



(51) International Patent Classification:

H04N 19/60 (2014.01) *H04N 19/119* (2014.01)
H04N 19/70 (2014.01) *H04N 19/122* (2014.01)
H04N 19/186 (2014.01) *H04N 19/423* (2014.01)

(21) International Application Number:

PCT/US2021/015504

(22) International Filing Date:

28 January 2021 (28.01.2021)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

62/966,871 28 January 2020 (28.01.2020) 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, 100085 (CN).

(72) Inventor; and

(71) Applicant: **WANG, Xianglin** [US/US]; 10524 Blue Granite Drive, San Diego, CA 92127 (US).

(72) Inventors: **XIU, Xiaoyu**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100085 (CN). **MA, Tsung-Chuan**; Room 101D1-7, 1st

Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100085 (CN). **CHEN, Yi-Wen**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100085 (CN). **JHU, Hong-Jheng**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100085 (CN). **YU, Bing**; Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100085 (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, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(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,

(54) Title: METHODS AND APPARATUS ON TRANSFORM AND COEFFICIENT SIGNALING

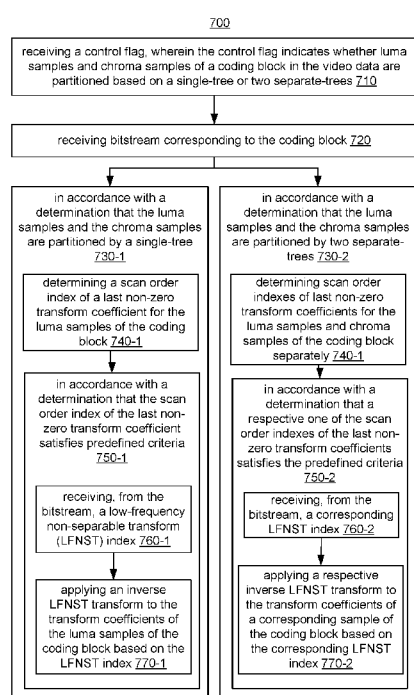


FIG. 7

(57) Abstract: An electronic apparatus performs a method of decoding video data. The method comprises: receiving a control flag, wherein the control flag indicates whether luma samples and chroma samples of a coding block in the video data are partitioned based on a single-tree or two separate-trees; receiving bitstream corresponding to the coding block; in accordance with a determination that the luma samples and the chroma samples are partitioned by a single-tree: determining a scan order index of a last non-zero transform coefficient for the luma samples of the coding block; in accordance with a determination that the scan order index of the last non-zero transform coefficient satisfies predefined criteria: receiving, from the bitstream, a low-frequency non-separable transform (LFNST) index; and applying an inverse LFNST transform to the transform coefficients of the luma samples of the coding block based on the LFNST index; in accordance with a determination that the luma samples and the chroma samples are partitioned by two separate-trees: determining scan order indexes of last non-zero transform coefficients for the luma samples and chroma samples of the coding block separately; in accordance with a determination that a respective one of the scan order indexes of the last non-zero transform coefficients satisfies the predefined criteria: receiving, from the bitstream, a corresponding LFNST index; and applying a respective inverse LFNST transform to the transform coefficients of a corresponding sample of the coding block based on the corresponding LFNST index.

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))*

METHODS AND APPARATUS ON TRANSFORM AND COEFFICIENT SIGNALING

RELATED APPLICATIONS

[0001] The present application claims priority to US Provisional Patent Application No. 62/966,871, entitled “METHODS AND APPARATUS ON TRANSFORM AND COEFFICIENT SIGNALING” filed January 28, 2020, which is incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The present application generally relates to video coding and compression, and more specifically, to methods and apparatus on improving and simplifying existing designs of transform and coefficient coding methods in the versatile video coding (VVC) standard.

BACKGROUND

[0003] 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 neighboring 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 neighboring blocks within the same video frame or temporal prediction with respect to reference samples in other previous and/or future reference video frames.

[0004] Spatial or temporal prediction based on a reference block that has been previously encoded, e.g., a neighboring 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.

[0005] 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.

[0006] 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

[0007] The present application describes implementations related to video data encoding and decoding and, more particularly, to methods and apparatus on improving and simplifying existing designs of transform and coefficient coding methods.

[0008] According to a first aspect of the present application, a method of decoding video data is performed at a computing device and the method includes the following steps: receiving a control flag, wherein the control flag indicates whether luma samples and chroma samples of a coding block in the video data are partitioned based on a single-tree or two separate-trees; receiving bitstream corresponding to the coding block; in accordance with a determination that the luma samples and the chroma samples are partitioned by a single-tree: determining a scan order index of a last non-zero transform coefficient for the luma samples of the coding block; in accordance with a determination that the scan order index of the last non-zero transform coefficient satisfies predefined criteria: receiving, from the bitstream, a low-frequency non-separable transform (LFNST) index; and applying an inverse LFNST transform to the transform coefficients of the luma samples of the coding block based on the LFNST index; in accordance with a determination that the luma samples and the chroma samples are partitioned by two separate-trees: determining scan order indexes of last non-zero transform coefficients for the luma samples and chroma samples of the coding block separately; in accordance with a determination that a respective one of the scan order indexes of the last non-zero transform coefficients satisfies the predefined criteria: receiving, from the bitstream, a corresponding LFNST index; and applying a respective inverse LFNST

transform to the transform coefficients of a corresponding sample of the coding block based on the corresponding LFNST index.

[0009] According to a second aspect of the present application, an electronic apparatus includes one or more processing units, memory and a plurality of programs stored in the memory. The programs, when executed by the one or more processing units, cause the electronic apparatus to perform the method of decoding video data as described above.

[0010] 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 processing units. The programs, when executed by the one or more processing units, cause the electronic apparatus to perform the method of decoding video data as described above.

BRIEF DESCRIPTION OF DRAWINGS

[0011] 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.

[0012] FIG. 1 is a block diagram illustrating an exemplary video encoding and decoding system in accordance with some implementations of the present disclosure.

[0013] FIG. 2 is a block diagram illustrating an exemplary video encoder in accordance with some implementations of the present disclosure.

[0014] FIG. 3 is a block diagram illustrating an exemplary video decoder in accordance with some implementations of the present disclosure.

[0015] 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.

[0016] FIG. 5 is a block diagram illustrating an exemplary low-frequency non-separable transform (LFNST) process as a secondary transform for compacting energy of the transform coefficients of intra coded blocks after primary transform in accordance with some implementations of the present disclosure.

[0017] FIG. 6 is a block diagram illustrating an exemplary transform block with non-zero transform coefficients in accordance with some implementations of the present disclosure.

[0018] FIG. 7 is a flowchart illustrating an exemplary process by which a video coder implements the techniques of signaling the LFNST conditionally based on the different components of a transform block in accordance with some implementations of the present disclosure.

DETAILED DESCRIPTION

[0019] 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.

[0020] 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 at a later time 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, smart phones, 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.

[0021] 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.

[0022] 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.

[0023] 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.

[0024] 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 (or alternatively) 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.

[0025] 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.

[0026] 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.

[0027] 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.

[0028] 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.

[0029] 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.

[0030] 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.

[0031] 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.

[0032] 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). 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.

[0033] 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 neighboring 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.

[0034] 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.

[0035] 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.

[0036] 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.

[0037] 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.

[0038] 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.

[0039] 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.

[0040] 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.

[0041] 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.

[0042] 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.

[0043] 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.

[0044] 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.

[0045] 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.

[0046] 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.

[0047] 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.

[0048] 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.

[0049] 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.

[0050] 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.

[0051] 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.

[0052] 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.

[0053] 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.

[0054] 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.

[0055] 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.

[0056] 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.

[0057] 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.

[0058] 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.

[0059] 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.

[0060] 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.

[0061] 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 divided 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 partitioning types, i.e., quaternary partitioning, horizontal binary partitioning, vertical binary partitioning, horizontal ternary partitioning, and vertical ternary partitioning.

[0062] In some implementations, video encoder 20 may further partition a coding block of a CU into one or more MxN 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.

[0063] 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.

[0064] 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.

[0065] 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.

[0066] 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.

[0067] 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.

[0068] 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.

[0069] As noted above, video coding achieves video compression using primarily two modes, i.e., intra-frame prediction (or intra-prediction) and inter-frame prediction (or inter-prediction). Palette-based coding is another coding scheme that has been adopted by many video coding standards. In palette-based coding, which may be particularly suitable for screen-generated content coding, a video coder (e.g., video encoder 20 or video decoder 30) forms a palette table of colors representing the video data of a given block. The palette table includes the most dominant (e.g., frequently used) pixel values in the given block. Pixel values that are not frequently represented in the video data of the given block are either not included in the palette table or included in the palette table as escape colors.

