



(51) International Patent Classification:

H04N 19/11 (2014.01) H04N 19/523 (2014.01)
H04N 19/117 (2014.01) H04N 19/557 (2014.01)
H04N 19/119 (2014.01) H04N 19/577 (2014.01)
H04N 19/122 (2014.01) H04N 19/625 (2014.01)
H04N 19/176 (2014.01) H04N 19/82 (2014.01)
H04N 19/186 (2014.01)

(21) International Application Number:

PCT/IB2019/056095

(22) International Filing Date:

17 July 2019 (17.07.2019)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

PCT/CN2018/095918

17 July 2018 (17.07.2018)

CN

PCT/CN2018/106661

20 September 2018 (20.09.2018) 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, CA 90066 (US).

(72) Inventors: **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, CA 90066 (US). **ZHANG, Li**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, CA 90066 (US). **LIU, Hongbin**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN). **CHUANG, Hsiao Chiang**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, CA 90066 (US). **WANG, Yue**; 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,

(54) Title: BLOCK SIZE RESTRICTIONS FOR VISUAL MEDIA CODING

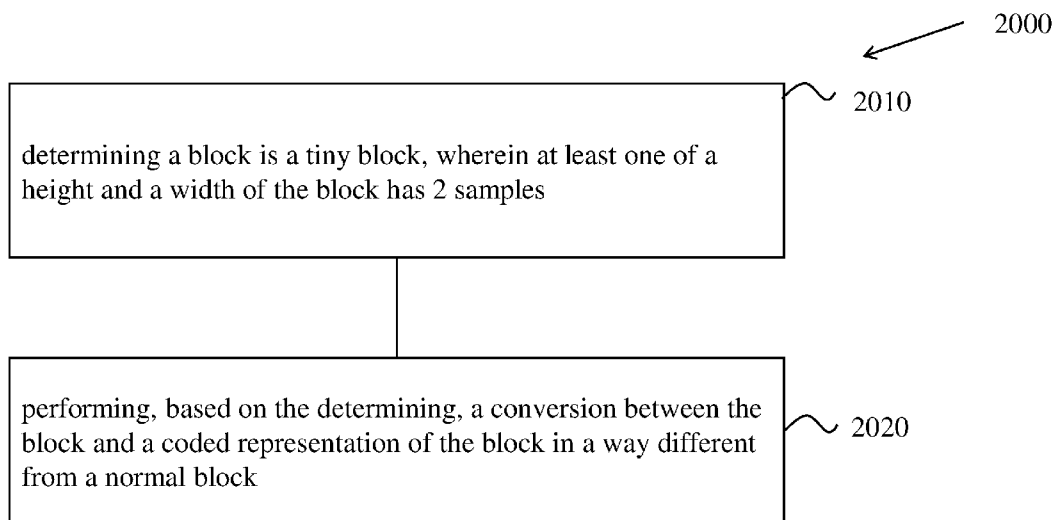


FIG. 20

(57) Abstract: Method and apparatus for video bitstream processing, and the method includes: determining a block is a tiny block, wherein at least one of a height and a width of the block has 2 samples; and performing, based on the determining, a conversion between the block and a coded representation of the block in a way different from a normal block.

CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

- *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*
- *in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE*

BLOCK SIZE RESTRICTIONS FOR VISUAL MEDIA CODING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Applications No. PCT/CN2018/095918, filed on July 17, 2018 and No. PCT/CN2018/106661, filed on September 20, 2018. The entire disclosure of the International Patent Applications No. PCT/CN2018/095918 and NO. PCT/CN2018/106661 is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[0002] This patent document is directed generally to image and video decoding and encoding technologies.

BACKGROUND

[0003] Digital video accounts for the largest bandwidth use on the internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, it is expected that the bandwidth demand for digital video usage will continue to grow.

SUMMARY

[0004] Devices, systems and methods related to using block size restrictions to perform video coding methods are described. For example, the presently disclosed technology discloses selecting a prediction mode or determining whether to split the block of video data (e.g., in a picture, slice, tile and the like) based on a property (or characteristic) of the luma or chroma components of the block of video data. The described methods may be applied to both the existing video coding standards (e.g., High Efficiency Video Coding (HEVC)) and future video coding standards or video codecs.

[0005] In one example aspect, a video processing method is disclosed. The method includes: receiving a bitstream representation of video data including a video block wherein the video

block comprises samples in a first component and in a second component; deriving a first coding information for the first component from second coding information of sub-blocks for the second component in case that the video block for the second component is split into the sub-blocks; performing a conversion between the video block and the bitstream representation of the video block based on the first coding information.

[0006] In another example aspect, a method for video decoding is disclosed. The method comprises receiving a bitstream representation of video data including a video block wherein the video block comprises samples in a first component, the samples in the first component having a dimension of $M \times N$; determining, based on one or more of specific conditions is satisfied, a first prediction mode for decoding the first component of the block is not a bi-prediction; and decoding the first component by using the first prediction mode

[0007] In another example aspect, a method for video decoding is disclosed. The method comprises: receiving a bitstream representation of video data including a video block wherein the video block comprises samples in a first component and samples in a second component; determining a first prediction mode for decoding the first component of the block and determining a second prediction mode for decoding the second component of the block; decoding the first and second component by using the first and the second prediction mode respectively.

[0008] In another example aspect, a method for video decoding is disclosed. The method comprises receiving a bitstream representation of video data including a block wherein the block comprises samples associated with a first component and second components, wherein samples associated with the first component of the block have a dimension $M \times N$; and decoding the first component and the second components of the block; wherein decoding the first component of the block comprises, based on the dimension, decoding a plurality of sub-blocks for the first component of the block, and the plurality of the sub-blocks are generated by performing a splitting operation only on the samples associated with the first component of the block and not on the samples associated with the second components of the block.

[0009] In another example aspect, a method for video bitstream processing is disclosed. The method includes: determining a block is a tiny block, wherein at least one of a height and a width of the block has 2 samples; and performing, based on the determining, a conversion between the block and a coded representation of the block in a way different from a normal block.

[0010] In yet another representative aspect, the above-described method is embodied in the form of processor-executable code and stored in a computer-readable program medium.

[0011] In yet another representative aspect, a device that is configured or operable to perform the above-described method is disclosed. The device may include a processor that is programmed to implement this method.

[0012] In yet another representative aspect, a video decoder apparatus may implement a method as described herein.

[0013] The above and other aspects and features of the disclosed technology are described in greater detail in the drawings, the description and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] FIG. 1 shows an example block diagram of a typical High Efficiency Video Coding (HEVC) video encoder and decoder.

[0015] FIG. 2 shows examples of macroblock (MB) partitions in H.264/AVC.

[0016] FIG. 3 shows examples of splitting coding blocks (CBs) into prediction blocks (PBs).

[0017] FIGS. 4A and 4B show an example of the subdivision of a coding tree block (CTB) into CBs and transform blocks (TBs), and the corresponding quadtree, respectively.

[0018] FIG. 5 shows an example of a partition structure of one frame.

[0019] FIGS. 6A and 6B show the subdivisions and signaling methods, respectively, of a CTB highlighted in the exemplary frame in FIG. 5.

[0020] FIGS. 7A and 7B show an example of the subdivisions and a corresponding QTBT (quadtree plus binary tree) for a largest coding unit (LCU).

[0021] FIGS. 8A–8E show examples of partitioning a coding block.

[0022] FIG. 9 shows an example subdivision of a CB based on a QTBT.

[0023] FIGS. 10A–10I show examples of the partitions of a CB supported the multi-tree type (MTT), which is a generalization of the QTBT.

[0024] FIG. 11 shows an example of tree-type signaling.

[0025] FIGS. 12A–12C show examples of CTBs crossing picture borders.

[0026] FIG. 13 shows an example encoding/decoding/signaling order if the luma component can be split but the chroma components cannot.

[0027] FIG. 14 shows a flowchart of an example method for video coding in accordance with

the presently disclosed technology.

[0028] FIG. 15 is a block diagram of an example of a hardware platform for implementing a visual media decoding or a visual media encoding technique described in the present document.

[0029] FIG. 16 shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

[0030] FIG. 17 shows a flowchart of another example method for video decoding in accordance with the presently disclosed technology.

[0031] FIG. 18 shows a flowchart of another example method for video decoding in accordance with the presently disclosed technology.

[0032] FIG. 19 shows a flowchart of another example method for video decoding in accordance with the presently disclosed technology.

[0033] FIG. 20 shows a flowchart of another example method for video decoding in accordance with the presently disclosed technology.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0034] Due to the increasing demand of higher resolution video, video coding methods and techniques are ubiquitous in modern technology. Video codecs typically include an electronic circuit or software that compresses or decompresses digital video, and are continually being improved to provide higher coding efficiency. A video codec converts uncompressed video to a compressed format or vice versa. There are complex relationships between the video quality, the amount of data used to represent the video (determined by the bit rate), the complexity of the encoding and decoding algorithms, sensitivity to data losses and errors, ease of editing, random access, and end-to-end delay (latency). The compressed format usually conforms to a standard video compression specification, e.g., the High Efficiency Video Coding (HEVC) standard (also known as H.265 or MPEG-H Part 2), the Versatile Video Coding standard to be finalized, or other current and/or future video coding standards.

[0035] Embodiments of the disclosed technology may be applied to existing video coding standards (e.g., HEVC, H.265) and future standards to improve compression performance. Section headings are used in the present document to improve readability of the description and do not in any way limit the discussion or the embodiments (and/or implementations) to the respective sections only.

1. Example embodiments of video coding

[0036] FIG. 1 shows an example block diagram of a typical HEVC video encoder and decoder (Reference [1]). An encoding algorithm producing an HEVC compliant bitstream would typically proceed as follows. Each picture is split into block-shaped regions, with the exact block partitioning being conveyed to the decoder. The first picture of a video sequence (and the first picture at each clean random access point into a video sequence) is coded using only intra-picture prediction (that uses some prediction of data spatially from region-to-region within the same picture, but has no dependence on other pictures). For all remaining pictures of a sequence or between random access points, inter-picture temporally predictive coding modes are typically used for most blocks. The encoding process for inter-picture prediction consists of choosing motion data comprising the selected reference picture and motion vector (MV) to be applied for predicting the samples of each block. The encoder and decoder generate identical inter-picture prediction signals by applying motion compensation (MC) using the MV and mode decision data, which are transmitted as side information.

[0037] The residual signal of the intra- or inter-picture prediction, which is the difference between the original block and its prediction, is transformed by a linear spatial transform. The transform coefficients are then scaled, quantized, entropy coded, and transmitted together with the prediction information.

[0038] The encoder duplicates the decoder processing loop (see gray-shaded boxes in FIG. 1) such that both will generate identical predictions for subsequent data. Therefore, the quantized transform coefficients are constructed by inverse scaling and are then inverse transformed to duplicate the decoded approximation of the residual signal. The residual is then added to the prediction, and the result of that addition may then be fed into one or two loop filters to smooth out artifacts induced by block-wise processing and quantization. The final picture representation (that is a duplicate of the output of the decoder) is stored in a decoded picture buffer to be used for the prediction of subsequent pictures. In general, the order of encoding or decoding processing of pictures often differs from the order in which they arrive from the source; necessitating a distinction between the decoding order (i.e., bitstream order) and the output order (i.e., display order) for a decoder.

[0039] Video material to be encoded by HEVC is generally expected to be input as progressive scan imagery (either due to the source video originating in that format or resulting

from deinterlacing prior to encoding). No explicit coding features are present in the HEVC design to support the use of interlaced scanning, as interlaced scanning is no longer used for displays and is becoming substantially less common for distribution. However, a metadata syntax has been provided in HEVC to allow an encoder to indicate that interlace-scanned video has been sent by coding each field (i.e., the even or odd numbered lines of each video frame) of interlaced video as a separate picture or that it has been sent by coding each interlaced frame as an HEVC coded picture. This provides an efficient method of coding interlaced video without burdening decoders with a need to support a special decoding process for it.

1.1. Examples of partition tree structures in H.264/AVC

[0040] The core of the coding layer in previous standards was the macroblock, containing a 16×16 block of luma samples and, in the usual case of 4:2:0 color sampling, two corresponding 8×8 blocks of chroma samples.

[0041] An intra-coded block uses spatial prediction to exploit spatial correlation among pixels. Two partitions are defined: 16×16 and 4×4.

[0042] An inter-coded block uses temporal prediction, instead of spatial prediction, by estimating motion among pictures. Motion can be estimated independently for either 16×16 macroblock or any of its sub-macroblock partitions: 16×8, 8×16, 8×8, 8×4, 4×8, 4×4, as shown in FIG. 2. Only one motion vector (MV) per sub-macroblock partition is allowed.

1.2 Examples of partition tree structures in HEVC

[0043] In HEVC, a coding tree unit (CTU) is split into coding units (CUs) by using a quadtree structure denoted as 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 prediction units (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 of key feature of the HEVC structure is that it has the multiple partition conceptions including CU, PU, and TU.

[0044] Certain features involved in hybrid video coding using HEVC include:

[0045] (1) Coding tree units (CTUs) and coding tree block (CTB) structure: The analogous

structure in HEVC is the coding tree unit (CTU), which has a size selected by the encoder and can be larger than a traditional macroblock. The CTU consists of a luma CTB and the corresponding chroma CTBs and syntax elements. The size $L \times L$ of a luma CTB can be chosen as $L = 16, 32$, or 64 samples, with the larger sizes typically enabling better compression. HEVC then supports a partitioning of the CTBs into smaller blocks using a tree structure and quadtree-like signaling.

[0046] (2) Coding units (CUs) and coding blocks (CBs): The quadtree syntax of the CTU specifies the size and positions of its 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 ordinarily 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, and each CU has an associated partitioning into prediction units (PUs) and a tree of transform units (TUs).

[0047] (3) Prediction units and prediction blocks (PBs): The decision whether to code a picture area using inter picture or intra picture prediction is made at the CU level. A PU partitioning structure has its 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). HEVC supports variable PB sizes from 64×64 down to 4×4 samples. FIG. 3 shows examples of allowed PBs for an $M \times M$ CU.

[0048] (4) Transform units (TUs) and transform blocks: The prediction residual is coded using block transforms. A TU tree structure has its root at the CU level. The luma CB residual may be identical to the luma transform block (TB) or may be further split into smaller luma TBs. The same applies to the chroma TBs. Integer basis functions similar to those of a discrete cosine transform (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 discrete sine transform (DST) is alternatively specified.

1.2.1. Examples of tree-structured partitioning into TBs and TUs

[0049] For residual coding, a CB can be recursively partitioned into transform blocks (TBs). The partitioning is signaled by a residual quadtree. Only square CB and TB partitioning is specified, where a block can be recursively split into quadrants, as illustrated in FIG. 4. For a given luma CB of size $M \times M$, a flag signals whether it is split into four blocks of size $M/2 \times M/2$.

If further splitting is possible, as signaled by a maximum depth of the residual quadtree indicated in the SPS, each quadrant is assigned a flag that indicates whether it is split into four quadrants. The leaf node blocks resulting from the residual quadtree are the transform blocks that are further processed by transform coding. The encoder indicates the maximum and minimum luma TB sizes that it will use. Splitting is implicit when the CB size is larger than the maximum TB size. Not splitting is implicit when splitting would result in a luma TB size smaller than the indicated minimum. The chroma TB size is half the luma TB size in each dimension, except when the luma TB size is 4×4, in which case a single 4×4 chroma TB is used for the region covered by four 4×4 luma TBs. In the case of intra-picture-predicted CUs, the decoded samples of the nearest-neighboring TBs (within or outside the CB) are used as reference data for intra picture prediction.

[0050] In contrast to previous standards, the HEVC design allows a TB to span across multiple PBs for inter-picture predicted CUs to maximize the potential coding efficiency benefits of the quadtree-structured TB partitioning.

1.2.2. Examples of picture border coding

[0051] The borders of the picture are defined in units of the minimally allowed luma CB size. As a result, at the right and bottom borders of the picture, some CTUs may cover regions that are partly outside the borders of the picture. This condition is detected by the decoder, and the CTU quadtree is implicitly split as necessary to reduce the CB size to the point where the entire CB will fit into the picture.

[0052] FIG. 5 shows an example of a partition structure of one frame, with a resolution of 416×240 pixels and dimensions 7 CTBs × 4 CTBs, wherein the size of a CTB is 64×64. As shown in FIG. 5, the CTBs that are partially outside the right and bottom border have implied splits (dashed lines, indicated as 502), and the CUs that fall outside completely are simply skipped (not coded).

[0053] In the example shown in FIG. 5, the highlighted CTB (504), with row CTB index equal to 2 and column CTB index equal to 3, has 64×48 pixels within the current picture, and doesn't fit a 64×64 CTB. Therefore, it is forced to be split to 32×32 without the split flag signaled. For the top-left 32×32, it is fully covered by the frame. When it chooses to be coded in smaller blocks (8×8 for the top-left 16×16, and the remaining are coded in 16×16) according to rate-distortion cost, several split flags need to be coded. These split flags (one for whether split

the top-left 32x32 to four 16x16 blocks, and flags for signaling whether one 16x16 is further split and 8x8 is further split for each of the four 8x8 blocks within the top-left 16x16) have to be explicitly signaled. A similar situation exists for the top-right 32x32 block. For the two bottom 32x32 blocks, since they are partially outside the picture border (506), further QT split needs to be applied without being signaled. FIGS. 6A and 6B show the subdivisions and signaling methods, respectively, of the highlighted CTB (504) in FIG. 5.

1.2.3. Examples of CTB size indications

[0054] An example RBSP (raw byte sequence payload) syntax table for the general sequence parameter set is shown in Table 1.

