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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 April 2020 (30.04.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/084604 A1

(51) International Patent Classification:
H04N 19/119 (2014.01) *H04N 19/70* (2014.01)
H04N 19/176 (2014.01) *H04N 19/96* (2014.01)

(21) International Application Number:
PCT/IB2019/059219

(22) International Filing Date:
28 October 2019 (28.10.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2018/111990
26 October 2018 (26.10.2018) CN
PCT/CN2018/119316
05 December 2018 (05.12.2018) 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, California 90066 (US).

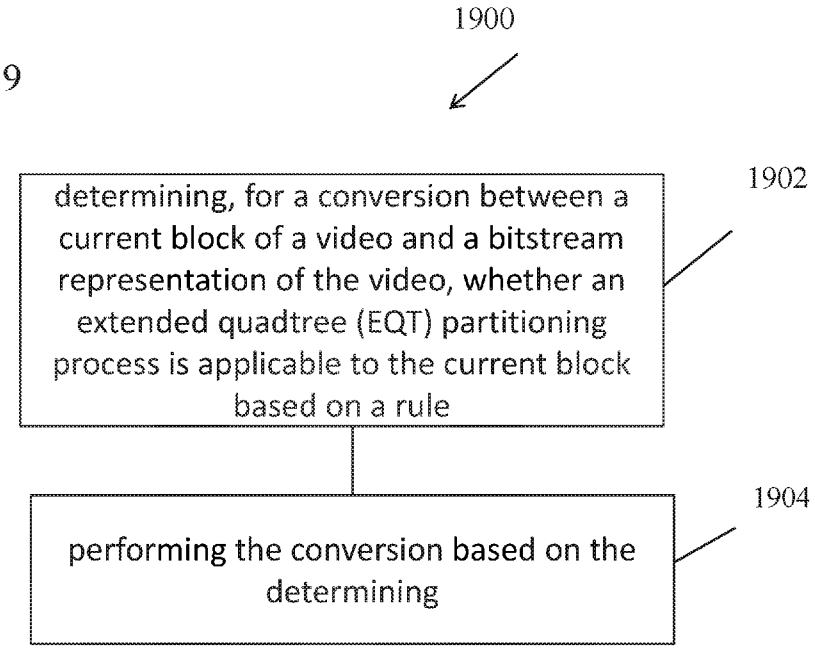
(72) Inventors: **ZHANG, Li**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **LIU, Hongbin**; Jinritoutiao Post Office, China Satellite Communications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

(54) Title: FAST METHODS FOR PARTITION TREE DECISION

FIG. 19



(57) Abstract: A method for video processing comprises determining, for a conversion between a current block of a video and a bitstream representation of the video, whether an extended quadtree (EQT) partitioning process is applicable to the current block based on a rule, and performing the conversion based on the determining. The EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and the rule specifies a maximum depth of the EQT partitioning process based on an attribute associated with the current block.

WO 2020/084604 A1

[Continued on next page]

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:

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

— *in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE*

FAST METHODS FOR PARTITION TREE DECISION

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 benefit of International Patent Application No. PCT/CN2018/111990, filed on October 26, 2018 and International Patent Application No. PCT/CN2018/119316, filed on December 5, 2018. The entire disclosure of the above applications are incorporated by reference as part of the disclosure of this patent document.

TECHNICAL FIELD

[0002] The present document relates to video and image coding technology.

BACKGROUND

[0003] Currently, efforts are underway to improve the performance of current video codec technologies to provide better compression ratios or provide video coding and decoding schemes that allow for lower complexity or parallelized implementations. Industry experts have recently proposed several new video coding tools and tests are currently underway for determining their effectivity.

SUMMARY

[0004] Some disclosed embodiments relate to coding and decoding of images and video pictures using a rule based extended quadtree partitioning process. In one beneficial aspect, certain aspects of the rule are pre-defined, allowing encoder and decoder embodiments to generate partition tree and performing decoding using fewer computational resources that traditional image and video coding techniques.

[0005] In one example aspect, a method for video process is disclosed. The method includes determining, for a conversion between a current block of a video and a bitstream representation of the video, whether an extended quadtree (EQT) partitioning process is applicable to the current block based on a rule, and performing the conversion based on the determining. The EQT partitioning process includes partitioning a given block into exactly four

sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and the rule specifies a maximum depth of the EQT partitioning process based on an attribute associated with the current block.

[0006] In another example aspect, a method of visual media processing includes performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and wherein the rule specifies that in case that the rule is used for partitioning the current block, then each subblock is further split into a binary tree (BT) partitioning or another EQT partitioning, and both BT and the another EQT partitioning have depths that meet a pre-defined relationship.

[0007] In another example aspect, a method of visual media processing is disclosed. The method includes performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and wherein the rule allows the EQT partitioning process for the current block based on a width or a height of the current block.

[0008] In yet another aspect, another method of visual media processing is disclosed. The method includes performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and wherein the rule allows the EQT partitioning process for the current block based on a position of the current block.

[0009] In yet another aspect, another method of visual media processing is disclosed. The method includes performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning

a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and wherein the rule allows a maximum depth of the EQT partitioning process to depend on a distance between a current picture of the current block and a reference picture for the current block or a quantization parameter of the current block or a temporal layer id of the current picture.

[0010] These, and other, aspects are described in greater detail throughout the document.

BRIEF DESCRIPTIONS OF THE DRAWINGS

[0011] FIG. 1 shows a block diagram for an example implementation of video encoding and decoding.

[0012] FIG. 2 shows an example of macroblock (MB) partitioning according to the H.264/Audio Video Codec (AVC) standard.

[0013] FIG. 3 shows an example of modes for splitting a coding block (CB) into prediction blocks (PBs) subject to certain size constraints. E.g., intra-pictures are allowed to use only M x M and M/2 x M/2 sizes.

[0014] FIG. 4 shows an example of subdivisions of a coding tree block (CTB) into CBs and transform blocks (TBs). In the drawing, solid lines indicate CB boundaries and dashed lines indicate TB boundaries. Left side is CTB with partitioning, and right side the corresponding quadtree.

[0015] FIG. 5 is an example illustration of a Quad Tree Binary Tree (QTBT) structure.

[0016] FIG. 6 shows various examples of block partitioning.

[0017] FIG. 7A - 7K show examples of block partitioning.

[0018] FIG. 8A - 8D show examples of block partitioning.

[0019] FIG. 9A-9B shows an example of generalized triple tree partitioning (GTT).

[0020] FIG. 10 shows an example of syntax and semantics for versatile boundary partitioning.

[0021] FIG. 11A - 11B shows an example of allowed EQT patterns that may be further split into EQT or BT.

[0022] FIG. 12 shows an example of binarization of partitioning.

[0023] FIG. 13A and 13B show examples of horizontal and vertical EQTs.

[0024] FIG. 14 shows an example hardware platform for implementing some disclosed

methods.

[0025] FIG. 15 shows another example hardware platform for implementing some disclosed methods.

[0026] FIG. 16 is a flowchart of an example method of visual media processing.

[0027] FIG. 17 shows an example of 7 specific positions used in EQT partition early termination.

[0028] FIG. 18 is a block diagram of an example video processing system in which disclosed techniques may be implemented.

[0029] FIG. 19 is a flowchart representation of a method for video processing in accordance with the present disclosure.

DETAILED DESCRIPTION

[0030] The present document provides several techniques that can be embodied into digital video or image, collectively called visual media, encoders and decoders. Section headings are used in the present document for clarity of understanding and do not limit scope of the techniques and embodiments disclosed in each section only to that section.

[0031] **1. Brief Overview**

[0032] This document is related to image/video coding, especially on the partition structure, i.e., how to split one Coding Tree Unit (CTU) into multiple Coding Units (CUs) and how to accelerate encoders to select the best partition structure. It may be applied to the existing video coding standard like HEVC, or the standard (Versatile Video Coding) to be finalized. It may be also applicable to future video coding standards or video codec.

[0033] **2. Introduction to video coding and decoding technologies**

[0034] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. An example of a typical HEVC encoder framework is depicted in FIG. 1.

[0035] **2.1 Partition tree structure in H.264/AVC**

[0036] 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.

[0037] An intra-coded block uses spatial prediction to exploit spatial correlation among pixels. Two partitions are defined: 16x16 and 4x4.

[0038] An inter-coded block uses temporal prediction, instead of spatial prediction, by estimating motion among pictures. Motion can be estimated independently for either 16x16 macroblock or any of its sub-macroblock partitions: 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 (see FIG. 5). Only one motion vector (MV) per sub-macroblock partition is allowed.

[0039] FIG. 2 shows an example of MB partitions in H.264/AVC.

[0040] **2.2 Partition tree structure in HEVC**

[0041] In HEVC, a CTU is split into 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 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.

[0042] In the following, the various features involved in hybrid video coding using HEVC are highlighted as follows.

[0043] 1) Coding tree units 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×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.

[0044] 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).

[0045] 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.

[0046] 4) 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.

[0047] FIG. 3 shows an example of modes for splitting a coding block (CB) into prediction blocks (PBs) subject to certain size constraints. E.g., intra-pictures are allowed to use only M x M and M/2 x M/2 sizes.

[0048] FIG. 4 shows an example of subdivisions of a coding tree block (CTB) into CBs and transform blocks (TBs). In the drawing, solid lines indicate CB boundaries and dashed lines indicate TB boundaries. Left side is CTB with partitioning, and right side the corresponding quadtree.

[0049] **2.3 Quadtree plus binary tree block structure with larger CTUs in JEM**

[0050] To explore the future video coding technologies beyond HEVC, Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM).

[0051] **2.3.1 QTBT block partitioning structure**

[0052] Different from HEVC, 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. 5, 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.

[0053] The following parameters are defined for the QTBT partitioning scheme.

- CTU size: the root node size of a quadtree, the same concept as in HEVC
- *MinQTSize*: the minimum allowed quadtree leaf node size
- *MaxBTSize*: the maximum allowed binary tree root node size
- *MaxBTDepth*: the maximum allowed binary tree depth
- *MinBTSize*: the minimum allowed binary tree leaf node size

[0054] 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 (e.g., the *MinQTSize*) to 128×128 (e.g., the CTU size). If the quadtree leaf node is 128×128, it will not be further split by the binary tree since the size exceeds the *MaxBTSize* (e.g., 64×64). Otherwise, the quadtree leaf 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* (e.g., 4), no further splitting is considered. When the binary tree node has width equal to *MinBTSize* (e.g., 4), no further horizontal splitting is considered. Similarly, when the binary tree node has 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.

[0055] In each splitting (i.e., non-leaf) node of the binary tree, for example as shown in FIG. 5, 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.

[0056] 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 luma 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.

[0057] 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.

[0058] **2.4 Triple-tree for VVC**

[0059] In some cases, tree types other than quad-tree and binary-tree are supported. In the implementation, two more triple tree (TT) partitions, i.e., horizontal and vertical center-side triple-trees are introduced, as shown in Figure 6, items (d) and (e).

[0060] FIG. 6 shows example of block partitioning patterns. (a) quad-tree partitioning (b) vertical binary-tree partitioning (c) horizontal binary-tree partitioning (d) vertical center-side triple-tree partitioning (e) horizontal center-side triple-tree partitioning

[0061] There are two levels of trees, region tree (quad-tree) and prediction tree (binary-tree or triple-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’.

[0062] 2.5 Extended Quad Tree

[0063] A extended quad tree (EQT) partitioning structure corresponding to a block partitioning process including an extended quad tree partitioning process for the block of video data, wherein the extended quad partitioning structure represents partitioning the block of video data into final sub-blocks, and when the extended quad tree partitioning process decides to apply extended quad tree partition to one given block, said one given block is always split into four sub-block; decoding the final sub-blocks based on the video bitstream; and decoding the block of video data based on the final sub-blocks decoded according to the EQT structure derived. EQT is presented in the above-captioned patent application, incorporated by reference herein.

[0064] The EQT partitioning process can be applied to a given block recursively to generate EQT leaf nodes. Alternatively, when EQT is applied to a certain block, for each of the sub-block due to EQT, it may further be split into BT and/or QT and/or TT and/or EQT and/or other kinds of partition trees.

[0065] In one example, EQT and QT may share the same depth increment process and same restrictions of leaf node sizes. In this case, the partitioning of one node could be implicitly terminated when the size of the node reaches a minimum allowed quad tree leaf node size or EQT depth associated with the node reaches a maximum allowed quad tree depth.

[0066] In some embodiments, EQT and QT may share different depth increment process and/or restrictions of leaf node sizes. The partitioning of one node by EQT is implicitly terminated when the size of the node reaches a minimum allowed EQT leaf node size or EQT depth associated with the node reaches a maximum allowed EQT depth. In one example, furthermore, the EQT depth and/or the minimum allowed EQT leaf node sizes may be signaled in sequences parameter set (SPS), and/or picture parameter set (PPS), and/or slice header, and/or CTU, and/or regions, and/or tiles, and/or CUs.

[0067] Instead of using the current quad tree partition applied to a square block, for a block with MxN (M and N are non-zero positive integer values, either equal or unequal) size, in EQT, one block may be split equally into four partitions, such as M/4 x N or M x N/4 (examples are depicted in FIG. 7A and FIG. 7B or split equally into four partitions and the partition size is dependent on the maximum and minimum values of M and N. In one example, one 4x32 block may be split into four 4x8 sub-blocks while a 32x4 block may be split into four 8x4 sub-blocks.

[0068] Instead of using the current quad tree partition applied to a square block, for a block

with MxN (M and N are non-zero positive integer values, either equal or unequal) size, in EQT, one block may be split unequally into four partitions, such as two partitions are with size equal to (M*w0/w) x (N*h0/h) and the other two are with (M*(w-w0)/w) x (N*(h-h0)/h).

[0069] For example, w0 and w may be equal to 1 and 2, respectively that is the width is reduced by half while the height could use other ratios instead of 2:1 to get the sub-blocks. Examples for this case are depicted in FIG. 7C and FIG. 7E. Alternatively, h0 and h may be equal to 1 and 2, respectively, that is the height is reduced by half while the width could use other ratios instead of 2:1. Examples for this case are depicted in FIG. 7D and FIG. 7F.

[0070] FIG. 7G and 7H show two alternative examples of quad tree partitioning.

[0071] FIG. 7I shows a more general case of quad tree partitioning with different shapes of partitions

[0072] FIG. 7J and 7K show general examples of FIGs. 7A and 7B.

