(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 December 2020 (24.12.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/256136 A1

(51) International Patent Classification:
H04N 19/70 (2014.01)

(21) International Application Number:
PCT/JP2020/024230

(22) International Filing Date:
19 June 2020 (19.06.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/863,776 19 June 2019 (19.06.2019) US
62/868,641 28 June 2019 (28.06.2019) US

(71) Applicant: **SHARP KABUSHIKI KAISHA** [JP/JP]; 1, Takumi-cho, Sakai-ku, Sakai City, Osaka, 5908522 (JP).

(72) Inventor: **DESHPANDE, Sachin G.**.

(74) Agent: **HARAKENZO WORLD PATENT & TRADEMARK**; Daiwa Minamimorimachi Building, 2-6, Tenjinbashi 2-chome Kita, Kita-ku, Osaka-shi, Osaka, 5300041 (JP).

(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, 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.

(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))*
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

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

(54) **Title:** SYSTEMS AND METHODS FOR SIGNALING DECODED PICTURE BUFFER INFORMATION IN VIDEO CODING

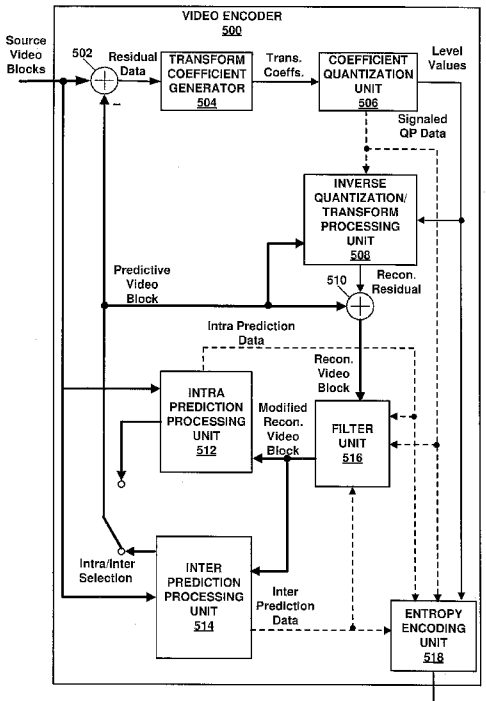

FIG. 5

(57) **Abstract:** This disclosure relates to video coding and more particularly to techniques for signaling decoded picture buffer information for coded video. According to an aspect of an invention, a syntax element indicating a presence of decoded picture buffer parameters is conditionally parsed only when it is determined that a value of a syntax element specifying a maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero.

WO 2020/256136 A1

Description

Title of Invention: SYSTEMS AND METHODS FOR SIGNALING DECODED PICTURE BUFFER INFORMATION IN VIDEO CODING

Technical Field

[0001] This disclosure relates to video coding and more particularly to techniques for signaling decoded picture buffer information for coded video.

Background Art

[0002] Digital video capabilities can be incorporated into a wide range of devices, including digital televisions, laptop or desktop computers, tablet computers, digital recording devices, digital media players, video gaming devices, cellular telephones, including so-called smartphones, medical imaging devices, and the like. Digital video may be coded according to a video coding standard. Video coding standards define the format of a compliant bitstream encapsulating coded video data. A compliant bitstream is a data structure that may be received and decoded by a video decoding device to generate reconstructed video data. Video coding standards may incorporate video compression techniques. Examples of video coding standards include ISO/IEC MPEG-4 Visual and ITU-T H.264 (also known as ISO/IEC MPEG-4 AVC) and High-Efficiency Video Coding (HEVC). HEVC is described in High Efficiency Video Coding (HEVC), Rec. ITU-T H.265, December 2016, which is incorporated by reference, and referred to herein as ITU-T H.265. Extensions and improvements for ITU-T H.265 are currently being considered for the development of next generation video coding standards. For example, the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC (Moving Picture Experts Group (MPEG) (collectively referred to as the Joint Video Exploration Team (JVET)) are working to standardized video coding technology with a compression capability that significantly exceeds that of the current HEVC standard. The Joint Exploration Model 7 (JEM 7), Algorithm Description of Joint Exploration Test Model 7 (JEM 7), ISO/IEC JTC1/SC29/WG11 Document: JVET-G1001, July 2017, Torino, IT, which is incorporated by reference herein, describes the coding features that were under coordinated test model study by the JVET as potentially enhancing video coding technology beyond the capabilities of ITU-T H.265. It should be noted that the coding features of JEM 7 are implemented in JEM reference software. As used herein, the term JEM may collectively refer to algorithms included in JEM 7 and implementations of JEM reference software. Further, in response to a "Joint Call for Proposals on Video Compression with Capabilities beyond HEVC," jointly issued by VCEG and MPEG, multiple descriptions of video coding tools were proposed by

various groups at the 10th Meeting of ISO/IEC JTC1/SC29/WG11 16-20 April 2018, San Diego, CA. From the multiple descriptions of video coding tools, a resulting initial draft text of a video coding specification is described in "Versatile Video Coding (Draft 1)," 10th Meeting of ISO/IEC JTC1/SC29/WG11 16-20 April 2018, San Diego, CA, document JVET-J1001-v2, which is incorporated by reference herein, and referred to as JVET-J1001. The current development of a next generation video coding standard by the VCEG and MPEG is referred to as the Versatile Video Coding (VVC) project. "Versatile Video Coding (Draft 5)," 14th Meeting of ISO/IEC JTC1/SC29/WG11 19-27 March 2019, Geneva, CH, document JVET-N1001-v7, which is incorporated by reference herein, and referred to as JVET-N1001, represents the current iteration of the draft text of a video coding specification corresponding to the VVC project.

[0003] Video compression techniques enable data requirements for storing and transmitting video data to be reduced. Video compression techniques may reduce data requirements by exploiting the inherent redundancies in a video sequence. Video compression techniques may sub-divide a video sequence into successively smaller portions (i.e., groups of pictures within a video sequence, a picture within a group of pictures, regions within a picture, sub-regions within regions, etc.). Intra prediction coding techniques (e.g., spatial prediction techniques within a picture) and inter prediction techniques (i.e., inter-picture techniques (temporal)) may be used to generate difference values between a unit of video data to be coded and a reference unit of video data. The difference values may be referred to as residual data. Residual data may be coded as quantized transform coefficients. Syntax elements may relate residual data and a reference coding unit (e.g., intra-prediction mode indices, and motion information). Residual data and syntax elements may be entropy coded. Entropy encoded residual data and syntax elements may be included in data structures forming a compliant bitstream.

Summary of Invention

[0004] In one example, a method of decoding video data, the method comprising: parsing a syntax element specifying a maximum number of temporal sublayers that may be present in a coded video sequence; determining whether a value of the syntax element specifying the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero; and conditionally parsing a syntax element indicating the presence of decoded picture buffer parameters only when it is determined that the value of the syntax element specifying the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero.

[0005] In one example, a device comprising one or more processors configured to: parse a

syntax element specifying a maximum number of temporal sublayers that may be present in a coded video sequence; determine whether a value of the syntax element specifying the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero; and conditionally parse a syntax element indicating the presence of decoded picture buffer parameters only when it is determined that the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero.

Brief Description of Drawings

[0006] [fig.1]FIG. 1 is a block diagram illustrating an example of a system that may be configured to encode and decode video data according to one or more techniques of this disclosure.

[fig.2]FIG. 2 is a conceptual diagram illustrating coded video data and corresponding data structures according to one or more techniques of this disclosure.

[fig.3]FIG. 3 is a conceptual diagram illustrating a data structure encapsulating coded video data and corresponding metadata according to one or more techniques of this disclosure.

[fig.4]FIG. 4 is a conceptual drawing illustrating an example of components that may be included in an implementation of a system that may be configured to encode and decode video data according to one or more techniques of this disclosure.

[fig.5]FIG. 5 is a block diagram illustrating an example of a video encoder that may be configured to encode video data according to one or more techniques of this disclosure.

[fig.6]FIG. 6 is a block diagram illustrating an example of a video decoder that may be configured to decode video data according to one or more techniques of this disclosure.

Description of Embodiments

[0007] In general, this disclosure describes various techniques for coding video data. In particular, this disclosure describes techniques for signaling decoded picture buffer information for coded video data. It should be noted that although techniques of this disclosure are described with respect to ITU-T H.264, ITU-T H.265, JEM, and JVET-N1001, the techniques of this disclosure are generally applicable to video coding. For example, the coding techniques described herein may be incorporated into video coding systems, (including video coding systems based on future video coding standards) including video block structures, intra prediction techniques, inter prediction techniques, transform techniques, filtering techniques, and/or entropy coding techniques other than those included in ITU-T H.265, JEM, and JVET-N1001. Thus, reference to ITU-T H.264, ITU-T H.265, JEM, and/or JVET-N1001 is for descriptive purposes and should not be construed to limit the scope of the techniques described herein. Further, it should be noted that incorporation by reference of

documents herein is for descriptive purposes and should not be construed to limit or create ambiguity with respect to terms used herein. For example, in the case where an incorporated reference provides a different definition of a term than another incorporated reference and/or as the term is used herein, the term should be interpreted in a manner that broadly includes each respective definition and/or in a manner that includes each of the particular definitions in the alternative.

[0008] In one example, a method of encoding video data comprises signaling a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video and signaling a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, wherein the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video is calculated as the sum of the first value and the second value.

[0009] In one example, a device comprises one or more processors configured to signal a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video and signal a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, wherein the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video is calculated as the sum of the first value and the second value.

[0010] In one example, a non-transitory computer-readable storage medium comprises instructions stored thereon that, when executed, cause one or more processors of a device to signal a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video and signal a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, wherein the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video is calculated as the sum of the first value and the second value.

[0011] In one example, an apparatus comprises means for signaling a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video and means for signaling a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, wherein the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video is calculated as the sum of the first value and the second value.

[0012] In one example, a method of decoding video data comprises parsing a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video, parsing a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, and calculating the maximum required size of a decoded picture buffer for the corresponding second

sub-layer of video as the sum of the first value and the second value.

[0013] In one example, a device comprises one or more processors configured to parse a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video, parse a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, and calculate the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video as the sum of the first value and the second value.

[0014] In one example, a non-transitory computer-readable storage medium comprises instructions stored thereon that, when executed, cause one or more processors of a device to parse a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video, parse a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, and calculate the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video as the sum of the first value and the second value.

[0015] In one example, an apparatus comprises means for parsing a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video, means for parsing a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, and means for calculating the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video as the sum of the first value and the second value.

[0016] The details of one or more examples are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

[0017] Video content includes video sequences comprised of a series of frames (or pictures). A series of frames may also be referred to as a group of pictures (GOP). Each video frame or picture may divided into one or more regions. Regions may be defined according to a base unit (e.g., a video block) and sets of rules defining a region. For example, a rule defining a region may be that a region must be an integer number of video blocks arranged in a rectangle. Further, video blocks in a region may be ordered according to a scan pattern (e.g., a raster scan). As used herein, the term video block may generally refer to an area of a picture or may more specifically refer to the largest array of sample values that may be predictively coded, sub-divisions thereof, and/or corresponding structures. Further, the term current video block may refer to an area of a picture being encoded or decoded. A video block may be defined as an array of sample values. It should be noted that in some cases pixel values may be described as including sample values for respective components of video data, which may also be referred to as color components, (e.g., luma (Y) and chroma (Cb and Cr) components

or red, green, and blue components). It should be noted that in some cases, the terms pixel value and sample value are used interchangeably. Further, in some cases, a pixel or sample may be referred to as a pel. A video sampling format, which may also be referred to as a chroma format, may define the number of chroma samples included in a video block with respect to the number of luma samples included in a video block. For example, for the 4:2:0 sampling format, the sampling rate for the luma component is twice that of the chroma components for both the horizontal and vertical directions.

[0018] A video encoder may perform predictive encoding on video blocks and sub-divisions thereof. Video blocks and sub-divisions thereof may be referred to as nodes. ITU-T H.264 specifies a macroblock including 16x16 luma samples. That is, in ITU-T H.264, a picture is segmented into macroblocks. ITU-T H.265 specifies an analogous Coding Tree Unit (CTU) structure (which may be referred to as a largest coding unit (LCU)). In ITU-T H.265, pictures are segmented into CTUs. In ITU-T H.265, for a picture, a CTU size may be set as including 16x16, 32x32, or 64x64 luma samples. In ITU-T H.265, a CTU is composed of respective Coding Tree Blocks (CTB) for each component of video data (e.g., luma (Y) and chroma (Cb and Cr). It should be noted that video having one luma component and the two corresponding chroma components may be described as having two channels, i.e., a luma channel and a chroma channel. Further, in ITU-T H.265, a CTU may be partitioned according to a quadtree (QT) partitioning structure, which results in the CTBs of the CTU being partitioned into Coding Blocks (CB). That is, in ITU-T H.265, a CTU may be partitioned into quadtree leaf nodes. According to ITU-T H.265, one luma CB together with two corresponding chroma CBs and associated syntax elements are referred to as a coding unit (CU). In ITU-T H.265, a minimum allowed size of a CB may be signaled. In ITU-T H.265, the smallest minimum allowed size of a luma CB is 8x8 luma samples. In ITU-T H.265, the decision to code a picture area using intra prediction or inter prediction is made at the CU level.

[0019] In ITU-T H.265, a CU is associated with a prediction unit (PU) structure having its root at the CU. In ITU-T H.265, PU structures allow luma and chroma CBs to be split for purposes of generating corresponding reference samples. That is, in ITU-T H.265, luma and chroma CBs may be split into respective luma and chroma prediction blocks (PBs), where a PB includes a block of sample values for which the same prediction is applied. In ITU-T H.265, a CB may be partitioned into 1, 2, or 4 PBs. ITU-T H.265 supports PB sizes from 64x64 samples down to 4x4 samples. In ITU-T H.265, square PBs are supported for intra prediction, where a CB may form the PB or the CB may be split into four square PBs. In ITU-T H.265, in addition to the square PBs, rectangular PBs are supported for inter prediction, where a CB may by halved vertically or horizontally to form PBs. Further, it should be noted that in ITU-T H.265, for inter

prediction, four asymmetric PB partitions are supported, where the CB is partitioned into two PBs at one quarter of the height (at the top or the bottom) or width (at the left or the right) of the CB. Intra prediction data (e.g., intra prediction mode syntax elements) or inter prediction data (e.g., motion data syntax elements) corresponding to a PB is used to produce reference and/or predicted sample values for the PB.

[0020] JEM specifies a CTU having a maximum size of 256x256 luma samples. JEM specifies a quadtree plus binary tree (QTBT) block structure. In JEM, the QTBT structure enables quadtree leaf nodes to be further partitioned by a binary tree (BT) structure. That is, in JEM, the binary tree structure enables quadtree leaf nodes to be recursively divided vertically or horizontally. In JVET-N1001, CTUs are partitioned according a quadtree plus multi-type tree (QTMT or QT+MTT) structure. The QTMT in JVET-N1001 is similar to the QTBT in JEM. However, in JVET-N1001, in addition to indicating binary splits, the multi-type tree may indicate so-called ternary (or triple tree (TT)) splits. A ternary split divides a block vertically or horizontally into three blocks. In the case of a vertical TT split, a block is divided at one quarter of its width from the left edge and at one quarter its width from the right edge and in the case of a horizontal TT split a block is at one quarter of its height from the top edge and at one quarter of its height from the bottom edge.

[0021] As described above, each video frame or picture may divided into one or more regions. For example, according to ITU-T H.265, each video frame or picture may be partitioned to include one or more slices and further partitioned to include one or more tiles, where each slice includes a sequence of CTUs (e.g., in raster scan order) and where a tile is a sequence of CTUs corresponding to a rectangular area of a picture. It should be noted that a slice, in ITU-T H.265, is a sequence of one or more slice segments starting with an independent slice segment and containing all subsequent dependent slice segments (if any) that precede the next independent slice segment (if any). A slice segment, like a slice, is a sequence of CTUs. Thus, in some cases, the terms slice and slice segment may be used interchangeably to indicate a sequence of CTUs arranged in a raster scan order. Further, it should be noted that in ITU-T H.265, a tile may consist of CTUs contained in more than one slice and a slice may consist of CTUs contained in more than one tile. However, ITU-T H.265 provides that one or both of the following conditions shall be fulfilled: (1) All CTUs in a slice belong to the same tile; and (2) All CTUs in a tile belong to the same slice.

[0022] With respect to JVET-N1001, slices are required to consist of an integer number of bricks instead of only being required to consist of an integer number of CTUs. In JVET-N1001, a brick is a rectangular region of CTU rows within a particular tile in a picture. Further, in JVET-N1001, a tile may be partitioned into multiple bricks, each of which consisting of one or more CTU rows within the tile. A tile that is not partitioned

into multiple bricks is also referred to as a brick. However, a brick that is a true subset of a tile is not referred to as a tile. As such, a slice including a set of CTUs which do not form a rectangular region of a picture may or may not be supported in some video coding techniques. Further, it should be noted that in some cases, a slice may be required to consist of an integer number of complete tiles and in this case is referred to as a tile group. The techniques described herein may applicable to bricks, slices, tiles, and/or tile groups. FIG. 2 is a conceptual diagram illustrating an example of a group of pictures including slices. In the example illustrated in FIG. 2, Pic_3 is illustrated as including two slices (i.e., $Slice_0$ and $Slice_1$). In the example illustrated in FIG. 2, $Slice_0$ includes one brick, i.e., $Brick_0$ and $Slice_1$ includes two bricks, i.e., $Brick_1$ and $Brick_2$. It should be noted that in some cases, $Slice_0$ and $Slice_1$ may meet the requirements of and be classified as tiles and/or tile groups.

[0023] For intra prediction coding, an intra prediction mode may specify the location of reference samples within a picture. In ITU-T H.265, defined possible intra prediction modes include a planar (i.e., surface fitting) prediction mode, a DC (i.e., flat overall averaging) prediction mode, and 33 angular prediction modes (predMode: 2-34). In JEM, defined possible intra-prediction modes include a planar prediction mode, a DC prediction mode, and 65 angular prediction modes. It should be noted that planar and DC prediction modes may be referred to as non-directional prediction modes and that angular prediction modes may be referred to as directional prediction modes. It should be noted that the techniques described herein may be generally applicable regardless of the number of defined possible prediction modes.

