United States Patent	12395647
Kind Code	B2
Date of Patent	August 19, 2025
Inventor(s)	Zhang; Li et al.

Condition dependent video block partition

Abstract

Devices, systems and methods for video processing are described. In a representative aspect, there is disclosed a method for video processing, including: determining, for a current video block, whether a first partition mode is applicable to the current video block in responsive to at least one condition, wherein the current video block is split into M sub-blocks in the first partition mode, and M>4; and performing a conversion for the current video block based on the determination.

Inventors: **Zhang; Li (San Diego, CA), Zhang; Kai (San Diego, CA), Liu; Hongbin (Beijing, CN), Xu; Jizheng (San Diego, CA), Wang; Yue (Beijing, CN)**

Applicant: **Beijing Bytedance Network Technology Co., Ltd.** (Beijing, CN); **Bytedance Inc.** (Los Angeles, CA)

Family ID: **1000008762642**

Assignee: **BEIJING BYTEDANCE NETWORK TECHNOLOGY CO. LTD. (Beijing, CN); BYTEDANCE INC. (Los Angeles, CA)**

Appl. No.: **17/398800**

Filed: **August 10, 2021**

Prior Publication Data

Document Identifier	Publication Date
US 20210368185 A1	Nov. 25, 2021

Foreign Application Priority Data

WO	PCT/CN2019/074762	Feb. 11, 2019
WO	PCT/CN2019/077161	Mar. 06, 2019

Related U.S. Application Data

continuation parent-doc WO PCT/CN2020/074746 20200211 PENDING child-doc US 17398800

Publication Classification

Int. Cl.: **H04N19/176** (20140101); **H04N19/119** (20140101); **H04N19/157** (20140101); **H04N19/174** (20140101); **H04N19/66** (20140101); **H04N19/96** (20140101)

U.S. Cl.:

CPC **H04N19/176** (20141101); **H04N19/119** (20141101); **H04N19/157** (20141101); **H04N19/174** (20141101); **H04N19/66** (20141101); **H04N19/96** (20141101);

Field of Classification Search

CPC: H04N (19/176); H04N (19/119); H04N (19/157); H04N (19/174); H04N (19/66); H04N (19/96)

USPC: 375/240.24

References Cited

U.S. PATENT DOCUMENTS

Patent No.	Issued Date	Patentee Name	U.S. Cl.	CPC
3015445	12/1961	Toshio	N/A	N/A
RE37222	12/2000	Yonemitsu et al.	N/A	N/A
9503702	12/2015	Chen et al.	N/A	N/A
9667942	12/2016	Chen et al.	N/A	N/A
9838712	12/2016	Lin et al.	N/A	N/A
9906813	12/2017	Zhang et al.	N/A	N/A
9992494	12/2017	Zhang et al.	N/A	N/A
10212411	12/2018	Zhang et al.	N/A	N/A
10275020	12/2018	Cohen et al.	N/A	N/A
10321130	12/2018	Dong et al.	N/A	N/A
10334281	12/2018	Zhang et al.	N/A	N/A
10708589	12/2019	Mishurovskiy et al.	N/A	N/A
10709371	12/2019	Kalra et al.	N/A	N/A
10812835	12/2019	Wang et al.	N/A	N/A
11051023	12/2020	Lee	N/A	N/A
11115672	12/2020	Zhang	N/A	N/A
11184639	12/2020	Lee	N/A	N/A
11206419	12/2020	Lee	N/A	N/A
11218704	12/2021	Lim et al.	N/A	N/A
11228761	12/2021	Zhang et al.	N/A	N/A
11250326	12/2021	Ko et al.	N/A	N/A
11350107	12/2021	Jun et al.	N/A	N/A
11375185	12/2021	Lee et al.	N/A	N/A
2009/0116558	12/2008	Chen et al.	N/A	N/A
2011/0135143	12/2010	Zou et al.	N/A	N/A
2014/0139627	12/2013	Chen et al.	N/A	N/A

2015/0237324	12/2014	Zhang et al.	N/A	N/A
2015/0264356	12/2014	Zhang et al.	N/A	N/A
2015/0264399	12/2014	Lin et al.	N/A	N/A
2016/0100190	12/2015	Zhang et al.	N/A	N/A
2016/0234510	12/2015	Lin et al.	N/A	N/A
2018/0035126	12/2017	Lee	N/A	N/A
2018/0139444	12/2017	Huang et al.	N/A	N/A
2018/0213264	12/2017	Zhang et al.	N/A	N/A
2018/0249146	12/2017	Zhang et al.	N/A	N/A
2018/0352226	12/2017	An et al.	N/A	N/A
2019/0364279	12/2018	Yasugi	N/A	H04N 19/157
2020/0359039	12/2019	Zhao	N/A	H04N 19/51
2021/0329233	12/2020	Tsai	N/A	H04N 19/186
2021/0337198	12/2020	Wang et al.	N/A	N/A
2021/0377528	12/2020	Zhang et al.	N/A	N/A
2022/0070448	12/2021	Kim	N/A	H04N 19/54
2022/0086439	12/2021	Tsai	N/A	H04N 19/186
2022/0217383	12/2021	Wang et al.	N/A	N/A

FOREIGN PATENT DOCUMENTS

Patent No.	**Application Date**	**Country**	**CPC**
107079160	12/2016	CN	N/A
107810631	12/2017	CN	N/A
107948661	12/2017	CN	N/A
113396589	12/2020	CN	N/A
20140130572	12/2013	KR	N/A
2013154869	12/2012	WO	N/A
2014107709	12/2013	WO	N/A
2015007164	12/2014	WO	N/A
2015055134	12/2014	WO	N/A
2015135175	12/2014	WO	N/A
2015192706	12/2014	WO	N/A
2017088608	12/2016	WO	N/A
2018001207	12/2017	WO	N/A
2018110600	12/2017	WO	N/A
2018141116	12/2017	WO	N/A
2018169267	12/2017	WO	N/A

