



(51) International Patent Classification:

H04N 19/119 (2014.01) H04N 19/129 (2014.01)

H04N 19/70 (2014.01) H04N 19/186 (2014.01)

H04N 19/60 (2014.01) H04N 19/176 (2014.01)

(21) International Application Number:

PCT/US2020/035075

(22) International Filing Date:

29 May 2020 (29.05.2020)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

62/854,949 30 May 2019 (30.05.2019) US

(71) Applicant: **BEIJING DAJIA INTERNET INFORMATION TECHNOLOGY CO., LTD.** [CN/CN]; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100089 (CN).

(72) Inventor; and

(71) Applicant: **WANG, Xianglin** [US/US]; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100089 (CN).

(72) Inventors: **CHEN, Yi-Wen**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100089 (CN). **MA, Tsung-Chuan**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100089 (CN). **XIU, Xiaoyu**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100089 (CN). **JHU, Hong-Jheng**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100089 (CN). **YE, Shuiming**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100089 (CN).

(74) Agent: **SUN, Yalei** et al.; Morgan Lewis & Bockius LLP, 1400 Page Mill Road, Palo Alto, CA 94304 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,

(54) Title: IMPROVING LOSSLESS CODING EFFICIENCY IN VIDEO CODING

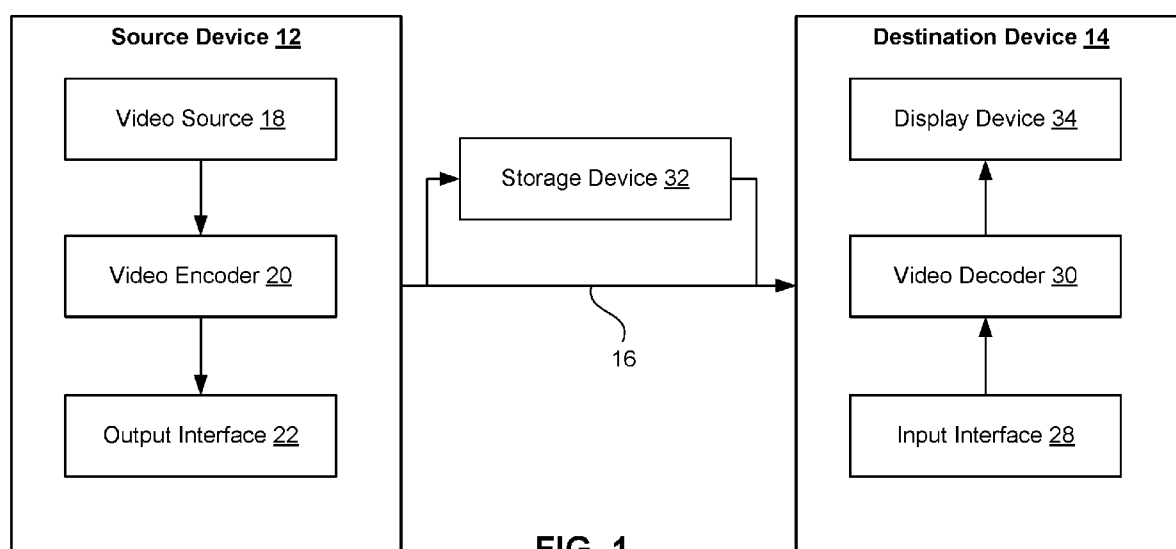


FIG. 1

(57) Abstract: An electronic apparatus performs a method of decoding video data by receiving, from a video bitstream having a hierarchical structure, first indication associated with a first partition level of the hierarchical structure; in accordance with a determination that the first indication indicates that a lossless mode is enabled at the first partition level: configuring one or more coding tools in accordance with the lossless mode; and decoding coding blocks at or below the first partition level using the configured one or more coding tools.

SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR,
TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

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

Published:

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

IMPROVING LOSSLESS CODING EFFICIENCY IN VIDEO CODING

TECHNICAL FIELD

[0001] The present application generally relates to video data encoding and decoding, and in particular, to systems and methods of improving lossless coding efficiency in video coding.

BACKGROUND

[0002] Digital video is supported by a variety of electronic devices, such as digital televisions, laptop or desktop computers, tablet computers, digital cameras, digital recording devices, digital media players, video gaming consoles, smart phones, video teleconferencing devices, video streaming devices, etc. The electronic devices transmit, receive, encode, decode, and/or store digital video data by implementing video compression/decompression standards as defined by MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Part 10, Advanced Video Coding (AVC), High Efficiency Video Coding (HEVC), and Versatile Video Coding (VVC) standard. Video compression typically includes performing spatial (intra frame) prediction and/or temporal (inter frame) prediction to reduce or remove redundancy inherent in the video data. For block-based video coding, a video frame is partitioned into one or more slices, each slice having multiple video blocks, which may also be referred to as coding tree units (CTUs). Each CTU may contain one coding unit (CU) or recursively split into smaller CUs until the predefined minimum CU size is reached. Each CU (also named leaf CU) contains one or multiple transform units (TUs) and each CU also contains one or multiple prediction units (PUs). Each CU can be coded in either intra, inter or IBC modes. Video blocks in an intra coded (I) slice of a video frame are encoded using spatial prediction with respect to reference samples in neighbor blocks within the same video frame. Video blocks in an inter coded (P or B) slice of a video frame may use spatial prediction with respect to reference samples in neighbor blocks within the same video frame or temporal prediction with respect to reference samples in other previous and/or future reference video frames.

[0003] Spatial or temporal prediction based on a reference block that has been previously encoded, e.g., a neighbor block, results in a predictive block for a current video block to be coded. The process of finding the reference block may be accomplished by block matching algorithm. Residual data representing pixel differences between the current block to be coded and the predictive block is referred to as a residual block or prediction errors. An inter-coded block is encoded according to a motion vector that points to a reference block in a reference frame forming the predictive block, and the residual block. The process of determining the motion vector is typically referred to as motion estimation. An intra coded block is encoded according to an intra prediction mode and the residual block. For further compression, the residual block is transformed from the pixel domain to a transform domain, e.g., frequency domain, resulting in residual transform coefficients, which may then be quantized. The quantized transform coefficients, initially arranged in a two-dimensional array, may be scanned to produce a one-dimensional vector of transform coefficients, and then entropy encoded into a video bitstream to achieve even more compression.

[0004] The encoded video bitstream is then saved in a computer-readable storage medium (e.g., flash memory) to be accessed by another electronic device with digital video capability or directly transmitted to the electronic device wired or wirelessly. The electronic device then performs video decompression (which is an opposite process to the video compression described above) by, e.g., parsing the encoded video bitstream to obtain syntax elements from the bitstream and reconstructing the digital video data to its original format from the encoded video bitstream based at least in part on the syntax elements obtained from the bitstream, and renders the reconstructed digital video data on a display of the electronic device.

[0005] With digital video quality going from high definition to 4Kx2K or even 8Kx4K, the amount of video data to be encoded/decoded grows exponentially. It is a constant challenge in terms of how the video data can be encoded/decoded more efficiently while maintaining the image quality of the decoded video data.

SUMMARY

[0006] The present application describes implementations related to video data encoding and decoding and, more particularly, to system and method of improving the lossless coding efficiency during video coding.

[0007] According to a first aspect of the present application, a method of decoding video data is performed at an electronic apparatus by receiving, from a video bitstream having a hierarchical structure, first indication associated with a first partition level of the hierarchical structure; in accordance with a determination that the first indication indicates that a lossless mode is enabled at the first partition level: configuring one or more coding tools in accordance with the lossless mode; and decoding coding blocks at or below the first partition level using the configured one or more coding tools.

[0008] According to a second aspect of the present application, an electronic apparatus includes one or more processors, memory and a plurality of programs stored in the memory. The programs, when executed by the one or more processors, cause the electronic apparatus to perform operations as described above.

[0009] According to a third aspect of the present application, a non-transitory computer-readable storage medium stores a plurality of programs for execution by an electronic apparatus having one or more processors. The programs, when executed by the one or more processors, cause the electronic apparatus to perform operations as described above.

BRIEF DESCRIPTION OF DRAWINGS

[0010] The accompanying drawings, which are included to provide a further understanding of the implementations and are incorporated herein and constitute a part of the specification, illustrate the described implementations and together with the description serve to explain the underlying principles. Like reference numerals refer to corresponding parts.

[0011] FIG. 1 is a block diagram illustrating an exemplary video encoding and decoding system in accordance with some implementations of the present disclosure.

[0012] FIG. 2 is a block diagram illustrating an exemplary video encoder in accordance with some implementations of the present disclosure.

[0013] FIG. 3 is a block diagram illustrating an exemplary video decoder in accordance with some implementations of the present disclosure.

[0014] FIGS. 4A through 4E are block diagrams illustrating how a frame is recursively partitioned into multiple video blocks of different sizes and shapes in accordance with some implementations of the present disclosure.

[0015] FIGS. 5A and 5B are block diagrams illustrating exemplary video frame partitioning schemes in accordance with some implementations of the present disclosure.

[0016] FIG. 6 is a flowchart illustrating an exemplary process by which a video coder implements the techniques of improving lossless coding efficiency in accordance with some implementations of the present disclosure.

DETAILED DESCRIPTION

[0017] Reference will now be made in detail to specific implementations, examples of which are illustrated in the accompanying drawings. In the following detailed description, numerous non-limiting specific details are set forth in order to assist in understanding the subject matter presented herein. But it will be apparent to one of ordinary skill in the art that various alternatives may be used without departing from the scope of claims and the subject matter may be practiced without these specific details. For example, it will be apparent to one of ordinary skill in the art that the subject matter presented herein can be implemented on many types of electronic devices with digital video capabilities.

[0018] FIG. 1 is a block diagram illustrating an exemplary system 10 for encoding and decoding video blocks in parallel in accordance with some implementations of the present disclosure. As shown in FIG. 1, system 10 includes a source device 12 that generates and encodes video data to be decoded subsequently by a destination device 14. Source device 12 and destination device 14 may comprise any of a wide variety of electronic devices,

including desktop or laptop computers, tablet computers, smartphones, set-top boxes, digital televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or the like. In some implementations, source device 12 and destination device 14 are equipped with wireless communication capabilities.

[0019] In some implementations, destination device 14 may receive the encoded video data to be decoded via a link 16. Link 16 may comprise any type of communication medium or device capable of moving the encoded video data from source device 12 to destination device 14. In one example, link 16 may comprise a communication medium to enable source device 12 to transmit the encoded video data directly to destination device 14 in real-time. The encoded video data may be modulated according to a communication standard, such as a wireless communication protocol, and transmitted to destination device 14. The communication medium may comprise any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 12 to destination device 14.

[0020] In some other implementations, the encoded video data may be transmitted from output interface 22 to a storage device 32. Subsequently, the encoded video data in storage device 32 may be accessed by destination device 14 via input interface 28. Storage device 32 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, DVDs, CD-ROMs, flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data. In a further example, storage device 32 may correspond to a file server or another intermediate storage device that may hold the encoded video data generated by source device 12. Destination device 14 may access the stored video data from storage device 32 via streaming or downloading. The file server may be any type of computer capable of storing encoded

video data and transmitting the encoded video data to destination device 14. Exemplary file servers include a web server (e.g., for a website), an FTP server, network attached storage (NAS) devices, or a local disk drive. Destination device 14 may access the encoded video data through any standard data connection, including a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., DSL, cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on a file server. The transmission of encoded video data from storage device 32 may be a streaming transmission, a download transmission, or a combination of both.

[0021] As shown in FIG. 1, source device 12 includes a video source 18, a video encoder 20 and an output interface 22. Video source 18 may include a source such as a video capture device, e.g., a video camera, a video archive containing previously captured video, a video feed interface to receive video from a video content provider, and/or a computer graphics system for generating computer graphics data as the source video, or a combination of such sources. As one example, if video source 18 is a video camera of a security surveillance system, source device 12 and destination device 14 may form camera phones or video phones. However, the implementations described in the present application may be applicable to video coding in general, and may be applied to wireless and/or wired applications.

[0022] The captured, pre-captured, or computer-generated video may be encoded by video encoder 20. The encoded video data may be transmitted directly to destination device 14 via output interface 22 of source device 12. The encoded video data may also be stored onto storage device 32 for later access by destination device 14 or other devices, for decoding and/or playback. Output interface 22 may further include a modem and/or a transmitter.

[0023] Destination device 14 includes an input interface 28, a video decoder 30, and a display device 34. Input interface 28 may include a receiver and/or a modem and receive the encoded video data over link 16. The encoded video data communicated over link 16, or provided on storage device 32, may include a variety of syntax elements generated by video

encoder 20 for use by video decoder 30 in decoding the video data. Such syntax elements may be included within the encoded video data transmitted on a communication medium, stored on a storage medium, or stored a file server.