[0055] Table 1: RBSP syntax structure

seq_parameter_set_rbsp() {	Descriptor
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_temporal_id_nesting_flag	u(1)
profile_tier_level(1, sps_max_sub_layers_minus1)	
...	
log2_min_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_luma_coding_block_size	ue(v)
log2_min_luma_transform_block_size_minus2	ue(v)
log2_diff_max_min_luma_transform_block_size	ue(v)
..	
}	

[0056] The corresponding semantics includes:

[0057] `log2_min_luma_coding_block_size_minus3` plus 3 specifies the minimum luma coding block size; and

[0058] `log2_diff_max_min_luma_coding_block_size` specifies the difference between the maximum and minimum luma coding block size.

[0059] The variables `MinCbLog2SizeY`, `CtbLog2SizeY`, `MinCbSizeY`, `CtbSizeY`, `PicWidthInMinCbsY`, `PicWidthInCtbsY`, `PicHeightInMinCbsY`, `PicHeightInCtbsY`, `PicSizeInMinCbsY`, `PicSizeInCtbsY`, `PicSizeInSamplesY`, `PicWidthInSamplesC` and

PicHeightInSamplesC are derived as follows:

- [0060] $\text{MinCbLog2SizeY} = \log_2_min_luma_coding_block_size_minus3 + 3$
- [0061] $\text{CtbLog2SizeY} = \text{MinCbLog2SizeY} + \log_2_diff_max_min_luma_coding_block_size$
- [0062] $\text{MinCbSizeY} = 1 \ll \text{MinCbLog2SizeY}$
- [0063] $\text{CtbSizeY} = 1 \ll \text{CtbLog2SizeY}$
- [0064] $\text{PicWidthInMinCbsY} = \text{pic_width_in_luma_samples} / \text{MinCbSizeY}$
- [0065] $\text{PicWidthInCtbsY} = \text{Ceil}(\text{pic_width_in_luma_samples} \div \text{CtbSizeY})$
- [0066] $\text{PicHeightInMinCbsY} = \text{pic_height_in_luma_samples} / \text{MinCbSizeY}$
- [0067] $\text{PicHeightInCtbsY} = \text{Ceil}(\text{pic_height_in_luma_samples} \div \text{CtbSizeY})$
- [0068] $\text{PicSizeInMinCbsY} = \text{PicWidthInMinCbsY} * \text{PicHeightInMinCbsY}$
- [0069] $\text{PicSizeInCtbsY} = \text{PicWidthInCtbsY} * \text{PicHeightInCtbsY}$
- [0070] $\text{PicSizeInSamplesY} = \text{pic_width_in_luma_samples} * \text{pic_height_in_luma_samples}$
- [0071] $\text{PicWidthInSamplesC} = \text{pic_width_in_luma_samples} / \text{SubWidthC}$
- [0072] $\text{PicHeightInSamplesC} = \text{pic_height_in_luma_samples} / \text{SubHeightC}$
- [0073] The variables CtbWidthC and CtbHeightC, which specify the width and height, respectively, of the array for each chroma CTB, are derived as follows:
- [0074] If chroma_format_idc is equal to 0 (monochrome) or separate_colour_plane_flag is equal to 1, CtbWidthC and CtbHeightC are both equal to 0;
- [0075] Otherwise, CtbWidthC and CtbHeightC are derived as follows:
- [0076] $\text{CtbWidthC} = \text{CtbSizeY} / \text{SubWidthC}$
- [0077] $\text{CtbHeightC} = \text{CtbSizeY} / \text{SubHeightC}$

1.3. Examples of quadtree plus binary tree block structures with larger CTUs in JEM

[0078] In some embodiments, future video coding technologies (Reference [3]) are explored using a reference software known as the Joint Exploration Model (JEM) (Reference [4]). In addition to binary tree structures, JEM describes quadtree plus binary tree (QTBT) and ternary tree (TT) structures.

1.3.1. Examples of the QTBT block partitioning structure

[0079] In contrast to HEVC (Reference [5]), the QTBT structure removes the concepts of multiple partition types, i.e. it removes the separation of the CU, PU and TU concepts, and supports more flexibility for CU partition shapes. In the QTBT block structure, a CU can have either a square or rectangular shape. As shown in FIG. 7A, a coding tree unit (CTU) is first

partitioned by a quadtree structure. The quadtree leaf nodes are further partitioned by a binary tree structure. There are two splitting types, symmetric horizontal splitting and symmetric vertical splitting, in the binary tree splitting. The binary tree leaf nodes are called coding units (CUs), and that segmentation is used for prediction and transform processing without any further partitioning. This means that the CU, PU and TU have the same block size in the QTBT coding block structure. In the JEM, a CU sometimes consists of coding blocks (CBs) of different colour components, e.g. one CU contains one luma CB and two chroma CBs in the case of P and B slices of the 4:2:0 chroma format and sometimes consists of a CB of a single component, e.g., one CU contains only one luma CB or just two chroma CBs in the case of I slices.

[0080] The following parameters are defined for the QTBT partitioning scheme:

[0081] -- CTU size: the root node size of a quadtree, the same concept as in HEVC

[0082] -- MinQTSIZE: the minimally allowed quadtree leaf node size

[0083] -- MaxBTSIZE: the maximally allowed binary tree root node size

[0084] -- MaxBTDepth: the maximally allowed binary tree depth

[0085] -- MinBTSIZE: the minimally allowed binary tree leaf node size

[0086] In one example of the QTBT partitioning structure, 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 (i.e., the MinQTSIZE) to 128×128 (i.e., the CTU size). If the leaf quadtree node is 128×128, it will not be further split by the binary tree since the size exceeds the MaxBTSIZE (i.e., 64×64). Otherwise, the leaf quadtree node could be further partitioned by the binary tree. Therefore, the quadtree leaf node is also the root node for the binary tree and it has the binary tree depth as 0. When the binary tree depth reaches MaxBTDepth (i.e., 4), no further splitting is considered. When the binary tree node has width equal to MinBTSIZE (i.e., 4), no further horizontal splitting is considered. Similarly, when the binary tree node has 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.

[0087] FIG. 7A shows an example of block partitioning by using QTBT, and FIG. 7B shows

the corresponding tree representation. The solid lines indicate quadtree splitting and dotted lines indicate binary tree splitting. In each splitting (i.e., non-leaf) node of the binary tree, one flag is signalled to indicate which splitting type (i.e., 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.

[0088] In addition, the QTBT scheme supports the ability for the luma and chroma to have a separate QTBT structure. Currently, for P and B slices, the luma and chroma CTBs in one CTU share the same QTBT structure. However, for I slices, the luma CTB is partitioned into CUs by a QTBT structure, and the chroma CTBs are partitioned into chroma CUs by another QTBT structure. This means that a CU in an I slice consists of a coding block of the luma component or coding blocks of two chroma components, and a CU in a P or B slice consists of coding blocks of all three colour components.

[0089] 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.

1.4. Ternary-tree (TT) for Versatile Video Coding (VVC)

[0090] FIG. 8A shows an example of quad-tree (QT) partitioning, and FIGS. 8B and 8C show examples of the vertical and horizontal binary-tree (BT) partitioning, respectively. In some embodiments, and in addition to quad-trees and binary-trees, ternary tree (TT) partitions, e.g., horizontal and vertical center-side ternary-trees (as shown in FIGS. 8D and 8E) are supported.

[0091] In some implementations, two levels of trees are supported: region tree (quad-tree) and prediction tree (binary-tree or ternary-tree). A CTU is firstly partitioned by 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 max PT depth is reached. A PT leaf is the basic coding unit. It is still called CU for convenience. 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’.

1.5. Examples of partitioning structures in alternate video coding technologies

[0092] In some embodiments, a tree structure called a Multi-Tree Type (MTT), which is a generalization of the QTBT, is supported. In QTBT, as shown in FIG. 9, a Coding Tree Unit

(CTU) is firstly partitioned by a quad-tree structure. The quad-tree leaf nodes are further partitioned by a binary-tree structure.

[0093] The structure of the MTT constitutes of two types of tree nodes: Region Tree (RT) and Prediction Tree (PT), supporting nine types of partitions, as shown in FIG. 10. A region tree can recursively split a CTU into square blocks down to a 4x4 size region tree leaf node. At each node in a region tree, a prediction tree can be formed from one of three tree types: Binary Tree, Ternary Tree, and Asymmetric Binary Tree. In a PT split, it is prohibited to have a quadtree partition in branches of the prediction tree. As in JEM, the luma tree and the chroma tree are separated in I slices.

[0094] In general, RT signaling is same as QT signaling in JEM with exception of the context derivation. For PT signaling, up to 4 additional bins are required, as shown in FIG. 11. The first bin indicates whether the PT is further split or not. The context for this bin is calculated based on the observation that the likelihood of further split is highly correlated to the relative size of the current block to its neighbors. If PT is further split, the second bin indicates whether it is a horizontal partitioning or vertical partitioning. In some embodiments, the presence of the center-sided triple tree and the asymmetric binary trees (ABTs) increase the occurrence of “tall” or “wide” blocks. The third bin indicates the tree-type of the partition, i.e., whether it is a binary-tree/triple-tree, or an asymmetric binary tree. In case of a binary-tree/triple-tree, the fourth bin indicates the type of the tree. In case of asymmetric binary trees, the four bin indicates up or down type for horizontally partitioned trees and right or left type for vertically partitioned trees.

1.5.1. Examples of restrictions at picture borders

[0095] In some embodiments, if the CTB/LCU size is indicated by $M \times N$ (typically M is equal to N , as defined in HEVC/JEM), and for a CTB located at picture (or tile or slice or other kinds of types) border, $K \times L$ samples are within picture border.

[0096] The CU splitting rules on the picture bottom and right borders may apply to any of the coding tree configuration QTBT+TT, QTBT+ABT or QTBT+TT+ABT. They include the two following aspects:

[0097] (1) If a part of a given Coding Tree node (CU) is partially located outside the picture, then the binary symmetric splitting of the CU is always allowed, along the concerned border direction (horizontal split orientation along bottom border, as shown in FIG. 12A, vertical split orientation along right border, as shown in FIG. 12B). If the bottom-right corner of the

current CU is outside the frame (as depicted in FIG. 12C), then only the quad-tree splitting of the CU is allowed. In addition, if the current binary tree depth is greater than the maximum binary tree depth and current CU is on the frame border, then the binary split is enabled to ensure the frame border is reached.

[0098] (2) With respect to the ternary tree splitting process, the ternary tree split is allowed in case the first or the second border between resulting sub-CU exactly lies on the border of the picture. The asymmetric binary tree splitting is allowed if a splitting line (border between two sub-CU resulting from the split) exactly matches the picture border.

2. Examples of existing implementations

[0099] Existing implementations enable a flexible block partitioning approach in JEM, VTM or BMS, which brings significant coding gains, but suffers several complexity issues. In one example, the smallest luma block size may be 4×4 . When bi-prediction is applied on a 4×4 block, the required bandwidth is huge.

[00100] In another example, with the 4:2:0 format, the smallest chroma block size is 2×2 . In addition to a similar bandwidth issue as for the luma component, two more issues unfriendly to the hardware design are introduced:

[00101] (i) $2 \times N$ or $N \times 2$ transform and inverse-transform, and

[00102] (ii) $2 \times N$ or $N \times 2$ intra-prediction.

3. Example methods using block size restrictions based on the disclosed technology

[00103] Embodiments of the presently disclosed technology overcome the drawbacks of existing implementations, thereby providing video coding with higher efficiencies. Specifically, the block size of the luma and/or chroma components are used to determine how the video coding is performed, e.g., what prediction mode is selected or whether the block of video data (and the luma and chroma components) are split.

[00104] The use of block-size restrictions to improve video coding efficiency and enhance both existing and future video coding standards is elucidated in the following examples described for various implementations. The examples of the disclosed technology provided below explain general concepts, and are not meant to be interpreted as limiting. In an example, unless explicitly indicated to the contrary, the various features described in these examples may be combined. In another example, the various features described in these examples may be applied to methods for picture border coding that employ block sizes that are backward

compatible and use partition trees for visual media coding.

[00105] Example 1. Suppose the current luma coding block size is $M \times N$, Bi-prediction is not allowed for the luma component if one or more of the following cases is (are) satisfied.

[00106] (a) $M \leq TX$ and $N \leq TY$. In one example, $TX=TY=4$;

[00107] (b) $M \leq TX$ or $N \leq TY$. In one example, $TX=TY=4$;

[00108] (c) The current coding block applies the sub-block based prediction, such as affine prediction or ATMVP.

[00109] Example 2. Suppose the current chroma coding block size is $M \times N$, Bi-prediction is not allowed for the chroma components if one or more of the following cases is (are) satisfied.

[00110] (a) $M \leq TX$ and $N \leq TY$. In one example, $TX=TY=2$;

[00111] (b) $M \leq TX$ or $N \leq TY$. In one example, $TX=TY=2$;

[00112] (c) The current coding block applies the sub-block based prediction, such as affine prediction or ATMVP.

[00113] Example 3. Whether bi-prediction is allowed or not can be different for the luma component and chroma components in the same block.

[00114] Example 4. If Bi-prediction is not allowed for a coding block, the flag or codeword to represent bi-prediction is omitted and inferred to be 0.

[00115] Example 5. Suppose the current luma coding block size is $M \times N$, the splitting operation (such as QT, BT or TT) is only applied on the luma component but not on the chroma component if one or more of the following cases is (are) satisfied:

[00116] (a) $M \leq TX$ and $N \leq TY$. In one example, $TX=TY=8$;

[00117] (b) $M \leq TX$ or $N \leq TY$. In one example, $TX=TY=8$;

[00118] Example 6. If a block B is signaled to be split into X sub-CUs (For example, $X=4$ for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split, then the encoding order, the decoding order or the signaling order can be designed as:

[00119] (a) SubB[0] for Luma, SubB[1] for Luma,.... SubB[X-1] for Luma, B for Cb component, B for Cr Component, as shown in FIG. 13.

[00120] (b) Alternatively, SubB[0] for Luma, SubB[1] for Luma,.... SubB[X-1] for Luma, B for Cr component, B for Cb Component;

[00121] (c) Alternatively, SubB[0] for Luma, B for Cb component, B for Cr Component, SubB[1] for Luma,.... SubB[X-1] for Luma;

[00122] (d) Alternatively, SubB[0] for Luma, B for Cr component, B for Cb Component, SubB[1] for Luma,... SubB[X-1] for Luma;

[00123] (e) Alternatively, B for Cb component, B for Cr Component, SubB[0] for Luma, SubB[1] for Luma,... SubB[X-1] for Luma;

[00124] (f) Alternatively, B for Cr component, B for Cb Component, SubB[0] for Luma, SubB[1] for Luma,... SubB[X-1] for Luma;

[00125] (g) It is possible SubB[0], SubB[1],... SubB[X-1] may be further split for the luma component.

[00126] Example 7. In one embodiment, chroma components of the block B are reconstructed after the luma component of all the sub-blocks of the block B have been reconstructed, if a block B is signaled to be split into X sub-CUs (For example, X=4 for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00127] Example 8. In one embodiment, the prediction mode (intra-coded or inter-coded) for chroma components of the block B can be derived from the prediction modes of sub-CUs for the luma component if a block B is signaled to be split into X sub-CUs (For example, X=4 for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00128] (a) In one example, the prediction mode(e.g., intra or inter or others; intra prediction direction, etc. al) for chroma components is derived as the prediction mode for the luma component of one subCUs, such as subCU[0]which is the first sub-CU in the encoding/decoding order.

[00129] (b) In one example, the prediction mode for chroma components is derived as the prediction mode for a sample of the luma component at a predefined positionin the luma block, such as the top-left position (TL) of B, the top-right position (TR) of B, the bottom-left position (BL) of B, the bottom-right position (BR) of B and the center position (C) of B.

[00130] (c) In one example, the prediction mode for chroma components is derived as inter-coded if at least one position inside B belongs to an inter-coded CU for the luma component.

[00131] (d) In one example, the prediction mode for chroma components is derived as intra-coded if at least one position inside B belongs to an intra-coded CU for the luma component.

[00132] (e) In one example, the prediction mode for chroma components is derived as

intra-coded if the area inside B belongs to intra-coded CUs is larger than that belongs to inter-coded CUs for the luma component. Otherwise, it is derived as inter-coded.

[00133] Example 9. In one embodiment, the prediction mode for chroma components of the block B can be coded separately from the luma component if a block B is signaled to be split into X sub-CUs (For example, X=4 for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00134] (a) In one example, the coding of the prediction mode for chroma components depends on the coding of the prediction modes for the luma component.

[00135] (i) In one example, the prediction mode derived from the luma component is treated as the prediction for the prediction mode for chroma components.

[00136] (ii) Alternatively, the prediction mode derived from the luma component is treated as the coding context for the prediction mode for chroma components.

[00137] Example 10. In one embodiment, the MV for chroma components of the block B can be derived from the MVs of sub-CUs for the luma component if a block B is signaled to be split into X sub-CUs (For example, X=4 for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00138] (a) In one example, the MV for chroma components is derived as the MV for the luma component of one subCUs, such as subCU[0].

[00139] (b) In one example, the MV for chroma components is derived as the MV for the luma component at a predefined position, such as the top-left position (TL) of B, the top-right position (TR) of B, the bottom-left position (BL) of B, the bottom-right position (BR) of B and the center position (C) of B.

[00140] (c) In one example, the MV for chroma components is derived as the first found MV of the luma component at a series of predefined positions in a checking order. For example, the series of predefined positions in the checking order are { C, TL, TR, BL, BR }, then C, TL, TR, BL and BR are checked one by one, the first one that belongs to an inter-coded CU is selected and the associated MV is chosen as the MV for chroma components.

[00141] (d) In one example, the MV for chroma components is derived as the MV of sub-CU S for the luma component if the prediction mode for chroma components is derived from sub-CU S.

[00142] (e) In one example, the MV for chroma components is derived as the MV of

position P for the luma component if the prediction mode for chroma components is derived from position P.

[00143] (f) In one example, the MV for chroma components is derived as a function of the MVs for the luma component from several subCUs or at several positions. Exemplary functions are average(), minimum(), maximum(), or median().