[0070] Each entry in the palette table includes an index for a corresponding pixel value that in the palette table. The palette indices for samples in the block may be coded to indicate which entry from the palette table is to be used to predict or reconstruct which

sample. This palette mode starts with the process of generating a palette predictor for a first block of a picture, slice, tile, or other such grouping of video blocks. As will be explained below, the palette predictor for subsequent video blocks is typically generated by updating a previously used palette predictor. For illustrative purpose, it is assumed that the palette predictor is defined at a picture level. In other words, a picture may include multiple coding blocks, each having its own palette table, but there is one palette predictor for the entire picture.

[0071] To reduce the bits needed for signaling palette entries in the video bitstream, a video decoder may utilize a palette predictor for determining new palette entries in the palette table used for reconstructing a video block. For example, the palette predictor may include palette entries from a previously used palette table or even be initialized with a most recently used palette table by including all entries of the most recently used palette table. In some implementations, the palette predictor may include fewer than all the entries from the most recently used palette table and then incorporate some entries from other previously used palette tables. The palette predictor may have the same size as the palette tables used for coding different blocks or may be larger or smaller than the palette tables used for coding different blocks. In one example, the palette predictor is implemented as a first-in-first-out (FIFO) table including 64 palette entries.

[0072] To generate a palette table for a block of video data from the palette predictor, a video decoder may receive, from the encoded video bitstream, a one-bit flag for each entry of the palette predictor. The one-bit flag may have a first value (e.g., a binary one) indicating that the associated entry of the palette predictor is to be included in the palette table or a second value (e.g., a binary zero) indicating that the associated entry of the palette predictor is not to be included in the palette table. If the size of palette predictor is larger than the palette table used for a block of video data, then the video decoder may stop receiving more flags once a maximum size for the palette table is reached.

[0073] In some implementations, some entries in a palette table may be directly signaled in the encoded video bitstream instead of being determined using the palette

predictor. For such entries, the video decoder may receive, from the encoded video bitstream, three separate m-bit values indicating the pixel values for the luma and two chroma components associated with the entry, where m represents the bit depth of the video data. Compared with the multiple m-bit values needed for directly signaled palette entries, those palette entries derived from the palette predictor only require a one-bit flag. Therefore, signaling some or all palette entries using the palette predictor can significantly reduce the number of bits needed to signal the entries of a new palette table, thereby improving the overall coding efficiency of palette mode coding.

[0074] In many instances, the palette predictor for one block is determined based on the palette table used to code one or more previously coded blocks. But when coding the first coding tree unit in a picture, a slice or a tile, the palette table of a previously coded block may not be available. Therefore a palette predictor cannot be generated using entries of the previously used palette tables. In such case, a sequence of palette predictor initializers may be signaled in a sequence parameter set (SPS) and/or a picture parameter set (PPS), which are values used to generate a palette predictor when a previously used palette table is not available. An SPS generally refers to a syntax structure of syntax elements that apply to a series of consecutive coded video pictures called a coded video sequence (CVS) as determined by the content of a syntax element found in the PPS referred to by a syntax element found in each slice segment header. A PPS generally refers to a syntax structure of syntax elements that apply to one or more individual pictures within a CVS as determined by a syntax element found in each slice segment header. Thus, an SPS is generally considered to be a higher level syntax structure than a PPS, meaning the syntax elements included in the SPS generally change less frequently and apply to a larger portion of video data compared to the syntax elements included in the PPS.

[0075] FIG. 5 is a block diagram illustrating an exemplary low-frequency non-separable transform (LFNST) process as a secondary transform for compacting energy of the transform coefficients of intra coded blocks after primary transform in accordance with some implementations of the present disclosure. As shown in the figure, LFNST is applied between

primary forward transform and quantization within video encoder 20 and between de-quantization and inverse primary transform within video decoder 30. In some embodiments, non-separable transforms with varying transform sizes are applied based on the size of one coding block, which can be depicted as the following matrix multiplication process.

Assuming that LFNTS is applied to one 4×4 block, the samples inside the 4×4 block, i.e.,

$$X = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix} \quad (1)$$

is first serialized as one vector as

$$\vec{X} = [X_{00} \ X_{01} \ X_{02} \ X_{03} \ X_{10} \ X_{11} \ X_{12} \ X_{13} \ X_{20} \ X_{21} \ X_{22} \ X_{23} \ X_{30} \ X_{31} \ X_{32} \ X_{33}]^T \quad (2)$$

[0076] Then, LFNTS is applied as $\vec{F} = T \cdot \vec{X}$, where $\vec{F}$ is the transform coefficient after LFNTS, T is the transform kernel. In this example, T is one 16×16 matrix. The 16x1 vector $\vec{F}$ is subsequently re-organized as 4×4 block according to one pre-defined scan order, wherein the coefficients located at the beginning of the vector will be associated with smaller scanning index in the 4×4 block.

[0077] As can be seen from the example above, LFNST is based on direct matrix multiplication, which is quite expensive in terms of computational operations and memory for storing transform coefficients. In some embodiments, one reduced non-separable transform kernel is used to reduce the implementation cost of LFNST. The main idea of such method is to map one N-dimension vector to one R-dimension vector in a different space, where $R < N$. Therefore, instead of doing N×N matrix, the forward LFNST becomes an R×N matrix as follows:

$$T_{R \times N} = \begin{bmatrix} t_{00} & t_{01} & \cdots & t_{0N-1} \\ t_{10} & t_{11} & \cdots & t_{1N-1} \\ \vdots & \vdots & \ddots & \vdots \\ t_{N-10} & t_{N-11} & \cdots & t_{N-1N-1} \end{bmatrix} \quad (3)$$

where the R base vectors in $T_{R \times N}$ are generated by picking the first R bases of the original N-dimension transform matrix (i.e., N×N).

[0078] After LFNST is applied, all transform coefficients beyond the top-left potentially non-zero LFNST coefficient region are forced to be zeros. For transform blocks with a size of 4×4 , 8×8 , $4 \times M$ and/or $M \times 4$, the top-left potentially non-zero LFNST coefficient region includes the first 8 coefficient positions along the coefficient scan order. For transform blocks of all other sizes, the top-left potentially non-zero LFNST coefficient region includes the coefficient positions in the top-left 4×4 sub-block. In the following description of the disclosure, such a potentially non-zero LFNST coefficient region is referred as “non-zero LFNST region” for simplicity.

[0079] In some embodiments, there are in total four transform sets with two non-separable transform kernels enabled per transform set. The transform set is selected according to the intra prediction mode of one intra block. The mapping from the intra prediction mode to the transform set is pre-defined as shown in Table 1 below. For each transform set, the selected non-separable secondary transform candidate is indicated by signaling one LFNST index in the video bitstream.

IntraPredMode	set
IntraPredMode < 0	1
0 <= IntraPredMode <= 1	0
2 <= IntraPredMode <= 12	1
13 <= IntraPredMode <= 23	2
24 <= IntraPredMode <= 44	3
45 <= IntraPredMode <= 55	2
56 <= IntraPredMode <= 80	1
81 <= IntraPredMode <= 83	0

Table 1 The mapping between intra mode and LFNST transform set

[0080] In some embodiments, only if all transform coefficients outside the first 4×4 subblock of a given transform block are zeros, the LFNST index is available for parsing at video decoder. Signaling of the LFNST index is dependent on the position of the last significant coefficient, which indicates the number of non-zero coefficients in the transform block. For example, for 4×4 and 8×8 coding blocks, the LFNST index is only signaled when the position of the last significant (i.e., non-zero) transform coefficient is less than 8; for other coding block sizes, the LFNST index is only signaled when the position of the last significant

transform coefficient is less than 16; otherwise, the LFNST index is not signaled, and is always inferred to be zero, i.e., the LFNST is disabled. In some other embodiments, a minimum threshold (e.g., 1) is set for the LFNST index such that LFNST index is not signaled when the total number of non-zero transform coefficients is equal to or less than the minimum threshold.

[0081] Furthermore, to reduce the cache buffer size of transform coefficients, the LFNST is disabled when either the width or the height of the current coding block is greater than the maximum transform size (e.g., 64) as signaled sequence parameter set (SPS). In some embodiments, the LFNST is only applied when the primary transform is DCT2. The LFNST is applied to intra coding blocks in both intra and inter slices, and for both luma and chroma components. If a dual-tree/local-tree (i.e., separate-tree) is enabled (where the partitions of luma and chroma components are misaligned), the LFNST indices are signaled separately for luma and chroma components (i.e., luma and chroma components can apply different LFNST transforms). Otherwise, when single-tree is applied (where the partitions of luma and chroma components are aligned), a single LFNST index is signaled and one same LFNST transform is shared by luma and chroma components.