OTHER PUBLICATIONS

Bordes et al. “Description of SDR, HDR and 360 degree Video Coding Technology Proposal by Qualcomm and Technicolor—Medium Complexity Version,” Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 10th Meeting, San Diego, US, Apr. 10-20, 2018, document JVET-J0022, 2018. cited by applicant
Chen et al. “Algorithm Description of Joint Exploration Test Model 7 (JEM 7),” Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 7th Meeting: Torino, IT, Jul. 13-21, 2017, document JVET-G1001, 2017. cited by applicant
Li et al. “Multi-Type-Tree,” Joint Video Exploration Tree (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 4th Meeting, Chengdu, CN, Oct. 15-21, 2016, document JVET-D0117, 2016. cited by applicant
Luthra et al. Overview of the H.264/AVC Video Coding Standard, Proc. SPIE, 5203, Applications of Digital Image Processing, Nov. 19, 2003, Optical Science and Technology, SPIE 48th annual

Meeting, San Diego, CA, US, 2003. cited by applicant
Ma et al. “Summary Report for CE1: Partitioning,” Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 12th Meeting, Macao, CN, Oct. 3-12, 2018, document JVET-L0021, 2018. cited by applicant
Sullivan et al. “Overview of the High Efficiency Video Coding (HEVC) Standard,” IEEE Transactions on Circuits and Systems for Video Technology, Dec. 2012, 22(12):1649-1668. cited by applicant
JEM-7.0: https://jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/tags/ HM-16.6-JEM-7.0. cited by applicant
https://www.itu.int/rec/T-REC-H.265. cited by applicant
International Search Report and Written Opinion from PCT/CN2020/074745 dated Apr. 30, 2020 (11 pages). cited by applicant
International Search Report and Written Opinion from PCT/CN2020/074746 dated Apr. 24, 2020 (9 pages). cited by applicant
Ponton, Lionel. “Two Trees Enumerating the Positive Rationals,” 2017. cited by applicant
Non-Final Office Action from U.S. Appl. No. 17/398,857 dated Jul. 12, 2023. cited by applicant
Ma J., et al., Summary report for CE1: Partitioning, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 12th Meeting: Macao, CN, Oct. 3-12, 2018, JVET-L0021-v3.zip JVET-L0021-v2.docx. cited by applicant
Notice of First Examination Opinion for Chinese Application No. 202080012488.3, mailed Dec. 30, 2024, 28 pages. cited by applicant
Chinese Notice of Allowance from Chinese Patent Application No. 202080012488.3 dated May 30, 2025, 8 pages. cited by applicant

---

Primary Examiner: Saltarelli; Dominic D

Attorney, Agent or Firm: Conley Rose, P.C.

Background/Summary

CROSS REFERENCE TO RELATED APPLICATIONS (1) This application is a continuation of International Application No. PCT/CN2020/074746, filed on Feb. 11, 2020, which claims the priority to and benefits of International Patent Applications PCT/CN2019/074762, filed on Feb. 11, 2019 and PCT/CN2019/077161, filed on Mar. 6, 2019. The entire disclosures of the aforementioned applications are incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD
(1) This patent document relates to video coding techniques, devices and systems.
BACKGROUND
(2) In spite of the advances in video compression, digital video still accounts for the largest bandwidth use on the internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, it is expected that the bandwidth demand for digital video usage will continue to grow.
SUMMARY
(3) Devices, systems and methods related to digital video coding, and specifically, to quinary tree partitioning in video coding are described. The described methods may be applied to both the existing video coding standards (e.g., High Efficiency Video Coding (HEVC)) and future video coding standards (e.g., Versatile Video Coding (VVC)) or codecs.

(4) In one representative aspect, there is disclosed a method for video processing, comprising: determining, for a current video block, whether a first partition mode is applicable to the current video block in responsive to at least one condition, wherein the current video block is split into M sub-blocks in the first partition mode, and M>4; and performing a conversion for the current video block based on the determination.
(5) In another representative aspect, there is disclosed a method for video processing, comprising: determining, for a video block, whether and/or how to apply a first partition mode to the video block based on an indication, wherein the video block is split into M portions in the first partition mode, M>4; and performing a conversion of the video block based on the determination.
(6) In another representative aspect, the above-described method is embodied in the form of processor-executable code and stored in a computer-readable program medium.
(7) In yet another representative aspect, a device that is configured or operable to perform the above-described method is disclosed. The device may include a processor that is programmed to implement this method.
(8) In yet another representative aspect, a video decoder apparatus may implement a method as described herein.
(9) The above and other aspects and features of the disclosed technology are described in greater detail in the drawings, the description and the claims.

Description

BRIEF DESCRIPTION OF THE DRAWINGS
(1) FIG. **1** shows examples of macroblock partitions in H.264/AVC.
(2) FIG. **2** shows examples of modes for splitting a coding block into prediction blocks.
(3) FIGS. **3**A and **3**B show examples of partitioning a coding tree block (CTB) and its corresponding quadtree, respectively.
(4) FIG. **4** shows an example of a quadtree plus binary tree (QTBT) structure.
(5) FIGS. **5**A-**5**F show examples of allowed partitions in VVC.
(6) FIGS. **6**A-**6**E show examples of allowed partitions between a parent split (solid) and a current split (dashed), with “X” denoting a disallowed partition.
(7) FIGS. **7**A and **7**B show examples of extended quad-tree (EQT) horizontal and vertical modes, respectively.
(8) FIG. **8** shows an example of a signaling structure of QTBT plus EQT partitioning.
(9) FIGS. **9**A-**9**H show examples of unsymmetrical quad-tree (UQT) partitioning.
(10) FIGS. **10**A-**10**E show examples of quinary tree (QUI-T) partitioning.
(11) FIGS. **11**A and **11**B show examples of senary-partition structures.
(12) FIG. **12** is a block diagram illustrating an example of an apparatus that can implement a video encoder and/or decoder, which can be used to implement various portions of the presently disclosed technology.
(13) FIG. **13** shows a flowchart of an example method for video processing in accordance with the disclosed technology.
(14) FIG. **14** shows a flowchart of another example method for video processing in accordance with the disclosed technology.
DETAILED DESCRIPTION
(15) Due to the increasing demand of higher resolution video, video coding methods and techniques are ubiquitous in modern technology. Video codecs typically include an electronic circuit or software that compresses or decompresses digital video, and are continually being improved to provide higher coding efficiency. A video codec converts uncompressed video to a compressed format or vice versa. There are complex relationships between the video quality, the

