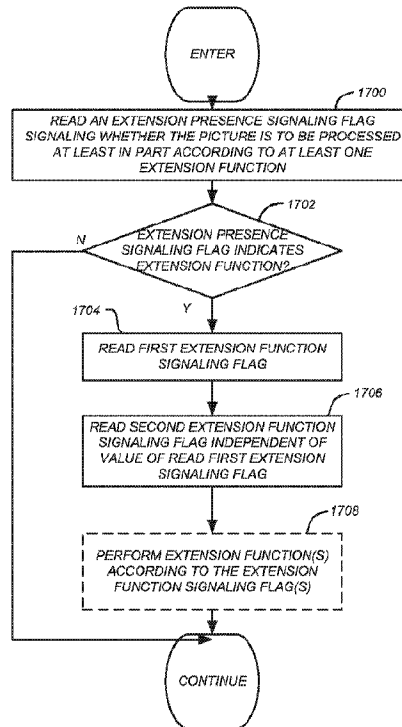




(86) Date de dépôt PCT/PCT Filing Date: 2014/12/30
 (87) Date publication PCT/PCT Publication Date: 2015/07/09
 (45) Date de délivrance/Issue Date: 2023/04/11
 (85) Entrée phase nationale/National Entry: 2016/06/28
 (86) N° demande PCT/PCT Application No.: US 2014/072695
 (87) N° publication PCT/PCT Publication No.: 2015/103221
 (30) Priorités/Priorities: 2014/01/03 (US61/923,334);
 2014/12/30 (US14/585,485)

(51) Cl.Int./Int.Cl. *H04N 19/70* (2014.01)
 (72) Inventeurs/Inventors:
 YU, YUE, US;
 WANG, LIMIN, US
 (73) Propriétaire/Owner:
 ARRIS INTERNATIONAL IP LTD, GB
 (74) Agent: GOWLING WLG (CANADA) LLP

(54) Titre : SYNTAXE D'EXTENSION ANALYSEE DE MANIERE CONDITIONNELLE POUR UN TRAITEMENT D'EXTENSION HEVC
 (54) Title: **CONDITIONALLY PARSED EXTENSION SYNTAX FOR HEVC EXTENSION PROCESSING**



(57) **Abrégé/Abstract:**

A system for signaling extension functions used in decoding a sequence including a plurality of pictures, each picture processed at least in part according to a picture parameter set is disclosed. An extension presence signaling flag is read and used to determine whether flags signaling the performance of extension functions are to be read. The flags are only read if indicated by the extension presence signaling flag.

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

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 July 2015 (09.07.2015)



(10) International Publication Number
WO 2015/103221 A9

- (51) International Patent Classification: *H04N 19/70* (2014.01)
- (21) International Application Number: PCT/US2014/072695
- (22) International Filing Date: 30 December 2014 (30.12.2014)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
61/923,334 3 January 2014 (03.01.2014) US
14/585,485 30 December 2014 (30.12.2014) US
- (71) Applicant: ARRIS ENTERPRISES, INC. [US/US]; 3871 Lakefield Drive, Suwanee, Georgia 30024 (US).
- (72) Inventors: YU, Yue; 10721 Calston Way, San Diego, California 92126 (US). WANG, Limin; 13641 Shoal Summit Drive, San Diego, California 92128 (US).
- (74) Agents: WARD, Thomas, A. et al.; 3871 Lakefield Drive, Suwanee, Georgia 30024 (US).
- (81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR,

[Continued on next page]

(54) Title: CONDITIONALLY PARSED EXTENSION SYNTAX FOR HEVC EXTENSION PROCESSING

(57) Abstract: A system for signaling extension functions used in decoding a sequence including a plurality of pictures, each picture processed at least in part according to a picture parameter set is disclosed. An extension presence signaling flag is read and used to determine whether flags signaling the performance of extension functions are to be read. The flags are only read if indicated by the extension presence signaling flag.

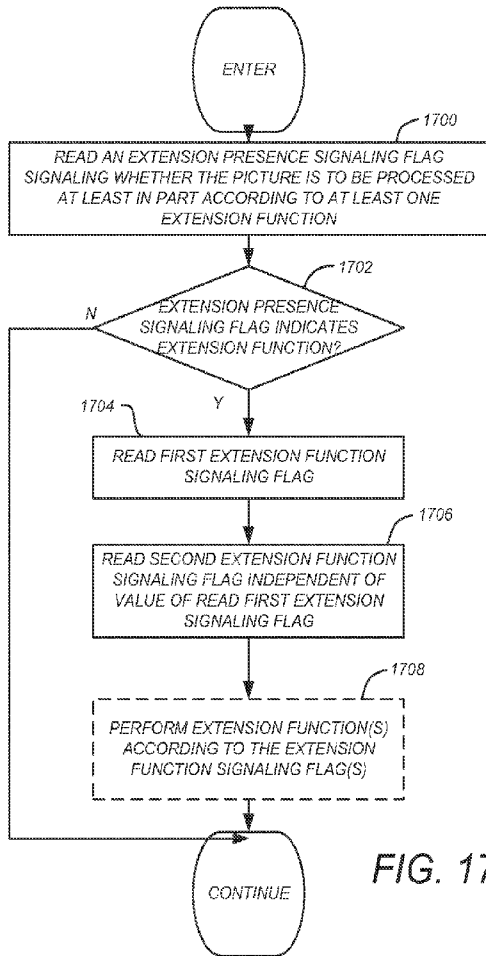


FIG. 17A

WO 2015/103221 A9

WO 2015/103221 A9

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.

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).

Published:

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

(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,

(48) Date of publication of this corrected version:

11 September 2015

(15) Information about Correction:
see Notice of 11 September 2015

CONDITIONALLY PARSED EXTENSION
SYNTAX FOR HEVC EXTENSION PROCESSING

BACKGROUND

1. Field of the Invention

[0001] The present invention relates to systems and methods for encoding and decoding data, and in particular to a system and method for generating and processing slice headers with high efficiency video coded data.

2. Description of the Related Art

[0002] There is rapid growth in the technologies associated with the generation, transmission, and reproduction of media programs. These technologies include coding schemes that permit digital versions of the media programs to be encoded to compress them to much smaller size and facilitate their transmission, storage, reception and playback. These technologies have application in personal video recorders (PVRs), video on demand (VOD), multiple channel media program offerings, interactivity, mobile telephony, and media program transmission.

[0003] Without compression, digital media programs are typically too large to transmit and/or store for a commercially acceptable cost. However, compression of such programs has made the transmission and storage of such digital media programs not only commercially feasible, but commonplace.

[0004] Initially, the transmission of media programs involved low to medium resolution images transmitted over high bandwidth transmission media such as cable television and satellite. However, such transmission has evolved to include lower bandwidth transmission media such as Internet transmission to fixed and mobile devices via computer networks, WiFi, Mobile TV and third and fourth generation (3G and 4G) networks. Further, such transmissions have also evolved to include high definition media programs such as high definition television (HDTV), which have significant transmission bandwidth and storage requirements.

[0005] The High Efficiency Video Coding (HEVC) coding standard (or H.265) is the most recent coding standard promulgated by the ISO/IEC MPEG standardization organizations. The coding standard preceding HEVC included the H.262/MPEG-2 and the subsequent

H.264/MPEG-4 Advanced Video Coding (AVC) standard. H.264/MPEG-4 has substantially replaced H.262/MPEG-2 in many application including high definition (HD) television. HEVC supports resolutions higher than HD, even in stereo or multi-view embodiments, and is more suitable for mobile devices such as tablet personal computers. Further information regarding HEVC can be found in the publication “Overview of the High Efficiency Video Coding (HEVC) Standard, by Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han and Thomas Wiegand, IEEE Transactions on Circuits and Systems for Video Technology, December 2012.

[0006] As in other coding standards, the bit stream structure and syntax of HEVC compliant data are standardized, such that every decoder conforming to the standard will produce the same output when provided with the same input. Some of the features incorporated into the HEVC standard include the definition and processing of a slice, one or more of which may together comprise one of the pictures in a video sequence. A video sequence comprises a plurality of pictures, and each picture may comprise one or more slices. Slices include non-dependent slices and dependent slices. A non-dependent slice (hereinafter simply referred to as a slice) is a data structure that can be decoded independently from other slices of the same picture in terms of entropy encoding, signal prediction, and residual signal construction. This data structure permits resynchronization of events in case of data losses. A “dependent slice” is a structure that permits information about the slice (such as those related with tiles within the slice or wavefront entries) to be carried to the network layer, thus making that data available to a system to more quickly process fragmented slices. Dependent slices are mostly useful for low-delay encoding.

[0007] HEVC and legacy coding standards define a parameter set structure that offers improved flexibility for operation over a wide variety of applications and network environments, and improved robustness to data losses. Parameter sets contain information that can be shared for decoding of different portions of the encoded video. The parameter set structure provides a secure mechanism for conveying data that is essential to the decoding process. H.264 defined both sequence parameter sets (SPS) that describe parameters for decoding a sequence of pictures and a picture parameter set (PPS) that describes parameters for decoding a picture of the sequence of pictures. HEVC introduces a new parameter set, the video parameter set (VPS).

[0008] The encoding and decoding of slices is performed according to information included in a slice header. The slice header includes syntax and logic for reading flags and data that are used in decoding the slice.

[0009] Like its predecessors, HEVC supports both temporal and spatial encoding of picture slices. HEVC defines slices to include I-slices, which are spatially, but not temporally encoded with reference to another slice. I-slices are alternatively described as “intra” slice encoded. HEVC also defines slices to include P (predictive) slices, which are spatially encoded and temporally encoded with reference to another slice. P-slices are alternatively described as “inter” slice encoded. HEVC also describes slices to include bi-predictive (B)-slices. B-slices are spatially encoded and temporally encoded with reference to two or more other slices. Further, HEVC consolidates the notion of P and B slices into general B slices that can be used as reference slice.

[0010] Currently, the HEVC syntax includes provision for extensions to expand the capabilities or capacities of HEVC beyond the baseline. Such extensions include range extensions (RExt), scalability extensions (SHVC), and multi-view extensions (MV-HEVC). Extensions may be signaled in the VPS, SPS, PPS, or combination thereof.

[0011] High Efficiency Video Coding (HEVC) Range Extensions text specification: Draft 4,” published by the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 13th Meeting: Incheon, KR, 18–26 April 2013, by David Flynn et al, defines a PPS syntax that controlled the execution of multiple extension functions by use of extension function-unique flags uniquely associated with each extension function. However, such flags were not independently read. For example, a first flag signaling the execution of one extension function in the PPS syntax may be read within syntax that is only parsed and executed if another (second) flag for a previously executed extension function has a particular state or value (e.g. a flag may not read unless the previously read flag tests “true”). This is not problematic when the execution of an extension function is not desired unless the previous extension function syntax has been executed. But it is problematic in cases where it is desirable to control the parsing or execution of the extension functions independently. What is needed is an improved system and method for parsing syntax that permits the parsing of extension functions to be independently controlled. This disclosure describes such a system and method.

SUMMARY

[0001] To address the requirements described above, this document discloses a device and method for signaling extension functions used in decoding a sequence comprising a plurality of pictures, each picture processed at least in part according to a picture parameter set. In one embodiment, the method comprises reading an extension presence signaling flag, determining if the read extension presence signaling flag indicates that the picture is to be processed at least in part according to at least one extension function, and reading a first extension function signaling flag signaling a first extension function and reading a second extension function signaling flag signaling a second extension independently of a value of the read first extension function signaling flag only if the read extension presence signaling flag indicates that the picture is to be processed at least in part according to the at least one extension function. This method can be performed with additional extension function signaling flags as necessary. Another embodiment is disclosed in which an apparatus is evidenced by a processor having a communicatively coupled memory storing instructions for performing the foregoing operations.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

[0003] FIG. 1 is a diagram depicting an exemplary embodiment of a video coding-decoding system that can be used for transmission and/or storage and retrieval of audio and/or video information;

[0004] FIG. 2A is a diagram of one embodiment of a codec system in which the encoded AV information is transmitted to and received at another location;

[0005] FIG. 2B is a diagram depicting an exemplary embodiment of codec system in which the encoded information is stored and later retrieved for presentation, hereinafter referred to as codec storage system;

[0006] FIG. 2C is a diagram depicting another embodiment for a content distribution system comprising a coding system or encoder and decoding system or decoder for transmitting and receiving HEVC data;

[0007] FIG. 3 is a block diagram illustrating one embodiment of the source encoder;

[0008] FIG. 4 is a diagram depicting a picture of AV information, such as one of the pictures in the picture sequence;

[0009] FIG. 5 is a diagram showing an exemplary partition of a coding tree block into coding units;

[0020] FIG. 6 is a diagram illustrating a representation of a representative quad tree and data parameters for the code tree block partitioning shown in FIG. 5;

[0021] FIG. 7 is a diagram illustrating the partition of a coding unit into one or more prediction units;

[0022] FIG. 8 is a diagram showing a coding unit partitioned into four prediction units and an associated set of transform units;

[0023] FIG. 9 is a diagram showing RQT codetree for the transform units associated with the coding unit in the example of FIG 8;

[0024] FIG. 10 is a diagram illustrating spatial prediction of prediction units;

[0025] FIG. 11 is a diagram illustrating temporal prediction;

[0026] FIG. 12 is a diagram illustrating the use of motion vector predictors (MVPs);

[0027] FIG. 13 illustrates an example of the use of the reference picture lists;

[0028] FIG. 14 is a diagram illustrating processes performed by the encoder according to the aforementioned standard;

[0029] FIG. 15 depicts the use of a collocated_from_10_flag by the decoder in decoding, according to the emerging HEVC standard;

[0030] FIGs. 16A and 16B are diagrams presenting a baseline PPS syntax;

[0031] FIGs. 16C and 16D are diagrams presenting an improved PPS syntax;

[0032] FIGs. 17A-17D illustrate an exemplary improved processing flows and syntax for extension processing;

[0033] FIG. 18 is a diagram presenting an exemplary PPS syntax for HEVC range extension;

[0034] FIGs. 19A-19C show further alternate embodiments of extension signaling syntax; and

[0035] FIG. 20 illustrates an exemplary processing system that could be used to implement the disclosed embodiments.

DETAILED DESCRIPTION

[0036] In the following description, reference is made to the accompanying drawings which form a part hereof, and which is shown, by way of illustration, several embodiments of the present invention. It is understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention.

Audio-Visual Information Transcription and Storage

[0037] FIG. 1 is a diagram depicting an exemplary embodiment of a video coding-decoding (codec) system 100 that can be used for transmission and/or storage and retrieval of audio and/or video information. The codec system 100 comprises an encoding system 104, which accepts audio-visual (AV) information 102 and processes the AV information 102 to generate encoded (compressed) AV information 106, and a decoding system 112, which processes the encoded AV information 106 to produce recovered AV information 114. Since the encoding and decoding processes are not lossless, the recovered AV information 114 is not identical to the initial AV information 102, but with judicious selection of the encoding processes and parameters, the differences between the recovered AV information 114 and the unprocessed AV information 102 are acceptable to human perception.

[0038] The encoded AV information 106 is typically transmitted or stored and retrieved before decoding and presentation, as performed by transception (transmission and reception) or storage/retrieval system 108. Transception losses may be significant, but storage/retrieval losses are typically minimal or non-existent, hence, the transcepted AV information 110 provided to the decoding system 112 is typically the same as or substantially the same as the encoded AV information 106.

[0039] FIG. 2A is a diagram of one embodiment of a codec system 200A in which the encoded AV information 106 is transmitted to and received at another location. A transmission segment 230 converts an input AV information 102 into a signal appropriate for transmission and transmits the converted signal over the transmission channel 212 to the reception segment 232. The reception segment 232 receives the transmitted signal, and converts the received signal into the recovered AV information 114 for presentation. As described above, due to coding and transmission losses and errors, the recovered AV information 114 may be of lower quality than the AV information 102 that was provided to the transmission segment 230. However, error-correcting systems may be included to reduce or eliminate such errors. For example, the encoded AV information 106 may be forward error correction (FEC) encoded by adding redundant information, and such redundant information can be used to identify and eliminate errors in the reception segment 230.

[0040] The transmission segment 102 comprises one or more source encoders 202 to encode multiple sources of AV information 102. The source encoder 202 encodes the AV information

102 primarily for purposes of compression to produce the encoded AV information 106, and may include, for example a processor and related memory storing instructions implementing a codec such as MPEG-1, MPEG-2, MPEG-4 AVC/H.264, HEVC or similar codec, as described further below.

[0041] The codec system 200A may also include optional elements indicated by the dashed lines in FIG. 2A. These optional elements include a video multiplex encoder 204, an encoding controller 208, and a video demultiplexing decoder 218. The optional video multiplex encoder 204 multiplexes encoded AV information 106 from an associated plurality of source encoder(s) 202 according to one or more parameters supplied by the optional encoding controller 208. Such multiplexing is typically accomplished in the time domain and is data packet based.

[0042] In one embodiment, the video multiplex encoder 204 comprises a statistical multiplexer, which combines the encoded AV information 106 from a plurality of source encoders 202 so as to minimize the bandwidth required for transmission. This is possible, since the instantaneous bit rate of the coded AV information 106 from each source encoder 202 can vary greatly with time according to the content of the AV information 102. For example, scenes having a great deal of detail and motion (e.g. sporting events) are typically encoded at higher bitrates than scenes with little motion or detail (e.g. portrait dialog). Since each source encoder 202 may produce information with a high instantaneous bitrate while another source encoder 202 produces information with a low instantaneous bit rate, and since the encoding controller 208 can command the source encoders 202 to encode the AV information 106 according to certain performance parameters that affect the instantaneous bit rate, the signals from each of the source encoders 106 (each having a temporally varying instantaneous bit rate) can be combined together in an optimal way to minimize the instantaneous bit rate of the multiplexed stream 205.

[0043] As described above, the source encoder 202 and the video multiplex coder 204 may optionally be controlled by a coding controller 208 to minimize the instantaneous bit rate of the combined video signal. In one embodiment, this is accomplished using information from a transmission buffer 206 which temporarily stores the coded video signal and can indicate the fullness of the buffer 206. This allows the coding performed at the source encoder 202 or video multiplex coder 204 to be a function of the storage remaining in the transmission buffer 206.

