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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
09 January 2020 (09.01.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/009124 A1

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

(21) International Application Number:
PCT/JP2019/026364

(22) International Filing Date:
02 July 2019 (02.07.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/693,898 03 July 2018 (03.07.2018) US
62/697,257 12 July 2018 (12.07.2018) US

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

(72) Inventors: **DESHPANDE, Sachin G.**. **BOSSEN, Frank**. **SEGALL, Christopher Andrew**.

(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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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))*

(54) Title: SYSTEMS AND METHODS FOR HIGH-LEVEL SYNTAX SIGNALING IN VIDEO CODING

(57) Abstract: This disclosure relates to video coding and more particularly to techniques for signaling of high-level syntax for coded video. According to an aspect of an invention, a value for a syntax element in a network abstraction layer unit indicating a number of syntax structures is signaled, and for each the number of syntax structures, a value of a syntax element indicating a syntax structure type is signaled.

[Fig. 5]

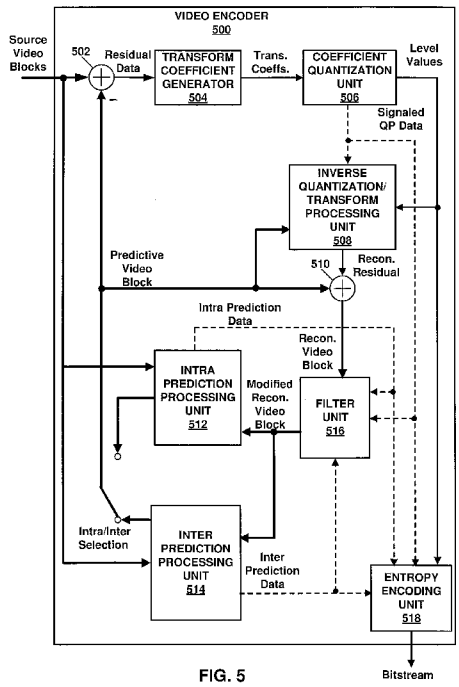

FIG. 5

WO 2020/009124 A1

[Continued on next page]

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

Description

Title of Invention: SYSTEMS AND METHODS FOR HIGH-LEVEL SYNTAX SIGNALING IN VIDEO CODING

Technical Field

[0001] This disclosure relates to video coding and more particularly to techniques for signaling of signal high-level syntax 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 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 studying the potential need for standardization of future 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 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 were proposed by various groups at the 10th Meeting of ISO/IEC JTC1/SC29/WG11 16-20 April 2018, San Diego, CA. As a result of the multiple descriptions of video coding, a 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.

[0003] Video compression techniques reduce data requirements for storing and transmitting video data 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 frames within a video sequence, a frame within a group of frames, slices within a frame, coding tree units (e.g., macroblocks) within a slice, coding blocks within a coding tree unit, etc.). Intra prediction coding techniques (e.g., intra-picture (spatial)) and inter prediction techniques (i.e., inter-picture (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, motion vectors, and block vectors). Residual data and syntax elements may be entropy coded. Entropy encoded residual data and syntax elements may be included in a compliant bitstream. Compliant bitstreams and associated metadata may be formatted according to data structures.

Summary of Invention

[0004] In one example, a method of signaling picture type information comprises signaling a value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit, and for each the number of syntax structures, signaling a value of a syntax element indicating a syntax structure type.

[0005] In one example, a method of decoding video data comprises parsing a value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit, for each the number of syntax structures parsing a value for syntax element indicating a syntax structure type, and generating video data based on values of the parsed syntax elements.

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 high-level syntax for coded video. Signaling of high-level syntax according to the techniques described herein may be particularly useful for improving video distribution system performance by lowering transmission bandwidth and/or facilitating parallelization of a video encoder and/or decoder. It should be noted that although techniques of this disclosure are described with respect to ITU-T H.264, ITU-T H.265, and JVET-J1001 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 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. Thus, reference to ITU-T H.264, ITU-T H.265, and JVET-J1001 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 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 device comprises one or more processors configured to signal a value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit, and for each the number of syntax structures, signaling a value for a syntax element indicating a syntax structure type.

[0009] 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 value for a syntax element in a network abstraction layer unit indicating a

number of syntax structures included in the network abstraction layer unit, and for each the number of syntax structures, signaling a value for a syntax element indicating a syntax structure type.

[0010] In one example, an apparatus comprises means for signaling a value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit, and means for signaling a value of a syntax element indicating a syntax structure type for each the number of syntax structures.

[0011] In one example, a device comprises one or more processors configured to parse a value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit, and for each the number of syntax structures parse a value for syntax element indicating a syntax structure type, and generate video data based on values of the parsed syntax elements.

[0012] 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 value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit, and for each the number of syntax structures parse a value for syntax element indicating a syntax structure type, and generate video data based on values of the parsed syntax elements.

[0013] In one example, an apparatus comprises means for parsing a value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit, means for parsing a value for syntax element indicating a syntax structure type for each the number of syntax structures, and means for generating video data based on values of the parsed syntax elements.

[0014] 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.

[0015] Video content typically includes video sequences comprised of a series of frames. A series of frames may also be referred to as a group of pictures (GOP). Each video frame or picture may include a one or more slices, where a slice includes a plurality of video blocks. A video block includes an array of pixel values (also referred to as samples) that may be predictively coded. Video blocks may be ordered according to a scan pattern (e.g., a raster scan). A video encoder performs predictive encoding on video blocks and sub-divisions thereof. ITU-T H.264 specifies a macroblock including 16 x 16 luma samples. ITU-T H.265 specifies an analogous Coding Tree Unit (CTU) structure (which may be referred to as a Largest Coding Unit (LCU)) where a picture may be split into CTUs of equal size and each CTU may include Coding Tree Blocks (CTB) having 16 x 16, 32 x 32, or 64 x 64 luma samples. 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 pixel values that may be predictively coded, sub-divisions thereof, and/or corresponding structures. Further, according to ITU-T H.265, each video frame or picture may be partitioned to include one or more tiles, where a tile is a sequence of coding tree units corresponding to a rectangular area of a picture.

[0016] In ITU-T H.265, a CTU is composed of respective CTBs for each component of video data (e.g., luma (Y) and chroma (Cb and Cr)). 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.

[0017] 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 respect 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 (i.e., intra prediction PB sizes type include MxM or M/2xM/2, where M is the height and width of the square CB). 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 (i.e., inter prediction PB types include MxM, M/2xM/2, M/2xM, or MxM/2). 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 (i.e., asymmetric partitions include M/4xM left, M/4xM right, MxM/4 top, and MxM/4 bottom). 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.

[0018] 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. Thus, the binary tree structure in JEM enables square and rectangular leaf nodes, where each leaf node includes a CB. As illustrated in FIG. 2, a picture included in a GOP may include slices, where each slice includes a sequence of CTUs and each CTU may be partitioned according to a QTBT structure. In JEM, CBs are used for prediction without any further partitioning. That is, in JEM, a CB may be a block of sample values on which the same prediction is applied. Thus, a JEM QTBT leaf node may be analogous a PB in ITU-T H.265.

[0019] Intra prediction data (e.g., intra prediction mode syntax elements) or inter prediction data (e.g., motion data syntax elements) may associate PUs with corresponding reference samples. Residual data may include respective arrays of difference values corresponding to each component of video data (e.g., luma (Y) and chroma (Cb and Cr)). 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 pixel difference values to generate transform coefficients. It should be noted that in ITU-T H.265, CUs may be further sub-divided into Transform Units (TUs). That is, an array of pixel difference values may be sub-divided for purposes of generating transform coefficients (e.g., four 8 x 8 transforms may be applied to a 16 x 16 array of residual values corresponding to a 16 x16 luma CB), such sub-divisions may be referred to as Transform Blocks (TBs). Transform coefficients may be quantized according to a quantization parameter (QP). Quantized transform coefficients (which may be referred to as level values) may be entropy coded according to an entropy encoding technique (e.g., content adaptive variable length coding (CAVLC), context adaptive binary arithmetic coding (CABAC), probability interval partitioning entropy coding (PIPE), etc.). Further, syntax elements, such as, a syntax element indicating a prediction mode, may also be entropy coded. Entropy encoded quantized transform coefficients and corresponding entropy encoded syntax elements may form a compliant bitstream that can be used to reproduce video data. A binarization process may be performed on syntax elements as part of an entropy coding process. Binarization refers to the process of converting a syntax value into a series of one or more bits. These bits may be referred to as "bins."

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. As described above, intra prediction data or inter prediction data may associate an area of a picture (e.g., a PB or a CB) with corresponding reference

samples. 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 (predMode: 0), a DC (i.e., flat overall averaging) prediction mode (predMode: 1), and 33 angular prediction modes (predMode: 2-34). In JEM, defined possible intra-prediction modes include a planar prediction mode (predMode: 0), a DC prediction mode (predMode: 1), and 65 angular prediction modes (predMode: 2-66). 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.

[0020] For inter prediction coding, a motion vector (MV) identifies reference samples in a picture other than the picture of a video block to be coded and thereby exploits temporal redundancy in video. For example, a current video block may be predicted from reference block(s) located in previously coded frame(s) and a motion vector may be used to indicate the location of the reference block. A motion vector and associated data may describe, for example, a horizontal component of the motion vector, a vertical component of the motion vector, 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), a prediction direction and/or a reference picture index value. Further, a coding standard, such as, for example ITU-T H.265, may support motion vector prediction. Motion vector prediction enables a motion vector to be specified using motion vectors of neighboring blocks. 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, JEM supports advanced temporal motion vector prediction (ATMVP), Spatial-temporal motion vector prediction (STMVP), Pattern matched motion vector derivation (PMMVD) mode, which is a special merge mode based on Frame-Rate Up Conversion (FRUC) techniques, and affine transform motion compensation prediction.