[0073] FIG. 7C shows a sub-block width fixed to be M/2, height equal to N/4 or 3N/4, smaller for top two partitions. FIG. 7D shows a sub-block height fixed to be N/2, width equal to M/4 or 3M/4, smaller for left two partitions

[0074] FIG. 7E shows a sub-block width fixed to be M/2, height equal to 3N/4 or N/4, smaller for bottom two partitions. FIG. 7F shows a sub-block height fixed to be N/2, width equal to 3M/4 or M/4, smaller for right two partitions. The following example dimensions are shown FIG. 7G M x N/4 and M/2 x N/2; FIG. 7H: N x M/4 and N/2 x M/2, FIG. 7I: M1xN1, (M-M1)xN1, M1x(N-N1) and (M-M1)x(N-N1); FIG. 7J M x N1, MxN2, MxN3 and MxN4, where N1+N2+N3+N4=N, FIG. 7K: M1 x N, M2xN, M3xN and M4xN where M1+M2+M3+M4=M.

[0075] A flexible tree (FT) partitioning structure corresponding to a block partitioning process including an FT partitioning process for the block of video data, wherein the FT partitioning structure represents partitioning the block of video data into final sub-blocks, and when FT partitioning process decides to apply FT partition to one given block, said one given block is split into K sub-blocks wherein K could be larger than 4; decoding the final sub-blocks based on the video bitstream; and decoding the block of video data based on the final sub-blocks decoded according to the FT structure derived.

[0076] The FT partitioning process can be applied to a given block recursively to generate FT tree leaf nodes. The partitioning of one node is implicitly terminated when the node reaches a minimum allowed FT leaf node size or FT depth associated with the node reaches a maximum

allowed FT depth.

[0077] In some embodiments, when FT is applied to a certain block, for each of the sub-block due to FT, it may further be split into BT, and/or QT, and/or EQT, and/or TT, and/or other kinds of partition trees.

[0078] In some embodiments, furthermore, the FT depth or the minimum allowed FT leaf node sizes or the minimum allowed partition size for FT may be signaled in sequences parameter set (SPS), and/or picture parameter set (PPS), and/or slice header, and/or CTU, and/or regions, and/or tiles, and/or CUs.

[0079] Similar to the proposed EQT, all of the sub-blocks due to FT partitions may be with the same size; alternatively, the sizes of different sub-blocks may be different.

[0080] In one example, K is equal to 6 or 8. Some examples are depicted in FIG. 8A-8D, which show examples of FT partitions (K=6 in FIGs. 8C and 8D, or 8 in FIG. 8A and 8B)

[0081] For the TT, the restriction of splitting along either horizonal or vertical may be removed.

[0082] In one example, a generalized TT (GTT) partition pattern may be defined as splitting for both horizontal and vertical. Examples are shown in FIG. 9A and 9B.

[0083] The proposed methods may be applied under certain conditions. In other words, when the condition(s) are not satisfied, there is no need to signal the partition types.

[0084] In some embodiments, the proposed methods may be used to replace the existing partition tree types. Alternatively, furthermore, the proposed methods may be only used as a replacement under certain conditions.

[0085] In one example, the condition may include the picture and/or slice types; and/or block sizes; and/or the coded modes; and/or whether one block is located at picture/slice/tile boundary.

[0086] In one example, the proposed EQT may be treated in the same way as QT. In this case, when it is indicated that the partition tree type is QT, more flags/indications of the detailed quad-tree partition patterns may be further signaled. Alternatively, EQT may be treated as additional partition patterns.

[0087] In one example, the signaling of partitioning methods of EQT or FT or GTT may be conditional, *e.g.,* one or some EQP/FT/GTT partitioning methods may not be used in some cases, and the bits corresponding to these partitioning methods are not signaled.

[0088] **2.6 Border handling**

[0089] In some embodiments, a boundary handling method is proposed to Versatile Video Coding (VVC). A similar method is also adopted into AVS-3.0.

[0090] Since the forced quadtree boundary partition solution in VVC is not optimized. It has been proposed that the boundary partition method can use regular block partition syntax to keep the continuity CABAC engine as well as matching the picture boundary

[0091] The versatile boundary partition obtains the following rules (both encoder and decoder):

[0092] Using exactly same partition syntax of the normal block (non-boundary) (for instance, VTM-1.0 like FIG. 10) for boundary located block, the syntax need to be unchanged.

[0093] If the no split mode is parsed for the boundary CU, used forced boundary partition (FBP) to match the picture boundary. After forced boundary partition (non-singling boundary partition), no further partition. The forced boundary partition is descripted as follow:

[0094] If the size of block is larger than the maximal allowed BT size, forced QT is used to perform the FBP in the current forced partition level;

[0095] Otherwise, if the bottom-right sample of current CU is located below the bottom picture boundary, and not extended the right boundary, forced horizontal BT is used to perform the FBP in the current forced partition level;

[0096] Otherwise, if the bottom-right sample of current CU is located at the right side of the right picture boundary, and not below the bottom boundary, forced vertical BT is used to perform the FBP in the current forced partition level;

[0097] Otherwise, if the bottom-right sample of current CU is located at the right side of the right picture boundary and below the bottom boundary, forced QT is used to perform the FBP in the current forced partition level.

[0098] **3. Problems and shortcomings of current implementations**

[0099] There may be some redundancy between partitions of EQT and QT/BT/TT. For example, for a block with size of MxN, it may be split into vertical BT three times (firstly split to two M/2 *N partitions, then for each M/2*N partition, further apply vertical BT split) to get four M/4*N partitions. Also, to get four M/4xN partitions, the block could choose directly using EQT as FIG. 7B. There still exists a problem how to signal EQT efficiently.

[00100] **4. Example Techniques and Embodiments**

[00101] To address the above-mentioned problems, and others, several methods are proposed

to handle the cases for EQT. Embodiments may include image or video encoder and decoders.

[00102] The techniques listed as examples below should be considered explanations to the general concepts. These embodiments should not be interpreted in a narrow way. Furthermore, these embodiments can be combined in any manner.

[00103] Example 1: When EQT is applied to a certain block, for each of the sub-block due to EQT, it may further be split into BT and/or EQT, and BT and EQT may share the same maximum depth value denoted by D_{BTMax} (e.g., *MaxBTDepth* in section 2.3.1)

[00104] In one example, only two kinds of EQT depicted in FIGs. 7A-7K may be allowed. The two allowed EQT patterns are depicted in FIG. 11A and 11B which shows an example of allowed EQT patterns which may be further split to EQT or BT. For example, one allowed EQT pattern may include a top partition that is full width and one-fourth height, followed by two side-by-side partitions of half width and half height of the block, followed by a bottom partition of full width and one-fourth the height of the block (e.g., FIG. 11A). Another allowed partition includes a lefr portion of full height and one-fourth width, followed by two partitions with half width and half height vertically stacked over each other, followed by a right partition that is full height and one-fourth width (e.g., FIG. 11B). It will be appreciated that, in one aspect, each partition has equal area.

[00105] Similarly, when BT is applied to a certain block, for each of the sub-blocks due to BT, it may further be split into BT and/or EQT, and BT and EQT may share the same maximum depth value.

[00106] EQT and BT may use different depth increment process. For example, when each block may be assigned with a depth value denoted by D_{BT} (D_{BT} may start from 0). If one block (with depth value equal to D_{BT}) is split with EQT, each of the sub-block's depth value is set to $D_{BT}+2$.

[00107] Whenever one block's associated depth is smaller than D_{BTMax}, it may be further split to EQT or BT.

[00108] In some embodiments, the maximum depth value allowed for EQT may be set to the sum of maximum depth value allowed for QT and maximum depth value allowed for BT.

[00109] Example 2: When EQT is allowed for coding a tile/slice/picture/sequence, it may share the same maximumly allowed binary tree root node size (e.g., *MaxBTSize* in section 2.3.1) for coding the same video data unit. Alternatively, EQT may use different maximumly allowed root node size different from that for BT.

[00110] In one example, the maximum EQT size is set to MxN, e.g., M=N=64 or 32. In some embodiments, maximumly allowed root node size for EQT may be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00111] Example 3: One flag is firstly signaled to indicate whether it is BT or EQT before signaling the direction of BT or EQT (e.g., horizontal or vertical), and one more flag may be further signaled to indicate it uses horizontal or vertical splitting direction.

[00112] In one example, the binarization of partitioning is shown in FIG. 12. Table 1 shows example bin value for each bin index. It should be noted that it is equivalent to exchange all the "0" and "1" in the table.

Table 1. Example of partition splitting patterns

Bin String					Partition types
Bin index	0	1	2	3	
value	1	-	-		Quad-Tree
value	0	0	-		Non-QT Split
value	0	1	0	0	Horizontal BT
value	0	1	0	1	Vertical BT
value	0	1	1	0	Horizontal EQT (marked as EQT_CEN_HOR)
value	0	1	1	1	Vertical EQT(marked as EQT_CEN_VER)
meaning	QT or not?	Further split with other partition trees?	BT or EQT?	Horizonal or vertical?	

[00113] In some embodiments, direction of BT or EQT is defined to be parallel or perpendicular to the current split direction.

[00114] In some embodiments, one flag may be firstly signaled to indicate whether QT or EQT or non-(EQT and QT) is used. If non-(EQT and QT) is selected, BT splitting information may be

further signaled.

[00115] Example 4: The flag to indicate whether EQT or BT is used may be context coded, and the context is dependent on the depth information of both current block's and its neighboring blocks

[00116] In one example, the neighboring blocks may be defined as the above and left blocks relative to the current block.

[00117] In one example, both the quad-tree depth and BT/EQT depth may be utilized in the coding of the flag.

[00118] One variable D_{ctx} can be derived for each block based on its depth information, e.g., it is set to (2*QT depth + BT/EQT depth). In some embodiments, (2*QT depth + BT/EQT depth) may be further quantized before being used for context selection.

[00119] Three contexts may be utilized for coding this flag. In one example, the context index is defined as: ((D_{ctx} of above block > D_{ctx} of current block)? 1: 0) + ((D_{ctx} of left block > D_{ctx} of current block)? 1: 0). In some embodiments, when a neighboring block is not available, its associated D_{ctx} is set to 0.

[00120] Example 5: In some embodiments, whether to and how to apply EQT split may depend on the width and height (denoted as W and H) of the block to be split.

[00121] In one example, no EQT splits are allowed when W>=T1 and H>=T2, where T1 and T2 are predefined integers e.g. T1=T2= 128 or T1=T2=64. Alternatively, T1 and/or T2 can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00122] In one example, no EQT splits are allowed when W>=T1 or H>=T2, where T1 and T2 are predefined integers e.g. T1=T2= 128 or T1=T2=64. Alternatively, T1 and/or T2 can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00123] In one example, no EQT splits are allowed when W<=T1 and H<=T2, where T1 and T2 are predefined integers e.g. T1=T2= 8 or T1=T2=16. Alternatively, T1 and/or T2 can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00124] In one example, no EQT splits are allowed when W<=T1 or H<=T2, where T1 and T2 are predefined integers e.g. T1=T2= 8 or T1=T2=16. Alternatively, T1 and/or T2 can be signaled

from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00125] In one example, horizontal EQT as shown in FIG. 11A is not allowed when W>=T, where T is a predefined integer e.g. T= 128 or T= 64. Alternatively, T can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00126] In one example, horizontal EQT as shown in FIG. 11A is not allowed when H>=T, where T is a predefined integer e.g. T= 128 or T= 64. Alternatively, T can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00127] In one example, horizontal EQT as shown in FIG. 11A is not allowed when W<=T, where T is a predefined integer e.g. T= 8 or T= 16. Alternatively, T can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00128] In one example, horizontal EQT as shown in FIG. 11A is not allowed when H<=T, where T is a predefined integer e.g. T= 8 or T= 16. Alternatively, T can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00129] In one example, vertical EQT as shown in FIG. 11B is not allowed when W>=T, where T is a predefined integer e.g. T= 128 or T= 64. Alternatively, T can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00130] In one example, vertical EQT as shown in FIG. 11B is not allowed when H>=T, where T is a predefined integer e.g. T= 128 or T= 64. Alternatively, T can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00131] In one example, vertical EQT as shown in FIG. 11B is not allowed when W<=T, where T is a predefined integer e.g. T= 8 or T= 16. Alternatively, T can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00132] In one example, vertical EQT as shown in FIG. 11B is not allowed when H<=T, where T is a predefined integer e.g. T= 8 or T= 16. Alternatively, T can be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile/CTU.

[00133] In one example, when any one of the four sub-block due to EQT split has a width or height equal to K and K×K transform is not supported/defined in the codec, EQT splitting is not allowed.

[00134] In some embodiments, whether to and how to apply EQT split may depend on the width and/or height (denoted as W and H) of the sub-block due to EQT splitting of one block.

[00135] Example 6: In some embodiments, whether to and how to apply EQT split may depend on the position of the block to be split.

[00136] In one example, whether to and how to apply EQT split may depend on whether the current block to be split is at the border of the picture or not. In one example, suppose (x,y) is the coordinate of the top-left position of the current block, (PW, PH) are the width and height of the picture, (W, H) are the width and height of a block with the current QT depth and BT/EQT depth. Then the current block is at the bottom border when y+H>PH; the current block is at the right border when x+W>PW; and the current block is at the bottom-right corner border when y+H>PH and x+W>PW.

[00137] In one example, no EQT splits are allowed when the current block is at the bottom border.

[00138] In one example, no EQT splits are allowed when the current block is at the right border.

[00139] In one example, no EQT splits are allowed when the current block is at the bottom-right corner border.

[00140] In one example, horizontal EQT as shown in FIG. 11A is not allowed when the current block is at the bottom border.

[00141] In one example, horizontal EQT as shown in FIG. 11A is not allowed when the current block is at the right border.

[00142] In one example, horizontal EQT as shown in FIG. 11A is not allowed when the current block is at the bottom-right corner border.

[00143] In one example, vertical EQT as shown in FIG. 11B is not allowed when the current block is at the bottom border.

[00144] In one example, vertical EQT as shown in FIG. 11B is not allowed when the current block is at the right border.

[00145] In one example, vertical EQT as shown in FIG. 11B is not allowed when the current block is at the bottom-right corner border.

[00146] In one example, horizontal EQT and horizontal BT may be allowed when the current block is at the bottom border.

[00147] In one example, vertical EQT and vertical BT may be allowed when the current block is at the right border.

[00148] Example 7: When one or some kinds of EQT is not allowed, the followings may apply.

[00149] In one example, the parsing procedure depends on whether one or some kinds of EQT is not allowed. The corresponding syntax elements related to EQT are not signaled if one or some kinds of EQT is not allowed.

[00150] In another example, the parsing procedure does not depend on whether one or some kinds of EQT is not allowed. The corresponding syntax elements related to EQT are signaled no matter one or some kinds of EQT is allowed or not.

[00151] In one example, a conformance encoder does not signal one or some kinds of EQT if they are not allowed.

[00152] In one example, a conformance decoder can interpret EQT as some other kinds of split such as QT, BT or not split when it parses out an EQT split but that kind of EQT is not allowed.