[0044] The transmission segment 230 also may comprise a transmission encoder 210, which further encodes the video signal for transmission to the reception segment 232. Transmission

encoding may include for example, the aforementioned FEC coding and/or coding into a multiplexing scheme for the transmission medium of choice. For example, if the transmission is by satellite or terrestrial transmitters, the transmission encoder 114 may encode the signal into a signal constellation before transmission via quadrature amplitude modulation (QAM) or similar modulation technique. Also, if the encoded video signal is to be streamed via an Internet protocol device and the Internet, the transmission encodes the signal according to the appropriate protocol. Further, if the encoded signal is to be transmitted via mobile telephony, the appropriate coding protocol is used, as further described below.

[0045] The reception segment 232 comprises a transmission decoder 214 to receive the signal that was coded by the transmission coder 210 using a decoding scheme complementary to the coding scheme used in the transmission encoder 214. The decoded received signal may be temporarily stored by optional reception buffer 216, and if the received signal comprises multiple video signals, the received signal is multiplex decoded by video multiplex decoder 218 to extract the video signal of interest from the video signals multiplexed by the video multiplex coder 204. Finally, the video signal of interest is decoded by source decoder 220 using a decoding scheme or codec complementary to the codec used by the source encoder 202 to encode the AV information 102.

[0046] In one embodiment, the transmitted data comprises a packetized video stream transmitted from a server (representing the transmitting segment 230) to a client (representing the receiving segment 232). In this case, the transmission encoder 210 may packetize the data and embed network abstract layer (NAL) units in network packets. NAL units define a data container that has header and coded elements, and may correspond to a video frame or other slice of video data.

[0047] The compressed data to be transmitted may be packetized and transmitted via transmission channel 212, which may include a Wide Area Network (WAN) or a Local Area Network (LAN). Such a network may comprise, for example, a wireless network such as WiFi, an Ethernet network, an Internet network or a mixed network composed of several different networks. Such communication may be affected via a communication protocol, for example Real-time Transport Protocol (RTP), User Datagram Protocol (UDP) or any other type of communication protocol. Different packetization methods may be used for each network abstract layer (NAL) unit of the bit stream. In one case, one NAL unit size is smaller than the maximum transport

unit (MTU) size corresponding to the largest packet size that can be transmitted over the network without being fragmented. In this case, the NAL unit is embedded into a single network packet. In another case, multiple entire NAL units are included in a single network packet. In a third case, one NAL unit may be too large to be transmitted in a single network packet and is thus split into several fragmented NAL units with each fragmented NAL unit being transmitted in an individual network packet. Fragmented NAL units are typically sent consecutively for decoding purposes.

[0048] The reception segment 232 receives the packetized data and reconstitutes the NAL units from the network packet. For fragmented NAL units, the client concatenates the data from the fragmented NAL units in order to reconstruct the original NAL unit. The client 232 decodes the received and reconstructed data stream and reproduces the video images on a display device and the audio data by a loud speaker.

[0049] FIG. 2B is a diagram depicting an exemplary embodiment of a codec system in which the encoded information is stored and later retrieved for presentation, hereinafter referred to as a codec storage system 200B. This embodiment may be used, for example, to locally store information in a digital video recorder (DVR), a flash drive, hard drive, or similar device. In this embodiment, the AV information 102 is source encoded by source encoder 202, optionally buffered by storage buffer 234 before storage in a storage device 236. The storage device 236 may store the video signal temporarily or for an extended period of time, and may comprise a hard drive, flash drive, RAM or ROM. The stored AV information is then retrieved, optionally buffered by retrieve buffer 238 and decoded by the source decoder 220.

[0050] FIG. 2C is another diagram depicting an exemplary content distribution system 200C comprising a coding system or encoder 202 and a decoding system or decoder 220 that can be used to transmit and receive HEVC data. In some embodiments, the coding system 202 can comprise an input interface 256, a controller 241, a counter 242, a frame memory 243, an encoding unit 244, a transmitter buffer 267 and an output interface 257. The decoding system 220 can comprise a receiver buffer 259, a decoding unit 260, a frame memory 261 and a controller 267. The coding system 202 and the decoding system 220 can be coupled with each other via a transmission path which can carry a compressed bit stream. The controller 241 of the coding system 202 can control the amount of data to be transmitted on the basis of the capacity of the transmitter buffer 267 or receiver buffer 259 and can include other parameters such as the

amount of data per a unit of time. The controller 241 can control the encoding unit 244 to prevent the occurrence of a failure of a received signal decoding operation of the decoding system 220. The controller 241 can be a processor or include, by way of a non-limiting example, a microcomputer having a processor, a random access memory and a read only memory.

[0051] Source pictures 246 supplied from, by way of a non-limiting example, a content provider can include a video sequence of frames including source pictures in a video sequence. The source pictures 246 can be uncompressed or compressed. If the source pictures 246 are uncompressed, the coding system 202 can have an encoding function. If the source pictures 246 are compressed, the coding system 202 can have a transcoding function. Coding units can be derived from the source pictures utilizing the controller 241. The frame memory 243 can have a first area that can be used for storing the incoming frames from the source pictures 246 and a second area that can be used for reading out the frames and outputting them to the encoding unit 244. The controller 241 can output an area switching control signal 249 to the frame memory 243. The area switching control signal 249 can indicate whether the first area or the second area is to be utilized.

[0052] The controller 241 can output an encoding control signal 250 to the encoding unit 244. The encoding control signal 250 can cause the encoding unit 202 to start an encoding operation, such as preparing the Coding Units based on a source picture. In response to the encoding control signal 250 from the controller 241, the encoding unit 244 can begin to read out the prepared Coding Units to a high-efficiency encoding process, such as a prediction coding process or a transform coding process which process the prepared Coding Units generating video compression data based on the source pictures associated with the Coding Units.

[0053] The encoding unit 244 can package the generated video compression data in a packetized elementary stream (PES) including video packets. The encoding unit 244 can map the video packets into an encoded video signal 248 using control information and a program time stamp (PTS) and the encoded video signal 248 can be transmitted to the transmitter buffer 267.

[0054] The encoded video signal 248, including the generated video compression data, can be stored in the transmitter buffer 267. The information amount counter 242 can be incremented to indicate the total amount of data in the transmitter buffer 267. As data is retrieved and removed from the buffer, the counter 242 can be decremented to reflect the amount of data in

the transmitter buffer 267. The occupied area information signal 253 can be transmitted to the counter 242 to indicate whether data from the encoding unit 244 has been added or removed from the transmitter buffer 267 so the counter 242 can be incremented or decremented. The controller 241 can control the production of video packets produced by the encoding unit 244 on the basis of the occupied area information 253 which can be communicated in order to anticipate, avoid, prevent, and/or detect an overflow or underflow from taking place in the transmitter buffer 267.

[0055] The information amount counter 242 can be reset in response to a preset signal 254 generated and output by the controller 241. After the information amount counter 242 is reset, it can count data output by the encoding unit 244 and obtain the amount of video compression data and/or video packets, which have been generated. The information amount counter 242 can supply the controller 241 with an information amount signal 255 representative of the obtained amount of information. The controller 241 can control the encoding unit 244 so that there is no overflow at the transmitter buffer 267.

[0056] In some embodiments, the decoding system 220 can comprise an input interface 266, a receiver buffer 259, a controller 267, a frame memory 261, a decoding unit 260 and an output interface 267. The receiver buffer 259 of the decoding system 220 can temporarily store the compressed bit stream, including the received video compression data and video packets based on the source pictures from the source pictures 246. The decoding system 220 can read the control information and presentation time stamp information associated with video packets in the received data and output a frame number signal 263 which can be applied to the controller 220. The controller 267 can supervise the counted number of frames at a predetermined interval. By way of a non-limiting example, the controller 267 can supervise the counted number of frames each time the decoding unit 260 completes a decoding operation.

[0057] In some embodiments, when the frame number signal 263 indicates the receiver buffer 259 is at a predetermined capacity, the controller 267 can output a decoding start signal 264 to the decoding unit 260. When the frame number signal 263 indicates the receiver buffer 259 is at less than a predetermined capacity, the controller 267 can wait for the occurrence of a situation in which the counted number of frames becomes equal to the predetermined amount. The controller 267 can output the decoding start signal 263 when the situation occurs. By way of a non-limiting example, the controller 267 can output the decoding start signal 264 when the

frame number signal 263 indicates the receiver buffer 259 is at the predetermined capacity. The encoded video packets and video compression data can be decoded in a monotonic order (i.e., increasing or decreasing) based on presentation time stamps associated with the encoded video packets.

[0058] In response to the decoding start signal 264, the decoding unit 260 can decode data amounting to one picture associated with a frame and compressed video data associated with the picture associated with video packets from the receiver buffer 259. The decoding unit 260 can write a decoded video signal 269 into the frame memory 261. The frame memory 261 can have a first area into which the decoded video signal is written, and a second area used for reading out decoded pictures 262 to the output interface 267.

[0059] In various embodiments, the coding system 202 can be incorporated or otherwise associated with a transcoder or an encoding apparatus at a headend and the decoding system 220 can be incorporated or otherwise associated with a downstream device, such as a mobile device, a set top box or a transcoder.

Source Encoding/Decoding

[0060] As described above, the encoders 202 employ compression algorithms to generate bit streams and/or files of smaller size than the original video sequences in the AV information 102. Such compression is made possible by reducing spatial and temporal redundancies in the original sequences.

[0061] Prior art encoders 202 include those compliant with the video compression standard H.264/MPEG-4 AVC (“Advanced Video Coding”) developed by between the “Video Coding Expert Group” (VCEG) of the ITU and the “Moving Picture Experts Group” (MPEG) of the ISO, in particular in the form of the publication “Advanced Video Coding for Generic Audiovisual Services” (March 2005).

[0062] HEVC “High Efficiency Video Coding” (sometimes known as H.265) is expected to replace the H.264/MPEG-4 AVC. HEVC introduces new coding tools and entities that are generalizations of the coding entities defined in H.264/AVC, as further described below.

[0063] FIG. 3 is a block diagram illustrating one embodiment of the source encoder 202. The source encoder 202 accepts AV information 102 and uses sampler 302 sample the AV information 102 to produce a sequence 303 of successive of digital images or pictures, each

having a plurality of pixels. A picture can comprise a frame or a field, wherein a frame is a complete image captured during a known time interval, and a field is the set of odd-numbered or even-numbered scanning lines composing a partial image.

[0064] The sampler 302 produces an uncompressed picture sequence 303. Each digital picture can be represented by one or more matrices having a plurality of coefficients that represent information about the pixels that together comprise the picture. The value of a pixel can correspond to luminance or other information. In the case where several components are associated with each pixel (for example red-green-blue components or luminance-chrominance components), each of these components may be separately processed.

[0065] Images can be segmented into “slices,” which may comprise a portion of the picture or may comprise the entire picture. In the H.264 standard, these slices are divided into coding entities called macroblocks (generally blocks of size 16 pixels×16 pixels) and each macroblock may in turn be divided into different sizes of data blocks 102, for example 4×4, 4×8, 8×4, 8×8, 8×16, 16×8. HEVC expands and generalizes the notion of the coding entity beyond that of the macroblock.

HEVC Coding Entities: CTU, CU, PU and TU

[0066] Like other video coding standards, HEVC is a block-based hybrid spatial and temporal predictive coding scheme. However, HEVC introduces new coding entities that are not included with H.264/AVC standard. These coding entities include (1) Coding tree block (CTUs), coding units (CUs), the predictive units (PUs) and transform units (TUs) and are further described below.

[0067] FIG. 4 is a diagram depicting a picture 400 of AV information 102, such as one of the pictures in the picture sequence 303. The picture 400 is spatially divided into non-overlapping square blocks known as coding tree units(s), or CTUs 402. Unlike H.264 and previous video coding standards where the basic coding unit is macroblock of 16x16 pixels, the CTU 402 is the basic coding unit of HEVC, and can be as large as 128x128 pixels. As shown in FIG. 4, the CTUs 402 are typically referenced within the picture 400 in an order analogous to a progressive scan.

[0068] Each CTU 402 may in turn be iteratively divided into smaller variable size coding units described by a “quad tree” decomposition further described below. Coding units are regions

formed in the image to which similar encoding parameters are applied and transmitted in the bit stream 314.

[0069] FIG. 5 is a diagram showing an exemplary partition of an CTU 402 into coding units (CUs) such as coding unit 502A and 502B (hereinafter alternatively referred to as coding unit(s) 502). A single CTU 402 can be divided into four CUs 502 such as CU 502A, each a quarter of the size of CTU 402. Each such divided CU 502A can be further divided into four smaller CUs 502B of quarter size of initial CU 502A.

[0070] The division of CTUs 402 into CUs 502A and into smaller CUs 502B is described by “quad tree” data parameters (e.g. flags or bits) that are encoded into the output bit stream 314 along with the encoded data as overhead known as syntax.

[0071] FIG. 6 is a diagram illustrating a representation of a representative quad tree 600 and data parameters for the CTU 402 partitioning shown in FIG. 5. The quad tree 600 comprises a plurality of nodes including first node 602A at one hierarchical level and second node 602B at a lower hierarchical level (hereinafter, quad tree nodes may be alternatively referred to as “nodes” 602). At each node 602 of a quad tree, a “split flag” or bit “1” is assigned if the node 602 is further split into sub-nodes, otherwise a bit “0” is assigned.

[0072] For example, the CTU 402 partition illustrated in FIG. 5 can be represented by the quad tree 600 presented in FIG. 6, which includes a split flag of “1” associated with node 602A at the top CU 502 level (indicating there are 4 additional nodes at a lower hierarchical level). The illustrated quad tree 600 also includes a split flag of “1” associated with node 602B at the mid CU 502 level to indicate that this CU is also partitioned into four further CUs 502 at the next (bottom) CU level. The source encoder 202 may restrict the minimum and maximum CU 502 sizes, thus changing the maximum possible depth of the CU 502 splitting.

[0073] The encoder 202 generates encoded AV information 106 in the form of a bit stream 314 that includes a first portion having encoded data for the CUs 502 and a second portion that includes overhead known as syntax elements. The encoded data includes data corresponding to the encoded CUs 502 (i.e. the encoded residuals together with their associated motion vectors, predictors, or related residuals as described further below). The second portion includes syntax elements that may represent encoding parameters which do not directly correspond to the encoded data of the blocks. For example, the syntax elements may comprise an address and

identification of the CU 502 in the image, a quantization parameter, an indication of the elected Inter/Intra coding mode, the quad tree 600 or other information.

[0074] CUs 502 correspond to elementary coding elements and include two related sub-units: prediction units (PUs) and a transform units (TUs), both of which have a maximum size equal to the size of the corresponding CU 502.

[0075] FIG. 7 is a diagram illustrating the partition of a CU 502 into one or more PUs 702. A PU 702 corresponds to a partitioned CU 502 and is used to predict pixels values for intra-picture or inter-picture types. PUs 702 are an extension of the partitioning of H.264/AVC for motion estimation, and are defined for each CU 502 that is not further subdivided into other CUs (“split flag”=0). At each leaf 604 of the quad tree 600, a final (bottom level) CU 502 of $2N \times 2N$ can possess one of four possible patterns of PUs: $2N \times 2N$ (702A), $2N \times N$ (702B), $N \times 2N$ (702C) and $N \times N$ (702D), as shown in FIG. 7.

[0076] A CU 502 can be either spatially or temporally predictive coded. If a CU 502 is coded in “intra” mode, each PU 702 of the CU 502 can have its own spatial prediction direction and image information as further described below. Also, in the “intra” mode, the PU 702 of the CU 502 may depend on another CU 502 because it may use a spatial neighbor, which is in another CU. If a CU 502 is coded in “inter” mode, each PU 702 of the CU 502 can have its own motion vector(s) and associated reference picture(s) as further described below.

[0077] FIG. 8 is a diagram showing a CU 502 partitioned into four PUs 702 and an associated set of transform units (TUs) 802. TUs 802 are used to represent the elementary units that are spatially transformed by a DCT (Discrete Cosine Transform). The size and location of each block transform TU 802 within a CU 502 is described by a “residual” quad tree (RQT) further illustrated below.

[0078] FIG. 9 is a diagram showing RQT 900 for TUs 802 for the CU 502 in the example of FIG. 8. Note that the “1” at the first node 902A of the RQT 900 indicates that there are four branches and that the “1” at the second node 902B at the adjacent lower hierarchical level indicates that the indicated node further has four branches. The data describing the RQT 900 is also coded and transmitted as an overhead in the bit stream 314.

[0079] The coding parameters of a video sequence may be stored in dedicated NAL units called parameter sets. Two types of parameter sets NAL units may be employed. The first parameter set type is known as a Sequence Parameter Set (SPS), and comprises a NAL unit that includes

parameters that are unchanged during the entire video sequence. Typically, an SPS handles the coding profile, the size of the video frames and other parameters. The second type of parameter set is known as a Picture Parameter Set (PPS), and codes different values that may change from one image to another.

Spatial and Temporal Prediction

[0080] One of the techniques used to compress a bit stream 314 is to forego the storage of pixel values themselves and instead, predict the pixel values using a process that can be repeated at the decoder 220 and store or transmit the difference between the predicted pixel values and the actual pixel values (known as the residual). So long as the decoder 220 can compute the same predicted pixel values from the information provided, the actual picture values can be recovered by adding the residuals to the predicted values. The same technique can be used to compress other data as well.

[0081] Referring back to FIG. 3, each PU 702 of the CU 502 being processed is provided to a predictor module 307. The predictor module 307 predicts the values of the PUs 702 based on information in nearby PUs 702 in the same frame (intra-frame prediction, which is performed by the spatial predictor 324) and information of PUs 702 in temporally proximate frames (inter-frame prediction, which is performed by the temporal predictor 330). Temporal prediction, however, may not always be based on a collocated PU, since collocated PUs are defined to be located at a reference/non-reference frame having the same x and y coordinates as the current PU 702. These techniques take advantage of spatial and temporal dependencies between PUs 702.