[00144] (g) The MV for chroma components derived from MV for the luma component may be scaled before motion compensation for chroma components. For example, $MV_chroma_x = MV_luma_x \gg scaleX$, $MV_chroma_y = MV_luma_y \gg scaleY$, where $scaleX = scaleY = 1$ for the 4:2:0 format.

[00145] Example 11. In one embodiment, the motion information syntax element E (such as skip flag, merge flag, merge index, inter direction (L0, L1 or Bi), reference index, mv difference (mvd), mv candidate index, affine flag, ic flag, imv flag ect.) for chroma components of the block B can be derived from Es of sub-CUs for the luma component if a block B is signaled to be split into X sub-CUs (For example, X=4 for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00146] (a) In one example, E for chroma components is derived as E for the luma component of one subCUs, such as subCU[0].

[00147] (b) In one example, E for chroma components is derived as E for the luma component at a predefined position, such as the top-left position (TL) of B, the top-right position (TR) of B, the bottom-left position (BL) of B, the bottom-right position (BR) of B and the center position (C) of B.

[00148] (c) In one example, E for chroma components is derived as the first found E of the luma component at a series of predefined positions in a checking order. For example, the series of predefined positions in the checking order are { C, TL, TR, BL, BR }, then C, TL, TR, BL and BR are checked one by one, the first one that belongs to an inter-coded CU is selected and the associated E is chosen as the E for chroma components.

[00149] (d) In one example, E for chroma components is derived as E of sub-CU S for the luma component if the prediction mode for chroma components is derived from sub-CU S.

[00150] (e) In one example, E for chroma components is derived as E of position P for the luma component if the prediction mode for chroma components is derived from position P.

[00151] (f) In one example, E for chroma components is derived as a function of the Es

for the luma component from several subCUs or at several positions. Exemplary functions are operator “and”, operator “or”, average(), minimum(), maximum(), or median().

[00152] Example 12. In one embodiment, the MVs for chroma components of the block B can be coded separately from the luma component if a block B is signaled to be split into X sub-CUs (For example, X=4 for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00153] (a) In one example, the coding of the motion information syntax element E (such as skip flag, merge flag, merge index, inter direction (L0, L1 or Bi), reference index, mv difference (mvd), mv candidate index, affine flag, ic flag, imv flag ect.) for chroma components depends on Es for the luma component.

[00154] (i) In one example, the E derived from the luma component is treated as the prediction for the E for chroma components.

[00155] (ii) Alternatively, the E derived from the luma component is treated as the coding context to code the E for chroma components.

[00156] Example 13. In one embodiment, the intra prediction mode (IPM) (such as DC, Planar, vertical etc.) for chroma components of the block B can be derived from the intra prediction mode of sub-CUs for the luma component if a block B is signaled to be split into X sub-CUs (For example, X=4 for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00157] (a) In one example, the IPM for chroma components is derived as the IPM for the luma component of one subCUs, such as subCU[0].

[00158] (b) In one example, the IPM for chroma components is derived as the IPM for the luma component at a predefined position, such as the top-left position (TL) of B, the top-right position (TR) of B, the bottom-left position (BL) of B, the bottom-right position (BR) of B and the center position (C) of B.

[00159] (c) In one example, the IPM for chroma components is derived as the first found IPM of the luma component at a series of predefined positions in a checking order. For example, the series of predefined positions in the checking order are { C, TL, TR, BL, BR}, then C, TL, TR, BL and BR are checked one by one, the first one that belongs to an intra-coded CU is selected and the associated IPM is chosen as the IPM for chroma components.

[00160] (d) In one example, the IPM for chroma components is derived as the IPM of sub-

CU S for the luma component if the prediction mode for chroma components is derived from sub-CU S.

[00161] (e) In one example, the IPM for chroma components is derived as the IPM of position P for the luma component if the prediction mode for chroma components is derived from position P.

[00162] (f) In one example, the IPM for chroma components is derived as a function of the IPMs for the luma component from several subCUs or at several positions. Exemplary functions are average(), minimum(), maximum(), or median().

[00163] (i) Alternatively, the IPM for chroma components is derived as Planar if at least one IPM for the luma component from several subCUs or at several positions is Planar;

[00164] (ii) Alternatively, the IPM for chroma components is derived as DC if at least one IPM for the luma component from several subCUs or at several positions is DC;

[00165] Example 14. In one embodiment, the IPM for chroma components of the block B can be coded separately from the luma component if a block B is signaled to be split into X sub-CUs (For example, X=4 for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00166] (a) In one example, the coding of IPM for chroma components depends on cbfs for the luma component.

[00167] (i) In one example, the IPM derived from the luma component is treated as the prediction for the IPM for chroma components. In a further example, one or more IPMs derived from the luma component is treated as one or more DM modes for the chroma components.

[00168] (ii) Alternatively, the IPM derived from the luma component is treated as the coding context to code the IPM for chroma components.

[00169] Example 15. In one embodiment, the coded block flag (cbf) (it is 0 if no residuals are coded) for chroma components of the block B can be derived from the cbf of sub-CUs for the luma component if a block B is signaled to be split into X sub-CUs (For example, X=4 for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00170] (a) In one example, the cbf for chroma components is derived as the cbf for the luma component of one subCUs, such as subCU[0] which is the first sub-CU in the encoding/decoding order.

[00171] (b) In one example, the cbf for chroma components is derived as the cbf for a sample of the luma component at a predefined position in the luma block, such as the top-left position (TL) of B, the top-right position (TR) of B, the bottom-left position (BL) of B, the bottom-right position (BR) of B and the center position (C) of B.

[00172] (c) In one example, the cbf for chroma components is derived as the first found non-zero cbf of the luma component at a series of predefined positions in a checking order. For example, the series of predefined positions in the checking order are { C, TL, TR, BL, BR }, then C, TL, TR, BL and BR are checked one by one, the first one that not equal to zero is selected and the associated cbf is chosen as the cbf for chroma components.

[00173] (d) In one example, the cbf for chroma components is derived as the first found zero cbf of the luma component at a series of predefined positions in a checking order. For example, the series of predefined positions in the checking order are { C, TL, TR, BL, BR }, then C, TL, TR, BL and BR are checked one by one, the first one that equal to zero is selected and the associated cbf is chosen as the cbf for chroma components.

[00174] (e) In one example, the IPM for chroma components is derived as the IPM of sub-CU S for the luma component if the prediction mode for chroma components is derived from sub-CU S.

[00175] (f) In one example, the IPM for chroma components is derived as the IPM of position P for the luma component if the prediction mode for chroma components is derived from position P.

[00176] (g) In one example, the cbf for chroma components is derived as a function of the cbfs for the luma component from several subCUs or at several positions. Exemplary functions are operator “and”, operator “or”, minimum(), and maximum().

[00177] (h) In one example, only cbfs from sub-CUs or positions for the luma component coded by the intra mode is under consideration if the chroma component is coded by the intra mode.

[00178] (i) In one example, only cbfs from sub-CUs or positions for the luma component coded by the inter mode is under consideration if the chroma component is coded by the inter mode.

[00179] Example 16. In one embodiment, the cbf for chroma components of the block B can be coded separately from the luma component if a block B is signaled to be split into X sub-CUs

(For example, $X=4$ for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00180] (a) In one example, the coding of cbf for chroma components depends on cbfs for the luma component.

[00181] (i) In one example, the cbf derived from the luma component is treated as the prediction for the cbf for chroma components.

[00182] (ii) Alternatively, the cbf derived from the luma component is treated as the coding context to code the cbf for chroma components.

[00183] Example 17. The in-loop filtering should be conducted differently for luma and chroma components. In one example, in-loop filtering is conducted at boundaries of CUs inside block B for the luma component, but not conducted for chroma components, if a block B is signaled to be split into X sub-CUs (For example, $X=4$ for QT, 3 for TT and 2 for BT), but it is inferred that the chroma components in block B cannot be split.

[00184] Example 18. Whether and how to apply the restrictions can be predefined, or they can be transmitted from the encoder to the decoder. For example, they can be signaled in Video Parameter Set (VPS), Sequence Parameter Set (SPS), Picture Parameter Set (PPS), Slice header, Coding Tree Unit (CTU) or Coding Unit (CU).

[00185] Example 19. Some constraint may be applied to blocks as small as 2×2 (which may be denoted as “tiny blocks”).

[00186] (a) Tiny blocks can be defined as 2×2 blocks or $2 \times N$ blocks or $N \times 2$ blocks or both $2 \times N$ blocks and $N \times 2$ blocks. In some embodiments, a chroma block may be a tiny block, but its corresponding luma block may not be a tiny block; in this case the chroma block and its corresponding luma block may be processed in different ways.

[00187] (b) In one embodiment, intra-prediction is conducted on a tiny block in a different way from a normal block (not a tiny block).

[00188] (i) In one example, no filtering is applied on the neighboring samples of a tiny block before conducting the intra-prediction.

[00189] (ii) In one example, no filtering is applied on the prediction samples of a tiny block after conducting the intra-prediction.

[00190] (iii) In one example, some intra-prediction mode is invalid for a tiny block, such as LM mode (in which the chroma block prediction is based on reconstructed luma block(s))

using the Linear Model).

[00191] (iv) In one example, the intra-prediction mode of a tiny block is not signaled but inferred to be the predefined intra-prediction mode. A tiny block always uses one predefined intra-prediction mode, such as DC, Planar, Vertical or Horizontal mode.

[00192] (v) Alternatively, a tiny block always uses one predefined intra-prediction mode, such as DC, Planar, Vertical or Horizontal mode.

[00193] (1) The intra-prediction mode of a tiny block signaled in a conformable bitstream must be the predefined intra-prediction mode.

[00194] (2) The intra-prediction mode of a tiny block signaled in a conformable bitstream is parsed but is ignored in the decoding process.

[00195] (3) Alternatively, the predefined intra-prediction mode can be a simplified DC mode. For example, all prediction samples are equal to $(1 \ll \text{bitDepth})/2$ (128 for 8-bit video sequences, 512 for 10-bit video sequences).

[00196] (vi) Alternatively, a subset of all intra-prediction modes can be used by a tiny block. A tiny block chooses one of the mode in the subset. The encoder can signal the mode to the decoder or the decoder can infer it.

[00197] (1) In one example, the subset may be fixed to be the same for all blocks within one slice/tile/CTU rows/group of CTUs.

[00198] (2) Alternatively, the subset may be adaptively changed. For example, the subset may depend on block sizes or block shapes.

[00199] (3) In one example, the subset includes DC and LM mode.

[00200] (4) In one example, the subset includes Horizontal and Vertical mode.

[00201] (a) In one example, the intra-prediction mode depends on the dimensions of the block. For example, if the block width is larger or equal to the block height, Vertical mode is used; otherwise, Horizontal mode is used.

[00202] (5) In one example, the subset includes LM mode, Horizontal and Vertical mode.

[00203] (a) In one example, the encoder signals whether LM mode is applied. When LM mode is not applied, if the block width is larger or equal to the block height, Vertical mode is used; otherwise, Horizontal mode is used.

[00204] (c) In one embodiment, transform and invert-transform in a tiny block is conducted in a different way from a normal block (not a tiny block).

[00205] (i) In one example, no transform and/or inverse-transform may be applied in a tiny block.

[00206] (ii) In one example, whether to and how to conduct transform/invert-transform may depend on the block dimensions.

[00207] (1) For example, both row transform/invert-transform (the samples before and after transform/invert-transform are in a row) and column transform/ invert-transform (the samples before and after transform/invert-transform are in a column) are not applied on a tiny block with size 2×2 ;

[00208] (2) For example, row transform/invert-transform is applied but column transform/ invert-transform is not applied on a tiny block with size $W \times 2$ where W is the width of the tiny block;

[00209] (3) For example, row transform/invert-transform is not applied but column transform/ invert-transform is applied on a tiny block with size $2 \times H$ where H is the height of the tiny block;

[00210] (iii) In one example, transform skip mode is applied in a tiny block.

[00211] (iv) In one example, transform skip flag of a tiny block is not signaled but inferred to be 1. A tiny block always uses transform skip mode.

[00212] (v) Alternatively, a tiny block always uses the transform skip mode.

[00213] (1) The transform skip flag of a tiny block signaled in a conformable bitstream must be 1.

[00214] (2) The transform skip flag of a tiny block signaled in a conformable bitstream is parsed but is ignored in the decoding process.

[00215] (vi) Alternatively, how to signal and/or how to infer transform skip flag may depend on the dimensions of the block. Suppose the width and height of the current block are denoted as W and H , respectively.

[00216] (1) In one example:

[00217] (a) If $W == 2$ or $H == 2$ (or equally $W \leq 2$ or $H \leq 2$), transform skip flag is not signaled and inferred to be 1.

[00218] (b) Otherwise, if $W \times H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll$

1)) where transformSkipLog2MaxSize is signaled from encoder to decoder in SPS, transform skip flag is signaled.

[00219] (c) Otherwise (neither (a) nor (b) satisfies), transform skip flag is not signaled and inferred to be 0.

[00220] (2) In another example:

[00221] (a) If $W == 2$ or $H == 2$ (or equally $W \leq 2$ or $H \leq 2$), transform skip flag is signaled.

[00222] (b) Otherwise, if $W \times H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$ where transformSkipLog2MaxSize is signaled from encoder to decoder in SPS, transform skip flag is signaled.

[00223] (c) Otherwise (neither (a) nor (b) satisfies), transform skip flag is not signaled and inferred to be 0. In one example, residues in a tiny block are always equal to 0.

[00224] (3) In still another example:

[00225] (a) If $W \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$ or $H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$ where transformSkipLog2MaxSize is signaled from encoder to decoder in SPS, transform skip flag is signaled.

[00226] (b) Otherwise, transform skip flag is not signaled and inferred to be 0. In one example, residues in a tiny block are always equal to 0.

[00227] (4) In still another example:

[00228] (a) If $W \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$ and $H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$ where transformSkipLog2MaxSize is signaled from encoder to decoder in SPS, transform skip flag is signaled.

[00229] (b) Otherwise, transform skip flag is not signaled and inferred to be 0.

[00230] (d) In one example, residues in a tiny block are always equal to 0.

[00231] (i) In one example, the coded block flag (cbf) of a tiny block is not signaled but inferred to be 0. A tiny block always has no residues.

[00232] (ii) Alternatively, a tiny block always has no residues.

[00233] (1) cbf of a tiny block signaled in a conformable bitstream is 0.

[00234] (2) cbf of a tiny block signaled in a conformable bitstream is parsed but is ignored in the decoding process.

[00235] (e) In one embodiment, only a DC value of residues is signaled if residues are not all

0 in a tiny block.

[00236] (i) For each sample in the tiny block, $S(x,y)=P(x,y)+D$, where $S(x,y)$ represent the reconstruction at position (x,y) , $P(x,y)$ represent predictive values at position (x,y) , D is the DC value of residues signaled for the tiny block. D is the same for all samples in the tiny block.

[00237] (ii) In one example, D is quantized before signaled at encoder and dequantized before constructing the reconstruction at decoder.

[00238] (1) For example, the quantization step is decided by the quantization parameter (QP) as in normal quantization after the transform.

[00239] (f) In one embodiment, motion compensation in a tiny block is conducted in a different way from a normal block (not a tiny block).

[00240] (i) No interpolation is done for a tiny block. The MV of the tiny block is rounded to a nearest (or adjacent) integer pixel.

[00241] (ii) Bilinear interpolation is used for a tiny block.

[00242] (iii) Bi-prediction cannot be used by a tiny block. If a chroma block is a tiny block, but its corresponding luma block is not a tiny block and it applies bi-prediction, then only one MV is used by the chroma block. This MV may be get from L0, or from L1.

[00243] (iv) Overlapped Block Motion Compensation (OBMC) cannot be used for a tiny block.

[00244] The examples described above may be incorporated in the context of the methods described below, e.g., method 1400, which may be implemented at a video decoder and/or video encoder.

[00245] FIG. 14 shows a flowchart of an exemplary method for video coding, which may be implemented in a video encoder. The method 1400 includes, at step 1410, receiving a bitstream representation of a block of video data comprising a luma component and a chroma component.

[00246] The method 1400 includes, at step 1420, processing the bitstream representation using a first prediction mode to generate the block of video data, where the first prediction mode is based on a property of the luma component or the chroma component. In some embodiments, the property includes dimensions of the luma component or the chroma component.

[00247] In some embodiments, the first prediction mode is not a bi-prediction mode, and the first prediction mode for the luma component is different from a second prediction mode for the

chroma component. In other embodiments, the first prediction mode is not a bi-prediction mode, and the first prediction mode for the luma component is identical to a second prediction mode for the chroma component.

[00248] The method 1400 may further include performing a splitting operation on the luma component or the chroma component. In some embodiments, a size of the luma component is $M \times N$, where $M \leq TX$ and/or $N \leq TY$ with TX and TY being integer thresholds, and where the splitting operation is performed on the luma component and not on the chroma component.

[00249] The method 1400 may further include performing, based on the property, a splitting operation on the block of video data to generate sub-blocks. In some embodiments, the chroma component cannot be split, and the splitting operation is performed on the luma component to generate luma components for each of the sub-blocks.

[00250] In an example, the chroma component is reconstructed after the luma components of the sub-blocks have been reconstructed.

[00251] In another example, the a characteristic of the chroma component is derived from the same characteristic of the luma components of the sub-blocks. In other words, characteristics from one of the luma sub-blocks can be copied over to the chroma block. The characteristic may be, but is not limited to, a prediction mode, motion vectors, a motion information syntax element, an intra prediction mode (IPM), or a coded block flag. In some embodiments, the motion information syntax element may be a skip flag, a merge flag, a merge index, an inter direction, a reference index, a motion vector candidate index, an affine flag, an illumination compensation flag or an integer motion vector flag.