[0021] 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. In ITU-T H.265, a CU is associated with a transform unit (TU) structure having its root at the CU level. That is, in ITU-T H.265, as described above, an array of difference values may be subdivided for purposes of generating transform coefficients (e.g., four 8x8 transforms

may be applied to a 16x16 array of residual values). It should be noted that in ITU-T H.265, TBs are not necessarily aligned with PBs.

[0022] It should be noted that in JEM, residual values corresponding to a CB are used to generate transform coefficients without further partitioning. That is, in JEM a QTBT leaf node may be analogous to both a PB and a TB in ITU-T H.265. It should be noted that in JEM, 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. Further, in JEM, whether a secondary transform is applied to generate transform coefficients may be dependent on a prediction mode.

[0023] A quantization process may be performed on transform coefficients. Quantization approximates transform coefficients by amplitudes restricted to a set of specified values. Quantization may be used in order to vary the amount of data required to represent a group of transform coefficients. Quantization may be realized through division of transform coefficients by a 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 scaling factor. 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 or 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.

[0024] 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.

Further, the following mathematical functions may be used:

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

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

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

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

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.

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

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

As described above, 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. FIG. 2 is a conceptual diagram illustrating an example of a group of pictures including slices. In the example illustrated in FIG. 2, Pic4 is illustrated as including two slices (i.e., Slice1 and Slice2) where each slice includes a sequence of CTUs (e.g., in raster scan order). It should be noted that a slice 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) within the same access unit. A slice segment, like a slice, is a sequence of coding tree units. In the examples described herein, in some cases the terms slice and slice segment may be used interchangeably to indicate a sequence of coding tree units. It should be noted that in ITU-T H.265, a tile may consist of coding tree units contained in more than one slice and a slice may consist of coding tree units 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 coding tree units in a slice belong to the same tile; and (2) All coding tree units in a tile belong to the same slice. Tile sets may be used to define boundaries for coding dependencies (e.g., intra-prediction dependencies, entropy encoding dependencies, etc.,) and as such, may enable parallelism in coding.

[0025] In ITU-T H.265, 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. In ITU-T H.265, a bitstream is described as including a sequence of NAL units forming one or more CVSs. It should be noted that

ITU-T H.265 supports multi-layer extensions, including format range extensions (REext), scalability (SHVC), multi-view (MV-HEVC), and 3-D (3D-HEVC). 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., High Definition rendering) 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) to be presented. In ITU-T H.265, 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 prediction techniques) by referencing one or more pictures (including scaled versions thereof) in a base layer. In ITU-T H.265, 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 bitstream forms a new compliant bitstream by discarding and/or modifying data in the received bitstream. For example, sub-bitstream extraction may be used to form a new compliant bitstream corresponding to a particular representation of video (e.g., a high quality representation).

[0026] Referring to the example illustrated in FIG. 2, each slice of video data included in Pic4 (i.e., Slice1 and Slice2) is illustrated as being encapsulated in a NAL unit. In ITU-T H.265, each of a video sequence, a GOP, a picture, a slice, and CTU may be associated with metadata that describes video coding properties. ITU-T H.265 defines parameters sets that may be used to describe video data and/or video coding properties. In ITU-T H.265, 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, ITU-T H.265 enables supplemental enhancement information (SEI) messages to be signaled. In ITU-T H.265, 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 ITU-T H.265, 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).

[0027] 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 units (i.e., Video Parameter Sets (VPS), Sequence Parameter Sets (SPS), and Picture Parameter Set (PPS) units) and an access unit delimiter NAL unit. ITU-T H.265 defines NAL unit

header semantics that specify the type of Raw Byte Sequence Payload (RBSP) data structure included in the NAL unit.

[0028] Table 1 illustrates the general NAL unit syntax provided in ITU-T H.265 and which is additionally used in JVET-J1001.

[0029]

nal_unit(NumBytesInNalUnit) {	**Descriptor**
nal_unit_header( )	
NumBytesInRbsp = 0	
for(i = 2; i < NumBytesInNalUnit; i++)	
if(i + 2 < NumBytesInNalUnit && next_bits(24) == 0x000003) {	
rbsp_byte[NumBytesInRbsp++]	b(8)
rbsp_byte[NumBytesInRbsp++]	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRbsp++]	b(8)
}	

Table 1

ITU-T H.265 provides the following general NAL unit semantics:

NumBytesInNalUnit specifies the size of the NAL unit in bytes. This value is required for decoding of the NAL unit. Some form of demarcation of NAL unit boundaries is necessary to enable inference of NumBytesInNalUnit. One such demarcation method is specified in [Annex B of ITU-T H.265] for the byte stream format. Other methods of demarcation may be specified outside of this Specification.

NOTE 1 – The video coding layer (VCL) is specified to efficiently represent the content of the video data. The NAL is specified to format that data and provide header information in a manner appropriate for conveyance on a variety of communication channels or storage media. All data are contained in NAL units, each of which contains an integer number of bytes. A NAL unit specifies a generic format for use in both packet-oriented and bitstream systems. The format of NAL units for both packet-oriented transport and byte stream is identical except that each NAL unit can be preceded by a start code prefix and extra padding bytes in the byte stream format specified [Annex B of ITU-T H.265].

rbsp_byte[i] is the i-th byte of an RBSP. An RBSP is specified as an ordered sequence of bytes as follows:

The RBSP contains an string of data bits (SODB) as follows:

- If the SODB is empty (i.e., zero bits in length), the RBSP is also empty.
- Otherwise, the RBSP contains the SODB as follows:
  1) The first byte of the RBSP contains the (most significant, left-most) eight bits of the SODB; the next byte of the RBSP contains the next eight bits of the SODB, etc., until fewer than eight bits of the SODB remain.
  2) rbsp_trailing_bits( ) are present after the SODB as follows:
     i) The first (most significant, left-most) bits of the final RBSP byte contains the remaining bits of the SODB (if any).
     ii) The next bit consists of a single rbsp_stop_one_bit equal to 1.
     iii) When the rbsp_stop_one_bit is not the last bit of a byte-aligned byte, one or more rbsp_alignment_zero_bit is present to result in byte alignment.
  3) One or more cabac_zero_word 16-bit syntax elements equal to 0x0000 may be present in some RBSPs after the rbsp_trailing_bits( ) at the end of the RBSP.

Syntax structures having these RBSP properties are denoted in the syntax tables using an "_rbsp" suffix. These structures are carried within NAL units as the content of the rbsp_byte[i] data bytes. The association of the RBSP syntax structures to the NAL units is as specified in [Table 3, herein].

NOTE 2 – When the boundaries of the RBSP are known, the decoder can extract the SODB from the RBSP by concatenating the bits of the bytes of the RBSP and discarding the rbsp_stop_one_bit, which is the last (least significant, right-most) bit equal to 1, and discarding any following (less significant, farther to the right) bits that follow it, which are equal to 0. The data necessary for the decoding process is contained in the SODB part of the RBSP.

emulation_prevention_three_byte is a byte equal to 0x03. When an emulation_prevention_three_byte is present in the NAL unit, it shall be discarded by the decoding process.

The last byte of the NAL unit shall not be equal to 0x00.

Within the NAL unit, the following three-byte sequences shall not occur at any byte-aligned position:

- 0x000000
- 0x000001
- 0x000002

Within the NAL unit, any four-byte sequence that starts with 0x000003 other than the following sequences shall not occur at any byte-aligned position:

- 0x00000300
- 0x00000301
- 0x00000302
- 0x00000303

Table 2 illustrates the NAL unit header syntax provided in ITU-T H.265.

[0030]

nal_unit_header( ) {	**Descriptor**
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
nuh_layer_id	u(6)
nuh_temporal_id_plus1	u(3)
}	

Table 2

ITU-T H.265 provides the following definitions for the respective syntax elements illustrated in Table 2.

forbidden_zero_bit shall be equal to 0.

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.

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

With respect to **nal_unit_type**, **nal_unit_type** specifies the type of RBSP data structure contained in the NAL unit. Table 3 illustrates the NAL unit types provided in ITU-T H.265.

nal_unit_type	Name of nal_unit_type	Content of NAL unit and RBSP syntax structure	NAL unit type class
0 1	TRAIL_N TRAIL_R	Coded slice segment of a non-TSA, non-STSA trailing picture slice_segment_layer_rbsp( )	VCL
2 3	TSA_N TSA_R	Coded slice segment of a TSA picture slice_segment_layer_rbsp( )	VCL
4 5	STSA_N STSA_R	Coded slice segment of a STSA picture slice_segment_layer_rbsp( )	VCL
6 7	RADL_N RADL_R	Coded slice segment of a random access decodable leading (RADL) picture slice_segment_layer_rbsp( )	VCL
8 9	RASL_N RASL_R	Coded slice segment of a random access skipped leading (RASL) picture slice_segment_layer_rbsp( )	VCL
10 12 14	RSV_VCL_N10 RSV_VCL_N12 RSV_VCL_N14	Reserved non-IRAP SLNR VCL NAL unit types	VCL
11 13 15	RSV_VCL_R11 RSV_VCL_R13 RSV_VCL_R15	Reserved non-IRAP sub-layer reference VCL NAL unit types	VCL
16 17 18	BLA_W_LP BLA_W_RADL BLA_N_LP	Coded slice segment of a BLA picture slice_segment_layer_rbsp( )	VCL