[0024] For inter prediction coding, a reference picture is determined and a motion vector (MV) identifies samples in the reference picture that are used to generate a prediction for a current video block. For example, a current video block may be predicted using reference sample values located in one or more previously coded picture(s) and a motion vector is used to indicate the location of the reference block relative to the current video block. A motion vector may describe, for example, a horizontal displacement component of the motion vector (i.e., MV_x), a vertical displacement component of the motion vector (i.e., MV_y), and a resolution for the motion vector (e.g., one-quarter pixel precision, one-half pixel precision, one-pixel precision, two-pixel precision, four-pixel precision). Previously decoded pictures, which may include pictures output before or after a current picture, may be organized into one or more to reference pictures lists and identified using a reference picture index value. Further, in inter prediction coding, uni-prediction refers to generating a prediction using sample values from a single reference picture and bi-prediction refers to generating a prediction using respective sample values from two reference pictures. That is, in uni-prediction, a single reference picture and corresponding motion vector are used to

generate a prediction for a current video block and in bi-prediction, a first reference picture and corresponding first motion vector and a second reference picture and corresponding second motion vector are used to generate a prediction for a current video block. In bi-prediction, respective sample values are combined (e.g., added, rounded, and clipped, or averaged according to weights) to generate a prediction. Pictures and regions thereof may be classified based on which types of prediction modes may be utilized for encoding video blocks thereof. That is, for regions having a B type (e.g., a B slice), bi-prediction, uni-prediction, and intra prediction modes may be utilized, for regions having a P type (e.g., a P slice), uni-prediction, and intra prediction modes may be utilized, and for regions having an I type (e.g., an I slice), only intra prediction modes may be utilized. As described above, reference pictures are identified through reference indices. For example, for a P slice, there may be a single reference picture list, RefPicList0 and for a B slice, there may be a second independent reference picture list, RefPicList1, in addition to RefPicList0. It should be noted that for uni-prediction in a B slice, one of RefPicList0 or RefPicList1 may be used to generate a prediction. Further, it should be noted that during the decoding process, at the onset of decoding a picture, reference picture list(s) are generated from previously decoded pictures stored in a decoded picture buffer (DPB).

[0025] Further, a coding standard may support various modes of motion vector prediction. Motion vector prediction enables the value of a motion vector for a current video block to be derived based on another motion vector. For example, a set of candidate blocks having associated motion information may be derived from spatial neighboring blocks and temporal neighboring blocks to the current video block. Further, generated (or default) motion information may be used for motion vector prediction. Examples of motion vector prediction include advanced motion vector prediction (AMVP), temporal motion vector prediction (TMVP), so-called "merge" mode, and "skip" and "direct" motion inference. Further, other examples of motion vector prediction include advanced temporal motion vector prediction (ATMVP) and Spatial-temporal motion vector prediction (STMVP). For motion vector prediction, both a video encoder and video decoder perform the same process to derive a set of candidates. Thus, for a current video block, the same set of candidates is generated during encoding and decoding.

[0026] As described above, for inter prediction coding, reference samples in a previously coded picture are used for coding video blocks in a current picture. Previously coded pictures which are available for use as reference when coding a current picture are referred as reference pictures. It should be noted that the decoding order does not necessary correspond with the picture output order, i.e., the temporal order of pictures in a video sequence. In ITU-T H.265, when a picture is decoded it is stored to a

decoded picture buffer (DPB) (which may be referred to as frame buffer, a reference buffer, a reference picture buffer, or the like). In ITU-T H.265, pictures stored to the DPB are removed from the DPB when they been output and are no longer needed for coding subsequent pictures. In ITU-T H.265, a determination of whether pictures should be removed from the DPB is invoked once per picture, after decoding a slice header, i.e., at the onset of decoding a picture. For example, referring to FIG. 2, Pic_3 is illustrated as referencing Pic_2. Similarly, Pic_4 is illustrated as referencing Pic_1. With respect to FIG. 2 assuming the picture number corresponds to the decoding order the DPB would be populated as follows: after decoding Pic_1, the DPB would include {Pic_1 }; at the onset of decoding Pic_2, the DPB would include {Pic_1}; after decoding Pic_2, the DPB would include {Pic_1, Pic_2}; at the onset of decoding Pic_3, the DPB would include {Pic_1, Pic_2}. Pic_3 would then be decoded with reference to Pic_2 and after decoding Pic_3, the DPB would include {Pic_1, Pic_2, Pic_3}. At the onset of decoding Pic_4, pictures Pic_2 and Pic_3 would be marked for removal from the DPB, as they are not needed for decoding Pic_4 (or any subsequent pictures, not shown) and assuming Pic_2 and Pic_3 have been output, the DPB would be updated to include {Pic_1}. Pic_4 would then be decoded with referencing Pic_1. The process of marking pictures for removal from a DPB may be referred to as reference picture set (RPS) management.

[0027] As described above, intra prediction data or inter prediction data is used to produce reference sample values for a block of sample values. The difference between sample values included in a current PB, or another type of picture area structure, and associated reference samples (e.g., those generated using a prediction) may be referred to as residual data. Residual data may include respective arrays of difference values corresponding to each component of video data. Residual data may be in the pixel domain. A transform, such as, a discrete cosine transform (DCT), a discrete sine transform (DST), an integer transform, a wavelet transform, or a conceptually similar transform, may be applied to an array of difference values to generate transform coefficients. It should be noted that in ITU-T H.265 and JVET-N1001, a CU is associated with a transform unit (TU) structure having its root at the CU level. That is, an array of difference values may be partitioned for purposes of generating transform coefficients (e.g., four 8x8 transforms may be applied to a 16x16 array of residual values). For each component of video data, such sub-divisions of difference values may be referred to as Transform Blocks (TBs). It should be noted that in some cases, a core transform and a subsequent secondary transforms may be applied (in the video encoder) to generate transform coefficients. For a video decoder, the order of transforms is reversed.

[0028] A quantization process may be performed on transform coefficients or residual sample values directly (e.g., in the case, of palette coding quantization). Quantization approximates transform coefficients by amplitudes restricted to a set of specified

values. Quantization essentially scales transform coefficients in order to vary the amount of data required to represent a group of transform coefficients. Quantization may include division of transform coefficients (or values resulting from the addition of an offset value to transform coefficients) by a quantization scaling factor and any associated rounding functions (e.g., rounding to the nearest integer). Quantized transform coefficients may be referred to as coefficient level values. Inverse quantization (or "dequantization") may include multiplication of coefficient level values by the quantization scaling factor, and any reciprocal rounding or offset addition operations. It should be noted that as used herein the term quantization process in some instances may refer to division by a scaling factor to generate level values and multiplication by a scaling factor to recover transform coefficients in some instances. That is, a quantization process may refer to quantization in some cases and inverse quantization in some cases. Further, it should be noted that although in some of the examples below quantization processes are described with respect to arithmetic operations associated with decimal notation, such descriptions are for illustrative purposes and should not be construed as limiting. For example, the techniques described herein may be implemented in a device using binary operations and the like. For example, multiplication and division operations described herein may be implemented using bit shifting operations and the like.

[0029] Quantized transform coefficients and syntax elements (e.g., syntax elements indicating a coding structure for a video block) may be entropy coded according to an entropy coding technique. An entropy coding process includes coding values of syntax elements using lossless data compression algorithms. Examples of entropy coding techniques include content adaptive variable length coding (CAVLC), context adaptive binary arithmetic coding (CABAC), probability interval partitioning entropy coding (PIPE), and the like. Entropy encoded quantized transform coefficients and corresponding entropy encoded syntax elements may form a compliant bitstream that can be used to reproduce video data at a video decoder. An entropy coding process, for example, CABAC, may include performing a binarization on syntax elements. Binarization refers to the process of converting a value of a syntax element into a series of one or more bits. These bits may be referred to as "bins." Binarization may include one or a combination of the following coding techniques: fixed length coding, unary coding, truncated unary coding, truncated Rice coding, Golomb coding, k-th order exponential Golomb coding, and Golomb-Rice coding. For example, binarization may include representing the integer value of 5 for a syntax element as 00000101 using an 8-bit fixed length binarization technique or representing the integer value of 5 as 11110 using a unary coding binarization technique. As used herein each of the terms fixed length coding, unary coding, truncated unary coding, truncated Rice coding, Golomb

coding, k-th order exponential Golomb coding, and Golomb-Rice coding may refer to general implementations of these techniques and/or more specific implementations of these coding techniques. For example, a Golomb-Rice coding implementation may be specifically defined according to a video coding standard. In the example of CABAC, for a particular bin, a context provides a most probable state (MPS) value for the bin (i.e., an MPS for a bin is one of 0 or 1) and a probability value of the bin being the MPS or the least probably state (LPS). For example, a context may indicate, that the MPS of a bin is 0 and the probability of the bin being 1 is 0.3. It should be noted that a context may be determined based on values of previously coded bins including bins in the current syntax element and previously coded syntax elements. For example, values of syntax elements associated with neighboring video blocks may be used to determine a context for a current bin.

[0030] With respect to the equations used herein, the following arithmetic operators may be used:

+	Addition
−	Subtraction
*	Multiplication, including matrix multiplication
x^y	Exponentiation. Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	Integer division with truncation of the result toward zero. For example, 7 / 4 and −7 / −4 are truncated to 1 and −7 / 4 and 7 / −4 are truncated to −1.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.

[0031]

Further, the following mathematical functions may be used:

Log2(x) the base-2 logarithm of x;

$$\text{Min}(x, y) = \begin{cases} x & ; \quad x <= y \\ y & ; \quad x > y \end{cases};$$

$$\text{Max}(x, y) = \begin{cases} x & ; \quad x >= y \\ y & ; \quad x < y \end{cases}$$

Ceil(x) the smallest integer greater than or equal to x.

[0032] With respect to the example syntax used herein, the following definitions of logical operators may be applied:

x && y Boolean logical "and" of x and y

x | | y Boolean logical "or" of x and y

! Boolean logical "not"

x ? y : z If x is TRUE or not equal to 0, evaluates to the value of y; otherwise, evaluates to the value of z.

[0033] Further, the following relational operators may be applied:

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to

[0034]

Further, it should be noted that in the syntax descriptors used herein, the following descriptors may be applied:

- b(8): byte having any pattern of bit string (8 bits). The parsing process for this descriptor is specified by the return value of the function read_bits(8).

-f(n): fixed-pattern bit string using n bits written (from left to right) with the left bit first. The parsing process for this descriptor is specified by the return value of the function read_bits(n).

- se(v): signed integer 0-th order Exp-Golomb-coded syntax element with the left bit first.

-tb(v): truncated binary using up to maxVal bits with maxVal defined in the semantics of the symtax element.

-tu(v): truncated unary using up to maxVal bits with maxVal defined in the semantics of the symtax element.

-u(n): unsigned integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a binary representation of an unsigned integer with most significant bit written first.

-ue(v): unsigned integer 0-th order Exp-Golomb-coded syntax element with the left bit first.

[0035] As described above, video content includes video sequences comprised of a series of frames (or pictures) and each video frame or picture may be divided into one or more regions. A coded video sequence (CVS) may be encapsulated (or structured) as a sequence of access units, where each access unit includes video data structured as network abstraction layer (NAL) units. A bitstream may be described as including a sequence of NAL units forming one or more CVSs. It should be noted that multi-layer extensions enable a video presentation to include a base layer and one or more additional enhancement layers. For example, a base layer may enable a video presentation having a basic level of quality (e.g., a High Definition rendering and/or a 30 Hz frame rate) to be presented and an enhancement layer may enable a video presentation having an enhanced level of quality (e.g., an Ultra High Definition rendering

and/or a 60 Hz frame rate) to be presented. An enhancement layer may be coded by referencing a base layer. That is, for example, a picture in an enhancement layer may be coded (e.g., using inter-layer prediction techniques) by referencing one or more pictures (including scaled versions thereof) in a base layer. Each NAL unit may include an identifier indicating a layer of video data the NAL unit is associated with. It should be noted that sub-bitstream extraction may refer to a process where a device receiving a compliant or conforming bitstream forms a new compliant or conforming bitstream by discarding and/or modifying data in the received bitstream. For example, sub-bitstream extraction may be used to form a new compliant or conforming bitstream corresponding to a particular representation of video (e.g., a high quality repre-sentation). Layers may also be coded independent of each other. In this case, there may not be an inter-layer prediction between two layers.

[0036] Referring to the example illustrated in FIG. 2, each slice of video data included in Pic_3 (i.e., $Slice_0$ and $Slice_1$) is illustrated as being encapsulated in a NAL unit. In JVET-N1001, each of a video sequence, a GOP, a picture, a slice, and CTU may be associated with metadata that describes video coding properties. JVET-N1001 defines parameters sets that may be used to describe video data and/or video coding properties. In particular, JVET-N1001 includes the following five types of parameter sets: decoding parameter set (DPS), video parameter set (VPS), sequence parameter set (SPS), picture parameter set (PPS), and adaption parameter set (APS). With respect to the SPS, JVET-N1001 includes the following definition:

sequence parameter set (SPS): A syntax structure containing syntax elements that apply to zero or more entire CVSs as determined by the content of a syntax element found in the PPS referred to by a syntax element found in each slice header.

[0037] In JVET-N1001, parameter sets may be encapsulated as a special type of NAL unit or may be signaled as a message. NAL units including coded video data (e.g., a slice) may be referred to as VCL (Video Coding Layer) NAL units and NAL units including metadata (e.g., parameter sets) may be referred to as non-VCL NAL units. Further, JVET-N1001 enables supplemental enhancement information (SEI) messages to be signaled. In JVET-N1001, SEI messages assist in processes related to decoding, display or other purposes, however, SEI messages may not be required for constructing the luma or chroma samples by the decoding process. In JVET-N1001, SEI messages may be signaled in a bitstream using non-VCL NAL units. Further, SEI messages may

be conveyed by some means other than by being present in the bitstream (i.e., signaled out-of-band).

[0038] An access unit may be called a layer access unit. As described above, multi-layer extensions enable a video presentation to include a base layer and one or more additional enhancement layers. It should be noted that in ITU-T H.265 a temporal true subset of a scalable layer is not referred to as a layer but referred to as a sub-layer or temporal sub-layer. That is, ITU-T H.265 provides the following definitions with respect to sub-layers:

sub-layer: A temporal scalable layer of a temporal scalable bitstream, consisting of VCL NAL units with a particular value of the TemporalId variable and the associated non-VCL NAL units.

The term sub-layer and temporal sub-layer may be used interchangeably.

[0039] As described above, sub-bitstream extraction may be used to form a new bitstream corresponding to a particular representation of video. It should be noted that for a particular temporal video representation (e.g., a 60 Hz representation), sub-bitstream extraction may include extracting a base layer and one or more enhancement layers, where a highest temporal sub-layer identifier (e.g., HighestTid) can be used to specify a particular representation. That is, for example, a 60 Hz representation may be formed by extracting a 10 Hz temporal sub-layer identifier = 0, a 30 Hz temporal sub-layer identifier = 1, and a 60 Hz temporal sub-layer identifier = 2. That is, each temporal sub-layer having a temporal sub-layer identifier less than equal to a target sub-layer, HighestTid, is extracted for the representation.

[0040] FIG. 3 illustrates an example of a bitstream including multiple CVSs, where a CVS is represented by NAL units included in a respective access unit. In the example illustrated in FIG. 3, non-VCL NAL units include respective parameter set NAL units (i.e., Sequence Parameter Sets (SPS), and Picture Parameter Set (PPS) NAL units), an SEI message NAL unit, and an access unit delimiter NAL unit. It should be noted that in FIG.3, HEADER is a NAL unit header.

[0041]

JVET-N1001 defines NAL unit header semantics that specify the type of Raw Byte Sequence Payload (RBSP) data structure included in the NAL unit. Table 1 illustrates the syntax of the NAL unit header provided in JVET-N1001.

nal_unit_header( ) {	**Descriptor**
zero_tid_required_flag	u(1)
nuh_temporal_id_plus1	u(3)
nal_unit_type_lsb	u(4)
nuh_layer_id	u(7)
nuh_reserved_zero_bit	u(1)
}	

Table 1

[0042]

JVET-N1001 provides the following definitions for the respective syntax elements illustrated in Table 1.

zero_tid_required_flag equal to 0 specifies that zero_tid_required_flag does not impose any additional constraints on the value of nuh_temporal_id_plus1.

nuh_temporal_id_plus1 minus 1 specifies a temporal identifier for the NAL unit.
The value of nuh_temporal_id_plus1 shall not be equal to 0. When zero_tid_required_flag is equal to 1, the value of nuh_temporal_id_plus1 shall be equal to 1.
The variable TemporalId is derived as follows:

TemporalId = nuh_temporal_id_plus1 − 1

NOTE – NAL unit types in the range of 16 to 31, inclusive, have zero_tid_required_flag equal to 1, and consequently have TemporalId equal to 0.

The value of TemporalId shall be the same for all VCL NAL units of a layer access unit. The value of TemporalId of a coded picture or a layer access unit is the value of the TemporalId of the VCL NAL units of the coded picture or the layer access unit.

The value of TemporalId for non-VCL NAL units is constrained as follows:

– If NalUnitType is equal to SPS_NUT, TemporalId is equal to 0 and the TemporalId of the layer access unit containing the NAL unit shall be equal to 0.

- Otherwise, if NalUnitType is equal to APS_NUT, TemporalId shall be equal to that of the layer access unit containing the NAL unit.
- Otherwise, when NalUnitType is not equal to EOS_NUT and not equal to EOB_NUT, TemporalId shall be greater than or equal to the TemporalId of the layer access unit containing the NAL unit.

NOTE – When the NAL unit is a non-VCL NAL unit, the value of TemporalId is equal to the minimum value of the TemporalId values of all layer access units to which the non-VCL NAL unit applies. When NalUnitType is equal to PPS_NUT, TemporalId may be greater than or equal to the TemporalId of the containing layer access unit, as all picture parameter sets (PPSs) may be included in the beginning of a bitstream, wherein the first coded picture has TemporalId equal to 0. When NalUnitType is equal to PREFIX_SEI_NUT or SUFFIX_SEI_NUT, TemporalId may be greater than or equal to the TemporalId of the containing layer access unit, as an SEI NAL unit may contain information that applies to a bitstream subset that includes layer access units for which the TemporalId values are greater than the TemporalId of the layer access unit containing the SEI NAL unit.

nuh_layer_id specifies the identifier of the layer to which a VCL NAL unit belongs or the identifier of a layer to which a non-VCL NAL unit applies. The value of nuh_layer_id shall be in the range of 0 to 126, inclusive. The value of 127 may be specified in the future by ITU-T | ISO/IEC. For purposes other than determining the amount of data in the decoding units of the bitstream, decoders shall ignore all data that follow the value 127 for nuh_layer_id in a NAL unit.

NOTE – The value of 127 for nuh_layer_id may be used to indicate an extended layer identifier in a future extension of this Specification.

The value of nuh_layer_id shall be the same for all VCL NAL units of a coded picture. The value of nuh_layer_id of a coded picture is the value of the nuh_layer_id of the VCL NAL units of the coded picture.

nuh_reserved_zero_bit shall be equal to '0'. The value 1 of nuh_reserved_zero_bit may be specified in the future by ITU-T | ISO/IEC. Decoders shall ignore (i.e. remove from the bitstream and discard) NAL units with nuh_reserved_zero_bit equal to '1'.

[0043] With respect to the syntax element **nal_unit_type_lsb**, JVET-N1001 provides the following:

nal_unit_type_lsb specifies the least significant bits for the NAL unit type.

The variable NalUnitType, which specifies the NAL unit type, i.e., the type of RBSP data structure contained in the NAL unit as specified in Table 2 is derived as follows:

NalUnitType = (zero_tid_required_flag << 4) + nal_unit_type_lsb

NAL units that have NalUnitType in the range of UNSPEC28..UNSPEC31, inclusive, for which semantics are not specified, shall not affect the decoding process specified in this Specification.

NOTE – NAL unit types in the range of UNSPEC28..UNSPEC31 may be used as determined by the application. No decoding process for these values of NalUnitType is specified in this Specification. Since different applications might use these NAL unit types for different purposes, particular care must be exercised in the design of encoders that generate NAL units with these NalUnitType values, and in the design of decoders that interpret the content of NAL units with these NalUnitType values. This Specification does not define any management for these values. These NalUnitType values might only be suitable for use in contexts in which "collisions" of usage (i.e., different definitions of the meaning of the NAL unit content for the same NalUnitType value) are unimportant, or not possible, or are managed – e.g., defined or managed in the controlling application or transport specification, or by controlling the environment in which bitstreams are distributed.

