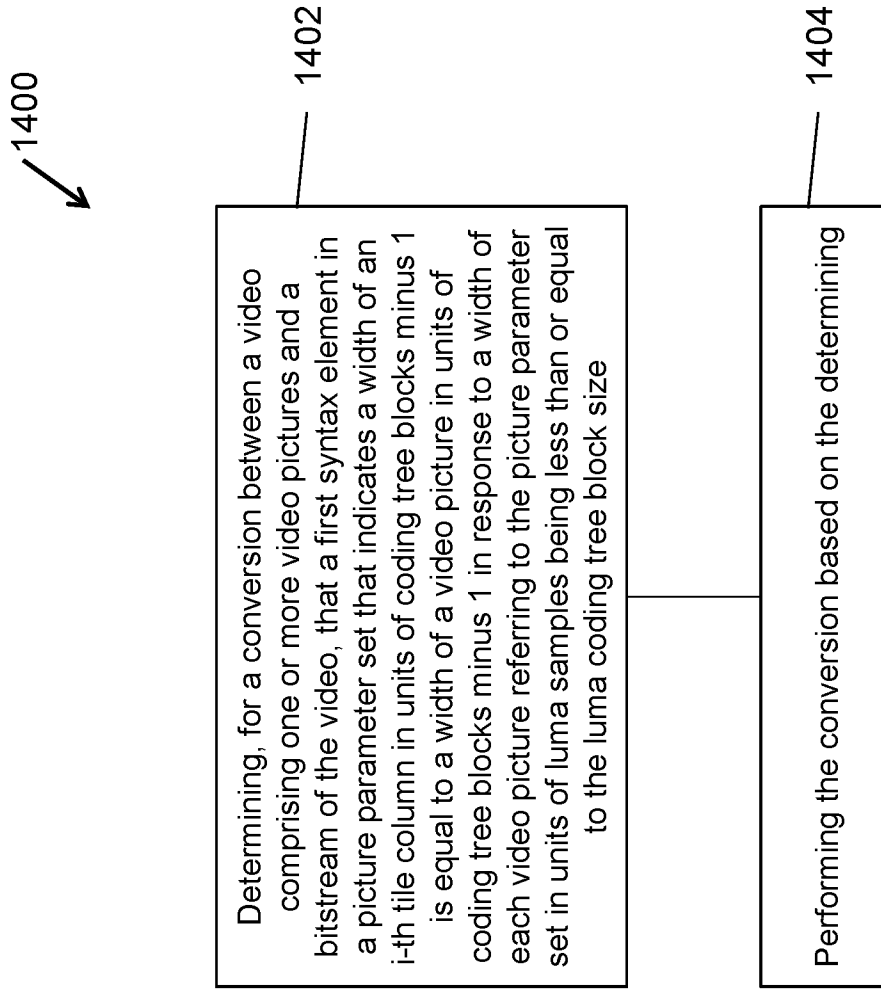
**FIG. 10**

**FIG. 11**

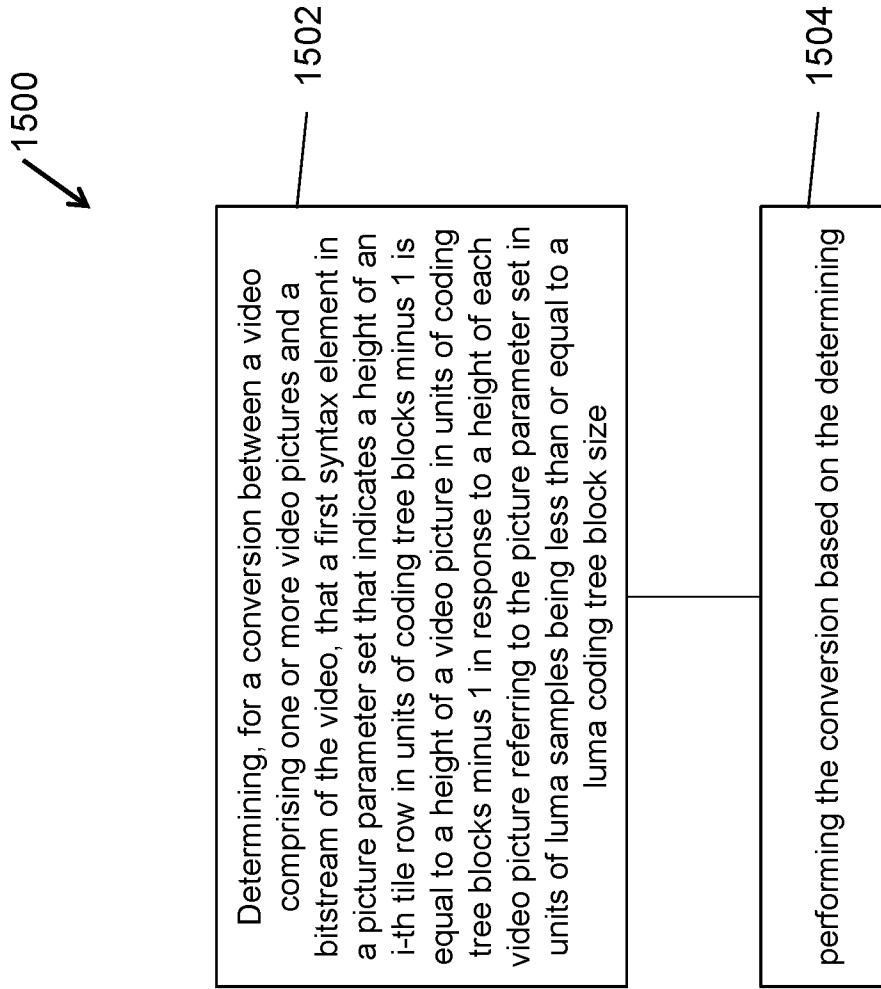


**FIG. 12**

**FIG. 13**



**FIG. 14**



**FIG. 15**

## INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 21/34259

## A. CLASSIFICATION OF SUBJECT MATTER

IPC - H04N 19/174 (2021.01)

CPC - H04N 19/17, H04N 19/174, H04N 19/436, H04N 19/50, H04N 19/70

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

See Search History document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

See Search History document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

See Search History document

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2020/057662 A1 (Huawei Technologies Co., Ltd.) 26 March 2020 (26.03.2020) entire document (especially para [0045]-[0046], [00108], [00123], [00134], [00210]-[00213])	1-8, and 34-36/(1-8)
X	WO 2019/145262 A1 (Fraunhofer-Gesellschaft zur Forderung der angewandten Forschung e.V.) 01 August 2019 (01.08.2019) entire document (especially pg. 17, ln 18-24; pg. 35, ln 17-25)	9, 20-31, and 34-36/(9, 20-31)
X	US 2020/0154101 A1 (Tencent America LLC ) 14 May 2020 (14.05.2020) entire document (especially para [0017], [0128], [0169]-[0170]).	10-19 and 34-36/(10-19)
X	US 2018/0124462 A1 (LIM et al.) 03 May 2018 (03.05.2018) entire document (especially para [0060], [0073]-[0074], [0077]-[0080]).	32-33, 34-36/(32-33)

 Further documents are listed in the continuation of Box C. See patent family annex.

\* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"D" document cited by the applicant in the international application

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&amp;" document member of the same patent family

Date of the actual completion of the international search

03 August 2021 (03.08.2021)

Date of mailing of the international search report

AUG 31 2021

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents  
P.O. Box 1450, Alexandria, Virginia 22313-1450

Facsimile No. 571-273-8300

Authorized officer

Kari Rodriguez

Telephone No. PCT Helpdesk: 571-272-4300

## INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 21/34259

**Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)**

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1.  Claims Nos.:  
because they relate to subject matter not required to be searched by this Authority, namely:
  
2.  Claims Nos.: 43  
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:  
Claim 43 are omnibus type claims, and therefore, not drafted in accordance with Rule 6.2(a).
  
3.  Claims Nos.: 37-42  
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

**Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)**

This International Searching Authority found multiple inventions in this international application, as follows:

1.  As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2.  As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3.  As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
  
4.  No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

**Remark on Protest**

- The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- No protest accompanied the payment of additional search fees.