[0024] In some implementations, destination device 14 may include a display device 34, which can be an integrated display device and an external display device that is configured to communicate with destination device 14. Display device 34 displays the decoded video data to a user, and may comprise any of a variety of display devices such as a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[0025] Video encoder 20 and video decoder 30 may operate according to proprietary or industry standards, such as VVC, HEVC, MPEG-4, Part 10, Advanced Video Coding (AVC), or extensions of such standards. It should be understood that the present application is not limited to a specific video coding/decoding standard and may be applicable to other video coding/decoding standards. It is generally contemplated that video encoder 20 of source device 12 may be configured to encode video data according to any of these current or future standards. Similarly, it is also generally contemplated that video decoder 30 of destination device 14 may be configured to decode video data according to any of these current or future standards.

[0026] Video encoder 20 and video decoder 30 each may be implemented as any of a variety of suitable encoder circuitry, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When implemented partially in software, an electronic device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the video coding/decoding operations disclosed in the present disclosure. Each of video encoder 20 and video decoder 30 may be

included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device.

[0027] FIG. 2 is a block diagram illustrating an exemplary video encoder 20 in accordance with some implementations described in the present application. Video encoder 20 may perform intra and inter predictive coding of video blocks within video frames. Intra predictive coding relies on spatial prediction to reduce or remove spatial redundancy in video data within a given video frame or picture. Inter predictive coding relies on temporal prediction to reduce or remove temporal redundancy in video data within adjacent video frames or pictures of a video sequence.

[0028] As shown in FIG. 2, video encoder 20 includes video data memory 40, prediction processing unit 41, decoded picture buffer (DPB) 64, summer 50, transform processing unit 52, quantization unit 54, and entropy encoding unit 56. Prediction processing unit 41 further includes motion estimation unit 42, motion compensation unit 44, partition unit 45, intra prediction processing unit 46, and intra block copy (BC) unit 48. In some implementations, video encoder 20 also includes inverse quantization unit 58, inverse transform processing unit 60, and summer 62 for video block reconstruction. A deblocking filter (not shown) may be positioned between summer 62 and DPB 64 to filter block boundaries to remove blockiness artifacts from reconstructed video. An in loop filter (not shown) may also be used in addition to the deblocking filter to filter the output of summer 62. Video encoder 20 may take the form of a fixed or programmable hardware unit or may be divided among one or more of the illustrated fixed or programmable hardware units.

[0029] Video data memory 40 may store video data to be encoded by the components of video encoder 20. The video data in video data memory 40 may be obtained, for example, from video source 18. DPB 64 is a buffer that stores reference video data for use in encoding video data by video encoder 20 (e.g., in intra or inter predictive coding modes). Video data memory 40 and DPB 64 may be formed by any of a variety of memory devices. In various

examples, video data memory 40 may be on-chip with other components of video encoder 20, or off-chip relative to those components.

[0030] As shown in FIG. 2, after receiving video data, partition unit 45 within prediction processing unit 41 partitions the video data into video blocks. This partitioning may also include partitioning a video frame into slices, tiles, or other larger coding units (CUs) according to a predefined splitting structures such as quad-tree structure associated with the video data. The video frame may be divided into multiple video blocks (or sets of video blocks referred to as tiles, each tile being defined as a rectangular region of a picture including a sequence of coding tree units). A tile group contains a number of tiles of a picture. Prediction processing unit 41 may select one of a plurality of possible predictive coding modes, such as one of a plurality of intra predictive coding modes or one of a plurality of inter predictive coding modes, for the current video block based on error results (e.g., coding rate and the level of distortion). Prediction processing unit 41 may provide the resulting intra or inter prediction coded block to summer 50 to generate a residual block and to summer 62 to reconstruct the encoded block for use as part of a reference frame subsequently. Prediction processing unit 41 also provides syntax elements, such as motion vectors, intra-mode indicators, partition information, and other such syntax information, to entropy encoding unit 56.

[0031] In order to select an appropriate intra predictive coding mode for the current video block, intra prediction processing unit 46 within prediction processing unit 41 may perform intra predictive coding of the current video block relative to one or more neighbor blocks in the same frame as the current block to be coded to provide spatial prediction. Motion estimation unit 42 and motion compensation unit 44 within prediction processing unit 41 perform inter predictive coding of the current video block relative to one or more predictive blocks in one or more reference frames to provide temporal prediction. Video encoder 20 may perform multiple coding passes, e.g., to select an appropriate coding mode for each block of video data.

[0032] In some implementations, motion estimation unit 42 determines the inter prediction mode for a current video frame by generating a motion vector, which indicates the displacement of a prediction unit (PU) of a video block within the current video frame relative to a predictive block within a reference video frame, according to a predetermined pattern within a sequence of video frames. Motion estimation, performed by motion estimation unit 42, is the process of generating motion vectors, which estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a PU of a video block within a current video frame or picture relative to a predictive block within a reference frame (or other coded unit) relative to the current block being coded within the current frame (or other coded unit). The predetermined pattern may designate video frames in the sequence as P frames or B frames. Intra BC unit 48 may determine vectors, e.g., block vectors, for intra BC coding in a manner similar to the determination of motion vectors by motion estimation unit 42 for inter prediction, or may utilize motion estimation unit 42 to determine the block vector.

[0033] A predictive block is a block of a reference frame that is deemed as closely matching the PU of the video block to be coded in terms of pixel difference, which may be determined by sum of absolute difference (SAD), sum of square difference (SSD), or other difference metrics. In some implementations, video encoder 20 may calculate values for sub-integer pixel positions of reference frames stored in DPB 64. For example, video encoder 20 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference frame. Therefore, motion estimation unit 42 may perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision.

[0034] Motion estimation unit 42 calculates a motion vector for a PU of a video block in an inter prediction coded frame by comparing the position of the PU to the position of a predictive block of a reference frame selected from a first reference frame list (List 0) or a second reference frame list (List 1), each of which identifies one or more reference frames

stored in DPB 64. Motion estimation unit 42 sends the calculated motion vector to motion compensation unit 44 and then to entropy encoding unit 56.

[0035] Motion compensation, performed by motion compensation unit 44, may involve fetching or generating the predictive block based on the motion vector determined by motion estimation unit 42. Upon receiving the motion vector for the PU of the current video block, motion compensation unit 44 may locate a predictive block to which the motion vector points in one of the reference frame lists, retrieve the predictive block from DPB 64, and forward the predictive block to summer 50. Summer 50 then forms a residual video block of pixel difference values by subtracting pixel values of the predictive block provided by motion compensation unit 44 from the pixel values of the current video block being coded. The pixel difference values forming the residual video block may include luma or chroma difference components or both. Motion compensation unit 44 may also generate syntax elements associated with the video blocks of a video frame for use by video decoder 30 in decoding the video blocks of the video frame. The syntax elements may include, for example, syntax elements defining the motion vector used to identify the predictive block, any flags indicating the prediction mode, or any other syntax information described herein. Note that motion estimation unit 42 and motion compensation unit 44 may be highly integrated, but are illustrated separately for conceptual purposes.

[0036] In some implementations, intra BC unit 48 may generate vectors and fetch predictive blocks in a manner similar to that described above in connection with motion estimation unit 42 and motion compensation unit 44, but with the predictive blocks being in the same frame as the current block being coded and with the vectors being referred to as block vectors as opposed to motion vectors. In particular, intra BC unit 48 may determine an intra-prediction mode to use to encode a current block. In some examples, intra BC unit 48 may encode a current block using various intra-prediction modes, e.g., during separate encoding passes, and test their performance through rate-distortion analysis. Next, intra BC unit 48 may select, among the various tested intra-prediction modes, an appropriate intra-

prediction mode to use and generate an intra-mode indicator accordingly. For example, intra BC unit 48 may calculate rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and select the intra-prediction mode having the best rate-distortion characteristics among the tested modes as the appropriate intra-prediction mode to use. Rate-distortion analysis generally determines an amount of distortion (or error) between an encoded block and an original, unencoded block that was encoded to produce the encoded block, as well as a bitrate (i.e., a number of bits) used to produce the encoded block. Intra BC unit 48 may calculate ratios from the distortions and rates for the various encoded blocks to determine which intra-prediction mode exhibits the best rate-distortion value for the block.

[0037] In other examples, intra BC unit 48 may use motion estimation unit 42 and motion compensation unit 44, in whole or in part, to perform such functions for Intra BC prediction according to the implementations described herein. In either case, for Intra block copy, a predictive block may be a block that is deemed as closely matching the block to be coded, in terms of pixel difference, which may be determined by sum of absolute difference (SAD), sum of squared difference (SSD), or other difference metrics, and identification of the predictive block may include calculation of values for sub-integer pixel positions.

[0038] Whether the predictive block is from the same frame according to intra prediction, or a different frame according to inter prediction, video encoder 20 may form a residual video block by subtracting pixel values of the predictive block from the pixel values of the current video block being coded, forming pixel difference values. The pixel difference values forming the residual video block may include both luma and chroma component differences.

[0039] Intra prediction processing unit 46 may intra-predict a current video block, as an alternative to the inter-prediction performed by motion estimation unit 42 and motion compensation unit 44, or the intra block copy prediction performed by intra BC unit 48, as described above. In particular, intra prediction processing unit 46 may determine an intra prediction mode to use to encode a current block. To do so, intra prediction processing unit

46 may encode a current block using various intra prediction modes, e.g., during separate encoding passes, and intra prediction processing unit 46 (or a mode select unit, in some examples) may select an appropriate intra prediction mode to use from the tested intra prediction modes. Intra prediction processing unit 46 may provide information indicative of the selected intra-prediction mode for the block to entropy encoding unit 56. Entropy encoding unit 56 may encode the information indicating the selected intra-prediction mode in the bitstream.

[0040] After prediction processing unit 41 determines the predictive block for the current video block via either inter prediction or intra prediction, summer 50 forms a residual video block by subtracting the predictive block from the current video block. The residual video data in the residual block may be included in one or more transform units (TUs) and is provided to transform processing unit 52. Transform processing unit 52 transforms the residual video data into residual transform coefficients using a transform, such as a discrete cosine transform (DCT) or a conceptually similar transform.

[0041] Transform processing unit 52 may send the resulting transform coefficients to quantization unit 54. Quantization unit 54 quantizes the transform coefficients to further reduce bit rate. The quantization process may also reduce the bit depth associated with some or all of the coefficients. The degree of quantization may be modified by adjusting a quantization parameter. In some examples, quantization unit 54 may then perform a scan of a matrix including the quantized transform coefficients. Alternatively, entropy encoding unit 56 may perform the scan.

[0042] Following quantization, entropy encoding unit 56 entropy encodes the quantized transform coefficients into a video bitstream using, e.g., context adaptive variable length coding (CAVLC), context adaptive binary arithmetic coding (CABAC), syntax-based context-adaptive binary arithmetic coding (SBAC), probability interval partitioning entropy (PIPE) coding or another entropy encoding methodology or technique. The encoded bitstream may then be transmitted to video decoder 30, or archived in storage device 32 for later

transmission to or retrieval by video decoder 30. Entropy encoding unit 56 may also entropy encode the motion vectors and the other syntax elements for the current video frame being coded.

[0043] Inverse quantization unit 58 and inverse transform processing unit 60 apply inverse quantization and inverse transformation, respectively, to reconstruct the residual video block in the pixel domain for generating a reference block for prediction of other video blocks. As noted above, motion compensation unit 44 may generate a motion compensated predictive block from one or more reference blocks of the frames stored in DPB 64. Motion compensation unit 44 may also apply one or more interpolation filters to the predictive block to calculate sub-integer pixel values for use in motion estimation.

[0044] Summer 62 adds the reconstructed residual block to the motion compensated predictive block produced by motion compensation unit 44 to produce a reference block for storage in DPB 64. The reference block may then be used by intra BC unit 48, motion estimation unit 42 and motion compensation unit 44 as a predictive block to inter predict another video block in a subsequent video frame.

[0045] FIG. 3 is a block diagram illustrating an exemplary video decoder 30 in accordance with some implementations of the present application. Video decoder 30 includes video data memory 79, entropy decoding unit 80, prediction processing unit 81, inverse quantization unit 86, inverse transform processing unit 88, summer 90, and DPB 92. Prediction processing unit 81 further includes motion compensation unit 82, intra prediction processing unit 84, and intra BC unit 85. Video decoder 30 may perform a decoding process generally reciprocal to the encoding process described above with respect to video encoder 20 in connection with FIG. 2. For example, motion compensation unit 82 may generate prediction data based on motion vectors received from entropy decoding unit 80, while intra-prediction unit 84 may generate prediction data based on intra-prediction mode indicators received from entropy decoding unit 80.