[0082] Encoded units can therefore be categorized to include two types: (1) non-temporally predicted units and (2) temporally predicted units. Non-temporally predicted units are predicted using the current frame, including adjacent or nearby PUs 702 within the frame (e.g. intra-frame prediction), and are generated by the spatial predictor 324. Temporally predicted units are predicted from one temporal picture (e.g. P-frames) or predicted from at least two reference pictures temporally ahead and/or behind (i.e. B-frames).

Spatial Prediction

[0083] FIG. 10 is a diagram illustrating spatial prediction of PUs 702. A picture may comprise a PU 702 and spatially proximate other PUs 1-4, including nearby PU 702N. The spatial predictor 324 predicts the current block (e.g. block C of FIG. 10) by means of an “intra-frame” prediction which uses PUs 702 of already-encoded other blocks of pixels of the current image.

[0084] The spatial predictor 324 locates a nearby PU (e.g. PU 1, 2, 3 or 4 of FIG. 10) that is appropriate for spatial coding and determines an angular prediction direction to that nearby PU. In HEVC, 35 directions can be considered, so each PU may have one of 35 directions associated with it, including horizontal, vertical, 45 degree diagonal, 135 degree diagonal, DC etc. The spatial prediction direction of the PU is indicated in the syntax.

[0085] Referring back to the spatial predictor 324 of FIG. 3, this located nearby PU is used to compute a residual PU 704 (e) as the difference between the pixels of the nearby PU 702N and the current PU 702, using element 305. The result is an intra-predicted PU element 1006 that comprises a prediction direction 1002 and the intra-predicted residual PU 1004. The prediction direction 1002 may be coded by inferring the direction from spatially proximate PUs, and the spatial dependencies of the picture, enabling the coding rate of the intra prediction direction mode to be reduced.

Temporal Prediction

[0086] FIG. 11 is a diagram illustrating temporal prediction. Temporal prediction considers information from temporally neighboring pictures or frames, such as the previous picture, picture $i-1$.

[0087] Generally, temporal prediction includes single-prediction (P-type), which predicts the PU 702 by referring to one reference area from only one reference picture, and multiple prediction (B-type), which predicts the PU by referring to two reference areas from one or two reference pictures. Reference images are images in the video sequence that have already been coded and then reconstructed (by decoding).

[0088] The temporal predictor 330 identifies, in one or several of these reference areas (one for P-type or several for B-type), areas of pixels in a temporally nearby frame so that they can be used as predictors of this current PU 702. In the case where several areas predictors are used (B-type), they may be merged to generate one single prediction. The reference area 1102 is identified in the reference frame by a motion vector (MV) 1104 that is defines the displacement

between the current PU 702 in current frame (picture i) and the reference area 1102 (refIdx) in the reference frame (picture i-1). A PU in a B-picture may have up to two MVs. Both MV and refIdx information are included in the syntax of the HEVC bit stream.

[0089] Referring again to FIG. 3, a difference between the pixel values between of the reference area 1102 and the current PU 702 may be computed by element 305 as selected by switch 306. This difference is referred to as the residual of the inter-predicted PU 1106. At the end of the temporal or inter-frame prediction process, the current PU 1006 is composed of one motion vector MV 1104 and a residual 1106.

[0090] However, as described above, one technique for compressing data is to generate predicted values for the data using means repeatable by the decoder 220, computing the difference between the predicted and actual values of the data (the residual) and transmitting the residual for decoding. So long as the decoder 220 can reproduce the predicted values, the residual values can be used to determine the actual values.

[0091] This technique can be applied to the MVs 1104 used in temporal prediction by generating a prediction of the MV 1104, computing a difference between the actual MV 1104 and the predicted MV 1104 (a residual) and transmitting the MV residual in the bit stream 314. So long as the decoder 220 can reproduce the predicted MV 1104, the actual MV 1104 can be computed from the residual. HEVC computes a predicted MV for each PU 702 using the spatial correlation of movement between nearby PUs 702.

[0092] FIG. 12 is a diagram illustrating the use of motion vector predictors (MVPs) in HEVC. Motion vector predictors V_1 , V_2 and V_3 are taken from the MVs 1104 of a plurality of blocks 1, 2, and 3 situated nearby or adjacent the block to encode (C). As these vectors refer to motion vectors of spatially neighboring blocks within the same temporal frame and can be used to predict the motion vector of the block to encode, these vectors are known as spatial motion predictors.

[0093] FIG. 12 also illustrates temporal motion vector predictor V_T which is the motion vector of the co-located block C' in a previously decoded picture (in decoding order) of the sequence (e. g. block of picture i-1 located at the same spatial position as the block being coded (block C of image i)).

[0094] The components of the spatial motion vector predictors V_1 , V_2 and V_3 and the temporal motion vector predictor V_T can be used to generate a median motion vector predictor V_M . In

HEVC, the three spatial motion vector predictors may be taken as shown in FIG. 12, that is, from the block situated to the left of the block to encode (V_1), the block situated above (V_3) and from one of the blocks situated at the respective corners of the block to encode (V_2), according to a predetermined rule of availability. This MV predictor selection technique is known as Advanced Motion Vector Prediction (AMVP).

[0095] A plurality of (typically five) MV predictor (MVP) candidates having spatial predictors (e.g. V_1 , V_2 and V_3) and temporal predictor(s) V_T is therefore obtained. In order to reduce the overhead of signaling the motion vector predictor in the bit stream, the set of motion vector predictors may be reduced by eliminating data for duplicated motion vectors (for example, MVs which have the same value as other MVs may be eliminated from the candidates).

[0096] The encoder 202 may select a “best” motion vector predictor from among the candidates, and compute a motion vector predictor residual as a difference between the selected motion vector predictor and the actual motion vector, and transmit the motion vector predictor residual in the bit stream 314. To perform this operation, the actual motion vector must be stored for later use by the decoder 220 (although it is not transmitted in the bit stream 314). Signaling bits or flags are included in the bit stream 314 to specify which MV residual was computed from the normalized motion vector predictor, and are later used by the decoder to recover the motion vector. These bits or flags are further described below.

[0097] Referring back to FIG. 3, the intra-predicted residuals 1004 and the inter-predicted residuals 1106 obtained from the spatial (intra) or temporal (inter) prediction process are then transformed by transform module 308 into the transform units (TUs) 802 described above. A TU 802 can be further split into smaller TUs using the RQT decomposition described above with respect to FIG. 9. In HEVC, generally 2 or 3 levels of decompositions are used and authorized transform sizes are from 32×32 , 16×16 , 8×8 and 4×4 . As described above, the transform is derived according to a discrete cosine transform (DCT) or discrete sine transform (DST).

[0098] The residual transformed coefficients are then quantized by quantizer 310. Quantization plays a very important role in data compression. In HEVC, quantization converts the high precision transform coefficients into a finite number of possible values. Although the quantization permits a great deal of compression, quantization is a lossy operation, and the loss by quantization cannot be recovered.

[0099] The coefficients of the quantized transformed residual are then coded by means of an entropy coder 312 and then inserted into the compressed bit stream 310 as a part of the useful data coding the images of the AV information. Coding syntax elements may also be coded using spatial dependencies between syntax elements to increase the coding efficiency. HEVC offers context-adaptive binary arithmetic coding (CABAC). Other forms of entropy or arithmetic coding may also be used.

[0100] In order to calculate the predictors used above, the encoder 202 decodes already encoded PUs 702 using “decoding” loop 315, which includes elements 316, 318, 320, 322, 328. This decoding loop 315 reconstructs the PUs and images from the quantized transformed residuals.

[0101] The quantized transform residual coefficients E are provided to dequantizer 316, which applies the inverse operation to that of quantizer 310 to produce dequantized transform coefficients of the residual PU (E') 708. The dequantized data 708 is then provided to inverse transformer 318 which applies the inverse of the transform applied by the transform module 308 to generate reconstructed residual coefficients of the PU (e) 710.

[0102] The reconstructed coefficients of the residual PU 710 are then added to the corresponding coefficients of the corresponding predicted PU (x') 702' selected from the intra-predicted PU 1004 and the inter-predicted PU 1106 by selector 306. For example, if the reconstructed residual comes from the “intra” coding process of the spatial predictor 324, the “intra” predictor (x') is added to this residual in order to recover a reconstructed PU (x'') 712 corresponding to the original PU 702 modified by the losses resulting from a transformation, for example in this case the quantization operations. If the residual 710 comes from an “inter” coding process of the temporal predictor 330, the areas pointed to by the current motion vectors (these areas belong to the reference images stored in reference buffer 328 referred by the current image indices) are merged then added to this decoded residual. In this way the original PU 702 is modified by the losses resulting from the quantization operations.

[0103] To the extent that the encoder 202 uses motion vector prediction techniques analogous to the image prediction techniques described above, the motion vector may be stored using motion vector buffer 329 for use in temporally subsequent frames. As further described below, a flag may be set and transferred in the syntax to indicate that the motion vector for the

currently decoded frame should be used for at least the subsequently coded frame instead of replacing the contents of the MV buffer 329 with the MV for the current frame.

[0104] A loop filter 322 is applied to the reconstructed signal (x') 712 in order to reduce the effects created by heavy quantization of the residuals obtained, and to improve the signal quality. The loop filter 322 may comprise, for example, a deblocking filter for smoothing borders between PUs to visually attenuate high frequencies created by the coding process and a linear filter that is applied after all of the PUs for an image have been decoded to minimize the sum of the square difference (SSD) with the original image. The linear filtering process is performed on a frame by frame basis and uses several pixels around the pixel to be filtered, and also uses spatial dependencies between pixels of the frame. The linear filter coefficients may be coded and transmitted in one header of the bit stream typically a picture or slice header.

[0105] The filtered images, also known as reconstructed images, are then stored as reference images from reference image buffer 328 in order to allow the subsequent “Inter” predictions taking place during the compression of the subsequent images of the current video sequence.

Reference Image Syntax

[0106] As described above, to reduce errors and improve compression, HEVC permits the use of several reference images for the estimation and motion compensation of the current image. Given a current PU 702 in a current picture, the collocated PU 1102 for a particular slice resides in associated nearby reference/non-reference picture. For example, in FIG. 12, the collocated PU 1102 for current PU 702 in picture (i) resides in the associated nearby reference picture (i-1). The best “inter” or temporal predictors of the current PU 702 are selected in some of the multiple reference/non-reference images, which may be based on pictures temporally prior to or after the current picture in display order (backwards and forward prediction, respectively).

[0107] For HEVC, the index to reference pictures is defined by reference picture lists that are described in the slice syntax. Forward prediction is defined by list_0 (RefPicList0), and backward prediction is defined by list_1 (RefPicList1), and both list 0 and list 1 can contain multiple reference pictures prior to or/and later than the current picture in the display order.

[0108] FIG. 13 illustrates an example of the use of the reference picture lists. Consider pictures 0, 2, 4, 5, 6, 8 and 10 shown in FIG. 13, wherein the numbers of each picture denote display order and the current picture is picture 5. In this case, the list_0 reference pictures with

ascending reference picture indices and starting with index equal to zero are 4, 2, 0, 6, 8 and 10, and the list_1 reference pictures with ascending reference picture indices and starting with index equal to zero are 6, 8, 10, 4, 2, and 0. A slice that the motion compensated prediction is restricted to the list_0 prediction is called a predictive or P-slice. Collocated pictures are indicated by using the collocated_ref_idx index in the HEVC. A slice for which the motion-compensated prediction includes more than one reference picture is a bi-predictive or B-slice. For B-slices, the motion compensated prediction may include reference pictures from list_1 prediction as well as list_0.

[0109] Hence, a collocated PU 1102 is disposed in a reference picture specified in either list_0 or list_1. A flag (collocated_from_l0_flag) is used to specify whether the collocated partition should be derived from list_0 or list_1 for a particular slice type. Each of the reference pictures is also associated with a motion vector.

[0110] The storage and retrieval of reference pictures and related motion vectors for the emerging HEVC standard is expressed in paragraph 8.4.1.2.9 of Benjamin Bross, Woo-Jin Han, Jens-Rainer Ohm, Gary J. Sullivan, Thomas Wiegand, “WD4: Working Draft 4 of High-Efficiency Video Coding, “Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JCTVC-F803_d5, 6th Meeting: Torino, IT, 14-22 July, 2011.

[0111] According to the standard, if the slice_type is equal to B and the collocated_from_l0_flag is 0, the collocated_ref_idx variable specifies the reference picture as the picture that contains the co-located partition as specified by RefPicList1. Otherwise (slice_type is equal to B and collocated_from_l0_flag is equal to 1 or slice_type is equal to P), the collocated_ref_idx variable specifies the reference picture as the picture that contains the collocated partition as specified by RefPicList0.

[0112] FIG. 14 is a diagram illustrating processes performed by the encoder 202 according to the aforementioned standard. Block 1402 determines whether the current picture is a reference picture for another picture. If not, there is no need to store reference picture or motion vector information. If the current picture is a reference picture for another picture, block 1504 determines whether the “another” picture is a P-type or a B-type picture. If the picture is a P-type picture, processing is passed to blocks 1410, which set the colloc_from_l0_flag to one and store the reference picture and motion vector in list 0. If the “another picture” is a B-type

picture, block 1406 nonetheless directs processing to blocks 1408 and 1410 if the desired reference picture is to be stored in list 0, and to blocks 1412 and 1414 if the desired reference picture and motion vector is to be stored in list 1. This decision may be based on whether it is desirable to select reference pictures from a temporally preceding or succeeding picture. Which of the multiple possible reference pictures is selected is determined according to the `collocated_ref_idx` index.

[0113] FIG. 15 depicts the use of a `collocated_from_10_flag` by the decoder 220 in decoding a according to the previous HEVC standard. Block 1502 determines if the current slice type being computed is an intra or I-type. Such slices do not use temporally nearby slices in the encoding/decoding process, and hence there is no need to find a temporally nearby reference picture. If the slice type is not I-type, block 1504 determines whether the slice is a B-slice. If the slice is not a B-type, it is a P-type slice, and the reference picture that contains the collocated partition is found in list 0, according to the value of `collocated_ref_idx`. If the slice is B-type, the `collocated_from_10_flag` determines whether the reference picture is found in list 0 or list 1. As the index indicates, the collocated picture is therefore defined as the reference picture having the indicated `collocated_ref_idx` in either list 0 or list 1, depending on the slice type (B-type or P-type) and the value of the `collocated_from_10_flag`. In one embodiment of HEVC, the first reference picture (the reference picture having index [0] as shown in FIG. 13 is selected as the collocated picture).

Baseline Picture Parameter Set Syntax

[0114] FIGs. 16A and 16B are diagrams presenting a baseline PPS Raw Byte Sequence Payload (RBSP) syntax. Syntax for dealing with extensions in the PPS are shown in FIG. 16B. Logic 1602 determines if the media is to be coded/decoded including a first extension and reads the appropriate signaling and data. Logic 1602 comprises statements 1606-1616. Statement 1606 reads a `pps_extensional_flag`, which indicates whether the first extension has been selected for the coding/decoding process. In one embodiment, a logical value of “1” indicates that the media is to be processed using the first extension, and a logical value of “0” indicates that the media is not to be processed using the first extension. Statement 1608 is a conditional statement that directs execution of statements 1612-1614 depending upon the value of a (previously read) `transform_skip_enabled_flag`. In particular, the illustrated logic performs the

operations shown in statements 1612-1614 if the `transform_skip_enabled_flag` is a logical “1” or true. The `transform_skip_enabled_flag` 1601 of the PPS syntax is shown in FIG. 16A.

[0115] Transform skipping is an extension that allows the DCT transform of a TU to be skipped under certain circumstances. Essentially, the DCT transform has the property that for media with highly correlated signals, it results in outstanding energy compaction. However, for media with highly uncorrelated signals (e.g. media having a large amount of detail), the compaction performance is much less. For some media, the DCT transform process has so little compaction performance, the process is better skipped for better processing performance. The `transform_skip_enabled_flag` indicates when skipping the DCT transform of a TU is permitted. This is described, for example, in “Early Termination of Transform Skip Mode for High Efficiency Video Coding,” by Do Kyung Lee, Miso Park, Hyung-Do Kim and Je-Chang Jeong in the Proceedings of the 2014 International Conference on Communications, Signal Processing and Computers. If the `transform_skip_enabled` flag is a logical 1 (true), processing is routed to statement 1612 and 1614. Otherwise, processing is routed to statement 1618.

Statement 1612 performs the operation of reading a value `log2_transform_skip_max_size_minus2`, which indicates the maximum TU size that may be skipped (if the `transform_skip_enabled_flag` indicates that performing the DCT transform of the TU is permitted). Statement 1614 performs the operation of reading a flag `pps_extension2_flag` indicating if a further extension (`extension2`) is implemented.

[0116] Next, logic 1604 is performed. Logic 1604 includes statements 1618-1622. Statement 1618 is a conditional statement that routes processing to the logic of statements 1620 and 1622 if the `pps_extension2_flag` is a logical 1. Statements 1620 and 1622 read additional `pps_extension_data_flags` while RBSP data exists.

[0117] In the foregoing PPS design of HEVC range extension, the `pps_extension2_flag` accounts for as yet unidentified extension data. According to the logic described above, if `pps_extension1_flag` is true, `pps_extension2_flag` is present. If `pps_extension1_flag` is not true, `pps_extension2_flag` is not present. If `pps_extension2_flag` is not present, `pps_extension2_flag` is inferred to be equal to 0. If `pps_extension2_flag` is 0, there is no additional extension data.

[0118] This logical formulation always checks the value of `pps_extension2_flag` for possible additional extension syntax regardless of the status of `pps_extension1_flag`. However, if `pps_extension1_flag` is 0, there is no need to check `pps_extension2_flag`, because if

pps_extension1_flag is 0, pps_extension2_flag will not be present, and if pps_extension2_flag is not present, it will be inferred to be equal to 0, which indicates that there is no further extension data.