amount of data used to represent the video (determined by the bit rate), the complexity of the encoding and decoding algorithms, sensitivity to data losses and errors, ease of editing, random access, and end-to-end delay (latency). The compressed format usually conforms to a standard video compression specification, e.g., the High Efficiency Video Coding (HEVC) standard (also known as H.265 or MPEG-H Part 2), the Versatile Video Coding (VVC) standard to be finalized, or other current and/or future video coding standards.
(16) Embodiments of the disclosed technology may be applied to existing video coding standards (e.g., HEVC, H.265) and future standards to improve runtime performance. Section headings are used in the present document to improve readability of the description and do not in any way limit the discussion or the embodiments (and/or implementations) to the respective sections only.
1. Overview of Video Coding Standards
(17) 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. 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). In April 2018, the Joint Video Expert Team (JVET) between VCEG (Q 6/16) and ISO/IEC JTC1 SC29/WG11 (MPEG) was created to work on the VVC standard targeting at 50% bitrate reduction compared to HEVC.
2. Exemplary Embodiments for Quinary Tree Partitioning
(18) 2.1 Partition Tree Structure in H.264/AVC
(19) The terminology used in H.264/AVS is macroblock and MB-mode/8×8-mode (partition). Macroblock is the unit wherein each picture/slice is split to and where intra/inter mode decision is applied. And partition defines the level wherein motion information is signaled.
(20) The core of the coding layer in H.264/AVC 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.
(21) 2.1.1 H.264/AVC Main Profile
(22) An intra-coded block uses spatial prediction to exploit spatial correlation among pixels. Two partitions are defined: 16×16 and 4×4.
(23) An inter-coded block uses temporal prediction, instead of spatial prediction, by estimating motion among pictures. Motion can be estimated independently for either 16×16 macroblock or any of its macroblock partitions: 16×8, 8×16, 8×8. An syntax element (MB-mode) is signaled to indicate whether 16×16, 16×8, 8×16 or 8×8 is chosen. If 8×8 is selected, another syntax element (8×8-mode) is further signaled to indicate whether 8×8, 8×4, 4×8, 4×4 (see, e.g., FIG. **1**) is used. Only one motion vector (MV) per partition is allowed.
(24) 2.1.2 H.264/AVC High Profile
(25) In the high profile, 8×8 transform and I_8×8 (8×8 intra prediction) is introduced. For intra-coded macroblock, the transform size is fixed, I_16×6 and I_4×4 uses 4×4 transform; I_8×8 uses 8×8 transform.
(26) For inter-coded macroblocks, either 4×4 or 8×8 transform could be selected. However, the transform size couldn't cross the partition size. For example, if one macroblock chooses 8×8 partition and further selects 8×4 sub-mode, only 4×4 transform may be applied. If one macroblock chooses 16×16, 16×8, 8×16 8×8 partition with 8×8 sub-mode, then either 4×4 or 8×8 transform could be selected.
(27) 2.1.3 Summary
(28) Mode selection is decided in macroblock-level. Transform size shall be no larger than the

partition sizes.
(29) 2.2 Partition Tree Structure in HEVC
(30) In HEVC, a coding tree unit (CTU, aka largest coding unit, LCU) is split into coding units (CUs) by using a quadtree structure denoted as coding tree to adapt to various local characteristics. The decision whether to code a picture area using inter-picture (temporal) or intra-picture (spatial) prediction is made at the CU level. Each CU can be further split into one, two or four 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.
(31) In the following, the various features involved in hybrid video coding using HEVC are highlighted as follows.
(32) 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.
(33) 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).
(34) 3) Prediction units (PUs) and prediction blocks (PBs): The decision whether to code a picture area using inter picture or intra picture prediction is made at the CU level. A PU partitioning structure has its root at the CU level. Depending on the basic prediction-type decision, the luma and chroma CBs can then be further split in size and predicted from luma and chroma prediction blocks (PBs). HEVC supports variable PB sizes from 64×64 down to 4×4 samples. FIG. **2** depicts the allowed PBs.
(35) 4) Transform units (Tus) and transform blocks: The prediction residual is coded using block transforms. A TU tree structure has its root at the CU level. The luma CB residual may be identical to the luma transform block (TB) or may be further split into smaller luma TBs. The same applies to the chroma TBs. Integer basis functions similar to those of a discrete cosine transform (DCT) are defined for the square TB sizes 4×4, 8×8, 16×16, and 32×32. For the 4×4 transform of luma intra picture prediction residuals, an integer transform derived from a form of discrete sine transform (DST) is alternatively specified.
(36) 2.2.1 Depth of Quadtree
(37) For a given luma CB of size M×M, a flag signals whether it is split into four blocks of size M/2×M/2. If further splitting is possible, as signaled by a maximum depth of the residual quadtree indicated in the SPS, each quadrant is assigned a flag that indicates whether it is split into four quadrants. The leaf node blocks resulting from the residual quadtree are the transform blocks that are further processed by transform coding. The encoder indicates the maximum and minimum luma TB sizes that it will use. Splitting is implicit when the CB size is larger than the maximum TB size. Not splitting is implicit when splitting would result in a luma TB size smaller than the indicated minimum. The chroma TB size is half the luma TB size in each dimension, except when the luma TB size is 4×4, in which case a single 4×4 chroma TB is used for the region covered by four 4×4