[0046] In some examples, a unit of video decoder 30 may be tasked to perform the implementations of the present application. Also, in some examples, the implementations of the present disclosure may be divided among one or more of the units of video decoder 30. For example, intra BC unit 85 may perform the implementations of the present application, alone, or in combination with other units of video decoder 30, such as motion compensation unit 82, intra prediction processing unit 84, and entropy decoding unit 80. In some examples, video decoder 30 may not include intra BC unit 85 and the functionality of intra BC unit 85 may be performed by other components of prediction processing unit 81, such as motion compensation unit 82.

[0047] Video data memory 79 may store video data, such as an encoded video bitstream, to be decoded by the other components of video decoder 30. The video data stored in video data memory 79 may be obtained, for example, from storage device 32, from a local video source, such as a camera, via wired or wireless network communication of video data, or by accessing physical data storage media (e.g., a flash drive or hard disk). Video data memory 79 may include a coded picture buffer (CPB) that stores encoded video data from an encoded video bitstream. Decoded picture buffer (DPB) 92 of video decoder 30 stores reference video data for use in decoding video data by video decoder 30 (e.g., in intra or inter predictive coding modes). Video data memory 79 and DPB 92 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magneto-resistive RAM (MRAM), resistive RAM (RRAM), or other types of memory devices. For illustrative purpose, video data memory 79 and DPB 92 are depicted as two distinct components of video decoder 30 in FIG. 3. But it will be apparent to one skilled in the art that video data memory 79 and DPB 92 may be provided by the same memory device or separate memory devices. In some examples, video data memory 79 may be on-chip with other components of video decoder 30, or off-chip relative to those components.

[0048] During the decoding process, video decoder 30 receives an encoded video bitstream that represents video blocks of an encoded video frame and associated syntax elements. Video decoder 30 may receive the syntax elements at the video frame level and/or the video block level. Entropy decoding unit 80 of video decoder 30 entropy decodes the bitstream to generate quantized coefficients, motion vectors or intra-prediction mode indicators, and other syntax elements. Entropy decoding unit 80 then forwards the motion vectors and other syntax elements to prediction processing unit 81.

[0049] When the video frame is coded as an intra predictive coded (I) frame or for intra coded predictive blocks in other types of frames, intra prediction processing unit 84 of prediction processing unit 81 may generate prediction data for a video block of the current video frame based on a signaled intra prediction mode and reference data from previously decoded blocks of the current frame.

[0050] When the video frame is coded as an inter-predictive coded (i.e., B or P) frame, motion compensation unit 82 of prediction processing unit 81 produces one or more predictive blocks for a video block of the current video frame based on the motion vectors and other syntax elements received from entropy decoding unit 80. Each of the predictive blocks may be produced from a reference frame within one of the reference frame lists. Video decoder 30 may construct the reference frame lists, List 0 and List 1, using default construction techniques based on reference frames stored in DPB 92.

[0051] In some examples, when the video block is coded according to the intra BC mode described herein, intra BC unit 85 of prediction processing unit 81 produces predictive blocks for the current video block based on block vectors and other syntax elements received from entropy decoding unit 80. The predictive blocks may be within a reconstructed region of the same picture as the current video block defined by video encoder 20.

[0052] Motion compensation unit 82 and/or intra BC unit 85 determines prediction information for a video block of the current video frame by parsing the motion vectors and other syntax elements, and then uses the prediction information to produce the predictive

blocks for the current video block being decoded. For example, motion compensation unit 82 uses some of the received syntax elements to determine a prediction mode (e.g., intra or inter prediction) used to code video blocks of the video frame, an inter prediction frame type (e.g., B or P), construction information for one or more of the reference frame lists for the frame, motion vectors for each inter predictive encoded video block of the frame, inter prediction status for each inter predictive coded video block of the frame, and other information to decode the video blocks in the current video frame.

[0053] Similarly, intra BC unit 85 may use some of the received syntax elements, e.g., a flag, to determine that the current video block was predicted using the intra BC mode, construction information of which video blocks of the frame are within the reconstructed region and should be stored in DPB 92, block vectors for each intra BC predicted video block of the frame, intra BC prediction status for each intra BC predicted video block of the frame, and other information to decode the video blocks in the current video frame.

[0054] Motion compensation unit 82 may also perform interpolation using the interpolation filters as used by video encoder 20 during encoding of the video blocks to calculate interpolated values for sub-integer pixels of reference blocks. In this case, motion compensation unit 82 may determine the interpolation filters used by video encoder 20 from the received syntax elements and use the interpolation filters to produce predictive blocks.

[0055] Inverse quantization unit 86 inverse quantizes the quantized transform coefficients provided in the bitstream and entropy decoded by entropy decoding unit 80 using the same quantization parameter calculated by video encoder 20 for each video block in the video frame to determine a degree of quantization. Inverse transform processing unit 88 applies an inverse transform, e.g., an inverse DCT, an inverse integer transform, or a conceptually similar inverse transform process, to the transform coefficients in order to reconstruct the residual blocks in the pixel domain.

[0056] After motion compensation unit 82 or intra BC unit 85 generates the predictive block for the current video block based on the vectors and other syntax elements, summer 90

reconstructs decoded video block for the current video block by summing the residual block from inverse transform processing unit 88 and a corresponding predictive block generated by motion compensation unit 82 and intra BC unit 85. An in-loop filter (not pictured) may be positioned between summer 90 and DPB 92 to further process the decoded video block. The decoded video blocks in a given frame are then stored in DPB 92, which stores reference frames used for subsequent motion compensation of next video blocks. DPB 92, or a memory device separate from DPB 92, may also store decoded video for later presentation on a display device, such as display device 34 of FIG. 1.

[0057] In a typical video coding process, a video sequence typically includes an ordered set of frames or pictures. Each frame may include three sample arrays, denoted SL, SCb, and SCr. SL is a two-dimensional array of luma samples. SCb is a two-dimensional array of Cb chroma samples. SCr is a two-dimensional array of Cr chroma samples. In other instances, a frame may be monochrome and therefore includes only one two-dimensional array of luma samples.

[0058] As shown in FIG. 4A, video encoder 20 (or more specifically partition unit 45) generates an encoded representation of a frame by first partitioning the frame into a set of coding tree units (CTUs). A video frame may include an integer number of CTUs ordered consecutively in a raster scan order from left to right and from top to bottom. Each CTU is a largest logical coding unit and the width and height of the CTU are signaled by the video encoder 20 in a sequence parameter set, such that all the CTUs in a video sequence have the same size being one of 128×128 , 64×64 , 32×32 , and 16×16 . But it should be noted that the present application is not necessarily limited to a particular size. As shown in FIG. 4B, each CTU may comprise one coding tree block (CTB) of luma samples, two corresponding coding tree blocks of chroma samples, and syntax elements used to code the samples of the coding tree blocks. The syntax elements describe properties of different types of units of a coded block of pixels and how the video sequence can be reconstructed at the video decoder 30, including inter or intra prediction, intra prediction mode, motion vectors, and other

parameters. In monochrome pictures or pictures having three separate color planes, a CTU may comprise a single coding tree block and syntax elements used to code the samples of the coding tree block. A coding tree block may be an $N \times N$ block of samples.

[0059] To achieve a better performance, video encoder 20 may recursively perform tree partitioning such as binary-tree partitioning, ternary-tree partitioning, quad-tree partitioning or a combination of both on the coding tree blocks of the CTU and divide the CTU into smaller coding units (CUs). As depicted in FIG. 4C, the 64×64 CTU 400 is first divided into four smaller CU, each having a block size of 32×32 . Among the four smaller CUs, CU 410 and CU 420 are each partitioned into four CUs of 16×16 by block size. The two 16×16 CUs 430 and 440 are each further divided into four CUs of 8×8 by block size. FIG. 4D depicts a quad-tree data structure illustrating the end result of the partition process of the CTU 400 as depicted in FIG. 4C, each leaf node of the quad-tree corresponding to one CU of a respective size ranging from 32×32 to 8×8 . Like the CTU depicted in FIG. 4B, each CU may comprise a coding block (CB) of luma samples and two corresponding coding blocks of chroma samples of a frame of the same size, and syntax elements used to code the samples of the coding blocks. In monochrome pictures or pictures having three separate color planes, a CU may comprise a single coding block and syntax structures used to code the samples of the coding block. It should be noted that the quad-tree partitioning depicted in FIGS. 4C and 4D is only for illustrative purposes and one CTU can be split into CUs to adapt to varying local characteristics based on quad/ternary/binary-tree partitions. In the multi-type tree structure, one CTU is partitioned by a quad-tree structure and each quad-tree leaf CU can be further partitioned by a binary and ternary tree structure. As shown in FIG. 4E, there are five possible partitioning types of a coding block having a width W and a height H , i.e., quaternary partitioning, horizontal binary partitioning, vertical binary partitioning, horizontal ternary partitioning, and vertical ternary partitioning.

[0060] In some implementations, video encoder 20 may further partition a coding block of a CU into one or more $M \times N$ prediction blocks (PB). A prediction block is a

rectangular (square or non-square) block of samples on which the same prediction, inter or intra, is applied. A prediction unit (PU) of a CU may comprise a prediction block of luma samples, two corresponding prediction blocks of chroma samples, and syntax elements used to predict the prediction blocks. In monochrome pictures or pictures having three separate color planes, a PU may comprise a single prediction block and syntax structures used to predict the prediction block. Video encoder 20 may generate predictive luma, Cb, and Cr blocks for luma, Cb, and Cr prediction blocks of each PU of the CU.

[0061] Video encoder 20 may use intra prediction or inter prediction to generate the predictive blocks for a PU. If video encoder 20 uses intra prediction to generate the predictive blocks of a PU, video encoder 20 may generate the predictive blocks of the PU based on decoded samples of the frame associated with the PU. If video encoder 20 uses inter prediction to generate the predictive blocks of a PU, video encoder 20 may generate the predictive blocks of the PU based on decoded samples of one or more frames other than the frame associated with the PU.

[0062] After video encoder 20 generates predictive luma, Cb, and Cr blocks for one or more PUs of a CU, video encoder 20 may generate a luma residual block for the CU by subtracting the CU's predictive luma blocks from its original luma coding block such that each sample in the CU's luma residual block indicates a difference between a luma sample in one of the CU's predictive luma blocks and a corresponding sample in the CU's original luma coding block. Similarly, video encoder 20 may generate a Cb residual block and a Cr residual block for the CU, respectively, such that each sample in the CU's Cb residual block indicates a difference between a Cb sample in one of the CU's predictive Cb blocks and a corresponding sample in the CU's original Cb coding block and each sample in the CU's Cr residual block may indicate a difference between a Cr sample in one of the CU's predictive Cr blocks and a corresponding sample in the CU's original Cr coding block.

[0063] Furthermore, as illustrated in FIG. 4C, video encoder 20 may use quad-tree partitioning to decompose the luma, Cb, and Cr residual blocks of a CU into one or more

luma, Cb, and Cr transform blocks. A transform block is a rectangular (square or non-square) block of samples on which the same transform is applied. A transform unit (TU) of a CU may comprise a transform block of luma samples, two corresponding transform blocks of chroma samples, and syntax elements used to transform the transform block samples. Thus, each TU of a CU may be associated with a luma transform block, a Cb transform block, and a Cr transform block. In some examples, the luma transform block associated with the TU may be a sub-block of the CU's luma residual block. The Cb transform block may be a sub-block of the CU's Cb residual block. The Cr transform block may be a sub-block of the CU's Cr residual block. In monochrome pictures or pictures having three separate color planes, a TU may comprise a single transform block and syntax structures used to transform the samples of the transform block.

[0064] Video encoder 20 may apply one or more transforms to a luma transform block of a TU to generate a luma coefficient block for the TU. A coefficient block may be a two-dimensional array of transform coefficients. A transform coefficient may be a scalar quantity. Video encoder 20 may apply one or more transforms to a Cb transform block of a TU to generate a Cb coefficient block for the TU. Video encoder 20 may apply one or more transforms to a Cr transform block of a TU to generate a Cr coefficient block for the TU.