[0119] Related U.S. Utility Patent Application Serial No. 14/533,386, entitled “MODIFICATION OF PICTURE PARAMETER SET (PPS) FOR HEVC EXTENSIONS,” describes the modification of the foregoing syntax in which logic 1604 (statements 1616-1620) of FIG. 16B is incorporated within the conditional statement 1608, and is executed only if pps_extension1_flag tests to a logical 1. This allowed the logic of statements 1610-1620 to be skipped if pps_extension1_flag tests to a logical 0, thus saving execution time.

[0120] This design works when there is only one PPS extension (e.g. the transform skip extension) to be enabled and perhaps a second PPS extension to read additional data (e.g. signaled by the pps_extension2_flag) that is to be performed only if the first PPS extension is performed as well. However, if there are additional PPS extensions, this design may not be efficient because the syntax demands that later extensions must parse all previous extension syntax, even though the previously executed extension and/or syntax may be independent from or unrelated to later executed extension(s) and/or syntax.

Improved Picture Parameter Set Syntax

[0121] FIGs. 17A-17D are diagrams presenting a modified PPS Raw Byte Sequence Payload (RBSP) syntax. In summary, the modified RBSP syntax defines an extension presence signaling flag (pps_extension_present_flag) that signals whether the picture in the sequence is to be processed at least in part according to at least one extension function. If the pps_extension_present_flag tests false, it is known that there are no PPS extensions to follow, and the need for syntax logic defining and handling such extensions are no longer required and the processing associated with executing such syntax logic is no longer performed, thus saving processing resources, memory resources, and processing time. The modified PPS RBSP syntax also includes one or more extension signaling flags, each signaling the presence of an associated PPS extension function. This increases the efficiency in the parsing and execution of the PPS syntax, because the one or more extension signaling flags, associated data and logical instructions need not be stored in the syntax, read or executed by the processor.

[0122] In one embodiment, the PPS RBSP syntax is further modified so that the extension signaling flags are indexed and read iteratively. For example, n PPS extension signaling flags may be denoted as `pps_extension_flag[i]` wherein i is an index having a value from 0 to $n-1$. In one embodiment, seven PPS extension signaling flags are defined ($n = 7$) may be used. Each such individual PPS extension flag may control the parsing specific extension function syntax. For example, a first PPS extension flag may controls parsing of HEVC range extension related syntax and a second PPS extension flag may control parsing of MV-HEVC related syntax.

[0123] In another embodiment, the foregoing may be extended to accommodate greater than n ($n > 8$) extensions by use additional `pps_extension_7bits` syntax. This additional syntax permits signaling of further extensions that may be designated in the future above seven PPS flags be insufficient to the task. In a preferred embodiment, the number of extension bits (as well as the maximum value of the index described above) is set to multiples of 8 bits (0-7) so that byte-by-byte parsing may be easily accomplished.

[0124] FIG. 17A is a flow chart illustrating exemplary operations that can be employed to encode/decode a sequence of a plurality of pictures using one or more extension functions. In block 1700, an extension presence signaling flag is read. The extension presence signaling flag indicates whether the picture referenced by the PPS syntax is to be processed at least in part according to at least one extension function. In block 1702, a determination is made as to whether the read extension presence signaling flag indicates that the picture associated with the PPS syntax is to be processed at least in part with at least one extension function. In one embodiment, this is accomplished by determining whether the extension function presence signaling flag has a first value. The “value” may be a logical value (e.g. true or false) or may be a numerical or alphanumeric value (e.g. 1 or 0) indicating a logical value. If the extension presence signaling flag does not have a first value (indicating that picture associated with the PPS syntax is to be processed without any extension functions, the operations shown in blocks 1704-1708 are bypassed. If the extension presence signaling flag is determined to indicate that the picture is to be decoded at least in part according to at least one extension function, the first extension function signaling flag is read as shown in block 1704, and regardless of the value of the first extension function signaling flag, a second extension function signaling function is read, as shown in block 1706. Hence, because the second extension function signaling flag is read regardless of the value or status of the previously read first extension function signaling flag, the

reading of the second extension function signaling flag is independent of the value of the read first extension function signaling flag. This is in contrast to the syntax shown in FIG. 16B, in which the `pps_extension2_flag` is only read by logic 1614 if the `pps_extension1_flag` tests true in logic 1608. Finally, as depicted in block 1708, the extension function(s) signaled by the flags read in blocks 1704-1706 are performed.

[0125] FIG. 17B is a diagram presenting exemplary PPS syntax that can be used to perform the operations shown in FIG. 17A to decode a sequence of a plurality of pictures using one or more extension functions. In the illustrated embodiment, the extension function(s) include the TU DCT transform skipping extension functionality described above.

[0126] Logic 1712 reads the extension presence signaling flag. In the illustrated embodiment, the extension presence signaling flag comprise a `pps_extension_present_flag`. Logic 1714 tests to determine if the `pps_extension_present_flag` is logically true, and only if so, executes logic 1716-1740. If it is determined that the `pps_extension_present_flag` is logically false, processing is routed to logic 1740. Importantly, this means that no extension flags will be read, and no extension function processing will be performed.

[0127] Logic 1716-1720 of FIG. 17B performs the operations described in blocks 1704 and 1706 of FIG. 17A, and hence, reads the extension function signaling flag(s) (here, the `pps_extension_flag[i]` and/or `pps_extension_7bits` flag). In particular, logic 1716 and 1718 read `pps_extension_flag[i]` for $i=0$ to $n-1$ (in the exemplary embodiment $n=1$, so only one flag is read, namely `pps_extension_flag[0]`). Logic 1720 reads a value `pps_extension_7bits`, which is used to signal additional extension functionality beyond up to 7 extension functions referenced by `pps_extension_flag[0]` - `pps_extension_flag[6]`.

[0128] Logic 1722 and 1728 tests to determine if the read `pps_extension_flag[0]` had a logical value indicating that the related extension function (TU DCT skipping) is desired. If the `pps_extension_flag[0]` has such a value (e.g. tests logically true), logic 1724-1730 is executed.

[0129] Logic 1724 tests to determine if transform skipping is enabled by testing the `transform_skip_enabled_flag`. If enabled (e.g. the `transform_skip_enabled_flag` tests true), logic 1726-1728 of the PPS syntax is executed. Logic 1726-1728 reads the value represented by `log2_max_transform_skip_block_size_minus2`, which specifies the maximum transform unit (TU) block size for which the DCT transformation may be skipped.

[0130] Logic 1732 tests to determine if the value of pps_extension_7bits that was read by logic 1760 tests true. If so, logic 1734-1738 reads such additional signaling bits.

[0131] FIG. 17C is a flow chart further illustrating the exemplary PPS syntax presented in FIG. 17B, which as described above, reads all of the extension function signaling flags (e.g. pps_extension_flag[i] - pps_extension_flag[n-1]) first, and then performs each extension function, one after the other.

[0132] Referring to FIG. 17C, block 1750 reads the extension presence signaling flag. Block 1752 determines whether the extension presence signaling flag has a value indicating that at least one extension function is to be performed. If the extension presence signaling flag indicates that no extension functions are to be performed, processing is routed after block 1758. If the extension presence signaling flag indicates that one or more extension functions are to be performed, processing is routed to block 1753, which reads all extension function signaling flags (e.g. pps_extension_flag[i] - pps_extension_flag[n-1]). Processing is then passed to block 1754, which tests to determine if the first extension function signaling flag has a value that signals that the first extension function is to be performed. FIG. 17B illustrates exemplary syntax for performing this test at logic 1722.

[0133] If the extension function signaling flag indicates that the extension function is not to be performed, processing is around blocks 1756 and 1758. Syntax for performing these operations is exemplified by logic 1722 to logic 1730 of FIG. 17B. If the extension function signaling flag indicates that the extension function is to be performed, processing is routed to block 1756, and at least a portion of the extension function processing is performed. Syntax for performing these operations is illustrated in FIG. 17B by logic 1724-1728, which reads the maximum transform unit block size for which the DCT transform may be skipped if indicated by the transform_skip_enabled_flag 1601.

[0134] Block 1758 tests to determine whether all extension functions have been considered. If all extension functions have been considered, processing concludes (analogous to logic 1740 in the syntax illustrated in FIG. 17B). If all extension functions have not been considered, processing is routed to block 1760, which brings the next function signaling flag for consideration by block 1754.

[0135] The foregoing illustrates processing logic in which all of the extension function signaling flags are read, and each extension function is then executed, one at a time. This embodiment is

especially useful in embodiments wherein the extension function signaling flag is read with an incremented index as shown in logic 1716 and 1718, as it decouples the reading of the flag (which is indexed) from the execution of the extension function itself (which may or may not be indexed). For example, the processing loop represented by blocks 1754-1760 may be performed by simply including syntax for performing each extension function, one after the other (e.g. executing logic 1754-1756 and then executing further logic to perform the next extension function, with logical statements interposed between logic 1756 and 1758). Or, they may be performed using an incremented index, which may be the same index used to read the extension function signaling flags, or a different index.

[0136] FIG. 17D is a diagram of a flow chart illustrating an alternate embodiment wherein instead of reading all extension function signaling flags before beginning to perform the extension functions themselves, each extension function signaling flag is read and the extension function is performed before reading the next extension function signaling flag. Block 1760 reads a first extension function signaling flag (which may be indexed), and block 1762 tests whether the read first extension function signaling flag indicates that the first extension function is to be performed. If the function is not to be performed, processing is routed to block 1768, and the extension function is not performed. However, if the first extension function signaling flag indicates that the extension function is to be performed, processing is routed to block 1764, where such processing is performed before routing processing to block 1768. Upon completion of this processing, block 1768 determines if all extension function signaling flags have been read. If so, processing exits, but if not, the next extension function signaling flag is considered, as illustrated by block 1770. A second extension function signaling flag is read, and the operations of blocks 1760-1768 is repeated for that second extension function signaling flag and its associated second extension function. This may also be accomplished via the use of one or more incremented indexes, and a different index would be used for reading the extension function signaling flags and performing the extension functions themselves.

[0137] FIG. 18 is a diagram presenting an embodiment of the PPS syntax for HEVC range extension. As before, the `pps_extension_present_flag` read in logic statement 1712 specifies that at least one `pps_extension_flag[i]` is present in the PPS syntax. This `pps_extension_present_flag` is used in logic statement 1714 to indicate that logic statements 1716 and 1718 should be performed, and these logical statements read `pps_extension_flag[i]` for

$i = 0$ to n . A `pps_extension_flag[i]` value of 1 specifies that the syntax structure for the associated `pps_extension` is present, while a `pps_extension_flag[i]` value of 0 specifies that the syntax structure for the `pps_extension` associated with the flag is not present.

[0138] In the exemplary syntax shown in FIG. 18, a `pps_extension_flag[0]` value of 1 specifies that the following HEVC range extension-related elements are present in the PPS RBSP syntax structure, as shown in logic statements 1724, 1726 and 1804-1820:

- `log2_max_transform_skip_block_size_minus2`;
- `luma_chroma_prediction_enabled_flag`;
- `chroma_qp_adjustment_enabled_flag`;
- `diff_cu_chroma_qp_adjustment_depth`;
- `chroma_qp_adjustment_table_size_minus1`;
- `cb_qp_adjustment`;
- `cr_qp_adjustment`.

[0139] Conversely, a `pps_extension_flag[0]` equal to 0 specifies that these syntax elements are not present.

[0140] Further, a `pps_extension_7bits` value of 0 specifies that no more `pps_extension_data_flag` syntax elements are present in the PPS RBSP syntax structure, and logic statement 1822 skips logic statements 1824 and 1828. The `pps_extension_7bits` shall have a value of 0 in bit streams conforming to legacy specification versions, as `pps_extension_7bits` values not equal to 0 are reserved for future use by ITU-T/ ISO/IEC. HEVC Decoders shall allow the value of `pps_extension_7bits` to be not equal to 0 and shall ignore all `pps_extension_data_flag` syntax elements in an PPS NAL unit.

[0141] FIGs. 19A-19C show further alternate embodiments of extension signaling syntax. FIG. 19A illustrates a generalized syntax in which the extension presence signaling flag (`pps_extension_present_flag`) is used to signal whether further extension function syntax is present in the PPS. As before, logic statements 1712 reads the `pps_extension_present_flag`. Logic statement 1714 commands execution of logic statements 1716-1742 only if the `pps_extesion_present_flag` indicates that syntax for one or more extension functions is present in the PPS. Logic statement 1716 reads a `pps_extension_flag[i]` for all values of i , and logic

statement 1720 reads `pps_extension_7bits`. Logic statements 1732-1740 read a `pps_extension_data_flag` and associated data.

[0142] FIG. 19B illustrates a PPS syntax wherein the extension function signaling flags are read in separate statements, rather than via an incremented index within a processing loop. Specifically, logic statements 1902-1906 read a first flag (`pps_range_extension_flag`) indicating that the range extension processing is to be performed, a second flag (`pps_multilayer_extension_flag`), indicating that multilayer or multi-view (MV-HEVC) extension processing is to be performed, and a third flag (`pps_extension_bits6`) for reading further extension data. Logic statements 1910-1912 perform the `pps_range_extension()` processing (which may be disposed in a separate PPS range extension syntax referenced by the `pps_range_extension()` logical statement) as indicated by the `pps_range_extension_flag` read by logic statement 1902. Logic statements 1914-1916 perform the `pps_multilayer_extension()` (also possibly specified in distinct PPS syntax referenced by the `pps_multilayer_extension()` logical statement) as indicated by the `pps_multilayer_extension_flag`. Logic statements 1918-1926 read a `pps_extension_data_flag` and associated data.

[0143] FIG. 19C illustrates a PPS syntax where the extension function signaling flags are read using an incremented index, but tested and used to perform extension processing in separate, unindexed statements. Specifically, logic statements 1930 -1932 read two `pps_extension_flags`, namely `pps_extension_flag[0]` and `pps_extension_flag[1]` using an index *i* taking the value of 0 and 1. Logic statement 1934 reads the `pps_extension_6bits` value, and logic statements 1938-1952 operate analogously to logic statements 1910-1926, except the `pps_extension_flag` is referenced and distinguished by an index of [0] or [1] rather than a different name.

[0144] Other embodiments of the foregoing syntax are also envisioned. For example, it is possible to group the extension presence signaling flags (e.g. `pps_extension_flag`) by type or category. This permits extensions that have similar data requirements to be signaled and processed together, thereby saving syntax statements and decoder processing.

[0145] As described above, the signaled extension functions may be independent, or may be functionally related. For example, a second extension function may require use of a result of a first (previously processed or executed) extension function, before the second extension function may be completed. Or, the second extension function may be mutually exclusive from the first extension function (e.g. either the first extension function or the second extension

function are to be performed, but not both). Or, the second extension function may be a function that would not be performed unless the first extension function is also performed, hence the second extension function is only implicated or performed in the processing sequence if the first extension function is also performed. For example, a computation may require an output or result from both the first extension function and the second extension function, and hence, existence of the first extension function necessarily implicates the second extension function and vice-versa.

[0146] The foregoing operations are described with respect to a decoding process, which can take place in either a the source decoder 220 or an encoder 202, as a part of the encoding process. The encoding process may also be expressed as comprising determining if a slice of the one or more slices is an inter-predicted slice according to slice type data, and if the slice is an inter-predicted slice, configuring a first parameter in the slice header associated with the slice to a value signaling enablement of a state of weighted prediction of image data associated with the slice.

Hardware Environment

[0147] FIG. 20 illustrates an exemplary processing system 2000 that could be used to implement the embodiments of the invention. The computer 2002 comprises a processor 2004 and a memory, such as random access memory (RAM) 2006. The computer 2002 is operatively coupled to a display 2022, which presents images such as windows to the user on a graphical user interface 2018B. The computer 2002 may be coupled to other devices, such as a keyboard 2014, a mouse device 2016, a printer, etc. Of course, those skilled in the art will recognize that any combination of the above components, or any number of different components, peripherals, and other devices, may be used with the computer 2002.

[0148] Generally, the computer 2002 operates under control of an operating system 2008 stored in the memory 2006, and interfaces with the user to accept inputs and commands and to present results through a graphical user interface (GUI) module 2018A. Although the GUI module 2018A is depicted as a separate module, the instructions performing the GUI functions can be resident or distributed in the operating system 2008, the computer program 2010, or implemented with special purpose memory and processors. The computer 2002 also implements a compiler 2012 which allows an application program 2010 written in a

programming language such as COBOL, C++, FORTRAN, or other language to be translated into processor 2004 readable code. After completion, the application 2010 accesses and manipulates data stored in the memory 2006 of the computer 2002 using the relationships and logic that was generated using the compiler 2012. The computer 2002 also optionally comprises an external communication device such as a modem, satellite link, Ethernet card, or other device for communicating with other computers.

[0149] In one embodiment, instructions implementing the operating system 2008, the computer program 2010, and the compiler 2012 are tangibly embodied in a computer-readable medium, e.g., data storage device 2020, which could include one or more fixed or removable data storage devices, such as a zip drive, floppy disc drive 2024, hard drive, CD-ROM drive, tape drive, etc. Further, the operating system 2008 and the computer program 2010 are comprised of instructions which, when read and executed by the computer 2002, causes the computer 2002 to perform the steps necessary to implement and/or use the invention. Computer program 2010 and/or operating instructions may also be tangibly embodied in memory 2006 and/or data communications devices 2030, thereby making a computer program product or article of manufacture. As such, the terms “article of manufacture,” “program storage device” and “computer program product” as used herein are intended to encompass a computer program accessible from any computer readable device or media.

[0150] The processing system 2000 may also be embodied in a desktop, laptop, tablet, notebook computer, personal data assistant (PDA), cellphone, smartphone, or any device with suitable processing and memory capability. Further, the processing system 2000 may utilize special purpose hardware to perform some or all of the foregoing functionality. For example the encoding and decoding processes described above may be performed by a special purpose processor and associated memory.

[0151] Those skilled in the art will recognize many modifications may be made to this configuration without departing from the scope of the present disclosure. For example, those skilled in the art will recognize that any combination of the above components, or any number of different components, peripherals, and other devices, may be used. For example, particular functions described herein can be performed by hardware modules, or a processor executing instructions stored in the form of software or firmware. Further, the functionality described herein can be combined in single modules or expanded to be performed in multiple modules.