luma TBs. In the case of intrapicture-predicted CUs, the decoded samples of the nearest-neighboring TBs (within or outside the CB) are used as reference data for intrapicture prediction.
(38) 2.2.2 Summary
(39) One CTU may be recursively split into multiple CUs based on increased depth of quadtree (e.g., FIG. **3**B). Only square CB and TB partitioning is specified, where a block can be recursively split into quadrants, as illustrated in FIG. **3**A.
(40) Mode selection is decided in CU-level. Side information according to a selected mode is signaled in PU-level, such as motion information, intra prediction modes. Residual are signaled in TU-level.
(41) One PU shall be no larger than CU for inter-coded blocks and one PU shall be equal to CU for intra-coded blocks.
(42) TU could cross PU for inter-coded blocks, but shall be equal to PU for intra-coded blocks.
(43) 2.3 Quadtree Plus Binary Tree Block Structure with Larger CTUs in JEM
(44) 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).
(45) 2.3.1 QTBT Block Partitioning Structure
(46) Different from HEVC, the QTBT structure 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. In an example, 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 color 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.
(47) 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
(48) In one example of the QTBT partitioning structure, the CTU size is set as 128×128 luma samples with two corresponding 64×64 blocks of chroma samples, the MinQTSize is set as 16×16, the MaxBTSize is set as 64×64, the MinBTSize (for both width and height) is set as 4×4, and the MaxBTDepth is set as 4. The quadtree partitioning is applied to the CTU first to generate quadtree leaf nodes. The quadtree leaf nodes may have a size from 16×16 (i.e., the MinQTSize) to 128×128 (i.e., the CTU size). If the leaf quadtree node is 128×128, it will not be further split by the binary tree since the size exceeds the MaxBTSize (i.e., 64×64). Otherwise, the leaf quadtree node could be further partitioned by the binary tree. Therefore, the quadtree leaf node is also the root node for the binary tree and it has the binary tree depth as 0. When the binary tree depth reaches MaxBTDepth (i.e., 4), no further splitting is considered. When the binary tree node has width equal to MinBTSize (i.e., 4), no further horizontal splitting is considered. Similarly, when the binary tree node has height equal to MinBTSize, no further vertical splitting is considered. The leaf nodes of the binary tree are further processed by prediction and transform processing without any further partitioning. In the JEM, the maximum CTU size is 256×256 luma samples.
(49) FIG. **4** (left) illustrates an example of block partitioning by using QTBT, and FIG. **4** (right) illustrates the corresponding tree representation. The solid lines indicate quadtree splitting and dotted lines indicate binary tree splitting. In each splitting (i.e., non-leaf) node of the binary tree,

one flag is signaled 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.
(50) In addition, the QTBT scheme supports the ability for the luma and chroma to have a separate QTBT structure. Currently, for P and B slices, the luma and chroma CTBs in one CTU share the same QTBT structure. However, for I slices, the luma CTB is partitioned into CUs by a QTBT structure, and the chroma CTBs are partitioned into chroma CUs by another QTBT structure. This means that a CU in an I slice consists of a coding block of the luma component or coding blocks of two chroma components, and a CU in a P or B slice consists of coding blocks of all three color components.
(51) 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.
(52) 2.3.2 Summary of QTBT
(53) One CTU may be recursively split into multiple CUs based on increased depth of quadtree or binary tree. Square and rectangular CB (with width/height equal to ½ or 2) is specified.
(54) Mode selection is decided in CU-level. PU and TU are always equal to CU.
(55) 2.4 Multiple Type Trees (MTT) for VVC
(56) 2.4.1 Proposal in JVET-D0117
(57) It is proposed that tree types other than quad-tree and binary-tree are supported. In the implementation, two more ternary tree (TT) partitions, i.e., horizontal and vertical center-side triple-trees are introduced, as shown in FIG. **5**E and FIG. **5**F.
(58) In some embodiments, the one partition in BT/TT may be further split with BT/TT. Therefore, rectangular blocks are allowed.
(59) 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’.
(60) 2.4.2 Partition Tree in VVC
(61) Similarly, it is proposed that three types of partition structures are supported, i.e., QT, BT and TT, as shown in the examples in FIGS. **6**A-**6**E. A block split from QT may be further split by QT/BT/TT. a block split from BT or TT may be further split to BT or TT. However, a block split from BT or TT couldn't be further split to QT anymore.
(62) In VVC, several variables are signaled/derived to control the usage of different partitions. For example: maximum multi-type tree depth with offset maxMttDepth for luma and chroma, respectively, maximum binary tree size maxBtSize/ternary tree size maxTtSize minimum quadtree size MinQtSize/binary tree size MinBtSize/ternary tree size minTtSize
7.3.2.1 Sequence Parameter Set RBSP Syntax
(63) TABLE-US-00001 De- scriptor seq_parameter_set_rbsp( ) {   sps_seq_parameter_set_id ue(v) ...   qtbtt_dual_tree_intra_flag ue(v)   log2_ctu_size_minus2 ue(v) log2_min_luma_coding_block_size_minus2 ue(v)   partition_constraints_override_enabled_flag ue(v)   sps_log2_diff_min_qt_min_cb_intra_tile_group_luma ue(v) sps_log2_diff_min_qt_min_cb_inter_tile_group ue(v) sps_max_mtt_hierarchy_depth_inter_tile_groups ue(v) sps_max_mtt_hierarchy_depth_intra_tile_groups_luma ue(v)   if(sps_max_mtt_hierarchy_depth_intra_tile_groups_luma   != 0) { sps_log2_diff_max_bt_min_qt_intra_tile_group_luma ue(v)

sps_log2_diff_max_tt_min_qt_intra_tile_group_luma ue(v)   }   if(sps_max_mtt_hierarchy_depth_inter_tile_groups != 0) { sps_log2_diff_max_bt_min_qt_inter_tile_group ue(v) sps_log2_diff_max_tt_min_qt_inter_tile_group ue(v)   }   if(qtbtt_dual_tree_intra_flag) { sps_log2_diff_min_qt_min_cb_intra_tile_group_chroma ue(v) sps_max_mtt_hierarchy_depth_intra_tile_groups_chroma ue(v)   if (sps_max_mtt_hierarchy_depth_intra_tile_groups_chroma   != 0) { sps_log2_diff_max_bt_min_qt_intra_tile_group_chroma ue(v) sps_log2_diff_max_tt_min_qt_intra_tile_group_chroma ue(v)   }   }   sps_sao_enabled_flag u(1) ...   rbsp_trailing_bits( ) }

Semantics

(64) sps_max_mtt_hierarchy_depth_inter_tile_groups specifies the default maximum hierarchy depth for coding units resulting from multi-type tree splitting of a quadtree leaf in tile groups with tile_group_type equal to 0 (B) or 1 (P) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default maximum hierarchy depth can be overridden by tile_group_max_mtt_hierarchy_depth_luma present in the tile group header of the tile groups referring to the SPS. The value of sps_max_mtt_hierarchy_depth_inter_tile_groups shall be in the range of 0 to CtbLog2SizeY−MinCbLog2SizeY, inclusive.

