



(51) International Patent Classification:

H04N 19/50 (2014.01)

(21) International Application Number:

PCT/CN2022/085056

(22) International Filing Date:

02 April 2022 (02.04.2022)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

PCT/CN2021/085290

02 April 2021 (02.04.2021)

CN

(71) Applicants: **BEIJING BYTEDANCE NETWORK**

TECHNOLOGY CO., LTD. [CN/CN]; Room B-0035, 2/F, No.3 Building, No.30, Shixing Road, Shijingshan District, Beijing 100041 (CN). **BYTEDANCE INC.** [US/US]; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

(72) Inventors: **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **ZHANG, Li**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

ward, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **DENG, Zhipin**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN). **ZHANG, Na**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN). **WANG, Yang**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN).

(74) Agent: **LIU, SHEN & ASSOCIATES**; 10th Floor, Building 1, 10 Caihefang Road, Haidian District, Beijing 100080 (CN).

(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, JM, 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,

(54) Title: CONSTRAINTS ON PARTITIONING OF VIDEO BLOCKS

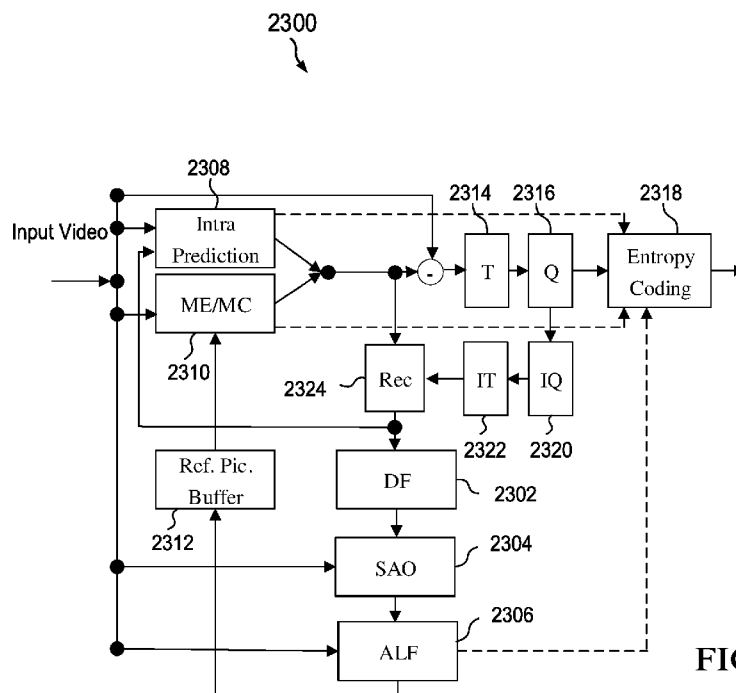


FIG. 21

(57) Abstract: A mechanism for processing video data is disclosed. The method includes determining, for a conversion between a visual media data and a bitstream, whether a partitioning scheme is enabled for coding of the video media data according to a rule. The method includes performing the conversion based on determined rule.

TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

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

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

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

Constraints on Partitioning of Video Blocks

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This patent application claims the benefit of International Application No. PCT/CN2021/085290 filed April 2, 2021 by Kai Zhang, et al., and titled “Constraints on Partitioning of Video Blocks,” which is hereby incorporated by reference.

TECHNICAL FIELD

[0002] This patent document relates to generation, storage, and consumption of digital audio video media information in a file format.

BACKGROUND

[0003] Digital video accounts for the largest bandwidth used on the Internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, the bandwidth demand for digital video usage is likely to continue to grow.

SUMMARY

[0004] A first aspect relates to a method for processing video data comprising determining, for a conversion between a visual media data and a bitstream, whether a partitioning scheme is enabled for coding of the video media data according to a rule and performing the conversion based on the determined rule.

[0005] Optionally, another implementation of the aspect provides that the rule is based on a color characteristic of the visual media data.

[0006] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the color characteristic comprises at least one of a color component of the visual media data or whether the visual media data is coded using separate plane coding.

[0007] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the rule specifies a type of unsymmetrical tree partitioning disallowed due to a video block being a luma block or a chroma block.

[0008] Optionally, in any of the preceding aspects, another implementation of the aspect provides that a video unit of the visual media data has a dimension $W \times H$, where W is the width of the video unit in pixels and H is the height of the video unit in pixels, and wherein the rule is based

on the dimensions of the video block and/or another partitioning scheme that is a disallowed partitioning scheme for the video block.

[0009] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the partitioning scheme is a horizontal or a vertical unsymmetric binary tree (UBT) or a horizontal or a vertical unsymmetric quad tree (UQT) or a horizontal or a vertical extended ternary tree (ETT) partition.

[0010] Optionally, in any of the preceding aspects, another implementation of the aspect provides that at least one of: the disallowed partitioning scheme includes binary tree (BT), ternary tree (TT), UBT, UQT, ETT, or quadtree (QT); UBT, UQT or ETT is not allowed responsive to W or H being larger than a maximum allowed size for a split using the disallowed partitioning scheme; the rule specifies that the partitioning scheme is disabled in response to W or H satisfying one or more conditions; or the partitioning scheme includes unsymmetric binary tree (UBT), unsymmetric quadtree (UQT) or extended ternary tree (ETT), multi-type tree (MTT), binary tree (BT), or ternary tree (TT).

[0011] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the one or more conditions are based on thresholds $T1 = t1 \gg S1$ and $T2 = t2 \gg S2$.

[0012] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the one or more conditions are dependent on a color format of the visual media data or a color component of a video unit of the visual media data or decoded information of the video unit.

[0013] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the rule specifies that in case W is greater than T1 and H is smaller than or equal to T2, the partitioning scheme is disabled for the video unit.

[0014] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the rule specifies that in case H is greater than T1 and W is smaller than or equal to T2, the partitioning scheme is disabled for the video unit.

[0015] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the rule is based on a multi-type tree depth (MTT-depth) or a binary tree depth (BT-depth) of a video unit of the visual media data.

[0016] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the partitioning scheme includes unsymmetric binary tree (UBT), unsymmetric quadtree (UQT) or extended ternary tree (ETT).

[0017] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the MTT-depth of the partitioning scheme is derived from a maximum allowed depth of another partitioning scheme comprising quadtree (QT), multi-tree type (MTT), binary tree (BT) or ternary tree (TT).

[0018] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the bitstream indicates the MTT depth.

[0019] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the rule specifies at least one of: that the partitioning scheme is disallowed responsive to the current MTT-depth being larger than a maximum allowed depth of another partitioning scheme; or that the partitioning scheme is disallowed responsive to the current MTT-depth being smaller than a minimum allowed depth of another partitioning scheme.

[0020] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the rule is based on whether a local-dual tree is used for a video unit of the visual media data.

[0021] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the partitioning scheme includes unsymmetric binary tree (UBT), unsymmetric quadtree (UQT) or extended ternary tree (ETT).

[0022] Optionally, in any of the preceding aspects, another implementation of the aspect provides that the rule specifies that the partitioning scheme is disallowed in case that the local-dual tree is used for the current video block or that UBT, UQT, or ETT is not allowed responsive to W or H being smaller than a minimum allowed size for a split using the disallowed partitioning scheme.

[0023] A second aspect relates to a non-transitory computer readable medium comprising a computer program product for use by a video coding device, the computer program product comprising computer executable instructions stored on the non-transitory computer readable medium such that when executed by a processor cause the video coding device to perform the method of any of the preceding aspects.

[0024] A third aspect relates to an apparatus for processing video data comprising: a processor; and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to perform the method of any of the preceding aspects.

[0025] For the purpose of clarity, any one of the foregoing embodiments may be combined with any one or more of the other foregoing embodiments to create a new embodiment within the scope of the present disclosure.

[0026] These and other features will be more clearly understood from the following detailed description taken in conjunction with the accompanying drawings and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0027] For a more complete understanding of this disclosure, reference is now made to the following brief description, taken in connection with the accompanying drawings and detailed description, wherein like reference numerals represent like parts.

[0028] FIG. 1 is a schematic diagram of an example coding and decoding (codec) for video coding.

[0029] FIG. 2 is a schematic diagram of example macroblock partitions.

[0030] FIG. 3 is a schematic diagram of example modes for partitioning coding blocks, for example according to High Efficiency Video Coding (HEVC).

[0031] FIG. 4 is a schematic diagram of example method for partitioning a picture for coding residual.

[0032] FIG. 5 is a schematic diagram of example method for partitioning a picture, for example according to a quad tree binary tree (QTBT) structure.

[0033] FIG. 6 is a schematic diagram of example partitioning structures used in Versatile Video Coding (VVC).

[0034] FIG. 7 is a schematic diagram of example extended quad tree (EQT) partitioning structures.

[0035] FIG. 8 is a schematic diagram of example flexible tree (FT) partitioning structures.

[0036] FIG. 9 is a schematic diagram of example generalized TT (GTT) partitioning structures.

[0037] FIG. 10 is a schematic diagram of example boundary partitioning tree.

[0038] FIG. 11 is a schematic diagram of example partitioning structures used in Audio and Video Coding Standard (AVS) part three (AVS-3.0).

[0039] FIG. 12 is a schematic diagram of example Unsymmetric Quad-Tree (UQT) partitioning structures.

[0040] FIG. 13 is a schematic diagram of example Extended Ternary-Tree (ETT) partitioning structures.

[0041] FIG. 14 is a schematic diagram of example 1/4 Unsymmetric Binary Tree (UBT) partitioning structures.

[0042] FIG. 15 is a block diagram showing an example video processing system.

[0043] FIG. 16 is a block diagram of an example video processing apparatus.

[0044] FIG. 17 is a flowchart for an example method of processing video data.

[0045] FIG. 18 is a block diagram that illustrates an example video coding system.

[0046] FIG. 19 is a block diagram that illustrates an example encoder.

[0047] FIG. 20 is a block diagram that illustrates an example decoder.

[0048] FIG. 21 is a schematic diagram of an example encoder.

DETAILED DESCRIPTION

[0049] It should be understood at the outset that although an illustrative implementation of one or more embodiments are provided below, the disclosed systems and/or methods may be implemented using any number of techniques, whether currently known or yet to be developed. The disclosure should in no way be limited to the illustrative implementations, drawings, and techniques illustrated below, including the exemplary designs and implementations illustrated and described herein, but may be modified within the scope of the appended claims along with their full scope of equivalents.

[0050] Versatile Video Coding (VVC), also known as H.266, terminology is used in some description only for ease of understanding and not for limiting scope of the disclosed techniques. As such, the techniques described herein are applicable to other video codec protocols and designs also. In the present document, editing changes are shown to text by bold italics indicating cancelled text and bold underline indicating added text, with respect to the VVC specification or International Organization for Standardization (ISO) base media file format (ISO/BMFF) file format specification.

[0051] This document is related to image/video coding, and more particularly to transforms on some special kinds of blocks. The disclosed mechanisms may be applied to the video coding standards such as High Efficiency Video Coding (HEVC) and/or Versatile Video Coding (VVC). Such mechanisms may also be applicable to other video coding standards and/or video codecs.

[0052] Video coding standards have evolved primarily through the development of the International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T) and International Organization for Standardization (ISO)/ International Electrotechnical Commission (IEC) standards. The ITU-T produced a H.261 standard and a H.263 standard, ISO/IEC

produced Motion Picture Experts Group (MPEG) phase one (MPEG-1) and MPEG phase four (MPEG-4) Visual standards, and the two organizations jointly produced the H.262/ MPEG phase two (MPEG-2) Video standard, the H.264/MPEG-4 Advanced Video Coding (AVC) standard, and the H.265/High Efficiency Video Coding (HEVC) standard. Since H.262, the video coding standards are based on a hybrid video coding structure that utilizes a temporal prediction plus a transform coding.

[0053] FIG. 1 is a schematic diagram of an example coding and decoding (codec) for video coding, for example according to HEVC. For example, codec 100 provides functionality to support converting a video file into a bitstream by encoding and/or decoding pictures. Codec 100 is generalized to depict components employed in both an encoder and a decoder. Codec 100 receives a stream of pictures as a video signal 101 and partitions the pictures. Codec 100 then compresses the pictures in the video signal 101 into a coded bitstream when acting as an encoder. When acting as a decoder, codec system 100 generates an output video signal from the bitstream. The codec 100 includes a general coder control component 111, a transform scaling and quantization component 113, an intra-picture estimation component 115, an intra-picture prediction component 117, a motion compensation component 119, a motion estimation component 121, a scaling and inverse transform component 129, a filter control analysis component 127, an in-loop filters component 125, a decoded picture buffer component 123, and a header formatting and context adaptive binary arithmetic coding (CABAC) component 131. Such components are coupled as shown. In FIG. 1, black lines indicate movement of data to be encoded/decoded while dashed lines indicate movement of control data that controls the operation of other components. The components of codec 100 may all be present in the encoder. The decoder may include a subset of the components of codec 100. For example, the decoder may include the intra-picture prediction component 117, the motion compensation component 119, the scaling and inverse transform component 129, the in-loop filters component 125, and the decoded picture buffer component 123. These components are now described.

[0054] The video signal 101 is a captured video sequence that has been partitioned into blocks of pixels by a coding tree. A coding tree employs various split modes to subdivide a block of pixels into smaller blocks of pixels. These blocks can then be further subdivided into smaller blocks. The blocks may be referred to as nodes on the coding tree. Larger parent nodes are split into smaller child nodes. The number of times a node is subdivided is referred to as the depth of the node/coding tree. The divided blocks can be included in coding units (CUs) in some cases. For example, a CU

can be a sub-portion of a CTU that contains a luma block, red difference chroma (Cr) block(s), and a blue difference chroma (Cb) block(s) along with corresponding syntax instructions for the CU. The split modes may include a binary tree (BT), triple tree (TT), and a quad tree (QT) employed to partition a node into two, three, or four child nodes, respectively, of varying shapes depending on the split modes employed. The video signal 101 is forwarded to the general coder control component 111, the transform scaling and quantization component 113, the intra-picture estimation component 115, the filter control analysis component 127, and the motion estimation component 121 for compression.

[0055] The general coder control component 111 is configured to make decisions related to coding of the images of the video sequence into the bitstream according to application constraints. For example, the general coder control component 111 manages optimization of bitrate/bitstream size versus reconstruction quality. Such decisions may be made based on storage space/bandwidth availability and image resolution requests. The general coder control component 111 also manages buffer utilization in light of transmission speed to mitigate buffer underrun and overrun issues. To manage these issues, the general coder control component 111 manages partitioning, prediction, and filtering by the other components. For example, the general coder control component 111 may increase compression complexity to increase resolution and increase bandwidth usage or decrease compression complexity to decrease resolution and bandwidth usage. Hence, the general coder control component 111 controls the other components of codec 100 to balance video signal reconstruction quality with bit rate concerns. The general coder control component 111 creates control data, which controls the operation of the other components. The control data is also forwarded to the header formatting and CABAC component 131 to be encoded in the bitstream to signal parameters for decoding at the decoder.

[0056] The video signal 101 is also sent to the motion estimation component 121 and the motion compensation component 119 for inter-prediction. A video unit (e.g., a picture, a slice, a CTU, etc.) of the video signal 101 may be divided into multiple blocks. Motion estimation component 121 and the motion compensation component 119 perform inter-predictive coding of the received video block relative to one or more blocks in one or more reference pictures to provide temporal prediction. Codec system 100 may perform multiple coding passes, e.g., to select an appropriate coding mode for each block of video data.

[0057] Motion estimation component 121 and motion compensation component 119 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation, performed by motion estimation component 121, is the process of generating motion vectors, which estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a coded object in a current block relative to a reference block. A reference block is a block that is found to closely match the block to be coded, in terms of pixel difference. Such pixel differences may be determined by sum of absolute difference (SAD), sum of square difference (SSD), or other difference metrics. HEVC employs several coded objects including a CTU, coding tree blocks (CTBs), and CUs. For example, a CTU can be divided into CTBs, which can then be divided into CBs for inclusion in CUs. A CU can be encoded as a prediction unit (PU) containing prediction data and/or a transform unit (TU) containing transformed residual data for the CU. The motion estimation component 121 generates motion vectors, PUs, and TUs by using a rate-distortion analysis as part of a rate distortion optimization process. For example, the motion estimation component 121 may determine multiple reference blocks, multiple motion vectors, etc. for a current block/frame, and may select the reference blocks, motion vectors, etc. having the best rate-distortion characteristics. The best rate-distortion characteristics balance both quality of video reconstruction (e.g., amount of data loss by compression) with coding efficiency (e.g., size of the final encoding).

[0058] In some examples, codec 100 may calculate values for sub-integer pixel positions of reference pictures stored in decoded picture buffer component 123. For example, video codec 100 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference picture. Therefore, motion estimation component 121 may perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision. The motion estimation component 121 calculates a motion vector for a PU of a video block in an inter-coded slice by comparing the position of the PU to the position of a reference block of a reference picture. Motion estimation component 121 outputs the calculated motion vector as motion data to header formatting and CABAC component 131 for encoding and to the motion compensation component 119.

[0059] Motion compensation, performed by motion compensation component 119, may involve fetching or generating a reference block based on the motion vector determined by motion estimation component 121. Motion estimation component 121 and motion compensation component 119 may be functionally integrated, in some examples. Upon receiving the motion vector for the PU of the

current video block, motion compensation component 119 may locate the reference block to which the motion vector points. A residual video block is then formed by subtracting pixel values of the reference block from the pixel values of the current block being coded, forming pixel difference values. In general, motion estimation component 121 performs motion estimation relative to luma components, and motion compensation component 119 uses motion vectors calculated based on the luma components for both chroma components and luma components. The reference block and residual block are forwarded to transform scaling and quantization component 113.

[0060] The video signal 101 is also sent to intra-picture estimation component 115 and intra-picture prediction component 117. As with motion estimation component 121 and motion compensation component 119, intra-picture estimation component 115 and intra-picture prediction component 117 may be highly integrated, but are illustrated separately for conceptual purposes. The intra-picture estimation component 115 and intra-picture prediction component 117 intra-predict a current block relative to blocks in a current picture, as an alternative to the inter-prediction performed by motion estimation component 121 and motion compensation component 119 between pictures, as described above. In particular, the intra-picture estimation component 115 determines an intra-prediction mode to use to encode a current block. In some examples, intra-picture estimation component 115 selects an appropriate intra-prediction mode to encode a current block from multiple tested intra-prediction modes. The selected intra-prediction modes are then forwarded to the header formatting and CABAC component 131 for encoding.

[0061] For example, the intra-picture estimation component 115 calculates rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and selects the intra-prediction mode having the best rate-distortion characteristics among the tested modes. 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 (e.g., a number of bits) used to produce the encoded block. The intra-picture estimation component 115 calculates 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. In addition, intra-picture estimation component 115 may be configured to code depth blocks of a depth map using a depth modeling mode (DMM) based on rate-distortion optimization (RDO).

[0062] The intra-picture prediction component 117 may generate a residual block from the reference block based on the selected intra-prediction modes determined by intra-picture estimation

component 115 when implemented on an encoder or read the residual block from the bitstream when implemented on a decoder. The residual block includes the difference in values between the reference block and the original block, represented as a matrix. The residual block is then forwarded to the transform scaling and quantization component 113. The intra-picture estimation component 115 and the intra-picture prediction component 117 may operate on both luma and chroma components.

[0063] The transform scaling and quantization component 113 is configured to further compress the residual block. The transform scaling and quantization component 113 applies a transform, such as a discrete cosine transform (DCT), a discrete sine transform (DST), or a conceptually similar transform, to the residual block, producing a video block comprising residual transform coefficient values. Wavelet transforms, integer transforms, sub-band transforms or other types of transforms could also be used. The transform may convert the residual information from a pixel value domain to a transform domain, such as a frequency domain. The transform scaling and quantization component 113 is also configured to scale the transformed residual information, for example based on frequency. Such scaling involves applying a scale factor to the residual information so that different frequency information is quantized at different granularities, which may affect final visual quality of the reconstructed video. The transform scaling and quantization component 113 is also configured to quantize the transform coefficients to further reduce bit rate. The quantization process may 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, the transform scaling and quantization component 113 may then perform a scan of the matrix including the quantized transform coefficients. The quantized transform coefficients are forwarded to the header formatting and CABAC component 131 to be encoded in the bitstream.

[0064] The scaling and inverse transform component 129 applies a reverse operation of the transform scaling and quantization component 113 to support motion estimation. The scaling and inverse transform component 129 applies inverse scaling, transformation, and/or quantization to reconstruct the residual block in the pixel domain, e.g., for later use as a reference block for another current block. The motion estimation component 121 and/or motion compensation component 119 may calculate a further reference block by adding the residual block back to a previous reference block for use in motion estimation of a later block/frame. Filters are applied to the reconstructed reference blocks to mitigate artifacts created during scaling, quantization, and transform. Such

artifacts could otherwise cause inaccurate prediction (and create additional artifacts) when subsequent blocks are predicted.

[0065] The filter control analysis component 127 and the in-loop filters component 125 apply the filters to the residual blocks and/or to reconstructed picture blocks. For example, the transformed residual block from the scaling and inverse transform component 129 may be combined with a corresponding reference block from intra-picture prediction component 117 and/or motion compensation component 119 to reconstruct the original image block. The filters may then be applied to the reconstructed image block. In some examples, the filters may instead be applied to the residual blocks. As with other components in FIG. 1, the filter control analysis component 127 and the in-loop filters component 125 are highly integrated and may be implemented together, but are depicted separately for conceptual purposes. Filters applied to the reconstructed reference blocks are applied to particular spatial regions and include multiple parameters to adjust how such filters are applied. The filter control analysis component 127 analyzes the reconstructed reference blocks to determine where such filters should be applied and sets corresponding parameters. Such data is forwarded to the header formatting and CABAC component 131 as filter control data for encoding. The in-loop filters component 125 applies such filters based on the filter control data. The filters may include a deblocking filter, a noise suppression filter, a SAO filter, and an adaptive loop filter. Such filters may be applied in the spatial/pixel domain (e.g., on a reconstructed pixel block) or in the frequency domain, depending on the example.