[00252] In some embodiments, the property or an indication of the property, or more generally, a determination of whether or not to perform one of the operations elucidated in the examples described above, is signaled in a Video Parameter Set (VPS), a Sequence Parameter Set (SPS), a Picture Parameter Set (PPS), a slice header, a coding tree unit (CTU) or a coding unit (CU).

[00253] 4. Example implementations of the disclosed technology

[00254] FIG. 15 is a block diagram of a video processing apparatus 1500. The apparatus 1500 may be used to implement one or more of the methods described herein. The apparatus 1500 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 1500 may include one or more processors 1502, one or more memories 1504 and

video processing hardware 1506. The processor(s) 1502 may be configured to implement one or more methods (including, but not limited to, method 1400) described in the present document. The memory (memories) 1504 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 1506 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00255] In some embodiments, a video decoder apparatus may implement a method of using zero-units as described herein is used for video decoding. The various features of the method may be similar to the above-described method 1400.

[00256] In some embodiments, the video decoding methods may be implemented using a decoding apparatus that is implemented on a hardware platform as described with respect to FIG.15.

[00257] FIG. 16 shows a flowchart of an exemplary method for video processing, which may be implemented in a video encoder/decoder. The method 1600 includes, at step 1610, receiving a bitstream representation of video data including a video block wherein the video block comprises samples in a first component and in a second component.

[00258] The method 1600 includes, at step 1620, deriving a first coding information for the first component from second coding information of sub-blocks for the second component in case that the video block for the second component is split into the sub-blocks.

[00259] The method 1600 includes, at step 1630, performing a conversion between the video block and the bitstream representation of the video block based on the first coding information.

[00260] FIG. 17 shows a flowchart of an exemplary method for video decoding, which may be implemented at a video decoding side.

[00261] As shown in FIG. 17, the method 1700 includes, at step 1710, receiving a bitstream representation of video data including a video block wherein the video block comprises samples in a first component, the samples in the first component having a dimension of $M \times N$.

[00262] The method 1700 further includes, at step 1720, determining, based on one or more of specific conditions is satisfied, a first prediction mode for decoding the first component of the block is not a bi-prediction.

[00263] The method 1700 further includes, at step 1730, decoding the first component by using the first prediction mode.

[00264] FIG.18 shows a flowchart of another example method for video decoding in

accordance with the presently disclosed technology.

[00265] As shown in FIG. 18, the method 1800 includes, at step 1810, receiving a bitstream representation of video data including a video block wherein the video block comprises samples in a first component and samples in a second component.

[00266] The method 1800 includes, at step 1820, determining a first prediction mode for decoding the first component of the block and determining a second prediction mode for decoding the second component of the block.

[00267] The method 1800 includes, at step 1830, decoding the first and second component by using the first and the second prediction mode respectively.

[00268] FIG.19 shows a flowchart of another example method for video decoding in accordance with the presently disclosed technology.

[00269] As shown in FIG. 19, the method 1900 includes, at step 1910, receiving a bitstream representation of video data including a block wherein the block comprises samples associated with a first component and second components, wherein samples associated with the first component of the block have a dimension $M \times N$.

[00270] The method 1900 includes, at step 1920, decoding the first component and the second components of the block; wherein decoding the first component of the block comprises, based on the dimension, decoding a plurality of sub-blocks for the first component of the block, and the plurality of the sub-blocks are generated by performing a splitting operation only on the samples associated with the first component of the block and not on the samples associated with the second components of the block.

[00271] FIG.20 shows a flowchart of another example method for video decoding in accordance with the presently disclosed technology.

[00272] As shown in FIG. 20, the method 2000 includes, at step 2010, determining a block is a tiny block, wherein at least one of a height and a width of the block has 2 samples. For example, the determining includes: receiving a bitstream representation of a block of video data comprising a first component and a second component, and, designating the first component of the block as a tiny block if the first component of block has dimensions of 2×2 , $2 \times N$ or $N \times 2$, and designating the second component of the block with dimensions different from the first component as a normal block.

[00273] The method 2000 includes, at step 2020, performing, based on the determining, a

conversion between the block and a coded representation of the block in a way different from a normal block.

[00274] Various embodiments and techniques disclosed in the present document can be described in the following listing of examples.

[00275] 1. A method for video bitstream processing, comprising: determining a block is a tiny block, wherein at least one of a height and a width of the block has 2 samples; and performing, based on the determining, a conversion between the block and a coded representation of the block in a way different from a normal block.

[00276] 2. The method of example 1, wherein the block comprises a luma component and/or chroma components.

[00277] 3. The method of example 1, wherein converting a chroma block defined as the tiny block and its corresponding luma component defined as the normal block with different ways.

[00278] 4. The method of example 1 or 2, wherein the conversion comprises conducting an intra-prediction on the tiny block in a different way from the normal block.

[00279] 5. The method of example 4, wherein no filtering is applied on neighbouring samples of the tiny block before conducting the intra-prediction.

[00280] 6. The method of example 4, wherein no filtering is applied on prediction samples of the tiny block after conducting the intra-prediction.

[00281] 7. The method of example 4, wherein at least one mode of the intra-prediction is invalid for the tiny block, and the at least one mode of intra-prediction comprises a Linear Model (LM) mode.

[00282] 8. The method of example 4, wherein the intra-prediction of the tiny block is inferred to be a pre-defined prediction mode without being signaled.

[00283] 9. The method of example 4, wherein the intra-prediction of the tiny block is a pre-defined prediction mode.

[00284] 10. The method of example 9, wherein the pre-defined prediction mode of the tiny block is signaled in a conformable bitstream.

[00285] 11. The method of example 10, wherein the pre-defined prediction mode of the tiny block signaled in the conformable bitstream is parsed and is ignored in the decoding process.

[00286] 12. The method of example 8 or 9, wherein the pre-defined prediction mode of the tiny block is a simplified DC prediction mode, and all prediction samples of the tiny block are

equal to $(1 \ll \text{bitDepth})/2$.

[00287] 13. The method of example 8 or 9, the pre-defined prediction mode is one of a DC mode, a Planar mode, a Vertical mode, a Horizontal mode or an LM mode.

[00288] 14. The method of example 4, wherein a subset of all modes of the intra-prediction can be applied on the tiny block, and an intra-prediction mode selected from the subset for being applied on the tiny block is signaled or inferred.

[00289] 15. The method of example 14, wherein the subset is fixed to be the same for all blocks within one slice, tile, coding tree unit(CTU) rows or a group of CTUs.

[00290] 16. The method of example 14, wherein the subset is adaptively changed based on a size or shape of the tiny block.

[00291] 17. The method of any one of examples 14-16, wherein the subset comprises a DC mode and an LM mode.

[00292] 18. The method of any one of examples 14-16, wherein the subset comprises a Horizontal and a Vertical mode.

[00293] 19. The method of example 18, wherein the intra-prediction mode to be selected from the subset depends on the dimension of the tiny block.

[00294] 20. The method of example 19, wherein if a width of the tiny block is larger or equal to a height of the tiny block, the vertical mode is selected for being applied on the tiny block; otherwise, the horizontal mode is selected for being applied on the tiny block.

[00295] 21. The method of any one of examples 14-16, wherein the subset comprises an LM mode, a Horizontal mode and a Vertical mode.

[00296] 22. The method of example 21, wherein it is signaled whether the LM mode is applied; and when the LM mode is not applied, the vertical mode is applied if a width of the tiny block is larger or equal to a height of the tiny block, otherwise, the horizontal mode is applied on the tiny block.

[00297] 23. The method of example 1, wherein no transform and invert-transform is applied on the tiny block.

[00298] 24. The method of example 1, wherein whether to and how to apply transform and/or invert-transform on the tiny block depends on dimension of the tiny block.

[00299] 25. The method of example 24, wherein none of row transform, row invert-transform, column transform, and column invert-transform is applied on the tiny block if the size of the tiny

block is 2×2 .

[00300] 26. The method of example 24, wherein a row transform and a row invert-transform are applied on the tiny block if the size of the tiny block is $W \times 2$, but a column transform and a column invert-transform are not applied, wherein W representing a width of the tiny block and is not equal to 2.

[00301] 27. The method of example 24, wherein a column transform and a column invert-transform are applied on the tiny block if the size of the tiny block is $2 \times H$, but a row transform and a row invert-transform are not applied, wherein H representing the height of the tiny block and is not equal to 2.

[00302] 28. The method of example 1, wherein a transform skip mode is applied on the tiny block.

[00303] 29. The method of example 28, wherein a transform skip flag of the tiny block is inferred to be 1 without being signaled.

[00304] 30. The method of example 28, wherein a transform skip flag of the tiny block is signaled in a conformable bitstream and is equal to 1.

[00305] 31. The method of example 30, wherein the transform skip flag of the tiny block signaled in the conformable bitstream is parsed and is ignored in the decoding processing.

[00306] 32. The method of example 28, wherein how to signal or how to infer the transform skip flag depends on dimension of the tiny block, and the size of the tiny block is $W \times H$, W and H representing a width and height of the tiny block respectively.

[00307] 33. The method of example 32, wherein if $W \leq 2$ or $H \leq 2$, the transform skip flag is not signaled and inferred to be 1; and if $W \times H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$, the transform skip flag is signaled, the $\text{transformSkipLog2MaxSize}$ being signaled in SPS; otherwise, the transform skip flag is not signaled and inferred to be 0.

[00308] 34. The method of example 32, wherein if $W \leq 2$ or $H \leq 2$, the transform skip flag is signaled; and if $W \times H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$, the transform skip flag is signaled, $\text{transformSkipLog2MaxSize}$ being signaled in SPS; otherwise, the transform skip flag is not signaled and inferred to be 0.

[00309] 35. The method of example 32, wherein if $W \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$ and/or $H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$, the transform skip flag is signaled, $\text{transformSkipLog2MaxSize}$ being signaled in SPS; otherwise, the transform skip flag

is not signaled and inferred to be 0.

[00310] 36. The method of example 1, wherein residues in the tiny block are equal to 0.

[00311] 37. The method of example 36, wherein a coded block flag(cbf) of the tiny block is not signaled and inferred to be 0; or the cbf of the tiny block is signaled in a conformable bitstream and equal to be 0.

[00312] 38. The method of example 36, wherein the cbf of the tiny block signaled in the conformable bitstream is parsed and is ignored in the decoding processing.

[00313] 39. The method of example 1, wherein only a DC value of residues of the tiny block is signaled if residues are not all 0 in the tiny block .

[00314] 40. The method of example 39, wherein for each sample in the tiny block, $S(x,y)=P(x,y)+D$, where $S(x,y)$ represent reconstruction values at position (x,y) , $P(x,y)$ represent predictive values at position (x,y) , D represents the DC value of residues signaled for the tiny block, and D is the same for all samples in the tiny block.

[00315] 41. The method of example 40, wherein D is quantized before signaled at an encoding side and dequantized before constructing the reconstruction value at a decoding side.

[00316] 42. The method of example 41, wherein a quantization step is decided by a quantization parameter(QP) as in a normal quantization after the transform.

[00317] 43. The method of example 1, wherein the conversion further comprises a motion compensation.

[00318] 44. The method of example 43, wherein no interpolation is applied on the tiny block, and the motion vector(MV) of the tiny block is rounded to a nearing integer pixel.

[00319] 45. The method of example 43, wherein a bilinear interpolation is applied on the tiny block.

[00320] 46. The method of example 43, wherein bi-prediction is not applied on the tiny block.

[00321] 47. The method of example 3, wherein a chroma block defined as a tiny block using only one motion vector, wherein its corresponding luma block is not a tiny block.

[00322] 48. The method of example 47, wherein the MV is obtained from L0 or L1.

[00323] 49. The method of example 1, wherein an overlapped block motion compensation (OBMC) is unallowed to be used for the tiny block.

[00324] 50. A video encoding apparatus comprising a processor configured to implement a method recited in any one of examples 1 to 49.

[00325] 51. A video decoding apparatus comprising a processor configured to implement a method recited in any one of examples 1 to 49.

[00326] 52. A video decoding apparatus comprising a processor configured to implement a method recited in any one of claims 1 to 49.

[00327] 53. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of claims 1-49.

[00328] 54. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of claims 1-49.

[00329] From the foregoing, it will be appreciated that specific embodiments of the presently disclosed technology have been described herein for purposes of illustration, but that various modifications may be made without deviating from the scope of the invention. Accordingly, the presently disclosed technology is not limited except as by the appended claims.

[00330] Implementations of the subject matter and the functional operations described in this patent document can be implemented in various systems, digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Implementations of the subject matter described in this specification can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a tangible and non-transitory 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 unit” or “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.

[00331] 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.

[00332] The processes and logic flows described in this specification 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).

[00333] 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 nonvolatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00334] It is intended that the specification, together with the drawings, be considered exemplary only, where exemplary means an example. As used herein, the singular forms “a”,

“an” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. Additionally, the use of “or” is intended to include “and/or”, unless the context clearly indicates otherwise.

[00335] While this patent document contains many specifics, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. 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.

[00336] 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.

[00337] 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.

CLAIMS

What is claimed is:

1. A method for video bitstream processing, comprising:
determining a block is a tiny block, wherein at least one of a height and a width of the block has 2 samples; and
performing, based on the determining, a conversion between the block and a coded representation of the block in a way different from a normal block.
2. The method of claim 1, wherein the block comprises a luma component and/or chroma components.
3. The method of claim 1, wherein converting a chroma block defined as the tiny block and its corresponding luma component defined as the normal block with different ways.
4. The method of claim 1 or 2, wherein the conversion comprises conducting an intra-prediction on the tiny block in a different way from the normal block.
5. The method of claim 4, wherein no filtering is applied on neighbouring samples of the tiny block before conducting the intra-prediction.
6. The method of claim 4, wherein no filtering is applied on prediction samples of the tiny block after conducting the intra-prediction.
7. The method of claim 4, wherein at least one mode of the intra-prediction is invalid for the tiny block, and the at least one mode of intra-prediction comprises a Linear Model (LM) mode.
8. The method of claim 4, wherein the intra-prediction of the tiny block is inferred to be a pre-defined prediction mode without being signaled.

9. The method of claim 4, wherein the intra-prediction of the tiny block is a pre-defined prediction mode.
10. The method of claim 9, wherein the pre-defined prediction mode of the tiny block is signaled in a conformable bitstream.
11. The method of claim 10, wherein the pre-defined prediction mode of the tiny block signaled in the conformable bitstream is parsed and is ignored in the decoding process.
12. The method of claim 8 or 9, wherein the pre-defined prediction mode of the tiny block is a simplified DC prediction mode, and all prediction samples of the tiny block are equal to $(1 < \text{bitDepth})/2$.
13. The method of claim 8 or 9, the pre-defined prediction mode is one of a DC mode, a Planar mode, a Vertical mode, a Horizontal mode or a LM mode.
14. The method of claim 4, wherein a subset of all modes of the intra-prediction is applied on the tiny block, and an intra-prediction mode selected from the subset for being applied on the tiny block is signaled or inferred.
15. The method of claim 14, wherein the subset is fixed to be the same for all blocks within one slice, tile, coding tree unit(CTU) rows or a group of CTUs.
16. The method of claim 14, wherein the subset is adaptively changed based on a size or shape of the tiny block.
17. The method of any one of claims 14-16, wherein the subset comprises a DC mode and a LM mode.
18. The method of any one of claims 14-16, wherein the subset comprises a Horizontal and a Vertical mode.

19. The method of claim 18, wherein the intra-prediction mode to be selected from the subset depends on the dimension of the tiny block.
20. The method of claim 19, wherein if a width of the tiny block is larger or equal to a height of the tiny block, the vertical mode is selected for being applied on the tiny block; otherwise, the horizontal mode is selected for being applied on the tiny block.
21. The method of any one of claims 14-16, wherein the subset comprises an LM mode, a Horizontal mode and a Vertical mode.
22. The method of claim 21, wherein it is signaled whether the LM mode is applied; and when the LM mode is not applied, the vertical mode is applied if a width of the tiny block is larger or equal to a height of the tiny block, otherwise, the horizontal mode is applied on the tiny block.
23. The method of claim 1, wherein no transform and invert-transform is applied on the tiny block.
24. The method of claim 1, wherein whether to and how to apply transform and/or invert-transform on the tiny block depends on dimension of the tiny block.
25. The method of claim 24, wherein none of row transform, row invert-transform, column transform, and column invert-transform is applied on the tiny block if the size of the tiny block is 2×2 .
26. The method of claim 24, wherein a row transform and a row invert-transform are applied on the tiny block if the size of the tiny block is $W \times 2$, but a column transform and a column invert-transform are not applied, wherein W representing a width of the tiny block and is not equal to 2.

27. The method of claim 24, wherein a column transform and a column invert-transform are applied on the tiny block if the size of the tiny block is $2 \times H$, but a row transform and a row invert-transform are not applied, wherein H representing the height of the tiny block and is not equal to 2.
28. The method of claim 1, wherein a transform skip mode is applied on the tiny block.
29. The method of claim 28, wherein a transform skip flag of the tiny block is inferred to be 1 without being signaled.
30. The method of claim 28, wherein a transform skip flag of the tiny block is signaled in a conformable bitstream and is equal to 1.
31. The method of claim 30, wherein the transform skip flag of the tiny block signaled in the conformable bitstream is parsed and is ignored in the decoding processing.
32. The method of claim 28, wherein how to signal or how to infer the transform skip flag depends on dimension of the tiny block, and the size of the tiny block is $W \times H$, W and H representing a width and height of the tiny block respectively.
33. The method of claim 32, wherein
- if $W \leq 2$ or $H \leq 2$, the transform skip flag is not signaled and inferred to be 1; and
 - if $W \times H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$, the transform skip flag is signaled, the $\text{transformSkipLog2MaxSize}$ being signaled in SPS;
 - otherwise, the transform skip flag is not signaled and inferred to be 0.
34. The method of claim 32, wherein
- if $W \leq 2$ or $H \leq 2$, the transform skip flag is signaled; and

if $W \times H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$, the transform skip flag is signaled, transformSkipLog2MaxSize being signaled in SPS;

otherwise, the transform skip flag is not signaled and inferred to be 0.