For purposes other than determining the amount of data in the decoding units of the bitstream, decoders shall ignore (remove from the bitstream and discard) the contents of all NAL units that use reserved values of NalUnitType.

NOTE – This requirement allows future definition of compatible extensions to this Specification.

NalUnitType	Name of NalUnitType	Content of NAL unit and RBSP syntax structure	NAL unit type class
0	PPS_NUT	Picture parameter set pic_parameter_set_rbsp( )	non-VCL
1	AUD_NUT	Access unit delimiter access_unit_delimiter_rbsp( )	non-VCL
2 3	PREFIX_SEI_NUT SUFFIX_SEI_NUT	Supplemental enhancement information sei_rbsp( )	non-VCL
4	APS_NUT	Adaptation parameter set adaptation_parameter_set_rbsp( )	non-VCL
6 5..7	RSV_NVCL65.. RSV_NVCL7	Reserved	non-VCL
8	TRAIL_NUT	Coded slice of a non-STSA trailing picture slice_layer_rbsp( )	VCL
9	STSA_NUT	Coded slice of an STSA picture slice_layer_rbsp( )	VCL
10	RADL_NUT	Coded slice of a RADL picture slice_layer_rbsp( )	VCL
11	RASL_NUT	Coded slice of a RASL picture slice_layer_rbsp( )	VCL
12..15	RSV_VCL_12..	Reserved non-IRAP VCL NAL unit types	VCL

	RSV_VCL_15		
16	DPS_NUT	Decoding parameter set decoding_parameter_set_rbsp( )	non-VCL
17	SPS_NUT	Sequence parameter set seq_parameter_set_rbsp( )	non-VCL
18	EOS_NUT	End of sequence end_of_seq_rbsp( )	non-VCL
19	EOB_NUT	End of bitstream end_of_bitstream_rbsp( )	non-VCL
20	VPS_NUT	Video parameter set video_parameter_set_rbsp( )	non-VCL
21..23	RSV_NVCL21.. RSV_NVCL23	Reserved	non-VCL
24 25	IDR_W_RADL IDR_N_LP	Coded slice of an IDR picture slice_layer_rbsp( )	VCL
26	CRA_NUT	Coded slice of a CRA picture slice_layer_rbsp( )	VCL
27	GRA_NUT	Coded slice of a gradual random access picture slice_layer_rbsp( )	VCL
28..31	UNSPEC28.. UNSPEC31	Unspecified	non-VCL

Table 2

NOTE – A clean random access (CRA) picture may have associated RASL or RADL pictures present in the bitstream.

NOTE – An instantaneous decoding refresh (IDR) picture having NalUnitType equal to IDR_N_LP does not have associated leading pictures present in the bitstream. An IDR picture having NalUnitType equal to IDR_W_RADL does not have associated RASL pictures present in the bitstream, but may have associated RADL pictures in the bitstream.

[0044] It should be noted that generally, for example with respect to ITU-T H.265, an IRAP a picture is a picture that does not refer to any pictures other than itself for inter prediction in its decoding process. Typically, the first picture in the bitstream in decoding order must be an IRAP picture. In ITU-T H.265, an IRAP picture may be a broken link access (BLA) picture, a clean random access (CRA) picture or an instantaneous decoder refresh (IDR) picture. ITU-T H.265 describes the concept of a leading picture, which is a picture that precedes the associated IRAP picture in output order. ITU-T H.265 further describes the concept of a trailing picture which is a non-IRAP picture that follows the associated IRAP picture in output order. Trailing pictures associated with an IRAP picture also follow the IRAP picture in decoding order. For IDR pictures, there are no trailing pictures that require reference to a picture decoded prior to the IDR picture. ITU-T H.265 provides where a CRA picture may have leading pictures that follow the CRA picture in decoding order and contain inter picture prediction references to pictures decoded prior to the CRA picture. Thus, when the CRA picture is used as a random access point these leading pictures may not be decodable and are identified as random access skipped leading (RASL) pictures. BLA pictures may also be followed by RASL pictures. These RASL pictures are always discarded for BLA pictures and discarded for CRA pictures when they are non-decodable, i.e., when a decoder that starts its decoding process at a CRA point. The other type of picture that can follow an IRAP picture in decoding order and precede it in output order is the random access decodable leading (RADL) picture, which cannot contain references to any pictures that precede the IRAP picture in decoding order.

[0045] As described above, in JVET-N1001, non-VCL NAL units include respective parameter set NAL units. Table 3 illustrates the sequence parameter set syntax provided in JVET-N1001.

seq_parameter_set_rbsp( ) {	Descriptor
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_reserved_zero_5bits	u(5)
profile_tier_level(sps_max_sub_layers_minus1)	
gra_enabled_flag	u(1)
sps_seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	u(1)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
conformance_window_flag	u(1)
if(conformance_window_flag) {	
conf_win_left_offset	ue(v)
conf_win_right_offset	ue(v)
conf_win_top_offset	ue(v)
conf_win_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
log2_max_pic_order_cnt_lsb_minus4	ue(v)
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1); i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_minus1[i]	ue(v)
sps_max_num_reorder_pics[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)

}	
long_term_ref_pics_flag	u(1)
sps_idr_rpl_present_flag	u(1)
rpl1_same_as_rpl0_flag	u(1)
for(i = 0; i < !rpl1_same_as_rpl0_flag ? 2 : 1; i++) {	
num_ref_pic_lists_in_sps[i]	ue(v)
for(j = 0; j < num_ref_pic_lists_in_sps[i]; j++)	
ref_pic_list_struct(i, j)	
}	
qtbtt_dual_tree_intra_flag	u(1)
log2_ctu_size_minus2	ue(v)
log2_min_luma_coding_block_size_minus2	ue(v)
partition_constraints_override_enabled_flag	u(1)
sps_log2_diff_min_qt_min_cb_intra_slice_luma	ue(v)
sps_log2_diff_min_qt_min_cb_inter_slice	ue(v)
sps_max_mtt_hierarchy_depth_inter_slice	ue(v)
sps_max_mtt_hierarchy_depth_intra_slice_luma	ue(v)
if(sps_max_mtt_hierarchy_depth_intra_slice_luma != 0) {	
sps_log2_diff_max_bt_min_qt_intra_slice_luma	ue(v)
sps_log2_diff_max_tt_min_qt_intra_slice_luma	ue(v)
}	
if(sps_max_mtt_hierarchy_depth_inter_slices != 0) {	
sps_log2_diff_max_bt_min_qt_inter_slice	ue(v)
sps_log2_diff_max_tt_min_qt_inter_slice	ue(v)
}	
if(qtbtt_dual_tree_intra_flag) {	
sps_log2_diff_min_qt_min_cb_intra_slice_chroma	ue(v)
sps_max_mtt_hierarchy_depth_intra_slice_chroma	ue(v)
if (sps_max_mtt_hierarchy_depth_intra_slice_chroma != 0) {	
sps_log2_diff_max_bt_min_qt_intra_slice_chroma	ue(v)
sps_log2_diff_max_tt_min_qt_intra_slice_chroma	ue(v)

}	
}	
sps_sao_enabled_flag	u(1)
sps_alf_enabled_flag	u(1)
sps_pcm_enabled_flag	u(1)
if(sps_pcm_enabled_flag) {	
pcm_sample_bit_depth_luma_minus1	u(4)
pcm_sample_bit_depth_chroma_minus1	u(4)
log2_min_pcm_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_luma_coding_block_size	ue(v)
pcm_loop_filter_disabled_flag	u(1)
}	
if((CtbSizeY / MinCbSizeY + 1) <= (pic_width_in_luma_samples / MinCbSizeY − 1)) {	
sps_ref_wraparound_enabled_flag	u(1)
if(sps_ref_wraparound_enabled_flag)	
sps_ref_wraparound_offset_minus1	ue(v)
}	
sps_temporal_mvp_enabled_flag	u(1)
if(sps_temporal_mvp_enabled_flag)	
sps_sbtmvp_enabled_flag	u(1)
sps_amvr_enabled_flag	u(1)
sps_bdof_enabled_flag	u(1)
sps_smvd_enabled_flag	u(1)
sps_affine_amvr_enabled_flag	u(1)
sps_dmvr_enabled_flag	u(1)
sps_mmvd_enabled_flag	u(1)
sps_isp_enabled_flag	u(1)
sps_mrl_enabled_flag	u(1)
sps_mip_enabled_flag	u(1)
sps_cclm_enabled_flag	u(1)
if(sps_cclm_enabled_flag && chroma_format_idc == 1)	

sps_cclm_colocated_chroma_flag	u(1)
sps_mts_enabled_flag	u(1)
if(sps_mts_enabled_flag) {	
sps_explicit_mts_intra_enabled_flag	u(1)
sps_explicit_mts_inter_enabled_flag	u(1)
}	
sps_sbt_enabled_flag	u(1)
if(sps_sbt_enabled_flag)	
sps_sbt_max_size_64_flag	u(1)
sps_affine_enabled_flag	u(1)
if(sps_affine_enabled_flag)	
sps_affine_type_flag	u(1)
sps_bcw_enabled_flag	u(1)
sps_ibc_enabled_flag	u(1)
sps_ciip_enabled_flag	u(1)
if(sps_mmvd_enabled_flag)	
sps_fpel_mmvd_enabled_flag	u(1)
sps_triangle_enabled_flag	u(1)
sps_lmcs_enabled_flag	u(1)
sps_ladf_enabled_flag	u(1)
if (sps_ladf_enabled_flag) {	
sps_num_ladf_intervals_minus2	u(2)
sps_ladf_lowest_interval_qp_offset	se(v)
for(i = 0; i < sps_num_ladf_intervals_minus2 + 1; i++) {	
sps_ladf_qp_offset[i]	se(v)
sps_ladf_delta_threshold_minus1[i]	ue(v)
}	
}	
timing_info_present_flag	u(1)
if(timing_info_present_flag) {	
num_units_in_tick	u(32)

time_scale	u(32)
hrd_parameters_present_flag	u(1)
if(hrd_parameters_present_flag)	
hrd_parameters(sps_max_sub_layers_minus1)	
}	
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters( )	
sps_extension_flag	u(1)
if(sps_extension_flag)	
while(more_rbsp_data( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

Table 3

[0046]

With respect to Table 3, JVET-N1001 provides the following semantics:

sps_decoding_parameter_set_id, when greater than 0, specifies the value of dps_decoding_parameter_set_id for the DPS referred to by the SPS. When sps_decoding_parameter_set_id is equal to 0, the SPS does not refer to a DPS and no DPS is active when decoding each CVS referring to the SPS.

sps_video_parameter_set_id, when greater than 0, specifies the value of vps_video_parameter_set_id for the VPS referred to by the SPS. When sps_video_parameter_set_id is equal to 0, the SPS does not refer to a VPS and no VPS is active when decoding each CVS referring to the SPS.

sps_max_sub_layers_minus1 plus 1 specifies the maximum number of temporal sub-layers that may be present in each CVS referring to the SPS. The value of sps_max_sub_layers_minus1 shall be in the range of 0 to 6, inclusive.

sps_reserved_zero_5bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for sps_reserved_zero_5bits are reserved for future use by ITU-T | ISO/IEC.

gra_enabled_flag equal to 1 specifies that GRA pictures may be present in CVSs referring to the SPS. gra_enabled_flag equal to 0 specifies that GRA pictures are not present in CVSs referring to the SPS.

sps_seq_parameter_set_id provides an identifier for the SPS for reference by other syntax elements. The value of sps_seq_parameter_set_id shall be in the range of 0 to 15, inclusive.

chroma_format_idc specifies the chroma sampling relative to the luma sampling as specified. The value of chroma_format_idc shall be in the range of 0 to 3, inclusive.

separate_colour_plane_flag equal to 1 specifies that the three colour components of the 4:4:4 chroma format are coded separately. separate_colour_plane_flag equal to 0 specifies that the colour components are not coded separately. When separate_colour_plane_flag is not present, it is inferred to be equal to 0. When separate_colour_plane_flag is equal to 1, the coded picture consists of three separate components, each of which consists of coded samples of one colour plane (Y, Cb, or Cr) and uses the monochrome coding syntax. In this case, each colour plane is associated with a specific colour_plane_id value.

NOTE – There is no dependency in decoding processes between the colour planes having different colour_plane_id values. For example, the decoding process of a monochrome picture with one value of colour_plane_id does not use any data from monochrome pictures having different values of colour_plane_id for inter prediction.

Depending on the value of separate_colour_plane_flag, the value of the variable ChromaArrayType is assigned as follows:

- If separate_colour_plane_flag is equal to 0, ChromaArrayType is set equal to chroma_format_idc.
- Otherwise (separate_colour_plane_flag is equal to 1), ChromaArrayType is set equal to 0.

pic_width_in_luma_samples specifies the width of each decoded picture in units of luma samples. pic_width_in_luma_samples shall not be equal to 0 and shall be an integer multiple of MinCbSizeY.

pic_height_in_luma_samples specifies the height of each decoded picture in units of luma samples. pic_height_in_luma_samples shall not be equal to 0 and shall be an integer multiple of MinCbSizeY.

conformance_window_flag equal to 1 indicates that the conformance cropping window offset parameters follow next in the SPS. conformance_window_flag equal to 0 indicates that the conformance cropping window offset parameters are not present.

conf_win_left_offset, **conf_win_right_offset**, **conf_win_top_offset**, and **conf_win_bottom_offset** specify the samples of the pictures in the CVS that are output from the decoding process, in terms of a rectangular region specified in picture coordinates for output. When conformance_window_flag is equal to 0, the values of conf_win_left_offset, conf_win_right_offset, conf_win_top_offset, and conf_win_bottom_offset are inferred to be equal to 0.

The conformance cropping window contains the luma samples with horizontal picture coordinates from SubWidthC * conf_win_left_offset to pic_width_in_luma_samples − (SubWidthC * conf_win_right_offset + 1) and vertical picture coordinates from SubHeightC * conf_win_top_offset to pic_height_in_luma_samples − (SubHeightC * conf_win_bottom_offset + 1), inclusive.

The value of SubWidthC * (conf_win_left_offset + conf_win_right_offset) shall be less than pic_width_in_luma_samples, and the value of SubHeightC * (conf_win_top_offset + conf_win_bottom_offset) shall be less than pic_height_in_luma_samples.

When ChromaArrayType is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having picture coordinates (x / SubWidthC, y / SubHeightC), where (x, y) are the picture coordinates of the specified luma samples.

NOTE – The conformance cropping window offset parameters are only applied at the output. All internal decoding processes are applied to the uncropped picture size.

bit_depth_luma_minus8 specifies the bit depth of the samples of the luma array $BitDepth_Y$ and the value of the luma quantization parameter range offset $QpBdOffset_Y$ as follows:

$$BitDepth_Y = 8 + bit\_depth\_luma\_minus8$$

$$QpBdOffset_Y = 6 * bit\_depth\_luma\_minus8$$

bit_depth_luma_minus8 shall be in the range of 0 to 8, inclusive.

bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays $BitDepth_C$ and the value of the chroma quantization parameter range offset $QpBdOffset_C$ as follows:

$$BitDepth_C = 8 + bit\_depth\_chroma\_minus8$$

$$QpBdOffset_C = 6 * bit\_depth\_chroma\_minus8$$

bit_depth_chroma_minus8 shall be in the range of 0 to 8, inclusive.

log2_max_pic_order_cnt_lsb_minus4 specifies the value of the variable MaxPicOrderCntLsb that is used in the decoding process for picture order count as follows:

$$MaxPicOrderCntLsb = 2^{(\ log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4\ )}$$

The value of log2_max_pic_order_cnt_lsb_minus4 shall be in the range of 0 to 12, inclusive.

sps_sub_layer_ordering_info_present_flag equal to 1 specifies that sps_max_dec_pic_buffering_minus1[i], sps_max_num_reorder_pics[i], and sps_max_latency_increase_plus1[i] are present for sps_max_sub_layers_minus1 + 1 sub-layers. sps_sub_layer_ordering_info_present_flag equal to 0 specifies that the values of sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1], sps_max_num_reorder_pics[sps_max_sub_layers_minus1], and sps_max_latency_increase_plus1[sps_max_sub_layers_minus1] apply to all sub-layers.

sps_max_dec_pic_buffering_minus1[i] plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i. The value of sps_max_dec_pic_buffering_minus1[i] shall be in the range of 0 to MaxDpbSize − 1, inclusive, where MaxDpbSize is as specified somewhere else. When i is greater than 0, sps_max_dec_pic_buffering_minus1[i] shall be greater than or equal to sps_max_dec_pic_buffering_minus1[i − 1]. When sps_max_dec_pic_buffering_minus1[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1].

sps_max_num_reorder_pics[i] indicates the maximum allowed number of pictures that can precede any picture in the CVS in decoding order and follow that picture in output order when HighestTid is equal to i. The value of sps_max_num_reorder_pics[i] shall be in the range of 0 to sps_max_dec_pic_buffering_minus1[i], inclusive. When i is greater than 0, sps_max_num_reorder_pics[i] shall be greater than or equal to sps_max_num_reorder_pics[i − 1]. When sps_max_num_reorder_pics[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_num_reorder_pics[sps_max_sub_layers_minus1].

sps_max_latency_increase_plus1[i] not equal to 0 is used to compute the value of SpsMaxLatencyPictures[i], which specifies the maximum number of pictures that can precede any picture in the CVS in output order and follow that picture in decoding order when HighestTid is equal to i.

When sps_max_latency_increase_plus1[i] is not equal to 0, the value of SpsMaxLatencyPictures[i] is specified as follows:

SpsMaxLatencyPictures[i] =
sps_max_num_reorder_pics[i] + sps_max_latency_increase_plus1[i] − 1

When sps_max_latency_increase_plus1[i] is equal to 0, no corresponding limit is expressed.

The value of sps_max_latency_increase_plus1[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. When sps_max_latency_increase_plus1[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_latency_increase_plus1[sps_max_sub_layers_minus1].

long_term_ref_pics_flag equal to 0 specifies that no LTRP is used for inter prediction of any coded picture in the CVS. long_term_ref_pics_flag equal to 1 specifies that LTRPs may be used for inter prediction of one or more coded pictures in the CVS.

sps_idr_rpl_present_flag equal to 1 specifies that reference picture list syntax elements are present in slice headers of IDR pictures. sps_idr_rpl_present_flag equal to 0 specifies that reference picture list syntax elements are not present in slice headers of IDR pictures.

rpl1_same_as_rpl0_flag equal to 1 specifies that the syntax structures num_ref_pic_lists_in_sps[1] and ref_pic_list_struct(1, rplsIdx) are not present and the following applies:

- The value of num_ref_pic_lists_in_sps[1] is inferred to be equal to the value of num_ref_pic_lists_in_sps[0].
- The value of each of syntax elements in ref_pic_list_struct(1, rplsIdx) is inferred to be equal to the value of corresponding syntax element in ref_pic_list_struct(0, rplsIdx) for rplsIdx ranging from 0 to num_ref_pic_lists_in_sps[0] − 1.

num_ref_pic_lists_in_sps[i] specifies the number of the ref_pic_list_struct(listIdx, rplsIdx) syntax structures with listIdx equal to i included in the SPS. The value of num_ref_pic_lists_in_sps[i] shall be in the range of 0 to 64, inclusive.