Conclusion

[0152] The foregoing description of the preferred embodiment has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the disclosure to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of rights be limited not by this detailed description, but rather by the claims appended hereto.

What is claimed is:

1. A method of decoding a plurality of pictures, each picture processed at least in part according to a picture parameter set, the method comprising:
 - receiving a bitstream comprising the plurality of pictures and the picture parameter set;
 - parsing the picture parameter set to determine for a picture in the plurality of pictures whether a pps_extension_present_flag signaling flag specifies presence of syntax structure pps_extension_Xbits at a picture level for the picture,
 - wherein the pps_extension_present_flag and pps_extension_Xbits signaling flags, when present, are adaptable per picture in the plurality of pictures, and
 - wherein the pps_extension_Xbits signaling flag is represented in the picture parameter set by multiple bits, where X = the number of said bits;
 - parsing the pps_extension_Xbits syntax structure to determine if any pps_extension_data_flag syntax structures are present in the picture parameter set;
 - wherein pps_extension_Xbits shall be equal to 0 for bitstreams conforming to High Efficiency Video Coding (HEVC) profiles, and
 - wherein pps_extension_Xbits not equal to 0 causes the pps_extension_data_flag syntax structures in a picture parameter set network abstraction layer (NAL) unit to be ignored during decoding.

2. The method of claim 1, further comprising:
 - reading a first extension function signaling flag signaled in the picture parameter set and signaling a first extension function for decoding the picture;
 - reading a second extension function signaling flag signaled in the picture parameter set and signaling a second extension independently of a value of the read first extension function signaling flag for decoding the picture;
 - performing decode processing at the picture level in accordance with a syntax structure in the picture parameter set according to the first extension function and the second extension function at the picture level in accordance with the syntax structure in the picture parameter set;
 - for each picture, reading a further extension presence signaling flag after processing the first extension function and the second extension function without processing further extension functions; and
 - for each picture, determining from the further extension presence signaling flag that the read extension presence signaling flag indicates that there are no extensions to follow, and upon determining no extensions follow, the decoding for the respective picture proceeds without checking or executing further syntax logic to handle processing of extensions,

wherein the second extension function is independent from the first extension function.

3. The method of claim 2, further comprising:

determining if the first extension function signaling flag indicates that the picture is to be processed at least in part according to the first extension function.

4. The method of claim 3, further comprising:

performing the first extension function only if the first extension function signaling flag indicates that the picture is to be processed at least in part according to the first extension function.

5. The method of claim 4, further comprising:

determining if the second extension function signaling flag indicates that the picture is to be processed at least in part according to the second extension function, and performing the second extension function only if the second extension function signaling flag indicates that the picture is to be processed at least in part according to the second extension function.

6. The method of claim 2, wherein the first extension function signaling flag is uniquely associated with a first value of an index, and the second extension function signaling flag is uniquely associated with a second value of the index, and wherein:

the reading the first extension function signaling flag and the second extension function signaling flag comprises:

reading the first extension function signaling flag value according to the index;

incrementing the index; and

reading the second extension function signaling flag value according to the incremented index.

7. The method of claim 6, wherein:

the first extension function comprises a range extension function; and

the second extension function comprises a high efficiency video coding (HEVC) multilayer or multiview extension function.

8. An apparatus for decoding a plurality of pictures, each picture processed at least in part according to a picture parameter set, the apparatus comprising:

a processor;

a memory, communicatively coupled to the processor, the memory storing a plurality of instructions comprising instructions for:

receiving a bitstream comprising the plurality of pictures and the picture parameter set;
parsing the picture parameter set to determine for a picture in the plurality of pictures whether a pps_extension_present_flag signaling flag specifies presence of syntax structure pps_extension_Xbits at a picture level for the picture,
wherein the pps_extension_present_flag and pps_extension_Xbits signaling flags, when present, are adaptable per picture in the plurality of pictures, and
wherein the pps_extension_Xbits signaling flag is represented in the picture parameter set by multiple bits, where X = the number of said bits;
parsing the pps_extension_Xbits syntax structure to determine if any pps_extension_data_flag syntax structures are present in the picture parameter set;
wherein pps_extension_Xbits shall be equal to 0 for bitstreams conforming to High Efficiency Video Coding (HEVC) profiles, and
wherein pps_extension_Xbits not equal to 0 causes the pps_extension_data_flag syntax structures in a picture parameter set network abstraction layer (NAL) unit to be ignored during decoding.

9. The apparatus of claim 8, further comprising:

reading a first extension function signaling flag signaled in the picture parameter set and signaling a first extension function for decoding the picture;

reading a second extension function signaling flag signaled in the picture parameter set and signaling a second extension independently of a value of the read first extension function signaling flag for decoding the picture;

performing decode processing at the picture level in accordance with a syntax structure in the picture parameter set according to the first extension function and the second extension function at the picture level in accordance with the syntax structure in the picture parameter set;

for each picture, reading a further extension presence signaling flag after processing the first extension function and the second extension function without processing further extension functions; and

for each picture, determining from the further extension presence signaling flag that the read extension presence signaling flag indicates that there are no extensions to follow, and upon determining no extensions follow, the decoding for the respective picture proceeds without checking or executing further syntax logic to handle processing of extensions, and

wherein the second extension function is independent from the first extension function.

10. The apparatus of claim 9, wherein the instructions further comprise instructions for: determining if the first extension function signaling flag indicates that the picture is to be processed at least in part according to the first extension function.

11. The apparatus of claim 10, wherein the instructions further comprise instructions for: performing the first extension function only if the first extension function signaling flag indicates that the picture is to be processed at least in part according to the first extension function.

12. The apparatus of claim 11, wherein the instructions further comprise instructions for: determining if the second extension function signaling flag indicates that the picture is to be processed at least in part according to the second extension function, and performing the second extension function only if the second extension function signaling flag indicates that the picture is to be processed at least in part according to the second extension function.

13. The apparatus of claim 9, wherein the first extension function signaling flag is uniquely associated with a first value of an index, and the second extension function signaling flag is uniquely associated with a second value of the index, and wherein:

the instructions for reading the first extension function signaling flag and the second extension function signaling flag comprises instructions for:

reading the first extension function signaling flag value according to the index;

incrementing the index; and

reading the second extension function signaling flag value according to the incremented index.

14. The apparatus of claim 13, wherein:

the first extension function comprises a range extension function; and

the second extension function comprises a high efficiency video coding (HEVC) multilayer or multiview extension function.

15. A method of decoding a sequence of a plurality of pictures in a processing device, each picture decoded at least in part according to a picture parameter set comprising:

(a) reading an extension presence signaling flag within said picture parameter set, where said picture parameter set applies to said decoding of said each picture inside said sequence of said plurality of pictures, where a sequence parameter set contains information which applies to all pictures of said sequence of said plurality of pictures, where said picture parameter set is separate from said sequence parameter set;

(b) determining if the read extension presence signaling flag within said picture parameter set indicates that the picture is to be processed at least in part according to at least one extension function;

(c) only if the read extension presence signaling flag within said picture parameter set indicates that the picture is to be processed at least in part according to the at least one extension function:

reading a first extension function signaling flag within said picture parameter set signaling a first extension function;

reading a second extension function signaling flag within said picture parameter set signaling a second extension function, independently of a value of the read first extension function signaling flag, in a raw byte sequence payload while there is data in said raw byte sequence payload;

reading at least one additional function, based upon said reading of said second extension function signaling flag, in the raw byte sequence payload while there is more data in said raw byte sequence payload.

16. The method of claim 15, wherein the second extension function is independent from the first extension function.

17. The method of claim 15, further comprising:

(d) determining if the first extension function signaling flag indicates that the picture is to be processed at least in part according to the first extension function.

18. The method of claim 17, wherein (d) is performed only if the read extension presence signaling flag indicates that the picture is to be processed at least in part according to the at least one extension function.

19. The method of claim 17, wherein (d) is performed subsequent to (a)-(c).

20. The method of claim 18, further comprising:

(e) performing the first extension function only if the first extension function signaling flag indicates that the picture is to be processed at least in part according to the first extension function.

21. The method of claim 20, further comprising:
- (f) determining if the second extension function signaling flag indicates that the picture is to be processed at least in part according to the second extension function, and performing the second extension function only if the second extension function signaling flag indicates that the picture is to be processed at least in part according to the second extension function.
22. The method of claim 15, wherein the first extension function signaling flag is uniquely associated with a first value of an index, and the second extension function signaling flag is uniquely associated with a second value of the index, and wherein:
- the reading the first extension function signaling flag and the second extension function signaling flag comprises:
- reading the first extension function signaling flag value according to the index;
 - incrementing the index; and
 - reading the second extension function signaling flag value according to the incremented index.
23. The method of claim 22, wherein:
- the first extension function comprises a range extension function; and
 - the second extension function comprises a high efficiency video coding (HEVC) multilayer or multiview extension function.
24. An apparatus for decoding a sequence comprising a plurality of pictures, each picture decoded at least in part according to a picture parameter set, the apparatus comprising:
- a processor;
 - a memory, communicatively coupled to the processor, the memory storing a plurality of instructions comprising instructions for:
- (a) reading an extension presence signaling flag within said picture parameter set, where said picture parameter set applies to said decoding of said each picture inside said sequence of said plurality of pictures, where a sequence parameter set contains information which applies to all pictures of said sequence of said plurality of pictures, where said picture parameter set is separate from said sequence parameter set;
- (b) determining if the read extension presence signaling flag within said picture parameter set indicates that the picture is to be processed at least in part according to at least one extension function;

(c) only if the read extension presence signaling flag within said picture parameter set indicates that the picture is to be processed at least in part according to the at least one extension function:

reading a first extension function signaling flag within said picture parameter set signaling a first extension function;

reading a second extension function signaling flag within said picture parameter set signaling a second extension function, independently of a value of the read first extension function signaling flag, in a raw byte sequence payload while there is data in said raw byte sequence payload;

reading at least one additional function, based upon said reading of said second extension function signaling flag, in the raw byte sequence payload while there is more data in said raw byte sequence payload.

25. The apparatus of claim 24, wherein the second extension function is independent from the first extension function.

26. The apparatus of claim 24, wherein the instructions further comprise instructions for:

(d) determining if the first extension function signaling flag indicates that the picture is to be processed at least in part according to the first extension function.

27. The apparatus of claim 26, wherein (d) is performed only if the read extension presence signaling flag indicates that the picture is to be processed at least in part according to the at least one extension function.

28. The apparatus of claim 26, wherein (d) is performed subsequent to (a)-(c).

29. The apparatus of claim 27, wherein the instructions further comprise instructions for:

(e) performing the first extension function only if the first extension function signaling flag indicates that the picture is to be processed at least in part according to the first extension function.

30. The apparatus of claim 29, wherein the instructions further comprise instructions for:

(f) determining if the second extension function signaling flag indicates that the picture is to be processed at least in part according to the second extension function, and performing the second extension function only if the second extension function signaling flag indicates that the picture is to be processed at least in part according to the second extension function.

31. The apparatus of claim 24, wherein the first extension function signaling flag is uniquely associated with a first value of an index, and the second extension function signaling flag is uniquely associated with a second value of the index, and wherein:

the instructions for reading the first extension function signaling flag and the second extension function signaling flag comprises instructions for:

reading the first extension function signaling flag value according to the index;

incrementing the index; and

reading the second extension function signaling flag value according to the incremented index.

32. The apparatus of claim 31, wherein:

the first extension function comprises a range extension function; and

the second extension function comprises a high efficiency video coding (HEVC) multilayer or multiview extension function.

33. One or more computer-readable memory or storage devices storing encoded data as part of a bitstream, the encoded data being organized to facilitate decoding by a video decoder performing operations, the video decoder being implemented using memory and one or more processing units, the operations comprising:

receiving the bitstream comprising a plurality of pictures and a picture parameter set;

parsing the picture parameter set to determine for a picture in the plurality of pictures whether a `pps_extension_present_flag` signaling flag specifies presence of syntax structure `pps_extension_Xbits` at a picture level for the picture,

wherein the `pps_extension_present_flag` and `pps_extension_Xbits` signaling flags, when present, are adaptable per picture in the plurality of pictures, and

wherein the `pps_extension_Xbits` signaling flag is represented in the picture parameter set by multiple bits, where X=the number of said bits;

parsing the `pps_extension_Xbits` syntax structure to determine if any `pps_extension_data_flag` syntax structures are present in the picture parameter set;

wherein the `pps_extension_Xbits` shall be equal to 0 for bitstreams conforming to High Efficiency Video Coding (HEVC) profiles, and

wherein the `pps_extension_Xbits` not equal to 0 causes the `pps_extension_data_flag` syntax structures in a picture parameter set network abstraction layer (NAL) unit to be ignored during decoding.

34. A computing device comprising a video encoder configured to perform operations comprising:

- providing a bitstream comprising a plurality of pictures and a picture parameter set;
- wherein the picture parameter set is used to determine for a picture in the plurality of pictures whether a `pps_extension_present_flag` signaling flag specifies presence of syntax structure `pps_extension_Xbits` at a picture level for the picture;
- wherein the `pps_extension_present_flag` and `pps_extension_Xbits` signaling flags, when present, are adaptable per picture in the plurality of pictures;
- wherein the `pps_extension_Xbits` signaling flag is represented in the picture parameter set by multiple bits, where X=the number of said bits;
- wherein the `pps_extension_Xbits` syntax structure is used to determine if any `pps_extension_data_flag` syntax structures are present in the picture parameter set;
- wherein the `pps_extension_Xbits` shall be equal to 0 for bitstreams conforming to High Efficiency Video Coding (HEVC) profiles, and
- wherein the `pps_extension_Xbits` not equal to 0 causes the `pps_extension_data_flag` syntax structures in a picture parameter set network abstraction layer (NAL) unit to be ignored during decoding.