35. The method of claim 32, wherein

if $W \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$ and/or $H \leq (1 \ll (\text{transformSkipLog2MaxSize} \ll 1))$, the transform skip flag is signaled, transformSkipLog2MaxSize being signaled in SPS;
otherwise, the transform skip flag is not signaled and inferred to be 0.

36. The method of claim 1, wherein residues in the tiny block are equal to 0.

37. The method of claim 36, wherein

a coded block flag (cbf) of the tiny block is not signaled and inferred to be 0; or
the cbf of the tiny block is signaled in a conformable bitstream and equal to be 0.

38. The method of claim 36, wherein the cbf of the tiny block signaled in the conformable bitstream is parsed and is ignored in the decoding processing.

39. The method of claim 1, wherein only a DC value of residues of the tiny block is signaled if residues are not all 0 in the tiny block.

40. The method of claim 39, wherein for each sample in the tiny block, $S(x,y) = P(x,y) + D$, where $S(x,y)$ represent reconstruction values at position (x,y) , $P(x,y)$ represent predictive values at position (x,y) , D represents the DC value of residues signaled for the tiny block, and D is the same for all samples in the tiny block.

41. The method of claim 40, wherein D is quantized before signaled at an encoding side and dequantized before constructing the reconstruction value at a decoding side.

42. The method of claim 41, wherein a quantization step is decided by a quantization parameter(QP) as in a normal quantization after the transform.
43. The method of claim 1, wherein the conversion further comprises a motion compensation.
44. The method of claim 43, wherein no interpolation is applied on the tiny block, and the motion vector(MV) of the tiny block is rounded to a nearing integer pixel.
45. The method of claim 43, wherein a bilinear interpolation is applied on the tiny block.
46. The method of claim 43, wherein bi-prediction is not applied on the tiny block.
47. The method of claim 3, wherein a chroma block defined as a tiny block using only one motion vector, wherein its corresponding luma block is not a tiny block.
48. The method of claim 47, wherein the MV is obtained from L0 or L1.
49. The method of claim 1, wherein an overlapped block motion compensation (OBMC) is unallowed to be used for the tiny block.
50. A video encoding apparatus comprising a processor configured to implement a method recited in any one of claims 1 to 49.
51. A video decoding apparatus comprising a processor configured to implement a method recited in any one of claims 1 to 49.
52. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of claims 1-49.

53. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of claims 1-49.