[0066] When operating as an encoder, the filtered reconstructed image block, residual block, and/or prediction block are stored in the decoded picture buffer component 123 for later use in motion estimation as discussed above. When operating as a decoder, the decoded picture buffer component 123 stores and forwards the reconstructed and filtered blocks toward a display as part of an output video signal. The decoded picture buffer component 123 may be any memory device capable of storing prediction blocks, residual blocks, and/or reconstructed image blocks.

[0067] The header formatting and CABAC component 131 receives the data from the various components of codec 100 and encodes such data into a coded bitstream for transmission toward a decoder. Specifically, the header formatting and CABAC component 131 generates various headers to encode control data, such as general control data and filter control data. Further, prediction data, including intra-prediction and motion data, as well as residual data in the form of quantized transform coefficient data are all encoded in the bitstream. The final bitstream includes all information desired

by the decoder to reconstruct the original partitioned video signal 101. Such information may also include intra-prediction mode index tables (also referred to as codeword mapping tables), definitions of encoding contexts for various blocks, indications of most probable intra-prediction modes, an indication of partition information, etc. Such data may be encoded by employing entropy coding. For example, the information may be encoded by employing context adaptive variable length coding (CAVLC), CABAC, syntax-based context-adaptive binary arithmetic coding (SBAC), probability interval partitioning entropy (PIPE) coding, or another entropy coding technique. Following the entropy coding, the coded bitstream may be transmitted to another device (e.g., a video decoder) or archived for later transmission or retrieval.

[0068] In order to encode and/or decode a picture as described above, the picture is first partitioned. FIG. 2 is a schematic diagram of example macroblock partitions 200, which can be created by a partition tree structure pursuant to H.264/AVC. The core of the coding layer in such standards is the macroblock, containing a 16×16 block of luma samples and, in the case of 4:2:0 color sampling, two corresponding 8×8 blocks of chroma samples. An intra-coded block uses spatial prediction to exploit spatial correlation among pixels. Two partitions are defined for an intra-coded block, namely a 16×16 sub-block and 4×4 sub-block. An inter-coded block uses temporal prediction, instead of spatial prediction, by estimating motion among pictures. Motion can be estimated independently for either a 16×16 macroblock or any sub-macroblock partitions. An inter-coded block can be partitioned into a 16×8 sub-block, an 8×16 sub-block, an 8×8 sub-block, an 8×4 sub-block, a 4×8 sub-block, and/or a 4×4 sub-block. All such values are measured in a number of samples. A Sample is a luma (light) value or chroma (color) value at a pixel.

[0069] FIG. 3 is a schematic diagram of example modes 300 for partitioning coding blocks, for example according to HEVC. In HEVC, a picture is partitioned into CTUs. A CTU is split into CUs by using a quadtree structure denoted as a coding tree to adapt to various local characteristics. The decision whether to code a picture area using inter-picture (temporal) or intra-picture (spatial) prediction is made at the CU level. Each CU can be further split into one, two, or four PUs according to the PU splitting type. Inside one PU, the same prediction process is applied and the relevant information is transmitted to the decoder on a PU basis. After obtaining the residual block by applying the prediction process based on the PU splitting type, a CU can be partitioned into transform units (TUs) according to another quadtree structure similar to the coding tree for the CU. One feature of the HEVC structure is that HEVC has multiple partition conceptions including CU, PU, and TU.

[0070] Various features involved in hybrid video coding using HEVC are highlighted as follows. HEVC includes the CTU, which is analogous to the macroblock in AVC. The CTU has a size selected by the encoder and can be larger than a macroblock. The CTU includes a luma coding tree block (CTB), corresponding chroma CTBs, and syntax elements. The size of a luma CTB, denoted as $L \times L$, can be chosen as $L = 16, 32$, or 64 samples with the larger sizes resulting in better compression. HEVC then supports a partitioning of the CTBs into smaller blocks using a tree structure and quadtree-like signaling.

[0071] The quadtree syntax of the CTU specifies the size and positions of corresponding luma and chroma CBs. The root of the quadtree is associated with the CTU. Hence, the size of the luma CTB is the largest supported size for a luma CB. The splitting of a CTU into luma and chroma CBs is signaled jointly. One luma CB and two chroma CBs, together with associated syntax, form a coding unit (CU). A CTB may contain only one CU or may be split to form multiple CUs. Each CU has an associated partitioning into prediction units (PUs) and a tree of transform units (TUs). The decision of whether to code a picture area using inter picture or intra picture prediction is made at the CU level. A PU partitioning structure has a root at the CU level. Depending on the basic prediction-type decision, the luma and chroma CBs can then be further split in size and predicted from luma and chroma prediction blocks (PBs) according to modes 300. HEVC supports variable PB sizes from 64×64 down to 4×4 samples. As shown, modes 300 can split a CB of size M pixels by M pixels into an $M \times M$ block, a $M/2 \times M$ block, a $M \times M/2$ block, a $M/2 \times M/2$ block, a $M/4 \times M$ (left) block, a $M/4 \times M$ (right) block, a $M \times M/4$ (up) block, and/or a $M \times M/4$ (down) block. It should be noted that the modes 300 for splitting CBs into PBs are subject to size constraints. Further, only $M \times M$ and $M/2 \times M/2$ are supported for intra picture predicted CBs.

[0072] FIG. 4 is a schematic diagram of example method 400 for partitioning a picture for coding residual, for example according to HEVC. As noted above, blocks are coded by reference to reference blocks. A difference between values of a current block and the reference blocks is known as the residual. Method 400 is employed to compress the residual. For example, the prediction residual is coded using block transforms. Method 400 employs a TU tree structure 403 to partition a CTB 401 and included CBs for application of transform blocks (TBs). Method 400 illustrates the subdivision of a CTB 401 into CBs and TBs. Solid lines indicate CB boundaries and dotted lines indicate TB boundaries. The TU tree structure 403 is an example quadtree that partitions the CTB 401. A transform, such as discrete cosine transform (DCT), is applied to each TB. The transform

converts the residual into transform coefficients that can be represented using less data than the uncompressed residual. The TU tree structure 403 has a root at the CU level. The luma CB residual area may be identical to the luma TB area or may be further split into smaller luma TBs. The same applies to the chroma TBs. Integer basis transform functions similar to those of a DCT are defined for the square TB sizes 4×4 , 8×8 , 16×16 , and 32×32 . For the 4×4 transform of luma intra picture prediction residuals, an integer transform derived from a form of DST is alternatively specified.

[0073] A quadtree plus binary tree block structure with larger CTUs in Joint Exploration Model (JEM) is discussed below. Joint Video Exploration Team (JVET) was founded by Video Coding Experts group (VCEG) and MPEG to explore video coding technologies beyond HEVC. JVET has adopted many improvements included such improvements into a reference software named Joint Exploration Model (JEM).

[0074] FIG. 5 is a schematic diagram of example method 500 for partitioning a picture, for example according to a quad tree binary tree (QTBT) structure 501. A tree representation 503 of the QTBT structure 501 is also shown. Unlike the partitioning structures in HEVC, the QTBT structure 501 removes the concepts of multiple partition types. For example, the QTBT structure 501 removes the separation of the CU, PU, and TU concepts, and supports more flexibility for CU partition shapes. In the QTBT structure 501, a CU can have either a square or rectangular shape. In method 500, a CTU is first partitioned by a quadtree structure. The quadtree leaf nodes are further partitioned by a binary tree structure. Symmetric horizontal splitting and symmetric vertical splitting are two splitting types used in the binary tree. The binary tree leaf nodes are called CUs, and that segmentation is used for prediction and transform processing without further partitioning. This causes the CU, PU, and TU to have the same block size in the QTBT structure 501. In the JEM, a CU sometimes includes CBs of different color components. For example, one CU may contain one luma CB and two chroma CBs in the case of unidirectional inter prediction (P) and bidirectional inter prediction (B) slices of the 4:2:0 chroma format. Further, the CU sometimes includes a CB of a single component. For example, one CU may contain only one luma CB or just two chroma CBs in the case of intra prediction (I) slices.

[0075] The following parameters are defined for the QTBT partitioning scheme. The CTU size is the root node size of a quadtree, which is the same concept as in HEVC. Minimum quad tree size (MinQTSIZE) is the minimum allowed quadtree leaf node size. Maximum binary tree size (MaxBTSIZE) is the maximum allowed binary tree root node size. Maximum binary tree depth

(MaxBTDepth) is the maximum allowed binary tree depth. Minimum binary tree size (MinBTSize) is the minimum allowed binary tree leaf node size.

[0076] In one example of the QTBT structure 501, the CTU size is set as 128×128 luma samples with two corresponding 64×64 blocks of chroma samples, the MinQTSIZE is set as 16×16 , the MaxBTSize is set as 64×64 , the MinBTSize (for both width and height) is set as 4×4 , and the MaxBTDepth is set as 4. The quadtree partitioning is applied to the CTU first to generate quadtree leaf nodes. The quadtree leaf nodes may have a size from 16×16 (the MinQTSIZE) to 128×128 (the CTU size). If the leaf quadtree node is 128×128 , the node is not to be further split by the binary tree since the size exceeds the MaxBTSize (e.g., 64×64). Otherwise, the leaf quadtree node can be further partitioned by the binary tree. Therefore, the quadtree leaf node is also the root node for the binary tree and has the binary tree depth as 0. When the binary tree depth reaches MaxBTDepth (e.g., 4), no further splitting is considered. When the binary tree node has width equal to MinBTSize (e.g., 4), no further horizontal splitting is considered. Similarly, when the binary tree node has a height equal to MinBTSize, no further vertical splitting is considered. The leaf nodes of the binary tree are further processed by prediction and transform processing without any further partitioning. In the JEM, the maximum CTU size is 256×256 luma samples.

[0077] Method 500 illustrates an example of block partitioning by using the QTBT structure 501, and tree representation 503 illustrates the corresponding tree representation. The solid lines indicate quadtree splitting and dotted lines indicate binary tree splitting. In each splitting (e.g., non-leaf) node of the binary tree, one flag is signalled to indicate which splitting type (e.g., horizontal or vertical) is used, where 0 indicates horizontal splitting and 1 indicates vertical splitting. For the quadtree splitting, there is no need to indicate the splitting type since quadtree splitting always splits a block both horizontally and vertically to produce 4 sub-blocks with an equal size.

[0078] In addition, the QTBT scheme supports the ability for the luma and chroma to have a separate QTBT structure 501. For example, in P and B slices the luma and chroma CTBs in one CTU share the same QTBT structure 501. However, in I slices the luma CTB is partitioned into CUs by a QTBT structure 501, and the chroma CTBs are partitioned into chroma CUs by another QTBT structure 501. Accordingly, a CU in an I slice can include a coding block of the luma component or coding blocks of two chroma components. Further, a CU in a P or B slice includes coding blocks of all three color components. In HEVC, inter prediction for small blocks is restricted to reduce the memory access of motion compensation, such that bi-prediction is not supported for 4×8 and 8×4

blocks, and inter prediction is not supported for 4×4 blocks. In the QTBT of the JEM, these restrictions are removed.

[0079] Triple-tree partitioning for VVC is now discussed. FIG. 6 is a schematic diagram 600 of example partitioning structures used in VVC. As shown, split types other than quad-tree and binary-tree are supported in VVC. For example, schematic diagram 600 includes a quad tree partition 601, a vertical binary tree partition 603, a horizontal binary tree partition 605, a vertical triple tree partition 607, and a horizontal triple tree partition 609. This approach introduces two triple tree (TT) partitions in addition to the quad tree and binary trees.

[0080] In an example implementation, two levels of trees are employed including a region tree (a quad-tree) and a prediction tree (binary-tree or triple-tree). A CTU is first partitioned by a region tree (RT). A RT leaf may be further split with prediction tree (PT). A PT leaf may also be further split with PT until a max PT depth is reached. A PT leaf is a basic coding unit. The PT may also be called a CU for convenience. In an example implementation, a CU cannot be further split. Prediction and transform are both applied on CU in the same way as JEM. The whole partition structure is named multiple-type-tree.

[0081] Partitioning constraints for VVC

[0082] There are several partitioning constraints in VVC. Some of them are defined in SPS.

[0083] Sequence parameter set RBSP syntax

[0084]

```
[0085] seq_parameter_set_rbsp( ) {      Descriptor
[0086]     sps_seq_parameter_set_id      u(4)
[0087]     sps_video_parameter_set_id    u(4)
[0088]     sps_max_sublayers_minus1     u(3)
[0089]     sps_chroma_format_idc        u(2)
[0090]     sps_log2_ctu_size_minus5     u(2)
[0091]     sps_ptl_dpb_hrd_params_present_flag      u(1)
[0092]     ...
[0093]     sps_log2_min_luma_coding_block_size_minus2      ue(v)
[0094]     sps_partition_constraints_override_enabled_flag  u(1)
[0095]     sps_log2_diff_min_qt_min_cb_intra_slice_luma    ue(v)
[0096]     sps_max_mtt_hierarchy_depth_intra_slice_luma    ue(v)
```

```

[0097]     if( sps_max_mtt_hierarchy_depth_intra_slice_luma != 0 ) {
[0098]         sps_log2_diff_max_bt_min_qt_intra_slice_luma    ue(v)
[0099]         sps_log2_diff_max_tt_min_qt_intra_slice_luma    ue(v)
[0100]     }
[0101]     if( sps_chroma_format_idc != 0 )
[0102]         sps_qtbtt_dual_tree_intra_flagu(1)
[0103]     if( sps_qtbtt_dual_tree_intra_flag ) {
[0104]         sps_log2_diff_min_qt_min_cb_intra_slice_chroma    ue(v)
[0105]         sps_max_mtt_hierarchy_depth_intra_slice_chroma    ue(v)
[0106]         if( sps_max_mtt_hierarchy_depth_intra_slice_chroma != 0 ) {
[0107]             sps_log2_diff_max_bt_min_qt_intra_slice_chroma    ue(v)
[0108]             sps_log2_diff_max_tt_min_qt_intra_slice_chroma    ue(v)
[0109]         }
[0110]     }
[0111]     sps_log2_diff_min_qt_min_cb_inter_slice    ue(v)
[0112]     sps_max_mtt_hierarchy_depth_inter_slice    ue(v)
[0113]     if( sps_max_mtt_hierarchy_depth_inter_slice != 0 ) {
[0114]         sps_log2_diff_max_bt_min_qt_inter_slice    ue(v)
[0115]         sps_log2_diff_max_tt_min_qt_inter_slice    ue(v)
[0116]     }
[0117]     if( CtbSizeY > 32 )
[0118]         sps_max_luma_transform_size_64_flag    u(1)
[0119]     ...
[0120]
[0121]     sps_log2_min_luma_coding_block_size_minus2 plus 2 specifies the minimum luma
coding block size. The value range of sps_log2_min_luma_coding_block_size_minus2 shall be in
the range of 0 to Min( 4, sps_log2_ctu_size_minus5 + 3 ), inclusive.
[0122]     The variables MinCbLog2SizeY, MinCbSizeY, IbcBufWidthY, IbcBufWidthC and
Vsize are derived as follows:
[0123]     MinCbLog2SizeY = sps_log2_min_luma_coding_block_size_minus2 + 2      (43)
[0124]     MinCbSizeY = 1 << MinCbLog2SizeY      (44)

```

[00125] $IbcBufWidthY = 256 * 128 / CtbSizeY$ (45)

[00126] $IbcBufWidthC = IbcBufWidthY / SubWidthC$ (46)

[00127] $VSize = \text{Min}(64, CtbSizeY)$ (47)

[00128] The value of `MinCbSizeY` shall less than or equal to `VSize`.

[00129] The variables `CtbWidthC` and `CtbHeightC`, which specify the width and height, respectively, of the array for each chroma CTB, are derived as follows:

[00130] – If `sps_chroma_format_idc` is equal to 0 (monochrome), `CtbWidthC` and `CtbHeightC` are both set equal to 0.

[00131] – Otherwise, `CtbWidthC` and `CtbHeightC` are derived as follows:

[00132] $CtbWidthC = CtbSizeY / SubWidthC$ (48)

[00133] $CtbHeightC = CtbSizeY / SubHeightC$ (49)

[00134] For `log2BlockWidth` ranging from 0 to 4 and for `log2BlockHeight` ranging from 0 to 4, inclusive, the up-right diagonal scan order array initialization process as specified in subclause 6.5.2 is invoked with $1 \ll \text{log2BlockWidth}$ and $1 \ll \text{log2BlockHeight}$ as inputs, and the output is assigned to `DiagScanOrder[log2BlockWidth][log2BlockHeight]`.

[00135] For `log2BlockWidth` ranging from 0 to 6 and for `log2BlockHeight` ranging from 0 to 6, inclusive, the horizontal and vertical traverse scan order array initialization process as specified in subclause 6.5.3 is invoked with $1 \ll \text{log2BlockWidth}$ and $1 \ll \text{log2BlockHeight}$ as inputs, and the output is assigned to `HorTravScanOrder[log2BlockWidth][log2BlockHeight]` and `VerTravScanOrder[log2BlockWidth][log2BlockHeight]`.

[00136] `sps_partition_constraints_override_enabled_flag` equal to 1 specifies the presence of `ph_partition_constraints_override_flag` in PH syntax structures referring to the SPS. `sps_partition_constraints_override_enabled_flag` equal to 0 specifies the absence of `ph_partition_constraints_override_flag` in PH syntax structures referring to the SPS.

[00137] `sps_log2_diff_min_qt_min_cb_intra_slice_luma` specifies the default difference between the base 2 logarithm of the minimum size in luma samples of a luma leaf block resulting from quadtree splitting of a CTU and the base 2 logarithm of the minimum coding block size in luma samples for luma CUs in slices with `sh_slice_type` equal to 2 (I) referring to the SPS. When `sps_partition_constraints_override_enabled_flag` is equal to 1, the default difference can be overridden by `ph_log2_diff_min_qt_min_cb_intra_slice_luma` present in PH syntax structures referring to the SPS. The value of `sps_log2_diff_min_qt_min_cb_intra_slice_luma` shall be in the

range of 0 to $\text{Min}(6, \text{CtbLog2SizeY}) - \text{MinCbLog2SizeY}$, inclusive. The base 2 logarithm of the minimum size in luma samples of a luma leaf block resulting from quadtree splitting of a CTU is derived as follows:

$$\text{[00138]} \quad \text{MinQtLog2SizeIntraY} = \text{sps_log2_diff_min_qt_min_cb_intra_slice_luma} + \text{MinCbLog2SizeY} \quad (50)$$

[00139] `sps_max_mtt_hierarchy_depth_intra_slice_luma` specifies the default maximum hierarchy depth for coding units resulting from multi-type tree splitting of a quadtree leaf in slices with `sh_slice_type` equal to 2 (I) referring to the SPS. When `sps_partition_constraints_override_enabled_flag` is equal to 1, the default maximum hierarchy depth can be overridden by `ph_max_mtt_hierarchy_depth_intra_slice_luma` present in PH syntax structures referring to the SPS. The value of `sps_max_mtt_hierarchy_depth_intra_slice_luma` shall be in the range of 0 to $2 * (\text{CtbLog2SizeY} - \text{MinCbLog2SizeY})$, inclusive.

[00140] `sps_log2_diff_max_bt_min_qt_intra_slice_luma` specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a binary split and the base 2 logarithm of the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in slices with `sh_slice_type` equal to 2 (I) referring to the SPS. When `sps_partition_constraints_override_enabled_flag` is equal to 1, the default difference can be overridden by `ph_log2_diff_max_bt_min_qt_luma` present in PH syntax structures referring to the SPS. The value of `sps_log2_diff_max_bt_min_qt_intra_slice_luma` shall be in the range of 0 to $\text{CtbLog2SizeY} - \text{MinQtLog2SizeIntraY}$, inclusive. When `sps_log2_diff_max_bt_min_qt_intra_slice_luma` is not present, the value of `sps_log2_diff_max_bt_min_qt_intra_slice_luma` is inferred to be equal to 0.

[00141] `sps_log2_diff_max_tt_min_qt_intra_slice_luma` specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a ternary split and the base 2 logarithm of the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in slices with `sh_slice_type` equal to 2 (I) referring to the SPS. When `sps_partition_constraints_override_enabled_flag` is equal to 1, the default difference can be overridden by `ph_log2_diff_max_tt_min_qt_luma` present in PH syntax structures referring to the SPS. The value of `sps_log2_diff_max_tt_min_qt_intra_slice_luma` shall be in the range of 0 to $\text{Min}(\text{CtbLog2SizeY} - \text{MinQtLog2SizeIntraY}, \text{CtbLog2SizeY} - \text{MinCbLog2SizeY})$, inclusive.

6, CtbLog2SizeY) – $\text{MinQtLog2SizeIntraY}$, inclusive. When $\text{sps_log2_diff_max_tt_min_qt_intra_slice_luma}$ is not present, the value of $\text{sps_log2_diff_max_tt_min_qt_intra_slice_luma}$ is inferred to be equal to 0.

[00142] $\text{sps_qtbtt_dual_tree_intra_flag}$ equal to 1 specifies that, for I slices, each CTU is split into coding units with 64×64 luma samples using an implicit quadtree split, and these coding units are the root of two separate coding tree syntax structure for luma and chroma. $\text{sps_qtbtt_dual_tree_intra_flag}$ equal to 0 specifies separate coding tree syntax structure is not used for I slices. When $\text{sps_qtbtt_dual_tree_intra_flag}$ is not present, it is inferred to be equal to 0. When $\text{sps_log2_diff_max_bt_min_qt_intra_slice_luma}$ is greater than $\text{Min}(6, \text{CtbLog2SizeY}) - \text{MinQtLog2SizeIntraY}$, the value of $\text{sps_qtbtt_dual_tree_intra_flag}$ shall be equal to 0.