[0082] FIG. 6 is a block diagram illustrating an exemplary transform block 600 with non-zero transform coefficients in accordance with some implementations of the present disclosure. Transform block 600 includes a first region 602 corresponding to the top-left meshed portion of the transform block 600 and a second region 604 represented by the dashed-line portion of the transform block 600. The first region 602 has a predefined size (e.g., a top-left 16x16 region) of the transform block 600 and includes one or more non-zero transform coefficients (e.g., 1st, 2nd, and 3rd non-zero coefficients 606, 608, and 610). The second region 604 is a region outside the first region 602 that may or may not include one or more non-zero transform coefficients.

[0083] In the current VVC, the signaling of the LFNST index is dependent on the availability of decoded transform coefficients of all components in the CU. Because after LFNST is applied, all the transform coefficients beyond the non-zero LFNST region are

forced to be zeros, the signaling of the LFNST is conditioned on the positions of the last non-zero coefficients of three components in the CU. Specifically, for 4x4 and 8x8 CUs, the LFNST index is only signaled when the positions of the last non-zero coefficients of all components (which apply transform for residual coding, i.e., non-transform-skip component) are less than 8; for other CU sizes, the LFNST index is only signaled when the positions of the last non-zero coefficients of all non-transform-skip components are less than 16. Such parsing dependency can cause undesirable latency for hardware encoder and decoder. For example, by such design, the decoding of the luma component in one TU cannot be started until the parsing of the chroma residuals are finished.

[0084] In some embodiments, one simplified LFNST signaling method is proposed to remove the parsing dependency of the LFNST index on the availability of the transform coefficients of both luma and chroma TBs in one CU. Due to the removed parsing dependency, decoder can timely obtain the information on whether the LFNST is applied to one current CU such that accurate CCB limit can be calculated based on the corresponding number of potentially non-zero coefficients.

[0085] As discussed earlier, LFNST is signaled at the end of one intra CU and the signaling of the LFNST index is dependent on the positions of last significant coefficients of all coded components. For example, due to the zero-out constraint applied to LFNST, the LFNST index is only signaled when the positions of the last non-zero coefficients of coded components are outside the corresponding zero-out region. To solve this problem, the LFNST signaling is conditioned only on the position of last significant coefficient of luma component, as illustrated in the syntax table below.

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	Descriptor
.....	
do {	
if(lastScanPos == 0) {	
lastScanPos = numSbCoeff	
lastSubBlock--	
}	

lastScanPos--	
xS = DiagScanOrder[log2TbWidth – log2SbW][log2TbHeight – log2SbH] [lastSubBlock][0]	
yS = DiagScanOrder[log2TbWidth – log2SbW][log2TbHeight – log2SbH] [lastSubBlock][1]	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) (yC != LastSignificantCoeffY))	
if((lastSubBlock == 0 && log2TbWidth >= 2 && log2TbHeight >= 2 && !transform_skip_flag[x0][y0][cIdx] && lastScanPos > 0) && (treeType != SINGLE_TREE cIdx == 0))	
LfnstDcOnly = 0	
if(((lastSubBlock > 0 && log2TbWidth >= 2 && log2TbHeight >= 2) (lastScanPos > 7 && (log2TbWidth == 2 log2TbWidth == 3) && log2TbWidth == log2TbHeight)) && (treeType != SINGLE_TREE cIdx == 0))	
LfnstZeroOutSigCoeffFlag = 0	
if((lastSubBlock > 0 lastScanPos > 0) && cIdx == 0)	
MtsDcOnly = 0	
.....	
}	

[0086] As illustrated in the syntax table above, in the single-tree case, the LFNST signalling is only conditioned on the position of the last significant coefficient of luma component in the proposed method. For example, for 4×4 and 8×8 coding blocks, the LFNST index is only signalled when the position of the last luma significant transform coefficient is less than 8; for other coding block sizes, the LFNST index is only signalled when the position of the last luma significant transform coefficient is less than 16. In the separate-tree case, the LFNST index is signalled separately for the luma and the chroma components. Additionally, the original DC only constraint is applied such that the LFNST index is only signalled when the position of the last luma significant coefficient is equal to or larger than 1.

[0087] As noted above, luma samples and chroma samples of a coding block may be partitioned using a single-tree or two separate trees. This feature may affect the signaling of the LFNST index. For example, when the luma samples and chroma samples of a coding block are partitioned by a single-tree, only the transform coefficients corresponding to the

luma samples are eligible for LFNST while the chroma samples do not. In this case, there is no need to verify the location of the last non-zero coefficient corresponding to any chroma samples of the coding block before receiving the LFNST index. Instead, only the location of the last non-zero coefficient corresponding to the luma samples of the coding block is relevant for determining whether LFNST has been enabled for the coding block. But when the luma samples and chroma samples of a coding block are partitioned by two separate trees, LFNST is applied separately to the luma samples and the chroma samples, each having its own LFNST index.

[0088] FIG. 7 is a flowchart 700 illustrating an exemplary process by which a video coder (e.g., video decoder 30) implements the techniques of signaling the LFNST conditionally based on the different components of a transform block in accordance with some implementations of the present disclosure.

[0089] The video decoder 30 receives (710) a control flag associated with a coding block or multiple coding blocks. This control flag indicates whether luma samples and chroma samples of the coding block in the video data are partitioned based on a single-tree or two separate-trees. The video decoder also receives (720) bitstream corresponding to the coding block, which may include the transform coefficients associated with different components of the coding block.

[0090] Next, the video decoder 30 determines the partition tree type of the coding block based on the control flag. When the control flag indicates that the luma samples and the chroma samples are partitioned by a single-tree (730-1), the video decoder 30 determines (740-1) a scan order index of a last non-zero transform coefficient for the luma samples of the coding block. As noted above, the single-tree partition means that only the luma samples of the coding block are eligible for LFNST. When the scan order index of the last non-zero transform coefficient satisfies predefined criteria (750-1), the video decoder then receives (760-1) from the bitstream, a LFNST index and applies (770-1) an inverse LFNST transform to the transform coefficients of the luma samples of the coding block based on the LFNST index.

[0091] When the control flag indicates that the luma samples and the chroma samples are partitioned by two separate-trees (730-2), the video decoder 30 determines (740-2) scan order indexes of last non-zero transform coefficients for the luma samples and chroma samples of the coding block separately. As noted above, the luma and chroma components are treated separately by LFNST. For example, when a respective one of the scan order indexes of the last non-zero transform coefficients corresponding to the luma or chroma samples satisfies the predefined criteria (750-2), the video decoder then receives (760-2) from the bitstream, a LFNST index corresponding to that component and applies (770-2) a respective inverse LFNST transform to the transform coefficients of the corresponding component of the coding block based on the corresponding LFNST index.

[0092] In some embodiments, before applying the inverse LFNST transform to the transform coefficients of the luma or chroma samples of the coding block, the video decoder 30 first determines the value of the LFNST index and then identifies a LFNST transform kernel based on the corresponding LFNST index when the corresponding LFNST index is non-zero. As noted above, the video coder has access to multiple LFNST transform kernels and the video encoder 20 chooses one of them for performing LFNST to a coding block and signals the index of the chosen LFNST transform kernel in the video data. The video decoder 30 then receives the LFNST index from the video data and then inverse-transforms the transform coefficients of the corresponding sample of the coding block using the identified LFNST transform kernel.

[0093] In some embodiments, the predefined criteria mentioned above are satisfied when the scan order index of the last non-zero transform coefficient is no less than a minimum threshold and less than a maximum threshold associated with the coding block. For example, the minimum threshold is 1 while the maximum threshold is dependent upon a size of the coding block such as the maximum threshold being 8 for a 4x4 or 8x8 coding block or 16 for other coding block sizes. Like MTS, the inverse LFNST transform is applied to non-zero transform coefficients within a top-left region of a transform block corresponding to the coding block and the scan order is a diagonal scan order.

[0094] 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 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.

[0095] 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.

[0096] 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.