19 20	IDR_W_RADL IDR_N_LP	Coded slice segment of a IDR picture slice_segment_layer_rbsp( )	VCL
21	CRA_NUT	Coded slice segment of a CRA picture slice_segment_layer_rbsp( )	VCL
22 23	RSV_IRAP_VCL22 RSV_IRAP_VCL23	Reserved IRAP VCL NAL unit types	VCL
24..31	RSV_VCL24.. RSV_VCL31	Reserved non-IRAP VCL NAL unit types	VCL
32	VPS_NUT	Video parameter set video_parameter_set_rbsp( )	non-VCL
33	SPS_NUT	Sequence parameter set seq_parameter_set_rbsp( )	non-VCL
34	PPS_NUT	Picture parameter set pic_parameter_set_rbsp( )	non-VCL
35	AUD_NUT	Access unit delimiter Access_unit_delimiter_rbsp( )	non-VCL
36	EOS_NUT	End of sequence end_of_seq_rbsp( )	non-VCL
37	EOB_NUT	End of bitstream end_of_bitstream_rbsp( )	non-VCL
38	FD_NUT	Filler data filler_data_rbsp( )	non-VCL
39 40	PREFIX_SEI_NUT SUFFIX_SEI_NUT	Supplemental enhancement information sei_rbsp( )	non-VCL
41..47	RSV_NVCL41..	Reserved	non-VCL

	RSV_NVCL47		
48..63	UNSPEC48.. UNSPEC63	Unspecified	non-VCL

Table 3

For the sake of brevity, a complete description of each of the NAL units types in ITU-T H.265 is not provided herein. However, reference is made to the relevant sections of ITU-T H.265.

[0031] As described above, JVET-J1001 is a draft text of a video coding specification. Table 4 illustrates the syntax of the NAL unit header in JVET-J1001.

[0032]

nal_unit_header( ) {	**Descriptor**
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
}	

Table 4

JVET-J1001 provides the following definitions for the respective syntax elements illustrated in Table 4.

forbidden_zero_bit shall be equal to 0.

With respect to **nal_unit_type**, JVET-J1001 provides where the NAL unit types are yet to be defined.

JVET-J1001 further provides a basic sequence parameter set syntax. Table 5 illustrates the syntax of the sequence parameter set provided in JVET-J1001.

seq_parameter_set_rbsp( ) {	**Descriptor**
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)
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
log2_ctu_size_minus2	ue(v)
log2_min_qt_size_intra_slices_minus2	ue(v)
log2_min_qt_size_inter_slices_minus2	ue(v)
max_mtt_hierarchy_depth_inter_slices	ue(v)
max_mtt_hierarchy_depth_intra_slices	ue(v)
rbsp_trailing_bits( )	
}	

Table 5

The basic definitions of the respective syntax elements illustrated in Table 5 are as follows:.

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. 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.

pic_width_in_luma_samples specifies the width of each decoded picture in units of luma samples.

pic_height_in_luma_samples specifies the height of each decoded picture in units of luma samples.

bit_depth_luma_minus8 specifies the bit depth of the samples of the luma array and the value of the luma quantization parameter range offset.

bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays and the value of the chroma quantization parameter range offset.

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

log2_min_qt_size_intra_slices_minus2 plus 2 specifies the minimum luma size of a leaf block resulting from quadtree splitting of a CTU in slices with slice_type equal to 2 (I).

log2_min_qt_size_inter_slices_minus2 plus 2 specifies the minimum luma size of a leaf block resulting from quadtree splitting of a CTU in slices with slice_type equal to 0 (B) or 1 (P).

max_mtt_hierarchy_depth_inter_slices specifies the 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).

max_mtt_hierarchy_depth_intra_slices specifies the 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).

JVET-J1001 further provides a basic picture parameter set syntax. Table 6 illustrates the syntax of the picture parameter set in JVET-J1001.

[0033]

pic_parameter_set_rbsp( ) {	**Descriptor**
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
rbsp_trailing_bits( )	
}	

Table 6

JVET-J1001 provides the following definitions for the respective syntax elements illustrated in Table 6.

pps_pic_parameter_set_id identifies the PPS for reference by other syntax elements. The value of pps_pic_parameter_set_id shall be in the range of 0 to 63, inclusive.

pps_seq_parameter_set_id specifies the value of sps_seq_parameter_set_id for the active SPS. The value of pps_seq_parameter_set_id shall be in the range of 0 to 15, inclusive.

JVET-J1001 further provides an access unit delimiter syntax. Table 7 illustrates the syntax of the access unit delimiter in JVET-J1001.

[0034]

access_unit_delimiter_rbsp( ) {	**Descriptor**
pic_type	u(3)
rbsp_trailing_bits( )	
}	

Table 7

JVET-J1001 provides the following definitions for the respective syntax elements illustrated in Table 7.

pic_type indicates that the slice_type values for all slices of the coded picture in the access unit containing the access unit delimiter NAL unit are members of the set listed in Table 8 for the given value of pic_type. The value of pic_type shall be equal to 0, 1 or 2 in bitstreams conforming to this version of this Specification. Other values of pic_type are reserved for future use. Decoders conforming to this version of this Specification shall ignore reserved values of pic_type.

pic_type	**slice_type values that may be present in the coded picture**
0	I
1	P, I
2	B, P, I

Table 8

It should be noted that a B slice refers to a slice where bi-prediction inter prediction, uni-prediction inter prediction, and intra predication are allowed; a P slice refers to a slice where uni-prediction inter prediction, and intra predication are allowed; and an I slice refers where only intra predication is allowed. It should be noted that in some cases B and P slices are collectively referred to as inter slices.

[0035] JVET-J1001 further provides an end of sequence syntax, an end of bitstream syntax, and a filler data syntax. Table 9 illustrates the end of sequence syntax provided in JVET-J1001, Table 10 illustrates the end of bitstream syntax provided in JVET-J1001, and Table 11 illustrates the end of bitstream syntax provided in JVET-J1001.

[0036]

end_of_seq_rbsp( ) {	**Descriptor**
}	

Table 9

end_of_bitstream_rbsp( ) {	**Descriptor**
}	

Table 10

filler_data_rbsp( ) {	**Descriptor**
while(next_bits(8) == 0xFF)	
ff_byte /* equal to 0xFF */	f(8)
rbsp_trailing_bits( )	
}	

Table 11

JVET-J1001 provides the following definitions for the respective syntax elements illustrated in Table 11.

ff_byte is a byte equal to 0xFF.

JVET-J1001 further provides a slice layer syntax including a slice header. Table 12 illustrates the slice layer syntax provided in JVET-J1001 and Table 13 illustrates the slice header provided in JVET-J1001.

[0037]

slice_layer_rbsp( ) {	**Descriptor**
slice_header( )	
slice_data( )	
rbsp_slice_trailing_bits( )	
}	

Table 12

slice_header( ) {	**Descriptor**
slice_pic_parameter_set_id	ue(v)
slice_address	u(v)
slice_type	ue(v)
if(slice_type != I)	
log2_diff_ctu_max_bt_size	ue(v)
byte_alignment( )	
}	

Table 13

JVET-J1001 provides the following definitions for the respective syntax elements illustrated in Table 13.

slice_pic_parameter_set_id specifies the value of pps_pic_parameter_set_id for the PPS in use. The value of slice_pic_parameter_set_id shall be in the range of 0 to 63, inclusive.

slice_address specifies the address of the first CTB in the slice, in CTB raster scan of a picture.

slice_type specifies the coding type of the slice according to Table 14.

slice_type	Name of slice_type
0	B (B slice)
1	P (P slice)
2	I (I slice)

Table 14

When nal_unit_type has a value in the range of [to be determined], inclusive, i.e., the picture is an IRAP picture, slice_type shall be equal to 2.

log2_diff_ctu_max_bt_size specifies the difference between the luma CTB size and the maximum luma size (width or height) of a coding block that can be split using a binary split.

This disclosure describes a high-level syntax and semantics that enable the grouping of related syntax elements in syntax structure groups; enable the ability to skip past one or more groups of syntax structures without parsing them, because, for example, a syntax structure is not used due to one or more related lower level coding tools not being used by a given profile, and enable the ability for a decoder to quickly access only relevant syntax elements without having to parse all the previous syntax elements which appear before the syntax element of interest, which may be especially applicable to signaling of various indicators and signaling in video usability information. It should be noted that in some cases high-level syntax may refer to syntax occurring prior to slice_data().

[0038] 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.

[0039] 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.

[0040] 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.

[0041] 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.

[0042] 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.

[0043] 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.

[0044] 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.

[0045] 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.

[0046] 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.

[0047] 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 mac-

roblocks, 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.

[0048] 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 sub-divisions 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.

[0049] 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).

[0050] 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.

[0051] 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 disclose.

[0052] 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.

[0053] In one example, data encapsulator 107 may be configured to generate high-level 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.

[0054] As described above, Table 1 illustrates the general NAL unit syntax provided in JVET-J1001. In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal a general syntax structure for signaling seq_parameter_set_rbsp( ), pic_parameter_set_rbsp( ), end_of_seq_rbsp( ), and/or

end_of_bitstream_rbsp( ). Table 15 provides an example of a general syntax structure.