100

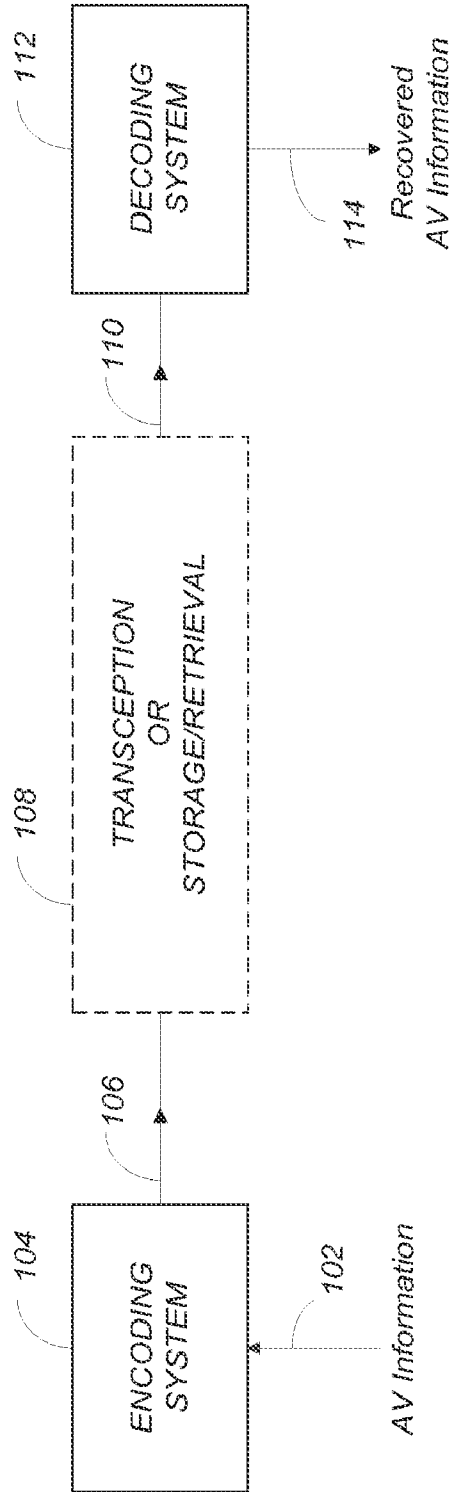


FIG. 1
Prior Art

200A

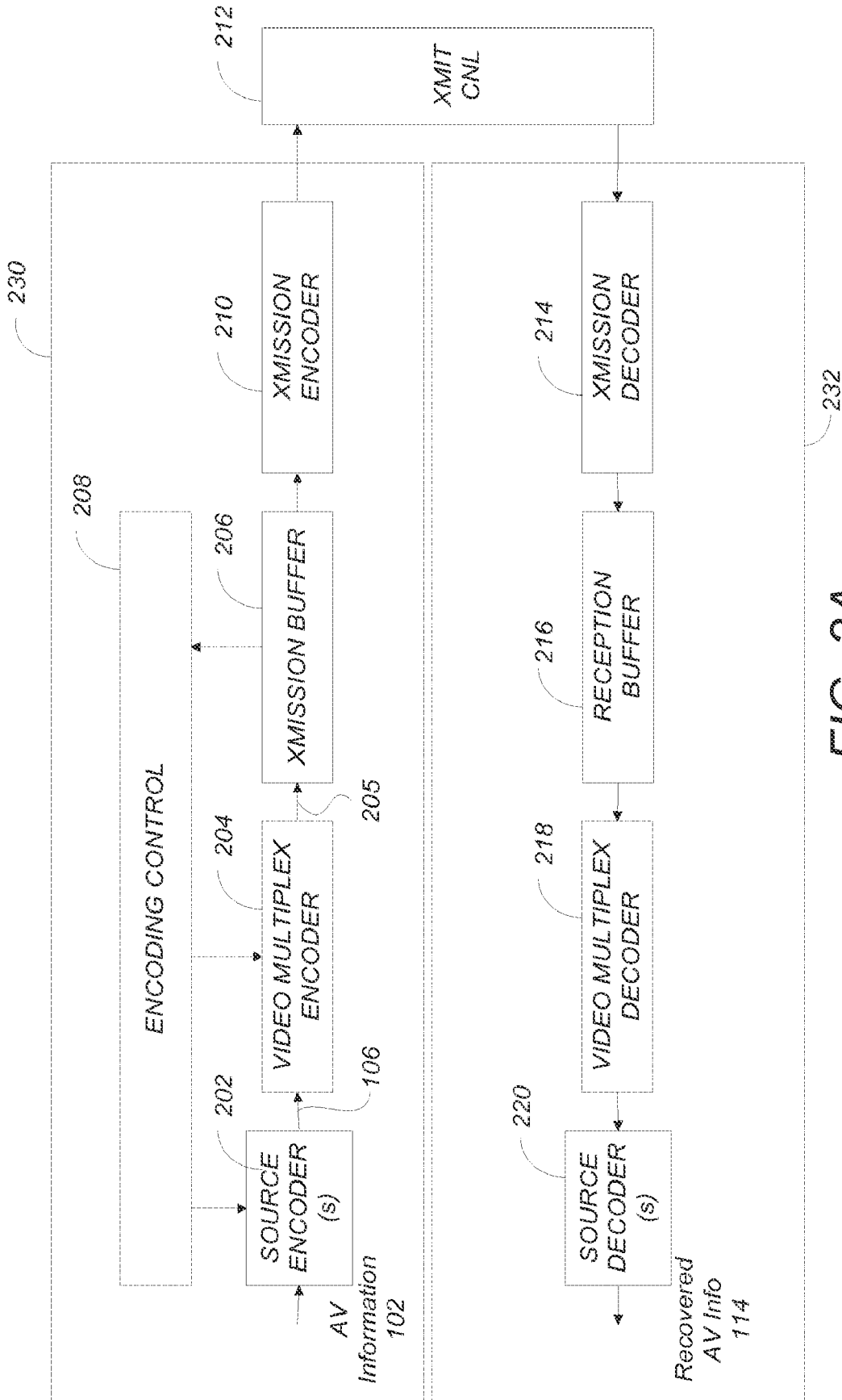


FIG. 2A
Prior Art

3/22

200B

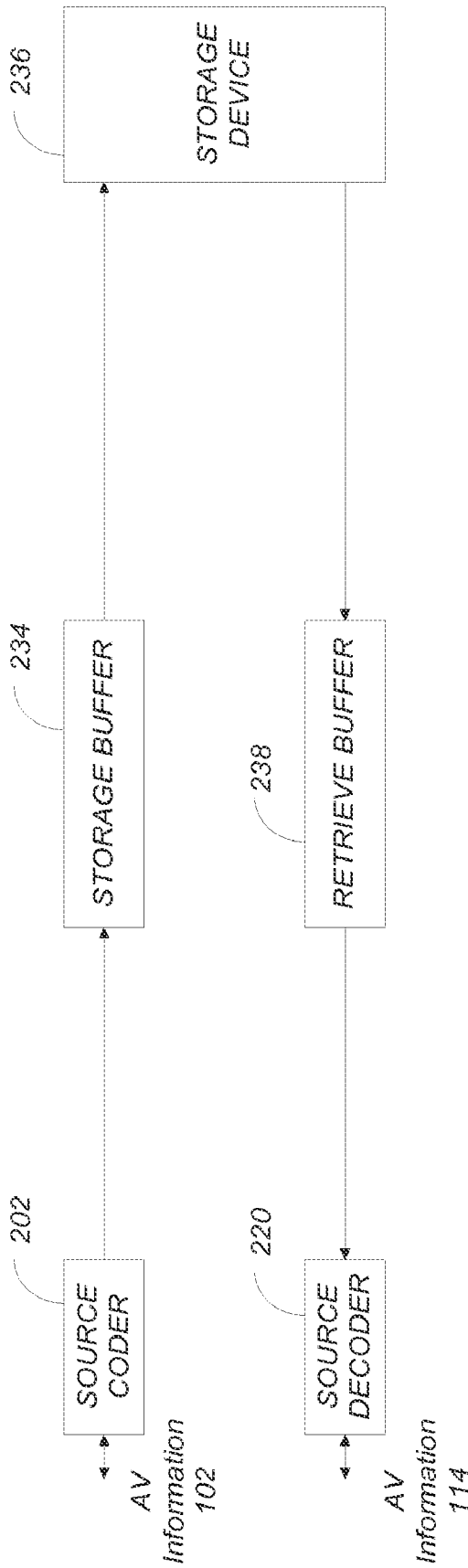


FIG. 2B
Prior Art

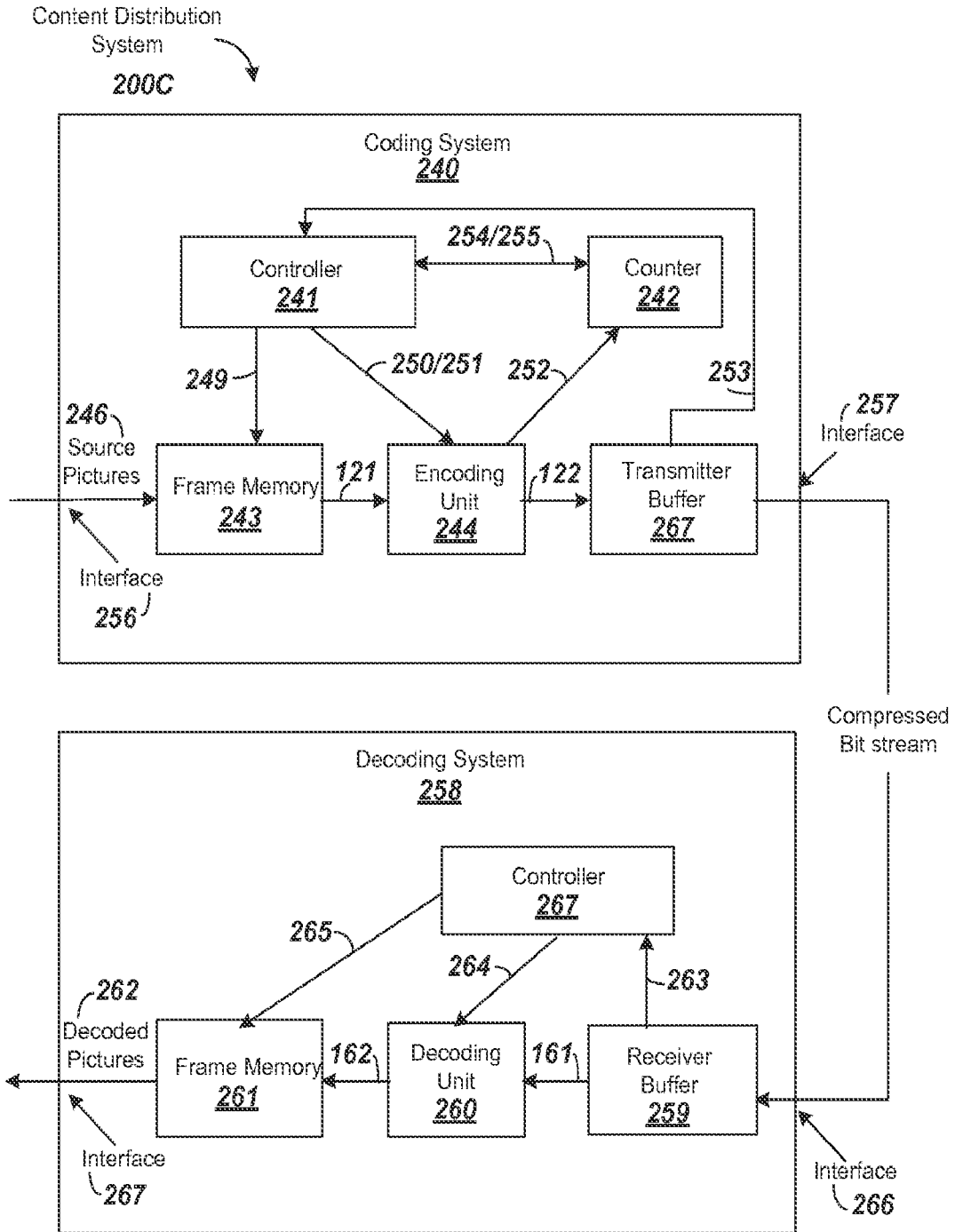


FIG. 2C
Prior Art

6/22

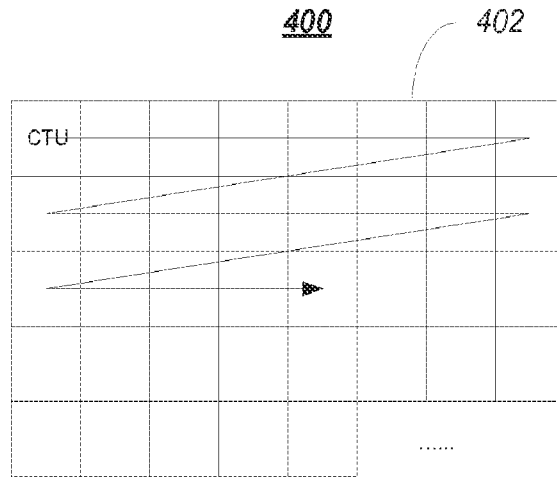


FIG. 4

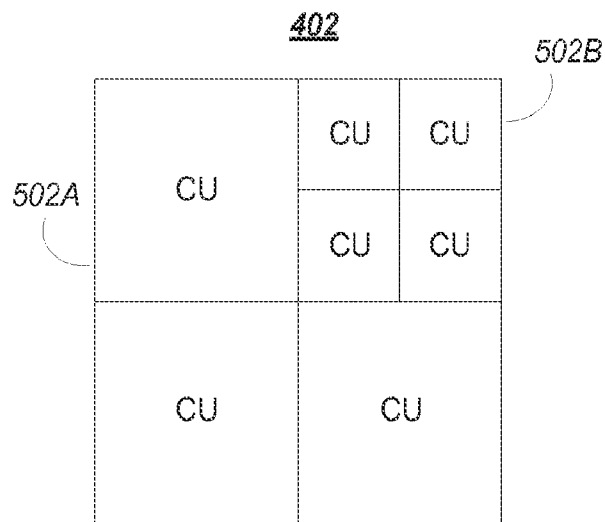


FIG. 5

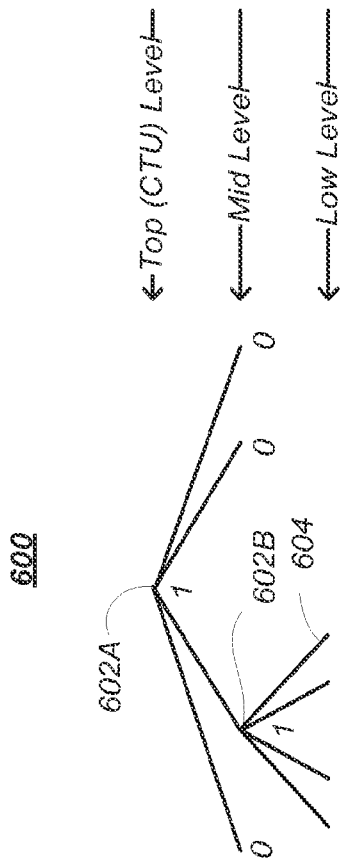


FIG. 6

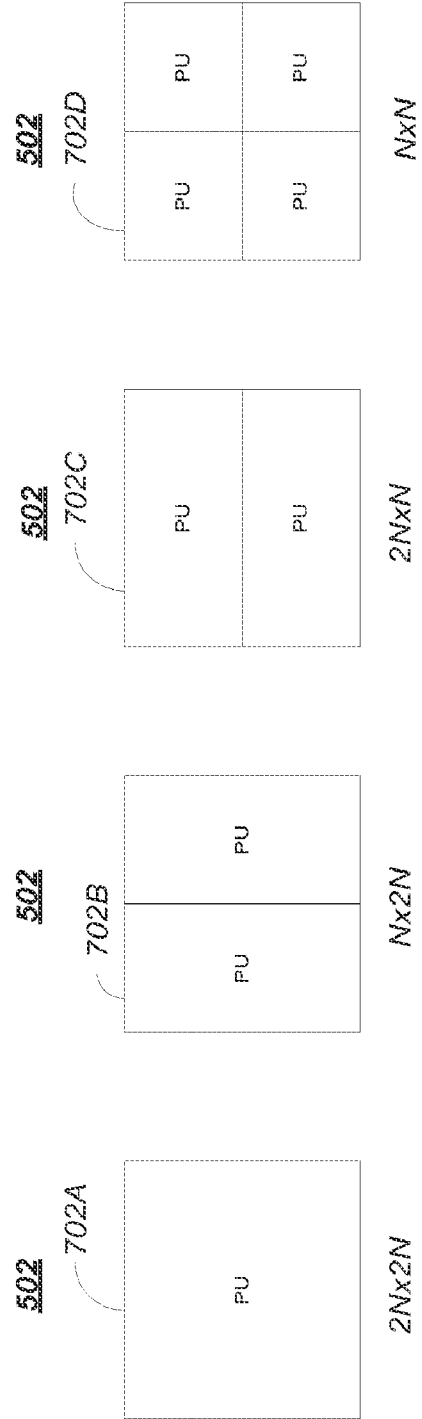


FIG. 7

8/22

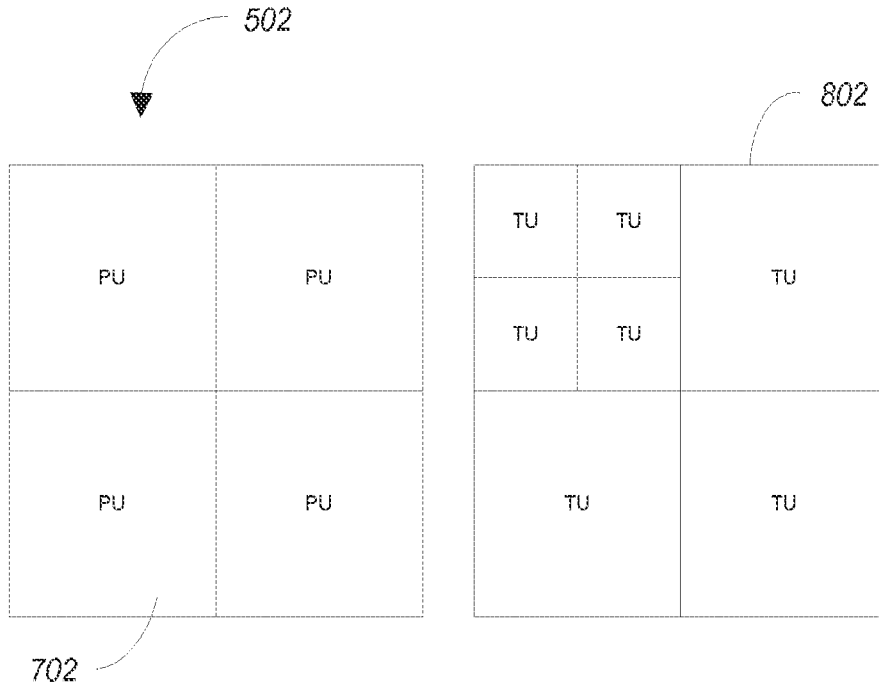


FIG. 8

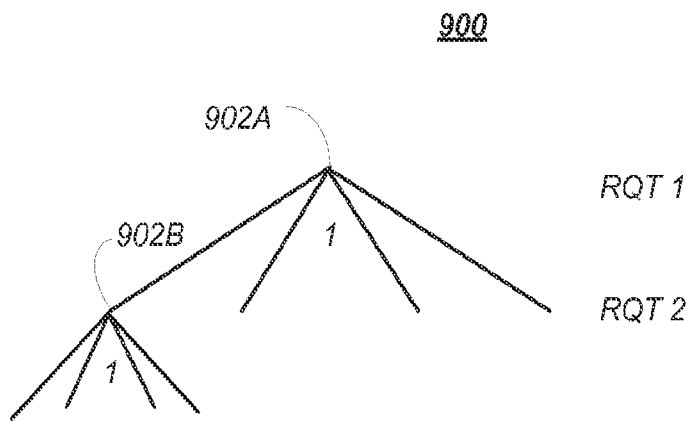


FIG. 9

9/22

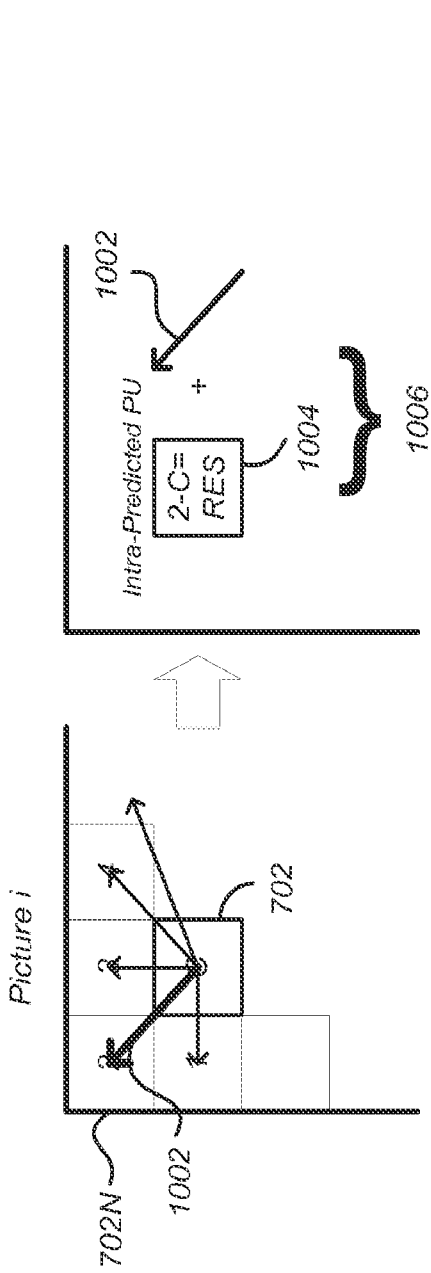


FIG. 10

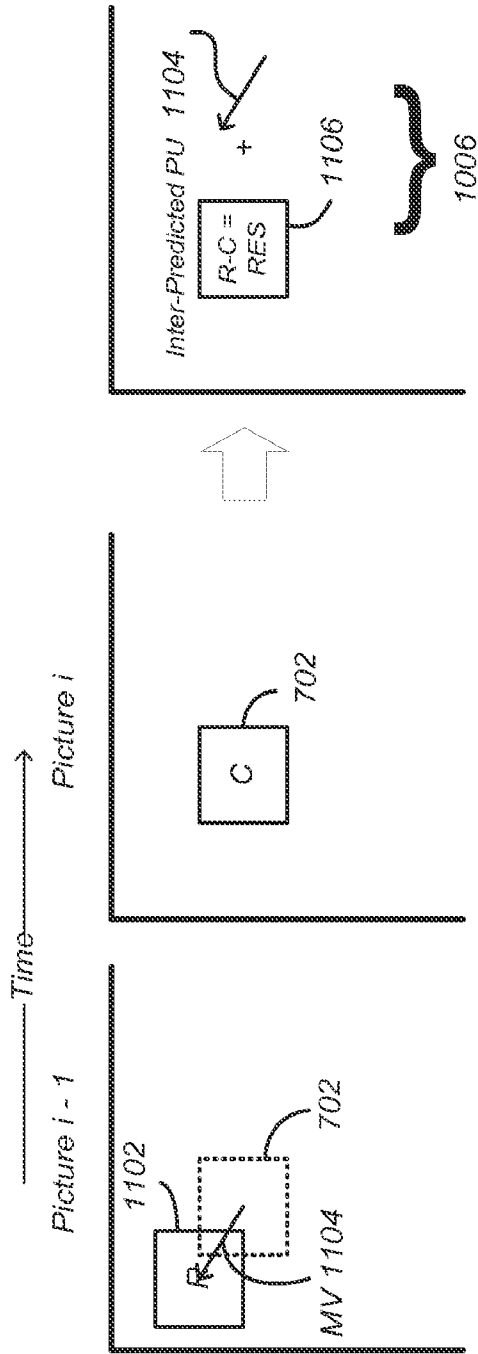


FIG. 11

10/22

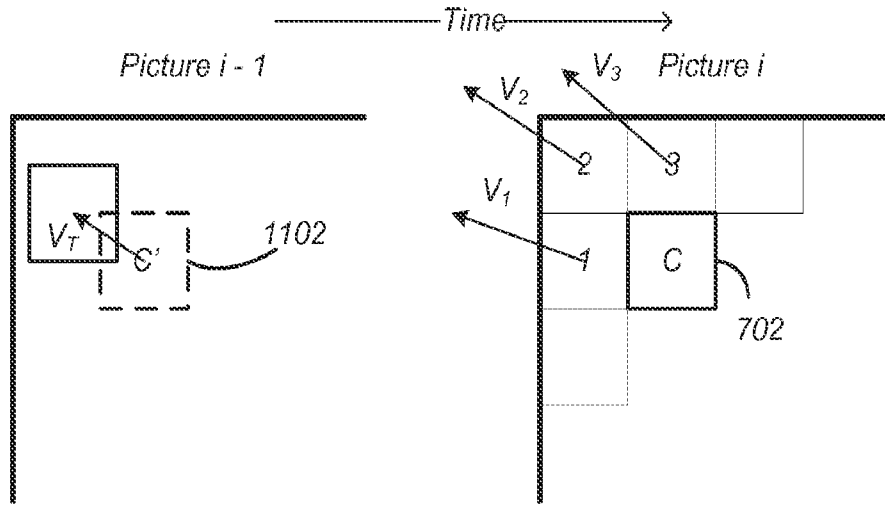
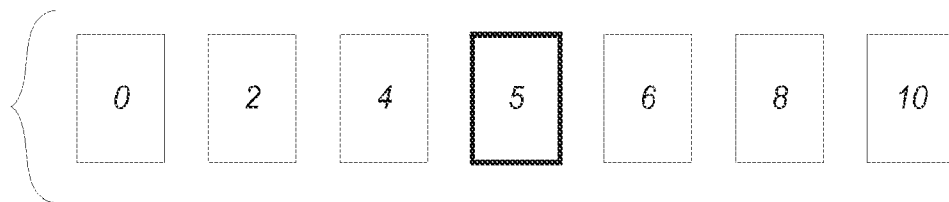


FIG. 12



P = List 0

P, B = List 1

<i>idx</i>	<i>R pix</i>
0	4
1	2
2	0
3	6
4	8
5	10

<i>idx</i>	<i>R pix</i>
0	6
1	8
2	10
3	4
4	2
5	0

FIG. 13

11/22

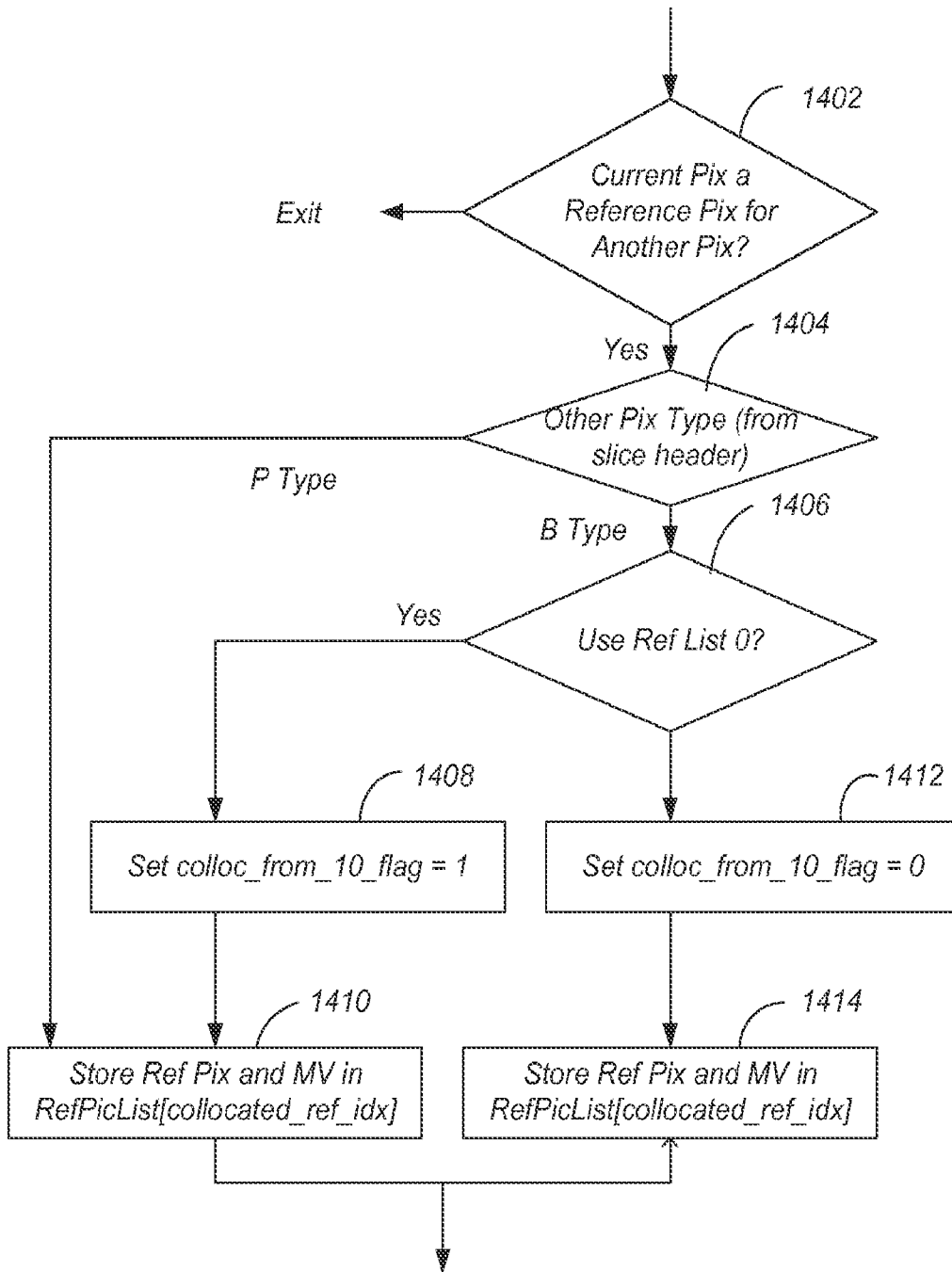


FIG. 14

12/22

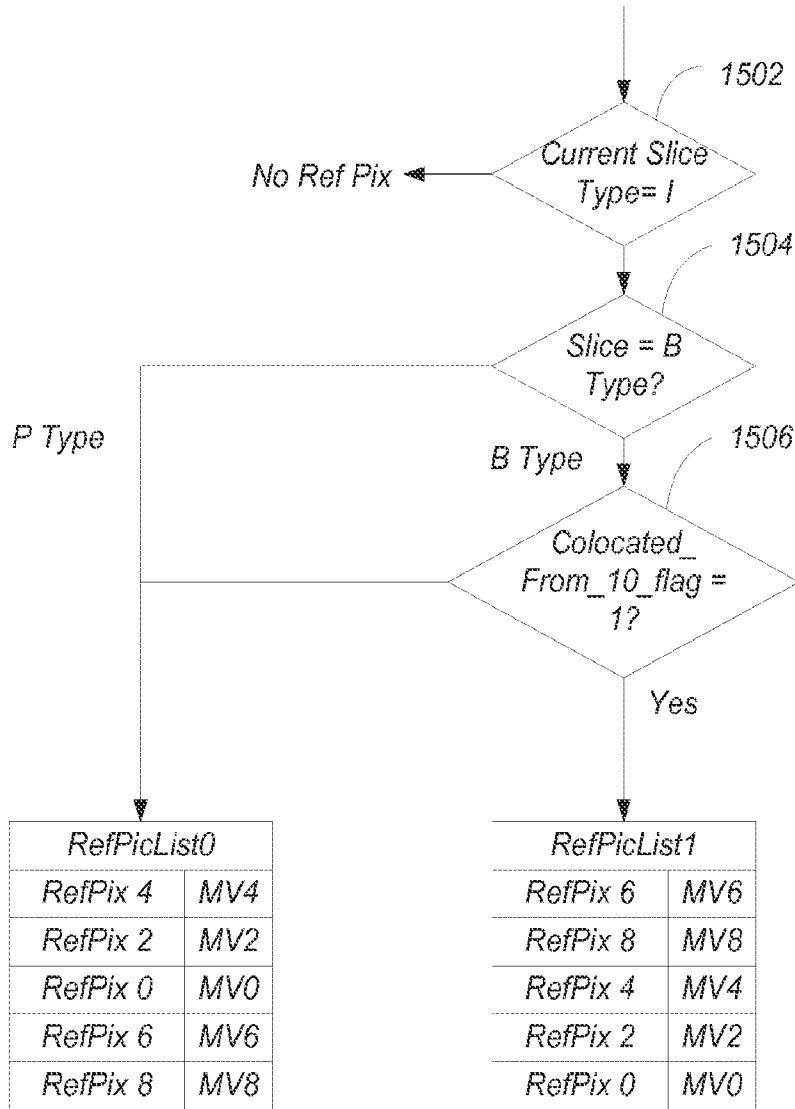


FIG. 15

13/22

PIC_PARAMETER_SET_RBSP()	DESCRIPTOR
PPS_PIC_PARAMETER_SET_ID	UE(V)
PPS_SEQ_PARAMETER_SET_ID	UE(V)
DEPENDENT_SLICE_SEGMENTS_ENABLED_FLAG	U(1)
OUTPUT_FLAG_PRESENT_FLAG	U(1)
NUM_EXTRA_SLICE_HEADER_BITS	U(3)
SIGN_DATA_HIDING_ENABLED_FLAG	U(1)
CABAC_INIT_PRESENT_FLAG	U(1)
NUM_REF_IDX_L0_DEFAULT_ACTIVE_MINUS1	UE(V)
NUM_REF_IDX_L1_DEFAULT_ACTIVE_MINUS1	UE(V)
INIT_QP_MINUS26	SE(V)
CONSTRAINED_INTRA_PRED_FLAG	U(1)
1601 — TRANSFORM_SKIP_ENABLED_FLAG	U(1)
CU_QP_DELTA_ENABLED_FLAG	U1(1)