NOTE – For each value of listIdx (equal to 0 or 1), a decoder should allocate memory for a total number of num_ref_pic_lists_in_sps[i] + 1 ref_pic_list_struct(listIdx, rplsIdx) syntax structures since there may be one ref_pic_list_struct(listIdx, rplsIdx) syntax structure directly signalled in the slice headers of a current picture.

qtbtt_dual_tree_intra_flag equal to 1 specifies that for I slices, each CTU is split into coding units with 64x64 luma samples using an implicit quadtree split and that these coding units are the root of two separate coding_tree syntax structure for luma and chroma.

log2_ctu_size_minus2 plus 2 specifies the luma coding tree block size of each CTU.

log2_min_luma_coding_block_size_minus2 plus 2 specifies the minimum luma coding block size.

The variables CtbLog2SizeY, CtbSizeY, MinCbLog2SizeY, MinCbSizeY, MinTbLog2SizeY, MaxTbLog2SizeY, MinTbSizeY, MaxTbSizeY, PicWidthInCtbsY, PicHeightInCtbsY, PicSizeInCtbsY, PicWidthInMinCbsY, PicHeightInMinCbsY, PicSizeInMinCbsY, PicSizeInSamplesY, PicWidthInSamplesC and PicHeightInSamplesC are derived as follows:

CtbLog2SizeY = log2_ctu_size_minus2 + 2

CtbSizeY = 1 << CtbLog2SizeY

MinCbLog2SizeY = log2_min_luma_coding_block_size_minus2 + 2

MinCbSizeY = 1 << MinCbLog2SizeY

MinTbLog2SizeY = 2

MaxTbLog2SizeY = 6

MinTbSizeY = 1 << MinTbLog2SizeY

MaxTbSizeY = 1 << MaxTbLog2SizeY

PicWidthInCtbsY = Ceil(pic_width_in_luma_samples ÷ CtbSizeY)

PicHeightInCtbsY = Ceil(pic_height_in_luma_samples ÷ CtbSizeY)

PicSizeInCtbsY = PicWidthInCtbsY * PicHeightInCtbsY

PicWidthInMinCbsY = pic_width_in_luma_samples / MinCbSizeY

PicHeightInMinCbsY = pic_height_in_luma_samples / MinCbSizeY

PicSizeInMinCbsY = PicWidthInMinCbsY * PicHeightInMinCbsY

PicSizeInSamplesY = pic_width_in_luma_samples * pic_height_in_luma_samples

PicWidthInSamplesC = pic_width_in_luma_samples / SubWidthC

PicHeightInSamplesC = pic_height_in_luma_samples / SubHeightC

The variables CtbWidthC and CtbHeightC, which specify the width and height, respectively, of the array for each chroma CTB, are derived as follows:

- If chroma_format_idc is equal to 0 (monochrome) or separate_colour_plane_flag is equal to 1, CtbWidthC and CtbHeightC are both equal to 0.
- Otherwise, CtbWidthC and CtbHeightC are derived as follows:

  CtbWidthC = CtbSizeY / SubWidthC

  CtbHeightC = CtbSizeY / SubHeightC

For log2BlockWidth ranging from 0 to 4 and for log2BlockHeight ranging from 0 to 4, inclusive, the up-right diagonal scan order array initialization process as specified is invoked with 1 << log2BlockWidth and 1 << log2BlockHeight as inputs, and the output is assigned to DiagScanOrder[log2BlockWidth][log2BlockHeight].

partition_constraints_override_enabled_flag equal to 1 specifies the presence of partition_constraints_override_flag in the slice headers for slices referring to the SPS. partition_constraints_override_enabled_flag equal to 0 specifies the absence of partition_constraints_override_ flag in the slice headers for slices referring to the SPS.

sps_log2_diff_min_qt_min_cb_intra_slice_luma specifies the default difference between the base 2 logarithm of the minimum size in luma samples of a luma leaf block resulting from quadtree splitting of a CTU and the base 2 logarithm of the minimum coding block size in luma samples for luma CUs in slices with slice_type equal to 2 (I) referring to the SPS. When partition_constraints_override_ flag is equal to 1, the default difference can be overridden by slice_log2_diff_min_qt_min_cb_luma present in the slice header of the slices referring to the SPS. The value of sps_log2_diff_min_qt_min_cb_intra_slice_luma shall be in the range of 0 to CtbLog2SizeY − MinCbLog2SizeY, inclusive. The base 2 logarithm of the minimum size in luma samples of a luma leaf block resulting from quadtree splitting of a CTU is derived as follows:

MinQtLog2SizeIntraY = sps_log2_diff_min_qt_min_cb_intra_slice_luma + MinCbLog2SizeY

sps_log2_diff_min_qt_min_cb_inter_slice specifies the default difference between the base 2 logarithm of the minimum size in luma samples of a luma leaf block resulting from quadtree splitting of a CTU and the base 2 logarithm of the minimum luma coding block size in luma samples for luma CUs in slices with slice_type equal to 0 (B) or 1 (P) referring to the SPS. When partition_constraints_override_ flag is equal to 1, the default difference can be overridden by slice_log2_diff_min_qt_min_cb_luma present in the slice header of the slices referring to the SPS. The value of sps_log2_diff_min_qt_min_cb_inter_slice shall be in the range of 0 to CtbLog2SizeY − MinCbLog2SizeY, inclusive. The base 2 logarithm of the minimum size in luma samples of a luma leaf block resulting from quadtree splitting of a CTU is derived as follows:

MinQtLog2SizeInterY = sps_log2_diff_min_qt_min_cb_inter_slice + MinCbLog2SizeY

sps_max_mtt_hierarchy_depth_inter_slice specifies the default maximum hierarchy depth for coding units resulting from multi-type tree splitting of a quadtree leaf in slices with slice_type equal to 0 (B) or 1 (P) referring to the SPS. When partition_constraints_override_ flag is equal to 1, the default maximum hierarchy depth can be overridden by slice_max_mtt_hierarchy_depth_luma present in the slice header of the slices referring to the SPS. The value of sps_max_mtt_hierarchy_depth_inter_slice shall be in the range of 0 to CtbLog2SizeY − MinCbLog2SizeY, inclusive.

sps_max_mtt_hierarchy_depth_intra_slice_luma specifies the default maximum hierarchy depth for coding units resulting from multi-type tree splitting of a quadtree leaf in slices with slice_type equal to 2 (I) referring to the SPS. When partition_constraints_override_ flag is equal to 1, the default maximum hierarchy depth can be overridden by slice_max_mtt_hierarchy_depth_luma present in the slice header of the slices referring to the SPS. The value of sps_max_mtt_hierarchy_depth_intra_slice_luma shall be in the range of 0 to CtbLog2SizeY − MinCbLog2SizeY, inclusive.

sps_log2_diff_max_bt_min_qt_intra_slice_luma specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a binary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in slices with slice_type equal to 2 (I) referring to the SPS. When partition_constraints_override_ flag is equal to 1, the default difference can be overridden by slice_log2_diff_max_bt_min_qt_luma present in the slice header of the slices referring to the SPS. The value of sps_log2_diff_max_bt_min_qt_intra_slice_luma shall be in the range of 0 to CtbLog2SizeY − MinQtLog2SizeIntraY, inclusive. When sps_log2_diff_max_bt_min_qt_intra_slice_luma is not present, the value of sps_log2_diff_max_bt_min_qt_intra_slice_luma is inferred to be equal to 0.

sps_log2_diff_max_tt_min_qt_intra_slice_luma specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a ternary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in slices with slice_type equal to 2 (I) referring to the SPS. When partition_constraints_override_ flag is equal to 1, the default difference can be overridden by slice_log2_diff_max_tt_min_qt_luma present in the slice header of the slices referring to the SPS. The value of sps_log2_diff_max_tt_min_qt_intra_slice_luma shall be in the range of 0 to CtbLog2SizeY − MinQtLog2SizeIntraY, inclusive. When sps_log2_diff_max_tt_min_qt_intra_slice_luma is not present, the value of sps_log2_diff_max_tt_min_qt_intra_slice_luma is inferred to be equal to 0.

sps_log2_diff_max_bt_min_qt_inter_slice specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a binary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in slices with slice_type equal to 0 (B) or 1 (P) referring to the SPS. When partition_constraints_override_ flag is equal to 1, the default difference can be overridden by slice_log2_diff_max_bt_min_qt_luma present in the slice header of the slices referring to the SPS. The value of sps_log2_diff_max_bt_min_qt_inter_slice shall be in the range of 0 to CtbLog2SizeY − MinQtLog2SizeInterY, inclusive. When sps_log2_diff_max_bt_min_qt_inter_slice is not present, the value of sps_log2_diff_max_bt_min_qt_inter_slice is inferred to be equal to 0.

sps_log2_diff_max_tt_min_qt_inter_slice specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a ternary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in slices with slice_type equal to 0 (B) or 1 (P) referring to the SPS. When partition_constraints_override_ flag is equal to 1, the default difference can be overridden by slice_log2_diff_max_tt_min_qt_luma present in the slice header of the slices referring to the SPS. The value of sps_log2_diff_max_tt_min_qt_inter_slice shall be in the range of 0 to CtbLog2SizeY − MinQtLog2SizeInterY, inclusive. When sps_log2_diff_max_tt_min_qt_inter_slice is not present, the value of sps_log2_diff_max_tt_min_qt_inter_slice is inferred to be equal to 0.

sps_log2_diff_min_qt_min_cb_intra_slice_chroma specifies the default difference between the base 2 logarithm of the minimum size in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA and the base 2 logarithm of the minimum coding block size in luma samples for chroma CUs with treeType equal to DUAL_TREE_CHROMA in slices with slice_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be overridden by slice_log2_diff_min_qt_min_cb_chroma present in the slice header of the slices referring to the SPS. The value of sps_log2_diff_min_qt_min_cb_intra_slice_chroma shall be in the range of 0 to CtbLog2SizeY − MinCbLog2SizeY, inclusive. When not present, the value of sps_log2_diff_min_qt_min_cb_intra_slice_chroma is inferred to be equal to 0. The base 2 logarithm of the minimum size in luma samples of a chroma leaf block resulting from quadtree splitting of a CTU with treeType equal to DUAL_TREE_CHROMA is derived as follows:

MinQtLog2SizeIntraC = sps_log2_diff_min_qt_min_cb_intra_slice_chroma + MinCbLog2SizeY

sps_max_mtt_hierarchy_depth_intra_slice_chroma specifies the default maximum hierarchy depth for chroma coding units resulting from multi-type tree splitting of a chroma quadtree leaf with treeType equal to DUAL_TREE_CHROMA in slices with slice_type equal to 2 (I) referring to the SPS. When partition_constraints_override_ flag is equal to 1, the default maximum hierarchy depth can be overridden by slice_max_mtt_hierarchy_depth_chroma present in the slice header of the slices referring to the SPS. The value of sps_max_mtt_hierarchy_depth_intra_slice_chroma shall be in the range of 0 to CtbLog2SizeY − MinCbLog2SizeY, inclusive. When not present, the value of sps_max_mtt_hierarchy_depth_intra_slice_chroma is inferred to be equal to 0.

sps_log2_diff_max_bt_min_qt_intra_slice_chroma specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a chroma coding block that can be split using a binary split and the minimum size (width or height) in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA in slices with slice_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be overridden by slice_log2_diff_max_bt_min_qt_chroma present in the slice header of the slices referring to the SPS. The value of sps_log2_diff_max_bt_min_qt_intra_slice_chroma shall be in the range of 0 to CtbLog2SizeY − MinQtLog2SizeIntraC, inclusive. When sps_log2_diff_max_bt_min_qt_intra_slice_chroma is not present, the value of sps_log2_diff_max_bt_min_qt_intra_slice_chroma is inferred to be equal to 0.

sps_log2_diff_max_tt_min_qt_intra_slice_chroma specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a chroma coding block that can be split using a ternary split and the minimum size (width or height) in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA in slices with slice_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be overridden by slice_log2_diff_max_tt_min_qt_chroma present in the slice header of the slices referring to the SPS. The value of sps_log2_diff_max_tt_min_qt_intra_slice_chroma shall be in the range of 0 to CtbLog2SizeY − MinQtLog2SizeIntraC, inclusive. When sps_log2_diff_max_tt_min_qt_intra_slice_chroma is not present, the value of sps_log2_diff_max_tt_min_qt_intra_slice_chroma is inferred to be equal to 0.

sps_sao_enabled_flag equal to 1 specifies that the sample adaptive offset process is applied to the reconstructed picture after the deblocking filter process. sps_sao_enabled_flag equal to 0 specifies that the sample adaptive offset process is not applied to the reconstructed picture after the deblocking filter process.

sps_alf_enabled_flag equal to 0 specifies that the adaptive loop filter is disabled. sps_alf_enabled_flag equal to 1 specifies that the adaptive loop filter is enabled.

sps_pcm_enabled_flag equal to 0 specifies that PCM-related syntax (pcm_sample_bit_depth_luma_minus1, pcm_sample_bit_depth_chroma_minus1, log2_min_pcm_luma_coding_block_size_minus3, log2_diff_max_min_pcm_luma_coding_block_size, pcm_loop_filter_disabled_flag, pcm_flag, pcm_alignment_zero_bit syntax elements and pcm_sample( ) syntax structure) is not present in the CVS.

NOTE – When MinCbLog2SizeY is equal to 6 and sps_pcm_enabled_flag is equal to 1, PCM sample data-related syntax (pcm_flag, pcm_alignment_zero_bit syntax elements and pcm_sample( ) syntax structure) is not present in the CVS, because the maximum size of coding blocks that can convey PCM sample data-related syntax is restricted to be less than or equal to Min(CtbLog2SizeY, 5). Hence, MinCbLog2SizeY equal to 6 with sps_pcm_enabled_flag equal to 1 is not an appropriate setting to convey PCM sample data in the CVS.

pcm_sample_bit_depth_luma_minus1 specifies the number of bits used to represent each of PCM sample values of the luma component as follows:

$$PcmBitDepth_Y = pcm\_sample\_bit\_depth\_luma\_minus1 + 1$$

The value of $PcmBitDepth_Y$ shall be less than or equal to the value of $BitDepth_Y$.

pcm_sample_bit_depth_chroma_minus1 specifies the number of bits used to represent each of PCM sample values of the chroma components as follows:

$$PcmBitDepth_C = pcm\_sample\_bit\_depth\_chroma\_minus1 + 1$$

The value of $PcmBitDepth_C$ shall be less than or equal to the value of $BitDepth_C$. When ChromaArrayType is equal to 0, pcm_sample_bit_depth_chroma_minus1 is not used in the decoding process and decoders shall ignore its value.

log2_min_pcm_luma_coding_block_size_minus3 plus 3 specifies the minimum size of coding blocks with pcm_flag equal to 1.

The variable Log2MinIpcmCbSizeY is set equal to log2_min_pcm_luma_coding_block_size_minus3 + 3. The value of Log2MinIpcmCbSizeY shall be in the range of Min(MinCbLog2SizeY, 5) to Min(CtbLog2SizeY, 5), inclusive.

log2_diff_max_min_pcm_luma_coding_block_size specifies the difference between the maximum and minimum size of coding blocks with pcm_flag equal to 1.

The variable Log2MaxIpcmCbSizeY is set equal to log2_diff_max_min_pcm_luma_coding_block_size + Log2MinIpcmCbSizeY. The value of Log2MaxIpcmCbSizeY shall be less than or equal to Min(CtbLog2SizeY, 5).

pcm_loop_filter_disabled_flag specifies whether the loop filter process is disabled on reconstructed samples in a coding unit with pcm_flag equal to 1 as follows:

- If pcm_loop_filter_disabled_flag is equal to 1, the deblocking filter, sample adaptive offset filter, and adaptive loop filter processes on the reconstructed samples in a coding unit with pcm_flag equal to 1 are disabled.
- Otherwise (pcm_loop_filter_disabled_flag value is equal to 0), the deblocking filter, sample adaptive offset filter, and adaptive loop filter processes on the reconstructed samples in a coding unit with pcm_flag equal to 1 are not disabled.

When pcm_loop_filter_disabled_flag is not present, it is inferred to be equal to 0.

sps_ref_wraparound_enabled_flag equal to 1 specifies that horizontal wrap-around motion compensation is applied in inter prediction. sps_ref_wraparound_enabled_flag equal to 0 specifies that horizontal wrap-around motion compensation is not applied. When not present, the value of sps_ref_wraparound_enabled_flag is inferred to be equal to 0.

sps_ref_wraparound_offset_minus1 plus 1 specifies the offset used for computing the horizontal wrap-around position in units of MinCbSizeY luma samples. The value of ref_wraparound_offset_minus1 shall be in the range of (CtbSizeY / MinCbSizeY) + 1 to (pic_width_in_luma_samples / MinCbSizeY) − 1, inclusive.

sps_temporal_mvp_enabled_flag equal to 1 specifies that slice_temporal_mvp_enabled_flag is present in the slice headers of slices with slice_type not equal to I in the CVS. sps_temporal_mvp_enabled_flag equal to 0 specifies that slice_temporal_mvp_enabled_flag is not present in slice headers and that temporal motion vector predictors are not used in the CVS.

sps_sbtmvp_enabled_flag equal to 1 specifies that subblock-based temporal motion vector predictors may be used in decoding of pictures with all slices having slice_type not equal to I in the CVS. sps_sbtmvp_enabled_flag equal to 0 specifies that subblock-based temporal motion vector predictors are not used in the CVS. When sps_sbtmvp_enabled_flag is not present, it is inferred to be equal to 0.

sps_amvr_enabled_flag equal to 1 specifies that adaptive motion vector difference resolution is used in motion vector coding. amvr_enabled_flag equal to 0 specifies that adaptive motion vector difference resolution is not used in motion vector coding.

sps_bdof_enabled_flag equal to 0 specifies that the bidirectional optical flow inter prediction is disabled. sps_bdof_enabled_flag equal to 1 specifies that the bidirectional optical flow inter prediction is enabled.

sps_smvd_enabled_flag equal to 1 specifies that symmetric motion vector difference may be used in motion vector decoding. sps_smvd_enabled_flag equal to 0 specifies that symmetric motion vector difference is not used in motion vector coding.

sps_affine_amvr_enabled_flag equal to 1 specifies that adaptive motion vector difference resolution is used in motion vector coding of affine inter mode. sps_affine_amvr_enabled_flag equal to 0 specifies that adaptive motion vector difference resolution is not used in motion vector coding of affine inter mode.

sps_dmvr_enabled_flag equal to 1 specifies that decoder motion vector refinement based inter bi-prediction is enabled. sps_dmvr_enabled_flag equal to 0 specifies that decoder motion vector refinement based inter bi-prediction is disabled.

sps_mmvd_enabled_flag equal to 1 specifies that merge mode with motion vector difference is enabled. sps_mmvd_enabled_flag equal to 0 specifies that merge mode with motion vector difference is disabled.

sps_isp_enabled_flag equal to 1 specifies that intra prediction with subpartitions is enabled. sps_isp_enabled_flag equal to 0 specifies that intra prediction with subpartitions is disabled.

sps_mrl_enabled_flag equal to 1 specifies that intra prediction with multiple reference lines is enabled. sps_mrl_enabled_flag equal to 0 specifies that intra prediction with multiple reference lines is disabled.

sps_mip_enabled_flag equal to 1 specifies that matrix-based intra prediction is enabled. sps_mrl_enabled_flag equal to 0 specifies that matrix-based intra prediction is disabled.

sps_cclm_enabled_flag equal to 0 specifies that the cross-component linear model intra prediction from luma component to chroma component is disabled. sps_cclm_enabled_flag equal to 1 specifies that the cross-component linear model intra prediction from luma component to chroma componenent is enabled.

sps_cclm_colocated_chroma_flag equal to 1 specifies that the top-left downsampled luma sample in cross-component linear model intra prediction is collocated with the top-left luma sample. sps_cclm_colocated_chroma_flag equal to 0 specifies that the top-left downsampled luma sample in cross-component linear model intra prediction is horizontally co-sited with the top-left luma sample but vertically shifted by 0.5 units of luma samples relatively to the top-left luma sample.

sps_mts_enabled_flag equal to 1 specifies that sps_explicit_mts_intra_enabled_flag is present in the sequence parameter set RBSP syntax and that sps_explicit_mts_inter_enabled_flag is present in the sequence parameter set RBSP syntax. sps_mts_enabled_flag equal to 0 specifies that sps_explicit_mts_intra_enabled_flag is not present in the sequence parameter set RBSP syntax and that sps_explicit_mts_inter_enabled_flag is not present in the sequence parameter set RBSP syntax.

sps_explicit_mts_intra_enabled_flag equal to 1 specifies that tu_mts_idx may be present in the transform unit syntax for intra coding units. sps_explicit_mts_intra_enabled_flag equal to 0 specifies that tu_mts_idx is not present in the transform unit syntax for intra coding units. When not present, the value of sps_explicit_mts_intra_enabled_flag is inferred to be equal to 0.

sps_explicit_mts_inter_enabled_flag equal to 1 specifies that tu_mts_idx may be present in the transform unit syntax for inter coding units. sps_explicit_mts_inter_enabled_flag equal to 0 specifies that tu_mts_idx is not present in the transform unit syntax for inter coding units. When not present, the value of sps_explicit_mts_inter_enabled_flag is inferred to be equal to 0.

sps_sbt_enabled_flag equal to 0 specifies that subblock transform for inter-predicted CUs is disabled. sps_sbt_enabled_flag equal to 1 specifies that subblock transform for inter-predicteds CU is enabled.

sps_sbt_max_size_64_flag equal to 0 specifies that the maximum CU width and height for allowing subblock transform is 32 luma samples. sps_sbt_max_size_64_flag equal to 1 specifies that the maximum CU width and height for allowing subblock transform is 64 luma samples.