[00143] $\text{sps_log2_diff_min_qt_min_cb_intra_slice_chroma}$ specifies the default difference between the base 2 logarithm of the minimum size in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA and the base 2 logarithm of the minimum coding block size in luma samples for chroma CUs with treeType equal to DUAL_TREE_CHROMA in slices with sh_slice_type equal to 2 (I) referring to the SPS. When $\text{sps_partition_constraints_override_enabled_flag}$ is equal to 1, the default difference can be overridden by $\text{ph_log2_diff_min_qt_min_cb_chroma}$ present in PH syntax structures referring to the SPS. The value of $\text{sps_log2_diff_min_qt_min_cb_intra_slice_chroma}$ shall be in the range of 0 to $\text{Min}(6, \text{CtbLog2SizeY}) - \text{MinCbLog2SizeY}$, inclusive. When not present, the value of $\text{sps_log2_diff_min_qt_min_cb_intra_slice_chroma}$ is inferred to be equal to 0. The base 2 logarithm of the minimum size in luma samples of a chroma leaf block resulting from quadtree splitting of a CTU with treeType equal to DUAL_TREE_CHROMA is derived as follows:

$$\text{MinQtLog2SizeIntraC} = \text{sps_log2_diff_min_qt_min_cb_intra_slice_chroma} + \text{MinCbLog2SizeY} \quad (51)$$

[00145] $\text{sps_max_mtt_hierarchy_depth_intra_slice_chroma}$ specifies the default maximum hierarchy depth for chroma coding units resulting from multi-type tree splitting of a chroma quadtree leaf with treeType equal to DUAL_TREE_CHROMA in slices with sh_slice_type equal to 2 (I) referring to the SPS. When $\text{sps_partition_constraints_override_enabled_flag}$ is equal to 1, the default maximum hierarchy depth can be overridden by $\text{ph_max_mtt_hierarchy_depth_chroma}$ present in PH syntax structures referring to the SPS. The value of $\text{sps_max_mtt_hierarchy_depth_intra_slice_chroma}$ shall be in the range of 0 to $2 * (\text{CtbLog2SizeY}$

– MinCbLog2SizeY), inclusive. When not present, the value of $\text{sps_max_mtt_hierarchy_depth_intra_slice_chroma}$ is inferred to be equal to 0.

[00146] $\text{sps_log2_diff_max_bt_min_qt_intra_slice_chroma}$ specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a chroma coding block that can be split using a binary split and the base 2 logarithm of the minimum size (width or height) in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA in slices with sh_slice_type equal to 2 (I) referring to the SPS. When $\text{sps_partition_constraints_override_enabled_flag}$ is equal to 1, the default difference can be overridden by $\text{ph_log2_diff_max_bt_min_qt_chroma}$ present in PH syntax structures referring to the SPS. The value of $\text{sps_log2_diff_max_bt_min_qt_intra_slice_chroma}$ shall be in the range of 0 to $\text{Min}(6, \text{CtbLog2SizeY}) - \text{MinQtLog2SizeIntraC}$, inclusive. When $\text{sps_log2_diff_max_bt_min_qt_intra_slice_chroma}$ is not present, the value of $\text{sps_log2_diff_max_bt_min_qt_intra_slice_chroma}$ is inferred to be equal to 0.

[00147] $\text{sps_log2_diff_max_tt_min_qt_intra_slice_chroma}$ specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a chroma coding block that can be split using a ternary split and the base 2 logarithm of the minimum size (width or height) in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA in slices with sh_slice_type equal to 2 (I) referring to the SPS. When $\text{sps_partition_constraints_override_enabled_flag}$ is equal to 1, the default difference can be overridden by $\text{ph_log2_diff_max_tt_min_qt_chroma}$ present in PH syntax structures referring to the SPS. The value of $\text{sps_log2_diff_max_tt_min_qt_intra_slice_chroma}$ shall be in the range of 0 to $\text{Min}(6, \text{CtbLog2SizeY}) - \text{MinQtLog2SizeIntraC}$, inclusive. When $\text{sps_log2_diff_max_tt_min_qt_intra_slice_chroma}$ is not present, the value of $\text{sps_log2_diff_max_tt_min_qt_intra_slice_chroma}$ is inferred to be equal to 0.

[00148] $\text{sps_log2_diff_min_qt_min_cb_inter_slice}$ specifies the default difference between the base 2 logarithm of the minimum size in luma samples of a luma leaf block resulting from quadtree splitting of a CTU and the base 2 logarithm of the minimum luma coding block size in luma samples for luma CUs in slices with sh_slice_type equal to 0 (B) or 1 (P) referring to the SPS. When $\text{sps_partition_constraints_override_enabled_flag}$ is equal to 1, the default difference can be

overridden by `ph_log2_diff_min_qt_min_cb_inter_slice` present in PH syntax structures referring to the SPS. The value of `sps_log2_diff_min_qt_min_cb_inter_slice` shall be in the range of 0 to $\text{Min}(6, \text{CtbLog2SizeY}) - \text{MinCbLog2SizeY}$, inclusive. The base 2 logarithm of the minimum size in luma samples of a luma leaf block resulting from quadtree splitting of a CTU is derived as follows:

$$\text{[00149]} \quad \text{MinQtLog2SizeInterY} = \text{sps_log2_diff_min_qt_min_cb_inter_slice} + \text{MinCbLog2SizeY} \quad (52)$$

[00150] `sps_max_mtt_hierarchy_depth_inter_slice` specifies the default maximum hierarchy depth for coding units resulting from multi-type tree splitting of a quadtree leaf in slices with `sh_slice_type` equal to 0 (B) or 1 (P) referring to the SPS. When `sps_partition_constraints_override_enabled_flag` is equal to 1, the default maximum hierarchy depth can be overridden by `ph_max_mtt_hierarchy_depth_inter_slice` present in PH syntax structures referring to the SPS. The value of `sps_max_mtt_hierarchy_depth_inter_slice` shall be in the range of 0 to $2 * (\text{CtbLog2SizeY} - \text{MinCbLog2SizeY})$, inclusive.

[00151] `sps_log2_diff_max_bt_min_qt_inter_slice` specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a binary split and the base 2 logarithm of the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in slices with `sh_slice_type` equal to 0 (B) or 1 (P) referring to the SPS. When `sps_partition_constraints_override_enabled_flag` is equal to 1, the default difference can be overridden by `ph_log2_diff_max_bt_min_qt_luma` present in PH syntax structures referring to the SPS. The value of `sps_log2_diff_max_bt_min_qt_inter_slice` shall be in the range of 0 to $\text{CtbLog2SizeY} - \text{MinQtLog2SizeInterY}$, inclusive. When `sps_log2_diff_max_bt_min_qt_inter_slice` is not present, the value of `sps_log2_diff_max_bt_min_qt_inter_slice` is inferred to be equal to 0.

[00152] `sps_log2_diff_max_tt_min_qt_inter_slice` specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a ternary split and the base 2 logarithm of the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in slices with `sh_slice_type` equal to 0 (B) or 1 (P) referring to the SPS. When `sps_partition_constraints_override_enabled_flag` is equal to 1, the default difference can be overridden by `ph_log2_diff_max_tt_min_qt_luma` present in PH syntax structures referring to the SPS. The value of `sps_log2_diff_max_tt_min_qt_inter_slice` shall be in the range of 0 to $\text{Min}(6,$

$\text{CtbLog2SizeY}) - \text{MinQtLog2SizeInterY}$, inclusive. When $\text{sps_log2_diff_max_tt_min_qt_inter_slice}$ is not present, the value of $\text{sps_log2_diff_max_tt_min_qt_inter_slice}$ is inferred to be equal to 0.

[00153] $\text{sps_max_luma_transform_size_64_flag}$ equal to 1 specifies that the maximum transform size in luma samples is equal to 64. $\text{sps_max_luma_transform_size_64_flag}$ equal to 0 specifies that the maximum transform size in luma samples is equal to 32. When not present, the value of $\text{sps_max_luma_transform_size_64_flag}$ is inferred to be equal to 0.

[00154] The variables MinTbLog2SizeY , MaxTbLog2SizeY , MinTbSizeY , and MaxTbSizeY are derived as follows:

$$\text{[00155]} \quad \text{MinTbLog2SizeY} = 2 \quad (53)$$

$$\text{[00156]} \quad \text{MaxTbLog2SizeY} = \text{sps_max_luma_transform_size_64_flag} ? 6 : 5 \quad (54)$$

$$\text{[00157]} \quad \text{MinTbSizeY} = 1 \ll \text{MinTbLog2SizeY} \quad (55)$$

$$\text{[00158]} \quad \text{MaxTbSizeY} = 1 \ll \text{MaxTbLog2SizeY} \quad (56)$$

[00159] An extended quad tree is now discussed. FIG. 7 is a schematic diagram 700 of example EQT partitioning structures. An EQT partitioning structure corresponding to a block partitioning process includes an extended quad tree partitioning process for the block of video data. The extended quad partitioning structure represents partitioning the block of video data into final sub-blocks. When the extended quad tree partitioning process decides to apply an extended quad tree partition to a block, the block is always split into four sub-blocks. Decoding of the final sub-blocks is based on the video bitstream. Decoding of the block of video data is based on the final sub-blocks decoded according to the EQT structure derived.

[00160] The EQT partitioning process can be applied to a block recursively to generate EQT leaf nodes. Alternatively, when EQT is applied to a certain block, for each of the sub-blocks resulting from the EQT split, may further be split into BT and/or QT and/or TT and/or EQT and/or other kinds of partition trees. In one example, EQT and QT may share the same depth increment process and the same restrictions of leaf node sizes. In this case, the partitioning of one node can be implicitly terminated when the size of the node reaches a minimum allowed quad tree leaf node size or EQT depth with the node reaches a maximum allowed quad tree depth. Alternatively, EQT and QT may share different depth increment processes and/or restrictions of leaf node sizes. The partitioning of one node by EQT may be implicitly terminated when the size of the node reaches a minimum allowed EQT leaf node size or the EQT depth associated with the node reaches a maximum allowed

EQT depth. In one example, the EQT depth and/or the minimum allowed EQT leaf node sizes may be signaled in a sequences parameter set (SPS), a picture parameter set (PPS), a slice header, a CTU, a region, a tile, and/or a CU.

[00161] EQT may not use a quad tree partition applied to a square block, for example where the block has a size of $M \times N$ where M and N are equal or unequal non-zero positive integer values. Instead, EQT splits one block equally into four partitions, such as an $M/4 \times N$ split 701 or an $M \times N/4$ split 703. Split 727 and split 729 show general examples of split 701 and 703, respectively. For example, split 727 is split into $M \times N_1$, $M \times N_2$, $M \times N_3$, and $M \times N_4$, where $N_1 + N_2 + N_3 + N_4 = N$. Further, split 729 is split into $M_1 \times N$, $M_2 \times N$, $M_3 \times N$ and $M_4 \times N$ where $M_1 + M_2 + M_3 + M_4 = M$.

[00162] In another example, the EQT can split the shape equally into four partitions where the partition size is dependent on the maximum and minimum values of M and N . In one example, one 4×32 block may be split into four 4×8 sub-blocks while a 32×4 block may be split into four 8×4 sub-blocks.

[00163] In another example, EQT splits one block equally into four partitions, such as two partitions are with size equal to $(M \cdot w_0/w) \times (N \cdot h_0/h)$ and the other two are with $(M \cdot (w-w_0)/w) \times (N \cdot (h-h_0)/h)$ as shown by split 705, split 707, split 709, and split 711. For example, w_0 and w may be equal to 1 and 2, respectively, such that the width is reduced by half while the height can use other ratios instead of 2:1 to get the sub-blocks. In another example, h_0 and h may be equal to 1 and 2, respectively, such that the height is reduced by half while the width can use other ratios instead of 2:1. For example, split 705 includes a sub-block width fixed to be $M/2$ with a height equal to $N/4$ or $3N/4$ with a smaller selection for the top two partitions. For example, split 707 includes a sub-block height fixed to be $N/2$ with a width equal to $M/4$ or $3M/4$ with a smaller selection for the left two partitions. For example, split 709 includes a sub-block width fixed to be $M/2$ with a height equal to $N/4$ or $3N/4$ with a smaller selection for the bottom two partitions. For example, split 711 includes a sub-block height fixed to be $N/2$ with a width equal to $M/4$ or $3M/4$ with a smaller selection for the right two partitions.

[00164] Split 713, split 715, split 717, split 719, split 721, and split 723 show other examples of quad tree partitioning. For example, split 713, split 715, and split 717 show options where the shape is split by $M \times N/4$ and $M/2 \times N/2$. For example, split 719, split 721, and split 723 show options where the shape is split by $N \times M/4$ and $N/2 \times M/2$.

[00165] Split 725 shows a more general case of quad tree partitioning with different shapes of partitions. In this case, split 725 is split such that $M1 \times N1$, $(M-M1) \times N1$, $M1 \times (N-N1)$ and $(M-M1) \times (N-N1)$.

[00166] FIG. 8 is a schematic diagram 800 of example flexible tree (FT) partitioning structures. A FT partitioning structure corresponds to a block partitioning process including an FT partitioning process for the block of video data. The FT partitioning structure represents a partitioning for a block of video data into final sub-blocks. When the FT partitioning process decides to apply a FT partition to a block, the block is split into K sub-blocks wherein K could be larger than 4. The final sub-blocks can be coded based on the video bitstream. Further, the block of video data can be decoded based on the final sub-blocks decoded according to the FT structure derived. The FT partitioning process can be applied to a given block recursively to generate FT tree leaf nodes. The partitioning of one node is implicitly terminated when the node reaches a minimum allowed FT leaf node size or FT depth associated with the node reaches a maximum allowed FT depth. Further, when FT is applied to a certain block, multiple sub-blocks can be created. Each of the sub-blocks created by FT may further be split into BT, QT, EQT, TT, and/or other kinds of partition trees. In an example, the FT depth or the minimum allowed FT leaf node sizes or the minimum allowed partition size for FT may be signaled in a SPS, a PPS, a slice header, a CTU, a region, a tile, and/or a CU. Similar to EQT, all of the sub-blocks created by FT partitions may be the same or different sizes.

[00167] Schematic diagram 800 includes example FT partitioning structures where the number of sub-blocks, denoted as K , is set equal to six or eight. Split 801 is a partitioning structure with $K=8$, $M/4 * N/2$. Split 803 is a partitioning structure with $K=8$, $M/2 * N/4$. Split 805 is a partitioning structure with $K=6$, $M/2 * N/2$ and $M/4 * N/2$. Split 807 is a partitioning structure with $K=6$, $M/2 * N/2$ and $M/2 * N/4$.

[00168] FIG. 9 is a schematic diagram 900 of example generalized TT (GTT) partitioning structures. For the TT partitioning structure, the restriction of splitting along either horizontal or vertical may be removed. The GTT partition pattern may be defined as splitting for both horizontal and vertical. Split 901 employs a left split from a vertical TT split and a horizontal BT split of the remaining area. Split 903 employs a bottom split from a horizontal TT split and a vertical BT split of the remaining area. In some examples, the partitioning EQT, FT, and/or GTT partitioning methods may be applied under certain conditions. In other words, when the condition(s) are not satisfied, there is no need to signal the partition types. In another example, the EQT, FT, and/or GTT

partitioning methods may be used to replace other partition tree types. In another example, the EQT, FT, and/or GTT partitioning methods may be only used as a replacement for other partition tree types under certain conditions. In one example, the condition may be based on the picture, slice types, block sizes, the coded modes; and/or whether a block is located at a picture, slice, and/ or tile boundary. In one example, EQT may be treated in the same way as QT. In this case, when the QT partition tree type is selected, more flags/indications of the detailed quad-tree partition patterns may be further signaled. In some examples, EQT may be treated as additional partition patterns. In one example, the signaling of partitioning methods of EQT, FT, and/or GTT may be conditional. For example, one or more EQP, FT, and/or GTT partitioning methods may not be used in some cases, and the bits corresponding to signal these partitioning methods are not signaled.

[00169] FIG. 10 is a schematic diagram of example boundary partitioning tree 1000, which is also known as a versatile boundary partition. The boundary partitioning tree 1000 is an example boundary handling method for VVC and/or Audio and Video Coding Standard Workgroup Part three (AVS-3.0). Since the forced quadtree boundary partition solution in VVC is not optimized, the boundary partitioning tree 1000 uses regular block partition syntax to maintain continuity with the CABAC engine as well as to match the picture boundary. The versatile boundary partition obtains the following rules (both encoder and decoder). Since the boundary partitioning tree 1000 uses exactly the same partition syntax of the normal block (non-boundary) for boundaries located at block, the syntax is not changed. If the no split mode is parsed for the boundary CU, the forced boundary partition (FBP) is used to match the picture boundary. After the forced boundary partition is used (non-singling boundary partition), no further partition is performed. The forced boundary partition is described as follows. If the size of block is larger than the maximal allowed BT size, forced QT is used to perform the FBP in the current forced partition level. Otherwise, if the bottom-right sample of current CU is located below the bottom picture boundary and not extended to the right boundary, a forced horizontal BT is used to perform the FBP in the current forced partition level. Otherwise, if the bottom-right sample of current CU is located at the right side of the right picture boundary and not below the bottom boundary, a forced vertical BT is used to perform the FBP in the current forced partition level. Otherwise, if the bottom-right sample of current CU is located at the right side of the right picture boundary and below the bottom boundary, a forced QT is used to perform the FBP in the current forced partition level.

[00170] FIG. 11 is a schematic diagram 1100 of example partitioning structures used in Audio and Video Coding Standard (AVS) part three (AVS-3.0). Partitioning in AVS-3.0 is now discussed. The Audio and Video Coding Standard (AVS) Workgroup of China was authorized to be established by the Science and Technology Department under the former Ministry of Industry and Information Technology of People's Republic of China. With the mandate of satisfying the demands from the rapidly growing information industry, AVS is committed to producing technical standards of high quality for compression, decompression, processing, and representation of digital audio and video, and thus providing digital audio-video equipment and systems with high-efficient and economical coding/decoding technologies. AVS can be applied in wide variety of significant information sectors including high-resolution digital broadcast, high-density laser-digital storage media, wireless broadband multimedia communication and internet broad-band stream media. AVS is one of the second generation of source coding/decoding standards and owns independent Chinese intellectual property rights. Source coding technology primarily addresses the problem of coding and compressing audio and video mass data from initial data and original sources. Hence AVS is known as digital video and audio coding technology, and is the premise of the subsequent digital transmission, storage, and broadcast. Further, AVS serves as a common standard for the digital video and audio industry.

[00171] AVS-3.0 employs a QT partitioning 1101, a vertical BT partitioning 1105, a horizontal BT partitioning 1103, and a horizontal extended quad-tree (EQT) partitioning 1107, and a vertical EQT partitioning 1109 to split a largest coding unit (LCU) into multiple CUs. QT partitioning, BT partitioning, and EQT partitioning can all be used for the root, internal nodes, or leaf nodes of the partitioning tree. However, QT partitioning is forbidden after any BT and/or EQT partitioning.

[00172] FIG. 12 is a schematic diagram 1200 of example Unsymmetrical Quad-Tree (UQT) partitioning structures. UQT partitioning employs a block with dimensions $W \times H$, which is split into four partitions with dimensions $W_1 \times H_1$, $W_2 \times H_2$, $W_3 \times H_3$ and $W_4 \times H_4$, where W_1 , W_2 , W_3 , W_4 , H_1 , H_2 , H_3 , H_4 are all integers. In one example, and at least one of the partitions has different block size compared to others. In one example, only two of the four partitions may have equal size, and the other two are different with each other and different from the two partitions with equal size. In one example, all the parameters are in the form of power of 2. For example, $W_1 = 2N_1$, $W_2 = 2N_2$, $W_3 = 2N_3$, $W_4 = 2N_4$, $H_1 = 2M_1$, $H_2 = 2M_2$, $H_3 = 2M_3$, $H_4 = 2M_4$. In one example, UQT only splits one partition in vertical direction, for example, $H_1 = H_2 = H_3 = H_4 = H$. In one example, in split 1201 $W_1 = W/8$, $W_2 = W/2$, $W_3 = W/8$, $W_4 = W/4$, $H_1 = H_2 = H_3 = H_4 = H$. This kind of UQT is vertical

split and named as UQT1-V. In one example, in split 1203 $W1=W/8$, $W2=W/2$, $W3=W/4$, $W4=W/8$, $H1=H2=H3=H4=H$. This kind of UQT is vertical split and named as UQT2-V. In one example in split 1205 $W1=W/4$, $W2=W/8$, $W3=W/2$, $W4=W/8$, $H1=H2=H3=H4=H$. This kind of UQT is vertical split and named as UQT3-V. In one example, in split 1207 $W1=W/8$, $W2=W/4$, $W3=W/2$, $W4=W/8$, $H1=H2=H3=H4=H$. This kind of UQT is vertical split and named as UQT4-V.

[00173] In one example, UQT only splits one partition in horizontal direction, for example, $W1=W2=W3=W4=W$. In one example, in split 1209 $H1=H/8$, $H2=H/2$, $H3=H/8$, $H4=H/4$, $W1=W2=W3=W4=W$. This kind of UQT is horizontal split and named as UQT1-H. In one example, in split 1211 $H1=H/8$, $H2=H/2$, $H3=H/4$, $H4=H/8$, $W1=W2=W3=W4=W$. This kind of UQT is horizontal split and named as UQT2-H. In one example, in split 1213 $H1=H/4$, $H2=H/8$, $H3=H/2$, $H4=H/8$, $W1=W2=W3=W4=W$. This kind of UQT is horizontal split and named as UQT3-H. In one example, in split 1215 $H1=H/8$, $H2=H/4$, $H3=H/2$, $H4=H/8$, $W1=W2=W3=W4=W$. This kind of UQT is horizontal split and named as UQT4-H.