IF(CU_QP_DELTA_ENABLED_FLAG)	
DIFF_CU_QP_DELTA_DEPTH	UE(V)
PPS_CB_QP_OFFSET	SE(V)
PPS_CR_QP_OFFSET	SE(V)
PPS_SLICE_CHROMA_QP_OFFSETS_PRESENT_FLAG	U(1)
WEIGHTED_PRED_FLAG	U(1)
WEIGHTED_BIPRED_FLAG	U(1)
TRANSQUANT_BYPASS_ENABLED_FLAG	U(1)
TILES_ENABLED_FLAG	U(1)
ENTROPY_CODING_SYNC_ENABLED_FLAG	U(1)
IF(TILES_ENABLED_FLAG) {	
NUM_TILE_COLUMNS_MINUS1	UE(V)
NUM_TILE_ROWS_MINUS1	UE(V)
UNIFORM_SPACING_FLAG	U(1)
IF(!UNIFORM_SPACING_FLAG) {	
FOR(I = 0; I < NUM_TILE_COLUMNS_MINUS1; I++)	
COLUMN_WIDTH_MINUS1 [I]	UE(V)
FOR(I = 0; I < NUM_TILE_ROWS_MINUS1; I++)	
ROW_HEIGHT_MINUS1 [I]	UE(V)
}	
LOOP_FILTER_ACROSS_TILES_ENABLED_FLAG	U(1)
}	

FIG. 16A

14/22

	PPS_LOOP_FILTER_ACROSS_SLICES_ENABLED_FLAG	U(1)
	DEBLOCKING_FILTER_CONTROL_PRESENT_FLAG	U(1)
	IF(DEBLOCKING_FILTER_CONTROL_PRESENT_FLAG){	
	DEBLOCKING_FILTER_OVERRIDE_ENABLED_FLAG	U(1)
	PPS_DEBLOCKING_FILTER_DISABLED_FLAG	U(1)
	IF(!PPS_DEBLOCKING_FILTER_DISABLED_FLAG){	
	PPS_BETA_OFFSET_DIV2	SE(V)
	PPS_TC_OFFSET_DIV2	SE(V)
	}	
	}	
	PPS_SCALING_LIST_DATA_PRESENT_FLAG	U(1)
	IF(PPS_SCALING_LIST_DATA_PRESENT_FLAG)	
	SCALING_LIST_DATA()	
	LISTS_MODIFICATION_PRESENT_FLAG	U(1)
	LOG2_PARALLEL_MERGE_LEVEL_MINUS2	UE(V)
	SLICE_SEGMENT_HEADER_EXTENSION_PRESENT_FLAG	U(1)
1606	PPS_EXTENSION1_FLAG	U(1)
1608	IF(PPS_EXTENSION_FLAG){	
1610	IF(TRANSFORM_SKIP_ENABLED_FLAG)	
1612	LOG2_TRANSFORM_SKIP_MAX_SIZE_MINUS2	UE(V)
1614	PPS_EXTENSION2_FLAG	U(1)
1616	}	
1618	IF(PPS_EXTENSION2_FLAG)	
1620	WHILE(MORE_RBSP_DATA())	
1622	PPS_EXTENSION_DATA_FLAG	U(1)
	RBSP_TRAILING_BITS()	
	}	