[0065] After generating a coefficient block (e.g., a luma coefficient block, a Cb coefficient block or a Cr coefficient block), video encoder 20 may quantize the coefficient block. Quantization generally refers to a process in which transform coefficients are quantized to possibly reduce the amount of data used to represent the transform coefficients, providing further compression. After video encoder 20 quantizes a coefficient block, video encoder 20 may entropy encode syntax elements indicating the quantized transform coefficients. For example, video encoder 20 may perform Context-Adaptive Binary Arithmetic Coding (CABAC) on the syntax elements indicating the quantized transform coefficients. Finally, video encoder 20 may output a bitstream that includes a sequence of bits

that forms a representation of coded frames and associated data, which is either saved in storage device 32 or transmitted to destination device 14.

[0066] After receiving a bitstream generated by video encoder 20, video decoder 30 may parse the bitstream to obtain syntax elements from the bitstream. Video decoder 30 may reconstruct the frames of the video data based at least in part on the syntax elements obtained from the bitstream. The process of reconstructing the video data is generally reciprocal to the encoding process performed by video encoder 20. For example, video decoder 30 may perform inverse transforms on the coefficient blocks associated with TUs of a current CU to reconstruct residual blocks associated with the TUs of the current CU. Video decoder 30 also reconstructs the coding blocks of the current CU by adding the samples of the predictive blocks for PUs of the current CU to corresponding samples of the transform blocks of the TUs of the current CU. After reconstructing the coding blocks for each CU of a frame, video decoder 30 may reconstruct the frame.

[0067] FIGS. 5A and 5B are block diagrams illustrating exemplary video frame partitioning schemes in accordance with some implementations of the present disclosure.

[0068] FIG. 5A shows an exemplary raster-scan slice partitioning of video frame 502 (e.g., raster-scan slice mode). Video frame 502 is a rectangular region including 12x18 CTUs (e.g., CTU 504a, CTU 504b, etc.) and is partitioned into three slices (e.g., slice 508a, 508b, and 508c in FIG. 5A with different filling patterns). A slice in a raster-scan slice partitioning scheme is defined as a set of continuous tiles (e.g., in the raster-scan order) in a video frame. For example, tile slice 508a includes two tile 506a and tile 506b, slice 508b includes five continuous tiles of video frame 502 in the raster-scan order, and slice 508c includes another five tiles of video frame 502 in the raster-scan order. Each tile is a rectangular region of a plurality of CTUs. As such, video frame 502 includes twelve tiles each including 3x6 CTUs.

[0069] FIG. 5B shows an exemplary rectangular slice partitioning of video frame 510 (e.g., rectangular slice mode). Video frame 510 is a rectangular region including 12x18 CTUs (e.g., CTU 512a, 512b, etc.) and is partitioned into nine slices (e.g., slice 516a, 516b, 516c,

etc. in FIG. 5B with different filling patterns). Unlike the raster-scan slice partitioning shown in FIG. 5A, slices (e.g., slice 516a, 516b, 516c) in video frame 510 are each a rectangular region including a plurality of tiles. For example, slice 516a is a rectangular region including two tiles (tiles 514a and 514b), and slice 516b is a square region including four tiles. In some embodiments, the tiles within each slice are arranged in raster-scan order. Video frame 510 includes twenty-four tiles (e.g., tile 514a, tile 514b, etc.) of equal size, with each tile including 3x3 CTUs.

[0070] In some embodiments, a slice consists of an integer number of complete tiles or an integer number of consecutive complete CTU rows within a tile of a picture. Consequently, each vertical slice boundary is always also a vertical tile boundary. It is possible that a horizontal boundary of a slice is not a tile boundary but consists of horizontal CTU boundaries within a tile; this occurs when a tile is split into multiple rectangular slices, each of which consists of an integer number of consecutive complete CTU rows within the tile. Two modes of slices are supported, namely the raster-scan slice mode (e.g., illustrated in FIG. 5A) and the rectangular slice mode (e.g., illustrated in FIG. 5B). In the raster-scan slice mode, a slice contains a sequence of complete tiles in a tile raster scan of a picture. In the rectangular slice mode, a slice contains either a number of complete tiles that collectively form a rectangular region of the picture or a number of consecutive complete CTU rows of one tile that collectively form a rectangular region of the picture. Tiles within a rectangular slice are scanned in tile raster scan order within the rectangular region corresponding to that slice.

[0071] Therefore, in either raster-scan slice partitioning scheme or rectangular slice partitioning scheme, a video frame is partitioned in a nested manner at different levels. For example, a video frame may be partitioned as a plurality of slices, a plurality of tiles within each slice, and a plurality of CTUs within each tile. In some embodiments, each partition is associated with a parameter set including instructions for coding the corresponding partition. For example, the video sequence including video frame 502 (or video frame 510) may be

associated with a sequence parameter set (SPS), and each video frame 502 (or video frame 510) may be further associated with a picture parameter set (PPS). Each of the slice (e.g., slice 508a or slice 516a) may be associated with a respective slice parameter set (e.g., within slice header), and each of the tiles (e.g., tile 506a or tile 514a) may be associated with a respective tile parameter set (e.g., within the tile header), each of the CTUs (e.g., CTU 504a or CTU 512a) may be associated with a respective CTU parameter set (CTU header), and each of the CUs may be associated with a respective CU parameter set (CU header). The respective parameter set for a partition may include information for coding the respective partition in (i) a lossy mode or (ii) a lossless mode (in which operations such as transform/inverse-transform, quantization/de-quantization, and in-loop filtering are being skipped).

[0072] Traditionally (e.g., in HEVC), lossless coding is enabled by a two-flag signaling mechanism. A first flag (e.g., “transquant_bypass_enabled_flag”) is signaled only at a video frame level (e.g., included in the PPS associated with the video frame) to indicate whether lossless mode is enabled in the video frame, and a second flag (e.g., “cu_transquant_bypass_flag”) is signaled only at a CU level to indicate whether a respective CU is coded in lossless or lossy mode. For example, if transquant_bypass_enabled_flag is set to a Boolean value “1” for a video frame, then a lossless mode is enabled for the video frame and further signaling by the second flag at CU levels is needed. On the other hand, if transquant_bypass_enabled_flag is set to a Boolean value “0” for the video frame, then a lossless mode is not enabled and further signaling by the second flag at CU levels is not needed (i.e., all CUs in the video frame are coded in lossy mode). In other words, the first flag, transquant_bypass_enabled_flag, is used to turn on the syntax signaling of the second flag, cu_transquant_bypass_flag. In the case where transquant_bypass_enabled_flag is set to “1” for the video frame, cu_transquant_bypass_flag is required to be signaled for each CU in the video frame to indicate if the respective CU is coded in lossless mode (e.g., cu_transquant_bypass_flag set to “1”) or lossy mode (e.g., cu_transquant_bypass_flag set to “0”).

“0”). In lossless mode, transform/inverse transform, quantization/de-quantization, and in-loop filtering processes are skipped for the respective CUs and the prediction residual values are directly coded by the entropy coding unit 56.

[0073] Such a two-flag signaling mechanism provides greater control granularity for enabling the lossless mode in video coding, but incurs excessive signaling overhead. For example, if the entire video frame or a desired portion of the video frame is to be coded in lossless mode (e.g., in applications that require transferring high definition images such as telemedicine, remote sensing or in applications that require transferring images including high definition portions such as slides in video conference or vehicle plates in traffic surveillance video) with the `transquant_bypass_enabled_flag` set to “1”, a `cu_transquant_bypass_flag` has to be signaled for every CU in that video frame, causing excessive signaling overhead. Furthermore, the two-flag signaling mechanism is unduly rigid as the flags can only be set at the two predefined partition levels (e.g., video frame level and CU level). Therefore, a more flexible signaling mechanism to indicate coding mode (e.g., lossless or lossy) at any partition levels (e.g., video sequence level, video frame level, slice level, tile level, CTU level, or CU level) is highly desired.

[0074] In some embodiments, a first syntax element may be signaled at any partition levels (e.g., video sequence level, video frame level, slice level, tile level, CTU level, or CU level) to turn on syntax signaling of lossless or lossy mode. If the first syntax element is set to “0” at a first partition level, then all CUs at or below the first partition level are coded in lossy mode (e.g., a respective second syntax element for each CU at and below the first partition level is implicitly interpreted to “0” to indicate that lossy mode is used). For example, if the first syntax element is set to “0” in the slice header of slice 516a of FIG. 5B, then all CTUs within slice 516a (e.g., CTU 512a, CTU 512b, etc.) are coded in lossy mode (e.g., further signaling of a respective second syntax element for each CU within slice 516a is no longer needed). On the other hand, if the first syntax element is set to “1” in the slice header of slice 516a of FIG. 5B, then all CTUs at or below the first partition level may be coded in either

lossless or lossy mode and further signaling by respective second syntax elements is needed. For example, if the first syntax element is set to “1” in the slice header of slice 516a of FIG. 5B, further syntax signaling by a respective second syntax element is needed at a lower level such as the tile level. If a respective second syntax element for tile 514a is set to “0,” then all the nine CTUs within tile 514a are coded in lossy mode (e.g., further signaling of a respective second syntax element for each CU within tile 514a is no longer needed). If a respective second syntax element for tile 514b is set to “1,” then further signaling by a respective third syntax element is needed for each CTU within tile 514b. Therefore, syntax signaling can be performed in hierarchical or cascading manner, and the presence of a lower-level syntax signaling is dependent on the value of a higher-level syntax signaling.

[0075] In the embodiment described above, a lossy mode is signaled in a bit-saving manner (e.g., a syntax element “0” at a partition level would cause all lower partition levels to be coded in lossy mode, and no further syntax signaling is needed). In some alternative embodiments, lossless mode, instead of lossy mode, is implicitly signaled based on signaling of lossless mode at a higher partition level. For example, if the first syntax element is set to “1” in the PPS of video frame 510, then all CTUs within the video frame 510 are coded using lossless mode and further signaling by second syntax elements is no longer needed. If the first syntax element is set to “0” in the PPS of video frame 510 (e.g., indicating that not all CUs in video frame 510 are coded in lossless mode), then further signaling by second syntax elements at lower slice levels are needed. The further signaling (e.g., by second, third, fourth, etc. syntax elements) at lower partition levels can be performed in a hierarchical or cascading manner. For example, if the second syntax element is set to “0” in the slice header for slice 516a, then further syntax signaling for all tiles (e.g., tile 514a and tile 514b) within slice 516a is needed (e.g., by respective third syntax elements, which can indicate either lossless mode or lossy mode). If the second syntax element is set to “1” in the slice header for slice 516a, then no further syntax signaling is needed at any partition level within slice 516a, and all CTUs within slice 516a are in a lossless mode.

[0076] Between lossless mode and lossy mode, different coding tools have been developed suitable for different coding modes. For example, under the lossy mode, video coder (video encoder 20 or video decoder 30) need to perform transform/inverse transform to a residual block using various transform-related coding tools including, e.g., a multiple transform selection (MTS) scheme used for residual coding both inter and intra coded blocks, a sub-block transform (SBT), transform skip and secondary transform, etc. But as noted above, these transform-related coding tools will not be used when video coder performs lossless coding because the residual block is directly coded by the entropy coding unit 56. Similarly, other tools used for, e.g., quantization/de-quantization, in-loop filtering, adaptive loop filtering, de-blocking filtering, sample adaptive offset filtering, luma mapping with chroma scaling, intra sub-partitioning, and joint coding of chroma residuals processes should not be available when video coder performs lossless coding. Therefore, in some implementations, those coding tools not required by the lossless coding is disabled when video coder performs lossless coding.

[0077] As a result, no signaling of these coding tools by video encoder 20 is needed and video decoder 30 can assign the syntax elements related to these coding tools based on the signaling or indication of signaling of lossless mode.

[0078] In one example, it is proposed to avoid signalling the control flags of the coding tools which may not be able to losslessly reconstruct the current block for a lossless coded CU (e.g. when the `cu_transquant_bypass_flag` is equal to 1). The coding tools include but not limited to the intra sub-partitioning and joint coding of chroma residual. In other words, when the current block is coded as lossless mode (i.e., the `cu_transquant_bypass_flag` is equal to 1), the `intra_subpartitions_mode_flag` is not signalled.

[0079] In another example, the coding tools that do not efficiently provide coding performance gain for a lossless coded block and their signalling are disabled. Such coding tools include, but are not limited to, decoder side motion vector refinement (DMVR), bi-directional optical flow (BDOF), subblock-based temporal motion vector prediction

(SbTMVP), adaptive motion vector resolution (AMVR), combined inter and intra prediction (CIIP), multiple reference line (MRL) intra prediction and separate block tree for the luma and chroma (or termed as dual_tree coding for luma and chroma components). In yet another example, it is proposed to avoid signalling delta QPs, transform skip, or sign data hiding for a CU coded as lossless mode.