[00153] Example 8: The maximum EQT depth may depend on an attribute associated with the current block and/or the current picture. In one example, the maximum EQT depth may depend on the distance between current picture and the reference picture, e.g., Picture Order Count (POC) difference.

[00154] In one example, the maximum EQT depth may depend on the temporal layer identifier of the current picture.

[00155] In one example, the maximum EQT depth may depend on whether the current picture is referenced by other pictures or not.

[00156] In one example, the maximum EQT depth may depend on the quantization parameter(s).

[00157] Example 9: Rate-distortion optimization (RDO) is a method of improving video quality in video compression. The method optimizes the amount of distortion (loss of video quality) against the amount of data required to encode the video. Rate-distortion cost estimation is useful for many H.264/advanced video coding (AVC) applications including rate-distortion optimization (RDO) for mode-decision and rate-control. In some embodiments, at the certain BT/EQT depth, when the best mode of current block and its neighboring blocks are both skip mode, there is no need to further check the rate-distortion cost calculation for further splitting.

[00158] In one example, when the best mode of current block and its neighboring blocks are skip mode or merge mode, there is no need to further check the rate-distortion cost calculation for further splitting.

[00159] In one example, if the best mode of the parent block is skip mode, there is no need to further check the rate-distortion cost calculation for further splitting.

[00160] Example 10: In some embodiments, the average EQT depth for EQT-split blocks of previously coded pictures/slices/tiles is recorded. When coding the current video unit, there is no need to further check the rate-distortion cost calculation for larger EQT depth as compared to the recorded average depth.

[00161] In one example, the average EQT value may be recorded for each temporal layer. In this case, for each video data to be coded, it only utilizes the recorded average value for the same temporal layer.

[00162] In one example, only the average EQT value for the first temporal layer is recorded. In this case, for each video data to be coded, it always utilizes the recorded average value for the first temporal layer.

[00163] Example 11: In some embodiments, the average size for EQT-split blocks of previously coded pictures/slices/tiles is recorded. when coding the current video unit, there is no need to further check the rate-distortion cost calculation for smaller block sizes compared to the recorded block size.

[00164] In one example, the average EQT block size may be recorded for each temporal layer. In this case, for each video data to be coded, it only utilizes the recorded average value for the same temporal layer.

[00165] In one example, only the average EQT block size for the first temporal layer is recorded. In this case, for each video data to be coded, it always utilizes the recorded average value for the first temporal layer.

[00166] Example 12: In some embodiments, whether to check an EQT splitting that has not been checked may depend on the depth of QT/BT/EQT splitting that have already been checked for the current block.

[00167] In some embodiments, whether to skip checking of the EQT splitting partitions or not may depend on the average QT/BT splitting depth of current block;

[00168] The average splitting depth is estimated after checking BT horizontal splitting;

[00169] The average splitting depth is estimated after checking BT vertical splitting;

[00170] The average splitting depth is estimated after checking BT horizontal and vertical splitting;

[00171] In one example, for estimation, block depths are collected at some specific positions coordinating with EQT splitting characteristic and then averaged. One example of 7 specific

positions is depicted in FIG. 17. In FIG. 17, all horizontal lines represent quarter splitting of the block in horizontal direction and vertical lines depict quarter splitting of blocks in vertical direction.

[00172] In one example, a threshold is calculated as a function of the estimated average splitting depth and the current splitting depth.

[00173] In some embodiments, in addition, a threshold table is applied to find the threshold. The estimated average splitting depth and the current splitting depth is used as the key to fetch a corresponding threshold stored in the table.

[00174] When the average splitting depth is smaller than the threshold, there is no need to further check the EQT splitting partitions.

[00175] In one example, the above method is applicable to certain slice/picture type, such as I slices or I pictures.

[00176] **5. Embodiment examples**

Syntax changes on top of existing design are in **bold**.

coding_quadtree (x0, y0, uiAbsPartIdx, uiDepth, uiWidth, uiHeight) {	
CUPelX = x0	
CUPelY = y0	
uiLPelX = CUPelX + g_auiRasterToPelX[g_auiZscanToRaster[uiAbsPartIdx]]	
uiRPelX = uiLPelX + uiWidth – 1	
uiTPelY = CUPelY + g_auiRasterToPelY[g_auiZscanToRaster[uiAbsPartIdx]]	
uiBPelY = uiTPelY + uiHeight – 1	
uiQTWidth = uiCTUSize >> uiDepth	
uiQTHeight = uiCTUSize >> uiDepth	
uiBTDepth=g_aucConvertToBit[uiQTWidth]-g_aucConvertToBit[uiWidth]+g_aucConvertToBit[uiQTHeight] - g_aucConvertToBit[uiHeight]	

if (uiCTUSize >> uiDepth == uiWidth && uiWidth == uiHeight) {	
if ((uiRPelX < PicWidth) && (uiBPelY < PicHeight)) {	
split_flag	ae(v)
}	
else if (uiWidth == MinQTSize) {	
SplitFlag = 0	
}	
else{	
SplitFlag = 1	
}	
}	
if (SplitFlag == 0 && uiHeight > uiMinBTSize || uiWidth>uiMinBTSize) && uiWidth <= uiMaxBTSize && uiHeight <= uiMaxBTSize && uiBTDepth < uiMaxBTD){	
bt_split_flag	**ae(v)**
if (BtSplitFlag) {	
eqt_split_flag	**ae(v)**
if (EqtSplitFlag) {	
eqt_split_dir	**ae(v)**
BTSplitMode = (EqtSplitDir == 0) ? 3: 4	
}	
else {	
bt_split_dir	**ae(v)**
BTSplitMode = (BtSplitDir == 0) ? 1:2;	

}	
else {	
BtSplitFlag = 0;	
BTSplitMode = 0;	
}	
}	
if (SplitFlag) {	
uiQNumParts = (NumPartInCU >> (uiDepth << 1)) >> 2	
for (UInt uiPartUnitIdx = 0; uiPartUnitIdx < 4; uiPartUnitIdx++, uiAbsPartIdx += uiQNumParts) {	
uiLPelX = CUPelX + g_auiRasterToPelX[g_auiZscanToRaster[uiAbsPartIdx]]	
uiTPelY = CUPelY + g_auiRasterToPelY[g_auiZscanToRaster[uiAbsPartIdx]]	
if ((uiLPelX < PicWidth) && (uiTPelY < PicHeight)) {	
coding_quadtree(x0, y0, uiAbsPartIdx, uiDepth + 1, uiWidth >> 1, uiHeight >>1)	
}	
}	
}	
else if (BTSplitMode == 1) {	
for (UInt uiPartUnitIdx = 0; uiPartUnitIdx < 2; uiPartUnitIdx++) {	
if (uiPartUnitIdx == 1) {	

uiAbsPartIdx = g_auiRasterToZscan[g_auiZscanToRaster[uiAbsPartIdx] + (uiHeight>>1) / MinCUHeight * NumPartInCtuWidth]	
}	
coding_quadtree(x0, y0, uiAbsPartIdx,uiDepth, uiWidth, uiHeight >> 1)	
}	
}	
else if (BTSplitMode == 2) {	
for (UInt uiPartUnitIdx = 0; uiPartUnitIdx < 2; uiPartUnitIdx++) {	
if (uiPartUnitIdx == 1) {	
uiAbsPartIdx = g_auiRasterToZscan[g_auiZscanToRaster[uiAbsPartIdx] + (uiWidth >> 1) / MinCUWidth]	
}	
coding_quadtree(x0, y0, uiAbsPartIdx, uiDepth, uiWidth >> 1, uiHeight)	
}	
}	
else if (BTSplitMode == 3) { /*notes: related to EQT horizontal split*/	
UInt orgAbsPartIdx = uiAbsPartIdx	
UInt uiSubWidth = 0	
UInt uiSubHeight = 0	
for (UInt uiPartUnitIdx = 0; uiPartUnitIdx < 4; uiPartUnitIdx++) {	
if (uiPartUnitIdx == 0 || uiPartUnitIdx == 3) {	

uiSubWidth = uiWidth	
uiSubHeight = uiHeight >> 2	
uiAbsPartIdx = orgAbsPartIdx	
uiAbsPartIdx = g_auiRasterToZscan[g_auiZscanToRaster[uiAbsPartIdx] + ((uiHeight >> 2) * uiPartUnitIdx) / MinCUHeight * NumPartInCtuWidth]	
}	
else if (uiPartUnitIdx == 1) {	
uiSubWidth = uiWidth >> 1	
uiSubHeight = uiHeight >> 1	
uiAbsPartIdx = orgAbsPartIdx	
uiAbsPartIdx = g_auiRasterToZscan[g_auiZscanToRaster[uiAbsPartIdx] + (uiHeight >> 2) / MinCUHeight * NumPartInCtuWidth]	
}	
else if (uiPartUnitIdx == 2) {	
uiSubWidth = uiWidth >> 1	
uiSubHeight = uiHeight >> 1	
uiAbsPartIdx = g_auiRasterToZscan[g_auiZscanToRaster[uiAbsPartIdx] + (uiWidth >> 1) / MinCUWidth]	
}	
coding_quadtree(x0, y0, uiAbsPartIdx, uiDepth, uiSubWidth, uiSubHeight)	
}	
}	

else if (BTSplitMode == 4) { /*notes: related to EQT Vertical split*/	
UInt orgAbsPartIdx = uiAbsPartIdx	
UInt uiSubWidth = 0	
UInt uiSubHeight = 0	
for (UInt uiPartUnitIdx = 0; uiPartUnitIdx < 4; uiPartUnitIdx++) {	
if (uiPartUnitIdx == 0 || uiPartUnitIdx == 3) {	
uiSubWidth = uiWidth >> 2	
uiSubHeight = uiHeight	
uiAbsPartIdx = orgAbsPartIdx	
uiAbsPartIdx = g_auiRasterToZscan[g_auiZscanToRaster[uiAbsPartIdx] + ((uiWidth >> 2) * uiPartUnitIdx) / MinCUWidth]	
}	
else if (uiPartUnitIdx == 1) {	
uiSubWidth = uiWidth >> 1	
uiSubHeight = uiHeight >> 1	
uiAbsPartIdx = orgAbsPartIdx	
uiAbsPartIdx = g_auiRasterToZscan[g_auiZscanToRaster[uiAbsPartIdx] + (uiWidth >> 2) / MinCUWidth]	
}	
else if (uiPartUnitIdx == 2) {	
uiSubWidth = uiWidth >> 1	
uiSubHeight = uiHeight >> 1	

uiAbsPartIdx = g_auiRasterToZscan[g_auiZscanToRaster[uiAbsPartIdx] + (uiHeight >> 1) / MinCUHeight * NumPartInCtuWidth]	
}	
coding_quadtree(x0, y0, uiAbsPartIdx, uiDepth, uiSubWidth, uiSubHeight)	
}	
}	
else {	
coding_quadtree (x0, y0, uiAbsPartIdx, uiDepth, uiWidth, uiHeight)	
}	
}	

[00177] Example of Semantics

[00178] eqt_split_flag

[00179] - a flag to indicate whether EQT is enabled or disabled forone block

[00180] eqt_split_dir

[00181] -a flag to indicate whether horizontal EQT is used or vertical EQT is used. FIG. 13A and 13B show example of quadtree partitioning for horizotnal EQT partitionss and vertical EQT partitions.

[00182] FIG. 14 is a block diagram illustrating an example of the architecture for a computer system or other control device 1400 that can be utilized to implement various portions of the presently disclosed technology. In FIG. 14, the computer system 1400 includes one or more processors 1405 and memory 1410 connected via an interconnect 1425. The interconnect 1425 may represent any one or more separate physical buses, point to point connections, or both, connected by appropriate bridges, adapters, or controllers. The interconnect 1425, therefore, may include, for example, a system bus, a Peripheral Component Interconnect (PCI) bus, a HyperTransport or industry standard architecture (ISA) bus, a small computer system interface

(SCSI) bus, a universal serial bus (USB), IIC (I2C) bus, or an Institute of Electrical and Electronics Engineers (IEEE) standard 674 bus, sometimes referred to as "Firewire."

[00183] The processor(s) 1405 may include central processing units (CPUs) to control the overall operation of, for example, the host computer. In certain embodiments, the processor(s) 1405 accomplish this by executing software or firmware stored in memory 1410. The processor(s) 1405 may be, or may include, one or more programmable general-purpose or special-purpose microprocessors, digital signal processors (DSPs), programmable controllers, application specific integrated circuits (ASICs), programmable logic devices (PLDs), or the like, or a combination of such devices.

[00184] The memory 1410 can be or include the main memory of the computer system. The memory 1410 represents any suitable form of random access memory (RAM), read-only memory (ROM), flash memory, or the like, or a combination of such devices. In use, the memory 1410 may contain, among other things, a set of machine instructions which, when executed by processor 1405, causes the processor 1405 to perform operations to implement embodiments of the presently disclosed technology.

[00185] Also connected to the processor(s) 1405 through the interconnect 1425 is a (optional) network adapter 1415. The network adapter 1415 provides the computer system 1400 with the ability to communicate with remote devices, such as the storage clients, and/or other storage servers, and may be, for example, an Ethernet adapter or Fiber Channel adapter.

[00186] FIG. 15 shows a block diagram of an example embodiment of a device 1500 that can be utilized to implement various portions of the presently disclosed technology. The mobile device 1500 can be a laptop, a smartphone, a tablet, a camcorder, or other types of devices that are capable of processing videos. The mobile device 1500 includes a processor or controller 1501 to process data, and memory 1502 in communication with the processor 1501 to store and/or buffer data. For example, the processor 1501 can include a central processing unit (CPU) or a microcontroller unit (MCU). In some implementations, the processor 1501 can include a field-programmable gate-array (FPGA). In some implementations, the mobile device 1500 includes or is in communication with a graphics processing unit (GPU), video processing unit (VPU) and/or wireless communications unit for various visual and/or communications data processing functions of the smartphone device. For example, the memory 1502 can include and store processor-executable code, which when executed by the processor 1501, configures the

mobile device 1500 to perform various operations, e.g., such as receiving information, commands, and/or data, processing information and data, and transmitting or providing processed information/data to another device, such as an actuator or external display. To support various functions of the mobile device 1500, the memory 1502 can store information and data, such as instructions, software, values, images, and other data processed or referenced by the processor 1501. For example, various types of Random Access Memory (RAM) devices, Read Only Memory (ROM) devices, Flash Memory devices, and other suitable storage media can be used to implement storage functions of the memory 1502. In some implementations, the mobile device 1500 includes an input/output (I/O) unit 1503 to interface the processor 1501 and/or memory 1502 to other modules, units or devices. For example, the I/O unit 1503 can interface the processor 1501 and memory 1502 with to utilize various types of wireless interfaces compatible with typical data communication standards, e.g., such as between the one or more computers in the cloud and the user device. In some implementations, the mobile device 1500 can interface with other devices using a wired connection via the I/O unit 1503. The mobile device 1500 can also interface with other external interfaces, such as data storage, and/or visual or audio display devices 1504, to retrieve and transfer data and information that can be processed by the processor, stored in the memory, or exhibited on an output unit of a display device 1504 or an external device. For example, the display device 1504 can display a video frame in accordance with the disclosed technology.