[0055]

syntax_struct_rbsp(syntaxStructureType, syntaxStructureLength){	**Descriptor**
if(syntaxStructureType==1 && nal_unit_type == SPS_NUT)	
representation_format(syntaxStructureLength)	
else if(syntaxStructureType==2 && nal_unit_type == SPS_NUT)	
ctu_params(syntaxStructureLength)	
else if(syntaxStructureType==3 && nal_unit_type == PPS_NUT)	
pps_params(syntaxStructureLength)	
else	
reserved_syntax_structure(syntaxStructureLength)	
}	

Table 15

In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal sequence parameter set syntax and a **representation_format**(syntaxStructureLength) and a **ctu_params**(syntaxStructureLength) according to Tables 16, 17, and 18. It should be noted that in Tables 16, 17, and 18, commonly named syntax elements as those described above may have similar respective definitions to those described above.

seq_parameter_set_rbsp( ) {	**Descriptor**
sps_seq_parameter_set_id	ue(v)
num_syntax_structures	ue(v)
for(i=0;i<num_syntax_structures;i++) {	
syntax_struct_type[i]	ue(v)
syntax_struct_length[i]	ue(v)
}	
while(!byte_aligned( ))	
sps_alignment_zero_bit	u(1)
for(i=0;i<num_syntax_structures;i++)	
syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i])	
}	

Table 16

representation_format(syntaxStructureLength) {	
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)
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
rbsp_trailing_bits( )	
}	

Table 17

ctu_params(syntaxStructureLength) {	
log2_ctu_size_minus2	ue(v)
log2_min_qt_size_intra_slices_minus2	ue(v)
log2_min_qt_size_inter_slices_minus2	ue(v)
max_mtt_hierarchy_depth_inter_slices	ue(v)
max_mtt_hierarchy_depth_intra_slices	ue(v)
rbsp_trailing_bits( )	
}	

Table 18

In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal picture parameter set syntax and a **pps_params**(syntaxStructureLength) according to Tables 19 and 20. It should be noted that in Tables 19 and 20, commonly named syntax elements as those described above may have similar respective definitions to those described above.

pic_parameter_set_rbsp( ) {	**Descriptor**
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
num_syntax_structures	ue(v)
for(i=0;i<num_syntax_structures;i++) {	
syntax_struct_type[i]	ue(v)
syntax_struct_length[i]	ue(v)
}	
while(!byte_aligned( ))	
pps_alignment_zero_bit	u(1)
for(i=0;i<num_syntax_structures;i++)	
syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i])	
}	

Table 19

pps_params(syntaxStructureLength) {	
num_extra_slice_header_bits	u(3)
rbsp_trailing_bits( )	
}	

Table 20

In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal an end of sequence syntax and an end of bitstream syntax according to Tables 21 and 22.

[0056]

end_of_seq_rbsp( ) {	**Descriptor**
num_syntax_structures //value signalled=0	u(16)
rbsp_trailing_bits( )	
}	

Table 21

end_of_seq_rbsp( ) {	**Descriptor**
num_syntax_structures //value signalled=0	u(16)
rbsp_trailing_bits( )	
}	

Table 22

With respect to Tables 16-22, one of more of the following examples definitions may be used for the respective syntax elements.

num_syntax_structures specifies the number of syntax structures that are specified in this nal_unit.

syntax_struct_type[i] specifies the structure type of the i-th syntax structure. syntax_struct_types are defined above.

syntax_struct_length[i] specifies the length in bytes of the i-th syntax structure that follows this syntax element. The length specified for the i-th syntax structure does not include the length of syntax_struct_type[i] and syntax_struct_length[i] elements.

syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i]) is the raw byte sequence payload of the syntax structure of the type syntax_struct_type[i] and of length syntax_struct_length[i]

sps_alignment_zero_bit shall be equal to 0.

pps_alignment_zero_bit shall be equal to 0.

num_extra_slice_header_bits specifies the number of extra slice header bits that are present in the slice header RBSP for coded pictures referring to the PPS. The value of num_extra_slice_header_bits shall be in the range of 0 to 2, inclusive, in bitstreams conforming to this version of this Specification. Other values for num_extra_slice_header_bits are reserved for future use. However, decoders shall allow num_extra_slice_header_bits to have any value.

In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal a slice_header() in conjunction with the general syntax structure provided above according to the example illustrated in Table 23. It should be noted that in Table 23 commonly named syntax elements as those described above may have similar respective definitions to those described above.

[0057]

slice_header( ) {	**Descriptor**
slice_pic_parameter_set_id	ue(v)
slice_address	u(v)
for(I = 0; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	**u(1)**
slice_type	ue(v)
if(slice_type != I)	
log2_diff_ctu_max_bt_size	ue(v)
byte_alignment( )	
}	

Table 23

With respect to Table 23, **slice_reserved_flag**[i] may be based on the following definition:

slice_reserved_flag[i] has semantics and values that are reserved for future use. Decoders shall ignore the presence and value of slice_reserved_flag[i].

In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal an **reserved_syntax_structure**(syntaxStructureLength) according to the example illustrated in Table 24.

reserved_syntax_struct(syntaxStructureLength) {	**Descriptor**
for(i=0; i< syntaxStructureLength; i++)	
reserved_syntax_structure_byte	b(8)
}	

Table 24

With respect to Table 24, **reserved_syntax_structure_byte** may be based on the following definition:

reserved_syntax_structure_byte is 8-bits of data reserved for future use.

It should be noted that according to the high-level syntax provided in Tables 15-24, in some examples, syntax for an access unit delimiter and filler data may not be provided. Further, it should be noted that according to the high-level syntax provided in Tables 15-24 multiple syntax structures of various types may be present in a NAL unit. In particular, various types of parameter sets and subsets thereof may be included in a NAL unit.

[0058] In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal a general syntax structure for signaling

seq_parameter_set_rbsp( ), pic_parameter_set_rbsp( ), end_of_seq_rbsp( ), and/or end_of_bitstream_rbsp( ) according to the example a general syntax structures illustrated in Table 25 and 26.

[0059]

nal_unit_rbsp(NumBytesInRbsp) {	**Descriptor**
num_syntax_structures	ue(v)
for(i=0;i<num_syntax_structures;i++) {	
syntax_struct_type[i]	ue(v)
syntax_struct_length[i]	ue(v)
}	
for(i=0;i<num_syntax_structures;i++) {	
syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i])	
}	
}	

Table 25

syntax_struct_rbsp(syntaxStructureType, syntaxStructureLength){	**Descriptor**
if(syntaxStructureType==1)	
sps_id_representation_format(syntaxStructureLength)	
else if(syntaxStructureType==2)	
ctu_params(syntaxStructureLength)	
else if(syntaxStructureType==3)	
pps_id_params(syntaxStructureLength)	
else	
reserved_syntax_structure(syntaxStructureLength)	
}	

Table 26

With respect to Table 25 and 26 **num_syntax_structures**, **syntax_struct_type[i]**, **syntax_struct_length[i]**, and **syntax_struct_rbsp** may be based on the following definitions:

num_syntax_structures specifies the number of syntax structures that are specified in this nal_unit.

syntax_struct_type[i] specifies the structure type of the i-th syntax structure. syntax_struct_types are defined below.

syntax_struct_length[i] specifies the length in bytes of the i-th syntax structure that follows this syntax element. The length specified for the i-th syntax structure does not include the length of syntax_struct_type[i] and syntax_struct_length[i] elements.

syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i]) is the raw byte sequence payload of the syntax structure of the type syntax_struct_type[i] and of length syntax_struct_length[i]

With respect to Table 26 **sps_id_representation_format**(), **ctu_params**(), **pps_id_params**(), and **reserved_syntax_structure**() may be based example syntax illustrated in respective Table 27, Table 28, Table 29, and Table 30. It should be noted that in Table 27, Table 28, Table 29, and Table 30, commonly named syntax elements as those described above may have similar respective definitions to those described above.

sps_id_representation_format(syntaxStructureLength) {	**Descriptor**
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)
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
}	

Table 27

ctu_params(syntaxStructureLength) {	**Descriptor**
log2_ctu_size_minus2	ue(v)
log2_min_qt_size_intra_slices_minus2	ue(v)
log2_min_qt_size_inter_slices_minus2	ue(v)
max_mtt_hierarchy_depth_inter_slices	ue(v)
max_mtt_hierarchy_depth_intra_slices	ue(v)
rbsp_trailing_bits( )	
}	

Table 28

pps_id_params(syntaxStructureLength) {	**Descriptor**
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
num_extra_slice_header_bits	u(3)
rbsp_trailing_bits( )	
}	

Table 29

reserved_syntax_struct(syntaxStructureLength) {	**Descriptor**
for(i=0; i< syntaxStructureLength; i++)	
reserved_syntax_structure_byte	b(8)
}	

Table 30

In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal a general syntax structure for signaling seq_parameter_set_rbsp( ), pic_parameter_set_rbsp( ), end_of_seq_rbsp( ), and/or end_of_bitstream_rbsp( ) according to the example a general syntax structures illustrated in Table 31. It should be noted that in Table 31 commonly named syntax elements as those described above may have similar respective definitions to those described above.

[0060]

nal_unit_rbsp(NumBytesInRBSP) {	**Descriptor**
i=0;	
NumRemBytes= NumBytesInRBSP;	
while(NumRemBytes>0) {	
syntax_struct_type[i]	u(16)
syntax_struct_length[i]	u(16)
syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i])	
NumRemBytes-= (syntax_struct_length[i]+4);	
i++;	
}	
}	

Table 31

In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal a general syntax structure for signaling seq_parameter_set_rbsp( ), pic_parameter_set_rbsp( ), end_of_seq_rbsp( ), and/or end_of_bitstream_rbsp( ) according to the example a general syntax structures illustrated in Table 32. It should be noted that in Table 32 commonly named syntax elements as those described above may have similar respective definitions to those described above.