[0080] In some implementations, when width or height of a CU is larger than 64, its TU is implicitly divided into two smaller TUs with half size for the side which is larger than 64. For example, a 128x64 CU corresponds to 2 64x64 TUs and a 128x128 CU corresponds to 4 64x64 TUs. But this implicit TU split is disabled for a lossless coded CU because there is no transform deployed in a lossless coded CU.

[0081] In some implementations, the intra prediction result of a planar mode is refined by a position dependent intra prediction combination (PDPC) method. PDPC is a method that invokes a combination of intra prediction with the un-filtered boundary reference samples and the intra prediction with filtered boundary reference samples. But PDPC may not help the coding efficiency for a lossless coded block and is therefore disabled for a lossless coded block. Moreover, the filtering on the intra reference samples could also be disabled for a lossless coded block.

[0082] FIG. 6 is a flowchart illustrating an exemplary process 600 by which a video coder (e.g., video decoder 30 of FIG. 3) implements the techniques of improving lossless coding efficiency in accordance with some implementations of the present disclosure. In some embodiments, the video coder performs lossless coding by skipping transformation (e.g., the operation performed by inverse transform processing unit 88 of FIG. 3), quantization (e.g., the operation performed by inverse quantization unit 86 of FIG. 3), and in-loop filters (de-block filter, sample adaptive offset, adaptive loop filter, etc.) during a video coding process. As a result, prediction residual values are directly coded by an entropy coding module (e.g., entropy decoding unit 80 of FIG. 3). For convenience, the process 600 is described as being performed by a video coder (e.g., video decoder 30 of FIG. 3).

[0083] In the first step, the video coder receives, from a video bitstream having a hierarchical structure (e.g., on destination device 14 of FIG. 1) corresponding to video data encoding one or more pictures (e.g., a video), first indication associated with a first partition level of the hierarchical structure (605). For example, the one or more pictures may be one video frame or a video sequence and are to be reconstructed by the video coder on the destination device. In some embodiments, the hierarchical structure includes a plurality of group at different partition levels. For example, the one or more pictures may have been divided into a plurality of slices (e.g., slices 508a-508c of FIG. 5A), and each slice may have been further divided into a plurality of tiles (e.g., tile 506a and 506b within slice 508a of FIG. 5A), and each tile may have been divided into a plurality of CTUs (e.g., CTU 504a and 504b within tile 506a of FIG. 5A), and each CTU may have been divided into a plurality of CUs. In such an example, the first partition level may correspond to a video sequence level, the second partition level may correspond to a picture level (one or more pictures within the video sequence), the third partition level may correspond to a slice level, a fourth partition level may correspond to a tile level, a fifth partition level may correspond to a CTU level, and a sixth partition level may correspond to a CU level.

[0084] The video coder then checks first indication associated with the first partition level to determine whether a lossless mode is enabled at the first partition level or not. In some implementations, the first indication is a one-bit flag dedicated for indicating the lossless mode (e.g., `transquant_bypass_enabled_flag` equals to a Boolean value “1”). In some other embodiments, the first indication is a combination of multiple parameters, each parameter having a respective value, the combination of the respective values of the multiple parameters corresponding to the lossless mode. For example, a combination of a syntax element representing that `transform_skip` is activated and a value of the quantization parameter (QP) at certain level (e.g., 4) can be interpreted as indicating that the lossless mode is enabled for the coding blocks at or below of the first partition level. Note that the fact that the first indication is a combination of multiple parameters indicates that none of these

parameters have a dedicated relationship with the lossless mode like the one-bit flag `transquant_bypass_enabled_flag` and at least one of the multiple parameters and its associated value can be used in a lossy mode when one of the multiple parameters does not have a value corresponding to the lossless mode.

[0085] After video decoder 30 determines that the lossless mode is enabled, it configures one or more coding tools under its control in accordance with the lossless mode (610) and then decodes coding blocks at or below the first partition level using the configured one or more coding tools (615). As noted above, there are coding tools that are incompatible with the lossless mode and they are disabled when the video decoder 30 decodes the current block in accordance with the lossless mode. Since these coding tools are disabled, video decoder 30 does not need to look up in the video bitstream for their corresponding syntax elements because they may not exist in the bitstream at all. Even if any of them exists, the video decoder 30 will skip it. As noted above, the lossless mode requires disabling the transform coding tools including multiple transform selection (MTS), sub-block transform (SBT), intra sub-partition (ISP), secondary transform, etc. In some implementations, the lossless mode also requires the disabling of the signal data hiding.

[0086] Next video decoder 30 receives, from the video bitstream, second indication associated with a second partition level under the first partition level of the hierarchical structure (620) and determines whether the coding blocks at or below the second partition level are to be decoded under the lossless mode or lossy mode. Note that a sub-block within a large coding block marked with lossless mode can still be coded with either lossless mode or lossy mode because that the lossy mode has a lower requirement than the lossless mode. For example, slice 516a in FIG. 5B can be labeled as lossless mode while tile 514a can still be labeled as lossy mode. In contrast, if slice 516a has been labeled as lossy mode, tile 514a has to be labeled as lossy mode as well. If the second indication indicates that a lossy mode is enabled at the second partition level, video decoder 30 then changes a configuration of one of the one or more coding tools in accordance with the lossy mode (625) and decodes coding

blocks at or below the first partition level using the configured one or more coding tools (630). In some implementations, video decoder 30 changes the configuration of a coding tool based on a syntax element corresponding to the coding tool in the video bitstream. For example, the syntax element may indicate that MTS or SBT is enabled at the second partition level or below. As noted above, all the coding blocks at or below the second partition level will be treated as being decoded in accordance with the lossy mode.

[0087] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over, as one or more instructions or code, a computer-readable medium and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates the transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the implementations described in the present application. A computer program product may include a computer-readable medium.

[0088] The terminology used in the description of the implementations herein is for the purpose of describing particular implementations only and is not intended to limit the scope of claims. As used in the description of the implementations and the appended claims, the singular forms “a,” “an,” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will also be understood that the term “and/or” as used herein refers to and encompasses any and all possible combinations of one or more of the associated listed items. It will be further understood that the terms “comprises” and/or

“comprising,” when used in this specification, specify the presence of stated features, elements, and/or components, but do not preclude the presence or addition of one or more other features, elements, components, and/or groups thereof.

[0089] It will also be understood that, although the terms first, second, etc. may be used herein to describe various elements, these elements should not be limited by these terms. These terms are only used to distinguish one element from another. For example, a first electrode could be termed a second electrode, and, similarly, a second electrode could be termed a first electrode, without departing from the scope of the implementations. The first electrode and the second electrode are both electrodes, but they are not the same electrode.

[0090] The description of the present application has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications, variations, and alternative implementations will be apparent to those of ordinary skill in the art having the benefit of the teachings presented in the foregoing descriptions and the associated drawings. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others skilled in the art to understand the invention for various implementations and to best utilize the underlying principles and various implementations with various modifications as are suited to the particular use contemplated. Therefore, it is to be understood that the scope of claims is not to be limited to the specific examples of the implementations disclosed and that modifications and other implementations are intended to be included within the scope of the appended claims.

WHAT IS CLAIMED IS:

1. A method of decoding video data, the method comprising:
receiving, from a video bitstream having a hierarchical structure, first indication associated with a first partition level of the hierarchical structure;
in accordance with a determination that the first indication indicates that a lossless mode is enabled at the first partition level:
configuring one or more coding tools in accordance with the lossless mode;
and
decoding coding blocks at or below the first partition level using the configured one or more coding tools.
2. The method of claim 1, wherein the configuring one or more coding tools in accordance with the lossless mode further comprises disabling a transform coding tool in accordance with the lossless mode.
3. The method of claim 2, wherein there is no syntax element related to the transform coding tool in the video bitstream.
4. The method of claim 2, wherein the transform coding tool includes multiple transform selection (MTS), sub-block transform (SBT), intra sub-partition (ISP), and secondary transform.
5. The method of any of claims 1 to 4, wherein the configuring one or more coding tools in accordance with the lossless mode further comprises disabling sign data hiding in accordance with the lossless mode.
6. The method of any of claims 1 to 5, wherein the first indication is a one-bit flag dedicated for indicating the lossless mode.

7. The method of any of claims 1 to 6, wherein the first indication is a combination of multiple parameters, each parameter having a respective value, the combination of the respective values of the multiple parameters corresponding to the lossless mode.

8. The method of claim 7, wherein at least one of the multiple parameters and its associated value can be used in a lossy mode when one of the multiple parameters does not have a value corresponding to the lossless mode.

9. The method of claim 1, further comprising:

receiving, from the video bitstream, second indication associated with a second partition level under the first partition level of the hierarchical structure;

in accordance with a determination that the second indication indicates that a lossy mode is enabled at the second partition level:

changing a configuration of one of the one or more coding tools in accordance with the lossy mode; and

decoding coding blocks at or below the second partition level using the coding tool with the changed configuration.

10. An electronic apparatus including one or more processors, memory and a plurality of programs stored in the memory, wherein the programs, when executed by the one or more processors, cause the electronic apparatus to perform methods as described in any of the claims 1-9.

11. A non-transitory computer-readable storage medium storing a plurality of programs for execution by an electronic apparatus having one or more processors, wherein the programs, when executed by the one or more processors, cause the electronic apparatus to perform methods as described in any of the claims 1-9.