[0097] 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 a control flag, wherein the control flag indicates whether luma samples and chroma samples of a coding block in the video data are partitioned based on a single-tree or two separate-trees;
 - receiving bitstream corresponding to the coding block;
 - in accordance with a determination that the luma samples and the chroma samples are partitioned by a single-tree:
 - determining a scan order index of a last non-zero transform coefficient for the luma samples of the coding block;
 - in accordance with a determination that the scan order index of the last non-zero transform coefficient satisfies predefined criteria:
 - receiving, from the bitstream, a low-frequency non-separable transform (LFNST) index; and
 - applying an inverse LFNST transform to the transform coefficients of the luma samples of the coding block based on the LFNST index;
 - in accordance with a determination that the luma samples and the chroma samples are partitioned by two separate-trees:
 - determining scan order indexes of last non-zero transform coefficients for the luma samples and chroma samples of the coding block separately;
 - in accordance with a determination that a respective one of the scan order indexes of the last non-zero transform coefficients satisfies the predefined criteria:
 - receiving, from the bitstream, a corresponding LFNST index; and
 - applying a respective inverse LFNST transform to the transform coefficients of a corresponding sample of the coding block based on the corresponding LFNST index.

2. The method of claim 1, wherein applying a respective inverse LFNST transform to the transform coefficients of a corresponding sample of the coding block based on the corresponding LFNST index further comprises:
 - in accordance with a determination that the LFNST index is non-zero;
 - identifying a LFNST transform kernel based on the corresponding LFNST index; and
 - inverse-transforming the transform coefficients of the corresponding sample of the coding block using the identified LFNST transform kernel.
3. The method of claim 1, wherein the predefined criteria are satisfied when the scan order index of the last non-zero transform coefficient is no less than a minimum threshold and less than a maximum threshold associated with the coding block.
4. The method of claim 3, wherein the minimum threshold is 1 and the maximum threshold is dependent upon a size of the coding block.
5. The method of claim 4, wherein the maximum threshold is 8 for a 4x4 or 8x8 coding block or 16 for other coding block sizes.
6. The method of claim 1, wherein the inverse LFNST transform is applied to non-zero transform coefficients within a top-left region of a transform block corresponding to the coding block.
7. The method of claim 1, wherein the scan order is a diagonal scan order.
8. An electronic apparatus comprising:
 - one or more processing units;
 - memory coupled to the one or more processing units; and
 - a plurality of programs stored in the memory that, when executed by the one or more processing units, cause the electronic apparatus to perform the method of claims 1-7.
9. A non-transitory computer readable storage medium storing a plurality of programs for execution by an electronic apparatus having one or more processing units, wherein the

plurality of programs, when executed by the one or more processing units, cause the electronic apparatus to perform the method of claims 1-7.