[00187] FIG. 16 is a flowchart of a method 1600 of visual media processing. The method 1600 includes performing (1602), using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and wherein the rule specifies that in case that the rule is used for partitioning the current block, then each subblock is further split into a binary tree (BT) partitioning or another EQT partitioning, and both BT and the another EQT partitioning have depths that meet a pre-defined relationship.

[00188] Another method of visual media processing includes performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT

partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and wherein the rule allows the EQT partitioning process for the current block based on a width or a height of the current block.

[00189] Another method of visual media processing includes performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and wherein the rule allows the EQT partitioning process for the current block based on a position of the current block.

[00190] Another method of visual media processing includes performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and wherein the rule allows a maximum depth of the EQT partitioning process to depend on a distance between a current picture of the current block and a reference picture for the current block or a quantization parameter of the current block or a temporal layer id of the current picture.

[00191] In the disclosed embodiments, the bitstream representation of a current block of video may include bits of a bitstream (compressed representation of a video) that may be non-contiguous and may depend on header information, as is known in the art of video compression. Furthermore, a current block may include samples representative of one or more of luma and chroma components, or rotational variations thereof (e.g, YCrCb or YUV, and so on).

[00192] The listing of clauses below describes some embodiments and techniques as follows.

[00193] 1. A method of visual media processing, comprising: performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the

given block; and wherein the rule specifies that in case that the rule is used for partitioning the current block, then each subblock is further split into a binary tree (BT) partitioning or another EQT partitioning, and both BT and the another EQT partitioning have depths that meet a pre-defined relationship.

[00194] 2. The method of clause 1, wherein the conversion includes generating the current block from the bitstream representation.

[00195] 3. The method of clause 1, wherein the conversion includes generating the bitstream representation from the current block.

[00196] 4. The method of any of clauses 1 to 3, wherein the EQT partitioning process partitions the current block into one of only two possible partitionings.

[00197] 5. The method of clause 4, wherein the current block comprises MxN pixels, where M and N are integers and wherein the two possible partitions include a first partitioning comprising an M x N/4 top portion, followed by two side-by-side M/2 x N/2 middle portions, followed by an M x N/4 bottom portions, or a second partitioning comprising an M/4 x N left portion, two M/2 x N/2 middle portions and one M/4 x N right portion.

[00198] 6. The method of clause 1, wherein the pre-defined relationship specifies that a BT and EQT partitions have different values or the pre-defined relationship specifies that the depth of EQT partitions is equal to a sum of the depth of the BT partitions and quadtree (QT) partitions.

[00199] 7. The method of clause 1, wherein the pre-defined relationship specifies that a BT and EQT partitions have a same value.

[00200] 8. The method of clause 1, wherein the rule specifies that in case that the current block is partitioned using BT, each partition is partitioned using one of BT or EQT partitioning.

[00201] 9. The method of clause 1, wherein the rule specifies that, in case that the current block is partitioned using EQT, each resulting sub-block has a depth value two more than that of the current block.

[00202] 10. The method of any of clauses 1 to 9, wherein the rule further specifies to use a same allowed root node size for all blocks in a picture tile or a slice or a picture or a sequence of pictures as that used for binary tree partitioning.

[00203] 11. The method of any of clauses 1 to 9, wherein the rule further specifies to use a different allowed root node size for all blocks in a picture tile or a slice or a picture or a sequence of pictures as that used for binary tree partitioning.

[00204] 12. The method of any of clauses 1 to 9, wherein the bitstream representation is configured to indicate a maximum allowed root node size for the EQT partitioning process at a video level, or sequence level or picture level or picture header level or slice header level or tile group header level or tile level or coding tree unit level.

[00205] 13. The method of any of clauses 1 to 11, wherein the bitstream representation is configured to include a first field indicative of partitioning of the current block between EQT partitioning or BT partitioning and a second field indicative of a splitting direction for the current block between horizontal and vertical directions.

[00206] 14. The method of clause 11 to 14, wherein the splitting direction is relative to a split direction of a previous block.

[00207] 15. The method of any of clauses 11 to 14, wherein the first field or the second field are context coded depending on a depth information of one or more neighboring blocks or a depth information of a current block.

[00208] 16. The method of clause 15, wherein the neighboring block is an above block or a left block relative to the current block.

[00209] 17. The method of any of clauses 15 and 16, wherein a quantized value of the depth information of the one or more neighboring blocks or the depth information of the current block is used for the context coding.

[00210] 18. A method of visual media processing, comprising: performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block; and wherein the rule allows the EQT partitioning process for the current block based on a width or a height of the current block.

[00211] 19. The method of clause 18, wherein the conversion includes generating the current block from the bitstream representation.

[00212] 20. The method of clause 18, wherein the conversion includes generating the bitstream representation from the current block.

[00213] 21. The method of any of clauses 18 to 20, wherein the rule disallows the EQT partitioning when the width is greater than or equal to T1 or the height is greater than or equal to

T2, wherein T1 and T2 are integers.

[00214] 22. The method of clause 21, wherein T1 and T2 are pre-defined.

[00215] 23. The method of clause 21, wherein the bitstream representation is configured to carry an indication of T1 and T2.

[00216] 24. The method of clause 22, wherein the indication of T1 and T2 is indicated at a video level or a sequence level or a picture level or a slice header level or a tile group header level or a tile level or a coding tree unit level.

[00217] 25. The method of any of clauses 18 to 20, wherein the rule disallows the EQT partitioning when the width is less than or equal to T1 or the height is less than or equal to T2, wherein T1 and T2 are integers.

[00218] 26. The method of any of clauses 18 to 20, wherein the rule disallows the EQT partitioning when the width is greater than or equal to the height.

[00219] 27. The method of any of clauses 18 to 20, wherein the rule disallows the EQT partitioning when the width is less than the height.

[00220] 28. A method of visual media processing, comprising: performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block; and wherein the rule allows the EQT partitioning process for the current block based on a position of the current block.

[00221] 29. The method of clause 28, wherein the conversion includes generating the current block from the bitstream representation.

[00222] 30. The method of clause 28, wherein the conversion includes generating the bitstream representation from the current block.

[00223] 31. The method of any of clauses 28 to 30, wherein the rule disallows the EQT partitioning process for the current block that is at a bottom border of a video region.

[00224] 32. The method of any of clauses 28 to 30, wherein the rule disallows the EQT partitioning process for the current block that is at a right border of a video region.

[00225] 33. The method of any of clauses 28 to 30, wherein the rule disallows the EQT partitioning process for the current block that is a corner block of a video region.

[00226] 34. The method of clause 33, wherein the corner corresponds to a bottom right corner of the video region.

[00227] 35. The method of clause 28, wherein the rule allows use of a horizontal EQT partitioning or a horizontal binary tree partitioning the current block that is at a bottom border of a video region.

[00228] 36. The method of clause 28, wherein the rule allows use of a horizontal EQT partitioning or a horizontal binary tree partitioning the current block that is at a right border of the a video region.

[00229] 37. The method of any of clauses 1 to 34, wherein in case that the rule disallows the EQT partitioning process for the current block, then corresponding syntax elements are omitted from the bitstream representation.

[00230] 38. The method of any of clauses 1 to 33, wherein in case that the rule disallows the EQT partitioning process for the current block, then corresponding syntax elements are included with a default value in the bitstream representation

[00231] 39. A method of visual media processing, comprising: performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block; and wherein the rule specifies (1) a maximum depth of the EQT partitioning process to depend on a distance between a current picture of the current block and a reference picture for the current block or a quantization parameter of the current block or a temporal layer id of the current picture, or (2) using splitting depth of the current block of visual media data in deciding to check for additional EQT partitioning.

[00232] 40. The method of clause 39, wherein the conversion includes generating the current block from the bitstream representation.

[00233] 41. The method of clause 39, wherein the conversion includes generating the bitstream representation from the current block.

[00234] 42. The method of any of clauses 1 to 40, wherein the rule specifies to disable the EQT partitioning process in case that the current block and neighboring blocks are to be encoded using skip mode or in case that a coding depth of the current block is above an average coding depth of

previously coded blocks.

[00235] 43. The method of clause 42, wherein the average coding depth is calculated over previously encoded picture or a slice or a tile in which the current block is positioned.

[00236] 44. The method of clause 42, wherein the average coding depth is calculated for a temporal layer in which the current block is positioned.

[00237] 45. The method of clause 39, wherein the rule specifies to skip checking for EQT partitioning in case that an average splitting depth of the current block meets a condition.

[00238] 46. The method of clause 45, wherein the average splitting depth is estimated after checking a binary tree horizontal splitting.

[00239] 47. The method of any of clauses 45-46, wherein the average splitting depth is estimated after checking a binary tree vertical splitting.

[00240] 78. The method of any of clauses 45-47, wherein the average splitting depth is determined at a number of specific positions of pixels in the current block.

[00241] 49. The method of clause 48, wherein the number is equal to 7.

[00242] 50. The method any of clauses 45-49, wherein the condition includes comparing the average splitting depth with a threshold.

[00243] 51. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1 to 50.

[00244] 52. The apparatus of clause 51, wherein the apparatus is a video encoder.

[00245] 53. The apparatus of clause 51, wherein the apparatus is a video decoder.

[00246] 54. A computer readable media, which includes a program comprising code for a processor to carry out a method recited in any one or more of clauses 1 to 50.

[00247] With respect to the above listed clauses and the list of techniques in section 4, the partitioning techniques may be specified using parameter sets (picture or video parameter sets) or pre-specified based on a rule. Accordingly, the number of bits needed to signal partitioning of blocks may be reduced. Similarly, the partitioning decision may also be simplified due to the various rules specified in this document, thereby allowing for lower complexity implementations of encoders or decoders.

[00248] Furthermore, the position dependency of the partitioning rule may be based on a video region in which the current block is present (e.g., clause 26). The video region may include the current block or a larger portion such as a tile, or a slice or a picture in which the current block is

present.

[00249] FIG. 18 is a block diagram showing an example video processing system 1800 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 1800. The system 1800 may include input 1802 for receiving video content. The video content may be received in a raw or uncompressed format, e.g., 8 or 10 bit multi-component pixel values, or may be in a compressed or encoded format. The input 1802 may represent a network interface, a peripheral bus interface, or a storage interface. Examples of network interface include wired interfaces such as Ethernet, passive optical network (PON), etc. and wireless interfaces such as Wi-Fi or cellular interfaces.

[00250] The system 1800 may include a coding component 1804 that may implement the various coding or encoding methods described in the present document. The coding component 1804 may reduce the average bitrate of video from the input 1802 to the output of the coding component 1804 to produce a coded representation of the video. The coding techniques are therefore sometimes called video compression or video transcoding techniques. The output of the coding component 1804 may be either stored, or transmitted via a communication connected, as represented by the component 1806. The stored or communicated bitstream (or coded) representation of the video received at the input 1802 may be used by the component 1808 for generating pixel values or displayable video that is sent to a display interface 1810. The process of generating user-viewable video from the bitstream representation is sometimes called video decompression. Furthermore, while certain video processing operations are referred to as "coding" operations or tools, it will be appreciated that the coding tools or operations are used at an encoder and corresponding decoding tools or operations that reverse the results of the coding will be performed by a decoder.

[00251] Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or Displayport, and so on. Examples of storage interfaces include SATA (serial advanced technology attachment), PCI, IDE interface, and the like. The techniques described in the present document may be embodied in various electronic devices such as mobile phones, laptops, smartphones or other devices that are capable of performing digital data processing and/or video display.

[00252] FIG. 19 is a flowchart representation of a method 1900 for video processing in accordance with the present disclosure. The method 1900 includes, at operation 1902,

determining, for a conversion between a current block of a video and a bitstream representation of the video, whether an extended quadtree (EQT) partitioning process is applicable to the current block based on a rule. The EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block. The rule specifies a maximum depth of the EQT partitioning process based on an attribute associated with the current block. The method 1900 includes, at operation 1904, performing the conversion based on the determining. The conversion includes generating the current block from the bitstream representation. The conversion also includes generating the bitstream representation from the current block.

[00253] In some embodiments, the attribute comprises a distance between a current picture of the current block and a reference picture for the current block. In some embodiments, the attribute comprises a difference of a Picture Order Count (POC). In some embodiments, the attribute comprises a temporal layer identifier of a current picture of the current block. In some embodiments, the attribute comprises whether a current picture of the current block is referenced by other pictures of the video. In some embodiments, the attribute comprises a quantization parameter of the current block.

[00254] In some embodiments, the rule further specifies, at a predefined coding depth of the current block, to disable a subsequent partitioning process in case that the current block and neighboring blocks are coded using a skip mode. In some embodiments, the rule further specifies to disable a subsequent partitioning process in case that the current block and neighboring blocks are coded using a merge mode. In some embodiments, the rule further specifies to disable a subsequent partitioning process in case a parent block of the current block is coded using a skip mode.

[00255] In some embodiments, the rule further specifies to disable the EQT partitioning process in case a coding depth of the current block is larger than an average coding depth of previously coded blocks. The average coding depth of the previously coded blocks can be calculated based on a binary tree partitioning process, a quadtree partitioning process, or the EQT partitioning process. In some embodiments, a threshold representing a relationship between the coding depth of the current block and the average coding depth of the previously coded blocks is compared against a table of threshold values to determine whether the EQT partitioning process is disabled. In some embodiments, the average coding depth is calculated over

previously encoded pictures, slices, or tiles. In some embodiments, the average coding depth is calculated for a temporal layer in which the current block is positioned. In some embodiments, the average coding depth is calculated only for a first temporal layer of the video. In some embodiments, the average coding depth is calculated based on a binary tree horizontal splitting. In some embodiments, the average coding depth is calculated based on a binary tree vertical splitting. In some embodiments, the average coding depth is determined at a number of specific positions in the current block. In some embodiments, the number of specific positions is equal to 7.

[00256] In some embodiments, the rule further specifies to disable the EQT partitioning process in case a size of the currently block is smaller than an average size of previously coded blocks. In some embodiments, the average size is calculated for blocks of previously encoded pictures, slices, or tiles. In some embodiments, the average size is calculated for blocks of a temporal layer in which the current block is positioned. In some embodiments, the average size is calculated only for blocks of a first temporal layer of the video.