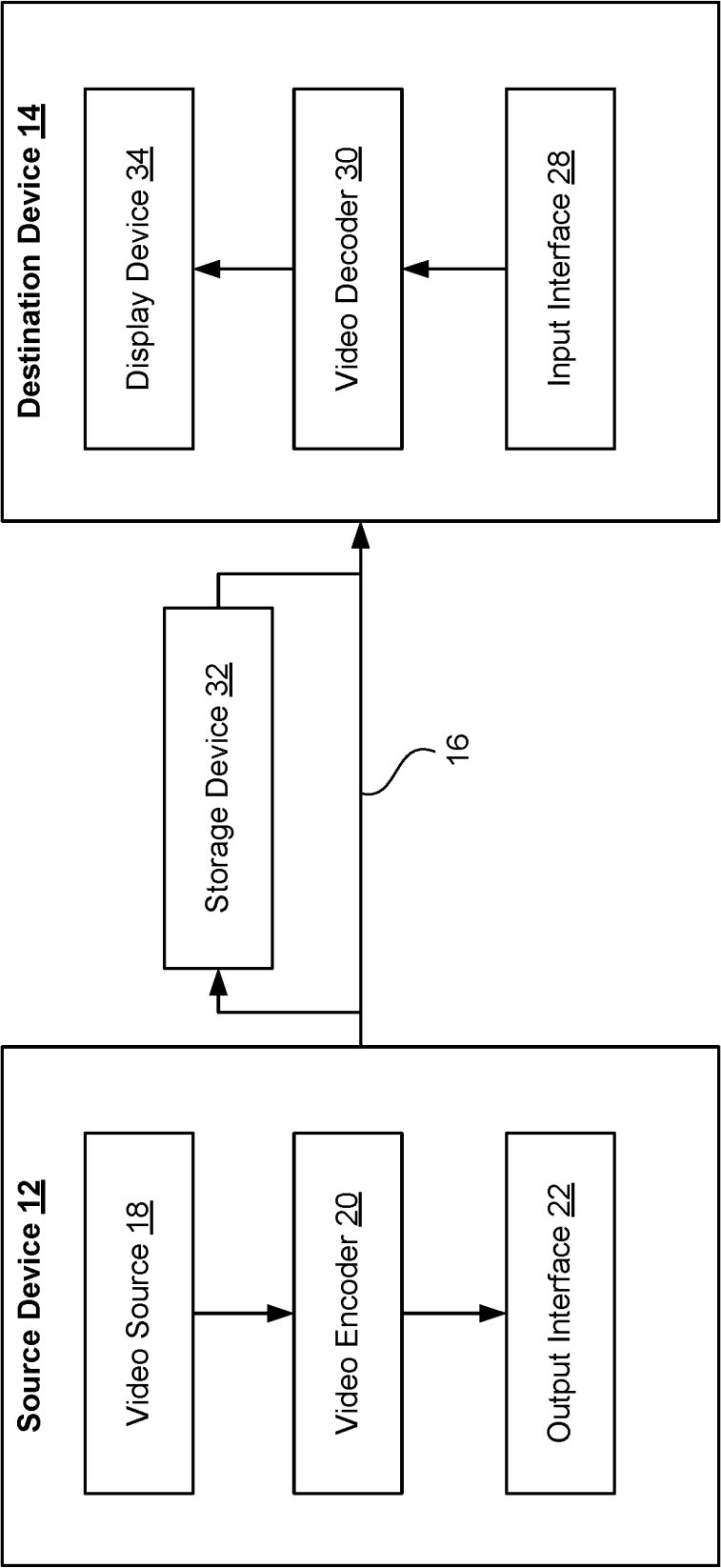


FIG. 1

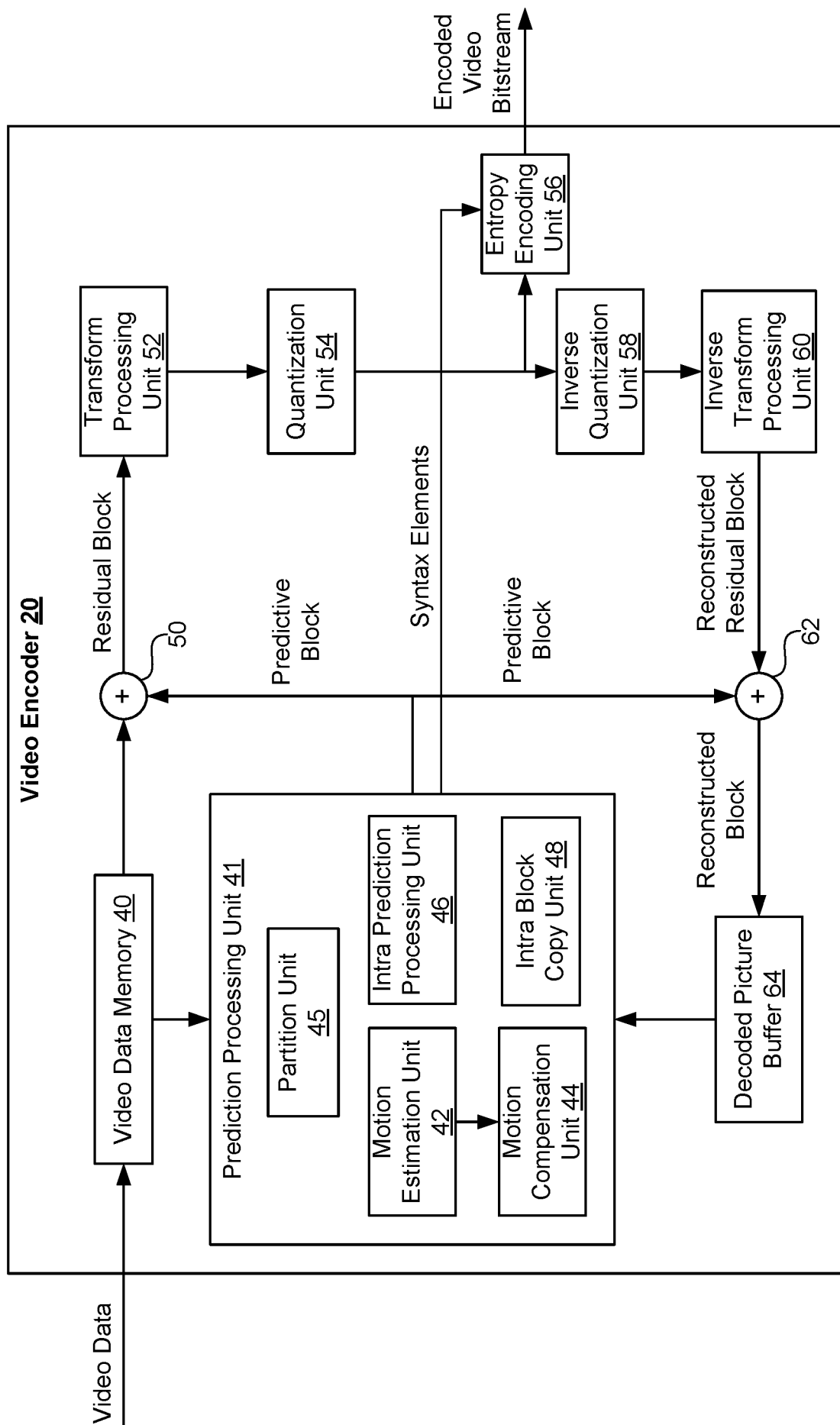


FIG. 2

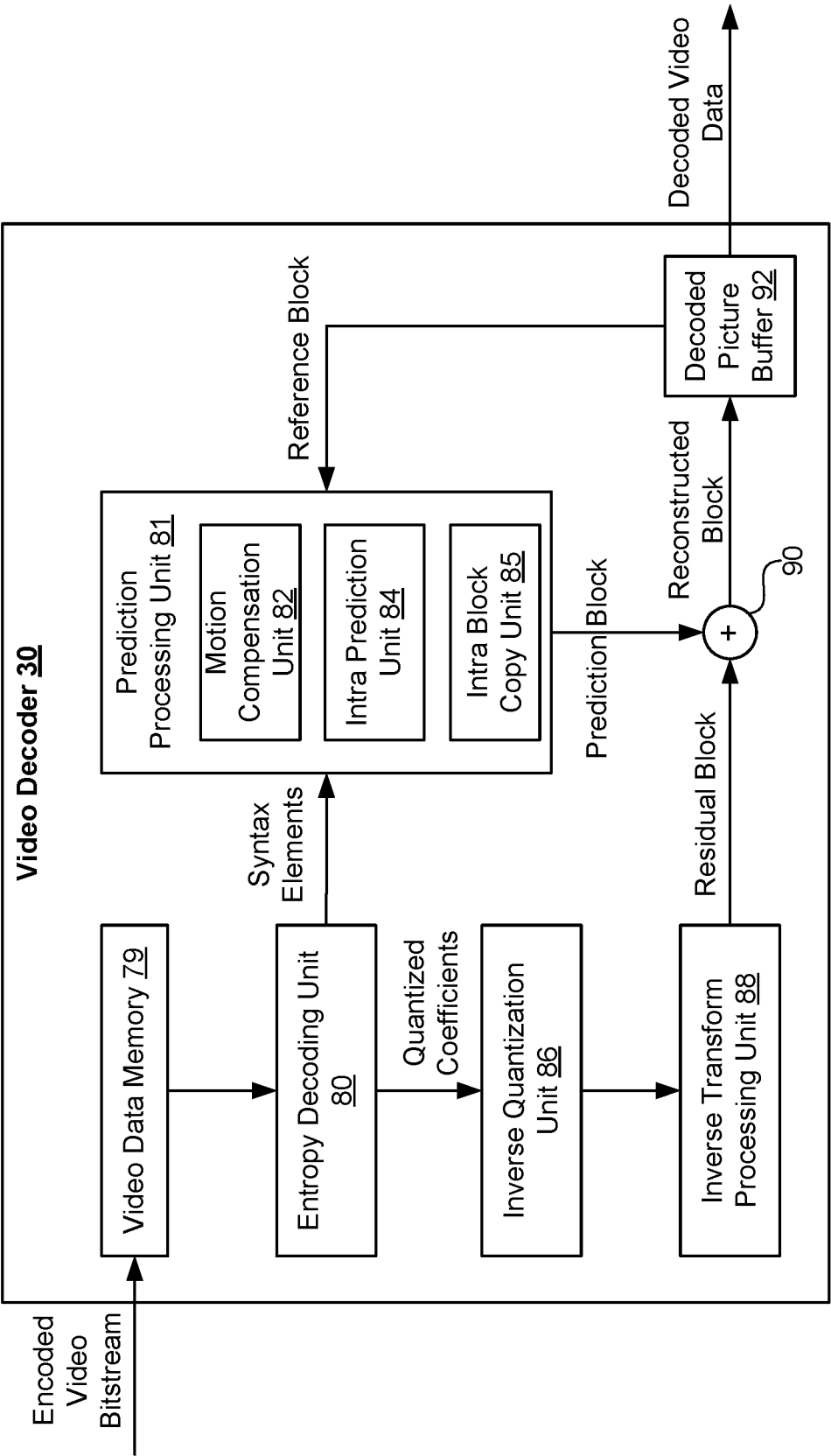


FIG. 3

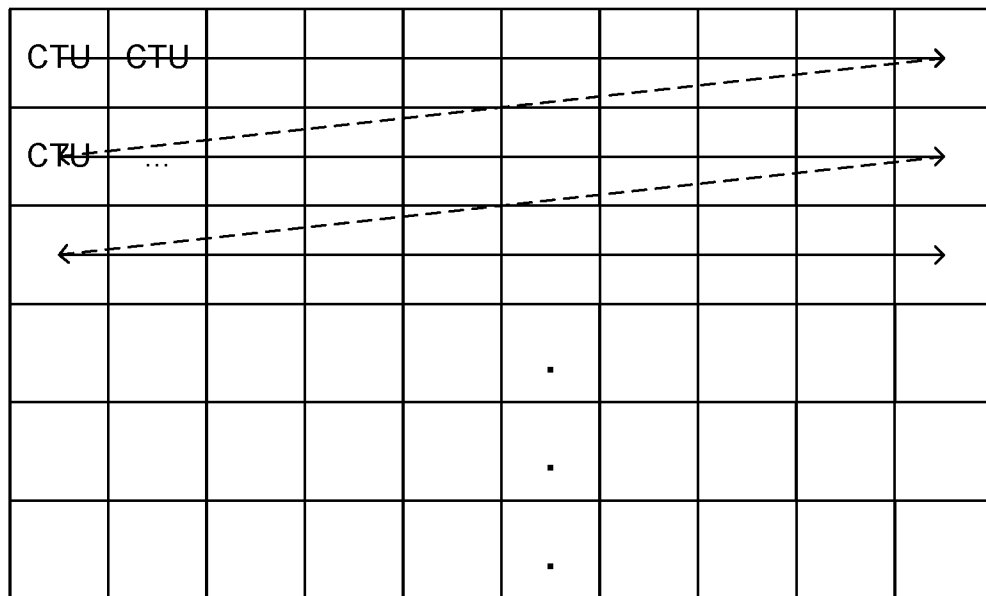


FIG. 4A