(65) sps_max_mtt_hierarchy_depth_intra_tile_groups_luma specifies the default maximum hierarchy depth for coding units resulting from multi-type tree splitting of a quadtree leaf in tile groups with tile_group_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default maximum hierarchy depth can be overridden by tile_group_max_mtt_hierarchy_depth_luma present in the tile group header of the tile groups referring to the SPS. The value of sps_max_mtt_hierarchy_depth_intra_tile_groups_luma shall be in the range of 0 to CtbLog2SizeY MinCbLog2SizeY, inclusive.

(66) sps_log2_diff_max_bt_min_qt_intra_tile_group_luma specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a binary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in tile groups with tile_group_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be overridden by tile_group_log2_diff_max_bt_min_qt_luma present in the tile group header of the tile groups referring to the SPS. The value of sps_log2_diff_max_bt_min_qt_intra_tile_group_luma shall be in the range of 0 to CtbLog2SizeY MinQtLog2SizeIntraY, inclusive. When sps_log2_diff_max_bt_min_qt_intra_tile_group_luma is not present, the value of sps_log2_diff_max_bt_min_qt_intra_tile_group_luma is inferred to be equal to 0.

(67) sps_log2_diff_max_tt_min_qt_infra_tile_group_luma specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a ternary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in tile groups with tile_group_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be overridden by tile_group_log2_diff_max_tt_min_qt_luma present in the tile group header of the tile groups referring to the SPS. The value of sps_log2_diff_max_tt_min_qt_intra_tile_group_luma shall be in the range of 0 to CtbLog2SizeY−MinQtLog2SizeIntraY, inclusive. When sps_log2_diff_max_tt_min_qt_intra_tile_group_luma is not present, the value of sps_log2_diff_max_tt_min_qt_intra_tile_group_luma is inferred to be equal to 0.

(68) sps_log2_diff_max_bt_min_qt_inter_tile_group specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block

that can be split using a binary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in tile groups with tile_group_type_equal to 0 (B) or 1 (P) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be overridden by tile_group_log2_diff_max_bt_min_qt_luma present in the tile group header of the tile groups referring to the SPS. The value of sps_log2_diff_max_bt_min_qt_inter_tile_group shall be in the range of 0 to CtbLog2SizeY−MinQtLog2SizeInterY, inclusive. When sps_log2_diff_max_bt_min_qt_inter_tile_group is not present, the value of sps_log2_diff_max_bt_min_qt_inter_tile_group is inferred to be equal to 0.
(69) sps_log2_diff_max_tt_min_qt_inter_tile_group specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a ternary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in tile groups with tile_group_type equal to 0 (B) or 1 (P) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be overridden by tile_group_log2_diff_max_tt_min_qt_luma present in the tile group header of the tile groups referring to the SPS. The value of sps_log2_diff_max_tt_min_qt_inter_tile_group shall be in the range of 0 to CtbLog2SizeY−MinQtLog2SizeInterY, inclusive. When sps_log2_diff_max_tt_min_qt_inter_tile_group is not present, the value of sps_log2_diff_max_tt_min_qt_inter_tile_group is inferred to be equal to 0.
(70) sps_log2_diff_min_qt_min_cb_intra_tile_group_chroma specifies the default difference between the base 2 logarithm of the minimum size in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA and the base 2 logarithm of the minimum coding block size in luma samples for chroma CUs with treeType equal to DUAL_TREE_CHROMA in tile groups with tile_group_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be overridden by tile_group_log2_diff_min_qt_min_cb_chroma present in the tile group header of the tile groups referring to the SPS. The value of sps_log2_diff_min_qt_min_cb_intra_tile_group_chroma shall be in the range of 0 to CtbLog2SizeY−MinCbLog2SizeY, inclusive. When not present, the value of sps_log2_diff_min_qt_min_cb_intra_tile_group_chroma is inferred to be equal to 0. The base 2 logarithm of the minimum size in luma samples of a chroma leaf block resulting from quadtree splitting of a CTU with treeType equal to DUAL_TREE_CHROMA is derived as follows:
MinQtLog2SizeIntra$C$=sps_log2_diff_min_qt_min_cb_intra_tile_group_chroma+MinCbLog2Size$Y$
(7 28)
(71) sps_max_mtt_hierarchy_depth_intra_tile_groups_chroma specifies the default maximum hierarchy depth for chroma coding units resulting from multi-type tree splitting of a chroma quadtree leaf with treeType equal to DUAL_TREE_CHROMA in tile groups with tile_group_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default maximum hierarchy depth can be overridden by tile_group_max_mtt_hierarchy_depth_chroma_present in the tile group header of the tile groups referring to the SPS. The value of sps_max_mtt_hierarchy_depth_intra_tile_groups_chroma shall be in the range of 0 to CtbLog2SizeY−MinCbLog2SizeY, inclusive. When not present, the value of sps_max_mtt_hierarchy_depth_intra_tile_groups_chroma is inferred to be equal to 0.
(72) sps_log2_diff_max_bt_min_qt_intra_tile_group_chroma specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a chroma coding block that can be split using a binary split and the minimum size (width or height) in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA in tile groups with tile_group_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be