MaxSbtSize = sps_sbt_max_size_64_flag ? 64 : 32

sps_affine_enabled_flag specifies whether affine model based motion compensation can be used for inter prediction. If sps_affine_enabled_flag is equal to 0, the syntax shall be constrained such that no affine model based motion compensation is used in the CVS, and inter_affine_flag and cu_affine_type_flag are not present in coding unit syntax of the CVS. Otherwise (sps_affine_enabled_flag is equal to 1), affine model based motion compensation can be used in the CVS.

sps_affine_type_flag specifies whether 6-parameter affine model based motion compensation can be used for inter prediction. If sps_affine_type_flag is equal to 0, the syntax shall be constrained such that no 6-parameter affine model based motion compensation is used in the CVS, and cu_affine_type_flag is not present in coding unit syntax in the CVS. Otherwise (sps_affine_type_flag is equal to 1), 6-parameter affine model based motion compensation can be used in the CVS. When not present, the value of sps_affine_type_flag is inferred to be equal to 0.

sps_bcw_enabled_flag specifies whether bi-prediction with CU weights can be used for inter prediction. If sps_bcw_enabled_flag is equal to 0, the syntax shall be constrained such that no bi-prediction with CU weights is used in the CVS, and bcw_idx is not present in coding unit syntax of the CVS. Otherwise (sps_bcw_enabled_flag is equal to 1), bi-prediction with CU weights can be used in the CVS.

sps_ibc_enabled_flag equal to 1 specifies that current picture referencing may be used in decoding of pictures in the CVS. sps_ibc_enabled_flag equal to 0 specifies that current picture referencing is not used in the CVS. When sps_ibc_enabled_flag is not present, it is inferred to be equal to 0.

sps_ciip_enabled_flag specifies that ciip_flag may be present in the coding unit syntax for inter coding units. sps_ciip_enabled_flag equal to 0 specifies that ciip_flag is not present in the coding unit syntax for inter coding units.

sps_fpel_mmvd_enabled_flag equal to 1 specifies that merge mode with motion vector difference is using integer sample precision. sps_fpel_mmvd_enabled_flag equal to 0 specifies that merge mode with motion vector difference can use fractional sample precision.

sps_triangle_enabled_flag specifies whether triangular shape based motion compensation can be used for inter prediction. sps_triangle_enabled_flag equal to 0 specifies that the syntax shall be constrained such that no triangular shape based motion compensation is used in the CVS, and merge_triangle_split_dir, merge_triangle_idx0, and merge_triangle_idx1 are not present in coding unit syntax of the CVS. sps_triangle_enabled_flag equal to 1 specifies that triangular shape based motion compensation can be used in the CVS.

sps_lmcs_enabled_flag equal to 1 specifies that luma mapping with chroma scaling is used in the CVS. sps_lmcs_enabled_flag equal to 0 specifies that luma mapping with chroma scaling is not used in the CVS.

sps_ladf_enabled_flag equal to 1, specifies that sps_num_ladf_intervals_minus2, sps_ladf_lowest_interval_qp_offset, sps_ladf_qp_offset[i], and sps_ladf_delta_threshold_minus1[i] are present in the SPS.

sps_num_ladf_intervals_minus2 plus 1 specifies the number of sps_ladf_delta_threshold_minus1[i] and sps_ladf_qp_offset[i] syntax elements that are present in the SPS. The value of sps_num_ladf_intervals_minus2 shall be in the range of 0 to 3, inclusive.

sps_ladf_lowest_interval_qp_offset specifies the offset used to derive the variable qP as specified. The value of sps_ladf_lowest_interval_qp_offset shall be in the range of 0 to 63, inclusive.

sps_ladf_qp_offset[i] specifies the offset array used to derive the variable qP as specified. The value of sps_ladf_qp_offset[i] shall be in the range of 0 to 63, inclusive.

sps_ladf_delta_threshold_minus1[i] is used to compute the values of SpsLadfIntervalLowerBound[i], which specifies the lower bound of the i-th luma intensity level interval. The value of sps_ladf_delta_threshold_minus1[i] shall be in the range of 0 to $2^{BitDepthY} - 3$, inclusive.

The value of SpsLadfIntervalLowerBound[0] is set equal to 0.

For each value of i in the range of 0 to sps_num_ladf_intervals_minus2, inclusive, the variable SpsLadfIntervalLowerBound[i + 1] is derived as follows:

$$\begin{aligned} SpsLadfIntervalLowerBound[\ i + 1\ ] = {} & SpsLadfIntervalLowerBound[\ i\ ] \\ & + sps\_ladf\_delta\_threshold\_minus1[\ i\ ] + 1 \end{aligned}$$

timing_info_present_flag equal to 1 specifies that the syntax elemnts num_units_in_tick, time_scale, and hrd_parameters_present_flag are present in the SPS RBSP syntax structure. timing_info_present_flag equal to 0 specifies that num_units_in_tick, time_scale, and hrd_parameters_present_flag are not present in the SPS RBSP syntax structure.

num_units_in_tick is the number of time units of a clock operating at the frequency time_scale Hz that corresponds to one increment (called a clock tick) of a clock tick counter. num_units_in_tick shall be greater than 0. A clock tick, in units of seconds, is equal to the quotient of num_units_in_tick divided by time_scale. For example, when the picture rate of a video signal is 25 Hz, time_scale may be equal to 27 000 000 and num_units_in_tick may be equal to 1 080 000, and consequently a clock tick may be equal to 0.04 seconds.

time_scale is the number of time units that pass in one second. For example, a time coordinate system that measures time using a 27 MHz clock has a time_scale of 27 000 000. The value of time_scale shall be greater than 0.

hrd_parameters_present_flag equal to 1 specifies that the syntax structure hrd_parameters( ) is present in the SPS RBSP syntax structure. hrd_parameters_present_flag equal to 0 specifies that the syntax structure hrd_parameters( ) is not present in the SPS RBSP syntax structure.

vui_parameters_present_flag equal to 1 specifies that the syntax structure vui_parameters( ) is present in the SPS RBSP syntax structure. vui_parameters_present_flag equal to 0 specifies that the syntax structure vui_parameters( ) is not present in the SPS RBSP syntax structure.

sps_extension_flag equal to 0 specifies that no sps_extension_data_flag syntax elements are present in the SPS RBSP syntax structure. sps_extension_flag equal to 1 specifies that there are sps_extension_data_flag syntax elements present in the SPS RBSP syntax structure.

sps_extension_data_flag may have any value. Its presence and value do not affect decoder conformance to profiles specified in this version of this Specification. Decoders conforming to this version of this Specification shall ignore all sps_extension_data_flag syntax elements.

[0047]

As described above, during the decoding process, at the onset of decoding a picture, reference picture list(s) are generated from previously decoded pictures stored in a decoded picture buffer (DPB). A parameter set may include information with respect to a DPB. For example, referring to Table 3, in JVET-N1001, an SPS includes syntax element **sps_max_dec_pic_buffering_minus1**[i] which plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i, syntax element **sps_max_num_reorder_pics**[i] which indicates the maximum allowed number of pictures that can precede any picture in the CVS in decoding order and follow that picture in output order when HighestTid is equal to i, and sytnax element **sps_max_latency_increase_plus1**[i] which may be used to specify the maximum number of pictures that can precede any picture in the CVS in output order and follow that picture in decoding order when HighestTid is equal to i. Thus, based on the values of **sps_max_dec_pic_buffering_minus1**, **sps_max_num_reorder_pics**, and/or **sps_max_latency_increase_plus1**[i], a video decoder may allocate resources for a DPB for a coded video sequence, e.g., a coded video sequence having a HighestTid equal to i.

The signaling DPB information in JVET-N1001 is less than ideal. In particular, in JVET-N1001, each of syntax elements **sps_max_dec_pic_buffering_minus1**[i] and **sps_max_num_reorder_pics**[i] are coded using ue(v) coding and there is a strictly monotonic non-decreasing relationship between values for the parameters as a function of index i. That is, when there are multiple sub-layers **sps_max_dec_pic_buffering_minus1**[i] and **sps_max_num_reorder_pics**[i] increase (or remain the same) as i increases. That is, for example, a 60 Hz enhancement layer of video requires a larger DPB than corresponding 30 Hz base layer of video. Thus, according to the signaling in JVET_N1001, **sps_max_dec_pic_buffering_minus1**[i] shall is greater than or equal to **sps_max_dec_pic_buffering_minus1**[i − 1] and **sps_max_num_reorder_pics**[i] is greater than or equal **to** **sps_max_num_reorder_pics**[i − 1]. According to the techniques herein, the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i and the maximum allowed number of pictures that can precede any picture in the CVS in decoding order and follow that picture in output order when HighestTid is equal to i are signaled in a more efficient manner and thus, bit savings may be achieved according to the techniques herein.

[0048] FIG. 1 is a block diagram illustrating an example of a system that may be configured to code (i.e., encode and/or decode) video data according to one or more techniques of this disclosure. System 100 represents an example of a system that may encapsulate video data according to one or more techniques of this disclosure. As illustrated in FIG. 1, system 100 includes source device 102, communications medium 110, and destination device 120. In the example illustrated in FIG. 1, source device 102 may include any device configured to encode video data and transmit encoded video data to communications medium 110. Destination device 120 may include any device configured to receive encoded video data via communications medium 110 and to decode encoded video data. Source device 102 and/or destination device 120 may include computing devices equipped for wired and/or wireless communications and may include, for example, set top boxes, digital video recorders, televisions, desktop, laptop or tablet computers, gaming consoles, medical imagining devices, and mobile devices, including, for example, smartphones, cellular telephones, personal gaming devices.

[0049] Communications medium 110 may include any combination of wireless and wired

communication media, and/or storage devices. Communications medium 110 may include coaxial cables, fiber optic cables, twisted pair cables, wireless transmitters and receivers, routers, switches, repeaters, base stations, or any other equipment that may be useful to facilitate communications between various devices and sites. Communications medium 110 may include one or more networks. For example, communications medium 110 may include a network configured to enable access to the World Wide Web, for example, the Internet. A network may operate according to a combination of one or more telecommunication protocols. Telecommunications protocols may include proprietary aspects and/or may include standardized telecommunication protocols. Examples of standardized telecommunications protocols include Digital Video Broadcasting (DVB) standards, Advanced Television Systems Committee (ATSC) standards, Integrated Services Digital Broadcasting (ISDB) standards, Data Over Cable Service Interface Specification (DOCSIS) standards, Global System Mobile Communications (GSM) standards, code division multiple access (CDMA) standards, 3rd Generation Partnership Project (3GPP) standards, European Telecommunications Standards Institute (ETSI) standards, Internet Protocol (IP) standards, Wireless Application Protocol (WAP) standards, and Institute of Electrical and Electronics Engineers (IEEE) standards.

[0050] Storage devices may include any type of device or storage medium capable of storing data. A storage medium may include a tangible or non-transitory computer-readable media. A computer readable medium may include optical discs, flash memory, magnetic memory, or any other suitable digital storage media. In some examples, a memory device or portions thereof may be described as non-volatile memory and in other examples portions of memory devices may be described as volatile memory. Examples of volatile memories may include random access memories (RAM), dynamic random access memories (DRAM), and static random access memories (SRAM). Examples of non-volatile memories may include magnetic hard discs, optical discs, floppy discs, flash memories, or forms of electrically programmable memories (EPROM) or electrically erasable and programmable (EEPROM) memories. Storage device(s) may include memory cards (e.g., a Secure Digital (SD) memory card), internal/external hard disk drives, and/or internal/external solid state drives. Data may be stored on a storage device according to a defined file format.

[0051] FIG. 4 is a conceptual drawing illustrating an example of components that may be included in an implementation of system 100. In the example implementation illustrated in FIG. 4, system 100 includes one or more computing devices 402A-402N, television service network 404, television service provider site 406, wide area network 408, local area network 410, and one or more content provider sites 412A-412N. The implementation illustrated in FIG. 4 represents an example of a system that may be

configured to allow digital media content, such as, for example, a movie, a live sporting event, etc., and data and applications and media presentations associated therewith to be distributed to and accessed by a plurality of computing devices, such as computing devices 402A-402N. In the example illustrated in FIG. 4, computing devices 402A-402N may include any device configured to receive data from one or more of television service network 404, wide area network 408, and/or local area network 410. For example, computing devices 402A-402N may be equipped for wired and/or wireless communications and may be configured to receive services through one or more data channels and may include televisions, including so-called smart televisions, set top boxes, and digital video recorders. Further, computing devices 402A-402N may include desktop, laptop, or tablet computers, gaming consoles, mobile devices, including, for example, "smart" phones, cellular telephones, and personal gaming devices.

[0052] Television service network 404 is an example of a network configured to enable digital media content, which may include television services, to be distributed. For example, television service network 404 may include public over-the-air television networks, public or subscription-based satellite television service provider networks, and public or subscription-based cable television provider networks and/or over the top or Internet service providers. It should be noted that although in some examples television service network 404 may primarily be used to enable television services to be provided, television service network 404 may also enable other types of data and services to be provided according to any combination of the telecommunication protocols described herein. Further, it should be noted that in some examples, television service network 404 may enable two-way communications between television service provider site 406 and one or more of computing devices 402A-402N. Television service network 404 may comprise any combination of wireless and/or wired communication media. Television service network 404 may include coaxial cables, fiber optic cables, twisted pair cables, wireless transmitters and receivers, routers, switches, repeaters, base stations, or any other equipment that may be useful to facilitate communications between various devices and sites. Television service network 404 may operate according to a combination of one or more telecommunication protocols. Telecommunications protocols may include proprietary aspects and/or may include standardized telecommunication protocols. Examples of standardized telecommunications protocols include DVB standards, ATSC standards, ISDB standards, DTMB standards, DMB standards, Data Over Cable Service Interface Specification (DOCSIS) standards, HbbTV standards, W3C standards, and UPnP standards.

[0053] Referring again to FIG. 4, television service provider site 406 may be configured to

distribute television service via television service network 404. For example, television service provider site 406 may include one or more broadcast stations, a cable television provider, or a satellite television provider, or an Internet-based television provider. For example, television service provider site 406 may be configured to receive a transmission including television programming through a satellite uplink/downlink. Further, as illustrated in FIG. 4, television service provider site 406 may be in communication with wide area network 408 and may be configured to receive data from content provider sites 412A-412N. It should be noted that in some examples, television service provider site 406 may include a television studio and content may originate therefrom.

[0054] Wide area network 408 may include a packet based network and operate according to a combination of one or more telecommunication protocols. Telecommunications protocols may include proprietary aspects and/or may include standardized telecommunication protocols. Examples of standardized telecommunications protocols include Global System Mobile Communications (GSM) standards, code division multiple access (CDMA) standards, 3rd Generation Partnership Project (3GPP) standards, European Telecommunications Standards Institute (ETSI) standards, European standards (EN), IP standards, Wireless Application Protocol (WAP) standards, and Institute of Electrical and Electronics Engineers (IEEE) standards, such as, for example, one or more of the IEEE 802 standards (e.g., Wi-Fi). Wide area network 408 may comprise any combination of wireless and/or wired communication media. Wide area network 408 may include coaxial cables, fiber optic cables, twisted pair cables, Ethernet cables, wireless transmitters and receivers, routers, switches, repeaters, base stations, or any other equipment that may be useful to facilitate communications between various devices and sites. In one example, wide area network 408 may include the Internet. Local area network 410 may include a packet based network and operate according to a combination of one or more telecommunication protocols. Local area network 410 may be distinguished from wide area network 408 based on levels of access and/or physical infrastructure. For example, local area network 410 may include a secure home network.

[0055] Referring again to FIG. 4, content provider sites 412A-412N represent examples of sites that may provide multimedia content to television service provider site 406 and/or computing devices 402A-402N. For example, a content provider site may include a studio having one or more studio content servers configured to provide multimedia files and/or streams to television service provider site 406. In one example, content provider sites 412A-412N may be configured to provide multimedia content using the IP suite. For example, a content provider site may be configured to provide multimedia content to a receiver device according to Real Time Streaming Protocol (RTSP),

HTTP, or the like. Further, content provider sites 412A-412N may be configured to provide data, including hypertext based content, and the like, to one or more of receiver devices computing devices 402A-402N and/or television service provider site 406 through wide area network 408. Content provider sites 412A-412N may include one or more web servers. Data provided by data provider site 412A-412N may be defined according to data formats.