FIG. 16B

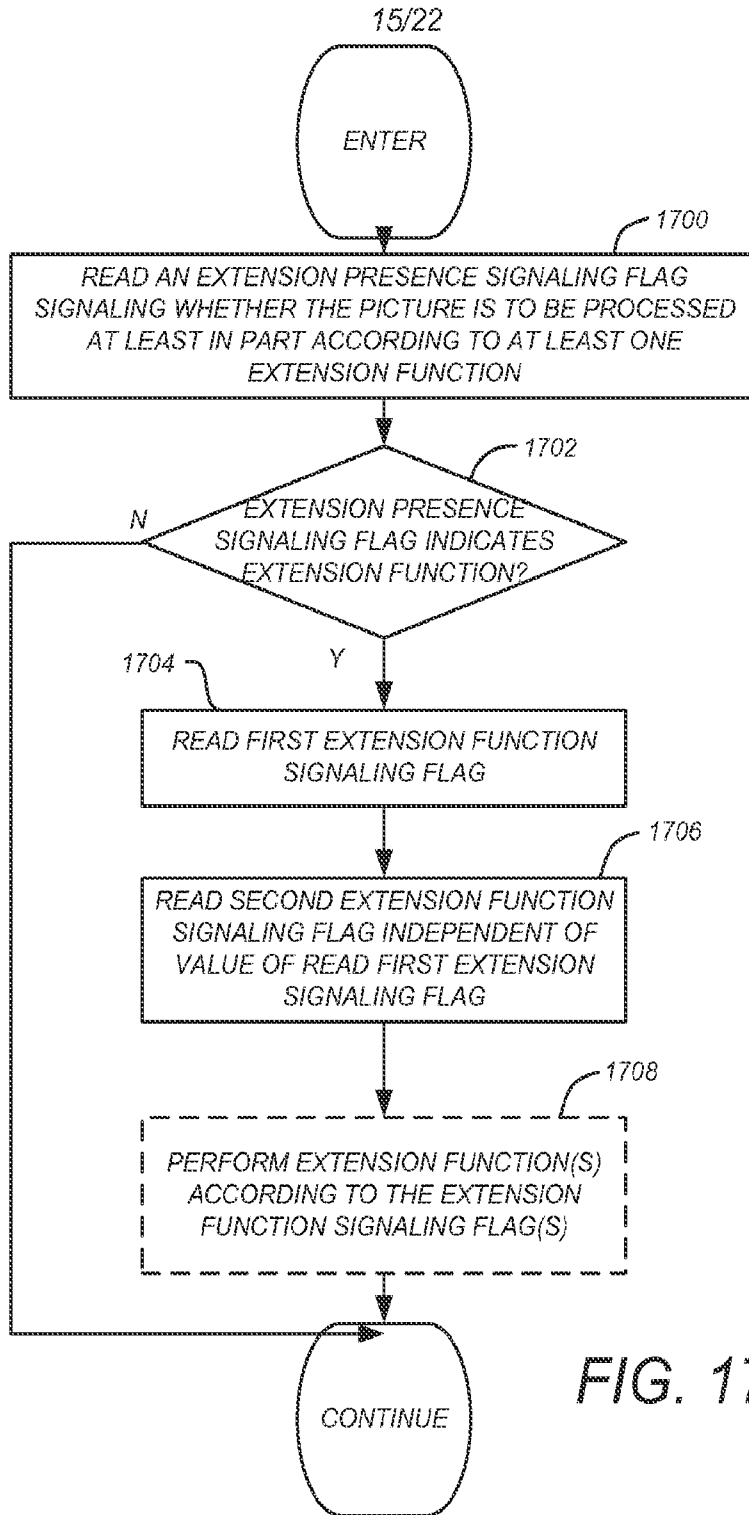
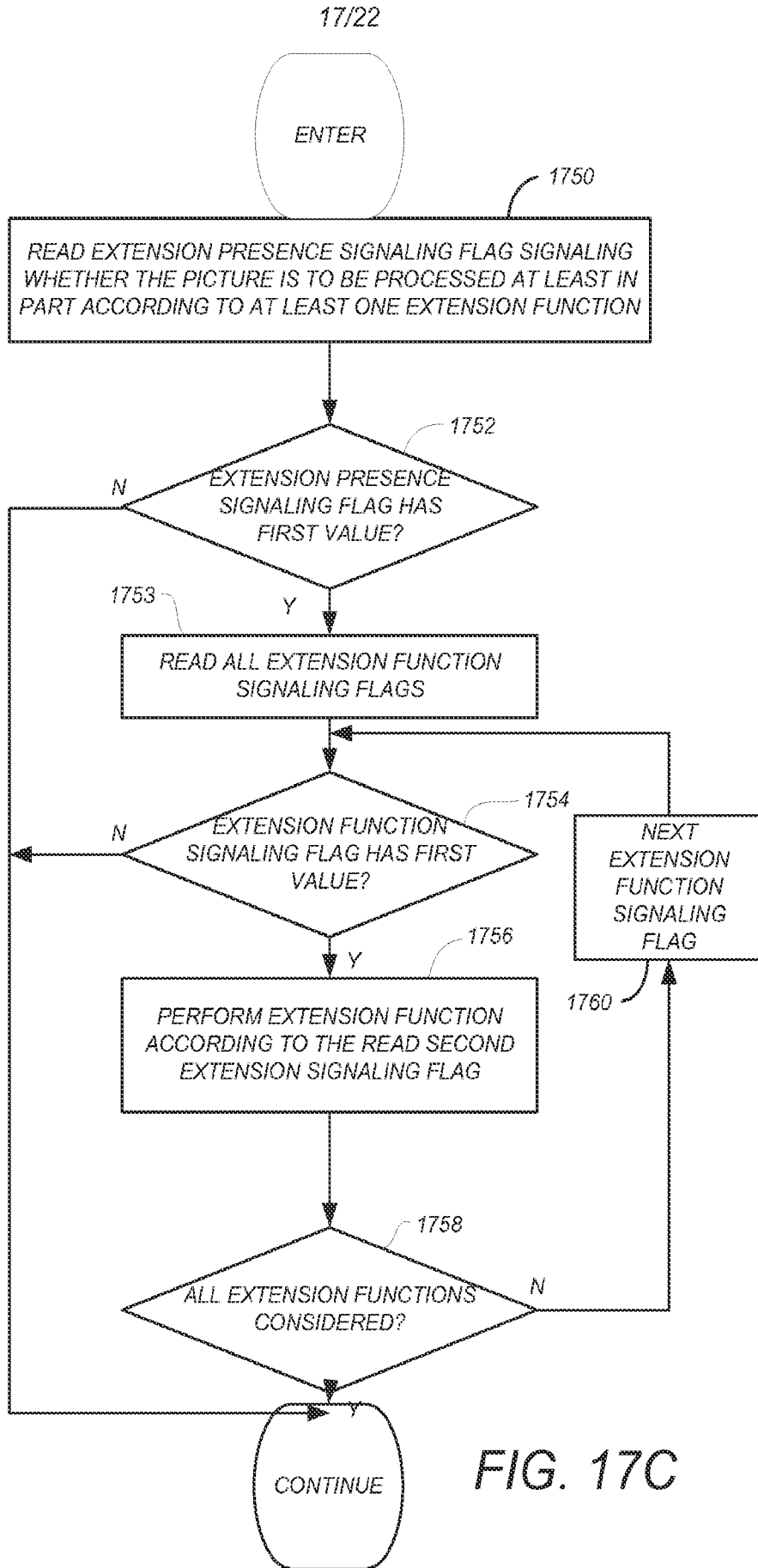


FIG. 17A

16/22

	PPS_LOOP_FILTER_ACROSS_SLICES_ENABLED_FLAG	U(1)
	DEBLOCKING_FILTER_CONTROL_PRESENT_FLAG	U(1)
	IF(DEBLOCKING_FILTER_CONTROL_PRESENT_FLAG){	
	DEBLOCKING_FILTER_OVERRIDE_ENABLED_FLAG	U(1)
	PPS_DEBLOCKING_FILTER_DISABLED_FLAG	U(1)
	IF(!PPS_DEBLOCKING_FILTER_DISABLED_FLAG){	
	PPS_BETA_OFFSET_DIV2	SE(V)
	PPS_TC_OFFSET_DIV2	SE(V)
	}	
	}	
	PPS_SCALING_LIST_DATA_PRESENT_FLAG	U(1)
	IF(PPS_SCALING_LIST_DATA_PRESENT_FLAG)	
	SCALING_LIST_DATA()	
	LISTS_MODIFICATION_PRESENT_FLAG	U(1)
	LOG2_PARALLEL_MERGE_LEVEL_MINUS2	UE(V)
	SLICE_SEGMENT_HEADER_EXTENSION_PRESENT_FLAG	U(1)
1712	PPS_EXTENSION_PRESENT_FLAG	U(1)
1714	IF(PPS_EXTENSION_PRESENT_FLAG){	
1716	FOR(I= 0; I < 1; I++)	
1718	PPS_EXTENSION_FLAG[I]	U(1)
1720	PPS_EXTENSION_7BITS	U(7)
1722	IF(PPS_EXTENSION_FLAG[0]){	
1724	IF(TRANSFORM_SKIP_ENABLED_FLAG)	
1726	LOG2_MAX_TRANSFORM_SKIP_BLOCK_SIZE_MINUS2	UE(V)
1728	}	
1730	}	
1732	IF(PPS_EXTENSION_7BITS)	
1734	WHILE(MORE_RBSP_DATA())	
1738	PPS_EXTENSION_DATA_FLAG	U(1)
1740	}	
	RBSP_TRAILING_BITS()	
	}	

FIG. 17B



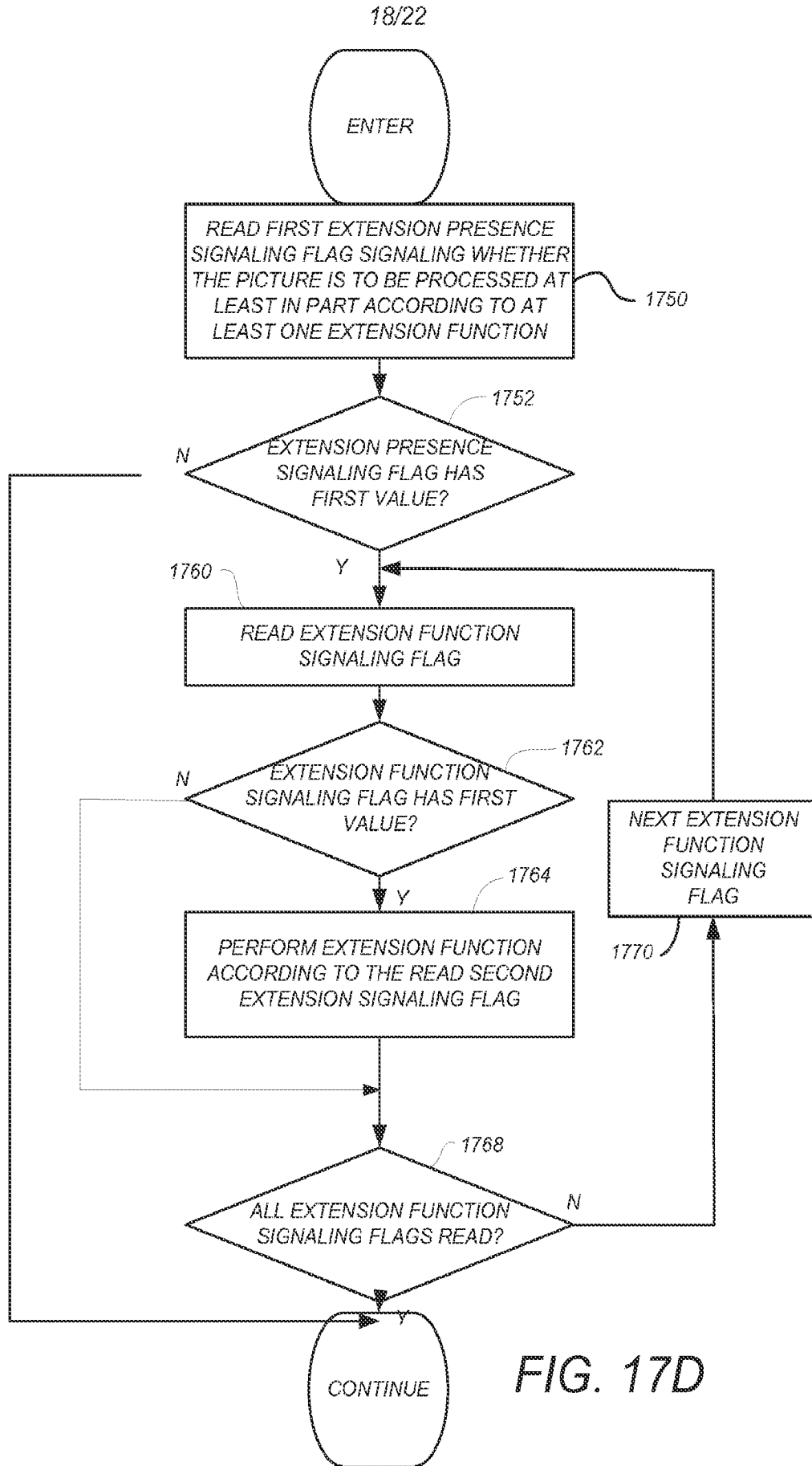


FIG. 17D

19/22

1712	PPS_EXTENSION_PRESENT_FLAG	U(1)
1714	IF(PPS_EXTENSION_PRESENT_FLAG){	
1716	FOR(I = 0; I < 1; I++)	
1718	PPS_EXTENSION_FLAG[I]	U(1)
1720	PPS_EXTENSION_7BITS	U(7)
1722	IF(PPS_EXTENSION_FLAG[0]){	
1724	IF(TRANSFORM_SKIP_ENABLED_FLAG)	
1726	LOG2_MAX_TRANSFORM_SKIP_BLOCK_SIZE_MINUS2	UE(V)
1804	LUMA_CHROMA_PREDICTION_ENABLED_FLAG	U(1)
1806	CHROMA_QP_ADJUSTMENT_ENABLED_FLAG	U(1)
1808	IF(CHROMA_QP_ADJUSTMENT_ENABLED_FLAG){	
1810	DIFF_CU_CHROMA_QP_ADJUSTMENT_DEPTH	UE(V)
1812	CHROMA_QP_ADJUSTMENT_TABLE_SIZE_MINUS1	UE(V)
1814	FOR(I = 0; I <= CHROMA_QP_ADJUSTMENT_TABLE_SIZE_MINUS1; I++){	
1816	CB_QP_ADJUSTMENT[I]	SE(V)
1818	CR_QP_ADJUSTMENT[I]	SE(V)
1820	}	
1822	}	
1824	}	
1820		U(1)
1822	IF(PPS_EXTENSION_7BITS)	
1824	WHILE(MORE_RBSP_DATA())	
1828	PPS_EXTENSION_DATA_FLAG	U(1)
1830	}	
1832	RBSP_TRAILING_BITS()	
	}	

FIG. 18

20/22

1712	PPS_EXTENSION_PRESENT_FLAG	U(1)
1714	IF(PPS_EXTENSION_PRESENT_FLAG){	
1716	FOR (I=0; I < 1; I++)	
1718	PPS_EXTENSION_FLAG[I]	U(1)
1720	PPS_EXTENSION_7BITS	U(7)
1732	IF(PPS_EXTENSION_7BITS) {	
1734	WHILE(MORE_RBSP_DATA())	
1738	PPS_EXTENSION_DATA_FLAG	U(1)
1740	}	
1742	}	
	RBSP_TRAILING_BITS()	
	}	

FIG. 19A

1712	PPS_EXTENSION_PRESENT_FLAG	U(1)
1714	IF(PPS_EXTENSION_PRESENT_FLAG){	
1902	PPS_RANGE_EXTENSION_FLAG	U(1)
1904	PPS_MULTILAYER_EXTENSION_FLAG[I]	U(1)
1906	PPS_EXTENSION_6BITS	U(6)
1908	}	
1910	IF (PPS_RANGE_EXTENSION_FLAG)	
1912	PPS_RANGE_EXTENSION()	
1914	IF(PPS_MULTILAYER_EXTENSION_FLAG)	
1916	PPS_MULTILAYER_EXTENSION()	
1918	IF(PPS_EXTENSION_6BITS)	
1920	WHILE(MORE_RBSP_DATA())	
1922	PPS_EXTENSION_DATA_FLAG	U(1)
1924	RBSP_TRAILING_BITS()	
1926	}	

FIG. 19B

21/22

1712	PPS_EXTENSION_PRESENT_FLAG	U(1)
1714	IF(PPS_EXTENSION_PRESENT_FLAG){	
1930	FOR(I = 0; I < 2; I++)	
1932	PPS_EXTENSION_FLAG[I]	U(1)
1934	PPS_EXTENSION_6BITS	U(6)
1936	}	
1938	IF (PPS_EXTENSION_FLAG[0])	
1940	PPS_RANGE_EXTENSION()	
1942	IF(PPS_EXTENSION_FLAG[1])	
1944	PPS_MULTILAYER_EXTENSION()	
1946	IF(PPS_EXTENSION_6BITS)	
1948	WHILE(MORE_RBSP_DATA())	
1950	PPS_EXTENSION_DATA_FLAG	U(1)
1952	RBSP_TRAILING_BITS()	
1952	}	

FIG. 19C

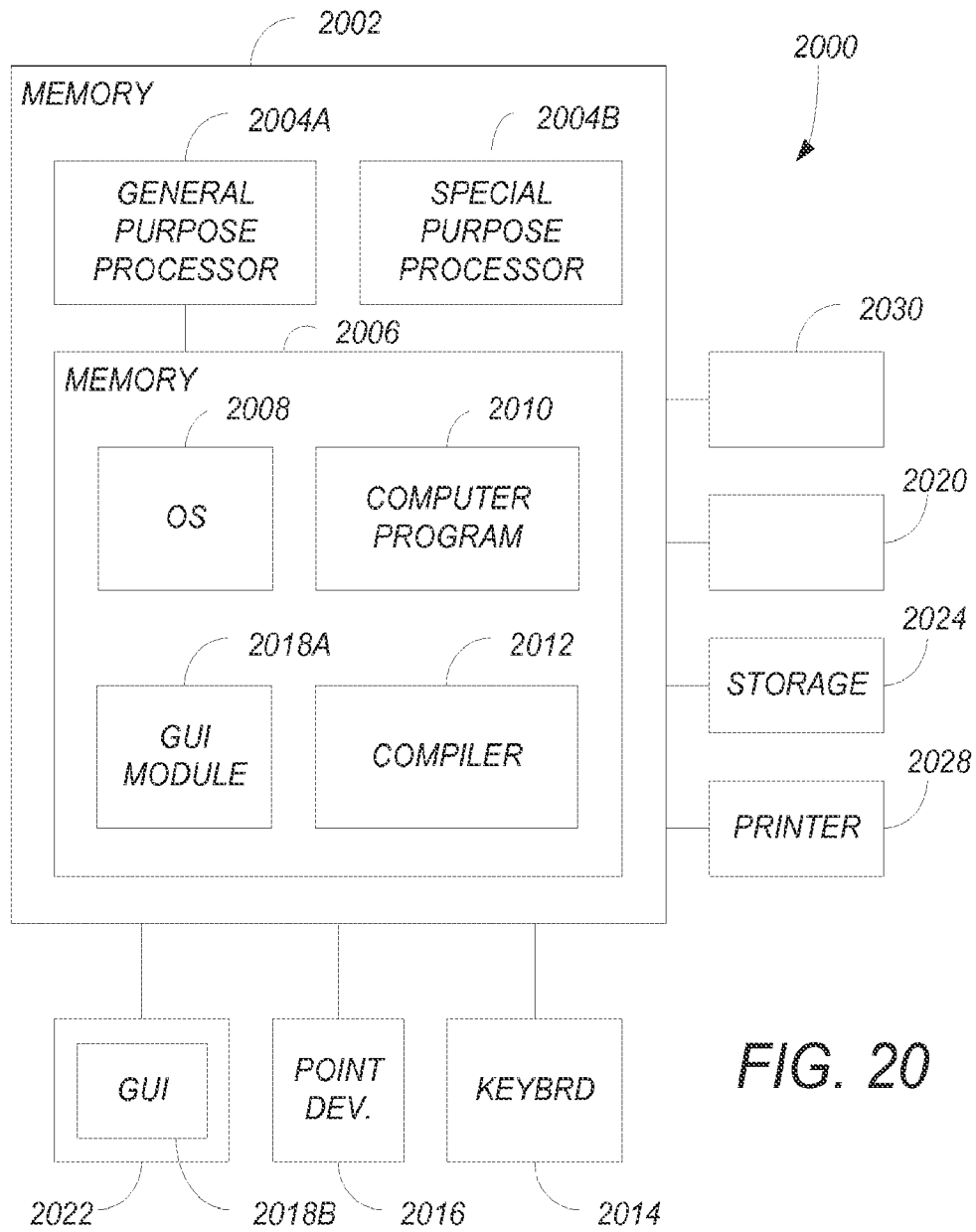


FIG. 20