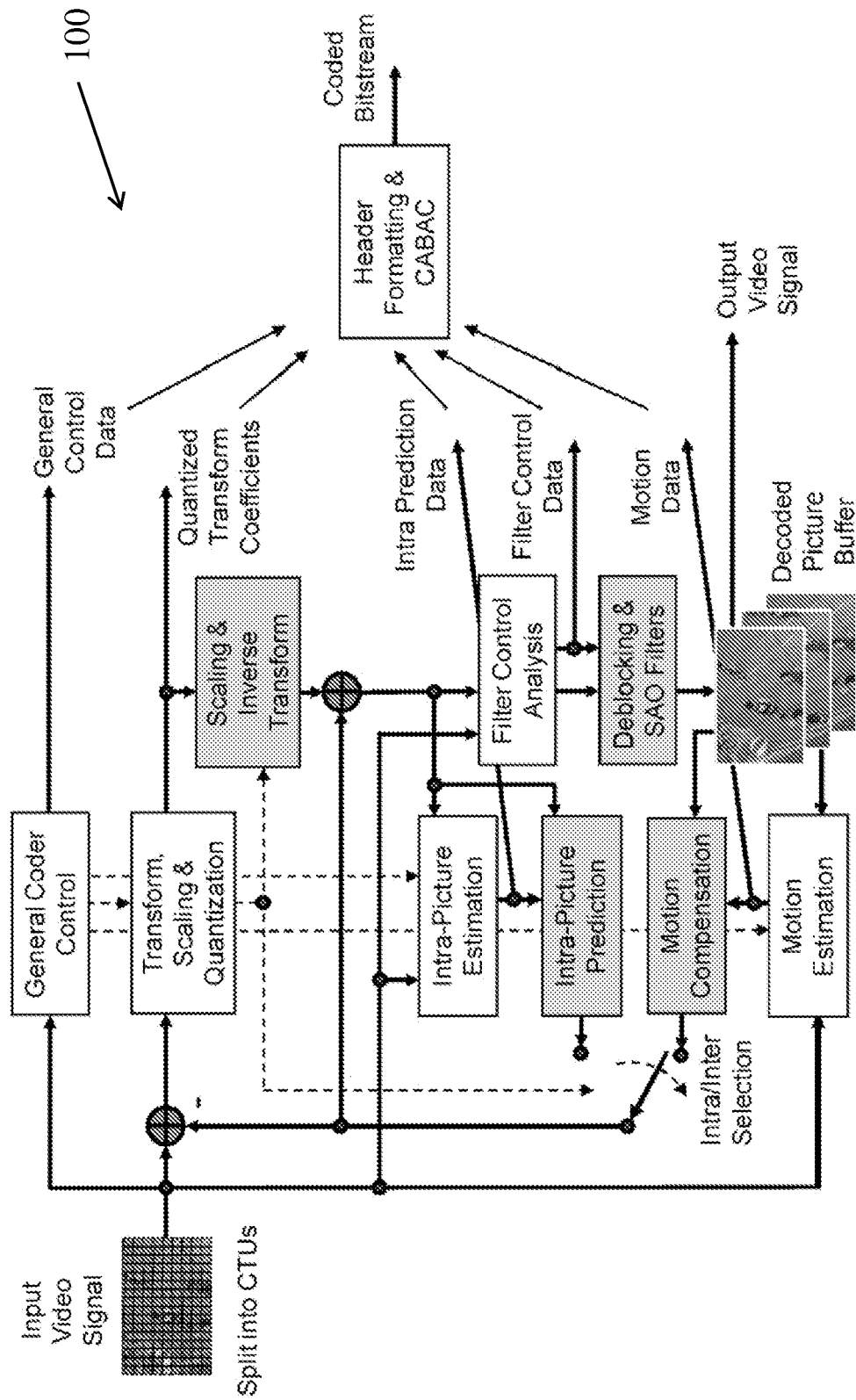


FIG. 1

200

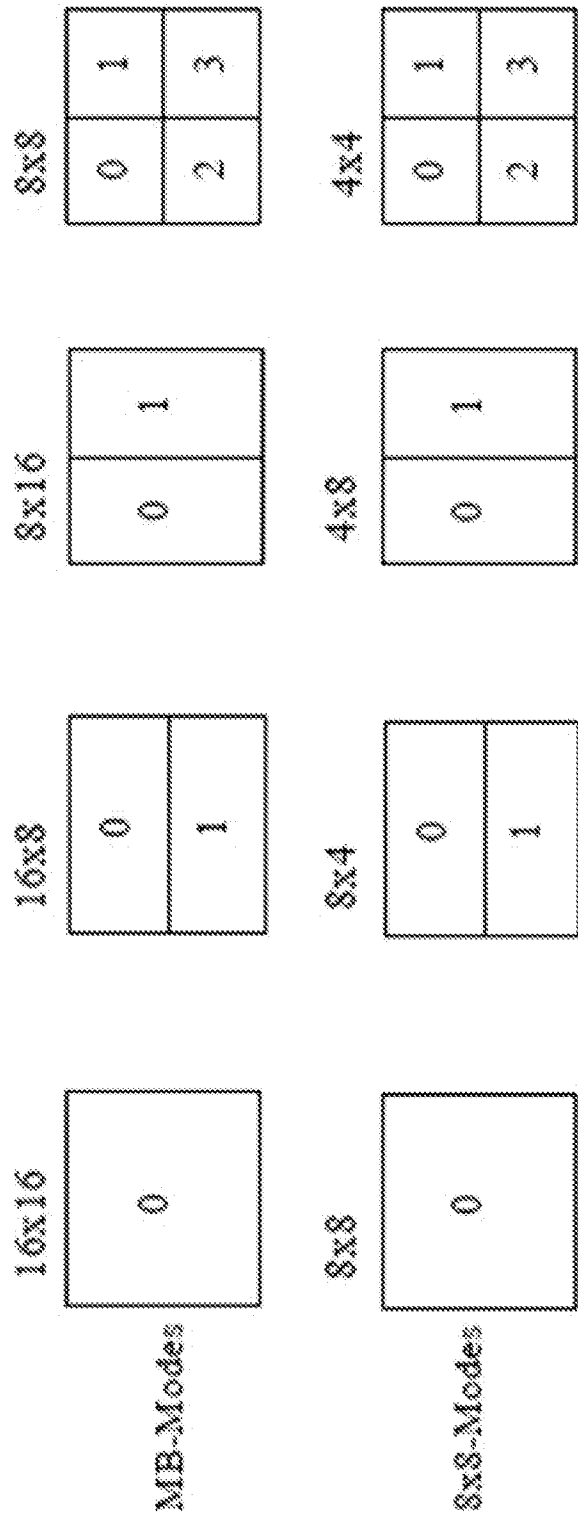


FIG. 2

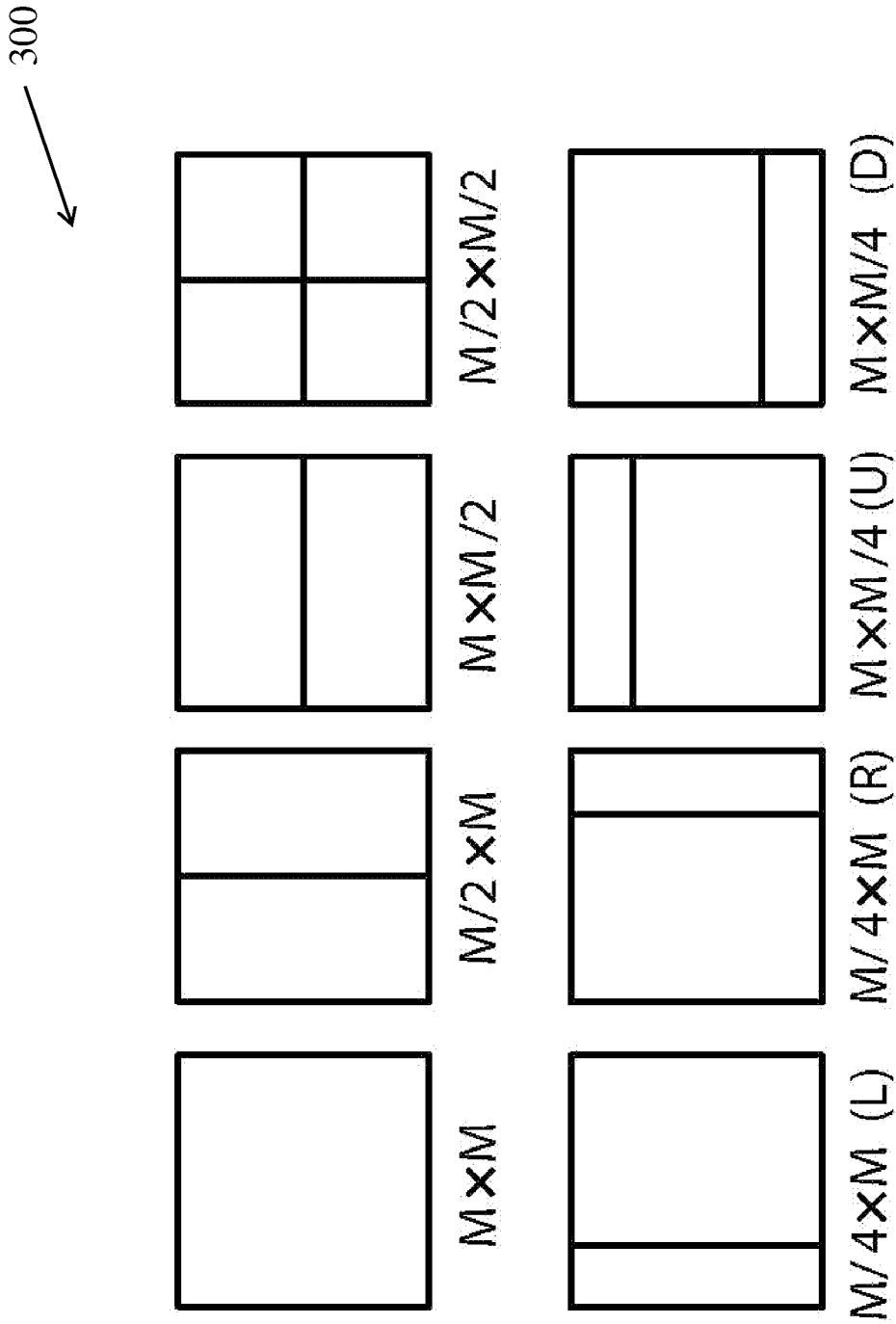


FIG. 3

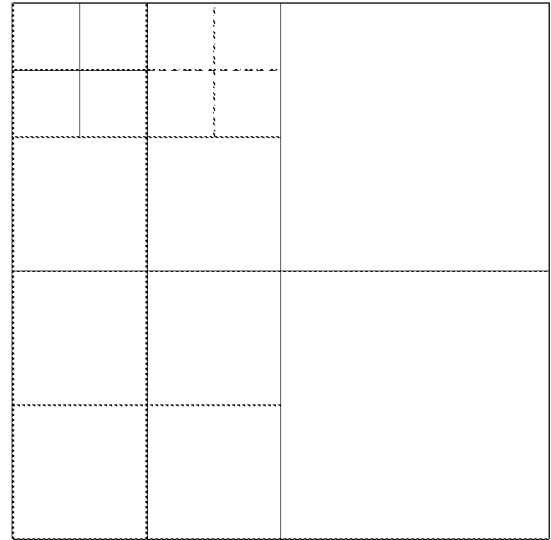


FIG. 4A

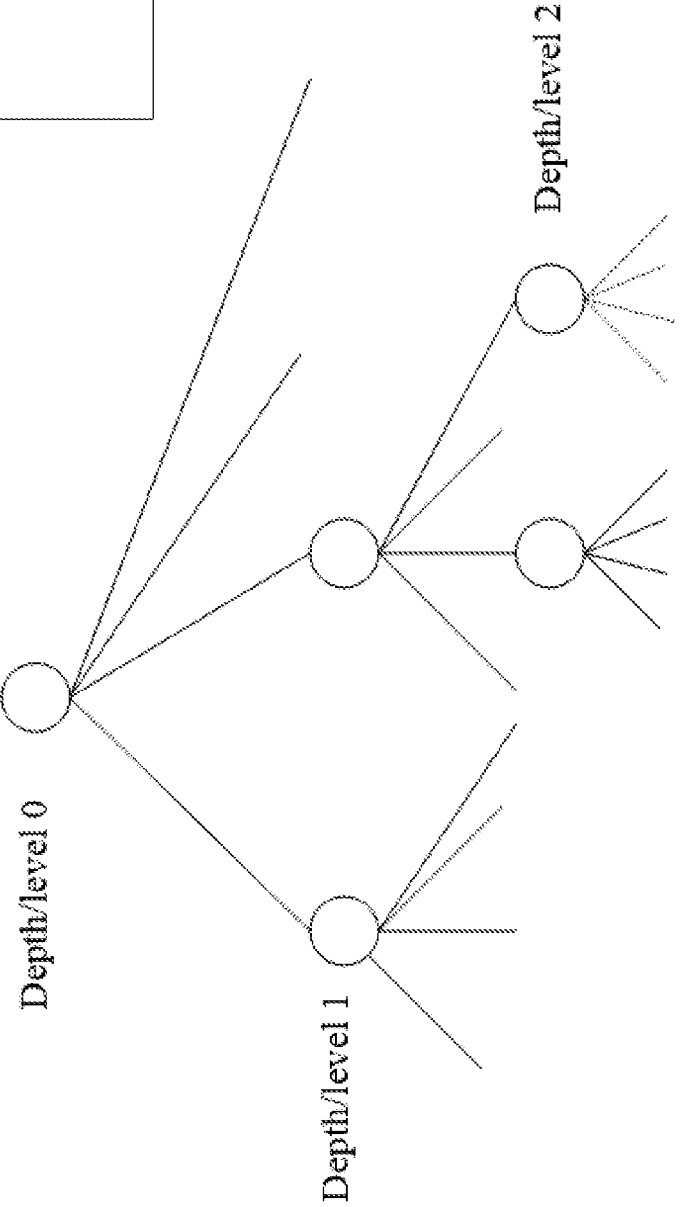


FIG. 4B

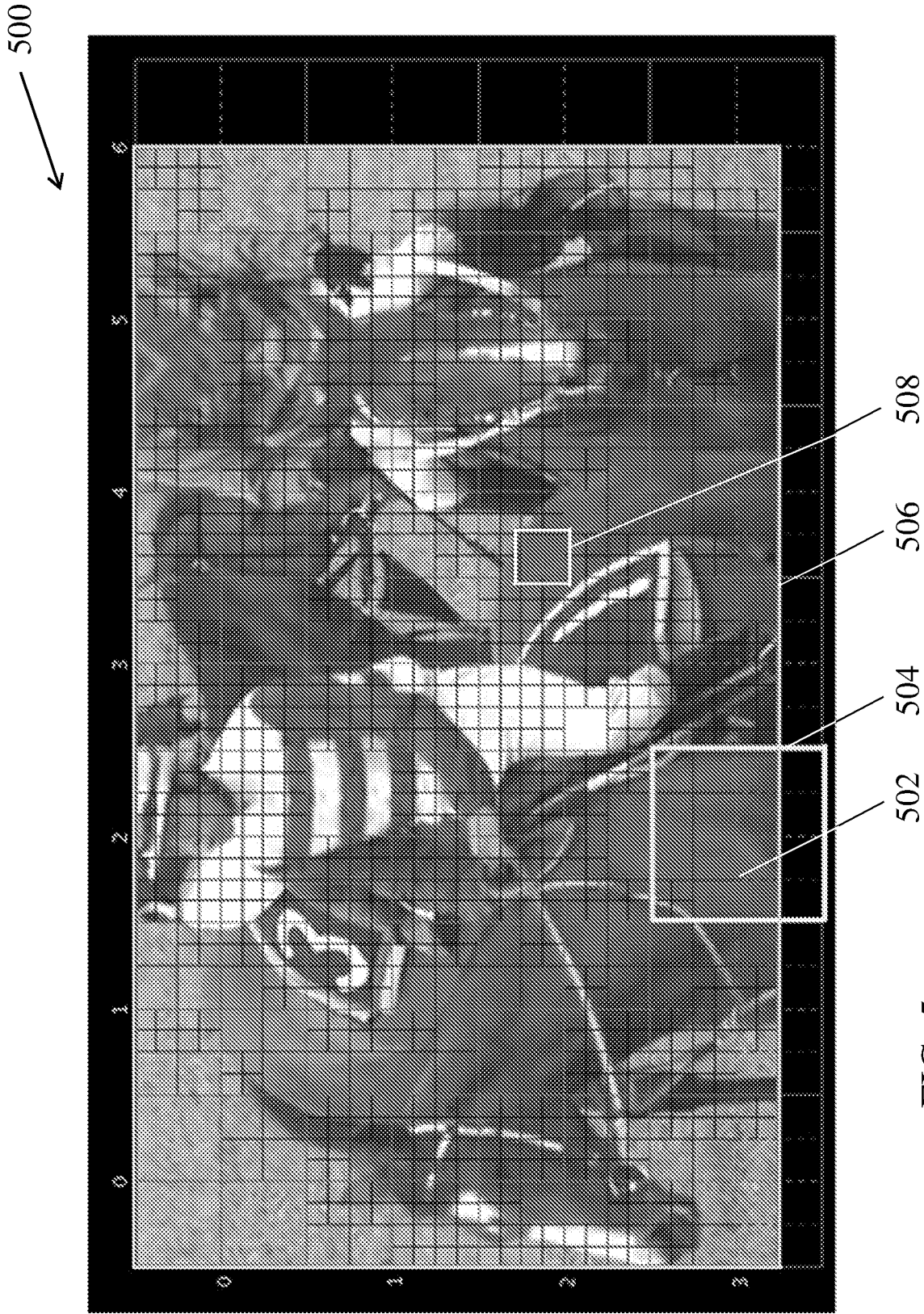


FIG. 5

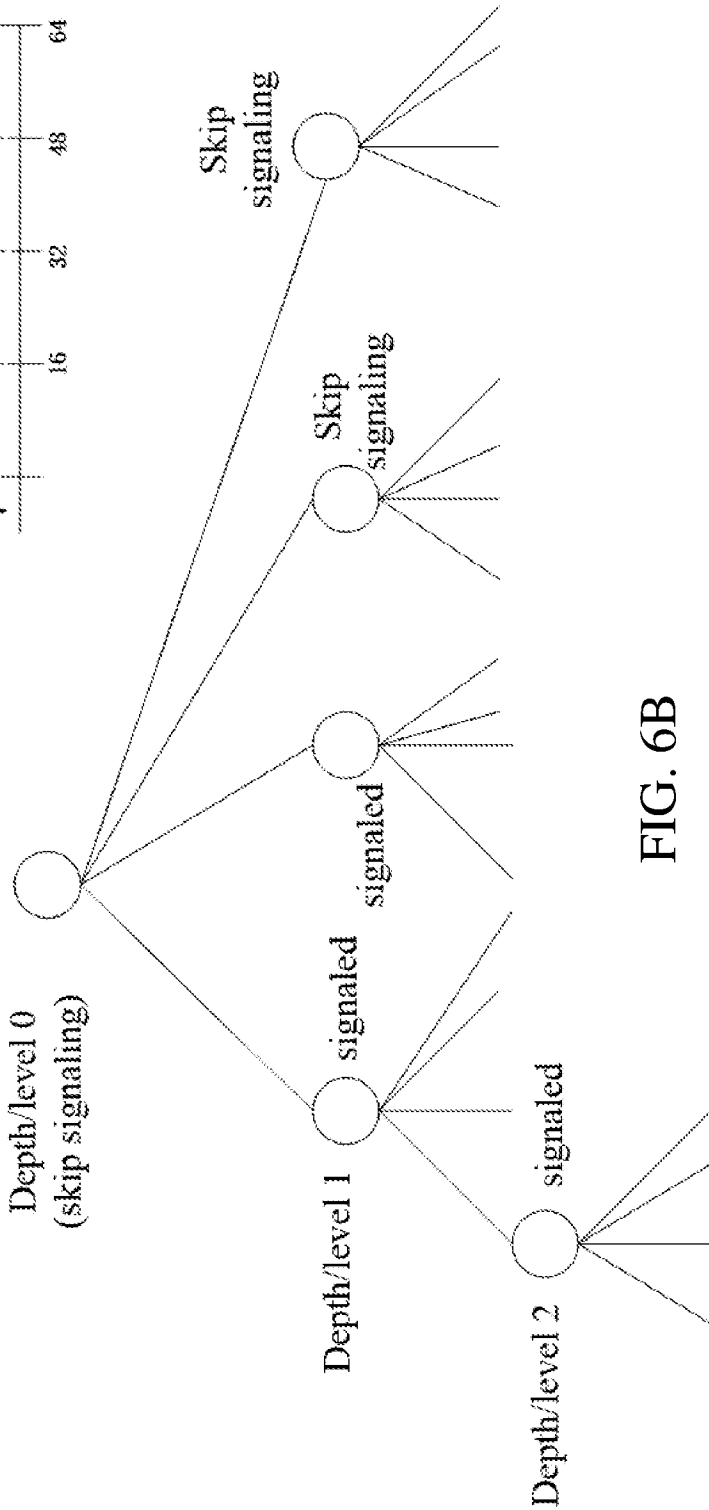
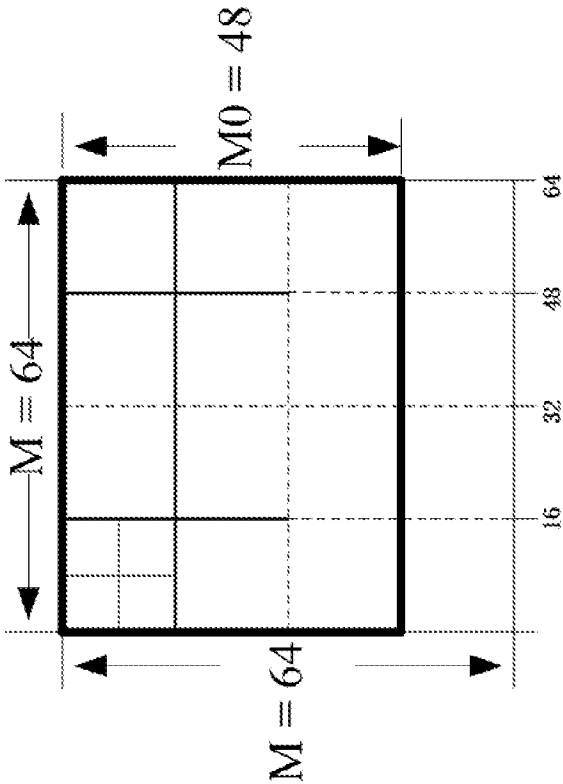


FIG. 7A

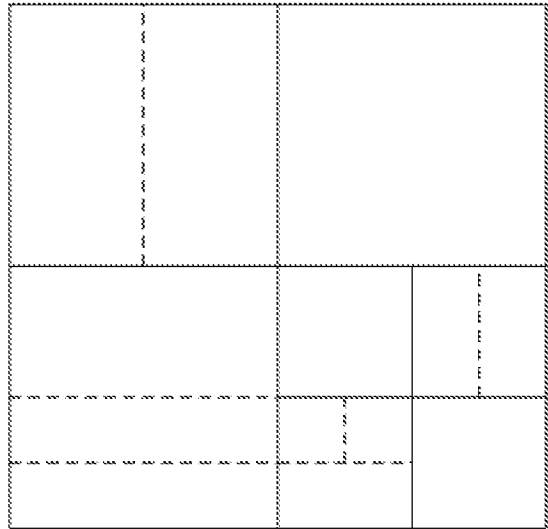
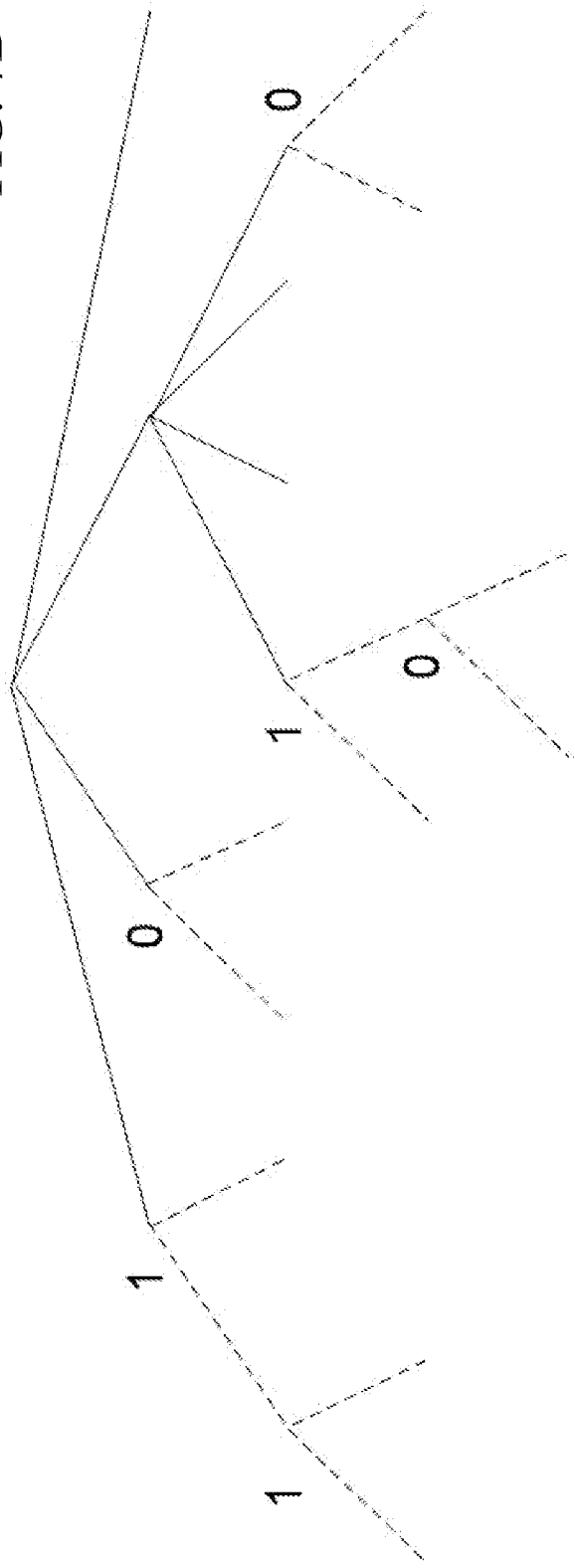