[00257] Some embodiments of the disclosed technology include making a decision or determination to enable a video processing tool or mode. In an example, when the video processing tool or mode is enabled, the encoder will use or implement the tool or mode in the processing of a block of video, but may not necessarily modify the resulting bitstream based on the usage of the tool or mode. That is, a conversion from the block of video to the bitstream representation of the video will use the video processing tool or mode when it is enabled based on the decision or determination. In another example, when the video processing tool or mode is enabled, the decoder will process the bitstream with the knowledge that the bitstream has been modified based on the video processing tool or mode. That is, a conversion from the bitstream representation of the video to the block of video will be performed using the video processing tool or mode that was enabled based on the decision or determination.

[00258] Some embodiments of the disclosed technology include making a decision or determination to disable a video processing tool or mode. In an example, when the video processing tool or mode is disabled, the encoder will not use the tool or mode in the conversion of the block of video to the bitstream representation of the video. In another example, when the video processing tool or mode is disabled, the decoder will process the bitstream with the knowledge that the bitstream has not been modified using the video processing tool or mode that

was enabled based on the decision or determination.

[00259] The disclosed and other embodiments, modules and the functional operations described in this document can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this document and their structural equivalents, or in combinations of one or more of them. The disclosed and other embodiments can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more them. The term "data processing apparatus" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[00260] 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.

[00261] The processes and logic flows described in this document can be performed by one or

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

[00262] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random-access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00263] 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.

[00264] 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.

[00265] 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 processing, comprising:

determining, for a conversion between a current block of a video and a bitstream representation of the video, whether an extended quadtree (EQT) partitioning process is applicable to the current block based on a rule, wherein the EQT partitioning process includes partitioning a given block into exactly four sub-blocks, at least one of which has a size different from half of width of the given block times half of a height of the given block, and wherein the rule specifies a maximum depth of the EQT partitioning process based on an attribute associated with the current block; and

performing the conversion based on the determining.

2. The method of claim 1, wherein the conversion includes generating the current block from the bitstream representation.

3. The method of claim 1, wherein the conversion includes generating the bitstream representation from the current block.

4. The method of any one of claims 1 to 3, wherein the attribute comprises a distance between a current picture of the current block and a reference picture for the current block.

5. The method of any one of claims 1 to 4, wherein the attribute comprises a difference of a Picture Order Count (POC).

6. The method of any one of claims 1 to 3, wherein the attribute comprises a temporal layer identifier of a current picture of the current block.

7. The method of any one of claims 1 to 3, wherein the attribute comprises whether a current picture of the current block is referenced by other pictures of the video.

8. The method of any one of claims 1 to 3, wherein the attribute comprises a quantization parameter of the current block.

9. The method of any one of claims 1 to 8, wherein the rule further specifies, at a predefined coding depth of the current block, to disable a subsequent partitioning process in case that the current block and neighboring blocks are coded using a skip mode.

10. The method of any one of claims 1 to 8, wherein the rule further specifies to disable a subsequent partitioning process in case that the current block and neighboring blocks are coded using a merge mode.

11. The method of any one of claims 1 to 8, wherein the rule further specifies to disable a subsequent partitioning process in case a parent block of the current block is coded using a skip mode.

12. The method of any one of claims 1 to 8, wherein the rule further specifies to disable the EQT partitioning process in case a coding depth of the current block is larger than an average coding depth of previously coded blocks.

13. The method of claim 12, wherein the average coding depth of the previously coded blocks is calculated based on a binary tree partitioning process, a quadtree partitioning process, or the EQT partitioning process that is performed on the previously coded blocks.

14. The method of claim 12 or 13, wherein a threshold representing a relationship between the coding depth of the current block and the average coding depth of the previously coded blocks is compared against a table of threshold values to determine whether the EQT partitioning process is disabled.

15. The method of any one of claims 12 to 14, wherein the average coding depth is calculated over previously encoded pictures, slices, or tiles.

16. The method of any one of claims 12 to 15, wherein the average coding depth is calculated for a temporal layer in which the current block is positioned.

17. The method of any one of claims 12 to 15, wherein the average coding depth is calculated only for a first temporal layer of the video.

18. The method of any one of claims 12 to 17, wherein the average coding depth is calculated based on a binary tree horizontal splitting.

19. The method of any one of claims 12 to 18, wherein the average coding depth is calculated based on a binary tree vertical splitting.

20. The method of any one of claims 12 to 19, wherein the average coding depth is determined at a number of specific positions in the current block.

21. The method of claim 20, wherein the number of specific positions is equal to 7.

22. The method of any one of claims 1 to 20, wherein the rule further specifies to disable the EQT partitioning process in case a size of the currently block is smaller than an average size of previously coded blocks.

23. The method of claim 22, wherein the average size is calculated for blocks of previously encoded pictures, slices, or tiles.

24. The method of claim 22 or 23, wherein the average size is calculated for blocks of a temporal layer in which the current block is positioned.

25. The method of any one of claims 22 to 24, wherein the average size is calculated only for blocks of a first temporal layer of the video.

26. A video processing apparatus comprising a processor configured to implement a method recited in any one of claims 1 to 25.

27. A computer readable media, which includes a program comprising code for a processor to carry out a method recited in any one of claims 1 to 25.

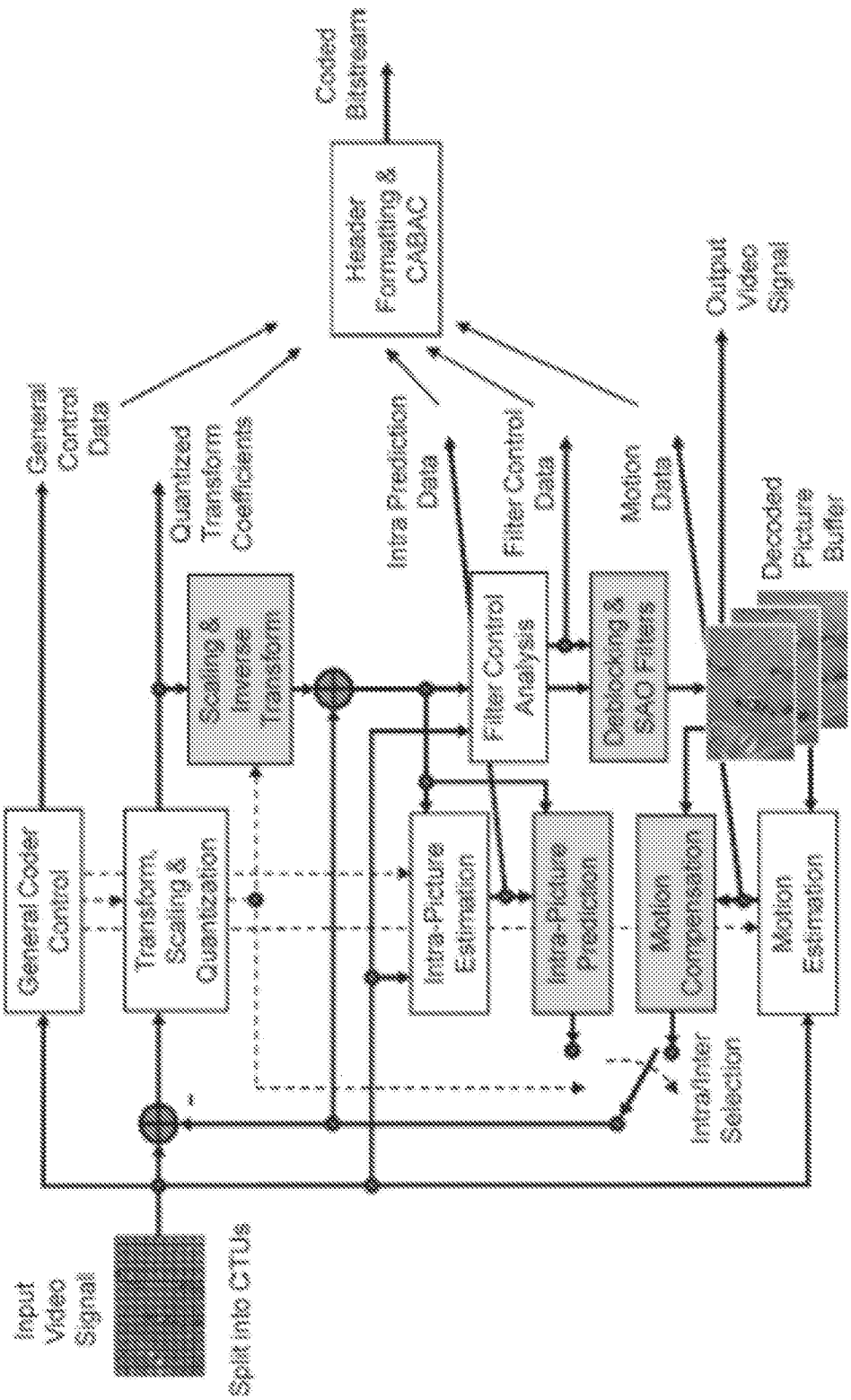

Coded Bitstream
Header Formatting & CABAC
Output Video Signal
General Control Data
Quantized Transform Coefficients
Intra Prediction Data
Filter Control Data
Motion Data
Decoded Picture Buffer
Scaling & Inverse Transform
Filter Control Analysis
Deblocking & SAO Filters
General Coder Control
Transform, Scaling & Quantization
Intra-Picture Estimation
Intra-Picture Prediction
Motion Compensation
Motion Estimation
Intra/Inter Selection
Input Video Signal
Split into CTUs

FIG. 1

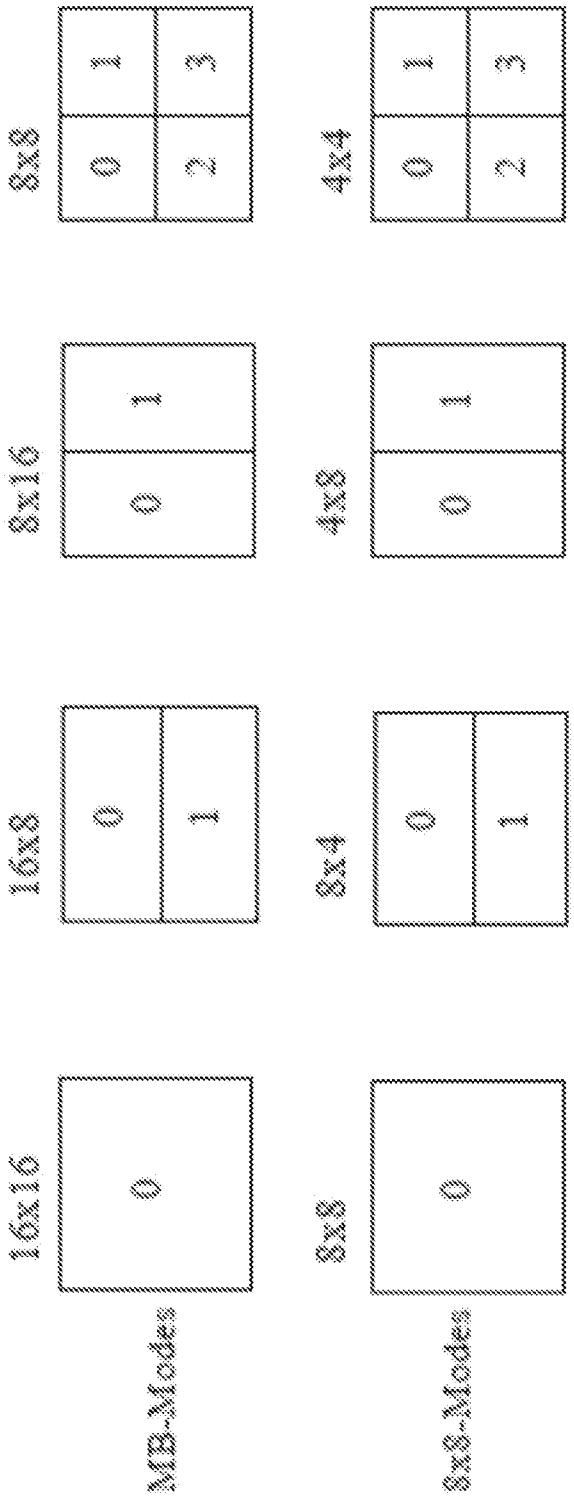

MB-Modes
16x16
0
16x8
0
1
8x16
0
1
8x8
0
1
2
3
8x8-Modes
8x8
0
8x4
0
1
4x8
0
1
4x4
0
1
2
3

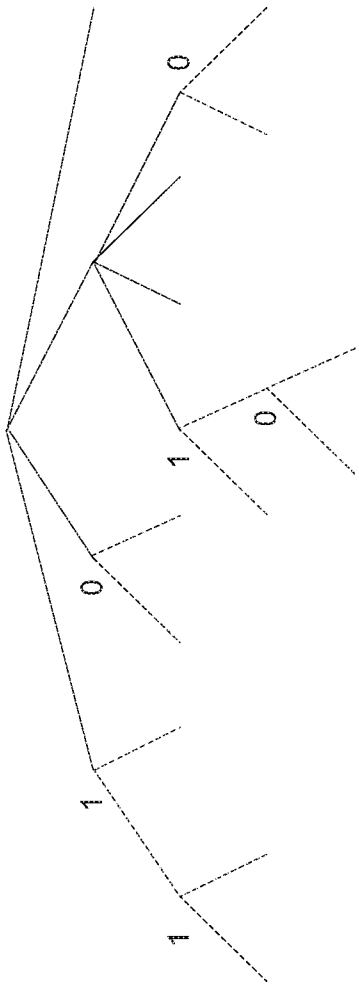
FIG. 2

Modes for splitting a CB into PBs, subject to certain size constraints. For intrapicture-predicted CBs, only $M \times M$ and $M/2 \times M/2$ are supported.

FIG. 3

Subdivision of a CTB into CBs [and transform block (TBs)]. Solid lines indicate CB boundaries and dotted lines indicate TB boundaries. (a) CTB with its partitioning. (b) Corresponding quadtree.