[00174] FIG. 13 is a schematic diagram 1300 of example ETT partitioning structures, including an ETT-V split 1301 and an ETT-H split 1303. When employing ETT, a block with dimensions width times height ($W \times H$) is split into three partitions with dimensions $W1 \times H1$, $W2 \times H2$, and $W3 \times H3$. $W1$, $W2$, $W3$, $H1$, $H2$, $H3$ are all integers. In an example, and at least one of the parameters is not in the form of power of 2. $W1$, $W2$, and $W3$ are widths of resulting sub-blocks. $H1$, $H2$, and $H3$ are heights of resulting sub-blocks. In one example, $W2$ cannot be in a form of $W2 = 2^{N2}$ with any positive integer $N2$. In another example, $H2$ cannot be in a form of $H2 = 2^{N2}$ with any positive integer $N2$. In one example, at least one of the parameters is in the form of power of 2. In one example, $W1$ is in a form of $W1 = 2^{N1}$ with a positive integer $N1$. In another example, $H1$ is in a form of $H1 = 2^{N1}$ with a positive integer $N1$.

[00175] In one example, ETT only splits one partition in a vertical direction, for example where $W1=a1*W$, $W2=a2*W$, and $W3=a3*W$, where $a1+a2+a3=1$, and where $H1=H2=H3=H$. This kind of ETT is vertical split and may be referred to as ETT-V. In one example, ETT-V split 1301 can be used where $W1=W/8$, $W2=3*W/4$, $W3=W/8$, and $H1=H2=H3=H$. In one example, ETT only splits one partition in horizontal direction, for example where $H1=a1*H$, $H2=a2*H$, and $H3=a3*H$, where $a1+a2+a3=1$, and where $W1=W2=W3=W$. This kind of ETT is a horizontal split and

may be referred to as ETT-H. In one example, ETT-H split 1303 can be used where $H1=H/8$, $H2=3*H/4$, $H3=H/8$, and $W1=W2=W3=W$.

[00176] FIG. 14 is a schematic diagram 1400 of example 1/4 UBT partitioning structures, which includes vertical UBT (UBT-V) partitions and horizontal UBT (UBT-H) partitions. A block of dimensions $W \times H$ can be split into two sub-blocks dimensions $W1 \times H1$ and $W2 \times H2$, where one of the sub-blocks is a dyadic block and the other is a non-dyadic block. Such a split is named as Unsymmetric Binary Tree (UBT) split. In one example, $W1 = a \times W$, $W2 = (1-a) \times W$, and $H1=H2=H$. In such a case, the partition may be called a vertical UBT (UBT-V). In one example, a may be smaller than $1/2$, such as $1/4$, $1/8$, $1/16$, $1/32$, $1/64$, etc. In such a case, the partition may be called a Type 0 UBT-V, an example of which is shown as split 1401. In one example, a may be larger than $1/2$, such as $3/4$, $7/8$, $15/16$, $31/32$, $63/64$, etc. In such a case, the partition is called a Type 1 UBT-V, an example of which is shown as split 1403. In one example, $H1 = a \times H$, $H2 = (1-a) \times H$, $W1=W2=W$. In such a case, the partition may be called a horizontal UBT (UBT-H). In one example, a may be smaller than $1/2$, such as $1/4$, $1/8$, $1/16$, $1/32$, $1/64$, etc. In such a case, the partition is called a Type 0 UBT-H, an example of which is shown as split 1405. In one example, a may be larger than $1/2$, such as $3/4$, $7/8$, $15/16$, $31/32$, $63/64$, etc. In such a case, the partition may be called a Type 1 UBT-H, an example of which is shown as split 1407.

[00177] Disclosed herein are systems and methods for introducing constraints on partitioning methods. As noted above, with the new proposed partitioning methods, there are too many combinations to be tested. Thus, it may be useful to introduce constraints on possible partitioning methods. As used herein the term “conversion” may refer to or include encoding and/or decoding a video media file or bit stream. In order to reduce the number of possible partitioning method methods, the present document provides for determining a whether a partitioning scheme is enabled for conversion of the video media file or bit stream according to a rule. The rule, in some embodiments, is based on a color characteristic of the video media file or bit stream. The conversion is performed based on the determined rule. The color characteristic may include a color component, color format, whether the video media file or bit stream is coded using separate plane coding and so on. The rule may specify, for example, a type unsymmetrical tree partitioning disallowed due to the video block being a luma block or a chroma block.

[00178] In some embodiments, the present document may determine for a conversion between a video block of a video with dimensions $W \times H$ and a bitstream of the video, whether a partitioning

scheme is enabled for coding of the video block according to a rule. The rule based on the dimensions of the video block and/or another partitioning scheme that is a disallowed partitioning scheme for the video block. The partitioning scheme may be a horizontal or a vertical unsymmetrical binary tree (UBT) or a horizontal or a vertical unsymmetrical quad tree (UQT) or a horizontal or a vertical extended ternary tree (ETT) partition. The disallowed partitioning scheme may include binary tree (BT), ternary tree (TT), UBT, UQT, ETT, or quadtree (QT). In some embodiments, UBT, UQT or ETT may not be allowed based on W or H being larger than a maximum allowed size for a split using the disallowed partition scheme. Further, the rule may specify that the partitioning scheme is disabled in response to W or H satisfying one or more conditions. The partitioning scheme may also include unsymmetric binary tree UBT, unsymmetric quadtree UQT or extended ternary tree ETT, multi-type tree MTT, binary tree BT, or ternary tree TT. In some embodiments, various conditions may be applied that depend on a color format or a color component of the video block or decoded information of the video block. In other embodiments, the rule may specify that where W is greater than T1 and H is smaller than or equal to T2, the partitioning scheme is disabled for the video block, or where H is greater than T1 and W is smaller than or equal to T2, the partitioning scheme is disabled for the video block.

[00179] In still other embodiments, the rule may be based on a multi-type tree depth (MTT-depth) or a binary tree depth (BT-depth) of the video block or stream. Similarly, the partitioning scheme may include unsymmetric binary tree UBT, unsymmetric quadtree UQT or extended ternary tree ETT. In some cases, the MTT-depth of the partitioning scheme is derived from a maximum allowed depth of another partitioning scheme comprising quadtree QT, multi-tree type MTT, binary tree BT or ternary tree TT. Further the bitstream may indicate the MTT depth. In some embodiments, the rule specifies that the partitioning scheme is disallowed responsive to the current MTT-depth being larger than a maximum allowed depth of another partitioning scheme. Further, the rule may specify that the partitioning scheme is disallowed responsive to the current MTT-depth being smaller than a minimum allowed depth of another partitioning scheme.

[00180] In other embodiments, the rule is based on whether a local-dual tree is used for the video block or stream. In this case, the partitioning scheme includes unsymmetric binary tree UBT, unsymmetric quadtree UQT or extended ternary tree ETT. As such, the rule may specify that the partitioning scheme is disallowed in case that the local-dual tree is used for the current video block.

Further, UBT UQT or ETT may not be allowed responsive to W or H being smaller than a minimum allowed size for a split using the disallowed partition scheme.

[00181] The detailed embodiments below should be considered as examples to explain general concepts. These embodiments should not be interpreted in a narrow way. Furthermore, these embodiments can be combined in any manner. In the following discussion, QT, BT, TT, EQT, UQT, and ETT may refer to QT split, BT split, TT split, EQT split, UQT split and ETT split, respectively. In the following discussion, a block is a dyadic block if both width and height is a dyadic number, which is in a form of a 2^N with N being a positive integer. In the following discussion, a block is a non-dyadic block if at least one of width and height is a non-dyadic number, which cannot be represented in a form of a 2^N with N being a positive integer. In the following discussion, split and partitioning have the same meaning.

[00182] Example 1

[00183] The determination of whether one kind of split is allowed may depend on the color component of the block to be coded/decoded, and/or color format such as 4:2:0 or 4:4:4 and/or whether separate plane coding is enabled.

[00184] For example, at least one kind of UBT is not allowed for luma or chroma components. In another example, at least one kind of UQT is not allowed for luma or chroma components. In another example, ETT is not allowed for luma or chroma components. In another example, TT is not allowed for luma or chroma components. In another example, BT is not allowed for luma or chroma components. In another example, QT is not allowed for luma or chroma components. In another example, whether one kind of split is allowed on one color component may be signaled from the encoder to the decoder, such as SPS/PPS/picture header/slice header. In another example,, whether one kind of split is allowed may depend on whether dual tree split or local dual tree split is applied.

[00185] Example 2

[00186] In one example, at least one of horizontal/vertical UBT and/or horizontal/vertical UQT and/or horizontal/vertical ETT is not allowed if XT is not allowed for a block with dimensions W×H.

[00187] XT may be BT, TT, UBT, UQT, ETT, or QT.

[00188] In one example, UBT and/or UQT and/or ETT is not allowed if W is larger than the maximum allowed size for XT split and/or H is larger than the maximum allowed size for XT split. For example, the maximum allowed size for XT split is an integer such as 64.

[00189] In one example, the maximum allowed size for XT split may be signaled from the encoder to the decoder.

[00190] In one example, the maximum allowed size for YT (e.g., YT=UBT, UQT, ETT) split may be dependent on slice/picture types.

[00191] In one example, for I slice and inter slices, the maximum allowed size for YT may be different.

[00192] In one example, the maximum allowed size for YT (e.g., YT=UBT, UQT, ETT) split is restricted to be no greater than the maximum allowed size for XT (e.g., XT=MTT/BT/TT).

[00193] In one example, the maximum allowed size for YT (e.g., YT=UBT, UQT, ETT) split may be signaled from the encoder to the decoder.

[00194] In one example, different maximum allowed size may be used or signaled for different partition trees.

[00195] In one example, indications of different maximum allowed size may be used or signaled for different partition trees.

[00196] In one example, the indication of the maximum allowed size for YT (e.g., YT=UBT, UQT, ETT) may be conditionally signalled (e.g., based on the value of maximum allowed size for XT (e.g., XT=QT/MTT/BT/TT)).

[00197] In one example, the indication of the maximum allowed size for YT may be signalled only when the MTT-depth is not equal to 0.

[00198] In one example, the maximum allowed size for YT (e.g., YT=UBT, UQT, ETT) may be predictively coded.

[00199] In one example, the difference between the maximum allowed size for YT and XT (e.g., XT=MTT/BT/TT) may be signalled.

[00200] In one example, a shared maximum allowed size may be used or indication of the shared maximum allowed size may be signaled for different XTs.

[00201] For example, a shared maximum allowed size may be used or indication of the shared maximum allowed size may be signaled for BT and UBT.

[00202] For example, a shared maximum allowed size may be used or indication of the shared maximum allowed size may be signaled for ETT and UBT.

[00203] For example, a shared maximum allowed size may be used or indication of the shared maximum allowed size may be signaled for UQT and UBT.

[00204] For example, a shared maximum allowed size may be used or indication of the shared maximum allowed size may be signaled for TT and UBT.

[00205] In one example, the maximum allowed sizes for XT split may be different for different color components.

[00206] Alternatively, the maximum allowed size for XT split is the same for different color components.

[00207] In one example, UBT and/or UQT and/or ETT is not allowed if W is smaller than the minimum allowed size for XT split and/or H is smaller than the minimum allowed size for XT split. For example, the minimum allowed size for XT split is an integer such as 4.

[00208] In one example, the minimum allowed size for XT split may be signaled from the encoder to the decoder.

[00209] In one example, the minimum allowed size for YT (e.g., YT=UBT, UQT, ETT) split may be dependent on slice/picture types.

[00210] In one example, for I slice and inter slices, the minimum allowed size for YT may be different.

[00211] In one example, the minimum allowed size for YT (e.g., YT=UBT, UQT, ETT) split is restricted to be no smaller than the minimum allowed size for XT (e.g., XT=MTT/BT/TT).

[00212] In one example, the minimum allowed size for YT (e.g., YT=UBT, UQT, ETT) split may be signaled from the encoder to the decoder.

[00213] E.g. different minimum allowed size may be used or signaled for different partitions.

[00214] In one example, indications of different minimum allowed size may be used or signaled for different partition trees.

[00215] In one example, the indication of the minimum allowed size for YT (e.g., YT=UBT, UQT, ETT) may be conditionally signalled (e.g., based on the value of minimum allowed size for XT (e.g., XT=QT/MTT/BT/TT)).

[00216] In one example, the indication of the minimum allowed size for YT may be signalled only when the MTT-depth is not equal to 0.

[00217] In one example, the minimum allowed size for YT (e.g., YT=UBT, UQT, ETT) may be predictively coded.

[00218] In one example, the difference between the minimum allowed size for YT and XT (e.g., XT=MTT/BT/TT) may be signalled.

[00219] In one example, a shared minimum allowed size may be used or indication of the shared minimum allowed size may be signaled for different XTs.

[00220] For example, a shared minimum allowed size may be used or indication of the shared minimum allowed size may be signaled for BT and UBT.

[00221] For example, a shared minimum allowed size may be used or indication of the shared minimum allowed size may be signaled for ETT and UBT.

[00222] For example, a shared minimum allowed size may be used or indication of the shared minimum allowed size may be signaled for UQT and UBT.

[00223] For example, a shared minimum allowed size may be used or indication of the shared minimum allowed size may be signaled for TT and UBT.

[00224] In one example, the minimum allowed sizes for XT split may be different for different color components.

[00225] Alternatively, the minimum allowed size for XT split is the same for different color components.

[00226] In all above example, XT could be one specific partition type of a partitioning tree, e.g., horizontal UBT/UQT/ETT or vertical UBT/UQT/ETT.

[00227] **Example 3**

[00228] In one example, XT (e.g., XT= UBT/UQT/ETT/MTT/BT/TT) is not allowed for a block with dimensions $W \times H$, if W and/or H satisfies one or more conditions.

[00229] In one example, in the following discussion, thresholds $T1 = t1 \gg S1$ and $T2 = t2 \gg S2$, may be used to determine whether XT is allowed. In some embodiment, $S1$ and $S2$ may be based on a color characteristic of the video media data or video unit and $t1$ and $t2$ may be based on a split of the video unit.

[00230] The relationship of W and $T1$, and/or the relationship of H and $T2$ may be used to determine whether XT is allowed.

[00231] In one example, the conditions may depend on color components and/or color format and/or other decoded information (e.g., slice/picture types).

[00232] In one example, $S1$ and/or $S2$ may depend on color component and/or color format.

[00233] For example, $S1=S2=1$ for the luma component.

[00234] For example, $S1=S2=2$ for the chroma component with color format 4:2:0.

[00235] For example, $S1=2$ for the chroma component with color format 4:2:2.

- [00236] For example, $S_2 = 1$ for the chroma component with color format 4:2:2.
- [00237] For example, $S_1 = S_2 = 1$ for the chroma component with color format 4:4:4.
- [00238] In one example, if W is greater than T_1 and H is smaller than or equal to T_2 , XT is not allowed.
- [00239] For example, $t_1 = 32$, $t_2 = 32$, XT is horizontal ETT.
- [00240] For example, $t_1 = 32$, $t_2 = 32$, XT is horizontal UQT.
- [00241] For example, $t_1 = 32$, $t_2 = 16$, XT is horizontal UBT.
- [00242] In one example, if H is greater than T_1 and W is smaller than or equal to T_2 , XT is not allowed.
- [00243] For example, $t_1 = 32$, $t_2 = 32$, XT is vertical ETT.
- [00244] For example, $t_1 = 32$, $t_2 = 32$, XT is vertical UQT.
- [00245] For example, $t_1 = 16$, $t_2 = 32$, XT is vertical UBT.
- [00246] In one example, if W is greater than or equal to T_1 and H is smaller than or equal to T_2 , XT is not allowed.
- [00247] For example, $t_1 = 64$, $t_2 = 16$, XT is horizontal TT.
- [00248] For example, $t_1 = 64$, $t_2 = 8$, XT is horizontal BT.
- [00249] In one example, if H is greater than or equal to T_1 and W is smaller than or equal to T_2 , XT is not allowed.
- [00250] For example, $t_1 = 16$, $t_2 = 64$, XT is vertical TT.
- [00251] For example, $t_1 = 8$, $t_2 = 64$, XT is vertical BT.
- [00252] In one example, indications of T_1 and/or T_2 and/or t_1 and/or t_2 may be pre-defined or signaled from the encoder to the decoder, such as in SPS/PPS/picture header/slice header.
- [00253] In one example, T_1 and/or T_2 and/or t_1 and/or t_2 may depend on picture/slice types/color component.
- [00254] For example, different T_1 and/or T_2 and/or t_1 and/or t_2 may be allowed or signaled for different split methods.
- [00255] For example, shared T_1 and/or T_2 and/or t_1 and/or t_2 may be allowed or signaled for different split methods.
- [00256] In one example, indications of T_1 and/or T_2 and/or t_1 and/or t_2 may be signalled for different picture/slice types.
- [00257] **Example 4**

[00258] In one example, XT may be disallowed if the current multi-type tree depth (MTT-depth) or binary-tree depth (BT-depth) or any other depth is larger than a maximum allowed depth, wherein XT may be UBT, UQT or ETT.

[00259] In one example, the maximum allowed depth for XT may be derived from the maximum allowed depth for YT (e.g., $YT = QT/MTT/BT/TT$).

[00260] In one example, the maximum allowed depth for XT may be set equal to the maximum allowed depth for YT (e.g., $YT = QT/MTT/BT/TT$).

[00261] The maximum allowed depth for XT may depend on color components.

[00262] In one example, indication of the maximum allowed depth for XT may be signaled from the encoder to the decoder, such as in SPS/PPS/picture header/slice header.

[00263] In one example, indication of the maximum allowed depth may be dependent on slice type, or picture type, or temporal layer, or color component/color format.

[00264] In one example, indication of the maximum allowed depth may be conditionally signalled.

[00265] In one example, indication of the maximum allowed depth may be predictively coded.

[00266] The indication or signaling of maximum allowed depth for XT may depend on color components.

[00267] In one example, the depth of a sub-block may be added by k (e.g., k is a non-zero integer) after it is split from a parent block by XT, wherein XT may be UBT, UQT or ETT.

[00268] In one example, k may be 1.

[00269] In one example, k may be different for different sub-block.

[00270] For example, k1 and k2 are added for two sub-blocks S1 and S2, respectively. k1 may be larger than k2 if S1 is smaller than S2.

[00271] For example, for UBT split, k = 2 for the smaller sub-block and k = 1 for the larger sub-block.

[00272] For example, for ETT split, k = 1 for the center-sub-block and k = 2 for the two side-sub-blocks.

[00273] **Example 5**

[00274] In one example, XT may be disallowed if the current multi-type tree depth (MTT-depth) or binary-tree depth (BT-depth) or any other depth is smaller than a minimum allowed depth, wherein XT may be UBT, UQT or ETT.

[00275] In one example, the minimum allowed depth for XT may be derived from the minimum allowed depth for YT (e.g., $YT = QT/MTT/BT/TT$).

[00276] In one example, the minimum allowed depth for XT may be set equal to the minimum allowed depth for YT (e.g., $YT = QT/MTT/BT/TT$).

[00277] The minimum allowed depth for XT may depend on color components.

[00278] In one example, indication of the minimum allowed depth for XT may be signaled from the encoder to the decoder, such as in SPS/PPS/picture header/slice header.

[00279] In one example, indication of the minimum allowed depth may be dependent on slice type, or picture type, or temporal layer, or color component/color format.

[00280] In one example, indication of the minimum allowed depth may be conditionally signalled.

[00281] In one example, indication of the minimum allowed depth may be predictively coded.

[00282] The indication or signaling of minimum allowed depth for XT may depend on color components.

[00283] **Example 6**

[00284] In one example, XT may be disallowed when local-dual tree is used, wherein XT may be UBT, UQT or ETT.

[00285] For example, XT is not allowed for chroma component when local dual tree is applied.

[00286] **Example 7**

[00287] In the disclosure, “signaling” may be applied in any unit such as SPS/sequence header/PPS/picture header/slice header/CTU line/CTU.

[00288] FIG. 15 is a block diagram showing an example video processing system 1700 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 1700. The system 1700 may include input 1702 for receiving video content. The video content may be received in a raw or uncompressed format, e.g., 8 or 10 bit multi-component pixel values, or may be in a compressed or encoded format. The input 1702 may represent a network interface, a peripheral bus interface, or a storage interface. Examples of a network interface include wired interfaces such as Ethernet, passive optical network (PON), etc. and wireless interfaces such as Wireless Fidelity (Wi-Fi) or cellular interfaces.

[00289] The system 1700 may include a coding component 1704 that may implement the various coding or encoding methods described in the present document. The coding component 1704 may

reduce the average bitrate of video from the input 1702 to the output of the coding component 1704 to produce a coded representation of the video. The coding techniques are therefore sometimes called video compression or video transcoding techniques. The output of the coding component 1704 may be either stored, or transmitted via a communication connection, as represented by the component 1706. The stored or communicated bitstream (or coded) representation of the video received at the input 1702 may be used by a component 1708 for generating pixel values or displayable video that is sent to a display interface 1710. The process of generating user-viewable video from the bitstream representation is sometimes called video decompression. Furthermore, while certain video processing operations are referred to as “coding” operations or tools, it will be appreciated that the coding tools or operations are used at an encoder and corresponding decoding tools or operations that reverse the results of the coding will be performed by a decoder.

[00290] Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or Displayport, and so on. Examples of storage interfaces include SATA (serial advanced technology attachment), peripheral component interconnect (PCI), integrated drive electronics (IDE) interface, and the like. The techniques described in the present document may be embodied in various electronic devices such as mobile phones, laptops, smartphones or other devices that are capable of performing digital data processing and/or video display.

[00291] FIG. 16 is a block diagram of an example video processing apparatus 1800. The apparatus 1800 may be used to implement one or more of the methods described herein. The apparatus 1800 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 1800 may include one or more processors 1802, one or more memories 1804 and video processing circuitry 1806. The processor(s) 1802 may be configured to implement one or more methods described in the present document. The memory (memories) 1804 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing circuitry 1806 may be used to implement, in hardware circuitry, some techniques described in the present document. In some embodiments, the video processing circuitry 1806 may be at least partly included in the processor 1802, e.g., a graphics co-processor.