[0056] Referring again to FIG. 1, source device 102 includes video source 104, video encoder 106, data encapsulator 107, and interface 108. Video source 104 may include any device configured to capture and/or store video data. For example, video source 104 may include a video camera and a storage device operably coupled thereto. Video encoder 106 may include any device configured to receive video data and generate a compliant bitstream representing the video data. A compliant bitstream may refer to a bitstream that a video decoder can receive and reproduce video data therefrom. Aspects of a compliant bitstream may be defined according to a video coding standard. When generating a compliant bitstream video encoder 106 may compress video data. Compression may be lossy (discernible or indiscernible to a viewer) or lossless. FIG. 5 is a block diagram illustrating an example of video encoder 500 that may implement the techniques for encoding video data described herein. It should be noted that although example video encoder 500 is illustrated as having distinct functional blocks, such an illustration is for descriptive purposes and does not limit video encoder 500 and/or sub-components thereof to a particular hardware or software architecture. Functions of video encoder 500 may be realized using any combination of hardware, firmware, and/or software implementations.

[0057] Video encoder 500 may perform intra prediction coding and inter prediction coding of picture areas, and, as such, may be referred to as a hybrid video encoder. In the example illustrated in FIG. 5, video encoder 500 receives source video blocks. In some examples, source video blocks may include areas of picture that has been divided according to a coding structure. For example, source video data may include macroblocks, CTUs, CBs, sub-divisions thereof, and/or another equivalent coding unit. In some examples, video encoder 500 may be configured to perform additional sub-divisions of source video blocks. It should be noted that the techniques described herein are generally applicable to video coding, regardless of how source video data is partitioned prior to and/or during encoding. In the example illustrated in FIG. 5, video encoder 500 includes summer 502, transform coefficient generator 504, coefficient quantization unit 506, inverse quantization and transform coefficient processing unit 508, summer 510, intra prediction processing unit 512, inter prediction processing unit 514, filter unit 516, and entropy encoding unit 518. As illustrated in FIG. 5, video encoder 500 receives source video blocks and outputs a bitstream.

[0058] In the example illustrated in FIG. 5, video encoder 500 may generate residual data by subtracting a predictive video block from a source video block. The selection of a predictive video block is described in detail below. Summer 502 represents a component configured to perform this subtraction operation. In one example, the subtraction of video blocks occurs in the pixel domain. Transform coefficient generator 504 applies a transform, such as a discrete cosine transform (DCT), a discrete sine transform (DST), or a conceptually similar transform, to the residual block or subdivisions thereof (e.g., four 8 x 8 transforms may be applied to a 16 x 16 array of residual values) to produce a set of residual transform coefficients. Transform coefficient generator 504 may be configured to perform any and all combinations of the transforms included in the family of discrete trigonometric transforms, including approximations thereof. Transform coefficient generator 504 may output transform coefficients to coefficient quantization unit 506. Coefficient quantization unit 506 may be configured to perform quantization of the transform coefficients. The quantization process may reduce the bit depth associated with some or all of the coefficients. The degree of quantization may alter the rate-distortion (i.e., bit-rate vs. quality of video) of encoded video data. The degree of quantization may be modified by adjusting a quantization parameter (QP). A quantization parameter may be determined based on slice level values and/or CU level values (e.g., CU delta QP values). QP data may include any data used to determine a QP for quantizing a particular set of transform coefficients. As illustrated in FIG. 5, quantized transform coefficients (which may be referred to as level values) are output to inverse quantization and transform coefficient processing unit 508. Inverse quantization and transform coefficient processing unit 508 may be configured to apply an inverse quantization and an inverse transformation to generate reconstructed residual data. As illustrated in FIG. 5, at summer 510, reconstructed residual data may be added to a predictive video block. In this manner, an encoded video block may be reconstructed and the resulting reconstructed video block may be used to evaluate the encoding quality for a given prediction, transformation, and/or quantization. Video encoder 500 may be configured to perform multiple coding passes (e.g., perform encoding while varying one or more of a prediction, transformation parameters, and quantization parameters). The rate-distortion of a bitstream or other system parameters may be optimized based on evaluation of reconstructed video blocks. Further, reconstructed video blocks may be stored and used as reference for predicting subsequent blocks.

[0059] Referring again to FIG. 5, intra prediction processing unit 512 may be configured to select an intra prediction mode for a video block to be coded. Intra prediction processing unit 512 may be configured to evaluate a frame and determine an intra prediction mode to use to encode a current block. As described above, possible intra

prediction modes may include planar prediction modes, DC prediction modes, and angular prediction modes. Further, it should be noted that in some examples, a prediction mode for a chroma component may be inferred from a prediction mode for a luma prediction mode. Intra prediction processing unit 512 may select an intra prediction mode after performing one or more coding passes. Further, in one example, intra prediction processing unit 512 may select a prediction mode based on a rate-distortion analysis. As illustrated in FIG. 5, intra prediction processing unit 512 outputs intra prediction data (e.g., syntax elements) to entropy encoding unit 518 and transform coefficient generator 504. As described above, a transform performed on residual data may be mode dependent (e.g., a secondary transform matrix may be determined based on a predication mode).

[0060] Referring again to FIG. 5, inter prediction processing unit 514 may be configured to perform inter prediction coding for a current video block. Inter prediction processing unit 514 may be configured to receive source video blocks and calculate a motion vector for PUs of a video block. A motion vector may indicate the displacement of a PU of a video block within a current video frame relative to a predictive block within a reference frame. Inter prediction coding may use one or more reference pictures. Further, motion prediction may be uni-predictive (use one motion vector) or bi-predictive (use two motion vectors). Inter prediction processing unit 514 may be configured to select a predictive block by calculating a pixel difference determined by, for example, sum of absolute difference (SAD), sum of square difference (SSD), or other difference metrics. As described above, a motion vector may be determined and specified according to motion vector prediction. Inter prediction processing unit 514 may be configured to perform motion vector prediction, as described above. Inter prediction processing unit 514 may be configured to generate a predictive block using the motion prediction data. For example, inter prediction processing unit 514 may locate a predictive video block within a frame buffer (not shown in FIG. 5). It should be noted that inter prediction processing unit 514 may further be configured to apply one or more interpolation filters to a reconstructed residual block to calculate sub-integer pixel values for use in motion estimation. Inter prediction processing unit 514 may output motion prediction data for a calculated motion vector to entropy encoding unit 518.

[0061] Referring again to FIG. 5, filter unit 516 receives reconstructed video blocks and coding parameters and outputs modified reconstructed video data. Filter unit 516 may be configured to perform deblocking and/or Sample Adaptive Offset (SAO) filtering. SAO filtering is a non-linear amplitude mapping that may be used to improve reconstruction by adding an offset to reconstructed video data. It should be noted that as illustrated in FIG. 5, intra prediction processing unit 512 and inter prediction processing

unit 514 may receive modified reconstructed video block via filter unit 216. Entropy encoding unit 518 receives quantized transform coefficients and predictive syntax data (i.e., intra prediction data and motion prediction data). It should be noted that in some examples, coefficient quantization unit 506 may perform a scan of a matrix including quantized transform coefficients before the coefficients are output to entropy encoding unit 518. In other examples, entropy encoding unit 518 may perform a scan. Entropy encoding unit 518 may be configured to perform entropy encoding according to one or more of the techniques described herein. In this manner, video encoder 500 represents an example of a device configured to generate encoded video data according to one or more techniques of this disclosure.

[0062] Referring again to FIG. 1, data encapsulator 107 may receive encoded video data and generate a compliant bitstream, e.g., a sequence of NAL units according to a defined data structure. A device receiving a compliant bitstream can reproduce video data therefrom. Further, as described above, sub-bitstream extraction may refer to a process where a device receiving a ITU-T H.265 compliant bitstream forms a new ITU-T H.265 compliant bitstream by discarding and/or modifying data in the received bitstream. It should be noted that the term conforming bitstream may be used in place of the term compliant bitstream. In one example, data encapsulator 107 may be configured to generate syntax according to one or more techniques described herein. It should be noted that data encapsulator 107 need not necessary be located in the same physical device as video encoder 106. For example, functions described as being performed by video encoder 106 and data encapsulator 107 may be distributed among devices illustrated in FIG. 4.

[0063]

As described above, the signaling of DPB information in JVET-N1001 is less than ideal. In one example, according to the techniques herein, **sps_max_dec_pic_buffering_minus1** and **sps_max_num_reorder_pics** may be coded as an unsigned integer using a fixed number of bits. That is, according to the techniques herein, the descriptor of ue(v) for syntax element **sps_max_dec_pic_buffering_minus1** and **sps_max_num_reorder_pics** in Table 3 may be changed to u(v). In this case, in one example, the semantics for **sps_max_dec_pic_buffering_minus1** and **sps_max_num_reorder_pics** may be based on the following:

sps_max_dec_pic_buffering_minus1[i] plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i. The length of the sps_max_dec_pic_buffering_minus1[i] syntax element is Ceil(Log2(MaxDpbSize − 1) bits.

When i is greater than 0, sps_max_dec_pic_buffering_minus1[i] shall be greater than or equal to sps_max_dec_pic_buffering_minus1[i − 1]. When sps_max_dec_pic_buffering_minus1[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1].

In one example, MaxDpbSize may be as defined below as provided in ITU-T H.265:

```
if( PicSizeInSamplesY <= ( MaxLumaPs >> 2 ) )
    MaxDpbSize = Min( 4 * maxDpbPicBuf, 16 )
else if( PicSizeInSamplesY <= ( MaxLumaPs >> 1 ) )
    MaxDpbSize = Min( 2 * maxDpbPicBuf, 16 )
else if( PicSizeInSamplesY <= ( ( 3 * MaxLumaPs ) >> 2 ) )
    MaxDpbSize = Min( ( 4 * maxDpbPicBuf ) / 3, 16 )
else
    MaxDpbSize = maxDpbPicBuf
```

where MaxLumaPs is specified in Table A and maxDpbPicBuf is equal to 6.

Level	Max luma picture size MaxLumaPs (samples)	Max CPB size MaxCPB (1000 bits)		Max slice segments per picture	Max # of tile rows	Max # of tile columns
		Main tier	High tier			
1	36 864	350	-	16	1	1
2	122 880	1 500	-	16	1	1
2.1	245 760	3 000	-	20	1	1
3	552 960	6 000	-	30	2	2
3.1	983 040	10 000	-	40	3	3
4	2 228 224	12 000	30 000	75	5	5
4.1	2 228 224	20 000	50 000	75	5	5
5	8 912 896	25 000	100 000	200	11	10
5.1	8 912 896	40 000	160 000	200	11	10
5.2	8 912 896	60 000	240 000	200	11	10
6	35 651 584	60 000	240 000	600	22	20
6.1	35 651 584	120 000	480 000	600	22	20
6.2	35 651 584	240 000	800 000	600	22	20

Table A

In another example, MaxDpbSize value may be different for different temporal sub-layer. Thus MaxDpbSize[i] may have different value for each i.

sps_max_num_reorder_pics[i] indicates the maximum allowed number of pictures that can precede any picture in the CVS in decoding order and follow that picture in output order when HighestTid is equal to i. The length of the sps_max_num_reorder_pics[i] syntax element is Ceil(Log2(MaxDpbSize − 1) bits.

When i is greater than 0, sps_max_num_reorder_pics[i] shall be greater than or equal to sps_max_num_reorder_pics[i − 1]. When sps_max_num_reorder_pics[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_num_reorder_pics[sps_max_sub_layers_minus1].

Further, in one example, the length of the sps_max_num_reorder_pics[i] syntax element may be Ceil(Log2(sps_max_dec_pic_buffering_minus1[i])) bits.

[0064] In one example, according to the techniques herein, the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i and/or the maximum allowed number of pictures that can precede any picture in the CVS in decoding order and follow that picture in output order when HighestTid is equal to i may be signaled using delta coding. Table 4 illustrates an example of syntax of a sequence parameter set, which may be signaled according to the techniques herein.

[0065]

seq_parameter_set_rbsp( ) {	Descriptor
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_reserved_zero_5bits	u(5)
profile_tier_level(sps_max_sub_layers_minus1)	
...	
log2_max_pic_order_cnt_lsb_minus4	ue(v)
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1); i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_delta[i]	ue(v)
sps_max_num_reorder_pics_delta[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
long_term_ref_pics_flag	u(1)
sps_idr_rpl_present_flag	u(1)
...	
}	

Table 4

With respect to Table 4, the semantics may be based on the semantics provided above with respect to Table 3, for syntax elements, **sps_max_dec_pic_buffering_delta**, **sps_max_num_reorder_pics_delta**, and **sps_max_latency_increase_plus1** in one example, the following semantics may be used:

sps_max_dec_pic_buffering_delta[i] is used to compute the value of SpsMaxDecPicBuffering[i] which specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i. The value of sps_max_dec_pic_buffering_delta [i] shall be in the range of 0 to MaxDpbSize − 1, inclusive, where MaxDpbSize is as specified somewhere else. In another example, the value of SpsMaxDecPicBuffering[i] shall be in the range of 0 to MaxDpbSize , inclusive, where MaxDpbSize is as specified somewhere else.

When sps_max_dec_pic_buffering_delta [i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to 0.

The variable SpsMaxDecPicBuffering[i] is calculated as follows:

```
if (i==0)
        SpsMaxDecPicBuffering[ i ] = sps_max_dec_pic_buffering_delta[ i ] + 1
else
      SpsMaxDecPicBuffering[ i ] = sps_max_dec_pic_buffering_delta[ i ] +
      SpsMaxDecPicBuffering[ i-1 ]
```

In another example, the variable SpsMaxDecPicBuffering[i] may be calculated as follows:

SpsMaxDecPicBuffering[i] = sps_max_dec_pic_buffering_delta[i] + (i==0)? 1: SpsMaxDecPicBuffering[i-1]

sps_max_num_reorder_pics_delta[i] is used to compute the value of SpsMaxNumReorderPics[i] which specifies the maximum allowed number of pictures that can precede any picture in the CVS in decoding order and follow that picture in output order when HighestTid is equal to i. The value of sps_max_num_reorder_pics_delta[i] shall be in the range of 0 to (SpsMaxDecPicBuffering[i]-1), inclusive. In another example the value of SpsMaxNumReorderPics [I] shall be in the range of 0 to (SpsMaxDecPicBuffering[i]-1). When sps_max_num_reorder_pics_delta[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to 0.

The variable SpsMaxNumReorderPics [i] is calculated as follows:

if (i==0)

SpsMaxNumReorderPics[i] = sps_max_num_reorder_pics_delta[i] + 1

else

SpsMaxNumReorderPics [i] = sps_max_num_reorder_pics_delta [i] + SpsMaxNumReorderPics [i-1]

In another example, the variable SpsMaxNumReorderPics [i] may be calculated as follows:

SpsMaxNumReorderPics [i] = sps_max_num_reorder_pics_delta [i] + (i==0)? 0: SpsMaxNumReorderPics [i-1].

sps_max_latency_increase_plus1[i] not equal to 0 is used to compute the value of SpsMaxLatencyPictures[i], which specifies the maximum number of pictures that can precede any picture in the CVS in output order and follow that picture in decoding order when HighestTid is equal to i.

When sps_max_latency_increase_plus1[i] is not equal to 0, the value of SpsMaxLatencyPictures[i] is specified as follows:

SpsMaxLatencyPictures[i] =
SpsMaxNumReorderPics[i] + sps_max_latency_increase_plus1[i] − 1

When sps_max_latency_increase_plus1[i] is equal to 0, no corresponding limit is expressed.

The value of sps_max_latency_increase_plus1[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. When sps_max_latency_increase_plus1[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_latency_increase_plus1[sps_max_sub_layers_minus1].

It should be noted that based on the example semantics for **sps_max_dec_pic_buffering_delta**, **sps_max_num_reorder_pics_delta**, and **sps_max_latency_increase_plus1**, according to the techniques herein, in JVET-N1001, the following modifications may be made:

- Replace all occurrence of sps_max_dec_pic_buffering_minus1[i] with SpsMaxDecPicBuffering[i]-1
- Replace all occurrence of (sps_max_dec_pic_buffering_minus1[i]+1) with SpsMaxDecPicBuffering[i]
- Replace all occurrence of sps_max_num_reorder_pics[i] with SpsMaxNumReorderPics [i]

In one example, according to the techniques herein, syntax element **sps_sub_layer_ordering_info_present_flag** may be conditionally signaled based on the value of syntax element **sps_max_sub_layers_minus1**. In particular, **sps_sub_layer_ordering_info_present_flag** syntax element will not be signalled when there is only one temporal sub-layer in the CVS or bitstream which refers to the SPS. Instead the value of **sps_sub_layer_ordering_info_present_flag** is inferred in this case. This saves one bit. Table 5 illustrates an example of syntax of a sequence parameter set, which may be signaled according to the techniques herein.

seq_parameter_set_rbsp( ) {	**Descriptor**
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_reserved_zero_5bits	u(5)
profile_tier_level(sps_max_sub_layers_minus1)	
…	
log2_max_pic_order_cnt_lsb_minus4	ue(v)
if(!sps_max_sub_layers_minus1)	
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1); i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_delta[i]	ue(v)
sps_max_num_reorder_pics_delta[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
long_term_ref_pics_flag	u(1)
sps_idr_rpl_present_flag	u(1)
…	
}	

Table 5

In another example, the condition if(!sps_max_sub_layers_minus1) may instead be written as if(sps_max_sub_layers_minus1>0) or some other similar test.

With respect to Table 5, the semantics may be based on the semantics provided above with respect to Table 3, for syntax elements, **sps_sub_layer_ordering_info_present_flag**, **sps_max_dec_pic_buffering_delta**, **sps_max_num_reorder_pics_delta**, and **sps_max_latency_increase_plus1** in one example, the following semantics may be used:

sps_sub_layer_ordering_info_present_flag equal to 1 specifies that sps_max_dec_pic_buffering_minus1[i], sps_max_num_reorder_pics[i], and sps_max_latency_increase_plus1[i] are present for sps_max_sub_layers_minus1 + 1 sub-layers. sps_sub_layer_ordering_info_present_flag equal to 0 specifies that the values of sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1], sps_max_num_reorder_pics[sps_max_sub_layers_minus1], and sps_max_latency_increase_plus1[sps_max_sub_layers_minus1] apply to all sub-layers. When not present sps_sub_layer_ordering_info_present_flag is inferred to be equal to 0.

In another example: When not present sps_sub_layer_ordering_info_present_flag is inferred to be equal to 1.

sps_max_dec_pic_buffering_delta[i] is used to compute the value of SpsMaxDecPicBuffering[i] which specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i. The value of SpsMaxDecPicBuffering[i] shall be in the range of 0 to MaxDpbSize, inclusive, where MaxDpbSize is as specified somewhere else. When sps_max_dec_pic_buffering_delta [i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to 0. The variable SpsMaxDecPicBuffering[i] is calculated as follows:

SpsMaxDecPicBuffering[i] = sps_max_dec_pic_buffering_delta[i] + (i==0)? 1: SpsMaxDecPicBuffering[i−1]

sps_max_num_reorder_pics_delta[i] is used to compute the value of SpsMaxNumReorderPics[i] which specifies the maximum allowed number of pictures that can precede any picture in the CVS in decoding order and follow that picture in output order when HighestTid is equal to i. The value of SpsMaxNumReorderPics[i] shall be in the range of 0 to (SpsMaxDecPicBuffering[i]-1), inclusive. When sps_max_num_reorder_pics_delta[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to 0. The variable SpsMaxNumReorderPics [i] is calculated as follows:

SpsMaxNumReorderPics [i] = sps_max_num_reorder_pics_delta [i] + (i==0)? 0: SpsMaxNumReorderPics [i-1].

sps_max_latency_increase_plus1[i] not equal to 0 is used to compute the value of SpsMaxLatencyPictures[i], which specifies the maximum number of pictures that can precede any picture in the CVS in output order and follow that picture in decoding order when HighestTid is equal to i.