FIG. 4



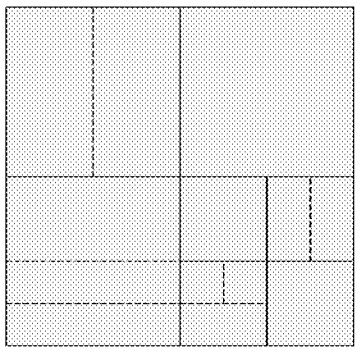


FIG. 5

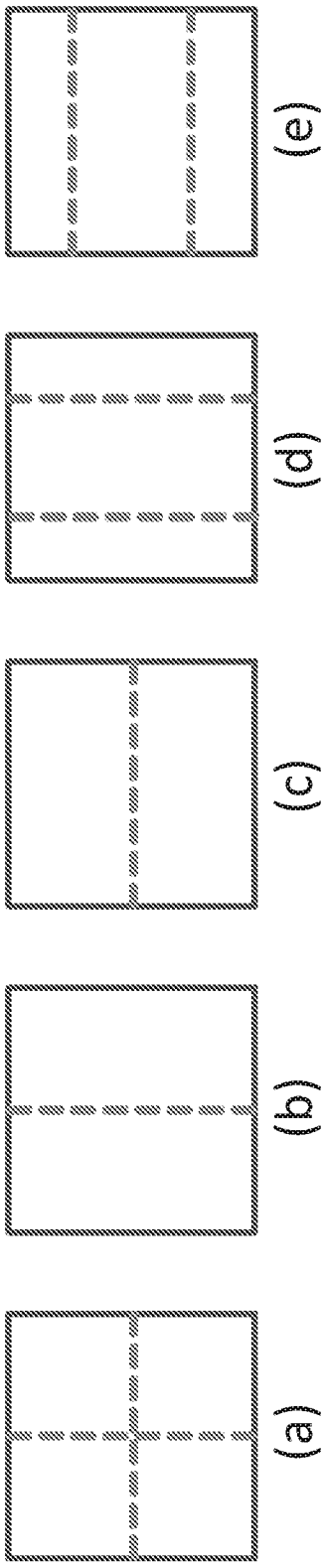


FIG. 6

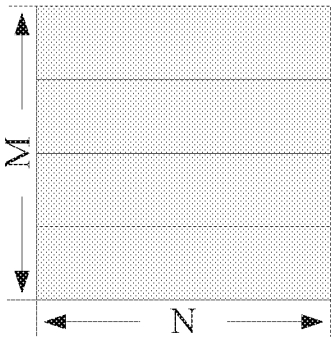


FIG. 7B

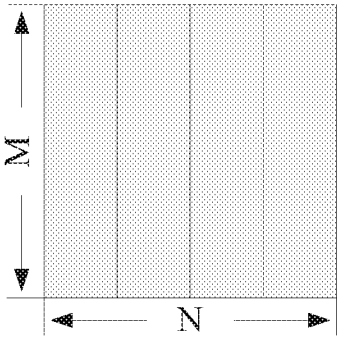


FIG. 7A

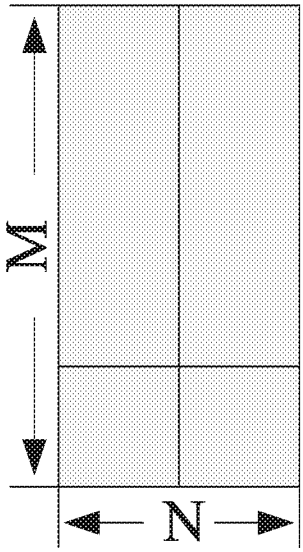


FIG. 7D

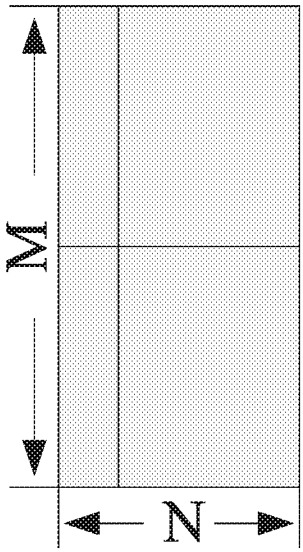


FIG. 7C

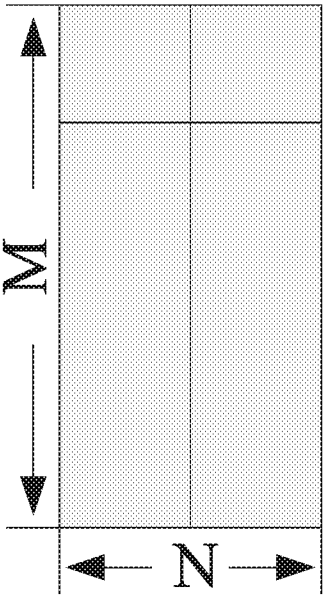


FIG. 7F

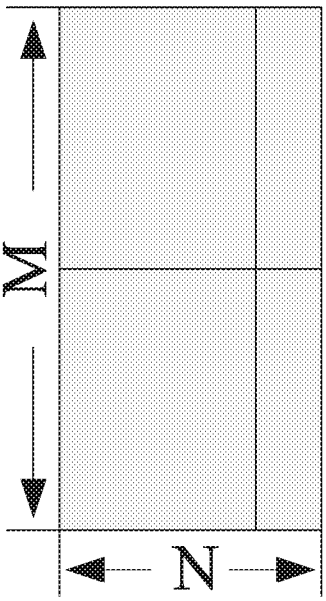
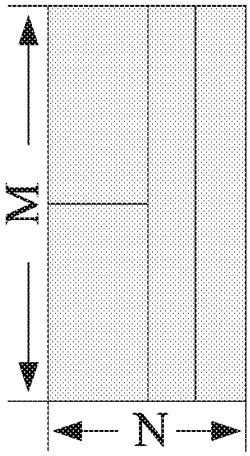
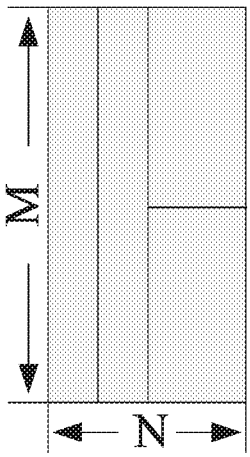