overridden by tile_group_log2_diff_max_bt_min_qt_chroma present in the tile group header of the tile groups referring to the SPS. The value of sps_log2_diff_max_bt_min_qt_intra_tile_group_chroma shall be in the range of 0 to CtbLog2SizeY−MinQtLog2SizeIntraC, inclusive. When sps_log2_diff_max_bt_min_qt_intra_tile_group_chroma is not present, the value of sps_log2_diff_max_bt_min_qt_intra_tile_group_chroma is inferred to be equal to 0.
(73) sps_log2_diff_max_tt_min_qt_infra_tile_group_chroma specifies the default difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a chroma coding block that can be split using a ternary split and the minimum size (width or height) in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA in tile groups with tile_group_type equal to 2 (I) referring to the SPS. When partition_constraints_override_flag is equal to 1, the default difference can be overridden by tile_group_log2_diff_max_tt_min_qt_chroma present in the tile group header of the tile groups referring to the SPS. The value of sps_log2_diff_max_tt_min_qt_intra_tile_group_chroma shall be in the range of 0 to CtbLog2SizeY−MinQtLog2SizeIntraC, inclusive. When sps_log2_diff_max_tt_min_qt_intra_tile_group_chroma is not present, the value of sps_log2_diff_max_tt_min_qt_intra_tile_group_chroma is inferred to be equal to 0.
(74) 2.4.2.1 Restrictions of Usage of BT and TT
(75) 2.4.2.1.1 Variable Definitions
(76) tile_group_log2_diff_min_qt_min_cb_luma specifies the difference between the base 2 logarithm of the minimum size in luma samples of a luma leaf block resulting from quadtree splitting of a CTU and the base 2 logarithm of the minimum coding block size in luma samples for luma CUs in the current tile group. The value of tile_group_log2_diff_min_qt_min_cb_luma shall be in the range of 0 to CtbLog2SizeY−MinCbLog2SizeY, inclusive. When not present, the value of tile_group_log2_diff_min_qt_min_cb_luma is inferred as follows: If tile_group_type equal to 2 (I), the value of tile_group_log2_diff_min_qt_min_cb_luma is inferred to be equal to sps_log2_diff_min_qt_min_cb_intra_tile_group_luma Otherwise (tile_group_type equal to 0 (B) or 1 (P)), the value of tile_group_log2_diff_min_qt_min_cb_luma is inferred to be equal to sps_log2_diff_min_qt_min_cb_inter_tile_group.
(77) tile_group_max_mtt_hierarchy_depth_luma specifies the maximum hierarchy depth for coding units resulting from multi-type tree splitting of a quadtree leaf in the current tile group. The value of tile_group_max_mtt_hierarchy_depth_luma shall be in the range of 0 to CtbLog2SizeY−MinCbLog2SizeY, inclusive. When not present, the value of tile_group_max_mtt_hierarchy_depth_luma is inferred as follows: If tile_group_type equal to 2 (I), the value of tile_group_max_mtt_hierarchy_depth_luma is inferred to be equal to sps_max_mtt_hierarchy_depth_intra_tile_groups_luma Otherwise (tile_group_type_equal to 0 (B) or 1 (P)), the value of tile_group_max_mtt_hierarchy_depth_luma is inferred to be equal to sps_max_mtt_hierarchy_depth_inter_tile_groups.
(78) tile_group_log2_diff_max_bt_min_qt_luma specifies the difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a binary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in the current tile group. The value of tile_group_log2_diff_max_bt_min_qt_luma shall be in the range of 0 to CtbLog2SizeY−MinQtLog2SizeY, inclusive. When not present, the value of tile_group_log2_diff_max_bt_min_qt_luma is inferred as follows: If tile_group_type equal to 2 (I), the value of tile_group_log2_diff_max_bt_min_qt_luma is inferred to be equal to sps_log2_diff_max_bt_min_qt_intra_tile_group_luma Otherwise (tile_group_type equal to 0 (B) or 1 (P)), the value of tile_group_log2_diff_max_bt_min_qt_luma is inferred to be equal to sps_log2_diff_max_bt_min_qt_inter_tile_group.

(79) tile_group_log2_diff_max_tt_min_qt_luma specifies the difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a luma coding block that can be split using a ternary split and the minimum size (width or height) in luma samples of a luma leaf block resulting from quadtree splitting of a CTU in in the current tile group. The value of tile_group_log2_diff_max_tt_min_qt_luma shall be in the range of 0 to CtbLog2SizeY−MinQtLog2SizeY, inclusive. When not present, the value of tile_group_log2_diff_max_tt_min_qt_luma is inferred as follows: If tile_group_type equal to 2 (I), the value of tile_group_log2_diff_max_tt_min_qt_luma is inferred to be equal to sps_log2_diff_max_tt_min_qt_intra_tile_group_luma Otherwise (tile_group_type equal to 0 (B) or 1 (P)), the value of tile_group_log2_diff_max_tt_min_qt_luma is inferred to be equal to sps_log2_diff_max_tt_min_qt_inter_tile_group.

(80) tile_group_log2_diff_min_qt_min_cb_chroma specifies the difference between the base 2 logarithm of the minimum size in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA and the base 2 logarithm of the minimum coding block size in luma samples for chroma CUs with treeType equal to DUAL_TREE_CHROMA in the current tile group. The value of tile_group_log2_diff_min_qt_min_cb_chroma shall be in the range of 0 to CtbLog2SizeY−MinCbLog2SizeY, inclusive. When not present, the value of tile_group_log2_diff_min_qt_min_cb_chroma is inferred to be equal to sps_log2_diff_min_qt_min_cb_intra_tile_group_chroma.

(81) tile_group_max_mtt_hierarchy_depth_chroma specifies the maximum hierarchy depth for coding units resulting from multi-type tree splitting of a quadtree leaf with treeType equal to DUAL_TREE_CHROMA in the current tile group. The value of tile_group_max_mtt_hierarchy_depth_chroma shall be in the range of 0 to CtbLog2SizeY−MinCbLog2SizeY, inclusive. When not present, the values of tile_group_max_mtt_hierarchy_depth_chroma is inferred to be equal to sps_max_mtt_hierarchy_depth_intra_tile_groups_chroma.