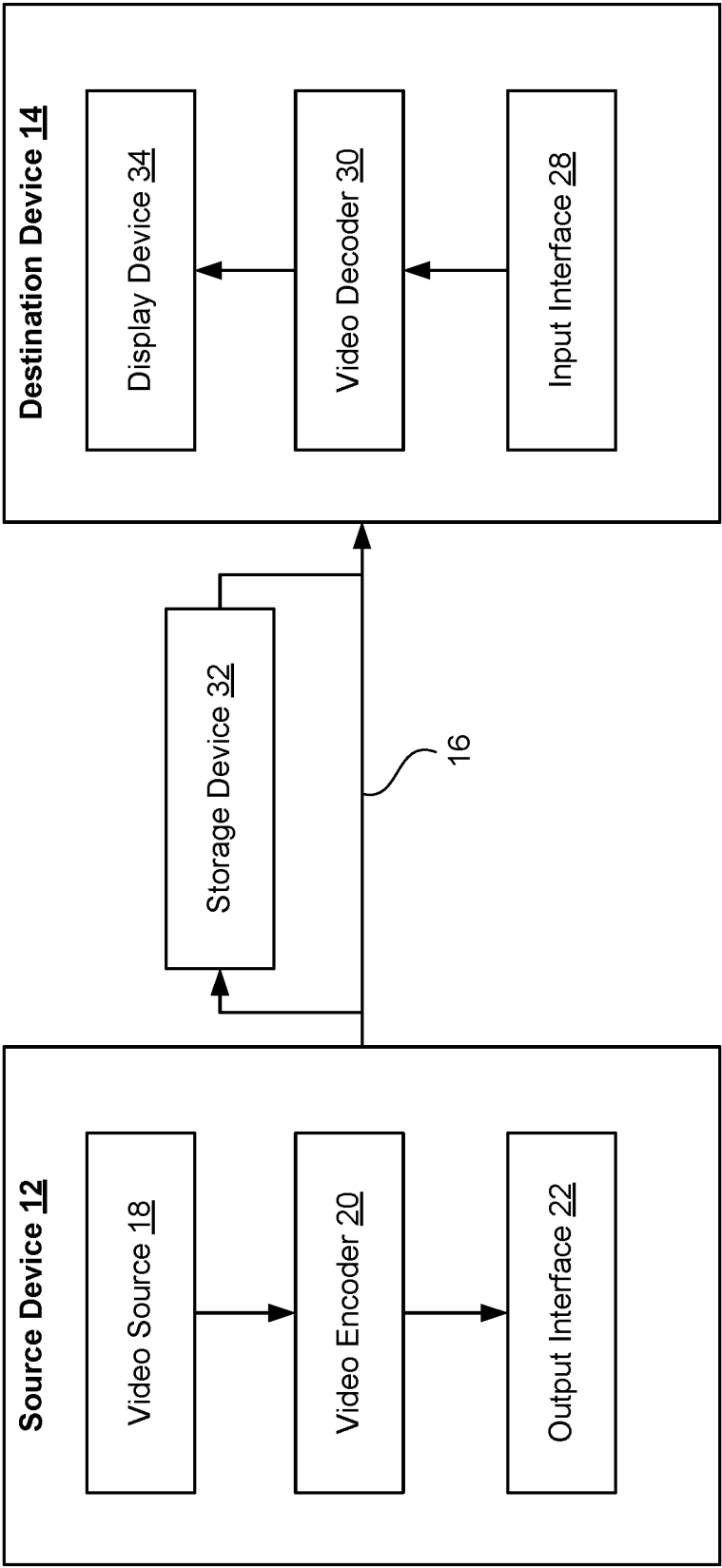


FIG. 1

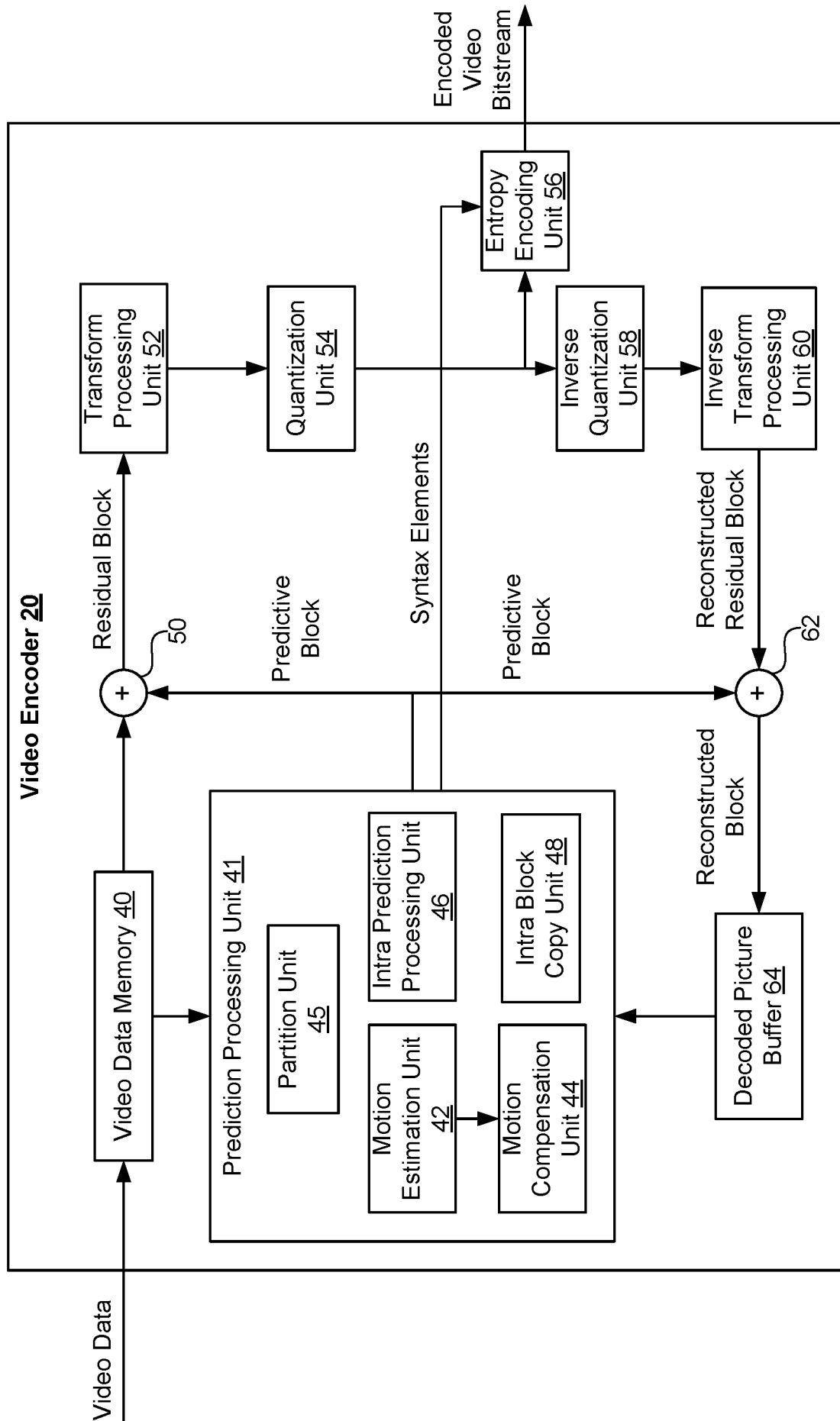


FIG. 2

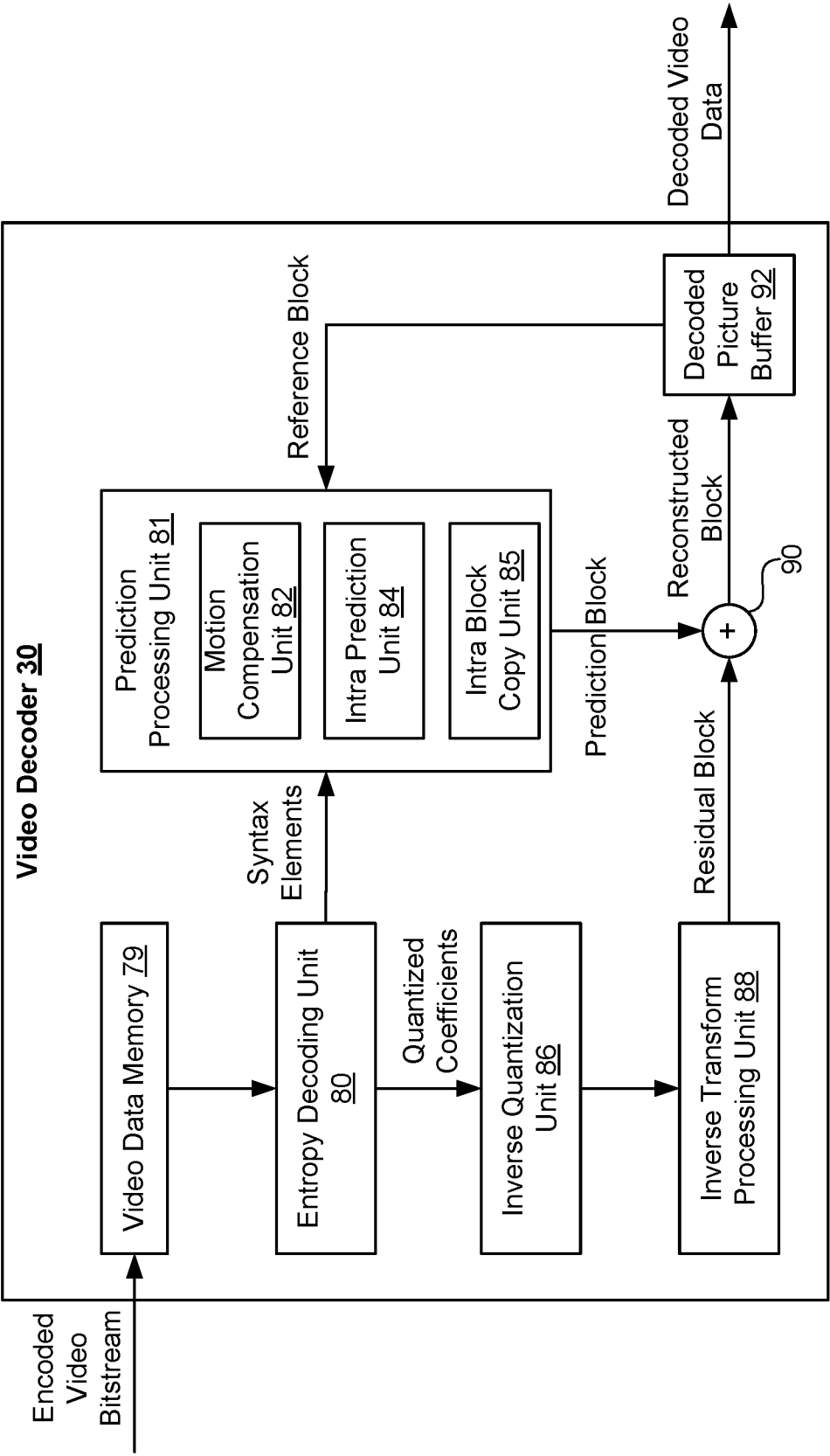


FIG. 3

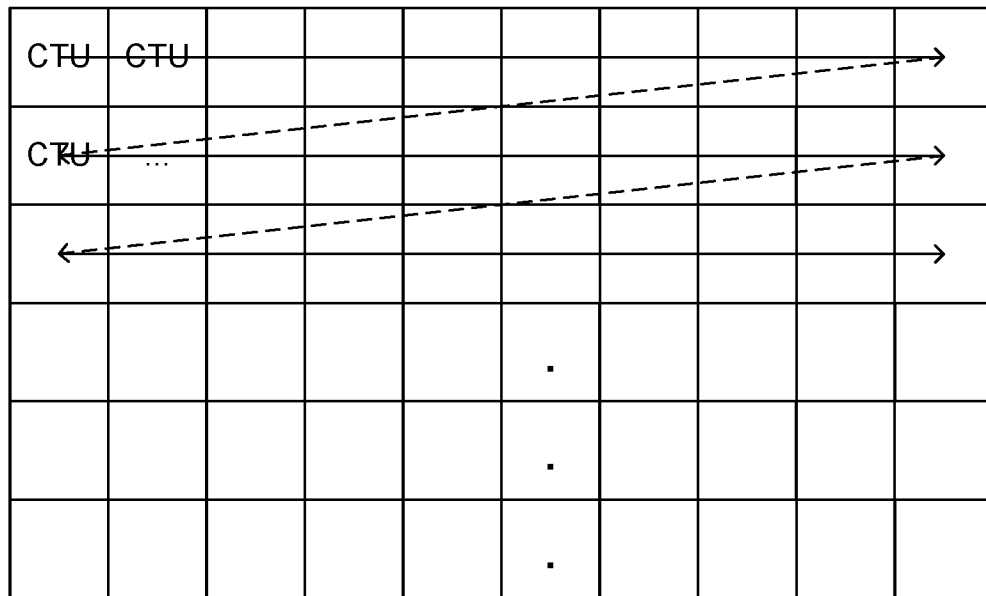


FIG. 4A

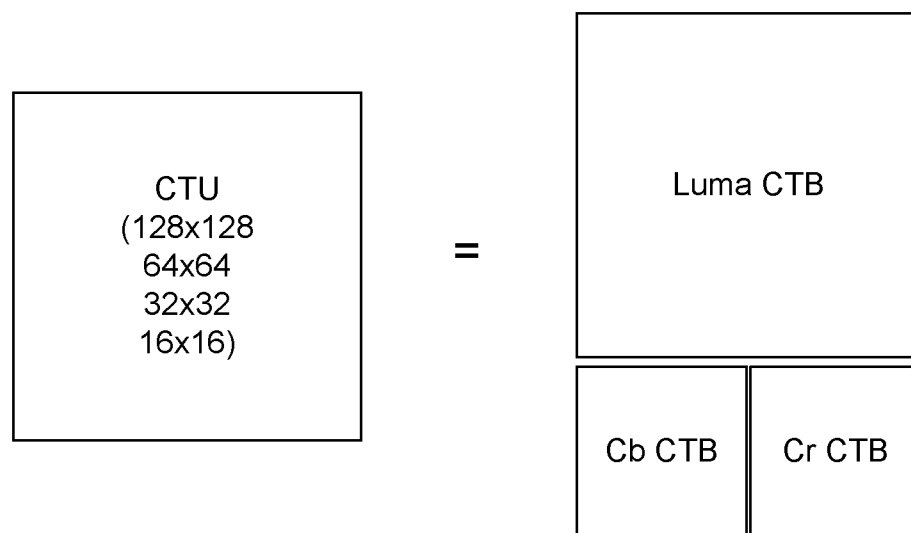


FIG. 4B

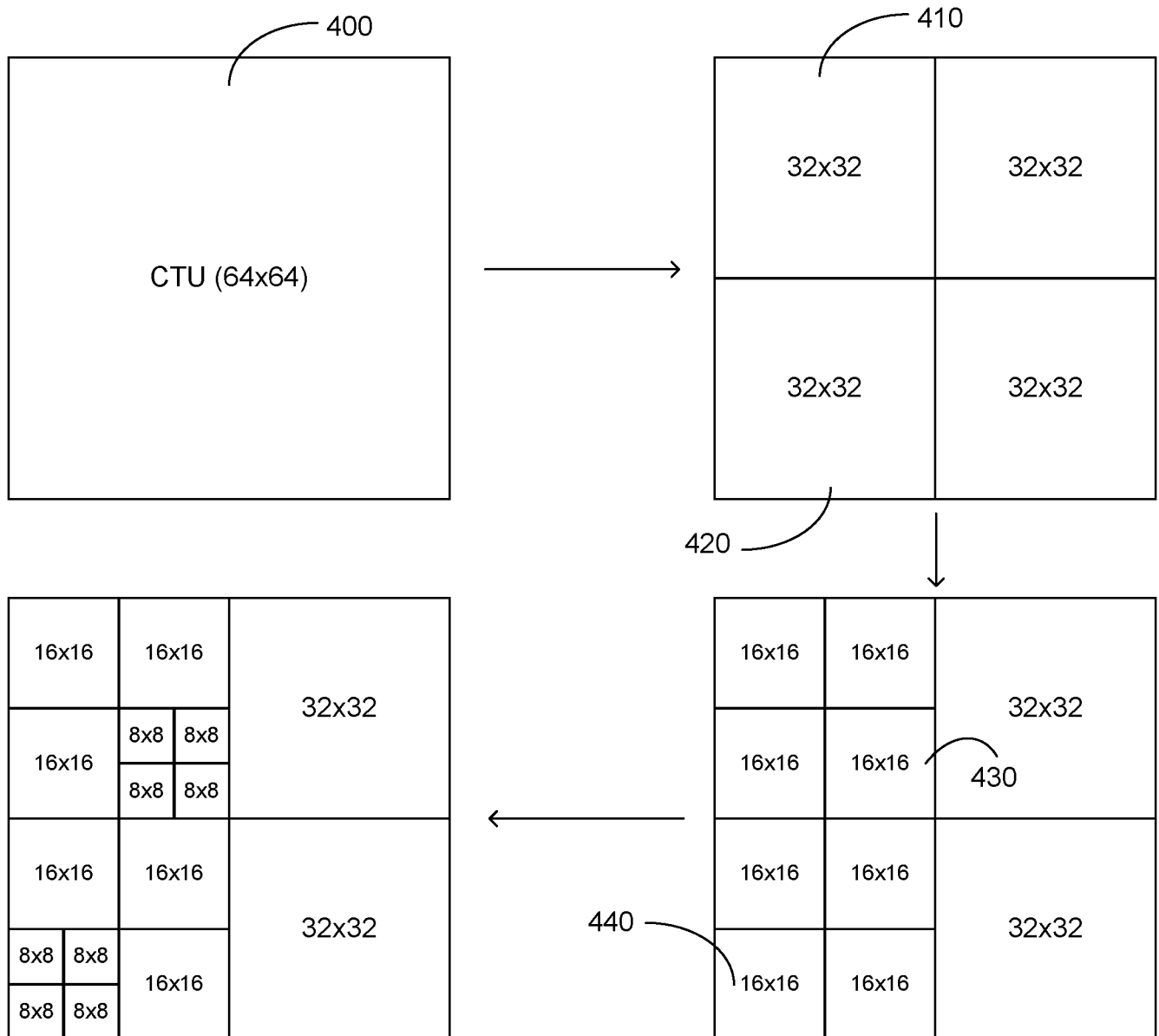


FIG. 4C

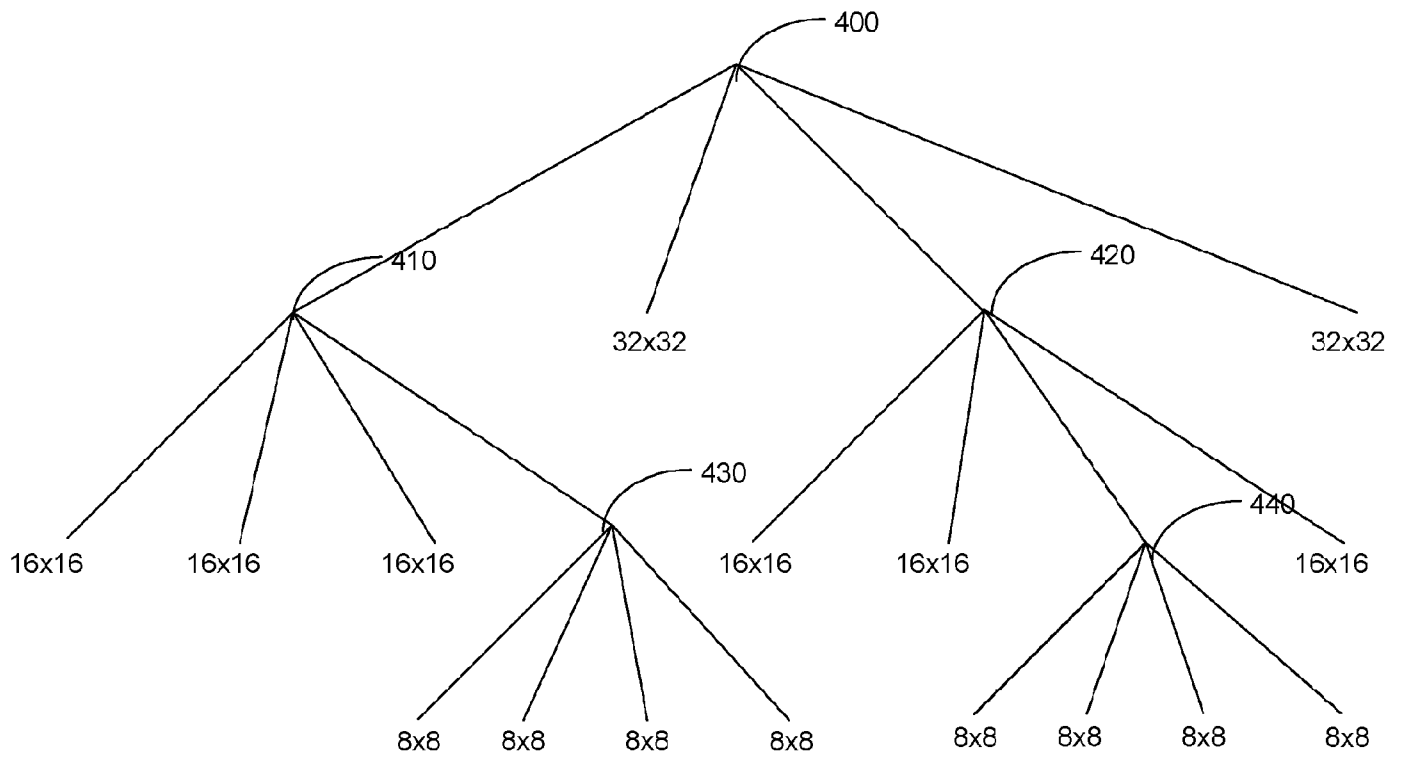


FIG. 4D

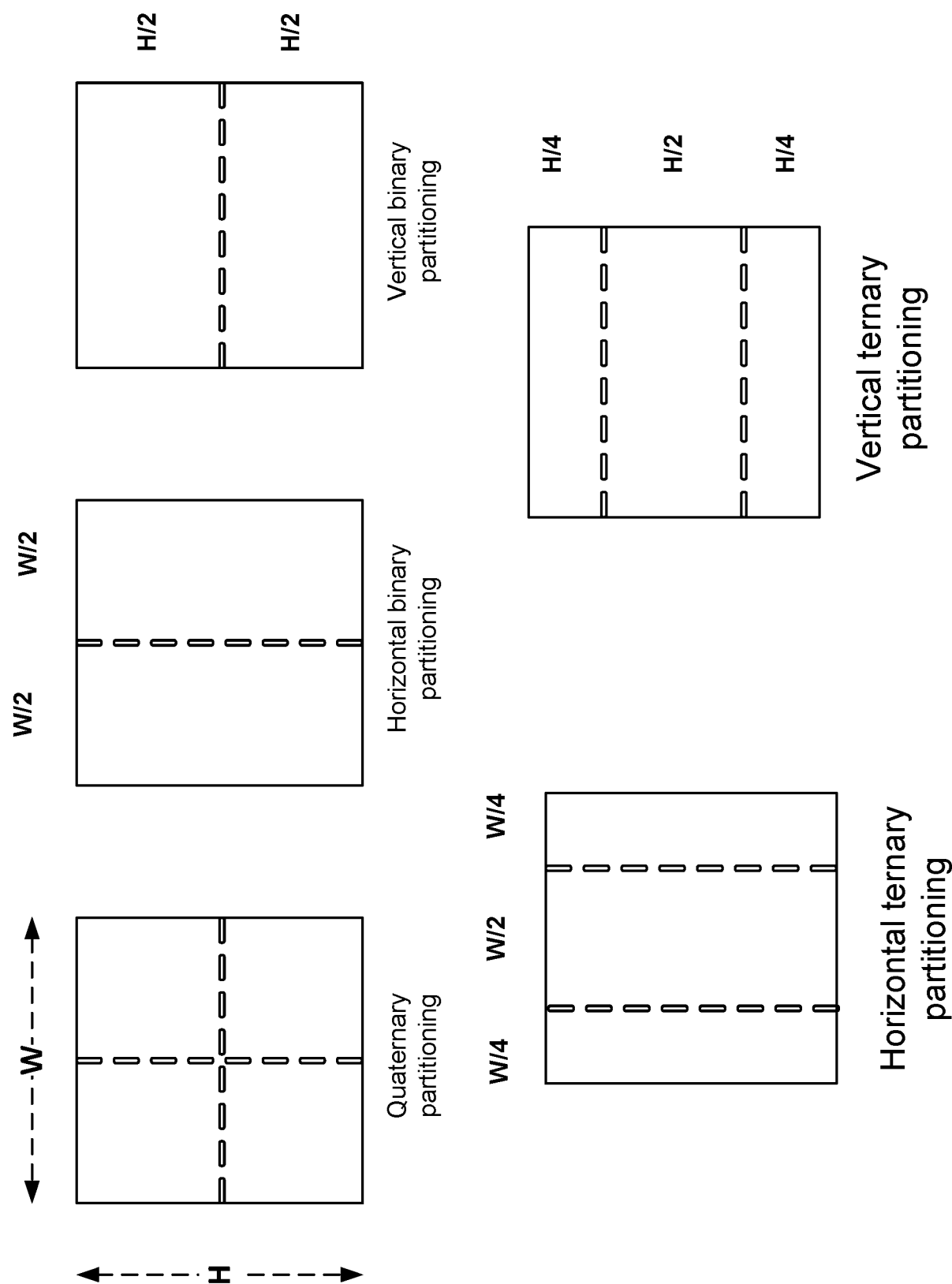


FIG. 4E

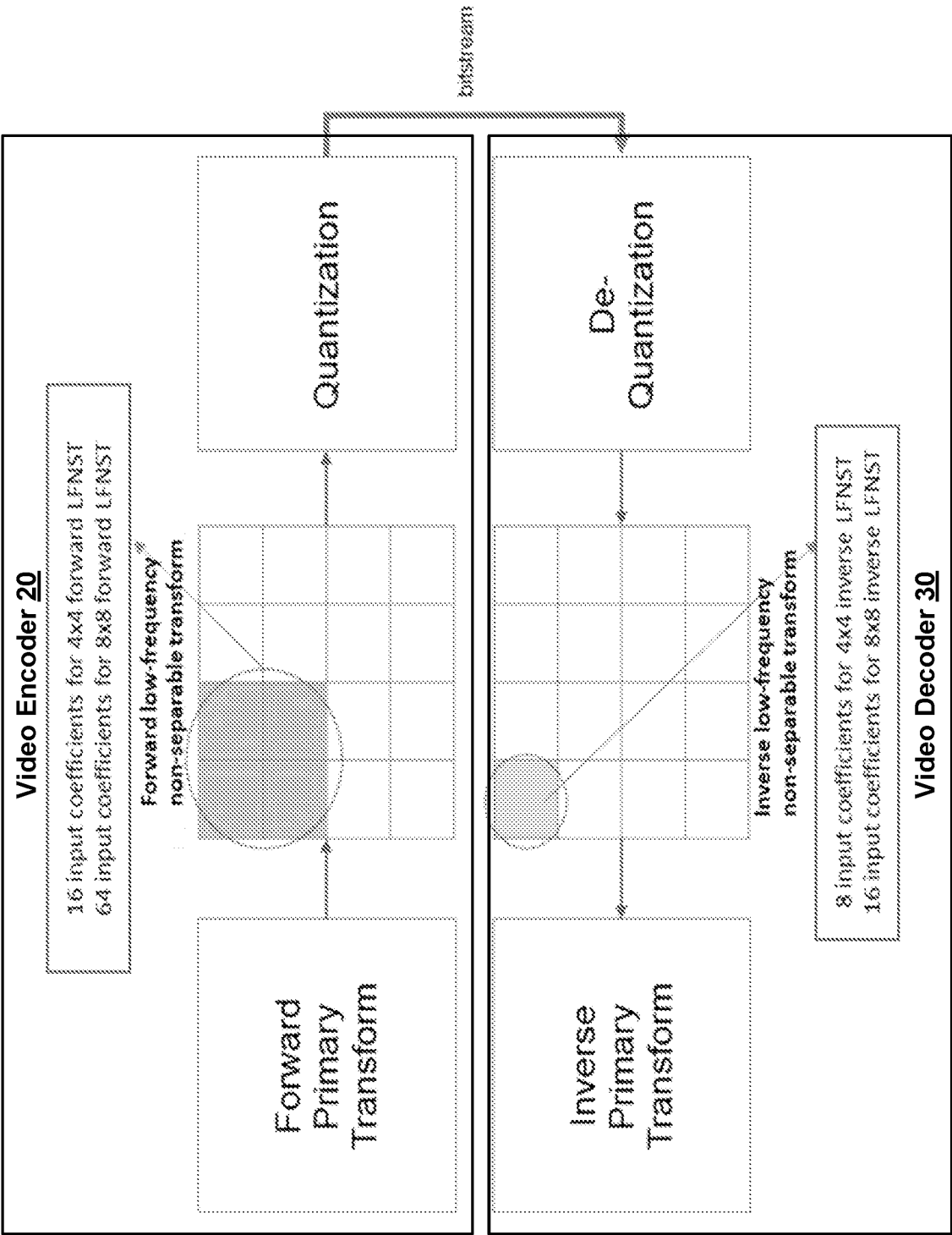


FIG. 5

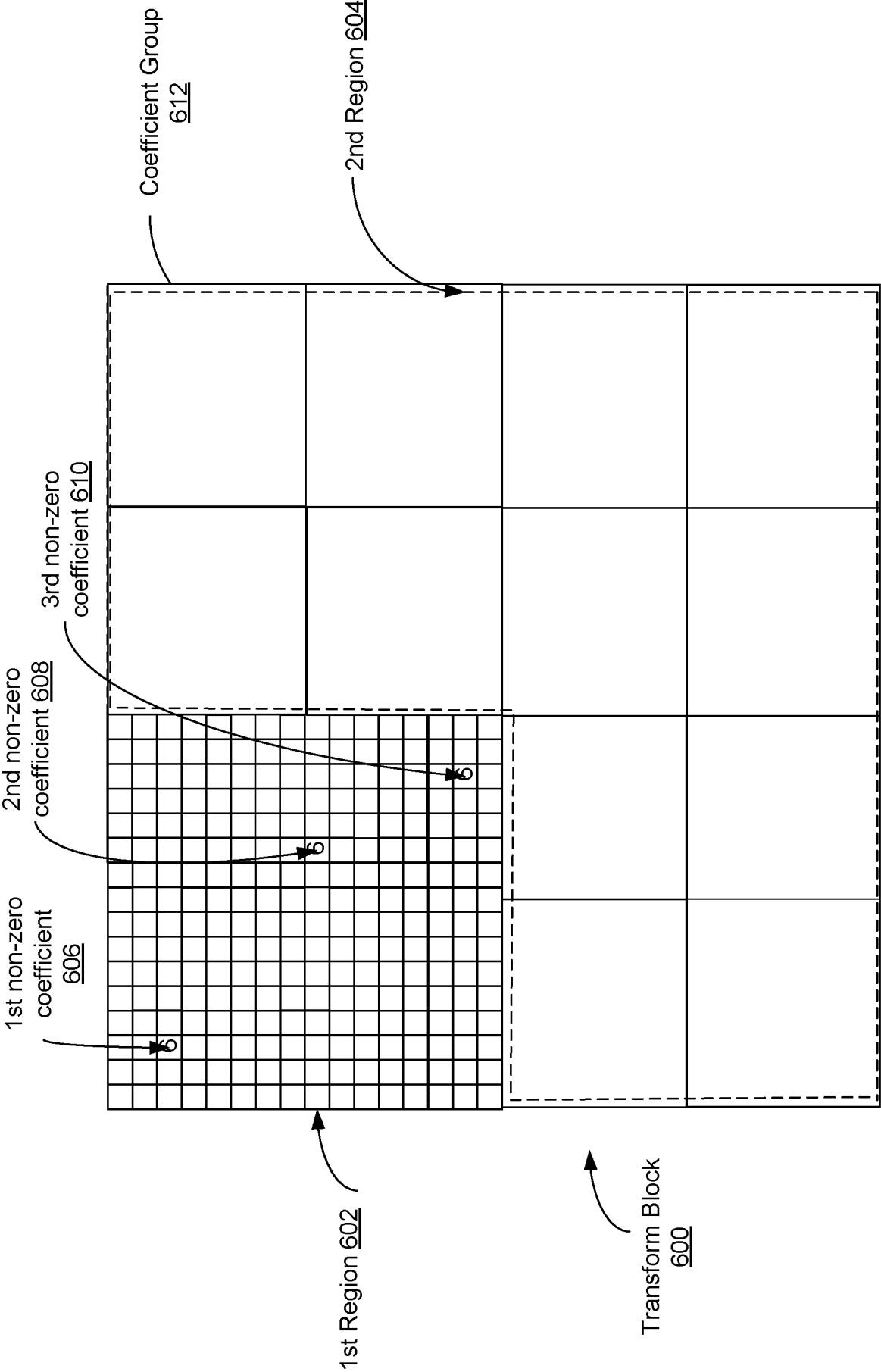
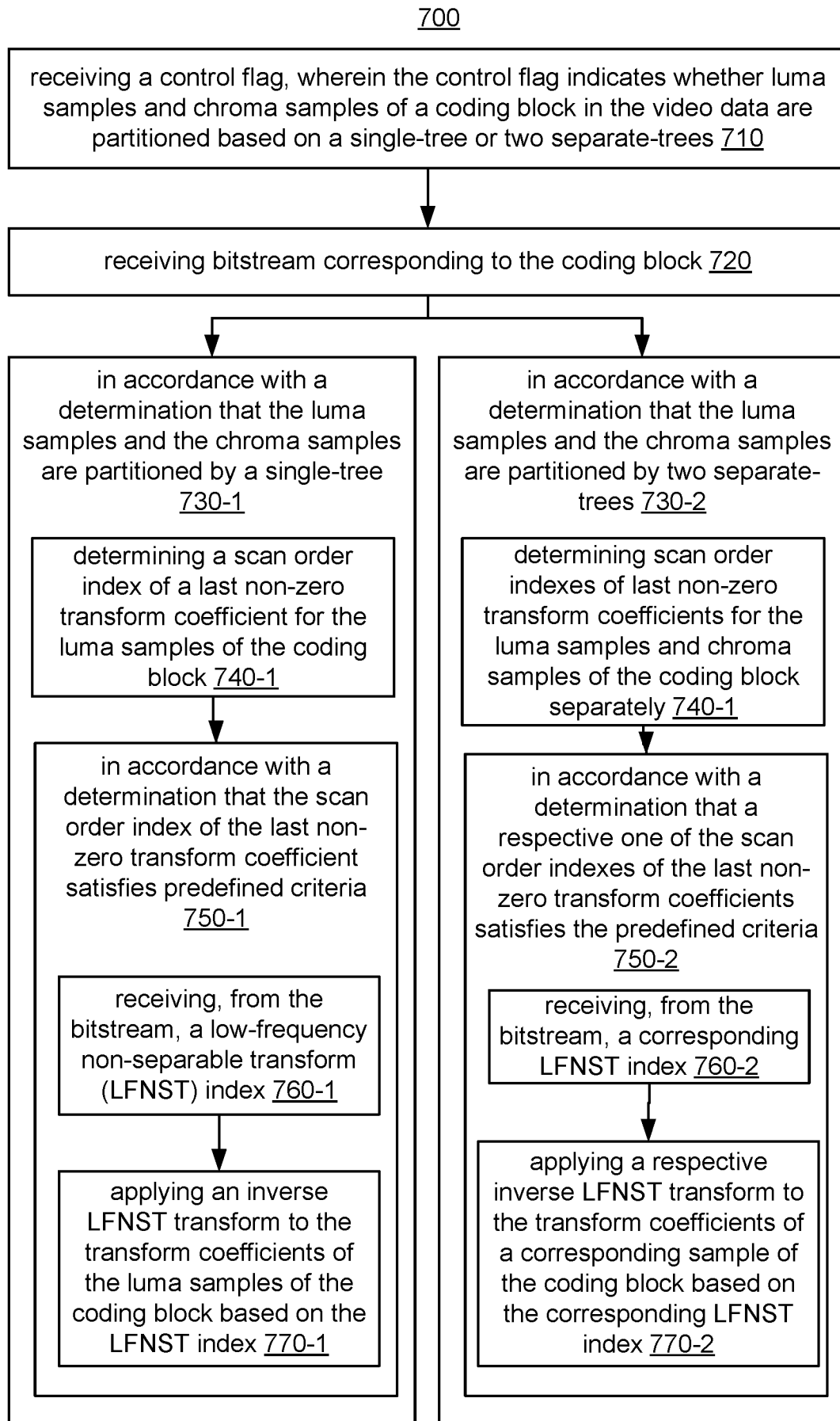


FIG. 6

**FIG. 7**

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US2021/015504

A. CLASSIFICATION OF SUBJECT MATTER		