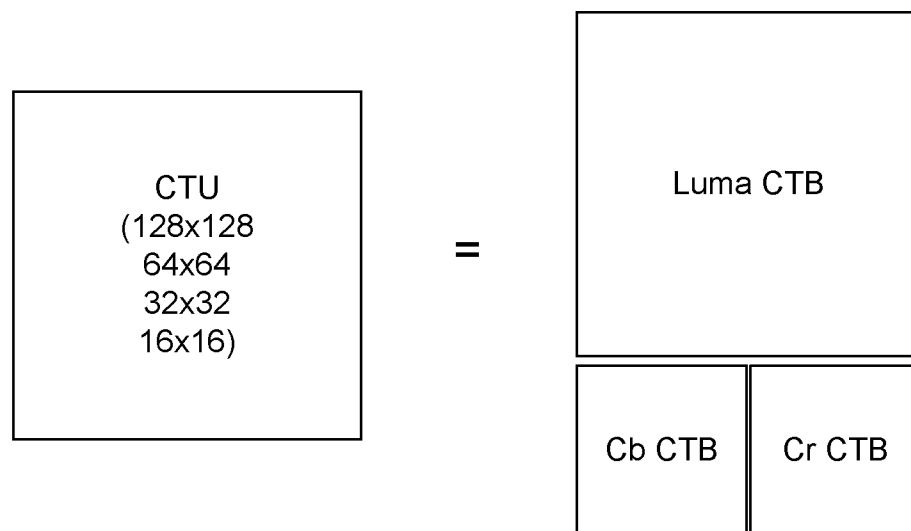


FIG. 4B

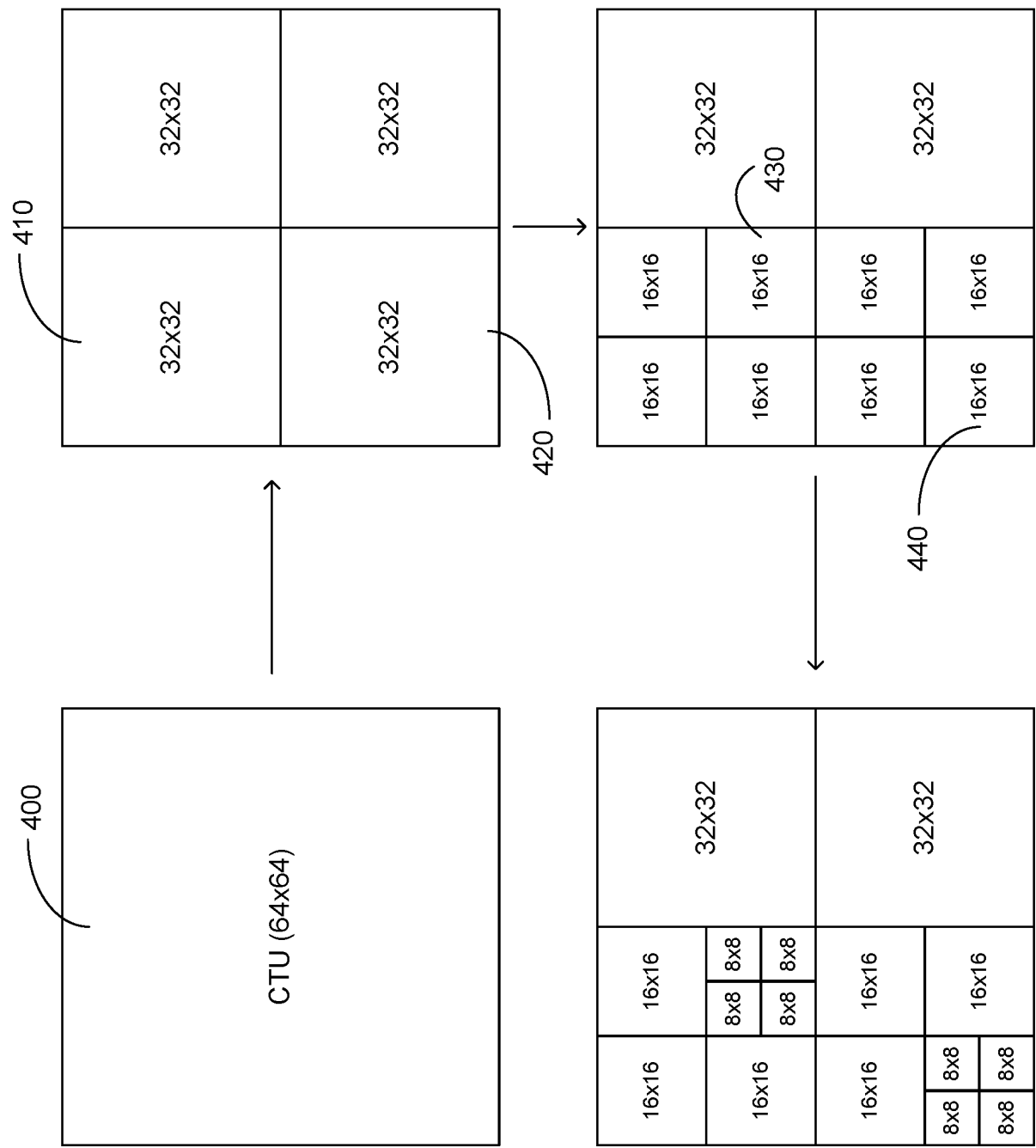


FIG. 4C

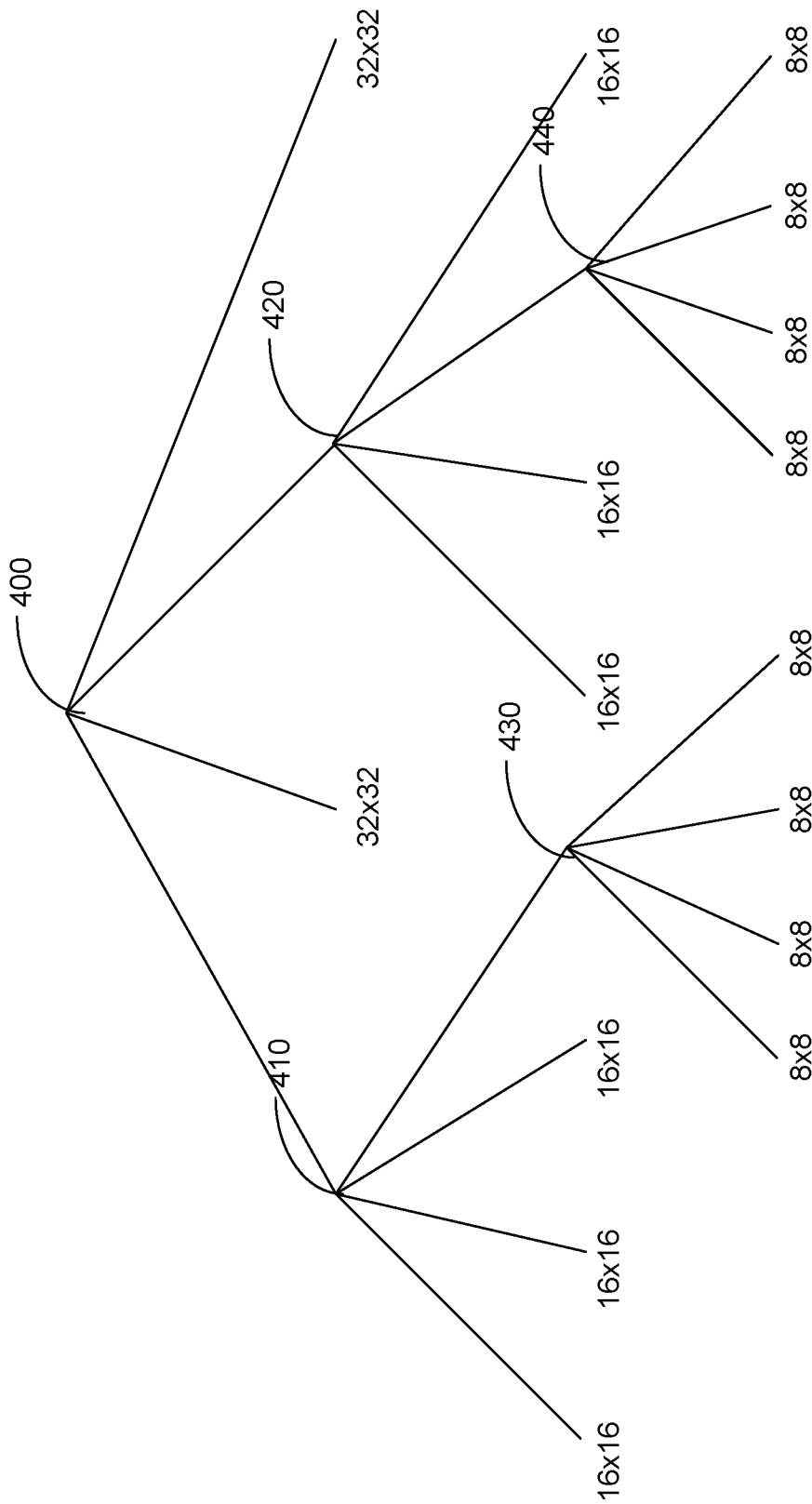


FIG. 4D

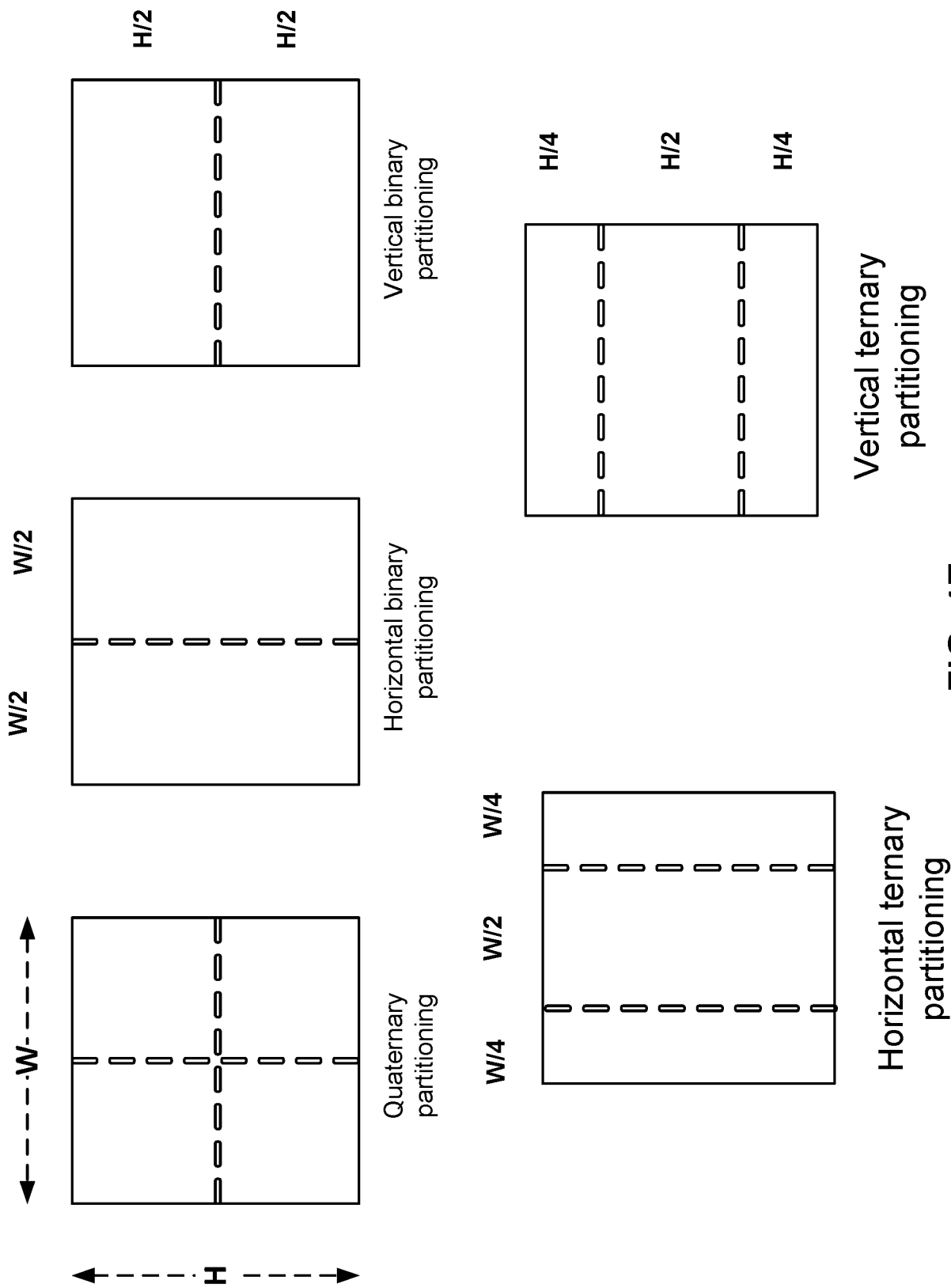
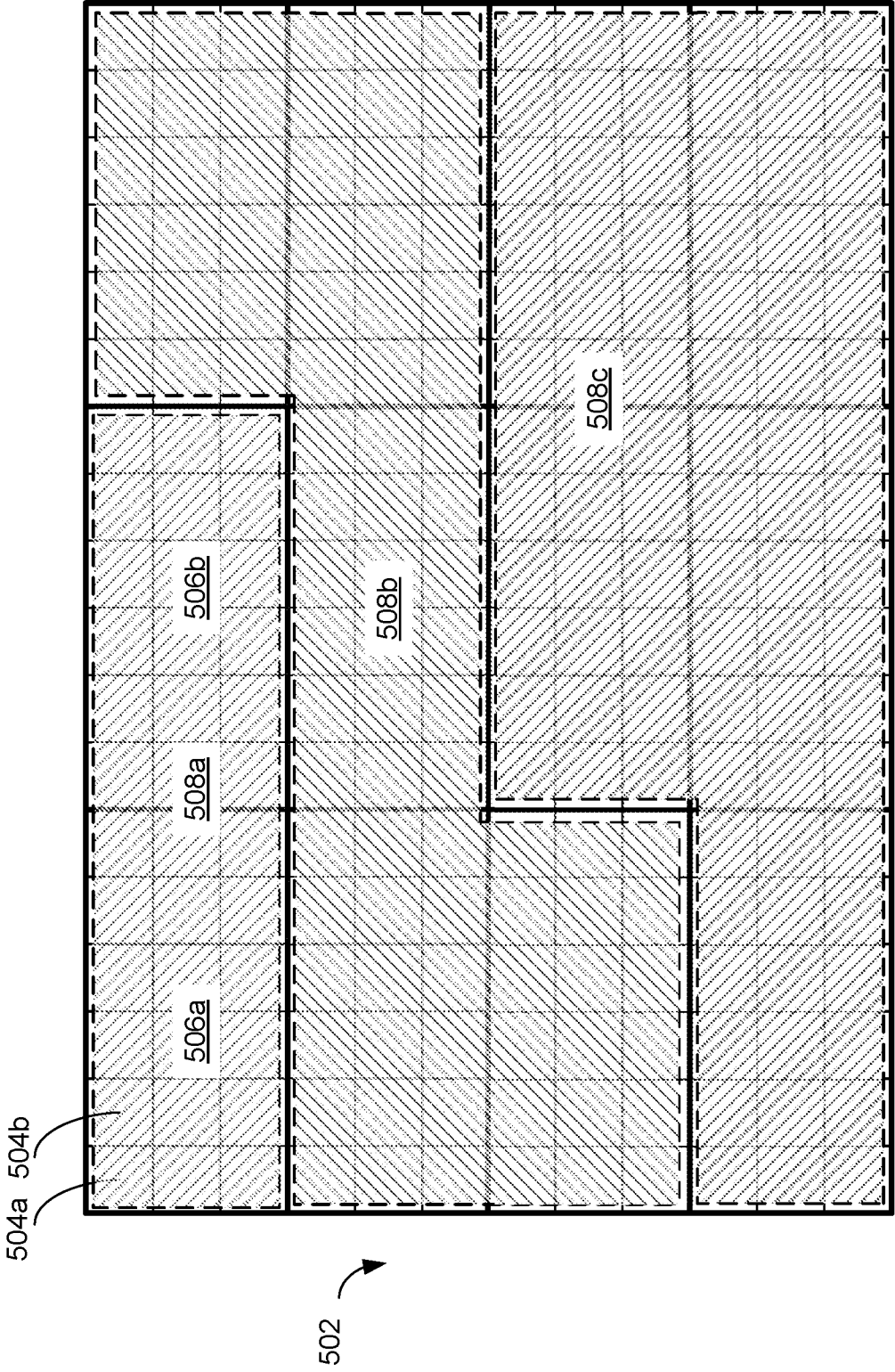