[0061]

nal_unit_rbsp(NumBytesInRBSP) {	**Descriptor**
num_syntax_structures	u(16)
for(i=0;i<num_syntax_structures;i++) {	
syntax_struct_type[i]	u(16)
num_related_tools[i]	u(16)
for(j=0;j<num_related_tools[i];j++)	
related_tool_id[i][j]	u(16)
syntax_struct_length[i]	u(16)
}	
for(i=0;i<num_syntax_structures;i++) {	
syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i])	
}	
}	

Table 32

It should be noted that in Table 32 **num_related_tools[i]** and **related_tool_id[i][j]** may be based on the following definitions:

num_related_tools[i] specifies the number of tools that need to use the information in the i-th signaled syntax structure syntax_struct_rbsp(syntax_struct_length[i]).

related_tool_id[i][j] specifies the identifier of the j-th tool that needs to use the information in the i-th signaled syntax structure syntax_struct_rbsp(syntax_struct_length[i]).

In one example, one or more of the syntax elements related_tool_id[i][j] may be optionally signaled. The optional signaling be controlled by a presence flag surrounding these syntax elements. In this case, the values of the syntax elements related_tool_id[i][j] may be inferred when not signaled. In one example, the value of syntax elements related_tool_id[i][j] may be inferred based on the syntax_struct_type[i]. In one example, the value of related_tool_id[i][j] may be statically known, hard-coded, and/or signaled via a look-up table.

[0062] In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal a general syntax structure for signaling seq_parameter_set_rbsp( ), pic_parameter_set_rbsp( ), end_of_seq_rbsp( ), and/or end_of_bitstream_rbsp( ) according to the example a general syntax structures illustrated in Table 33. It should be noted that in Table 33 commonly named syntax elements as those described above may have similar respective definitions to those described above.

[0063]

nal_unit_rbsp(NumBytesInRBSP) {	**Descriptor**
num_syntax_structures	u(16)
for(i=0;i<num_syntax_structures;i++) {	
syntax_struct_type[i]	u(16)
required_syntax_struct_flag[i]	u(1)
independent_syntax_struct_flag[i]	u(1)
reserved_zero_6bits	u(6)
syntax_struct_length[i]	u(16)
}	
for(i=0;i<num_syntax_structures;i++) {	
syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i])	
}	
}	

Table 33

It should be noted that in Table 33 **required_syntax_struct_flag[i]**, **independent_syntax_struct_flag[i]**, and **reserved_zero_6bits** may be based on the following definitions:

required_syntax_struct_flag[i] equal to 1 specifies that the information in the i-th signaled syntax structure syntax_struct_rbsp(syntax_struct_length[i]) is required for decoding of the bitstream irrespective of tools used. required_syntax_struct_flag[i] equal to 0 specifies that the information in the i-th signaled syntax structure syntax_struct_rbsp(syntax_struct_length[i]) is required for decoding of the bitstream only when certain tools are used.
Alternatively, **required_syntax_struct_flag[i]** may be a 2-bit field, taking one of 3 different values: a first value indicating that data in syntax_struct_rbsp() is required for parsing the coded bitstream, a second value indicating that data in syntax_struct_rbsp() is required for correctly reconstructing samples of the coded bitstream, and a third value indicating that data in syntax_struct_rbsp() is not required for correctly reconstructing samples of the coded bitstream.

independent_syntax_struct_flag[i] equal to 1 specifies that the information in the i-th signaled syntax structure syntax_struct_rbsp(syntax_struct_length[i]) can be decoded independently without any other information from other syntax structures. independent_syntax_struct_flag[i] equal to 0 specifies that the information in the i-th signaled syntax structure syntax_struct_rbsp(syntax_struct_length[i]) may or may not be decodable independently without some information from other syntax structures. (In a variant: **independent_syntax_struct_flag[i]** equal to 0 specifies that the information in the i-th signaled syntax structure syntax_struct_rbsp(syntax_struct_length[i]) can not be decoded independently without some information from other syntax structures.)

reserved_zero_6bits shall be equal to '000000'. Other values of reserved_zero_6bits may be specified in the future. Decoders shall ignore (i.e., remove from the bitstream and discard) NAL units with values of reserved_zero_6bits not equal to '000000'.

In one example, one or more of the following constraints may be imposed, with respect to Table 33:
Syntax structures with independent_syntax_struct_flag[i] equal to 1 shall be signaled first followed by syntax structures with independent_syntax_struct_flag[i] equal to 0.

[0064] In one example, one or more of the syntax elements required_syntax_struct_flag[i] and independent_syntax_struct_flag[i] may be optionally signaled. This may be controlled by a presence flag surrounding these syntax elements. In this case, the values of the syntax elements required_syntax_struct_flag[i], independent_syntax_struct_flag[i] may be inferred when not signaled. In one example, the value of syntax elements required_syntax_struct_flag[i], independent_syntax_struct_flag[i] may be inferred based on the syntax_struct_type[i]. In

one example, the value of required_syntax_struct_flag[i], independent_syntax_struct_flag[i] may be statically known, hard-coded, and/or signalled via a look-up table.

[0065] In one example, according to the techniques described herein, data encapsulator 107 may be configured to signal a general syntax structure for signaling seq_parameter_set_rbsp( ), pic_parameter_set_rbsp( ), end_of_seq_rbsp( ), and/or end_of_bitstream_rbsp( ) according to the example a general syntax structures illustrated in Table 34. It should be noted that in Table 34 commonly named syntax elements as those described above may have similar respective definitions to those described above. It should be noted that in Table 34, the last syntax structure is kept variable length (based on NAL unit size) and its length is not explicitly signaled. This may allow better future extensibility (e.g., by not signaling syntax_struct_type and signalling the last syntax structure as syntax_struct_rbsp(0xFFFF,NumRemBytes-4), where 0xFFFF syntax_structure_type value may be reserved. In another example, instead of 0xFFFF, some other agreed value may be signaled.). Also, in this embodiment Table 34 can save bits, as some of the syntax elements are not explicitly signaled for the last syntax structure.

[0066]

nal_unit_rbsp(NumBytesInRBSP) {	**Descriptor**
NumRemBytes= NumBytesInRBSP;	
num_fixedlen_syntax_structures	u(16)
for(i=0;i<num_fixedlen_syntax_structures;i++) {	
syntax_struct_type[i]	u(16)
required_syntax_struct_flag[i]	u(1)
independent_syntax_struct_flag[i]	u(1)
reserved_zero_6bits	u(6)
syntax_struct_length[i]	u(16)
}	
for(i=0;i<num_fixedlen_syntax_structures;i++) {	
syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i])	
NumRemBytes-= (syntax_struct_length[i]+5);	
}	
syntax_struct_type[num_fixedlen_syntax_structures]	u(16)
syntax_struct_rbsp(syntax_struct_type[num_fixedlen_syntax_structures],NumRemBytes-4)	
}	

Table 34

It should be noted that in Table 34, **num_fixedlen_syntax_structures** may be based on the following definitions:

num_fixedlen_syntax_structures specifies the number of syntax structures with expliclity signaled length that are specified in this nal_unit.

With respect to Table 34, in one example the syntax_struct_type[num_fixedlen_syntax_structures] may not be signaled for the last variable length syntax structure.

[0067] It should be noted that with respect to the examples above, although some syntax elements use u(16) some other fixed length or variable length signaling may be used instead. For example one or more u(16) may be changed to u(8) or u(12) or u(v) or ue(v). Additionally, in cases were u(v) coding is use, the number of bits used for u(v) coding may be signaled via another syntax element which may be ue(v) coded.

[0068] In another example, a coding such as the following may be applied for one or more syntax elements:

1xxx xxxx for values 0-127

01xx xxxx xxxx xxxx for values 128-16383

001x xxxx xxxx xxxx xxxx xxxx for values 16384-2097151

In another example, a coding such as the following may be applied for one or more syntax elements:

1xxx for values 0-7

01xx xxxx xxxx for values 8-1023

001x xxxx xxxx xxxx xxxx for values 1024-131071

Further, it should be noted that in some examples, with respect to the syntax elements above, minus 1 signaling may be used to indicate a number, where a signal value plus one specifies a number. For example:

Syntax element syntax_struct_length[i] may be instead signaled as syntax_struct_length_minus1[i], in which case its semantics may be for example as follows:

syntax_struct_length_minus1[i] plus 1 specifies the length in bytes of the i-th syntax structure that follows this syntax element. The length specified for the i-th syntax structure does not include the length of syntax_struct_type[i], num_related_tools and related_tool_id[i][j] and syntax_struct_length[i] elements.

Syntax element num_syntax_structures may be instead signaled as num_syntax_structures_minus1, in which case its semantics may be for example as follows:

num_syntax_structures_minus1 plus 1 specifies the number of syntax structures that are specified in this nal_unit.

Syntax element num_related_tools[i] may be instead signaled as num_related_tools_minus1[i], in which case its semantics may be for example as follows:

num_related_tools_minus1[i] plus 1 specifies the number of tools that need to use the information in the i-th signaled syntax structure syntax_struct_rbsp(syntax_struct_length[i]).

In one example, instead of syntax_struct_rbsp(syntax_struct_type[i], syntax_struct_length[i]), the syntax structure RBSP may be indicated as syntax_struct_rbsp(syntax_struct_length[i]).

[0069] It should be noted that, although the above example syntax is shown for any NAL unit type, in some examples, the above syntax structure may only be applied to certain types of NAL units. For example, the above syntax structure may only be applied to Parameter Set NAL units. In another example, the above syntax structure may only be applied to VCL NAL units. In another example, the above syntax structure may only be applied to Non-VCL NAL units.

[0070] In one example, certain values of syntax_struct_type[i] may be kept reserved for

future extensibility and/or certain values of syntax_struct_type[i] may be allocated for application dependent and/or private signaling.