(82) tile_group_log2_diff_max_bt_min_qt_chroma specifies the difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a chroma coding block that can be split using a binary split and the minimum size (width or height) in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA in the current tile group. The value of tile_group_log2_diff_max_bt_min_qt_chroma shall be in the range of 0 to CtbLog2SizeY−MinQtLog2SizeC, inclusive. When not present, the value of tile_group_log2_diff_max_bt_min_qt_chroma is inferred to be equal to sps_log2_diff_max_bt_min_qt_intra_tile_group_chroma

(83) tile_group_log2_diff_max_tt_min_qt_chroma specifies the difference between the base 2 logarithm of the maximum size (width or height) in luma samples of a chroma coding block that can be split using a ternary split and the minimum size (width or height) in luma samples of a chroma leaf block resulting from quadtree splitting of a chroma CTU with treeType equal to DUAL_TREE_CHROMA in the current tile group. The value of tile_group_log2_diff_max_tt_min_qt_chroma shall be in the range of 0 to CtbLog2SizeY−MinQtLog2SizeC, inclusive. When not present, the value of tile_group_log2_diff_max_tt_min_qt_chroma is inferred to be equal to sps_log2_diff_max_tt_min_qt_intra_tile_group_chroma

(84) The variables MinQtLog2SizeY, MinQtLog2SizeC, MinQtSizeY, MinQtSizeC, MaxBtSizeY, MaxBtSizeC, MinBtSizeY, MaxTtSizeY, MaxTtSizeC, MinTtSizeY, MaxMttDepthY and MaxMttDepthC are derived as follows:

MinQtLog2Size$Y$=MinCbLog2Size$Y$+tile_group_log2_diff_min_qt_min_cb_luma (7-33)

MinQtLog2Size$C$=MinCbLog2Size$Y$+tile_group_log2_diff_min_qt_min_cb_chroma (7-34)

MinQtSize*Y*=1<<MinQtLog2Size*Y* (7-35)
MinQtSize*C*=1<<MinQtLog2Size*C* (7-36)
MaxBtSize*Y*=1<<(MinQtLog2Size*Y*+tile_group_log2_diff_max_bt_min_qt_luma) (7-37)
MaxBtSize*C*=1<<(MinQtLog2Size*C*+tile_group_log2_diff_max_bt_min_qt_chroma) (7-38)
MinBtSize*Y*=1<<MinCbLog2Size*Y* (7-39)
MaxTtSize*Y*=1<<(MinQtLog2Size*Y*+tile_group_log2_diff_max_tt_min_qt_luma) (7-40)
MaxTtSize*C*=1<<(MinQtLog2Size*C*+tile_group_log2_diff_max_tt_min_qt_chroma) (7-41)
MinTtSize*Y*=1<<MinCbLog2Size*Y* (7-42)
MaxMttDepth*Y*=tile_group_max_mtt_hierarchy_depth_luma (7-43)
MaxMttDepth*C*=tile_group_max_mtt_hierarchy_depth_chroma (7-44)
(85) log2_ctu_size_minus2, log2_min_luma_coding_block_size_minus2 are signaled in SPS.
(86) log2_ctu_size_minus2 plus 2 specifies the luma coding tree block size of each CTU.
(87) log2_min_luma_coding_block_size_minus2 plus 2 specifies the minimum luma coding block size.
(88) The variables CtbLog2SizeY, CtbSizeY, MinCbLog2SizeY, MinCbSizeY, MinTbLog2SizeY, MaxTbLog2SizeY, MinTbSizeY, MaxTbSizeY, PicWidthInCtbsY, PicHeightInCtbsY, PicSizeInCtbsY, PicWidthInMinCbsY, PicHeightInMinCbsY, PicSizeInMinCbsY, PicSizeInSamplesY, PicWidthInSamplesC and PicHeightInSamplesC are derived as follows:
CtbLog2Size*Y*=log2_ctu_size_minus2+2 (7-7)
CtbSize*Y*=1<<CtbLog2Size*Y* (7-8)
MinCbLog2Size*Y*=log2_min_luma_coding_block_size_minus2+2 (7-9)
MinCbSize*Y*=1<<MinCbLog2Size*Y* (7-10)
MinTbLog2Size*Y*=2 (7-11)
MaxTbLog2Size*Y*=6 (7-12)
MinTbSize*Y*=1<<MinTbLog2Size*Y* (7-13)
MaxTbSize*Y*=1<<MaxTbLog2Size*Y* (7-14)
PicWidthInCtbs*Y*=Ceil(pic_width_in_luma_samples÷CtbSize*Y*) (7-15)
PicHeightInCtbs*Y*=Ceil(pic_height_in_luma_samples÷CtbSize*Y*) (7-16)
PicSizeInCtbs*Y*=PicWidthInCtbs*Y**PicHeightInCtbs*Y* (7-17)
PicWidthInMinCbs*Y*=pic_width_in_luma_samples/MinCbSize*Y* (7-18)
PicHeightInMinCbs*Y*=pic_height_in_luma_samples/MinCbSize*Y* (7-19)
PicSizeInMinCbs*Y*=PicWidthInMinCbs*Y**PicHeightInMinCbs*Y* (7-20)
PicSizeInSamples*Y*=pic_width_in_luma_samples*pic_height_in_luma_samples (7-21)
PicWidthInSamples*C*=pic_width_in_luma_samples/SubWidth*C* (7-22)
PicHeightInSamples*C*=pic_height_in_luma_samples/SubHeight*C* (7-23)
(89) [Ed. (BB): Currently the maximum transform size (64×64 luma samples and corresponding chroma sample size) and the minimum transform size (4×4 luma samples and corresponding chroma samples) is fixed, pending further specification development.]
(90) 2.4.2.1.2 Allowed Binary Split Process
(91) Inputs to this process are: a binary split mode btSplit, a coding block width cbWidth, a coding block height cbHeight, a location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture, a multi-type tree depth mttDepth, a maximum multi-type tree depth with offset maxMttDepth, a maximum binary tree size maxBtSize, a partition index partIdx.
(92) Output of this process is the variable allowBtSplit.
(93) TABLE-US-00002 TABLE 1 Specification of parallelTtSplit and cbSize based on btSplit btSplit == btSplit == SPLIT_BT_VER SPLIT_BT_HOR parallelTtSplit SPLIT_TT_VER SPLIT_TT_HOR cbSize cbWidth cbHeight
(94) The variables parallelTtSplit and cbSize are derived as specified in Table 1.
(95) The variable allowBtSplit is derived as follows: If one or more of the following conditions are