H04N 19/60(2014.01)i; H04N 19/70(2014.01)i; H04N 19/186(2014.01)i; H04N 19/119(2014.01)i; H04N 19/122(2014.01)i; H04N 19/423(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 19/60(2014.01); H04N 19/44(2014.01); H04N 19/625(2014.01); H04N 19/70(2014.01); H04N 19/96(2014.01)		
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: decoding, non-zero, coefficient, tree, luma, chroma, low-frequency non-separable transform, index		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	BENJAMIN BROSS et al., 'Versatile Video Coding (Draft 8)', JVET-Q2001-vB, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 17th Meeting: Brussels, BE, pp. 1-481, 27 January 2020 pages 25, 69-73, 81-82, 162, 310-314	1-9
Y	JANI LAINEMA, 'CE6-related: Latency reduction for LFNST signalling', JVET-P0196-v3, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 16th Meeting: Geneva, CH, pp. 1-4, 04 October 2019 pages 1-4	1-9
A	US 2019-0387241 A1 (LG ELECTRONICS INC.) 19 December 2019 (2019-12-19) paragraphs [0106]-[0107], [0396]	1-9
A	US 2019-0306538 A1 (QUALCOMM INCORPORATED) 03 October 2019 (2019-10-03) claims 1-8; and figures 10A-12	1-9
A	US 2015-0092862 A1 (ARRIS ENTERPRISES, INC.) 02 April 2015 (2015-04-02) claims 1-8; and figures 13-15B	1-9
☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "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		Date of mailing of the international search report
13 May 2021		17 May 2021
Name and mailing address of the ISA/KR		Authorized officer
Korean Intellectual Property Office 189 Cheongsa-ro, Seo-gu, Daejeon 35208, Republic of Korea		KIM, Sung Hoon
Facsimile No. +82-42-481-8578		Telephone No. +82-42-481-8710

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/US2021/015504

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
US	2019-0387241	A1	19 December 2019	CN	111937400	A	13 November 2020
				KR	10-2020-0078649	A	01 July 2020
				WO	2019-235797	A1	12 December 2019
US	2019-0306538	A1	03 October 2019	CN	111903129	A	06 November 2020
				EP	3772262	A1	10 February 2021
				US	10972758	B2	06 April 2021
				WO	2019-195280	A1	10 October 2019
US	2015-0092862	A1	02 April 2015	US	10291934	B2	14 May 2019
				WO	2015-051011	A1	09 April 2015