FIG. 7B



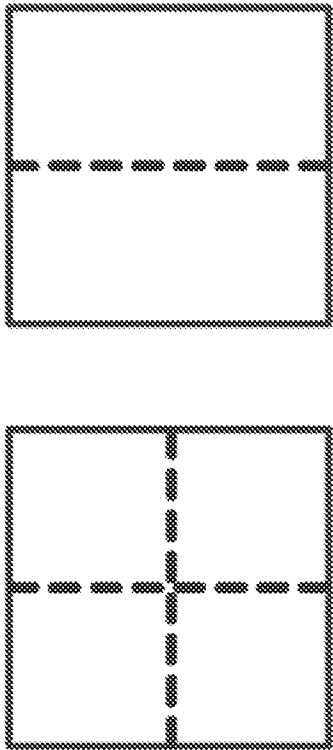


FIG. 8A

FIG. 8B

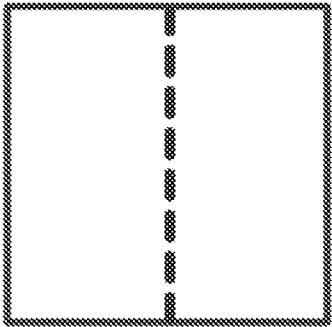


FIG. 8C

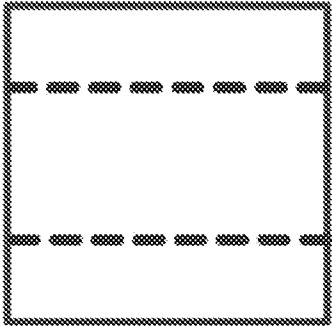


FIG. 8D

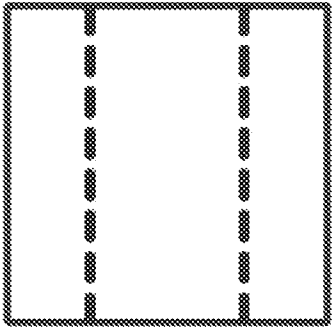


FIG. 8E

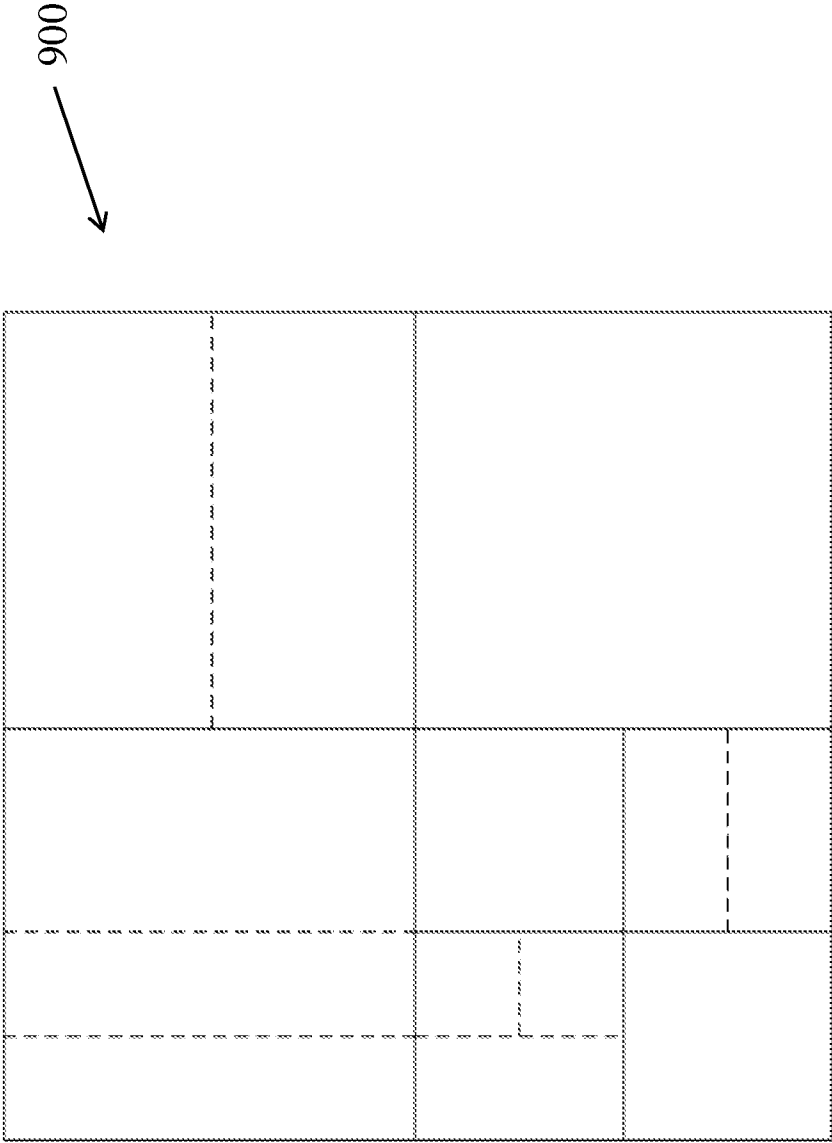


FIG. 9

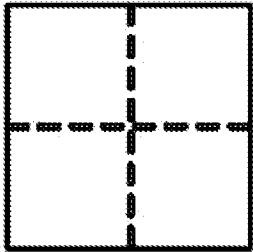


FIG. 10A

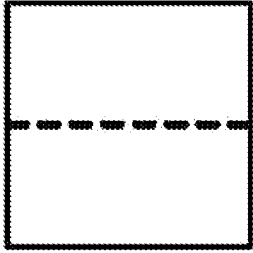


FIG. 10B

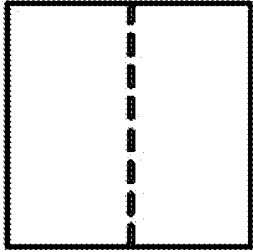


FIG. 10C

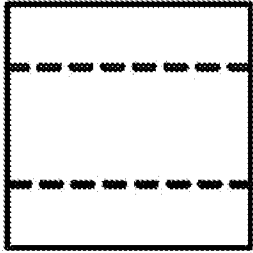


FIG. 10D

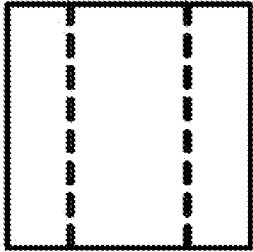


FIG. 10E

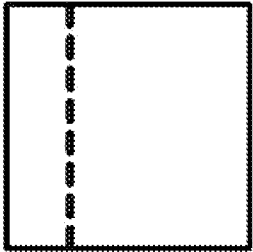


FIG. 10F

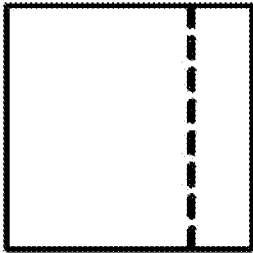


FIG. 10G

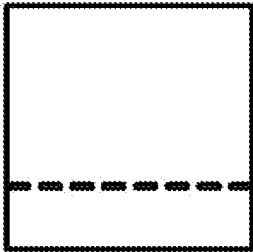