When sps_max_latency_increase_plus1[i] is not equal to 0, the value of SpsMaxLatencyPictures[i] is specified as follows:

SpsMaxLatencyPictures[i] =
SpsMaxNumReorderPics[i] + sps_max_latency_increase_plus1[i] − 1

When sps_max_latency_increase_plus1[i] is equal to 0, no corresponding limit is expressed.

The value of sps_max_latency_increase_plus1[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. When sps_max_latency_increase_plus1[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_latency_increase_plus1[sps_max_sub_layers_minus1].

In this manner, source device 102 represents an example of a device configured to signal a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video and signal a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, wherein the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video is calculated as the sum of the first value and the second value.

[0066] Referring again to FIG. 1, interface 108 may include any device configured to receive data generated by data encapsulator 107 and transmit and/or store the data to a communications medium. Interface 108 may include a network interface card, such as an Ethernet card, and may include an optical transceiver, a radio frequency transceiver, or any other type of device that can send and/or receive information. Further, interface

108 may include a computer system interface that may enable a file to be stored on a storage device. For example, interface 108 may include a chipset supporting Peripheral Component Interconnect (PCI) and Peripheral Component Interconnect Express (PCIe) bus protocols, proprietary bus protocols, Universal Serial Bus (USB) protocols, I^2C, or any other logical and physical structure that may be used to interconnect peer devices.

[0067] Referring again to FIG. 1, destination device 120 includes interface 122, data decapsulator 123, video decoder 124, and display 126. Interface 122 may include any device configured to receive data from a communications medium. Interface 122 may include a network interface card, such as an Ethernet card, and may include an optical transceiver, a radio frequency transceiver, or any other type of device that can receive and/or send information. Further, interface 122 may include a computer system interface enabling a compliant video bitstream to be retrieved from a storage device. For example, interface 122 may include a chipset supporting PCI and PCIe bus protocols, proprietary bus protocols, USB protocols, I^2C, or any other logical and physical structure that may be used to interconnect peer devices. Data decapsulator 123 may be configured to receive and parse any of the example syntax structures described herein.

[0068] Video decoder 124 may include any device configured to receive a bitstream (e.g., a sub-bitstream extraction) and/or acceptable variations thereof and reproduce video data therefrom. Display 126 may include any device configured to display video data. Display 126 may comprise one of a variety of display devices such as a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display. Display 126 may include a High Definition display or an Ultra High Definition display. It should be noted that although in the example illustrated in FIG. 1, video decoder 124 is described as outputting data to display 126, video decoder 124 may be configured to output video data to various types of devices and/or sub-components thereof. For example, video decoder 124 may be configured to output video data to any communication medium, as described herein.

[0069] FIG. 6 is a block diagram illustrating an example of a video decoder that may be configured to decode video data according to one or more techniques of this disclosure (e.g., the decoding process for reference-picture list construction described above). In one example, video decoder 600 may be configured to decode transform data and reconstruct residual data from transform coefficients based on decoded transform data. Video decoder 600 may be configured to perform intra prediction decoding and inter prediction decoding and, as such, may be referred to as a hybrid decoder. Video decoder 600 may be configured to parse any combination of the syntax elements described above in Tables 1-4. Further, video decoder 600 may be configured to allocate resources to a DPB based on parsed values of the syntax elements described

above in Tables 1-4. Video decoder 600 may decode a picture based on or according to the processes described above.

[0070] In the example illustrated in FIG. 6, video decoder 600 includes an entropy decoding unit 602, inverse quantization unit and transform coefficient processing unit 604, intra prediction processing unit 606, inter prediction processing unit 608, summer 610, post filter unit 612, and reference buffer 614. Video decoder 600 may be configured to decode video data in a manner consistent with a video coding system. It should be noted that although example video decoder 600 is illustrated as having distinct functional blocks, such an illustration is for descriptive purposes and does not limit video decoder 600 and/or sub-components thereof to a particular hardware or software architecture. Functions of video decoder 600 may be realized using any combination of hardware, firmware, and/or software implementations.

[0071] As illustrated in FIG. 6, entropy decoding unit 602 receives an entropy encoded bitstream. Entropy decoding unit 602 may be configured to decode syntax elements and quantized coefficients from the bitstream according to a process reciprocal to an entropy encoding process. Entropy decoding unit 602 may be configured to perform entropy decoding according any of the entropy coding techniques described above. Entropy decoding unit 602 may determine values for syntax elements in an encoded bitstream in a manner consistent with a video coding standard. As illustrated in FIG. 6, entropy decoding unit 602 may determine a quantization parameter, quantized coefficient values, transform data, and predication data from a bitstream. In the example, illustrated in FIG. 6, inverse quantization unit and transform coefficient processing unit 604 receives a quantization parameter, quantized coefficient values, transform data, and predication data from entropy decoding unit 602 and outputs reconstructed residual data.

[0072] Referring again to FIG. 6, reconstructed residual data may be provided to summer 610 Summer 610 may add reconstructed residual data to a predictive video block and generate reconstructed video data. A predictive video block may be determined according to a predictive video technique (i.e., intra prediction and inter frame prediction). Intra prediction processing unit 606 may be configured to receive intra prediction syntax elements and retrieve a predictive video block from reference buffer 614. Reference buffer 614 may include a memory device configured to store one or more frames of video data. Intra prediction syntax elements may identify an intra prediction mode, such as the intra prediction modes described above. Inter prediction processing unit 608 may receive inter prediction syntax elements and generate motion vectors to identify a prediction block in one or more reference frames stored in reference buffer 616. Inter prediction processing unit 608 may produce motion compensated blocks, possibly performing interpolation based on interpolation filters.

Identifiers for interpolation filters to be used for motion estimation with sub-pixel precision may be included in the syntax elements. Inter prediction processing unit 608 may use interpolation filters to calculate interpolated values for sub-integer pixels of a reference block. Post filter unit 614 may be configured to perform filtering on reconstructed video data. For example, post filter unit 614 may be configured to perform deblocking and/or Sample Adaptive Offset (SAO) filtering, e.g., based on parameters specified in a bitstream. Further, it should be noted that in some examples, post filter unit 614 may be configured to perform proprietary discretionary filtering (e.g., visual enhancements, such as, mosquito noise reduction). As illustrated in FIG. 6, a reconstructed video block may be output by video decoder 600. In this manner, video decoder 600 represents an example of a device configured to parse a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video, parse a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, and calculate the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video as the sum of the first value and the second value.

[0073] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2)a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

[0074] By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared,

radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0075] Instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly, the term "processor," as used herein may refer to any of the foregoing structure or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[0076] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

[0077] Moreover, each functional block or various features of the base station device and the terminal device used in each of the aforementioned embodiments may be implemented or executed by a circuitry, which is typically an integrated circuit or a plurality of integrated circuits. The circuitry designed to execute the functions described in the present specification may comprise a general-purpose processor, a digital signal processor (DSP), an application specific or general application integrated circuit (ASIC), a field programmable gate array (FPGA), or other programmable logic devices, discrete gates or transistor logic, or a discrete hardware component, or a combination thereof. The general-purpose processor may be a microprocessor, or alternatively, the processor may be a conventional processor, a controller, a microcontroller

or a state machine. The general-purpose processor or each circuit described above may be configured by a digital circuit or may be configured by an analogue circuit. Further, when a technology of making into an integrated circuit superseding integrated circuits at the present time appears due to advancement of a semiconductor technology, the integrated circuit by this technology is also able to be used.

[0078] Various examples have been described. These and other examples are within the scope of the following claims.

[0079] <Summary>

In one example, a method of signaling decoded picture buffer (DPB) information for decoding video data, the method comprising: signaling a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video; and signaling a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video, wherein the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video is calculated as the sum of the first value and the second value.

[0080] In one example, a method of decoding video data, the method comprising: parsing a first value indicating a maximum required size of a decoded picture buffer for a corresponding first sub-layer of video; parsing a second value indicating a maximum required size of a decoded picture buffer for a corresponding second sub-layer of video; and calculating the maximum required size of a decoded picture buffer for the corresponding second sub-layer of video as the sum of the first value and the second value.

[0081] In one example, the method, wherein the first value and the second value are included in a sequence parameter set.

[0082] In one example, a device for coding video data, the device comprising one or more processors configured to perform any and all combinations of the steps.

[0083] In one example, the device, wherein the device includes a video encoder.

[0084] In one example, the device, wherein the device includes a video decoder.

[0085] In one example, a system comprising: the device includes a video encoder; and the device includes a video decoder.

[0086] In one example, an apparatus for coding video data, the apparatus comprising means for performing any and all combinations of the steps.

[0087] In one example, a non-transitory computer-readable storage medium comprising instructions stored thereon that, when executed, cause one or more processors of a device for coding video data to perform any and all combinations of the steps.

[0088] In one example, a method of decoding video data, the method comprising: parsing a syntax element specifying a maximum number of temporal sublayers that may be present in a coded video sequence; determining whether a value of the syntax element

specifying the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero; and conditionally parsing a syntax element indicating the presence of decoded picture buffer parameters only when it is determined that the value of the syntax element specifying the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero.

[0089] In one example, the method, further comprising inferring the value of the syntax element indicating the presence of decoded picture buffer parameters to be zero when it is not parsed.

[0090] In one example, the method, wherein the syntax element is included in a sequence parameter set.

[0091] In one example, a device comprising one or more processors configured to: parse a syntax element specifying a maximum number of temporal sublayers that may be present in a coded video sequence; determine whether a value of the syntax element specifying the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero; and conditionally parse a syntax element indicating the presence of decoded picture buffer parameters only when it is determined that the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero.

[0092] In one example, the device, wherein the one or more processors are further configured to infer the value of the syntax element indicating the presence of decoded picture buffer parameters to be zero when it is not parsed.

[0093] In one example, the device, wherein the syntax element is included in a sequence parameter set.

[0094] In one example, the device, wherein the device is a video decoder.

[0095] <Cross Reference>

This Nonprovisional application claims priority under 35 U.S.C. § 119 on provisional Application No. 62/863,776 on June 19, 2019, No. 62/868,641 on June 28, 2019, the entire contents of which are hereby incorporated by reference.

Claims

[Claim 1] A method of decoding video data, the method comprising:
parsing a syntax element specifying a maximum number of temporal sublayers that may be present in a coded video sequence;
determining whether a value of the syntax element specifying the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero; and
conditionally parsing a syntax element indicating the presence of decoded picture buffer parameters only when it is determined that the value of the syntax element specifying the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero.

[Claim 2] The method of claim 1, further comprising inferring the value of the syntax element indicating the presence of decoded picture buffer parameters to be zero when it is not parsed.

[Claim 3] The method of claim 1, wherein the syntax element is included in a sequence parameter set.

[Claim 4] A device comprising one or more processors configured to:
parse a syntax element specifying a maximum number of temporal sublayers that may be present in a coded video sequence;
determine whether a value of the syntax element specifying the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero; and conditionally parse a syntax element indicating the presence of decoded picture buffer parameters only when it is determined that the maximum number of temporal sublayers that may be present in a coded video sequence is greater than zero.

[Claim 5] The device of claim 4, wherein the one or more processors are further configured to infer the value of the syntax element indicating the presence of decoded picture buffer parameters to be zero when it is not parsed.

[Claim 6] The device of claim 4, wherein the syntax element is included in a sequence parameter set.

[Claim 7] The device of claim 4, wherein the device is a video decoder.

[Fig. 1]

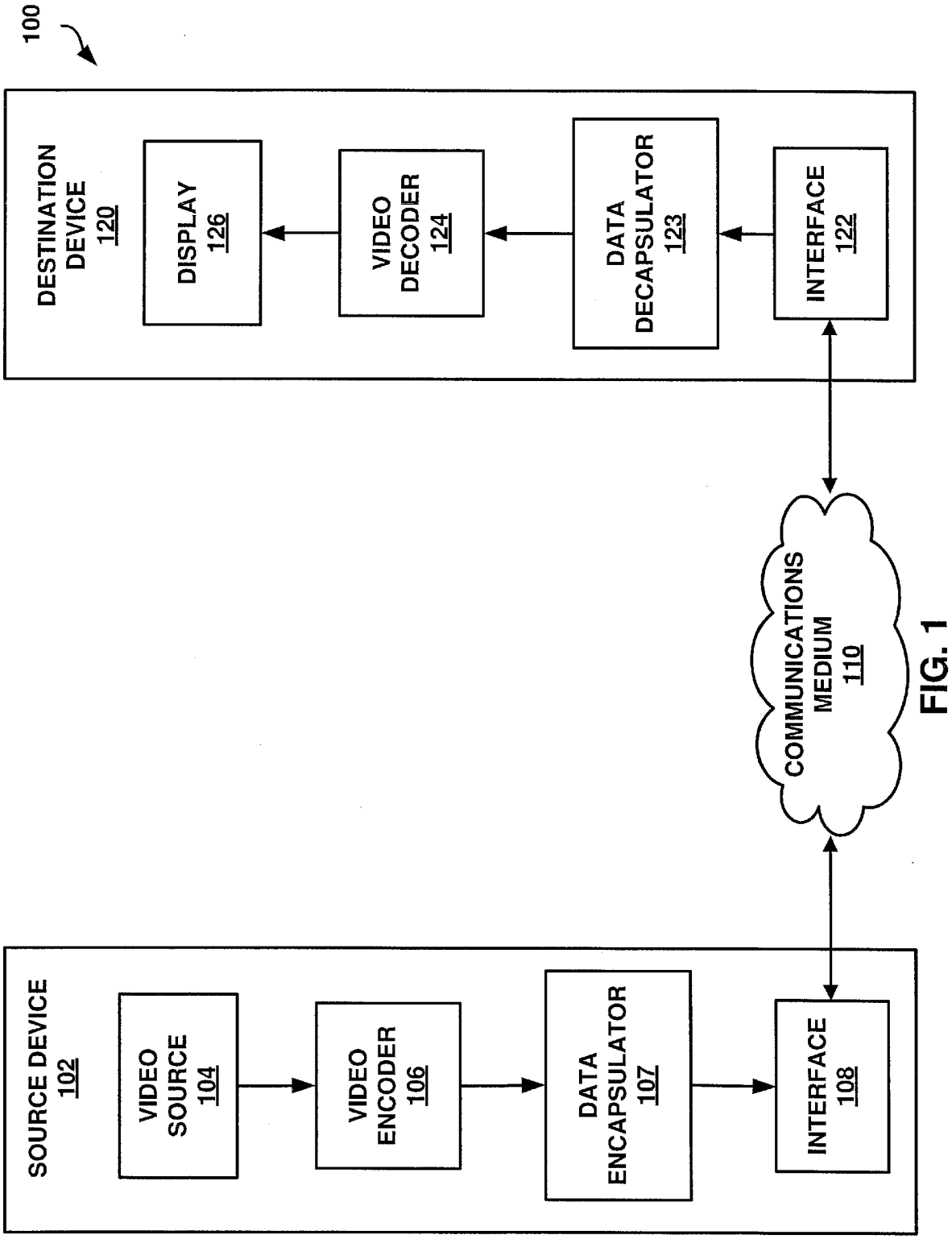
100
DESTINATION DEVICE 120
DISPLAY 126
VIDEO DECODER 124
DATA DECAPSULATOR 123
INTERFACE 122
COMMUNICATIONS MEDIUM 110
SOURCE DEVICE 102
VIDEO SOURCE 104
VIDEO ENCODER 106
DATA ENCAPSULATOR 107
INTERFACE 108

FIG. 1

[Fig. 2]

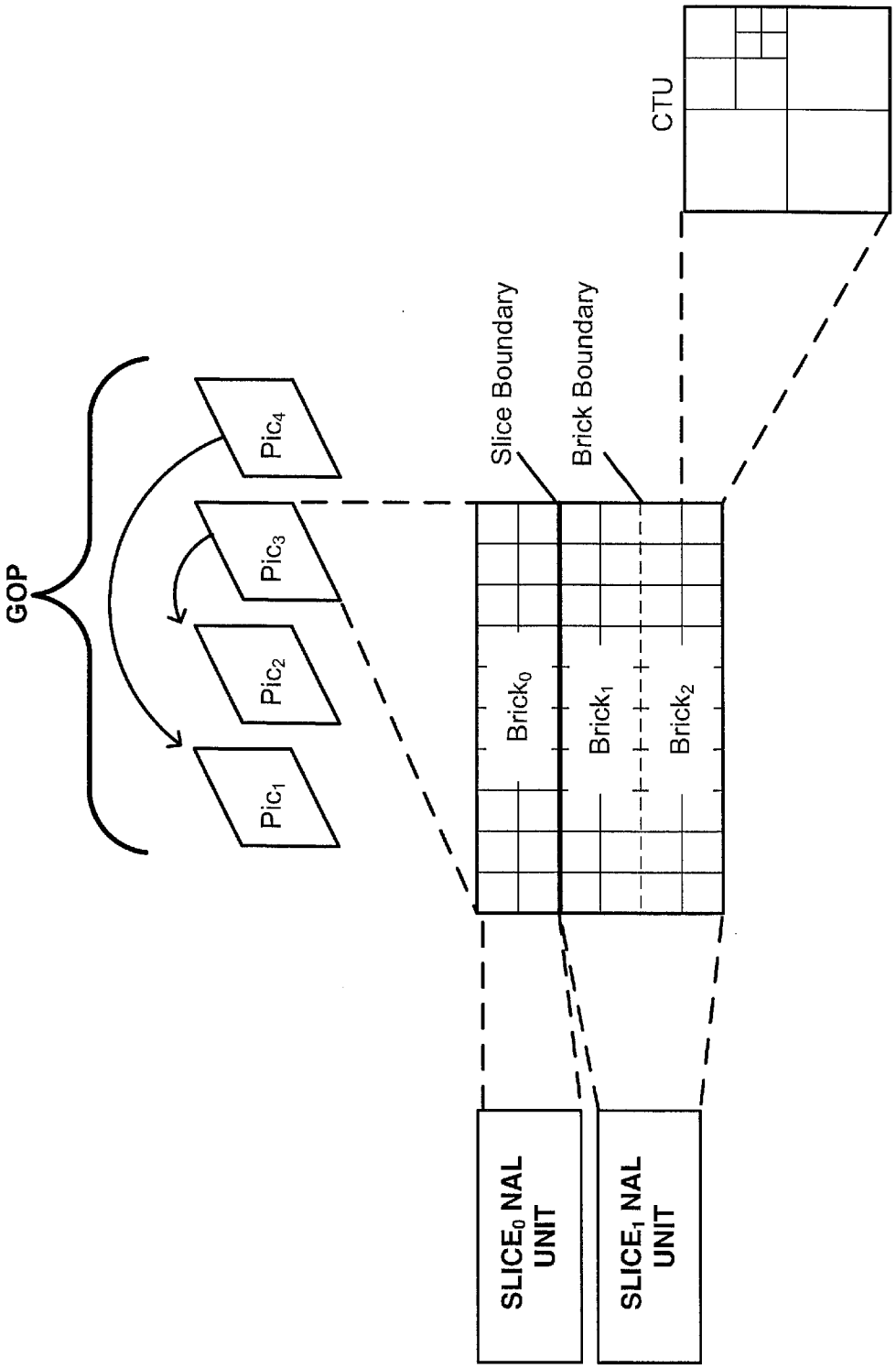
GOP
Pic4
Pic3
Pic2
Pic1
Slice Boundary
Brick Boundary
CTU
Brick0
Brick1
Brick2
SLICE0 NAL UNIT
SLICE1 NAL UNIT