FIG. 4E



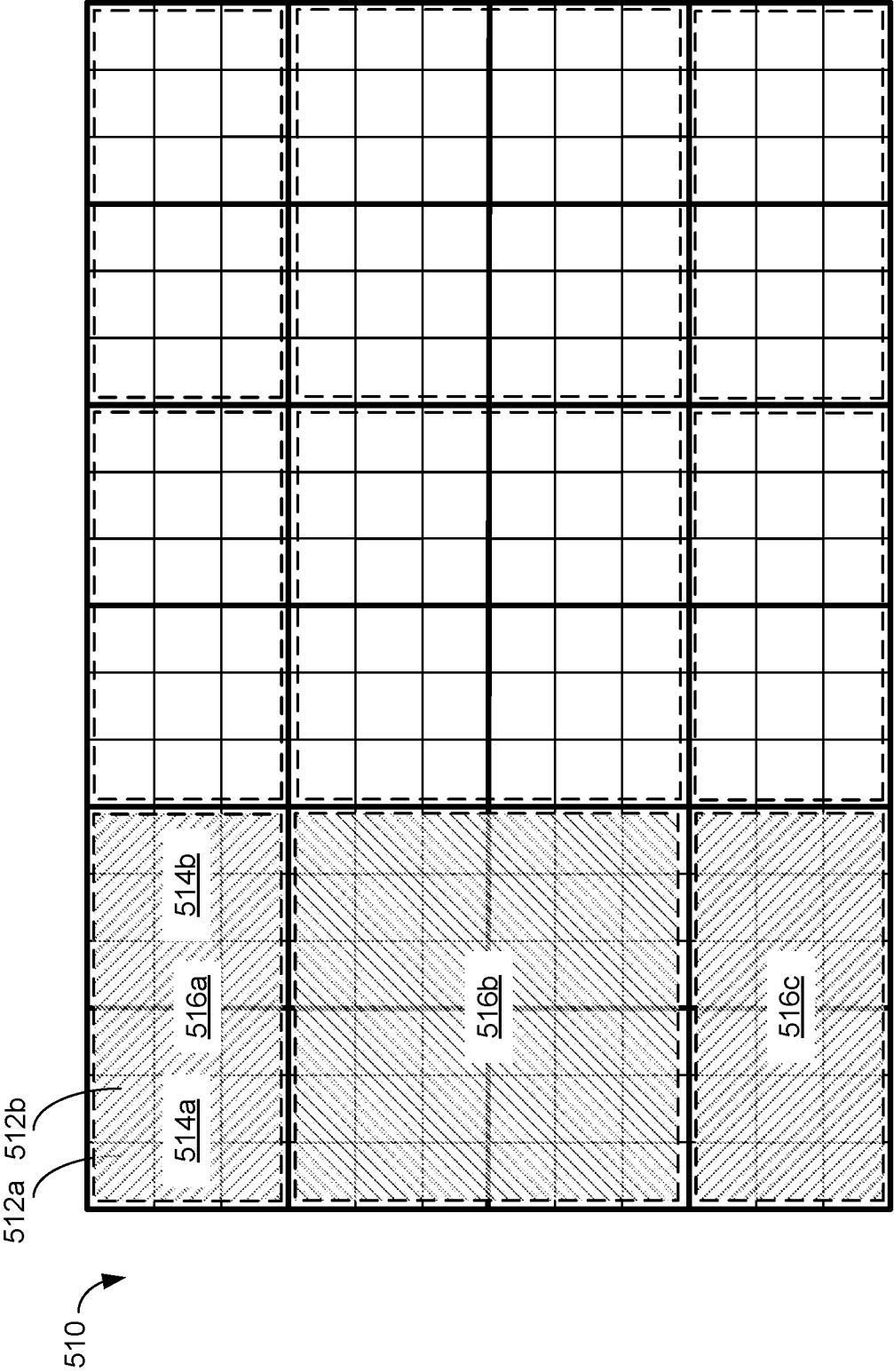
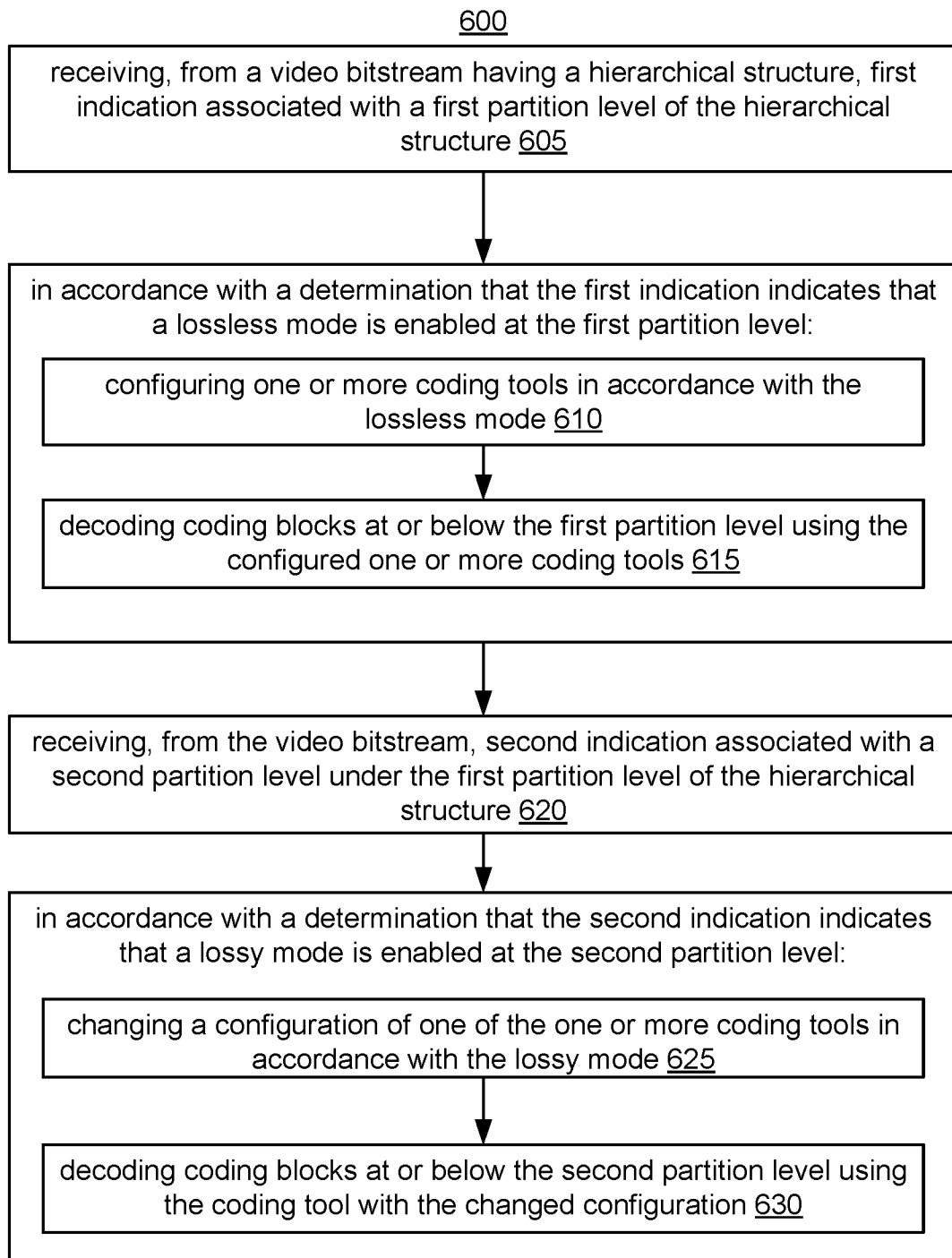


FIG. 5B

**FIG. 6**

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/119(2014.01)i, H04N 19/70(2014.01)i, H04N 19/60(2014.01)i, H04N 19/129(2014.01)i, H04N 19/186(2014.01)i, H04N 19/176(2014.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N 19/119; H04N 19/103; H04N 19/159; H04N 19/172; H04N 19/174; H04N 19/176; H04N 19/70; H04N 7/24; H04N 9/47; H04N 19/60; H04N 19/129; H04N 19/186

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: video coding, partition level, lossless mode, lossy mode, one-bit flag, transform coding tool

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2017-0180737 A1 (VID SCALE, INC.) 22 June 2017 paragraphs [0076], [0079], [0082]; and claims 30-32	1-11
Y	US 9667997 B2 (HFI INNOVATION INC.) 30 May 2017 column 3, lines 25-31; and claims 1, 7, 18	1-11
A	US 10116933 B2 (MEDIATEK INC.) 30 October 2018 claim 1	1-11
A	KR 10-0936208 B1 (SEO CO., LTD.) 12 January 2010 paragraph [0040]	1-11
A	WO 2015-138954 A1 (QUALCOMM INCORPORATED) 17 September 2015 paragraph [0058]	1-11



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

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

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search

10 September 2020 (10.09.2020)

Date of mailing of the international search report

11 September 2020 (11.09.2020)

Name and mailing address of the ISA/KR

International Application Division

Korean Intellectual Property Office

189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

YANG JEONG ROK

Telephone No. +82-42-481-5709



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2020/035075

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2017-0180737 A1	22/06/2017	AU 2015-231883 A1 AU 2015-231883 B2 CN 111064965 A EP 3120561 A2 JP 2020-065284 A JP 6640731 B2 KR 10-2016-0135262 A US 10404988 B2 WO 2015-142556 A2 WO 2015-142556 A3	06/10/2016 16/05/2019 24/04/2020 25/01/2017 23/04/2020 05/02/2020 25/11/2016 03/09/2019 24/09/2015 26/11/2015
US 9667997 B2	30/05/2017	CA 2876017 A1 CA 2876017 C CN 104380734 A CN 104380734 B EP 2859726 A1 KR 10-1880793 B1 KR 10-2015-0027788 A US 2015-0110180 A1 WO 2013-181821 A1 WO 2013-181979 A1	12/12/2013 27/02/2018 25/02/2015 15/09/2017 15/04/2015 20/07/2018 12/03/2015 23/04/2015 12/12/2013 12/12/2013
US 10116933 B2	30/10/2018	CN 105556962 A CN 105556962 B US 2016-0198154 A1 WO 2015-055111 A1	04/05/2016 24/05/2019 07/07/2016 23/04/2015
KR 10-0936208 B1	12/01/2010	WO 2010-067973 A2 WO 2010-067973 A3	17/06/2010 05/08/2010
WO 2015-138954 A1	17/09/2015	CN 106105205 A CN 106105205 B EP 3117614 A1 EP 3117614 B1 JP 2017-513324 A JP 6657110 B2 KR 10-1937548 B1 KR 10-2016-0132925 A US 10271052 B2 US 2015-0264363 A1 US 9948933 B2 WO 2015-138962 A1	09/11/2016 31/05/2019 18/01/2017 26/09/2018 25/05/2017 04/03/2020 10/01/2019 21/11/2016 23/04/2019 17/09/2015 17/04/2018 17/09/2015