FIG. 7E





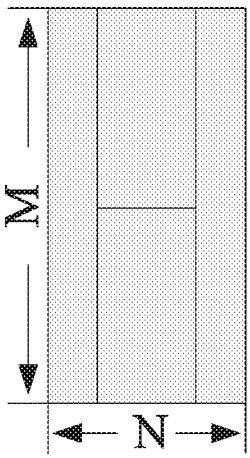


FIG. 7G

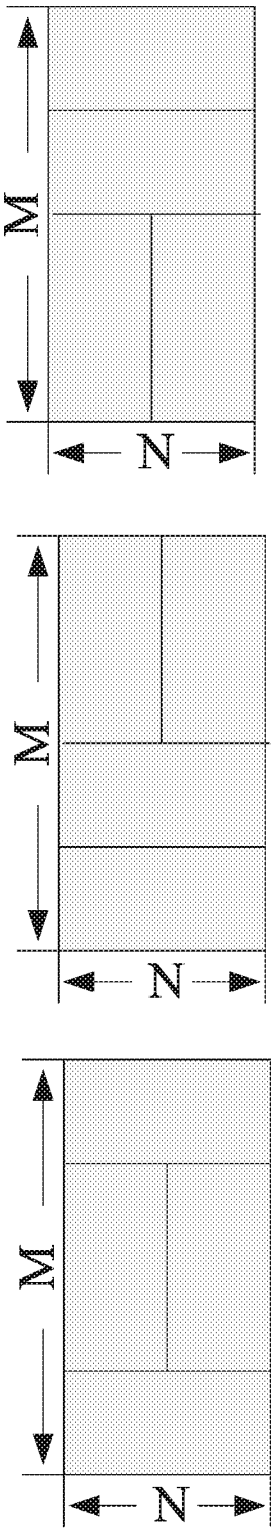


FIG. 7H

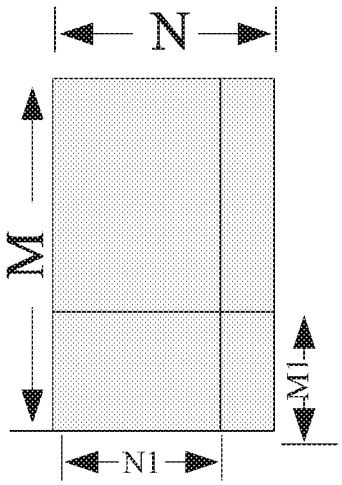


FIG. 7I

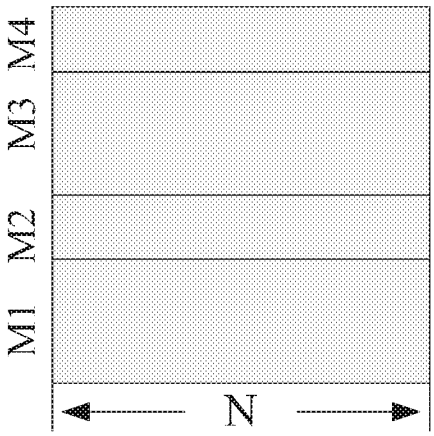


FIG. 7K

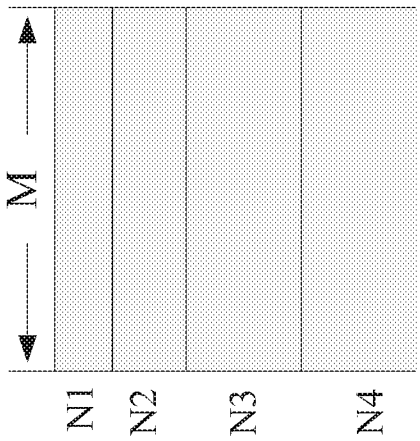


FIG. 7J

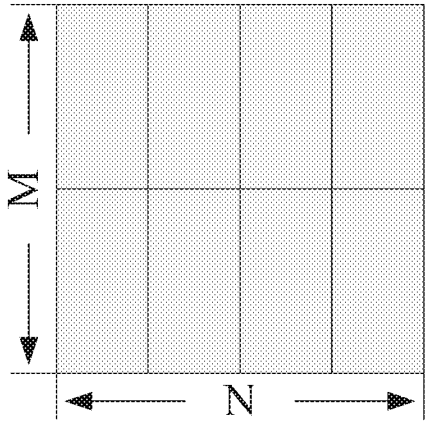


FIG. 8B

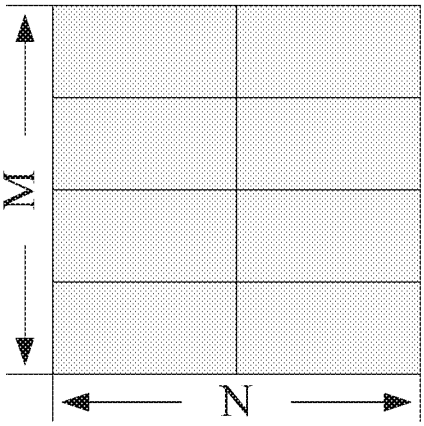


FIG. 8A

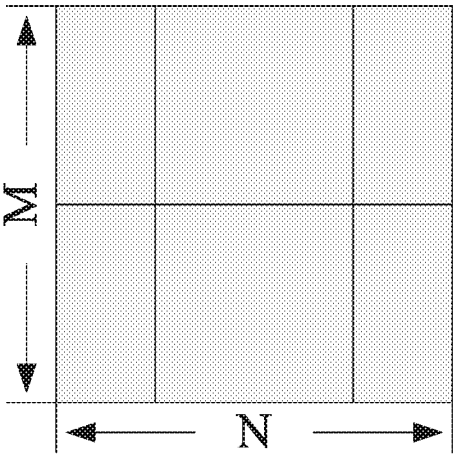


FIG. 8D

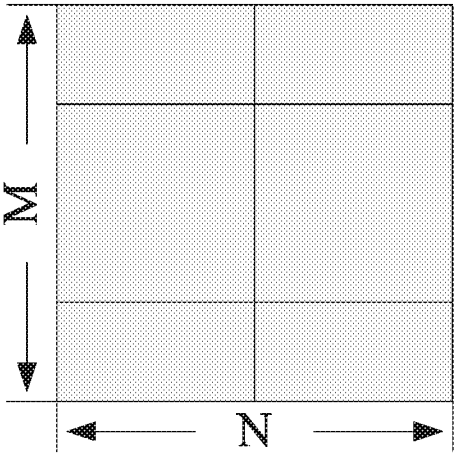


FIG. 8C

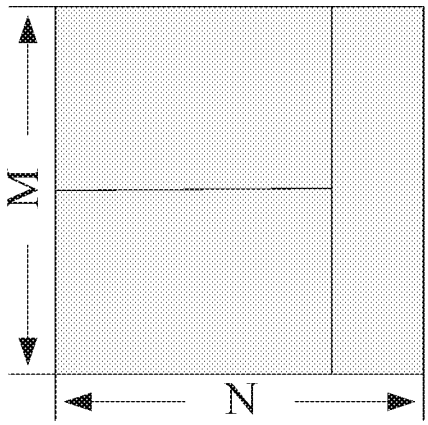


FIG. 9B

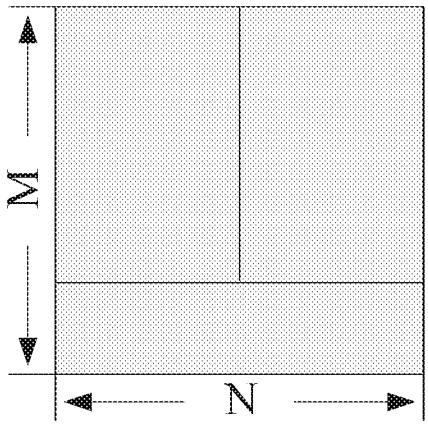


FIG. 9A

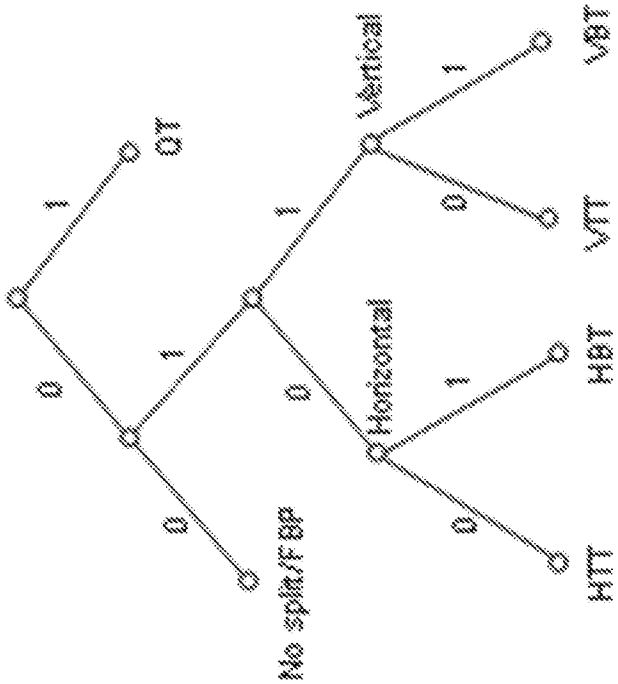


FIG. 10

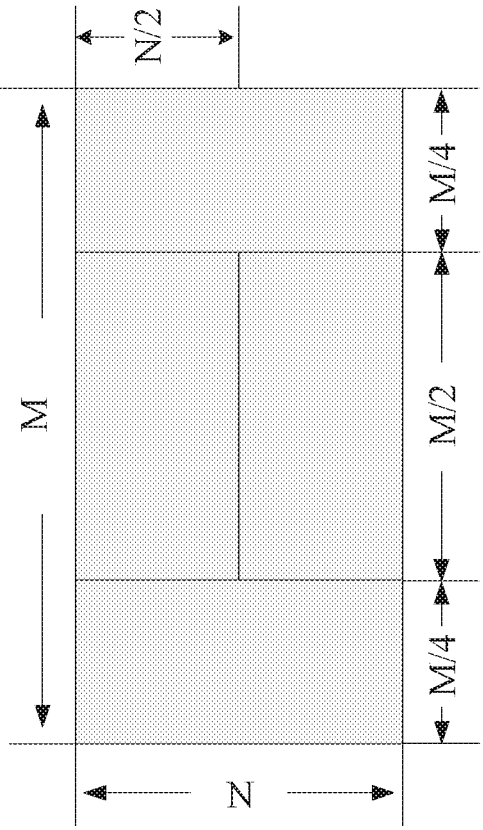


FIG. 11B

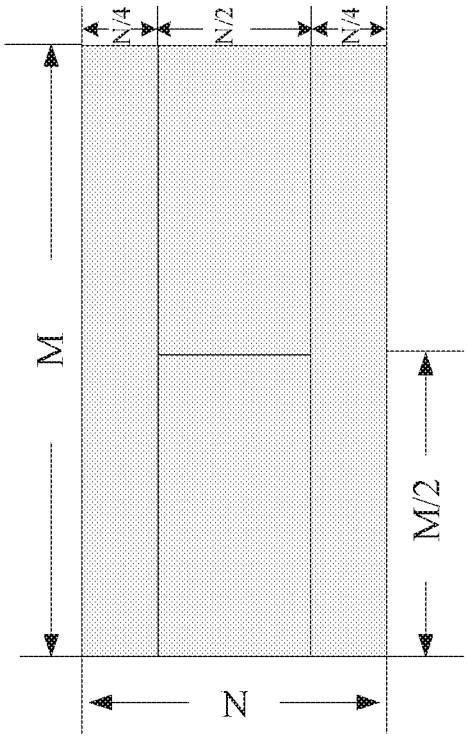


FIG. 11A

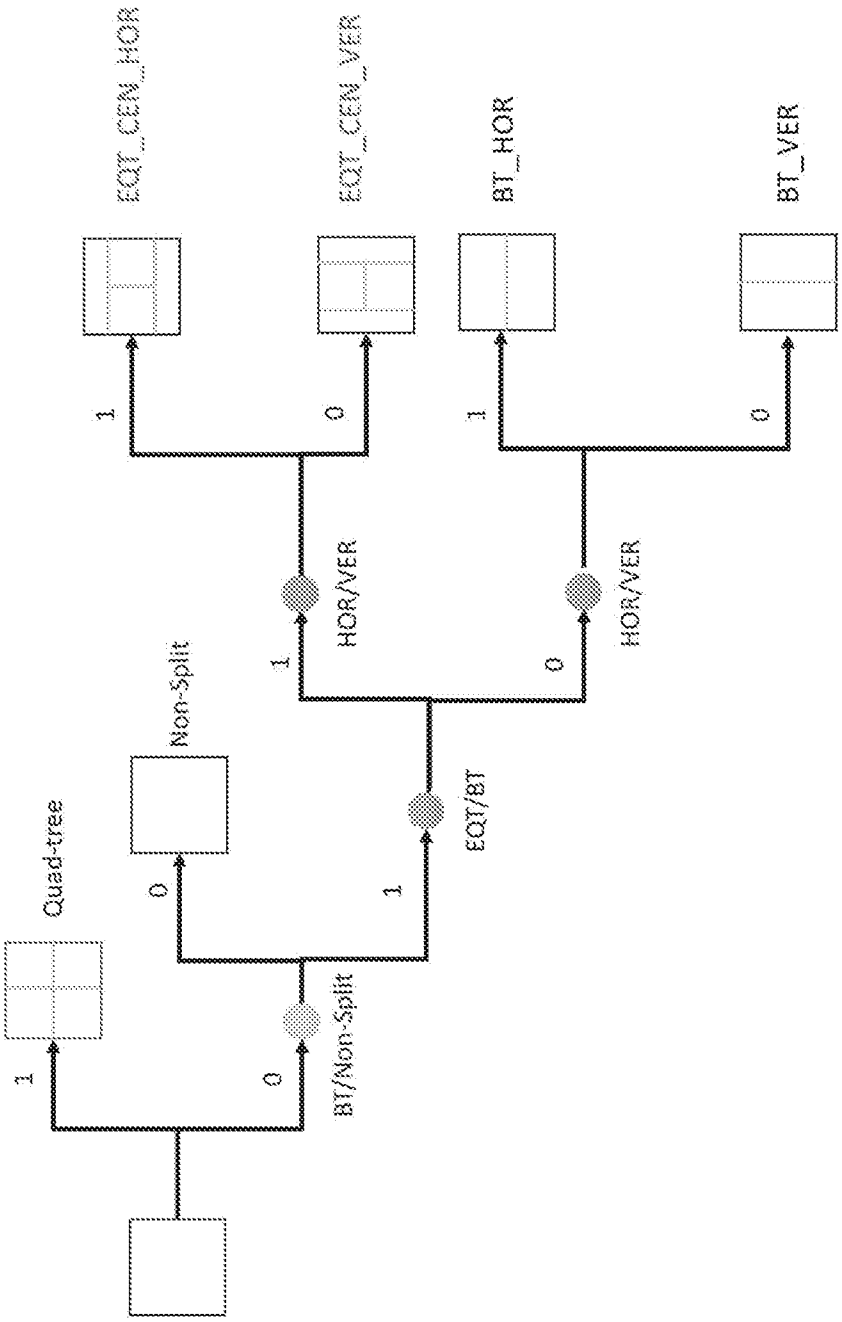

EQT_CEN_HOR
EQT_CEN_VER
BT_HOR
BT_VER
1
0
1
0
HOR/VER
HOR/VER
1
0
EQT/BT
Non-Split
0
1
Quad-tree
1
0
BT/Non-Split