FIG. 2

[Fig. 3]

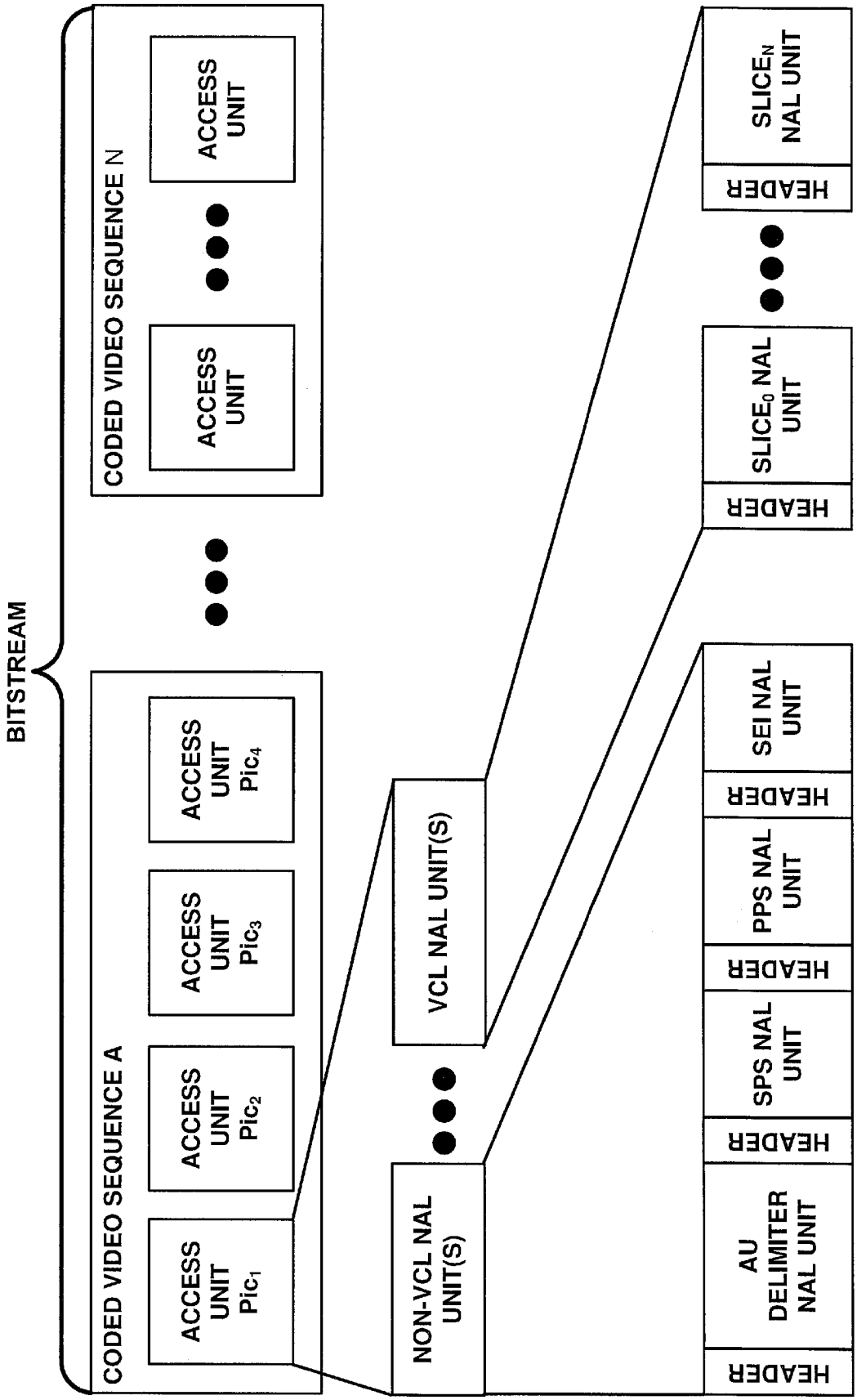
BITSTREAM
CODED VIDEO SEQUENCE A
ACCESS UNIT Pic1
ACCESS UNIT Pic2
ACCESS UNIT Pic3
ACCESS UNIT Pic4
CODED VIDEO SEQUENCE N
ACCESS UNIT
ACCESS UNIT
NON-VCL NAL UNIT(S)
VCL NAL UNIT(S)
HEADER
AU DELIMITER NAL UNIT
HEADER
SPS NAL UNIT
HEADER
PPS NAL UNIT
HEADER
SEI NAL UNIT
HEADER
SLICE0 NAL UNIT
HEADER
SLICEN NAL UNIT

FIG. 3

[Fig. 4]

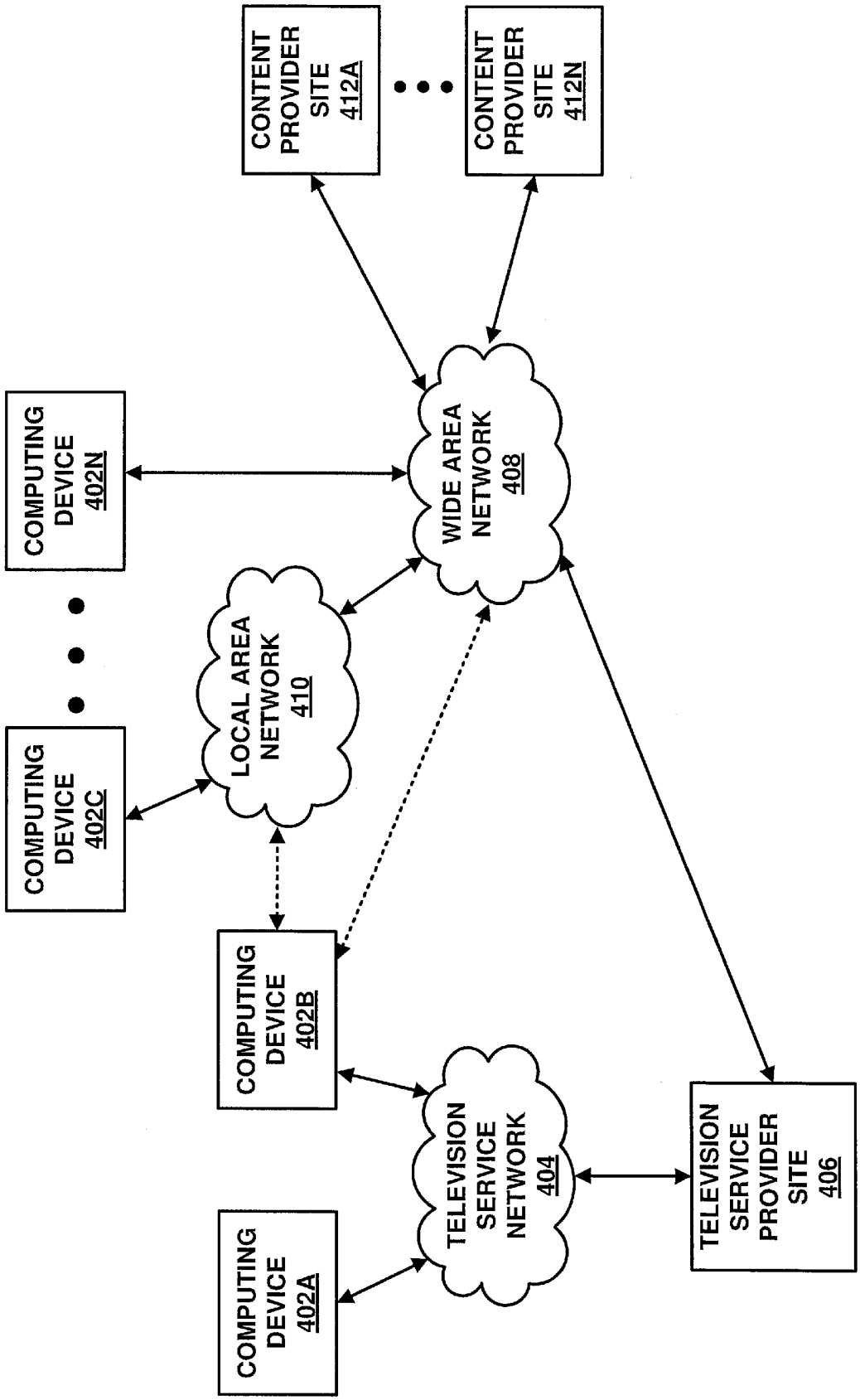
CONTENT PROVIDER SITE 412A
CONTENT PROVIDER SITE 412N
WIDE AREA NETWORK 408
COMPUTING DEVICE 402N
LOCAL AREA NETWORK 410
COMPUTING DEVICE 402C
COMPUTING DEVICE 402B
TELEVISION SERVICE NETWORK 404
TELEVISION SERVICE PROVIDER SITE 406
COMPUTING DEVICE 402A

FIG. 4

[Fig. 5]

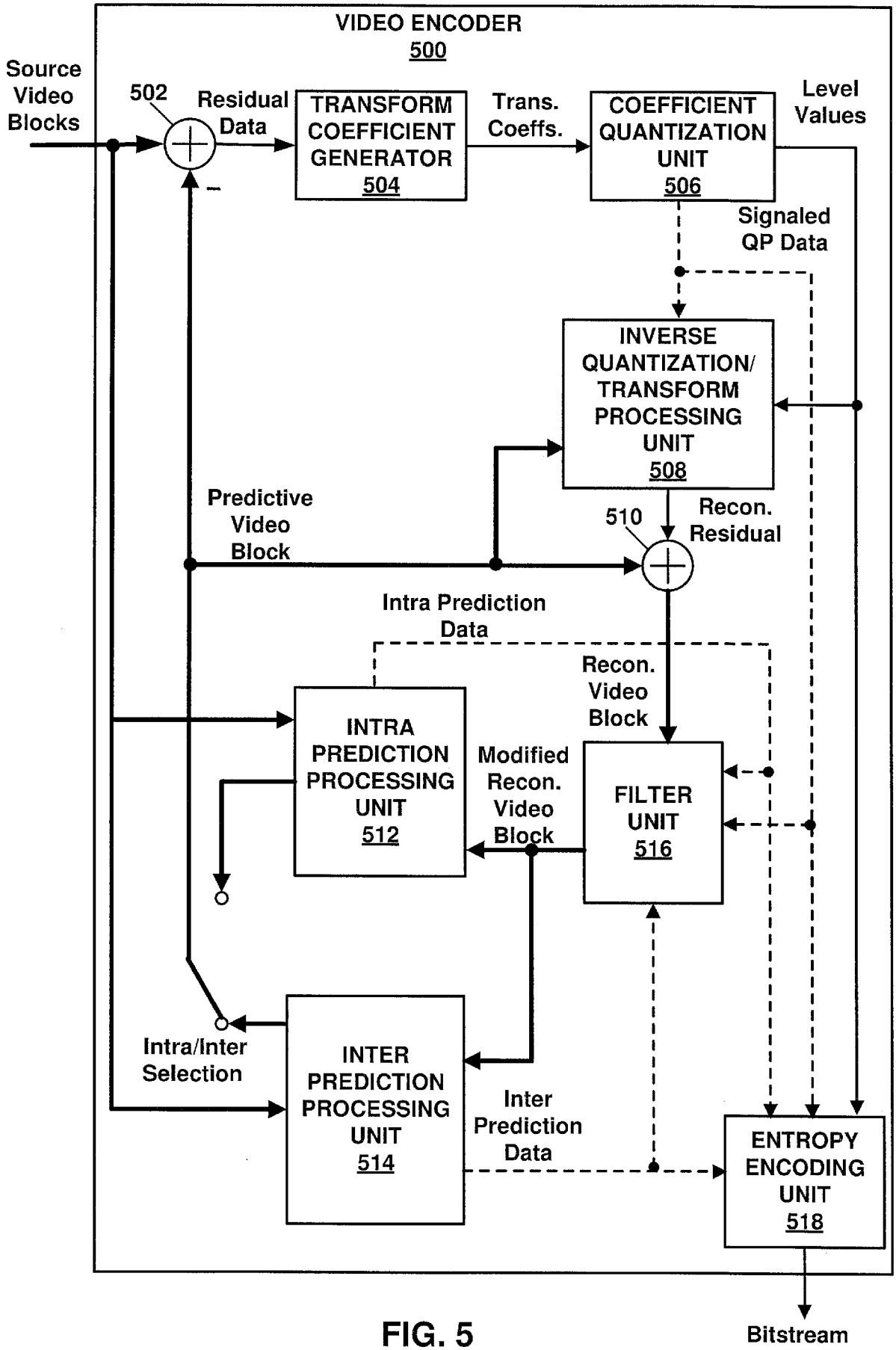
VIDEO ENCODER
500
Source Video Blocks
502
Residual Data
TRANSFORM COEFFICIENT GENERATOR
504
Trans. Coeffs.
COEFFICIENT QUANTIZATION UNIT
506
Level Values
Signaled QP Data
INVERSE QUANTIZATION/ TRANSFORM PROCESSING UNIT
508
Predictive Video Block
510
Recon. Residual
Intra Prediction Data
Recon. Video Block
INTRA PREDICTION PROCESSING UNIT
512
Modified Recon. Video Block
FILTER UNIT
516
Intra/Inter Selection
INTER PREDICTION PROCESSING UNIT
514
Inter Prediction Data
ENTROPY ENCODING UNIT
518
Bitstream

FIG. 5

[Fig. 6]

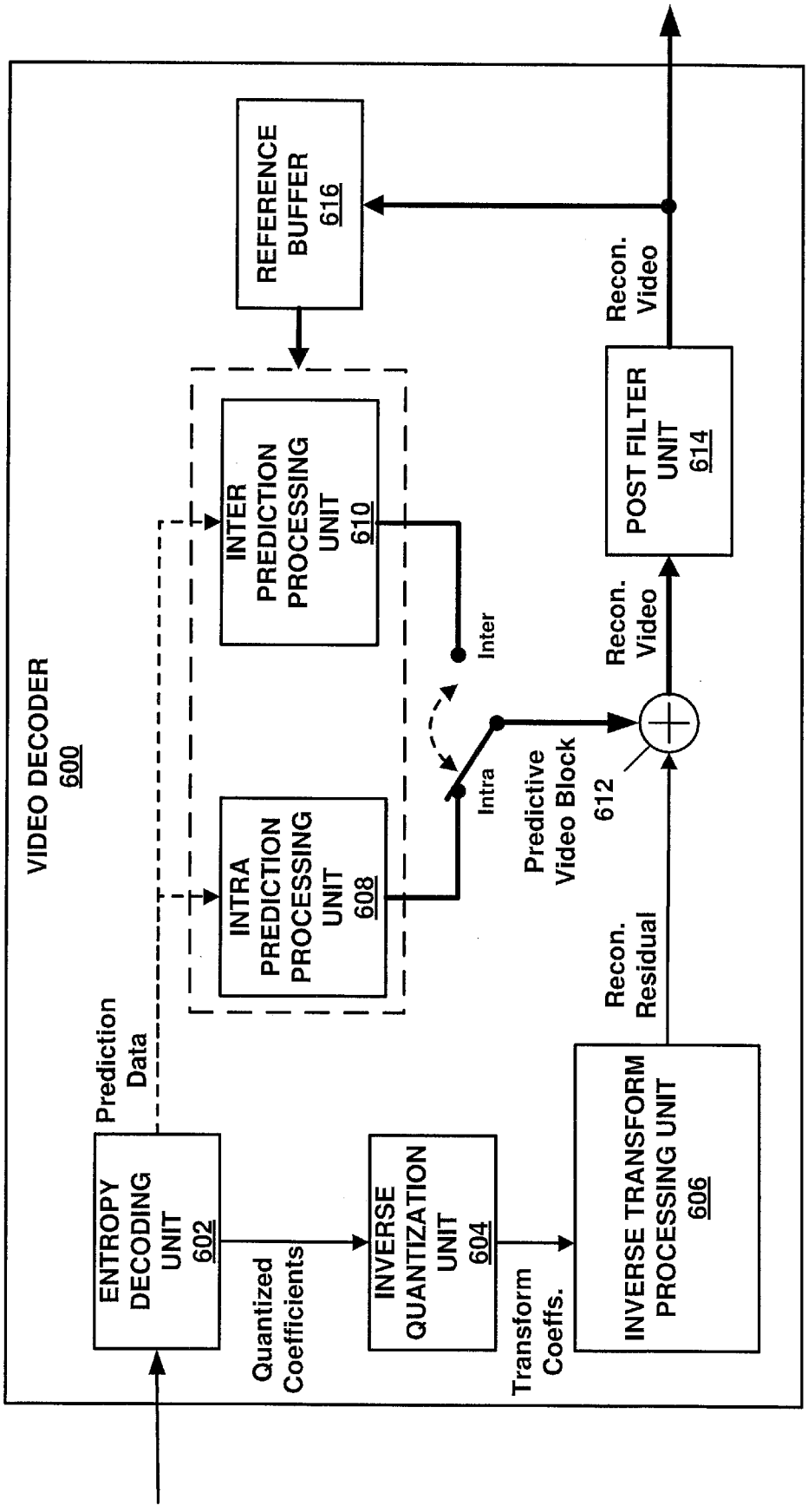
VIDEO DECODER
600
ENTROPY DECODING UNIT
602
INVERSE QUANTIZATION UNIT
604
INVERSE TRANSFORM PROCESSING UNIT
606
INTRA PREDICTION PROCESSING UNIT
608
INTER PREDICTION PROCESSING UNIT
610
REFERENCE BUFFER
616
POST FILTER UNIT
614
Prediction Data
Quantized Coefficients
Transform Coeffs.
Recon. Residual
Predictive Video Block
612
Intra
Inter
Recon. Video
Recon. Video

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2020/024230

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/70(2014.01)i
FI: H04N19/70

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)

H04N19/00-19/98

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

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2020
Registered utility model specifications of Japan 1996-2020
Published registered utility model applications of Japan 1994-2020

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	DESHPANDE, Sachin, On Sequence Parameter Set, Joint Collaborative Team on Video Coding (JCT-VC), 2014.01.04, [JCTVC-P0155] Section 2	1-7
A	CHOI, Byeongdoo et al., MV-HEVC/SHVC HLS: On temporal sub-layer management, Joint Collaborative Team on Video Coding (JCT-VC), 2013.07.22, [JCTVC-N0130r1] (version 3) Section 3	1-7

☐ 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
"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
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
03 August 2020	**11 August 2020**
Name and mailing address of the ISA/JP **Japan Patent Office** **3-4-3, Kasumigaseki, Chiyoda-ku, Tokyo** **100-8915, Japan**	Authorized officer **BANDO, Daigoro 5C 3241** Telephone No. **+81-3-3581-1101 Ext. 3541**