[00292] FIG. 17 is a flowchart for an example method 1900 of processing video data. The method 1900 includes determining, for a conversion between a visual media data and a bitstream, whether a partitioning scheme is enabled for coding of the video data according to a rule at step 1902. At step

1904, the method 1900 includes performing the conversion based on the rule. For example, step 1902 may include selecting equations and/or algorithms for use. The rule may be based on a color characteristic of the visual media data, or based on a multi-type tree depth (MTT-depth) or a binary tree depth (BT-depth) of a video unit of the visual media data. Where a video unit of the visual media data has a dimension $W \times H$, where W is the width of the video unit in pixels and H is the height of the video unit in pixels, the rule may be based on the dimensions of the video block and/or another partitioning scheme that is a disallowed partitioning scheme for the video block. The rule may be based on whether a local-dual tree is used for a video unit of the visual media data. Step 1904 can then include performing video encoding at an encoder or video decoding at a decoder, for example by employing the results determined by the equations and/or algorithms.

[00293] In some embodiments, the color characteristic comprises at least one of a color components of the visual media data or whether the visual media data is coded using separate plane coding. Further the rule may specify a type of unsymmetrical tree partitioning disallowed due to the video block being a luma block or a chroma block. Further, a video unit of the visual media data has a dimension $W \times H$, where W is the width of the video unit in pixels and H is the height of the video unit in pixels, and wherein the rule is based on the dimensions of the video block and/or another partitioning scheme that is a disallowed partitioning scheme for the video block.

[00294] In some embodiments, the partitioning scheme is a horizontal or a vertical unsymmetrical binary tree (UBT) or a horizontal or a vertical unsymmetrical quad tree (UQT) or a horizontal or a vertical extended ternary tree (ETT) partition. In other embodiments the disallowed partitioning scheme includes binary tree (BT), ternary tree (TT), UBT, UQT, ETT, or quadtree (QT), or UBT, UQT or ETT is not allowed responsive to W or H being larger than a maximum allowed size for a split using the disallowed partition scheme; or the rule specifies that the partitioning scheme is disabled in response to W or H satisfying one or more conditions; or the partitioning scheme includes includes unsymmetric binary tree UBT, unsymmetric quadtree UQT or extended ternary tree ETT, multi-type tree MTT, binary tree BT, or ternary tree TT. Further, the one or more conditions are based on thresholds $T1 = t1 \gg S1$ and $T2 = t2 \gg S2$.

[00295] In some embodiments, the one or more conditions are dependent on a color format of the visual media data or a color component of a video unit of the visual media data or decoded information of the video unit. Also, the rule may specify that in case that W is greater than $T1$ and H is smaller than or equal to $T2$, the partitioning scheme is disabled for the video unit. Also, the rule

may specify that in case that H is greater than $T1$ and W is smaller than or equal to $T2$, the partitioning scheme is disabled for the video unit.

[00296] In other embodiments, the rule is based on a multi-type tree depth (MTT-depth) or a binary tree depth (BT-depth) of a video unit of the visual media data. In this case, the partitioning scheme may include unsymmetric binary tree UBT, unsymmetric quadtree UQT or extended ternary tree ETT. Further, the MTT depth of the partitioning scheme is derived from a maximum allowed depth of another partitioning scheme comprising quadtree QT, multi-tree type MTT, binary tree BT or ternary tree TT. Also, the bitstream may indicate the MTT depth. The rule may specify that the partitioning scheme is disallowed responsive to the current MTT depth being larger than a maximum allowed depth of another partitioning scheme; or that the partitioning scheme is disallowed responsive to the current MTT depth being smaller than a minimum allowed depth of another partitioning scheme.

[00297] In still other embodiments, the rule is based on whether a local-dual tree is used for a video unit of the visual media data. In such case, the partitioning scheme includes unsymmetric binary tree UBT, unsymmetric quadtree UQT or extended ternary tree ETT. Further, the rule may specify that the partitioning scheme is disallowed in case that the local-dual tree is used for the current video block or that UBT UQT or ETT is not allowed responsive to W or H being smaller than a minimum allowed size for a split using the disallowed partitioning scheme.

[00298] A non-transitory computer readable medium comprising a computer program product for use by a video coding device, the computer program product comprising computer executable instructions stored on the non-transitory computer readable medium such that when executed by a processor cause the video coding device to perform the embodiments disclosed herein. Further, an apparatus for processing video data comprising: a processor; and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to perform the embodiments disclosed herein.

[00299] It should be noted that the method 1900 can be implemented in an apparatus for processing video data comprising a processor and a non-transitory memory with instructions thereon, such as video encoder 2100, video decoder 2200, and/or encoder 2300 (FIGs. 19-21). In such a case, the instructions upon execution by the processor, cause the processor to perform the method 1900. Further, the method 1900 can be performed by a non-transitory computer readable medium comprising a computer program product for use by a video coding device. The computer program

product comprises computer executable instructions stored on the non-transitory computer readable medium such that when executed by a processor cause the video coding device to perform the method 1900.

[00300] FIG. 18 is a block diagram that illustrates an example video coding system 2000 that may utilize the techniques of this disclosure. As shown in FIG. 18, video coding system 2000 may include a source device 2010 and a destination device 2020. Source device 2010 generates encoded video data which may be referred to as a video encoding device. Destination device 2020 may decode the encoded video data generated by source device 2010 which may be referred to as a video decoding device.

[00301] Source device 2010 may include a video source 2012, a video encoder 2014, and an input/output (I/O) interface 2016. Video source 2012 may include a source such as a video capture device, an interface to receive video data from a video content provider, and/or a computer graphics system for generating video data, or a combination of such sources. The video data may comprise one or more pictures. Video encoder 2014 encodes the video data from video source 2012 to generate a bitstream. The bitstream may include a sequence of bits that form a coded representation of the video data. The bitstream may include coded pictures and associated data. The coded picture is a coded representation of a picture. The associated data may include sequence parameter sets, picture parameter sets, and other syntax structures. I/O interface 2016 may include a modulator/demodulator (modem) and/or a transmitter. The encoded video data may be transmitted directly to destination device 2020 via I/O interface 2016 through network 2030. The encoded video data may also be stored onto a storage medium/server 2040 for access by destination device 2020.

[00302] Destination device 2020 may include an I/O interface 2026, a video decoder 2024, and a display device 2022. I/O interface 2026 may include a receiver and/or a modem. I/O interface 2026 may acquire encoded video data from the source device 2010 or the storage medium/server 2040. Video decoder 2024 may decode the encoded video data. Display device 2022 may display the decoded video data to a user. Display device 2022 may be integrated with the destination device 2020, or may be external to destination device 2020, which can be configured to interface with an external display device.

[00303] Video encoder 2014 and video decoder 2024 may operate according to a video compression standard, such as the High Efficiency Video Coding (HEVC) standard, Versatile Video Coding (VVM) standard and other current and/or further standards.

[00304] FIG. 19 is a block diagram that illustrates an example video encoder 2100, which may be video encoder 2014 in the system 2000 illustrated in FIG. 18. Video encoder 2100 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 19, video encoder 2100 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of video encoder 2100. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[00305] The functional components of video encoder 2100 may include a partition unit 2101, a prediction unit 2102 which may include a mode selection unit 2103, a motion estimation unit 2104, a motion compensation unit 2105, an intra prediction unit 2106, a residual generation unit 2107, a transform unit 2108, a quantization unit 2109, an inverse quantization unit 2110, an inverse transform unit 2111, a reconstruction unit 2112, a buffer 2113, and an entropy encoding unit 2114.

[00306] In other examples, video encoder 2100 may include more, fewer, or different functional components. In an example, prediction unit 2102 may include an intra block copy (IBC) unit. The IBC unit may perform prediction in an IBC mode in which at least one reference picture is a picture where the current video block is located.

[00307] Furthermore, some components, such as motion estimation unit 2104 and motion compensation unit 2105 may be highly integrated, but are represented in the example of FIG. 19 separately for purposes of explanation.

[00308] Partition unit 2101 may partition a picture into one or more video blocks. Video encoder 2100 and video decoder 2200 (FIG. 20) may support various video block sizes.

[00309] Mode selection unit 2103 may select one of the coding modes, intra or inter, e.g., based on error results, and provide the resulting intra or inter coded block to a residual generation unit 2107 to generate residual block data and to a reconstruction unit 2112 to reconstruct the encoded block for use as a reference picture. In some examples, mode selection unit 2103 may select a combination of intra and inter prediction (CIIP) mode in which the prediction is based on an inter prediction signal and an intra prediction signal. Mode selection unit 2103 may also select a resolution for a motion vector (e.g., a sub-pixel or integer pixel precision) for the block in the case of inter prediction.

[00310] To perform inter prediction on a current video block, motion estimation unit 2104 may generate motion information for the current video block by comparing one or more reference frames from buffer 2113 to the current video block. Motion compensation unit 2105 may determine a

predicted video block for the current video block based on the motion information and decoded samples of pictures from buffer 2113 other than the picture associated with the current video block.

[00311] Motion estimation unit 2104 and motion compensation unit 2105 may perform different operations for a current video block, for example, depending on whether the current video block is in an I slice, a P slice, or a B slice.

[00312] In some examples, motion estimation unit 2104 may perform uni-directional prediction for the current video block, and motion estimation unit 2104 may search reference pictures of list 0 or list 1 for a reference video block for the current video block. Motion estimation unit 2104 may then generate a reference index that indicates the reference picture in list 0 or list 1 that contains the reference video block and a motion vector that indicates a spatial displacement between the current video block and the reference video block. Motion estimation unit 2104 may output the reference index, a prediction direction indicator, and the motion vector as the motion information of the current video block. Motion compensation unit 2105 may generate the predicted video block of the current block based on the reference video block indicated by the motion information of the current video block.

[00313] In other examples, motion estimation unit 2104 may perform bi-directional prediction for the current video block, motion estimation unit 2104 may search the reference pictures in list 0 for a reference video block for the current video block and may also search the reference pictures in list 1 for another reference video block for the current video block. Motion estimation unit 2104 may then generate reference indexes that indicate the reference pictures in list 0 and list 1 containing the reference video blocks and motion vectors that indicate spatial displacements between the reference video blocks and the current video block. Motion estimation unit 2104 may output the reference indexes and the motion vectors of the current video block as the motion information of the current video block. Motion compensation unit 2105 may generate the predicted video block of the current video block based on the reference video blocks indicated by the motion information of the current video block.

[00314] In some examples, motion estimation unit 2104 may output a full set of motion information for decoding processing of a decoder. In some examples, motion estimation unit 2104 may not output a full set of motion information for the current video. Rather, motion estimation unit 2104 may signal the motion information of the current video block with reference to the motion information of another video block. For example, motion estimation unit 2104 may determine that

the motion information of the current video block is sufficiently similar to the motion information of a neighboring video block.

[00315] In one example, motion estimation unit 2104 may indicate, in a syntax structure associated with the current video block, a value that indicates to the video decoder 2200 (FIG. 20) that the current video block has the same motion information as another video block.

[00316] In another example, motion estimation unit 2104 may identify, in a syntax structure associated with the current video block, another video block and a motion vector difference (MVD). The motion vector difference indicates a difference between the motion vector of the current video block and the motion vector of the indicated video block. The video decoder 2200 may use the motion vector of the indicated video block and the motion vector difference to determine the motion vector of the current video block.

[00317] As discussed above, video encoder 2100 may predictively signal the motion vector. Two examples of predictive signaling techniques that may be implemented by video encoder 2100 include advanced motion vector prediction (AMVP) and merge mode signaling.

[00318] Intra prediction unit 2106 may perform intra prediction on the current video block. When intra prediction unit 2106 performs intra prediction on the current video block, intra prediction unit 2106 may generate prediction data for the current video block based on decoded samples of other video blocks in the same picture. The prediction data for the current video block may include a predicted video block and various syntax elements.

[00319] Residual generation unit 2107 may generate residual data for the current video block by subtracting the predicted video block(s) of the current video block from the current video block. The residual data of the current video block may include residual video blocks that correspond to different sample components of the samples in the current video block.

[00320] In other examples, there may be no residual data for the current video block, for example in a skip mode, and residual generation unit 2107 may not perform the subtracting operation.

[00321] Transform unit 2108 may generate one or more transform coefficient video blocks for the current video block by applying one or more transforms to a residual video block associated with the current video block.

[00322] After transform processing 2108 generates a transform coefficient video block associated with the current video block, quantization unit 2109 may quantize the transform

coefficient video block associated with the current video block based on one or more quantization parameter (QP) values associated with the current video block.

[00323] Inverse quantization unit 2110 and inverse transform unit 2111 may apply inverse quantization and inverse transforms to the transform coefficient video block, respectively, to reconstruct a residual video block from the transform coefficient video block. Reconstruction unit 2112 may add the reconstructed residual video block to corresponding samples from one or more predicted video blocks generated by the prediction unit 2102 to produce a reconstructed video block associated with the current block for storage in the buffer 2113.

[00324] After reconstruction unit 2112 reconstructs the video block, the loop filtering operation may be performed to reduce video blocking artifacts in the video block.

[00325] Entropy encoding unit 2114 may receive data from other functional components of the video encoder 2100. When entropy encoding unit 2114 receives the data, entropy encoding unit 2114 may perform one or more entropy encoding operations to generate entropy encoded data and output a bitstream that includes the entropy encoded data.

[00326] FIG. 20 is a block diagram that illustrates an example video decoder 2200 which may be video decoder 2024 in the system 2000 illustrated in FIG. 18.

[00327] The video decoder 2200 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 20, the video decoder 2200 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of the video decoder 2200. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[00328] In the example of FIG. 20, video decoder 2200 includes an entropy decoding unit 2201, a motion compensation unit 2202, an intra prediction unit 2203, an inverse quantization unit 2204, an inverse transformation unit 2205, a reconstruction unit 2206, and a buffer 2207. Video decoder 2200 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 2100 (FIG. 19).

[00329] Entropy decoding unit 2201 may retrieve an encoded bitstream. The encoded bitstream may include entropy coded video data (e.g., encoded blocks of video data). Entropy decoding unit 2201 may decode the entropy coded video data, and from the entropy decoded video data, motion compensation unit 2202 may determine motion information including motion vectors, motion vector

precision, reference picture list indexes, and other motion information. Motion compensation unit 2202 may, for example, determine such information by performing the AMVP and merge mode.

[00330] Motion compensation unit 2202 may produce motion compensated blocks, possibly performing interpolation based on interpolation filters. Identifiers for interpolation filters to be used with sub-pixel precision may be included in the syntax elements.

[00331] Motion compensation unit 2202 may use interpolation filters as used by video encoder 2100 during encoding of the video block to calculate interpolated values for sub-integer pixels of a reference block. Motion compensation unit 2202 may determine the interpolation filters used by video encoder 2100 according to received syntax information and use the interpolation filters to produce predictive blocks.

[00332] Motion compensation unit 2202 may use some of the syntax information to determine sizes of blocks used to encode frame(s) and/or slice(s) of the encoded video sequence, partition information that describes how each macroblock of a picture of the encoded video sequence is partitioned, modes indicating how each partition is encoded, one or more reference frames (and reference frame lists) for each inter coded block, and other information to decode the encoded video sequence.

[00333] Intra prediction unit 2203 may use intra prediction modes, for example, received in the bitstream to form a prediction block from spatially adjacent blocks. Inverse quantization unit 2204 inverse quantizes, i.e., de-quantizes, the quantized video block coefficients provided in the bitstream and decoded by entropy decoding unit 2201. Inverse transform unit 2205 applies an inverse transform.

[00334] Reconstruction unit 2206 may sum the residual blocks with the corresponding prediction blocks generated by motion compensation unit 2202 or intra prediction unit 2203 to form decoded blocks. If desired, a deblocking filter may also be applied to filter the decoded blocks in order to remove blockiness artifacts. The decoded video blocks are then stored in buffer 2207, which provides reference blocks for subsequent motion compensation/intra prediction and also produces decoded video for presentation on a display device.

[00335] FIG. 21 is a schematic diagram of an example encoder 2300. The encoder 2300 is suitable for implementing the techniques of VVC. The encoder 2300 includes three in-loop filters, namely a deblocking filter (DF) 2302, a sample adaptive offset (SAO) 2304, and an adaptive loop filter (ALF) 2306. Unlike the DF 2302, which uses predefined filters, the SAO 2304 and the ALF

2306 utilize the original samples of the current picture to reduce the mean square errors between the original samples and the reconstructed samples by adding an offset and by applying a finite impulse response (FIR) filter, respectively, with coded side information signaling the offsets and filter coefficients. The ALF 2306 is located at the last processing stage of each picture and can be regarded as a tool trying to catch and fix artifacts created by the previous stages.

[00336] The encoder 2300 further includes an intra prediction component 2308 and a motion estimation/compensation (ME/MC) component 2310 configured to receive input video. The intra prediction component 2308 is configured to perform intra prediction, while the ME/MC component 2310 is configured to utilize reference pictures obtained from a reference picture buffer 2312 to perform inter prediction. Residual blocks from inter prediction or intra prediction are fed into a transform (T) component 2314 and a quantization (Q) component 2316 to generate quantized residual transform coefficients, which are fed into an entropy coding component 2318. The entropy coding component 2318 entropy codes the prediction results and the quantized transform coefficients and transmits the same toward a video decoder (not shown). Quantization components output from the quantization component 2316 may be fed into an inverse quantization (IQ) components 2320, an inverse transform (IT) component 2322, and a reconstruction (REC) component 2324. The REC component 2324 is able to output images to the DF 2302, the SAO 2304, and the ALF 2306 for filtering prior to those images being stored in the reference picture buffer 2312.

[00337] A listing of solutions preferred by some examples is provided next.

[00338] The following solutions show examples of techniques discussed herein.

[00339] 1. A video processing method, comprising: determining, for a conversion between a video block of a video and a bitstream of the video, whether a partitioning scheme is enabled for coding of the video block according to a rule; and performing the conversion based on the determining, wherein the rule is based on a color characteristic of the video or the video block.

[00340] 2. The method of solution 1, wherein the color characteristic comprises a color component of the video block.

[00341] 3. The method of solutions 1-2, wherein the color characteristic comprises a color format of the video.

[00342] 4. The method of solutions 1-3, wherein the color characteristic comprises whether the video is coded using separate plane coding.

[00343] 5. The method of any of solutions 1-4, wherein the rule specifies a type of unsymmetrical tree partitioning disallowed due to a video block being a luma block or a chroma block.

[00344] 6. A method of video processing, comprising: determining, for a conversion between a video block of a video with dimensions WxH and a bitstream of the video, whether a partitioning scheme is enabled for coding of the video block according to a rule; and performing the conversion based on the determining, wherein the rule is based on the dimensions of the video block and/or another partitioning scheme that is a disallowed partitioning scheme for the video block.

[00345] 7. The method of solution 6, wherein the partitioning scheme is a horizontal or a vertical unsymmetric binary tree (UBT) or a horizontal or a vertical unsymmetric quad tree (UQT), or a horizontal or a vertical extended ternary tree (ETT) partition.

[00346] 8. The method of any of solutions 6-7, wherein the disallowed partitioning scheme includes binary tree (BT), ternary tree (TT), UBT, UQT, ETT, or quadtree (QT).

[00347] 9. The method of any of solutions 7-8, wherein UBT, UQT or ETT is not allowed responsive to W or H being larger than a maximum allowed size for a split using the disallowed partitioning scheme.

[00348] 10. The method of solution 6, wherein the rule specifies that the partitioning scheme is disabled in response to W or H satisfying one or more conditions.

[00349] 11. The method of solution 10, wherein the partitioning scheme includes unsymmetric binary tree (UBT), unsymmetric quadtree (UQT) or extended ternary tree (ETT), multi-type tree (MTT), binary tree (BT), or ternary tree (TT).

[00350] 12. The method of any of solutions 10-11, wherein the one or more conditions are based on thresholds $T1 = t1 \gg S1$ and $T2 = t2 \gg S2$.

[00351] 13. The method of solution 10, wherein the one or more conditions are dependent on a color format of the video or a color component of the video block or decoded information of the video block.

[00352] 14. The method of solutions 12-13, wherein the rule specifies that in case that W is greater than T1 and H is smaller than or equal to T2, the partitioning scheme is disabled for the video block.

[00353] 15. The method of solutions 12-13, wherein the rule specifies that in case that H is greater than T1 and W is smaller than or equal to T2, the partitioning scheme is disabled for the video block.

[00354] 16. A method of video processing, comprising: determining, for a conversion between a video block of a video and a bitstream of the video, whether further partitioning of the video block

using a partitioning scheme is allowed based on a rule; and performing the conversion based on the determining, wherein the rule is based on a multi-type tree depth (MTT-depth) or a binary tree depth (BT-depth) of the video block.

[00355] 17. The method of solution 16, wherein the partitioning scheme includes unsymmetric binary tree (UBT), unsymmetric quadtree (UQT) or extended ternary tree (ETT).

[00356] 18. The method of solutions 16-17, wherein the MTT-depth of the partitioning scheme is derived from a maximum allowed depth of another partitioning scheme comprising quadtree (QT), multi-tree type (MTT), binary tree (BT) or ternary tree (TT).

[00357] 19. The method of solutions 16-18, wherein the bitstream indicated the MTT-depth.

[00358] 20. The method of any of solutions 16-19, wherein the rule specifies that the partitioning scheme is disallowed responsive to the current MTT-depth being larger than a maximum allowed depth of another partitioning scheme.

[00359] 21. The method of any of solutions 16-19, wherein the rule specifies that the partitioning scheme is disallowed responsive to the current MTT-depth being smaller than a minimum allowed depth of another partitioning scheme.

[00360] 22. A method of video processing, comprising: determining, for a conversion between a video block of a video and a bitstream of the video, whether a partitioning scheme is allowed for partitioning the video block; and performing the conversion based on the determining, wherein the rule is based on whether a local-dual tree is used for the video block.

[00361] 23. The method of solution 22, wherein the partitioning scheme includes unsymmetric binary tree (UBT), unsymmetric quadtree (UQT) or extended ternary tree (ETT).

[00362] 24. The method of solution 22-23, wherein the rule specifies that the partitioning scheme is disallowed in case that the local-dual tree is used for the current video block.

[00363] 25. The method of any of solutions 23-24, wherein UBT, UQT, or ETT is not allowed responsive to W or H being smaller than a minimum allowed size for a split using the disallowed partitioning scheme.