FIG. 12

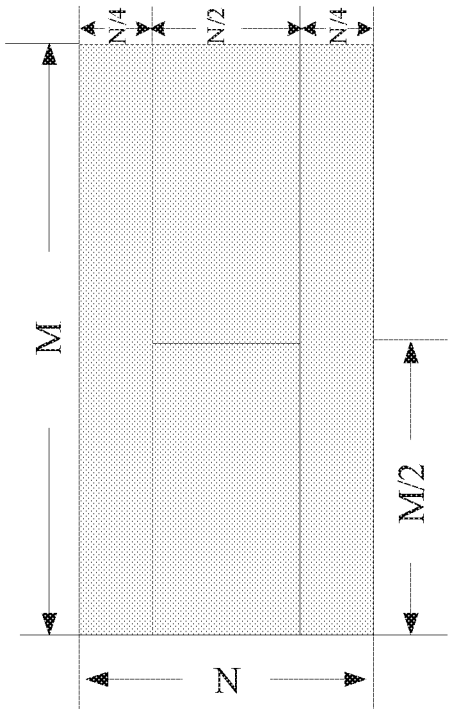


FIG. 13B

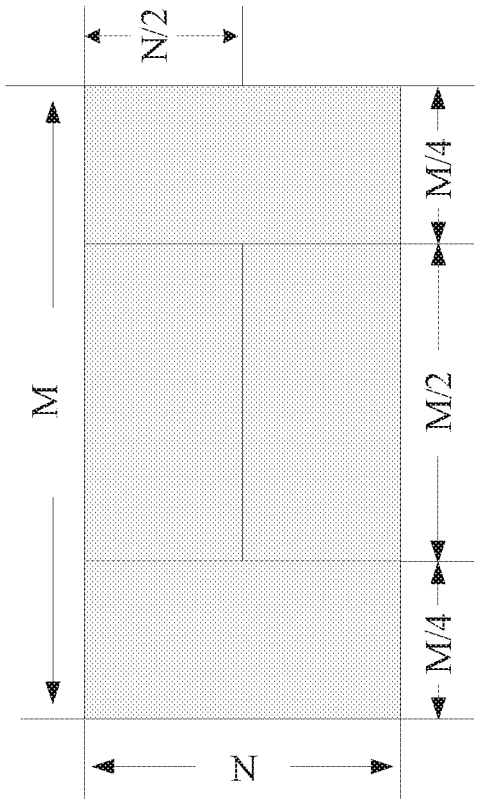


FIG. 13A

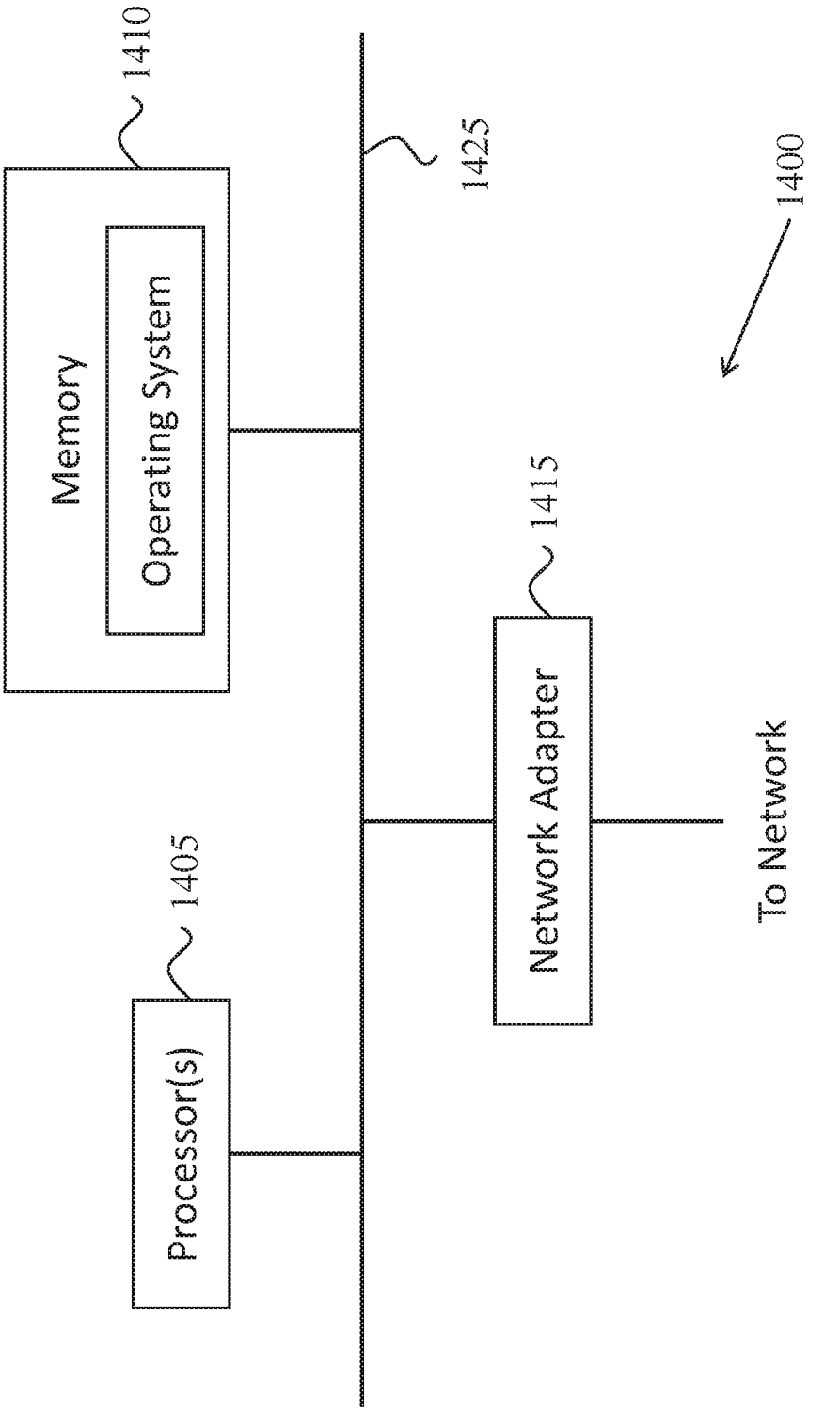

1410
Memory
Operating System
1425
1400
1415
Network Adapter
To Network
1405
Processor(s)

FIG. 14

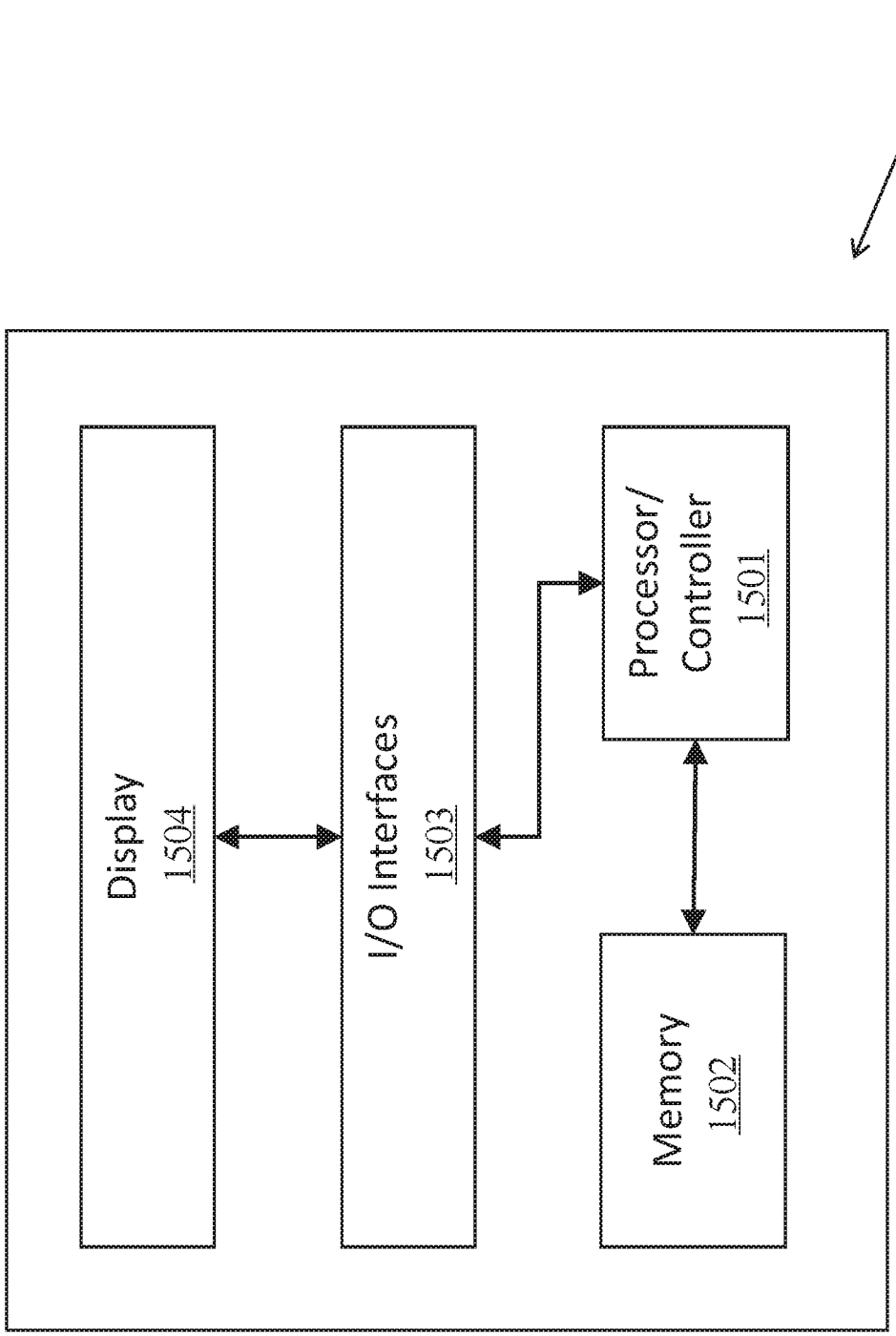

1500
Display
1504
I/O Interfaces
1503
Processor/
Controller
1501
Memory
1502

FIG. 15

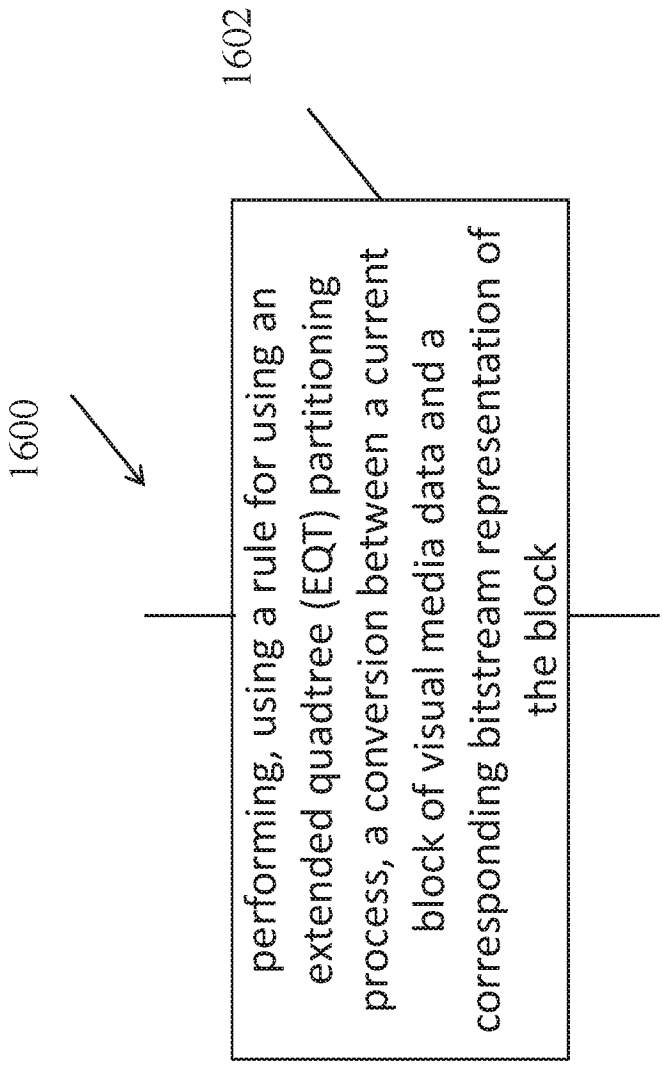

1600
1602
performing, using a rule for using an extended quadtree (EQT) partitioning process, a conversion between a current block of visual media data and a corresponding bitstream representation of the block

FIG. 16

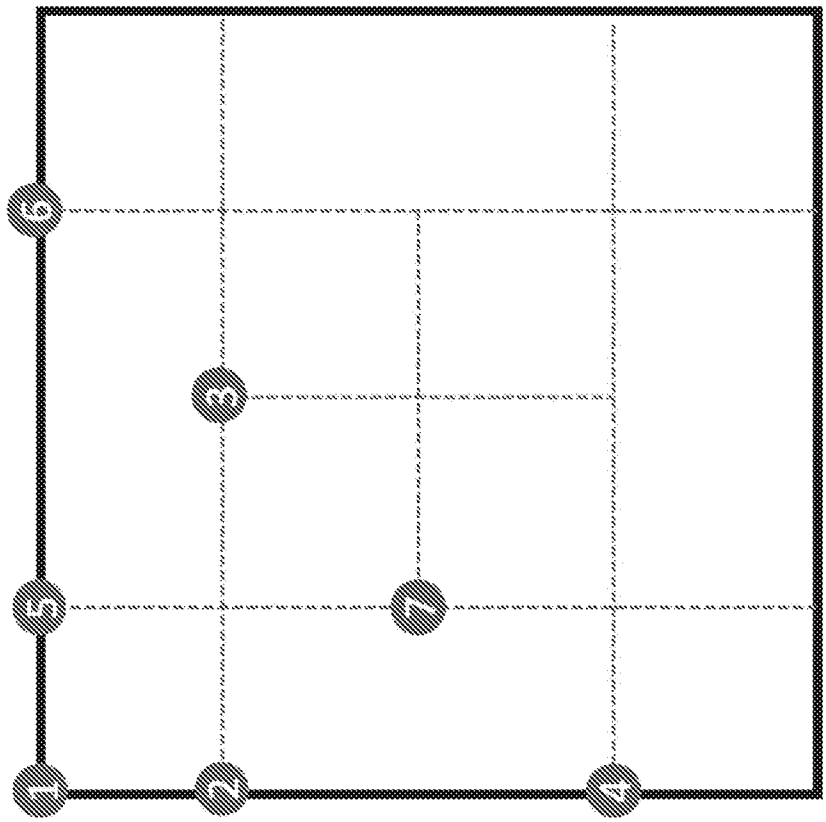


FIG. 17

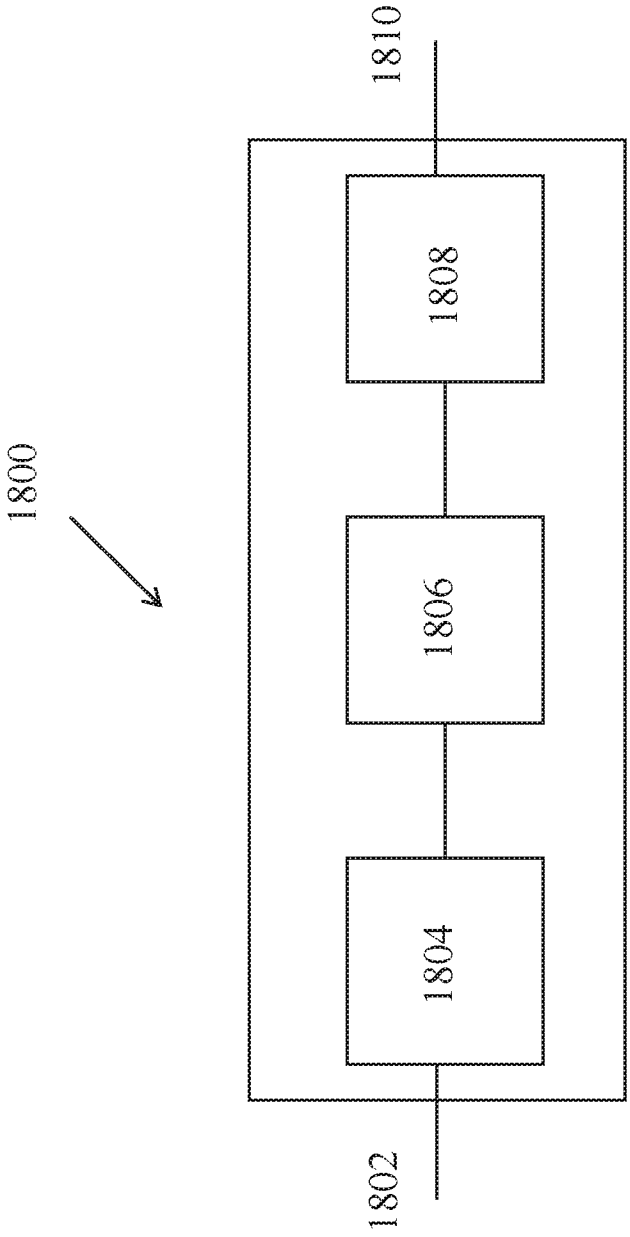


FIG. 18

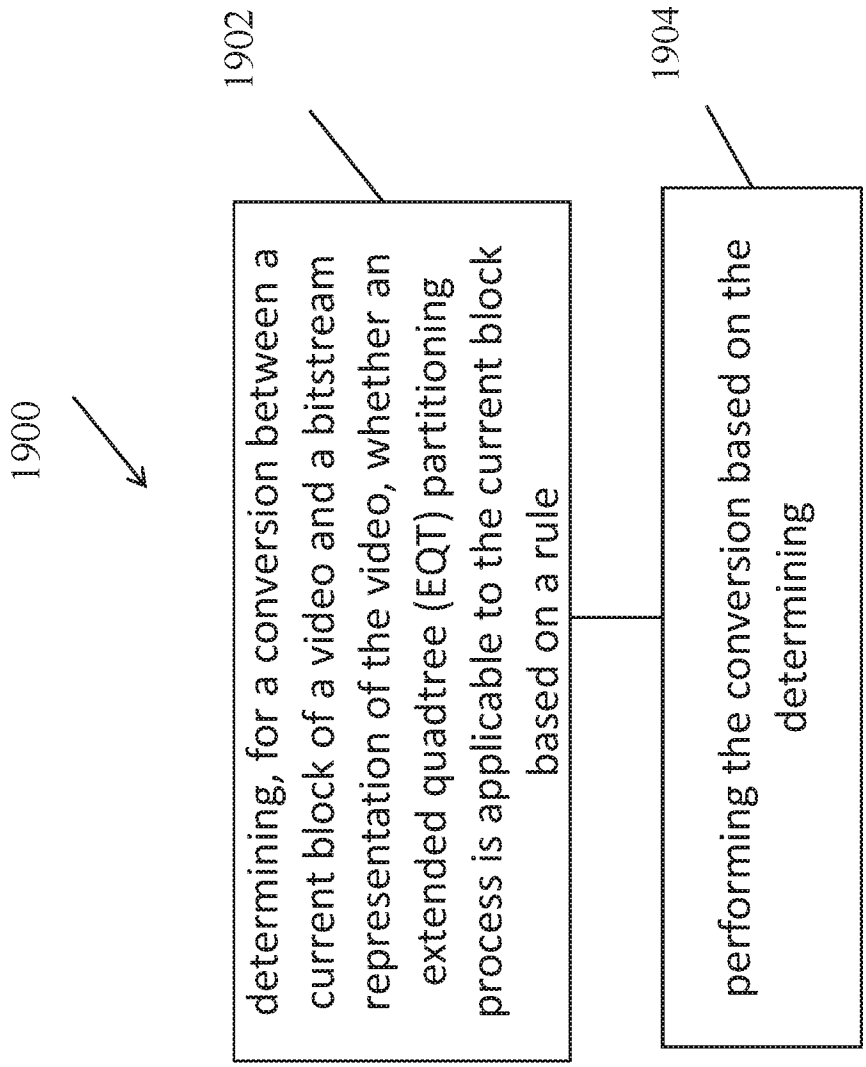

1900
1902
determining, for a conversion between a current block of a video and a bitstream representation of the video, whether an extended quadtree (EQT) partitioning process is applicable to the current block based on a rule
1904
performing the conversion based on the determining

FIG. 19

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2019/059219

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N19/119 H04N19/176 H04N19/70 H04N19/96
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04N

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, COMPENDEX, INSPEC, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 2018/088805 A1 (KT CORP) 17 May 2018 (2018-05-17) the whole document & US 2019/313129 A1 (LEE BAE KEUN [KR]) 10 October 2019 (2019-10-10) figures 1, 2, 11, 12 paragraph [0101] - paragraph [0183] -----	1-26
Y	EP 3 383 045 A1 (THOMSON LICENSING [FR]) 3 October 2018 (2018-10-03) paragraph [0058] - paragraph [0061] -----	1-3,6,7, 26,27

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 January 2020	06/02/2020

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Stoufs, Maryse

3

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	MISRA K ET AL: "Description of SDR and HDR video coding technology proposal by Sharp and Foxconn", 10. JVET MEETING; 10-4-2018 - 20-4-2018; SAN DIEGO; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16); URL: HTTP://PHENIX.INT-EVRY.FR/JVET/,, no. JVET-J0026-v9, 13 April 2018 (2018-04-13), XP030151194, table 2.1 page 8, paragraph 1 -----	1-3, 9-11,26, 27
Y	WU F ET AL: "Description of SDR video coding technology proposal by University of Science and Technology of China, Peking University, Harbin Institute of Technology, and Wuhan University (IEEE 1857.10 Study Group)", 10. JVET MEETING; 10-4-2018 - 20-4-2018; SAN DIEGO; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16); URL: HTTP://PHENIX.INT-EVRY.FR/JVET/,, no. JVET-J0032-v4, 12 April 2018 (2018-04-12), XP030151203, section 2.1.4.1 "QT-BT/TT tree" -----	1-3,26, 27
Y	CN 104 519 362 B (UNIV ELECTRONIC SCIENCE & TECH) 17 October 2017 (2017-10-17) paragraph [0009] - paragraph [0018] paragraph [0045] -----	1-5,8, 12-27

3

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/IB2019/059219

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2018088805 A1	17-05-2018	CN 109923866 A	21-06-2019
		KR 20180051424 A	16-05-2018
		US 2019313129 A1	10-10-2019
		WO 2018088805 A1	17-05-2018
EP 3383045 A1	03-10-2018	CN 110476423 A	19-11-2019
		EP 3383045 A1	03-10-2018
		EP 3603066 A1	05-02-2020
		WO 2018177953 A1	04-10-2018
CN 104519362 B	17-10-2017	NONE	