[0071] It should be noted that the flexible syntax described herein may be used to enable features (or tools) in the bitstream. For example features may be disabled by default, and may be enabled when a corresponding syntax_struct_rbsp() is encoded in the bitstream. In one example, such a syntax_struct_rbsp() may have zero length. Alternatively, such a syntax_struct_rbsp() may contain one or more flags that explicitly enable/disable features. The benefit of this approach is that no parsing associated with a given feature is required when the feature is not used.

[0072] In this manner, source device 102 represents an example of a device configured to signal a value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit, and for each the number of syntax structures, signaling a value for a syntax element indicating a syntax structure type.

[0073] 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, I2C, or any other logical and physical structure that may be used to interconnect peer devices.

[0074] 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, I2C, 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.

[0075] Video decoder 124 may include any device configured to receive a bitstream (e.g., a MCTS 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.

[0076] 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. 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 1-24. Video decoder 600 may perform video decoding based on the values of parsed syntax elements. For example, different video decoding techniques may be performed based on whether a picture is of a particular type.

[0077] 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.

[0078] 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.

[0079] 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 value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit, and for each the number of syntax structures parse a value for syntax element indicating a syntax structure type, and generate video data based on values of the parsed syntax elements.

[0080] 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.

[0081] 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.

[0082] 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.

[0083] 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.

[0084] 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.

[0085] Various examples have been described. These and other examples are within the scope of the following claims.

[0086] <Cross Reference>

This Nonprovisional application claims priority under 35 U.S.C. § 119 on provisional Application No. 62/693,898 on July 3, 2018, No. 62/697,257 on July 12, 2018, the entire contents of which are hereby incorporated by reference.

Claims

[Claim 1] A method of signaling picture type information, the method comprising:
signaling value for a syntax element in a network abstraction layer unit indicating a number of syntax structures; and
for each the number of syntax structures, signaling a value of a syntax element indicating a syntax structure type.

[Claim 2] A method of decoding video data, the method comprising:
parsing a value for a syntax element in a network abstraction layer unit indicating a number of syntax structures included in the network abstraction layer unit;
for each the number of syntax structures parsing a value for syntax element indicating a syntax structure type; and
generating video data based on values of the parsed syntax elements.

[Claim 3] The method of any of claims 1 and 2, wherein a syntax structure includes a parameter set and a value indicating a syntax structure type indicates a parameter set type.

[Claim 4] A device comprising one or more processors configured to perform any and all combinations of the steps of claims 1-3.

[Claim 5] The device of claim 4, wherein the device includes a video encoder.

[Claim 6] The device of claim 4, wherein the device includes a video decoder.

[Claim 7] A system comprising:
the device of claim 5; and
the device of claim 6.

[Claim 8] An apparatus comprising means for performing any and all combinations of the steps of claims 1-3.

[Claim 9] A non-transitory computer-readable storage medium comprising instructions stored thereon that, when executed, cause one or more processors of a device to perform any and all combinations of the steps of claims 1-3.

[Fig. 1]

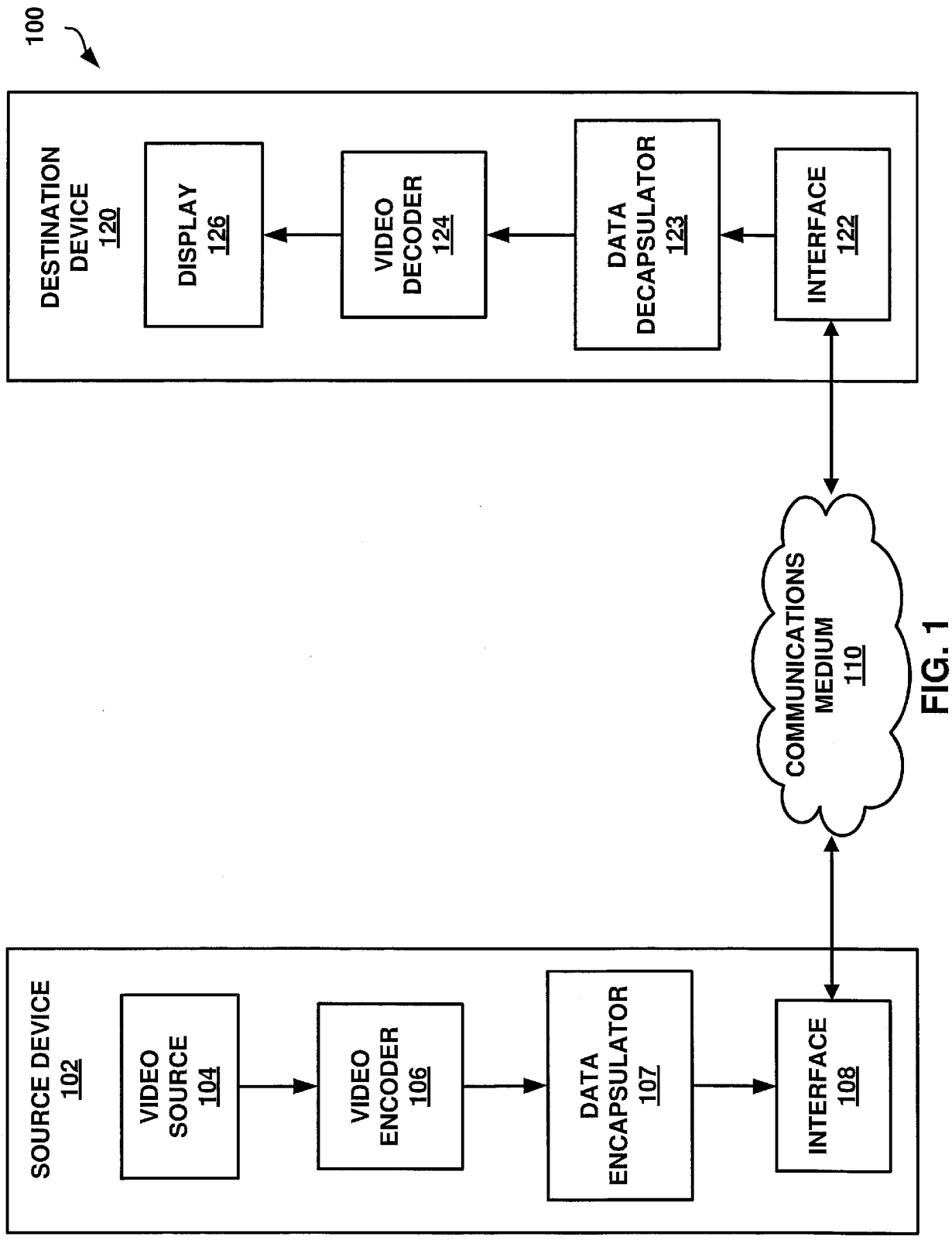
100
DESTINATION DEVICE 120
DISPLAY 126
VIDEO DECODER 124
DATA DECAPSULATOR 123
INTERFACE 122
COMMUNICATIONS MEDIUM 110
SOURCE DEVICE 102
VIDEO SOURCE 104
VIDEO ENCODER 106
DATA ENCAPSULATOR 107
INTERFACE 108

FIG. 1

[Fig. 2]

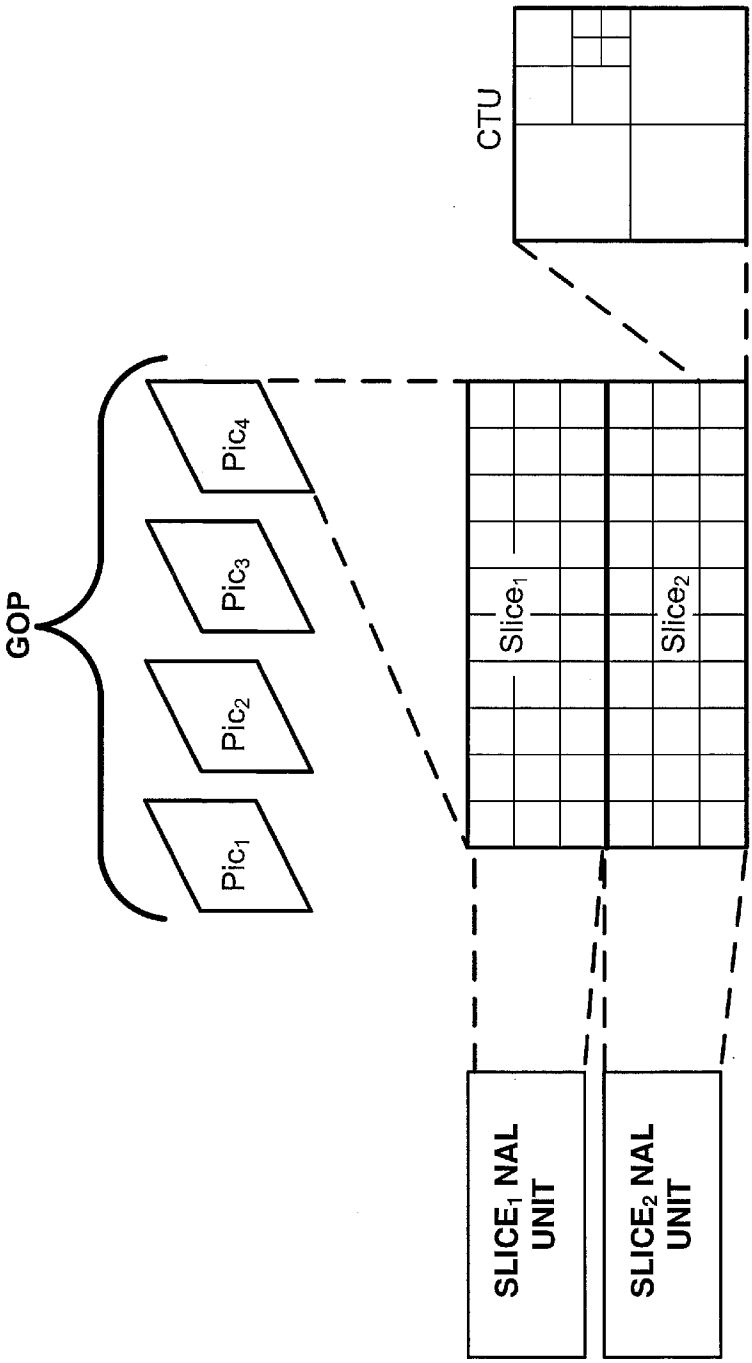
CTU
GOP
Pic4
Pic3
Pic2
Pic1
Slice1
Slice2
SLICE1 NAL UNIT
SLICE2 NAL UNIT