[00364] 26. The method of any of solutions 1-25, wherein the conversion includes generating the video from the bitstream.

[00365] 27. The method of any of solutions 1-25, wherein the conversion includes generating the bitstream from the video.

[00366] 28. A method of storing a bitstream on a computer-readable medium, comprising generating a bitstream according to a method recited in any one or more of solutions 1-26 and storing the bitstream on the computer-readable medium.

[00367] 29. A computer-readable medium having a bitstream of a video stored thereon, the bitstream, when processed by a processor of a video decoder, causing the video decoder to generate the video, wherein the bitstream is generated according to a method recited in one or more of solutions 1-27.

[00368] 30. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of solutions 1-27.

[00369] 31. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of solutions 1-27.

[00370] 32. A computer program product having computer code stored thereon, the code, when executed by a processor, causes the processor to implement a method recited in any of solutions 1 to 27.

[00371] 33. A computer readable medium on which a bitstream complying to a bitstream format that is generated according to any of solutions 1-27.

[00372] 34. A method, an apparatus, a bitstream generated according to a disclosed method or a system described in the present document.

[00373] In the solutions described herein, an encoder may conform to the format rule by producing a coded representation according to the format rule. In the solutions described herein, a decoder may use the format rule to parse syntax elements in the coded representation with the knowledge of presence and absence of syntax elements according to the format rule to produce decoded video.

[00374] In the present document, the term “video processing” may refer to video encoding, video decoding, video compression, or video decompression. For example, video compression algorithms may be applied during conversion from pixel representation of a video to a corresponding bitstream representation or vice versa. The bitstream representation of a current video block may, for example, correspond to bits that are either co-located or spread in different places within the bitstream, as is defined by the syntax. For example, a macroblock may be encoded in terms of transformed and coded error residual values and also using bits in headers and other fields in the bitstream. Furthermore, during conversion, a decoder may parse a bitstream with the knowledge that some

fields may be present, or absent, based on the determination, as is described in the above solutions. Similarly, an encoder may determine that certain syntax fields are or are not to be included and generate the coded representation accordingly by including or excluding the syntax fields from the coded representation.

[00375] The disclosed and other solutions, examples, embodiments, modules and the functional operations described in this document can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this document and their structural equivalents, or in combinations of one or more of them. The disclosed and other embodiments can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more of them. The term “data processing apparatus” encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[00376] A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[00377] The processes and logic flows described in this document can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[00378] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random-access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and compact disc read-only memory (CD ROM) and Digital versatile disc-read only memory (DVD-ROM) disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00379] While this patent document contains many specifics, these should not be construed as limitations on the scope of any subject matter or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular techniques. Certain features that are described in this patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[00380] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. Moreover, the separation of various system components in the embodiments described in this patent document should not be understood as requiring such separation in all embodiments.

[00381] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.

[00382] A first component is directly coupled to a second component when there are no intervening components, except for a line, a trace, or another medium between the first component and the second component. The first component is indirectly coupled to the second component when there are intervening components other than a line, a trace, or another medium between the first component and the second component. The term “coupled” and its variants include both directly coupled and indirectly coupled. The use of the term “about” means a range including $\pm 10\%$ of the subsequent number unless otherwise stated.

[00383] While several embodiments have been provided in the present disclosure, it should be understood that the disclosed systems and methods might be embodied in many other specific forms without departing from the spirit or scope of the present disclosure. The present examples are to be considered as illustrative and not restrictive, and the intention is not to be limited to the details given herein. For example, the various elements or components may be combined or integrated in another system or certain features may be omitted, or not implemented.

[00384] In addition, techniques, systems, subsystems, and methods described and illustrated in the various embodiments as discrete or separate may be combined or integrated with other systems, modules, techniques, or methods without departing from the scope of the present disclosure. Other items shown or discussed as coupled may be directly connected or may be indirectly coupled or communicating through some interface, device, or intermediate component whether electrically, mechanically, or otherwise. Other examples of changes, substitutions, and alterations are ascertainable by one skilled in the art and could be made without departing from the spirit and scope disclosed herein.

CLAIMS

What is claimed is:

1. A method for processing video data comprising:
determining, for a conversion between a video media data and a bitstream, whether a partitioning scheme is enabled for coding of the video media data according to a rule; and
performing the conversion based on the determined rule.
2. The method of claim 1, wherein the rule is based on a color characteristic of the video media data.
3. The method of claim 2, wherein the color characteristic comprises at least one of a color component of the visual media data or whether the visual media data is coded using separate plane coding.
4. The method of any of claims 2 or 3, wherein the rule specifies a type of unsymmetrical tree partitioning disallowed due to a video block being a luma block or a chroma block.
5. The method of claim 1, wherein a video unit of the visual media data has a dimension $W \times H$, where W is the width of the video unit in pixels and H is the height of the video unit in pixels, and wherein the rule is based on the dimensions of a video block and/or another partitioning scheme that is a disallowed partitioning scheme for the video block.
6. The method of claim 5, wherein the partitioning scheme is a horizontal or a vertical unsymmetric binary tree (UBT), or a horizontal or a vertical unsymmetric quad tree (UQT), or a horizontal or a vertical extended ternary tree (ETT) partition.
7. The method of claim 5, wherein at least one of:
the disallowed partitioning scheme includes binary tree (BT), ternary tree (TT), UBT, UQT, ETT, or quadtree (QT);

UBT, UQT or ETT is not allowed responsive to W or H being larger than a maximum allowed size for a split using the disallowed partitioning scheme;

the rule specifies that the partitioning scheme is disabled in response to W or H satisfying one or more conditions; or

the partitioning scheme includes unsymmetric binary tree (UBT), unsymmetric quadtree (UQT) or extended ternary tree (ETT), multi-type tree (MTT), binary tree (BT), or ternary tree (TT).

8. The method of any of claims 5-7, wherein the one or more conditions are based on thresholds $T1 = t1 \gg S1$ and $T2 = t2 \gg S2$, where S1 and S2 are based on a color characteristic of the video media data and t1 and t2 are based on a split of the video unit;

sides of two sub-blocks split from a current block of the visual media data with a side S and where T1 and T1 thresholds for the block and t1 and t2 are thresholds for the sub-blocks are and where $\gg$ are right bit shift operators.

9. The method of claim 8, wherein the one or more conditions are dependent on a color format of the visual media data or a color component of a video unit of the visual media data or decoded information of the video unit.

10. The method of claims 8-9, wherein the rule specifies that in case W is greater than T1 and H is smaller than or equal to T2, the partitioning scheme is disabled for the video unit.

11. The method of claims 8-9, wherein the rule specifies that in case H is greater than T1 and W is smaller than or equal to T2, the partitioning scheme is disabled for the video unit.

12. The method of claim 1, wherein the rule is based on a multi-type tree depth (MTT-depth) or a binary tree depth (BT-depth) of a video unit of the visual media data.

13. The method of claim 12, wherein the partitioning scheme includes unsymmetric binary tree (UBT), unsymmetric quadtree (UQT), or extended ternary tree (ETT).

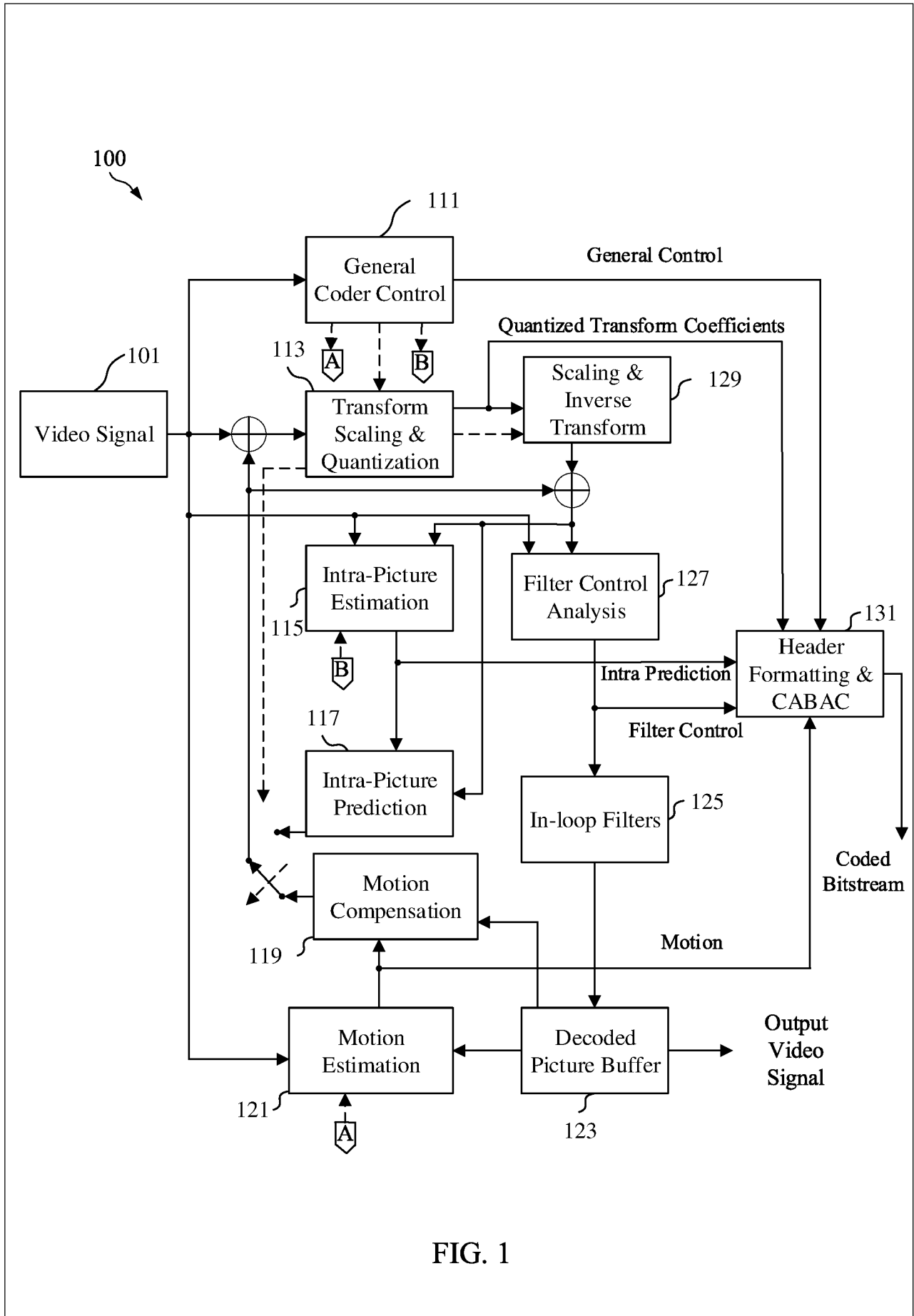
14. The method of claims 12-13, wherein the MTT-depth of the partitioning scheme is derived from a maximum allowed depth of another partitioning scheme comprising quadtree (QT), multi-tree type (MTT), binary tree (BT) or ternary tree (TT).
15. The method of claims 12-14, wherein the bitstream indicates the MTT-depth.
16. The method of any of claims 12-15, wherein the rule specifies at least one of:
that the partitioning scheme is disallowed responsive to the current MTT-depth being larger than a maximum allowed depth of another partitioning scheme; or
that the partitioning scheme is disallowed responsive to the current MTT-depth being smaller than a minimum allowed depth of another partitioning scheme.
17. The method of claim 1, wherein the rule is based on whether a local-dual tree is used for a video unit of the visual media data.
18. The method of claim 17 wherein the partitioning scheme includes unsymmetric binary tree (UBT), unsymmetric quadtree (UQT), or extended ternary tree (ETT).
19. The method of claims 17-18, wherein the rule specifies that the partitioning scheme is disallowed in case that the local-dual tree is used for the current video block or that UBT, UQT, or ETT is not allowed responsive to W or H being smaller than a minimum allowed size for a split using the disallowed partitioning scheme.
20. A non-transitory computer readable medium comprising a computer program product for use by a video coding device, the computer program product comprising computer executable instructions stored on the non-transitory computer readable medium such that when executed by a processor cause the video coding device to perform the method of claims 1-19.
21. An apparatus for processing video data comprising: a processor; and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to perform the method of claims 1-19.

22. A non-transitory computer-readable of any of the preceding claims, the non-transitory computer readable medium recording medium storing a bitstream of a video which is generated by a method performed by a video processing apparatus, wherein the method comprises:

determining, for a conversion between a video media data and a bitstream, whether a partitioning scheme is enabled for coding of the video media data according to a rule; and
generating the bitstream based on the determining.

23. A method of any of the preceding claims for storing bitstream of a video, comprising:

determining, for a conversion between a video media data and a bitstream, whether a partitioning scheme is enabled for coding of the video media data according to a rule;
generating the bitstream based on the determining; and
storing the bitstream in a non-transitory computer-readable recording medium.



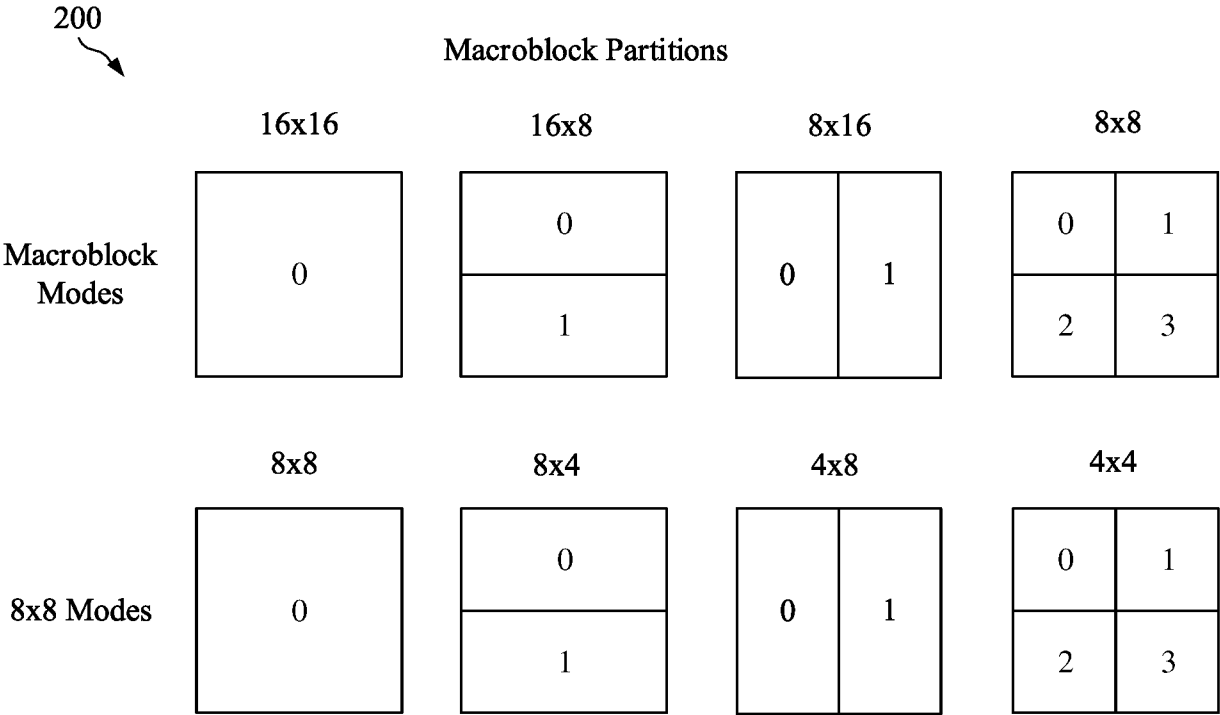


FIG. 2

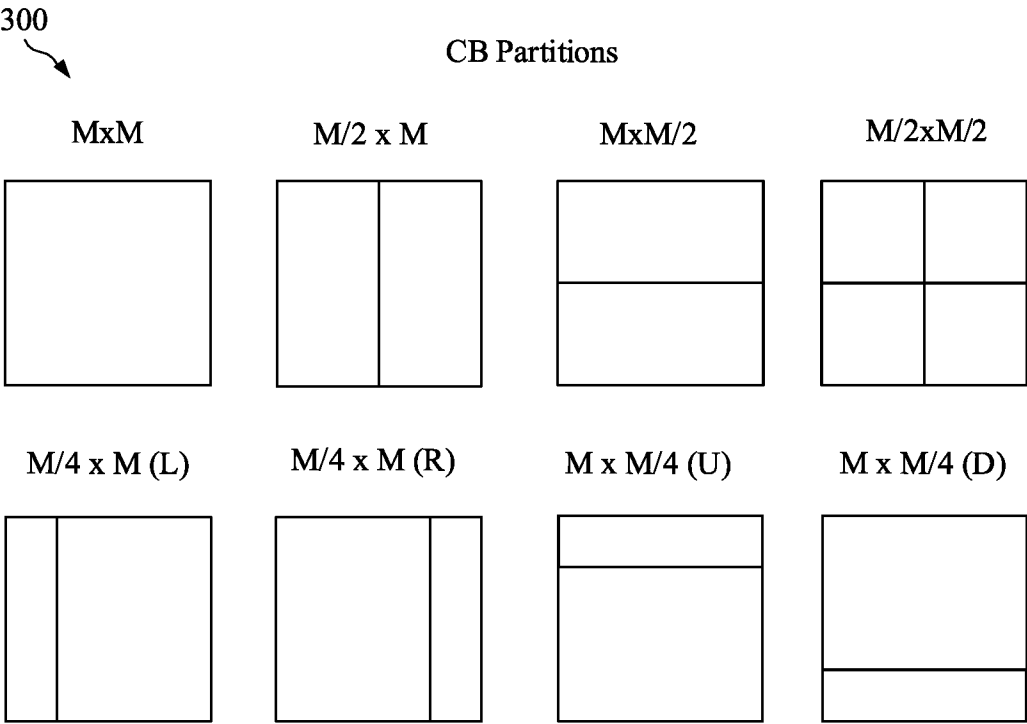
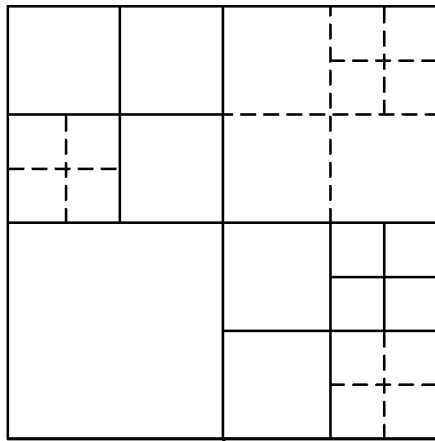


FIG. 3

400

Macroblock Partitions



401

403

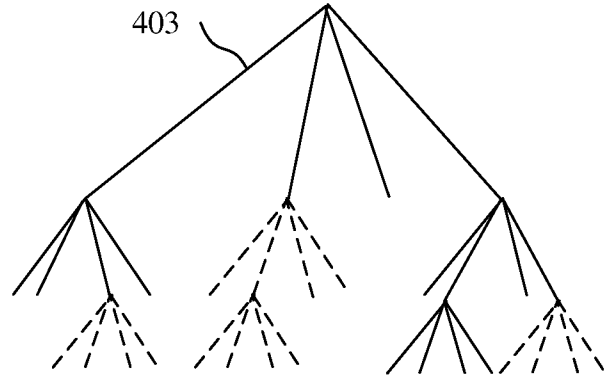
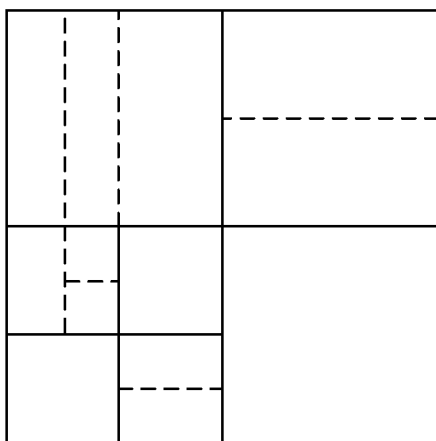