FIG. 10H

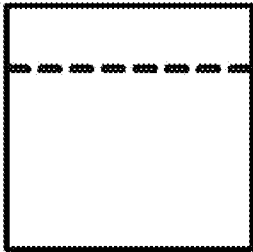


FIG. 10I

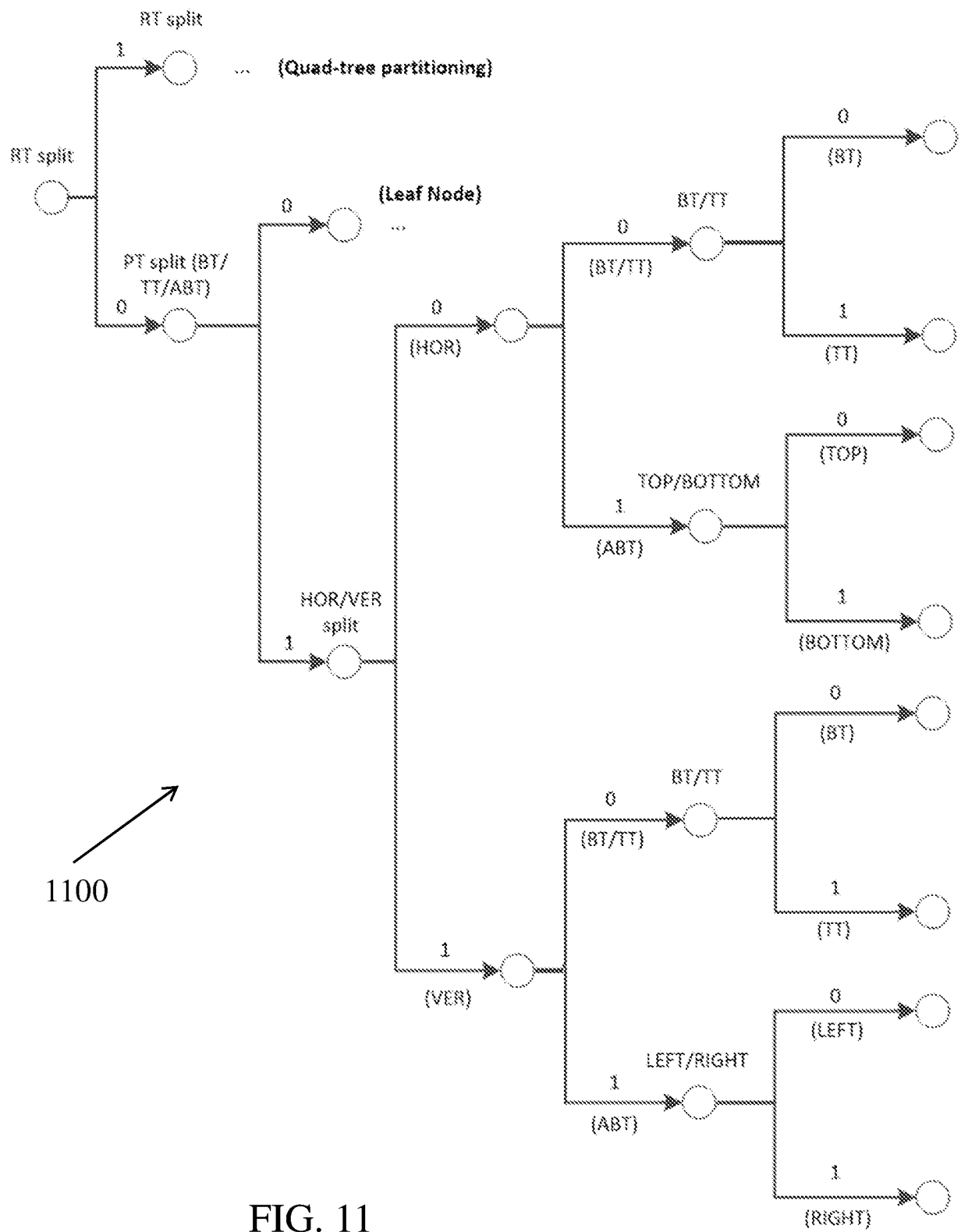


FIG. 11

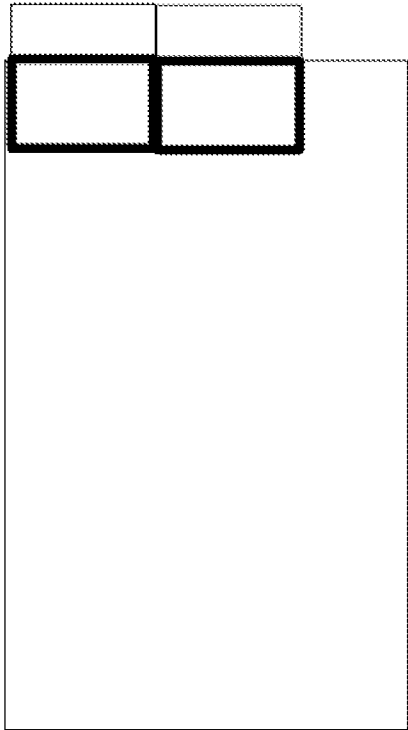


FIG. 12B

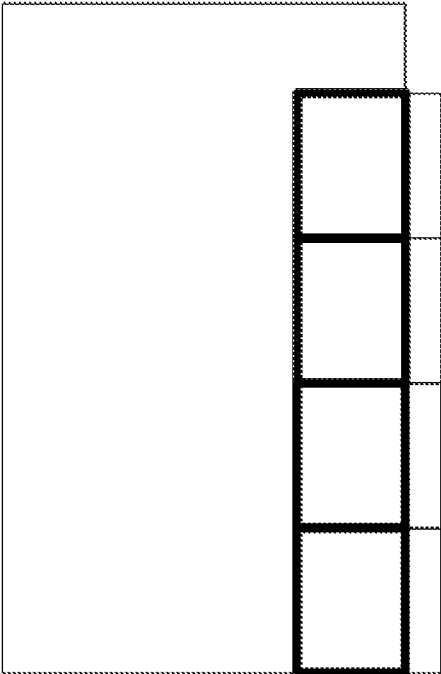


FIG. 12A

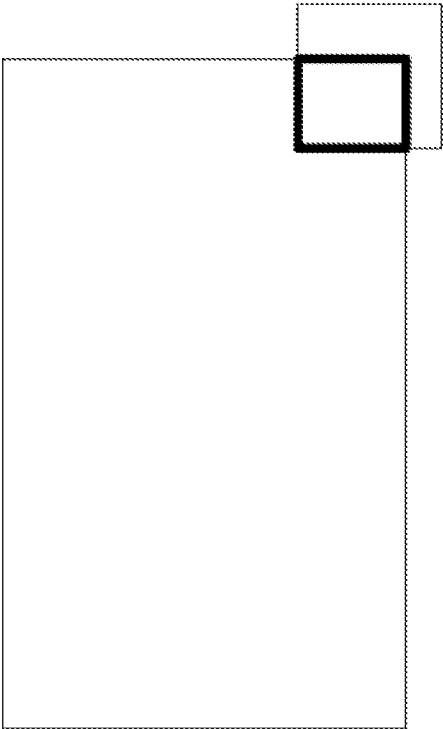


FIG. 12C

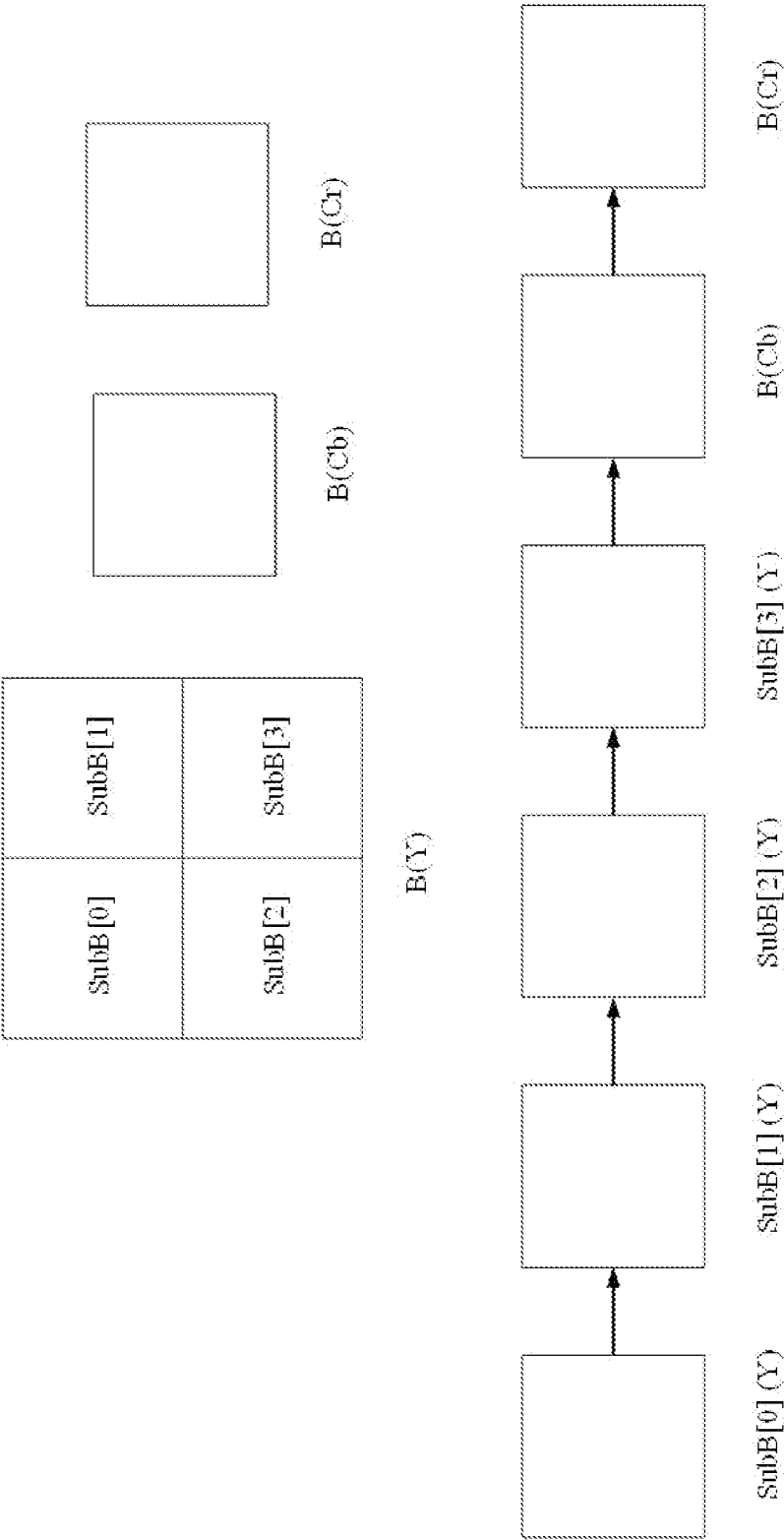


FIG. 13

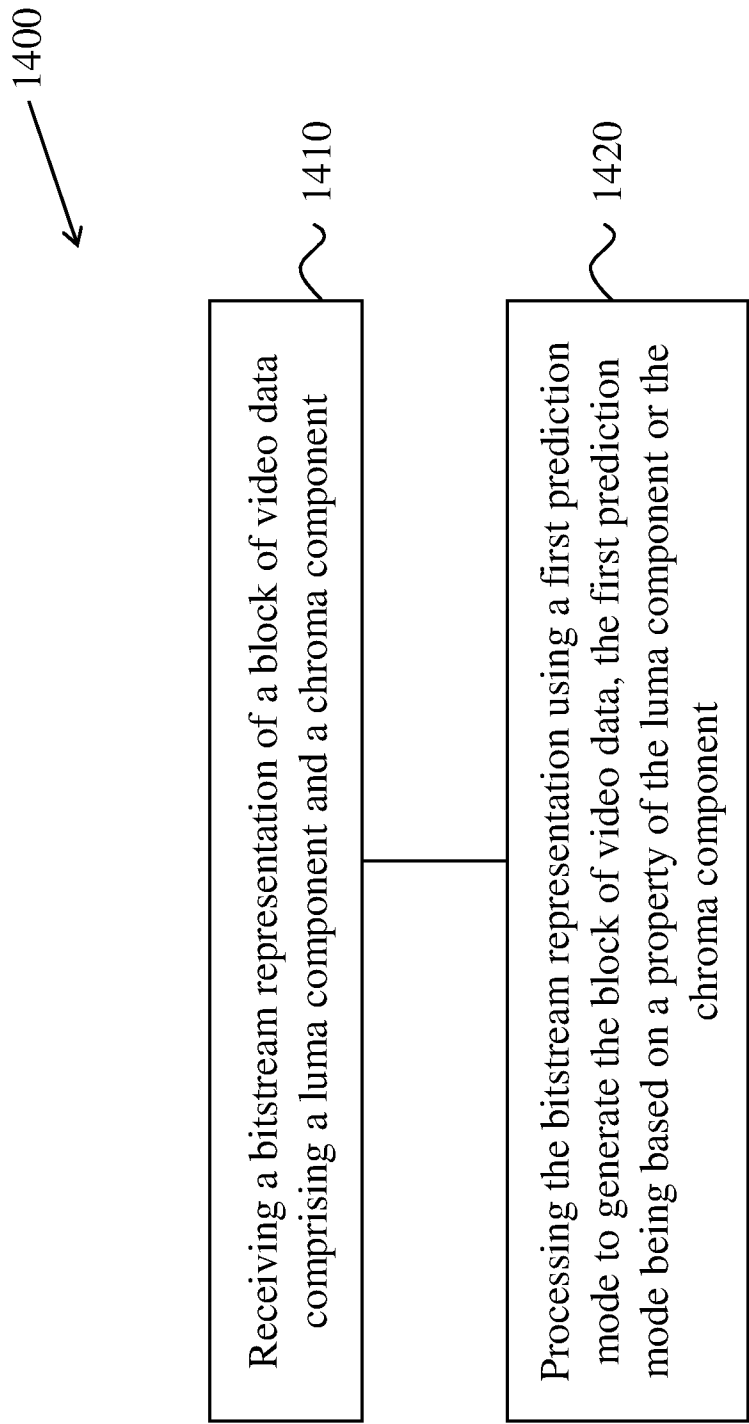


FIG. 14

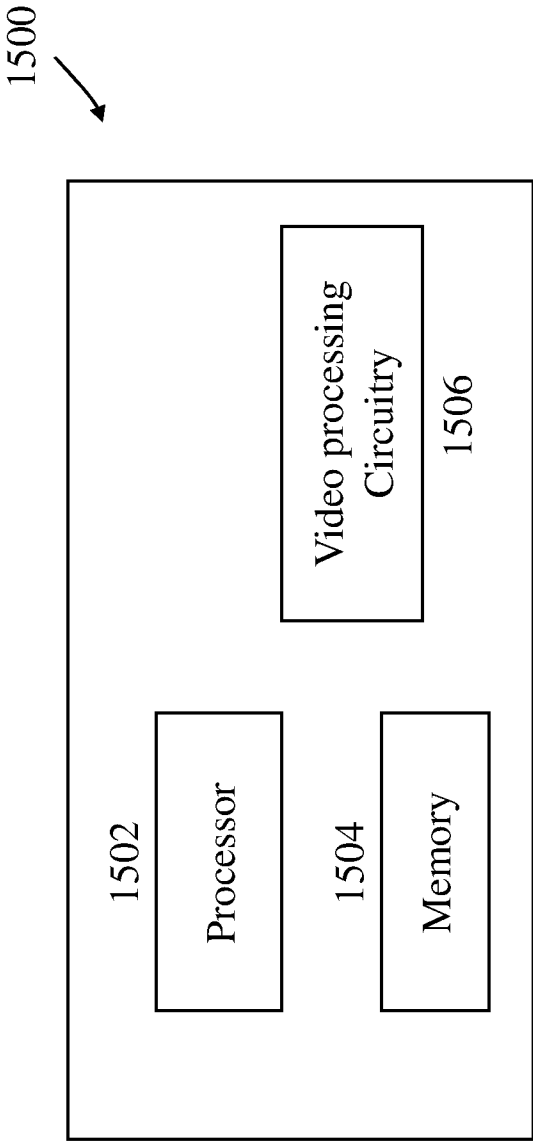


FIG. 15

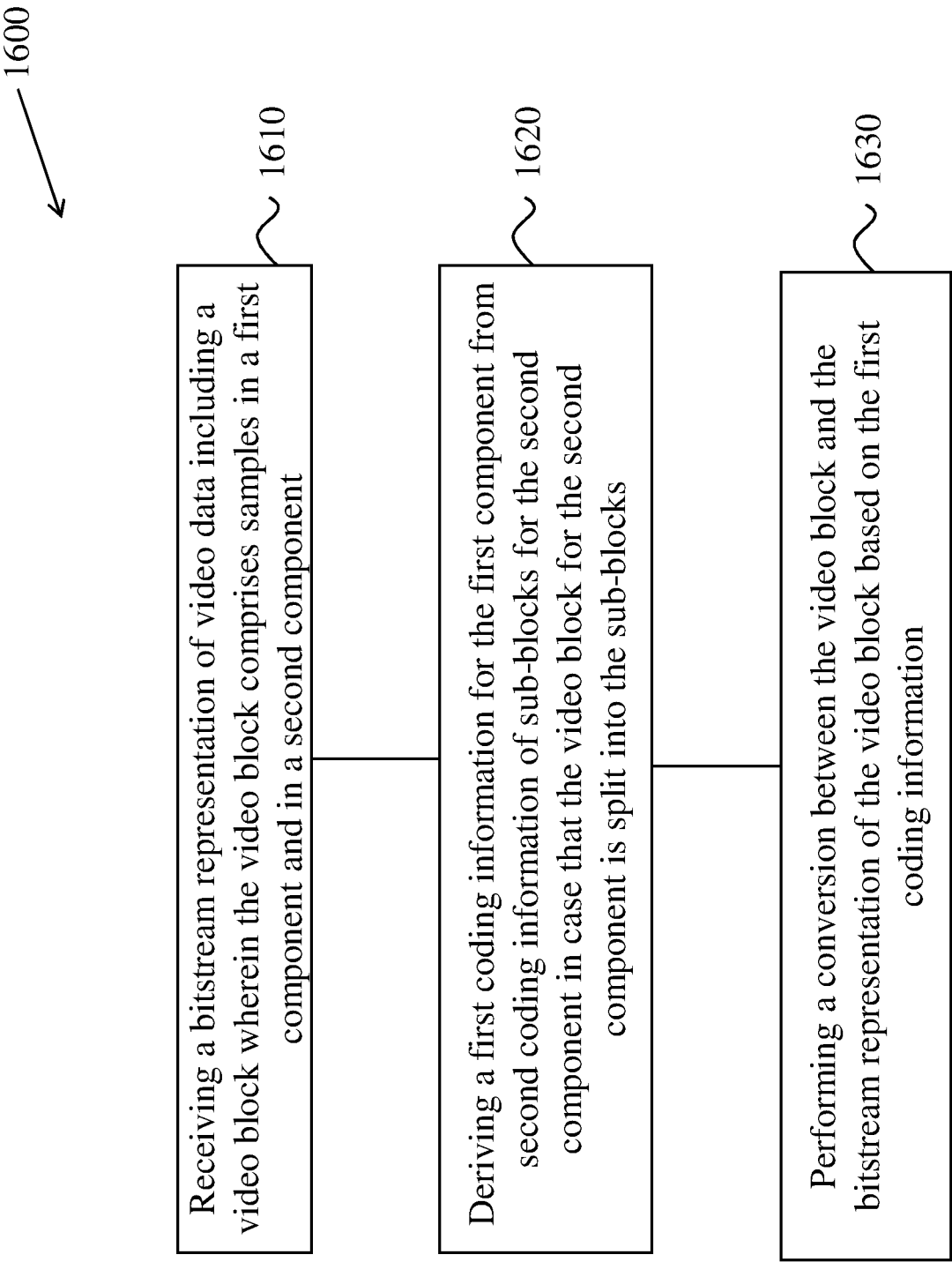


FIG. 16

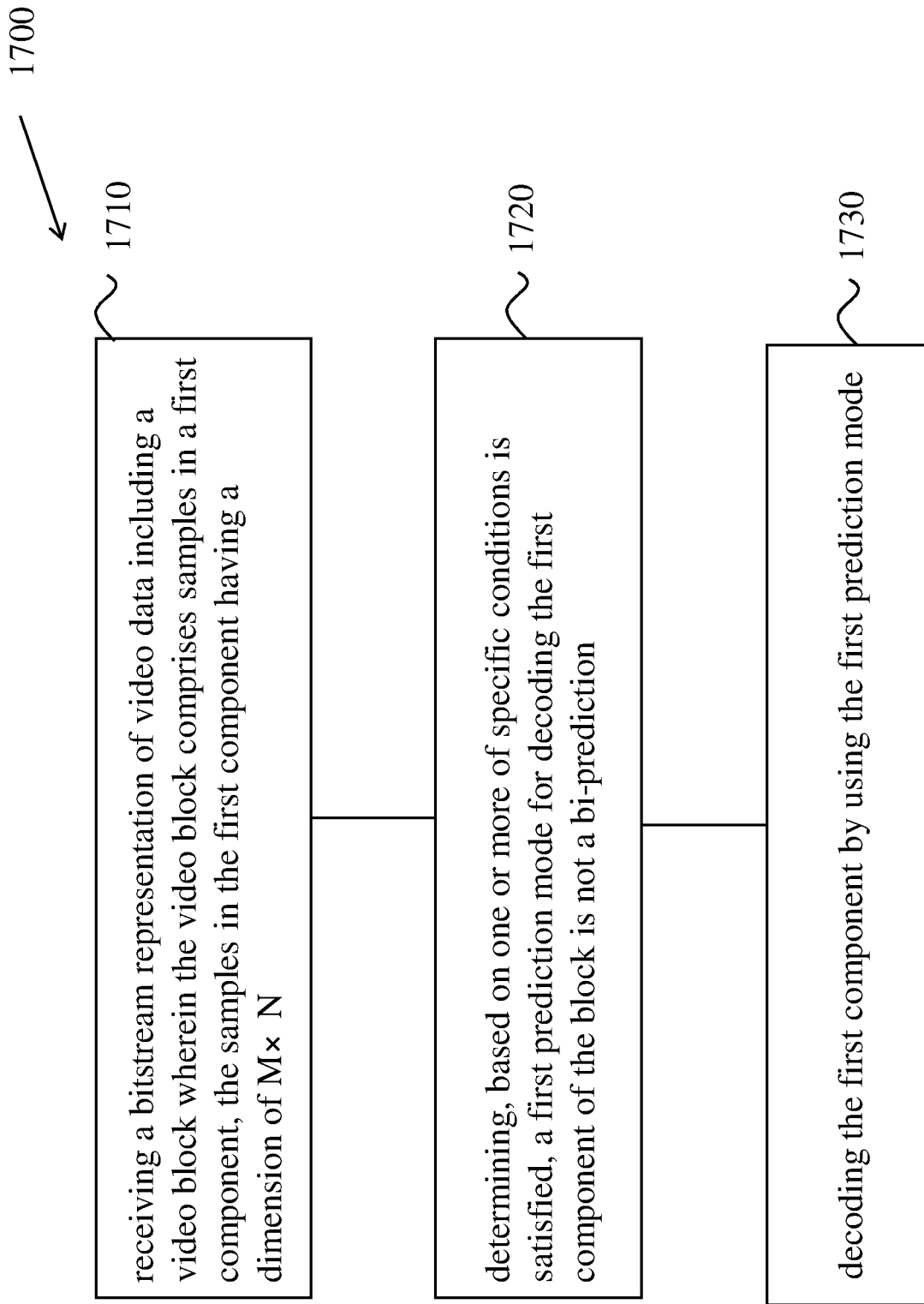


FIG. 17

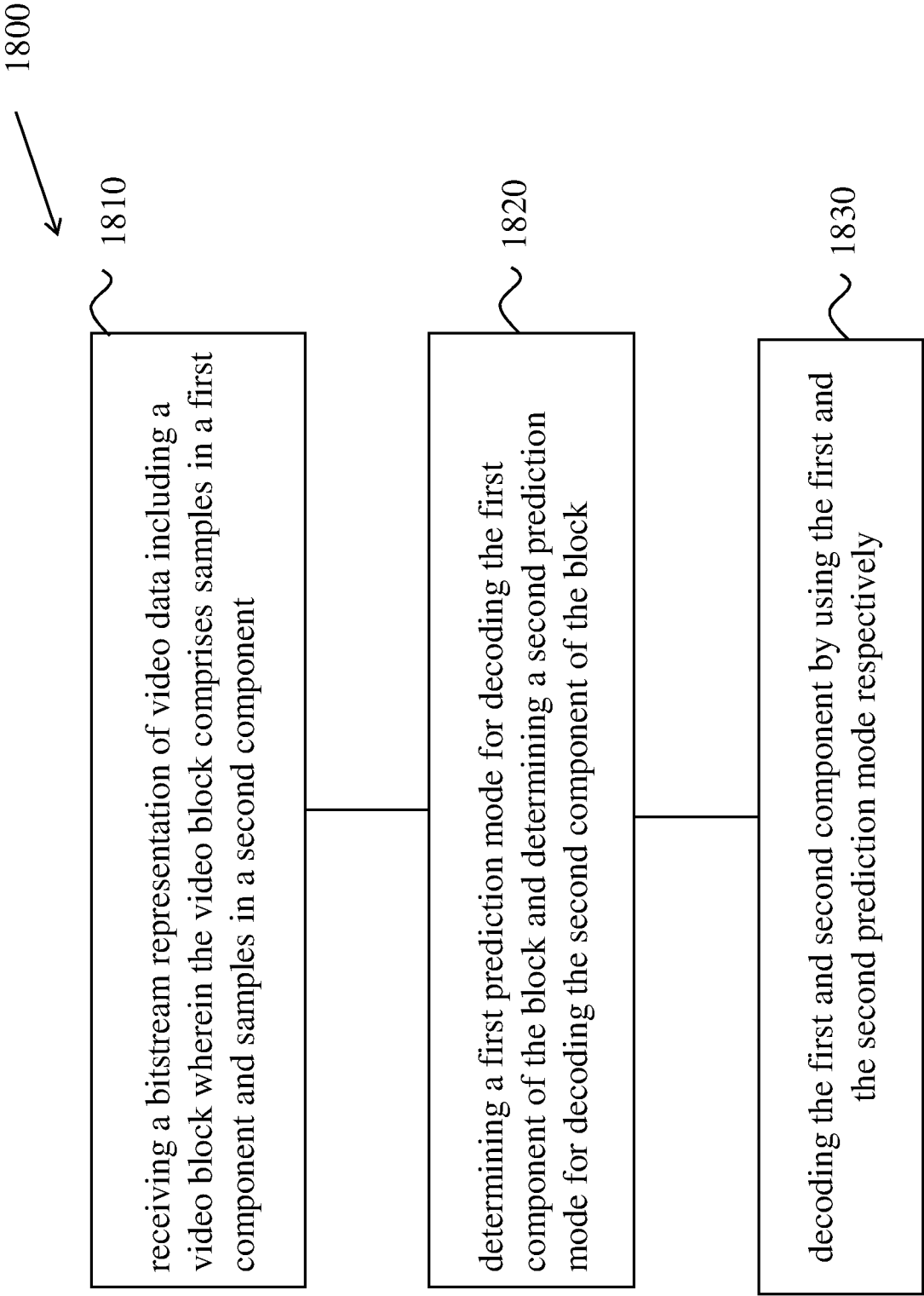


FIG. 18

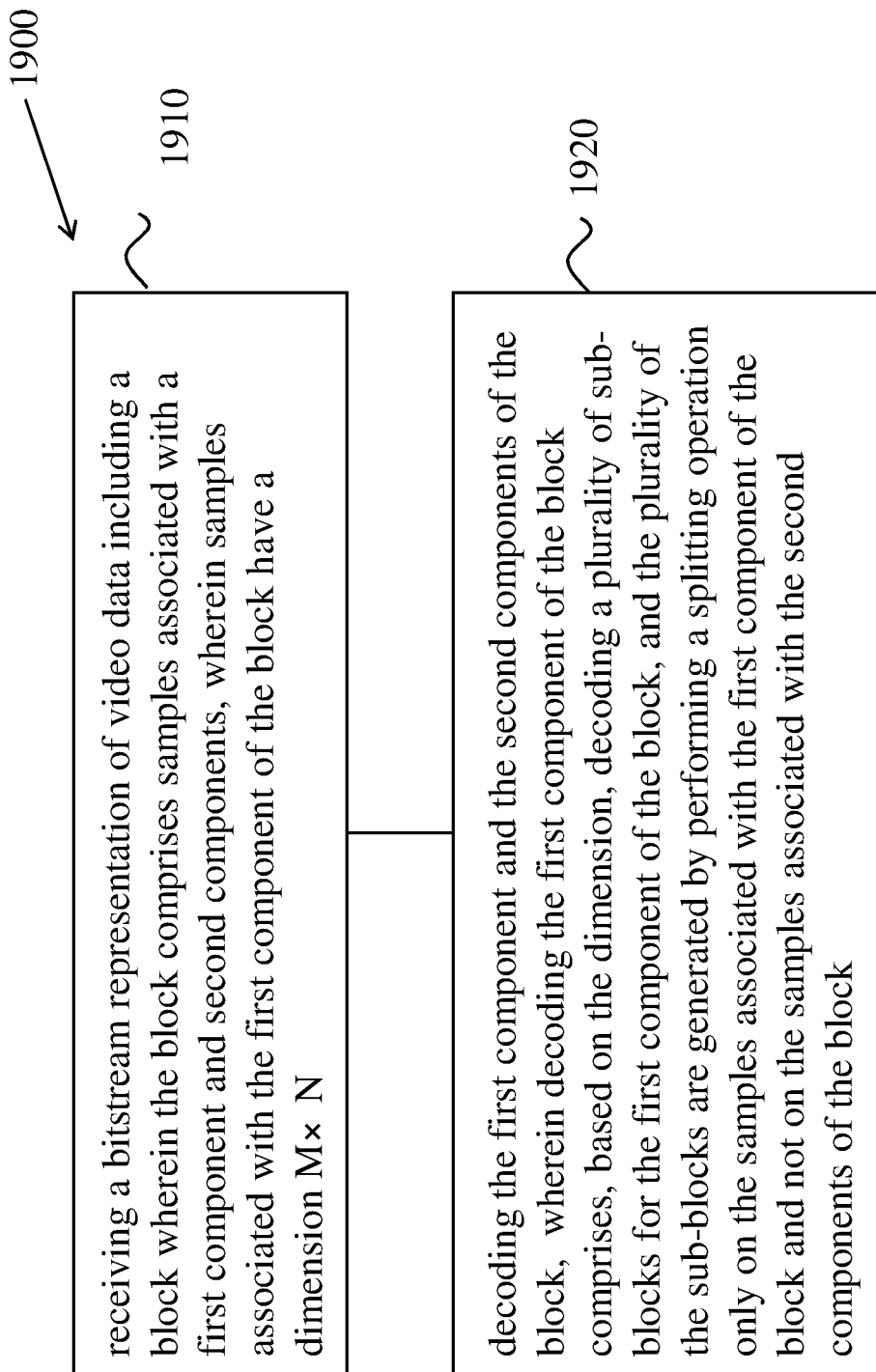


FIG. 19

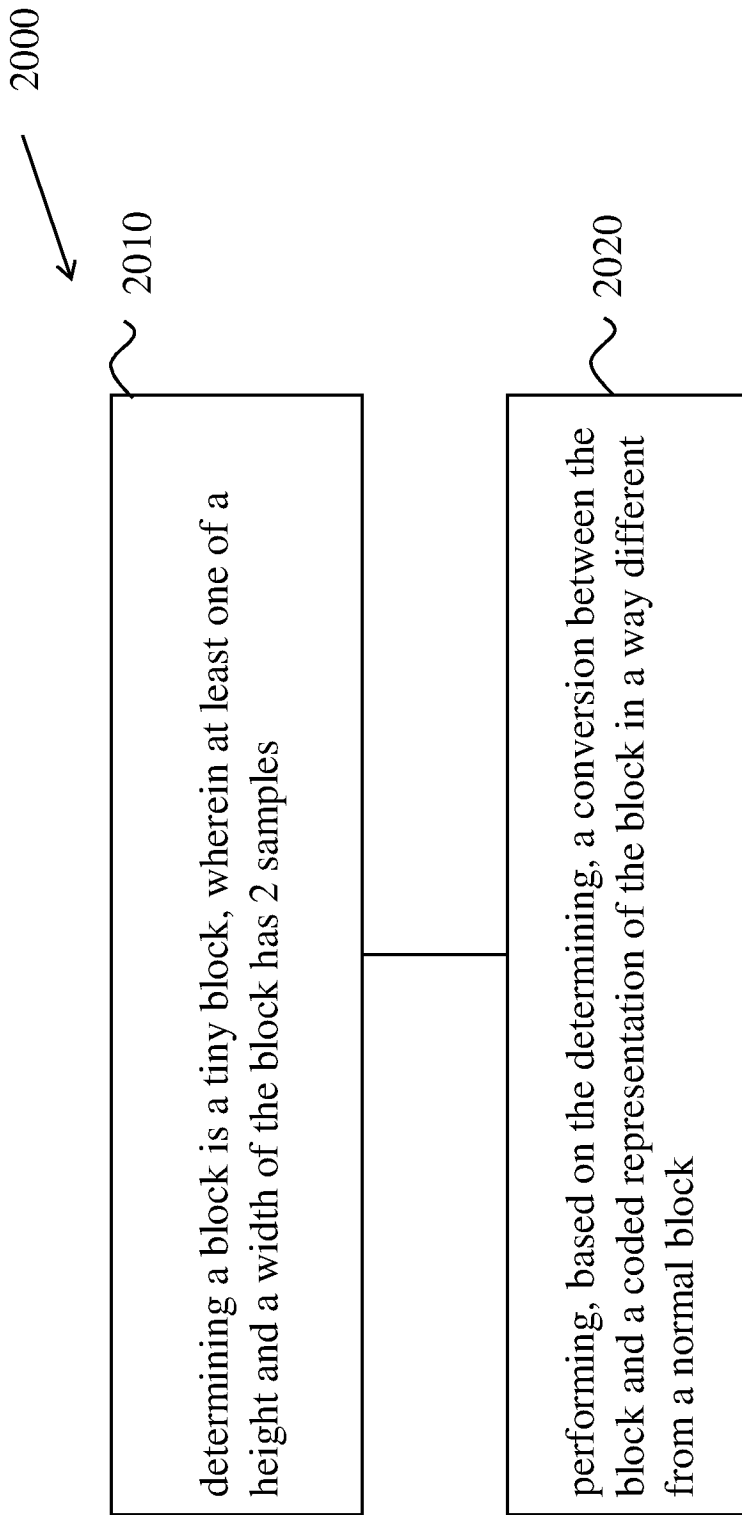


FIG. 20