FIG. 2

[Fig. 3]

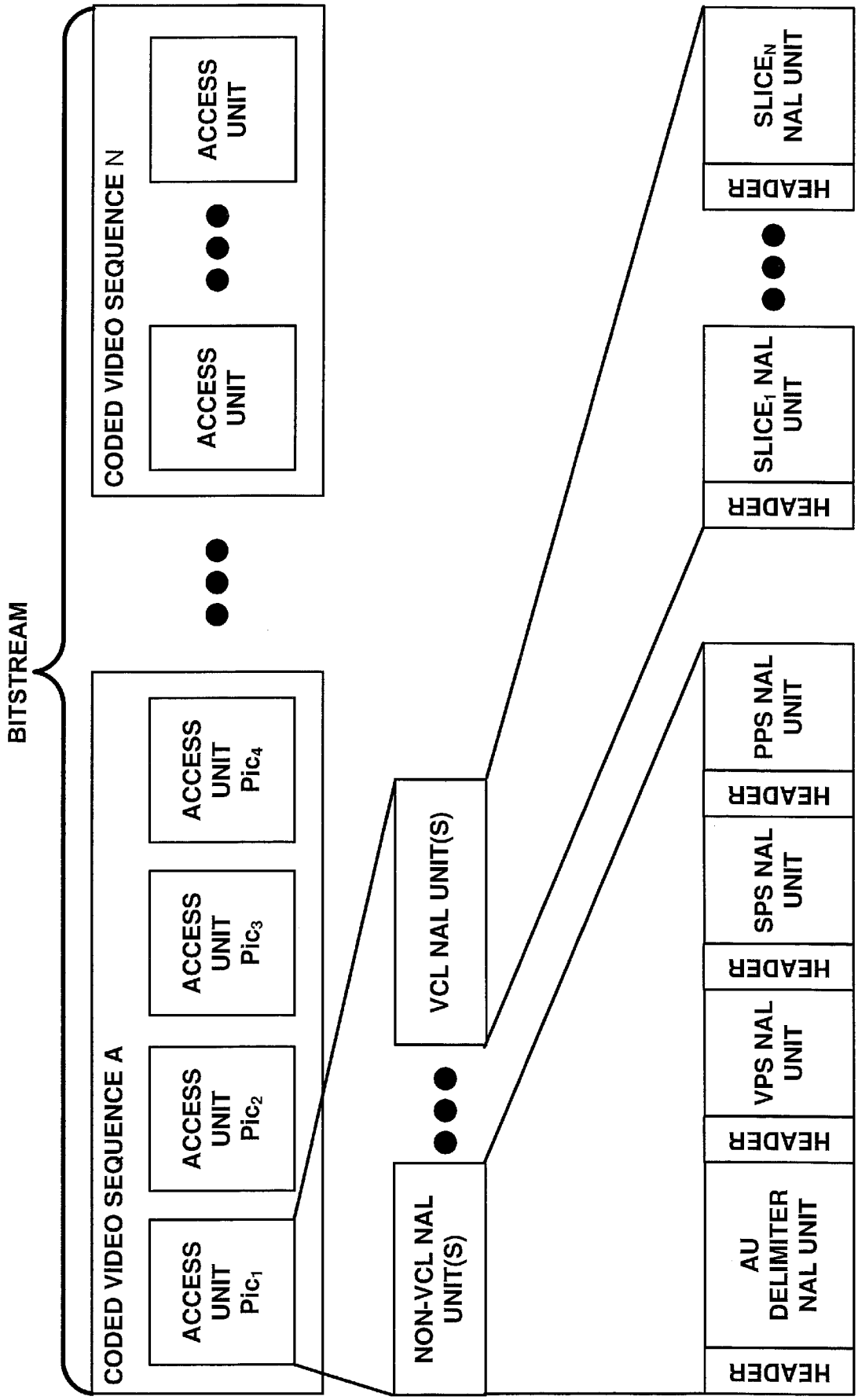
BITSTREAM
CODED VIDEO SEQUENCE A
ACCESS UNIT Pic1
ACCESS UNIT Pic2
ACCESS UNIT Pic3
ACCESS UNIT Pic4
CODED VIDEO SEQUENCE N
ACCESS UNIT
ACCESS UNIT
NON-VCL NAL UNIT(S)
VCL NAL UNIT(S)
HEADER
AU DELIMITER NAL UNIT
HEADER
VPS NAL UNIT
HEADER
SPS NAL UNIT
HEADER
PPS NAL UNIT
HEADER
SLICE1 NAL UNIT
HEADER
SLICEN NAL UNIT

FIG. 3

[Fig. 4]

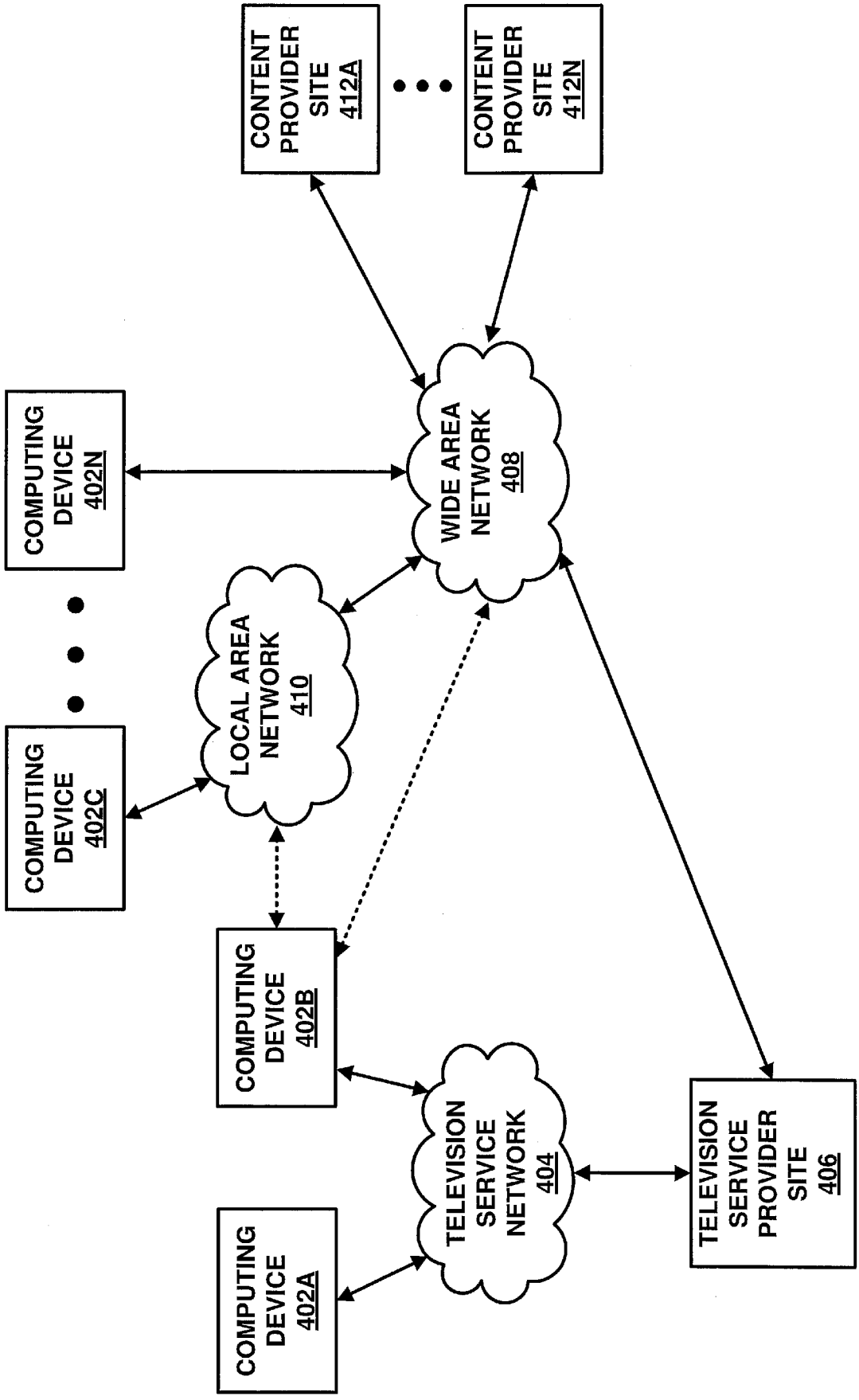
CONTENT PROVIDER SITE 412A
CONTENT PROVIDER SITE 412N
WIDE AREA NETWORK 408
COMPUTING DEVICE 402N
LOCAL AREA NETWORK 410
COMPUTING DEVICE 402C
COMPUTING DEVICE 402B
TELEVISION SERVICE NETWORK 404
TELEVISION SERVICE PROVIDER SITE 406
COMPUTING DEVICE 402A