FIG. 4

500

QTBT Partitions



501

503

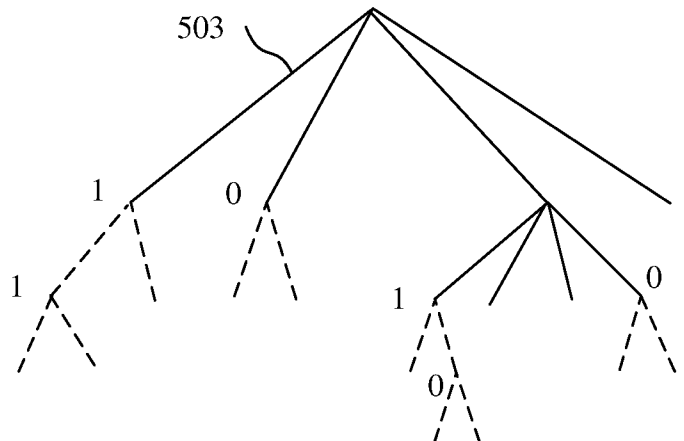


FIG. 5

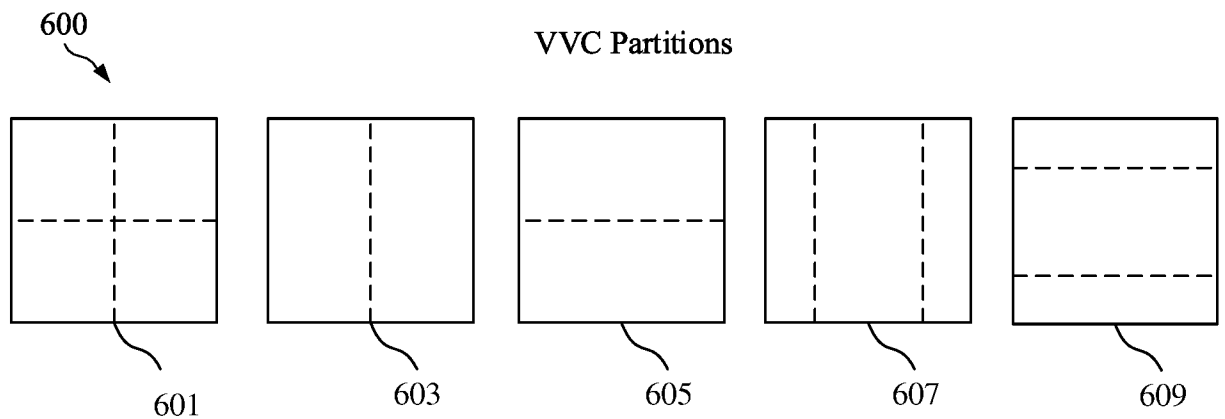


FIG. 6

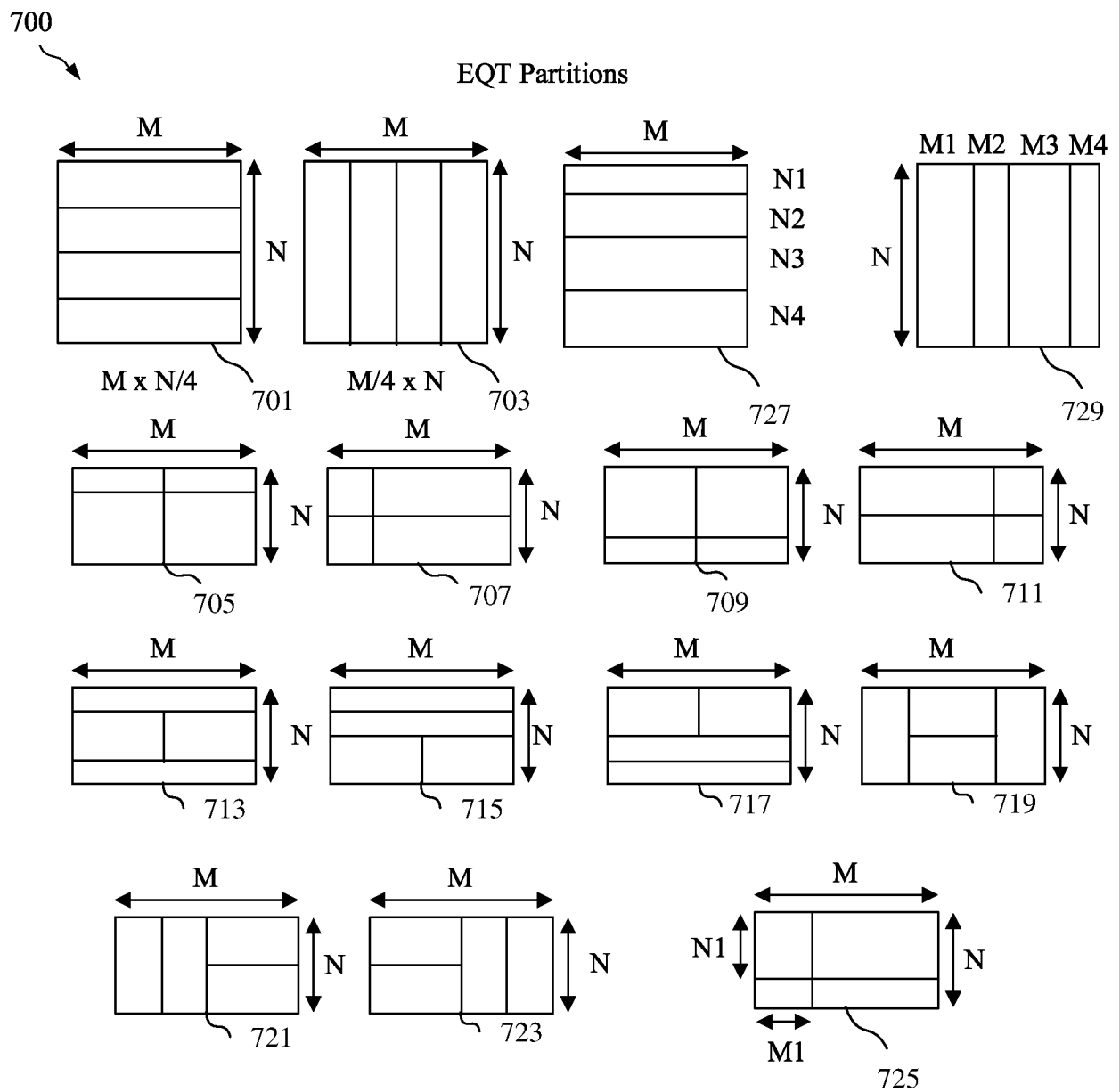


FIG. 7

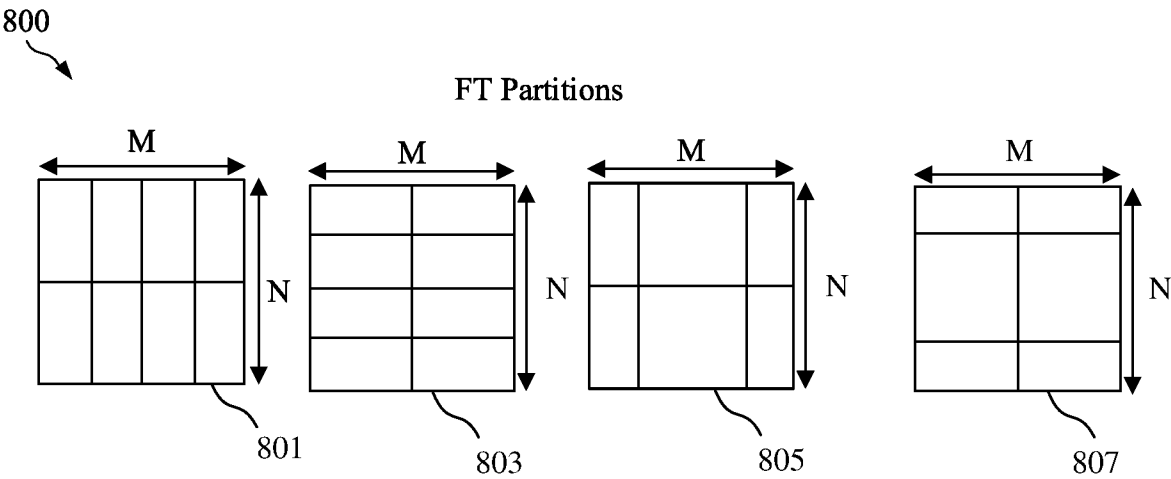


FIG.8

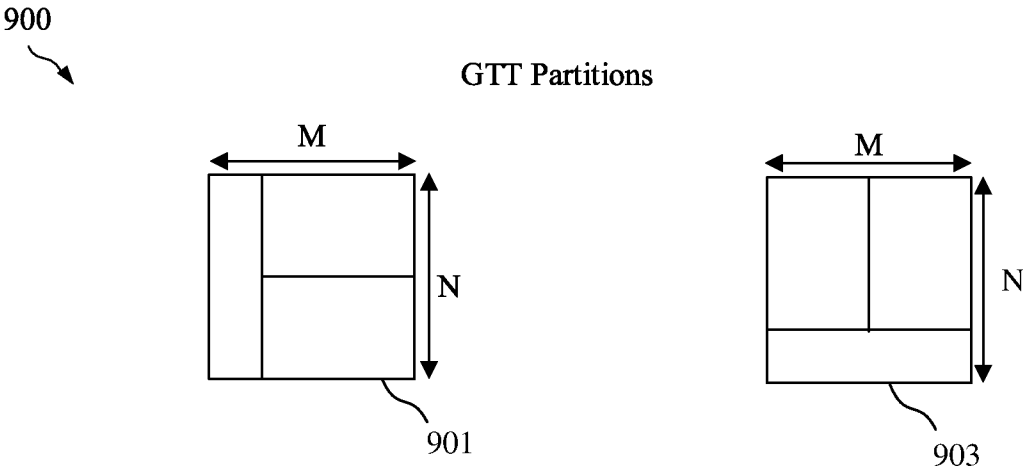


FIG. 9

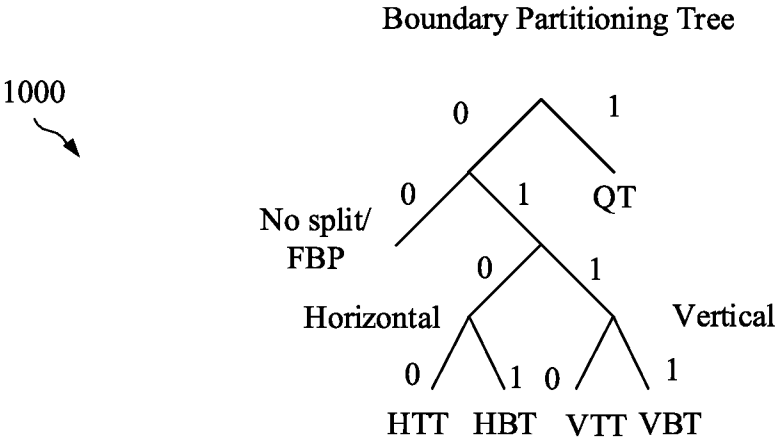


FIG. 10

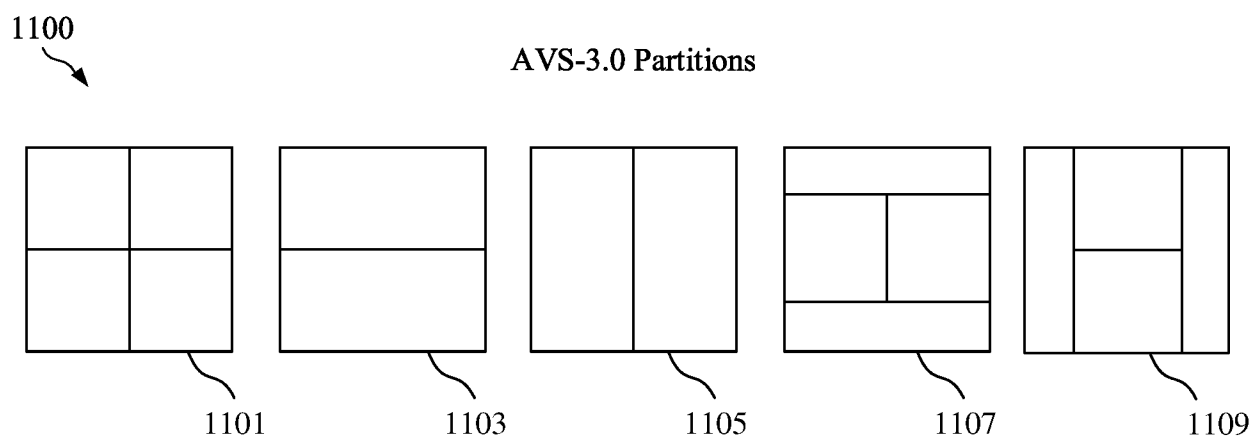


FIG. 11

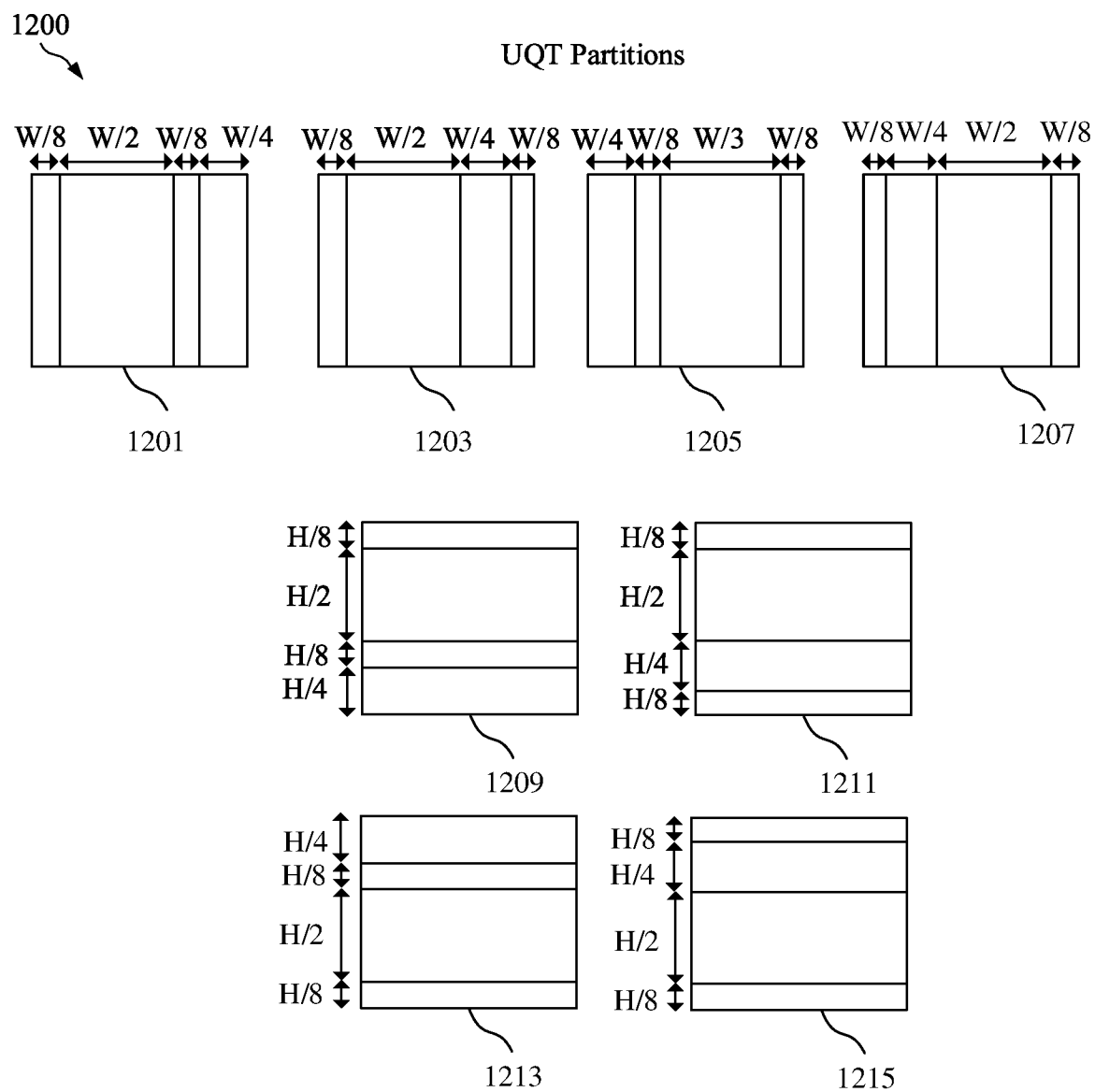


FIG. 12

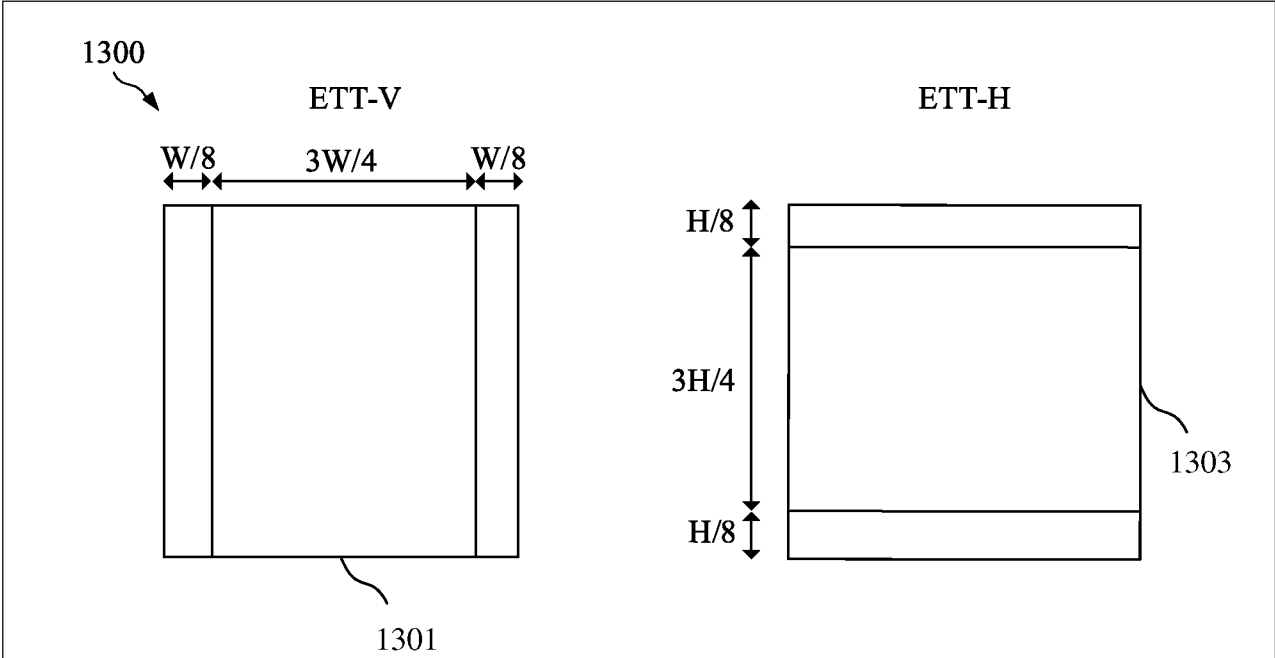


FIG. 13

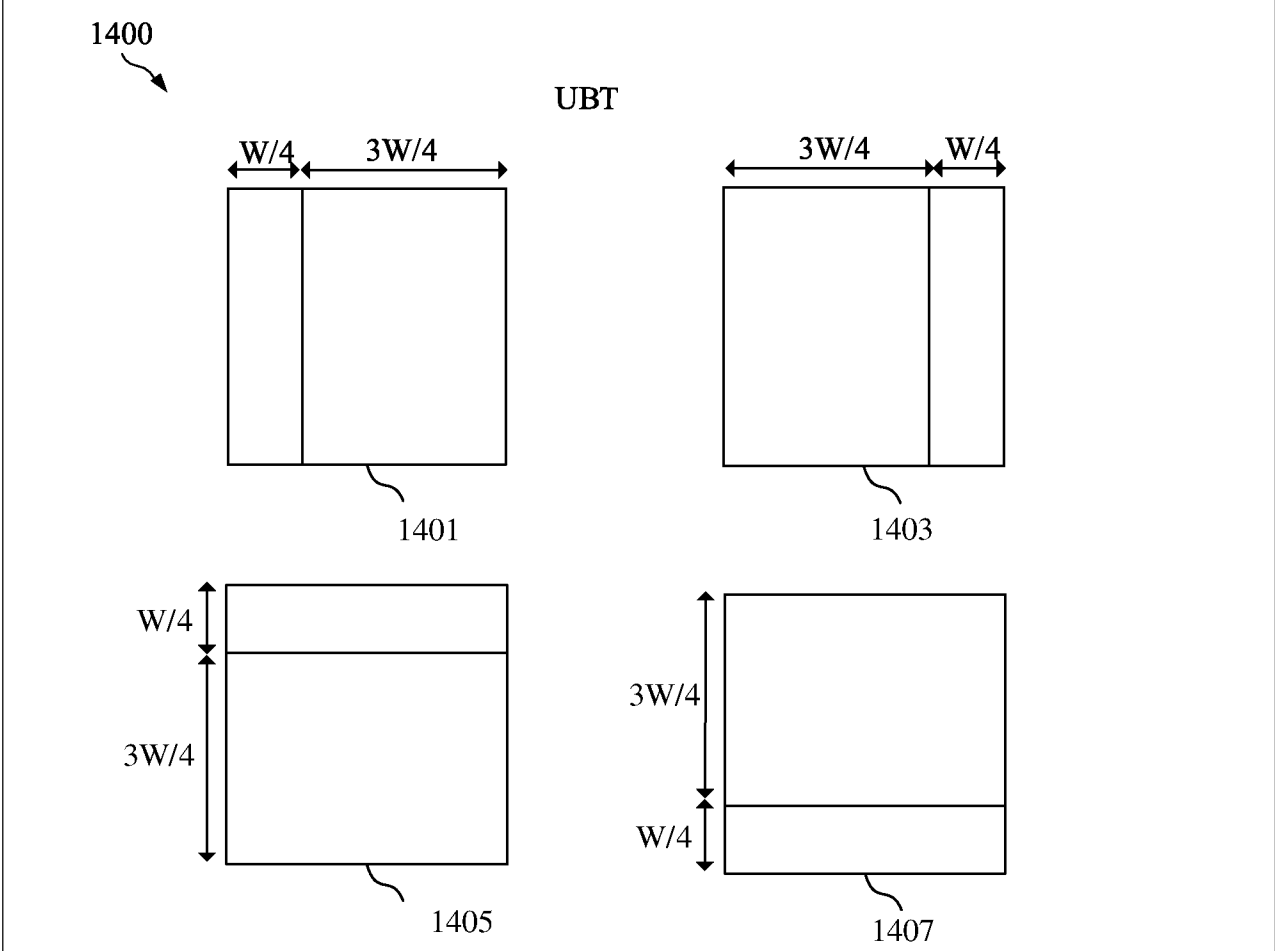


FIG. 14

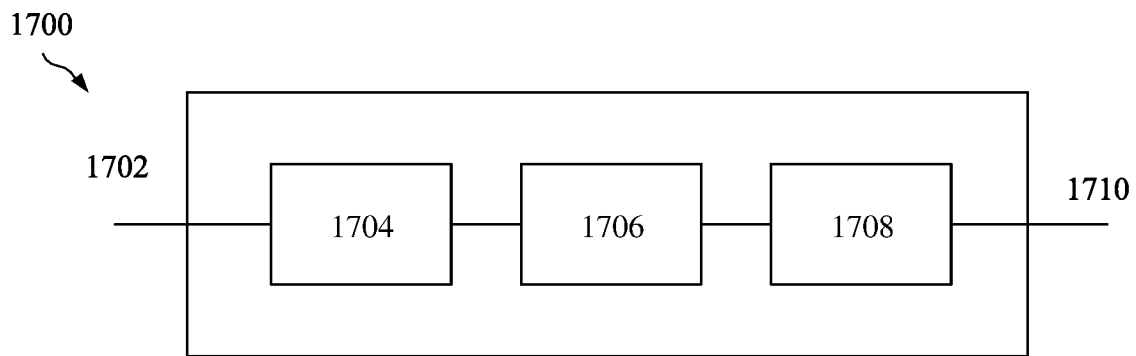


FIG. 15

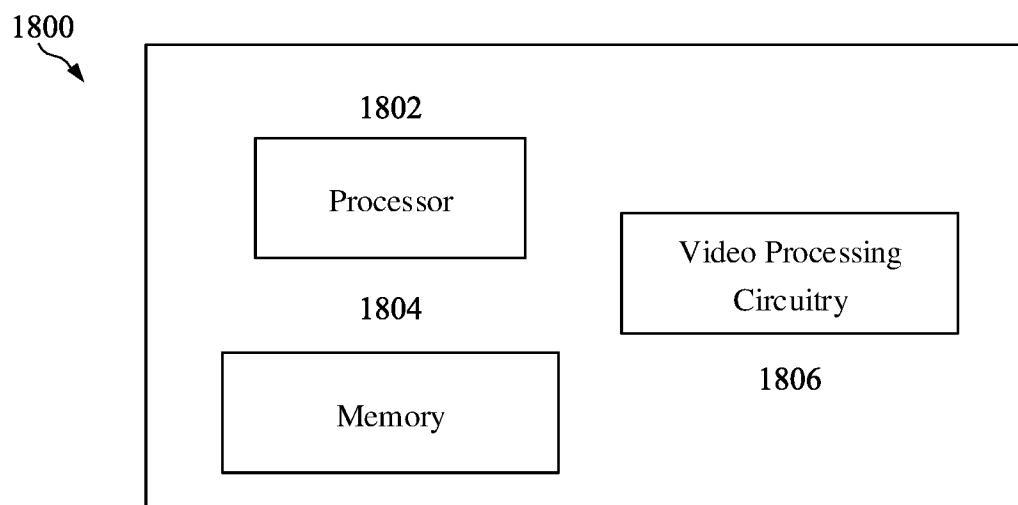


FIG. 16

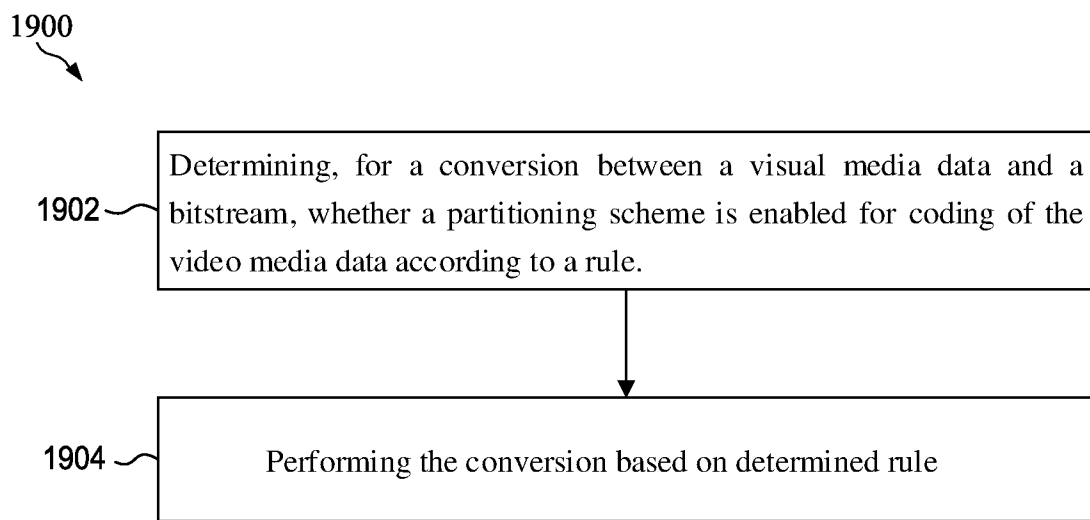


FIG. 17

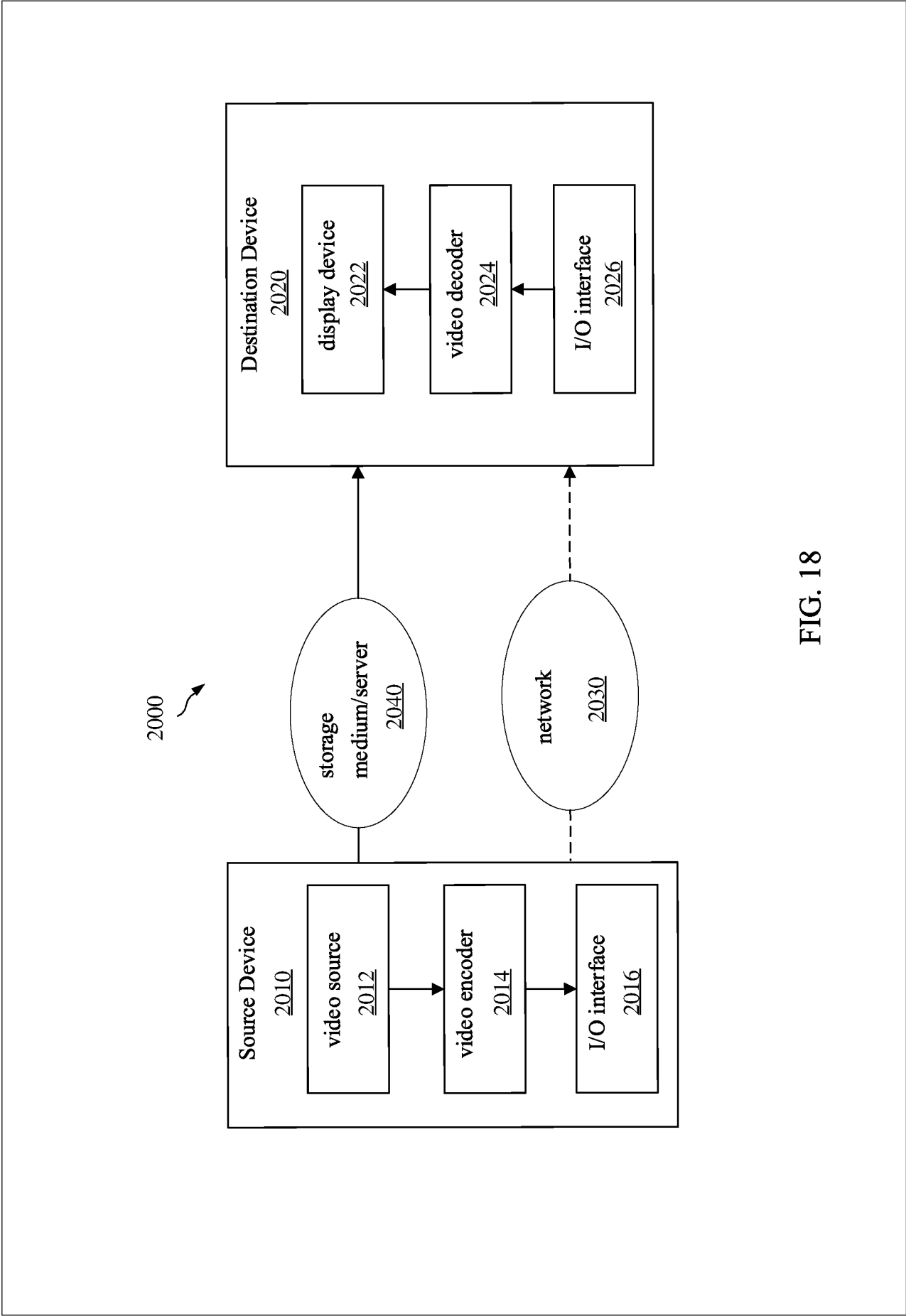


FIG. 18

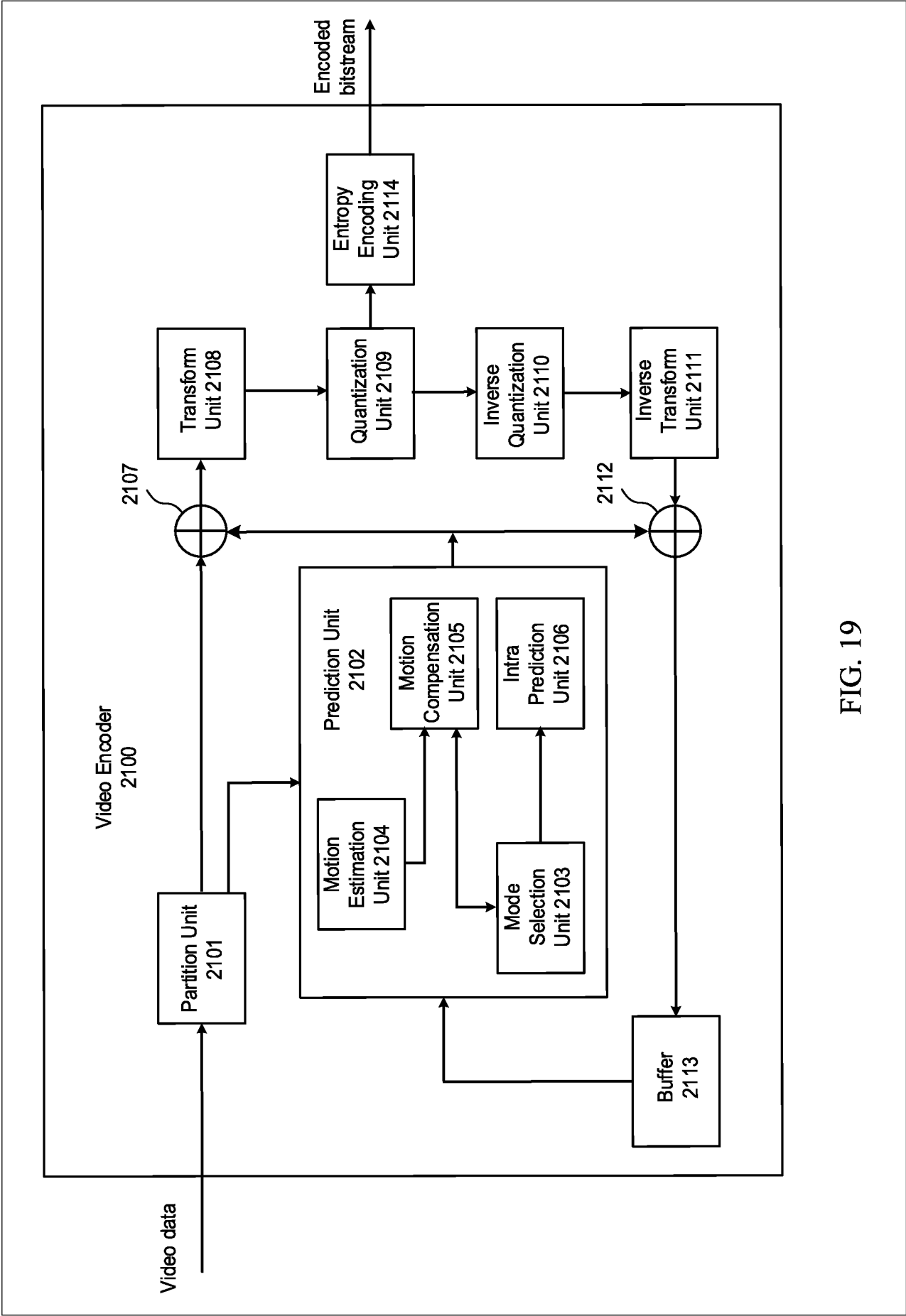


FIG. 19

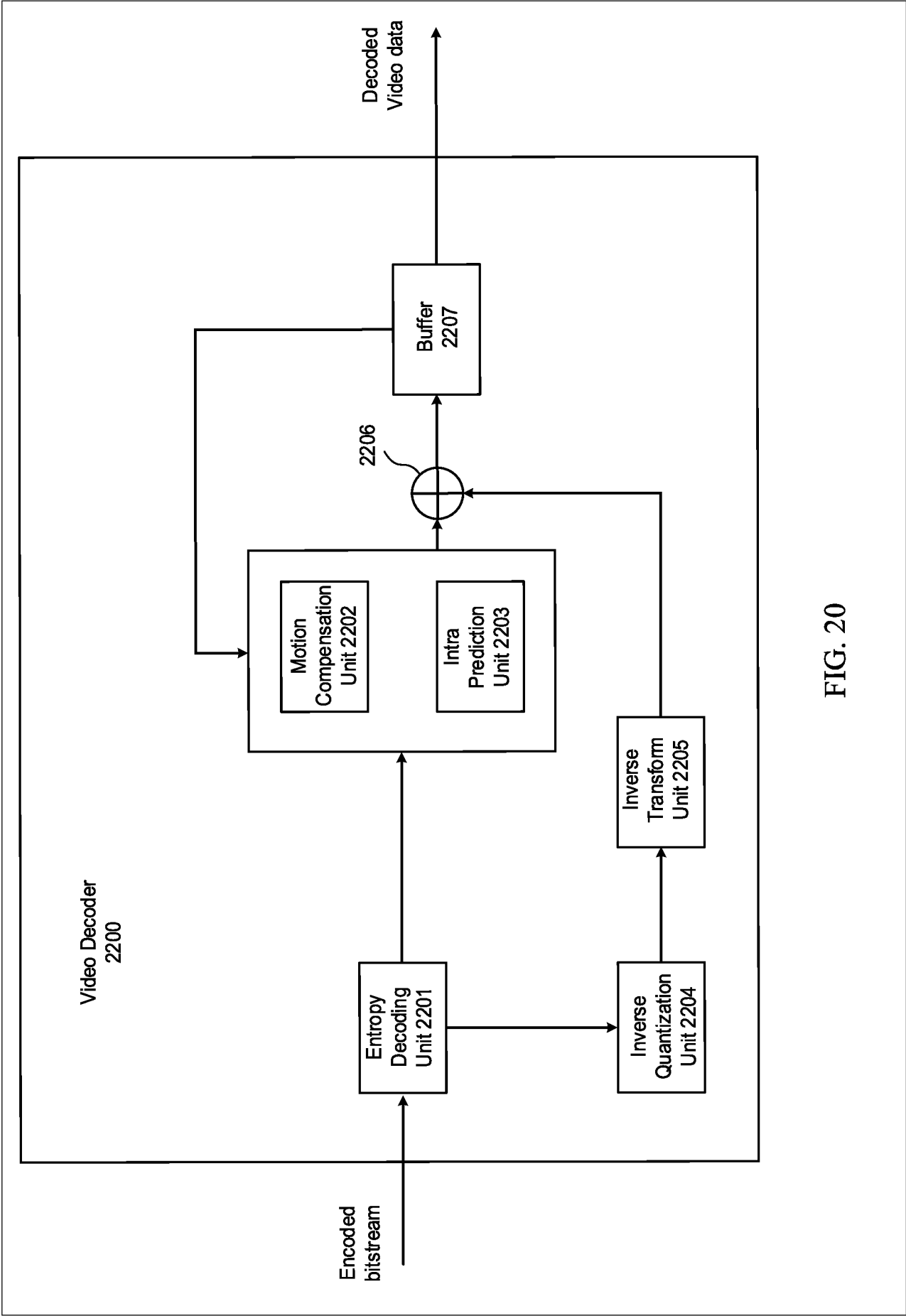


FIG. 20

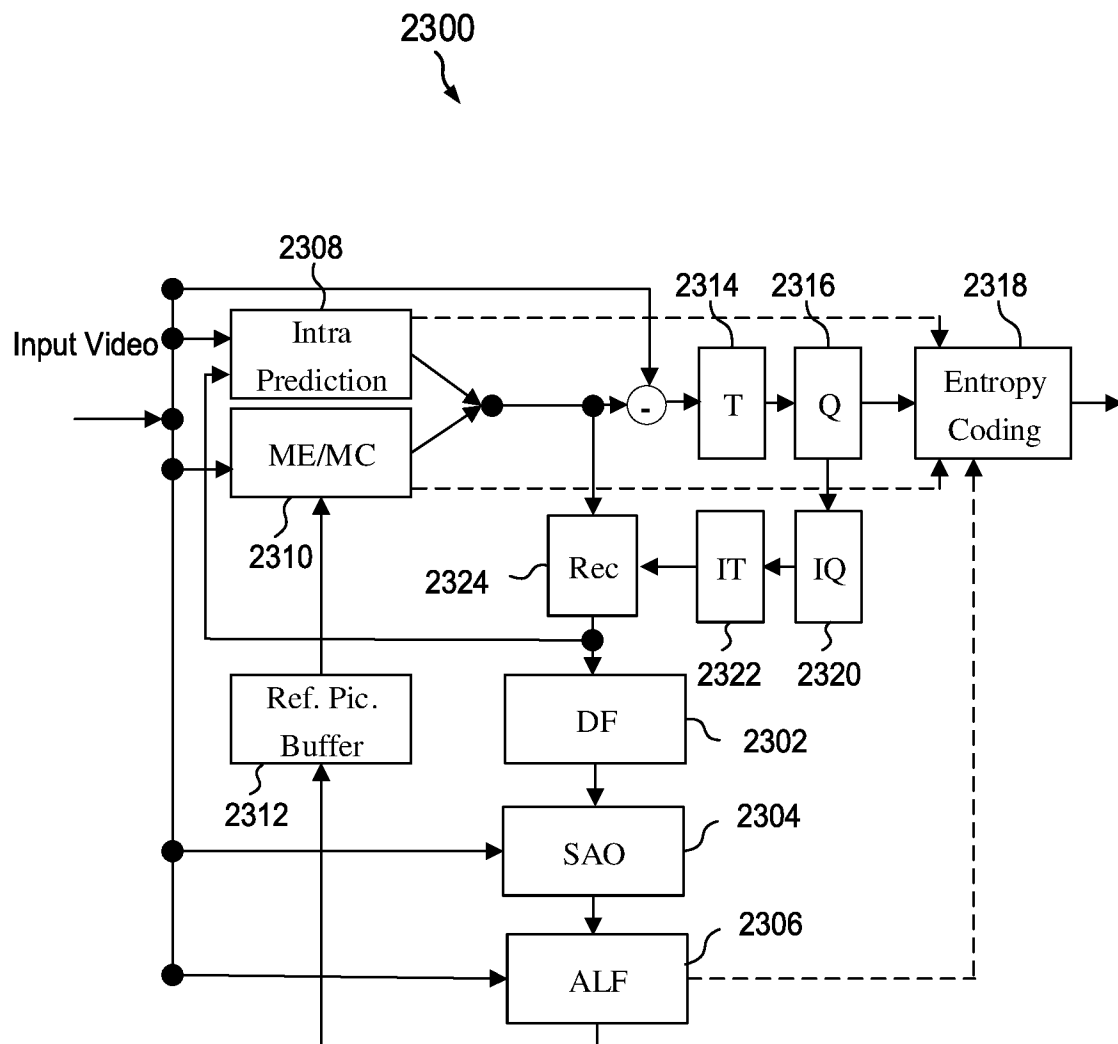


FIG. 21

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2022/085056

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/50(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

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

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

CNKI,CNPAT,DWPI,EPODOC,IEEE:video, data, split, bitstream, partition+, scheme, coding, decoding, rule, color, tree, binary, ternary, quadtree, luma, chroma, sub w block?, depth

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2013116081 A2 (QUALCOMM INCORPORATED) 08 August 2013 (2013-08-08) description, paragraphs 0026-0114, figures 1-11	1-6, 12-16, 20-23
A	WO 2020185009 A1 (HYUNDAI MOTOR COMPANY et al.) 17 September 2020 (2020-09-17) the whole document	1-23
A	WO 2016090568 A1 (MEDIATEK SINGAPORE PTE. LTD.) 16 June 2016 (2016-06-16) the whole document	1-23
A	WO 2018056702 A1 (KT CORPORATION) 29 March 2018 (2018-03-29) the whole document	1-23
A	WO 2014163461 A1 (SAMSUNG ELECTRONICS CO., LTD.) 09 October 2014 (2014-10-09) the whole document	1-23



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

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