FIG. 4

[Fig. 5]

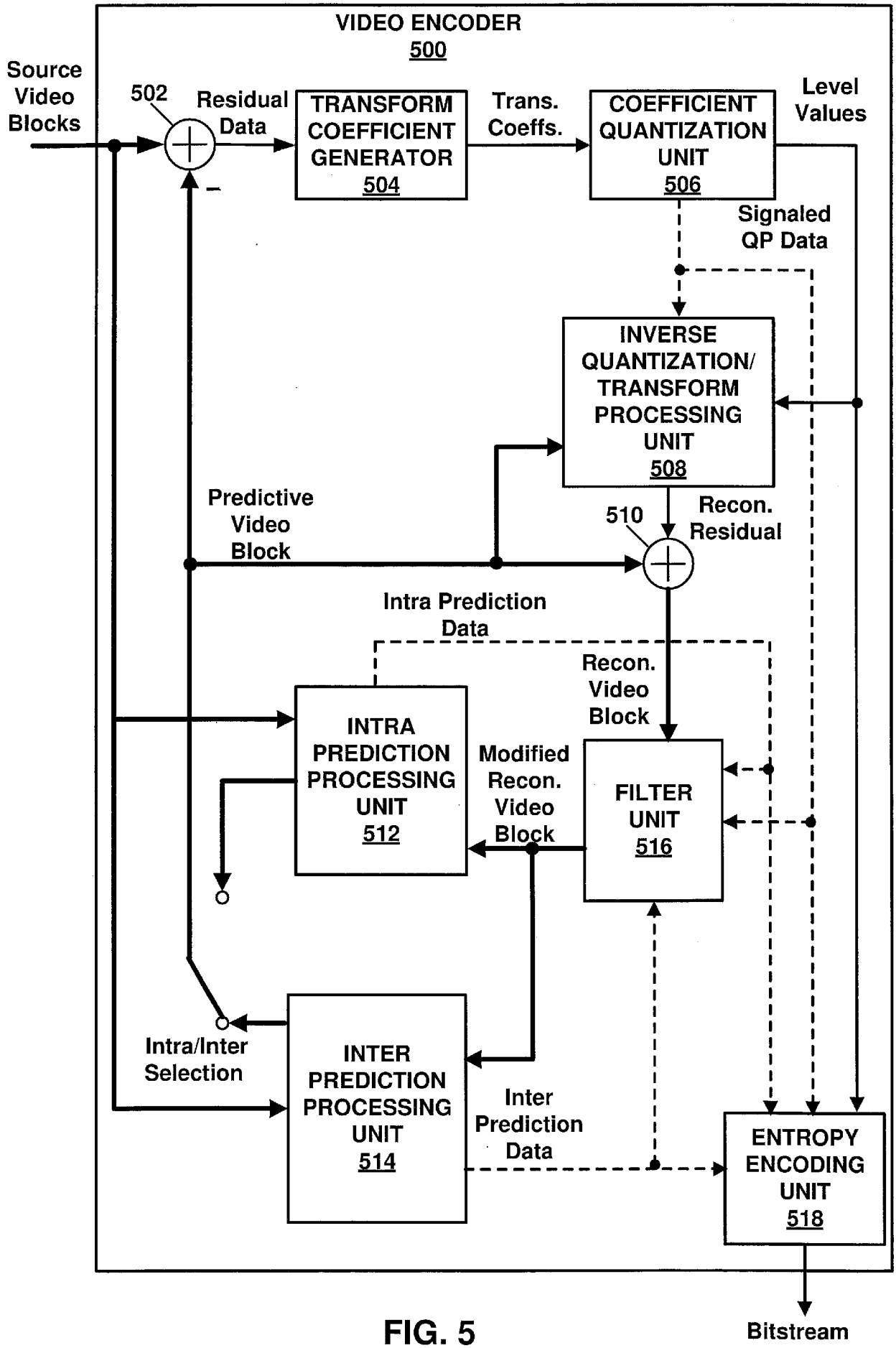
VIDEO ENCODER
500
Source Video Blocks
502
Residual Data
TRANSFORM COEFFICIENT GENERATOR
504
Trans. Coeffs.
COEFFICIENT QUANTIZATION UNIT
506
Level Values
Signaled QP Data
INVERSE QUANTIZATION/ TRANSFORM PROCESSING UNIT
508
Predictive Video Block
510
Recon. Residual
Intra Prediction Data
Recon. Video 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]

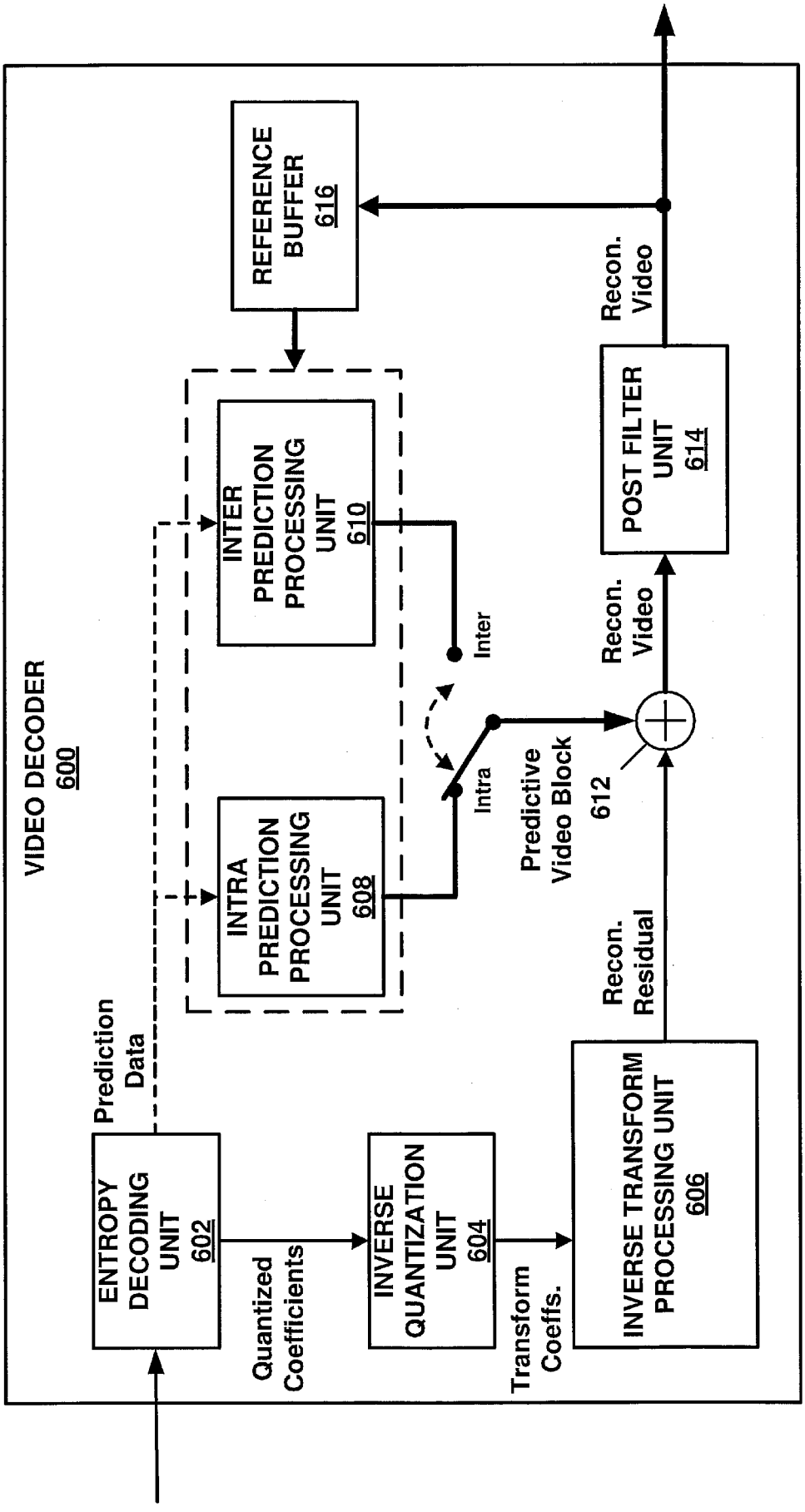
VIDEO DECODER
600
ENTROPY DECODING UNIT
602
INVERSE QUANTIZATION UNIT
604
INVERSE TRANSFORM PROCESSING UNIT
606
INTRA PREDICTION PROCESSING UNIT
608
INTER PREDICTION PROCESSING UNIT
610
REFERENCE BUFFER
616
POST FILTER UNIT
614
Prediction Data
Quantized Coefficients
Transform Coeffs.
Recon. Residual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

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2019/026364

A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl. H04N19/70(2014.01)i

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)

Int.Cl. 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 1971-2019
Registered utility model specifications of Japan 1996-2019
Published registered utility model applications of Japan 1994-2019

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.
X	"Recommendation ITU-T H.265 (10/2014)", pp.43-44,452	1-9
A	"ITU-T Recommendation H.222.0 (07/95)", pp.44-46,59-60	1-9

☐ 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
04.09.2019	17.09.2019

Name and mailing address of the ISA/JP	Authorized officer
Japan Patent Office 3-4-3, Kasumigaseki, Chiyoda-ku, Tokyo 100-8915, Japan	BANDO, Daigoro Telephone No. +81-3-3581-1101 Ext. 3541

5C 3241