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

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

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

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

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

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

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

“&” document member of the same patent family

Date of the actual completion of the international search

15 June 2022

Date of mailing of the international search report

01 July 2022

Name and mailing address of the ISA/CN

National Intellectual Property Administration, PRC
6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing
100088, China

Facsimile No. (86-10)62019451

Authorized officer

WANG,Jing

Telephone No. 86-(10)-53961317

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2022/085056

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
WO	2013116081	A2	08 August 2013	JP	2015508953	A	23 March 2015
				US	2013195199	A1	01 August 2013
				EP	2810437	A2	10 December 2014
				KR	20140123978	A	23 October 2014
				CN	104081777	A	01 October 2014
WO	2020185009	A1	17 September 2020	CN	113574877	A	29 October 2021
				KR	20200110220	A	23 September 2020
				IN	202117046306	A	04 February 2022
WO	2016090568	A1	16 June 2016	MX	2019010795	A	24 October 2019
				KR	20170077203	A	05 July 2017
				CN	111314695	A	19 June 2020
				US	2018098097	A1	05 April 2018
				SG	10201805297	A1	30 July 2018
WO	2018056702	A1	29 March 2018	US	2019268623	A1	29 August 2019
				US	2021076078	A1	11 March 2021
				EP	3975573	A1	30 March 2022
				US	2021067810	A1	04 March 2021
				US	2021076077	A1	11 March 2021
				EP	3518548	A1	31 July 2019
				ES	2711230	A2	30 April 2019
				KR	20180031615	A	28 March 2018
				CN	109716775	A	03 May 2019
				IN	201917010970	A	21 June 2019
				IN	202118037899	A	11 February 2022
WO	2014163461	A1	09 October 2014	EP	2966866	A1	13 January 2016
				CN	105308966	A	03 February 2016
				US	2016037178	A1	04 February 2016
				KR	20140122196	A	17 October 2014