true, allowBtSplit is set equal to FALSE: //according to block size and maximum allowed MTT depth cbSize is less than or equal to MinBtSizeY cbWidth is greater than maxBtSize cbHeight is greater than maxBtSize mttDepth is greater than or equal to maxMttDepth Otherwise, if all of the following conditions are true, allowBtSplit is set equal to FALSE //according to picture boundary (no vertical BT for bottom picture boundary and bottom-right picture boundary) btSplit is equal to SPLIT_BT_VER y0+cbHeight is greater than pic_height_in_luma_samples Otherwise, if all of the following conditions are true, allowBtSplit is set equal to FALSE //according to picture boundary (no horizontal BT for right picture boundary) btSplit is equal to SPLIT_BT_HOR x0+cbWidth is greater than pic_width_in_luma_samples y0+cbHeight is less than or equal to pic_height_in_luma_samples Otherwise, if all of the following conditions are true, allowBtSplit is set equal to FALSE: //according to TT partition in above level (mttDepth−1) mttDepth is greater than 0 partIdx is equal to 1 MttSplitMode[x0][y0][mttDepth−1] is equal to parallelTtSplit //according to transform sizes (e.g., when MaxTbSizeY is equal to 64, for 64×128, no vertical BT; for 128×64, no horizontal BT) Otherwise if all of the following conditions are true, allowBtSplit is set equal to FALSE btSplit is equal to SPLIT_BT_VER cbWidth is less than or equal to MaxTbSizeY cbHeight is greater than MaxTbSizeY Otherwise if all of the following conditions are true, allowBtSplit is set equal to FALSE btSplit is equal to SPLIT_BT_HOR cbWidth is greater than MaxTbSizeY cbHeight is less than or equal to MaxTbSizeY Otherwise, allowBtSplit is set equal to TRUE.

2.4.2.1.3 Allowed Ternary Split Process

(96) Inputs to this process are: a ternary split mode ttSplit, a coding block width cbWidth, a coding block height cbHeight, a location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture, a multi-type tree depth mttDepth a maximum multi-type tree depth with offset maxMttDepth, a maximum binary tree size maxTtSize.

(97) Output of this process is the variable allowTtSplit.

(98) TABLE-US-00003 TABLE 2 Specification of cbSize based on ttSplit ttSplit == SPLIT_TT_VER ttSplit == SPLIT_TT_HOR cbSize cbWidth cbHeight

(99) The variable cbSize is derived as specified in Table 2.

(100) The variable allowTtSplit is derived as follows: If one or more of the following conditions are true, allowTtSplit is set equal to FALSE: //according to block size cbSize is less than or equal to 2*MinTtSizeY cbWidth is greater than Min(MaxTbSizeY, maxTtSize) cbHeight is greater than Min(MaxTbSizeY, maxTtSize) //according to maximum allowed MTT depth mttDepth is greater than or equal to maxMttDepth //according to whether it is located at picture boundary x0+cbWidth is greater than pic_width_in_luma_samples y0+cbHeight is greater than pic_height_in_luma_samples Otherwise, allowTtSplit is set equal to TRUE.

2.5 Partition Tree Structure in AVS3

(101) In AVS3, Extended Quad-tree (EQT) partitioning is adopted, which further extends the QTBT scheme and increases the partitioning flexibility. More specially, EQT splits a parent CU into four sub-CUs of different sizes, which can adequately model the local image content that cannot be elaborately characterized with QTBT. Meanwhile, EQT partitioning allows the interleaving with BT partitioning for enhanced adaptability.

(102) With the EQT partitioning, a parent CU is split into four sub-CUs with different sizes. As shown in FIG. **7**, EQT divides a M×N parent CU into two M×N/4 CUs and two M/2×N/2 CUs in the horizontal direction. Analogously, EQT vertical partitioning generates two N×M/4 CUs and two M/2×N/2 CUs. In particular, EQT sub-blocks size is always the power of 2, such that additional transformations are not necessarily involved.

(103) In the structure of QTBT, a QT splitting flag is first signaled to indicate whether the current CU is split by QT. As such, when this ag is false, the second signal will be encoded to denote whether the current CU splitting mode is non-splitting or BT splitting. For a BT splitting CU, the third bin (DIR) is signaled to discriminate horizontal BT or vertical BT splitting. When EQT

partitioning is introduced, one additional bin termed as is EQT is signaled to indicate whether it is an EQT-split, in case that BT and EQT are both available, as shown in FIG. **8**.
(104) 2.6 UQT
(105) Unsymmetrical Quad-Tree (UQT) partitioning is proposed in our P1809119401H. With UQT, a block with dimensions W×H is split into four partitions with dimensions W1×H1, W2×H2, W3×H3 and W4×H4, where W1, W2, W3, W4, H1, H2, H3, H4 are all integers. All the parameters are in the form of power of 2. For example, W1=2N1, W2=2N2, W3=2N3, W4=2N4, H1=2M1, H2=2M2, H3=2M3, H4=2M4. Some examples are shown in FIGS. **9**A-**9**F.
3. Drawbacks and Problems in Existing Systems
(106) Although the QT/BT/TT coding tree structure in VVC is quite flexible, there is still some partitioning patterns that cannot be attained by QT/BT/TT/EQT/UQT.
4. Exemplary Methods for Quinary Tree Partitioning
(107) To address the problem, several methods are proposed to introduce other kinds of partition structures that may split one block to more than 4 partitions.
(108) The detailed inventions below should be considered as examples to explain general concepts. These embodiments should not be interpreted in a narrow way. Furthermore, these embodiments can be combined in any manner.
(109) In the following discussion, partition trees may indicate QT, BT, TT or Unsymmetrical Quad-Tree (UQT), EQT or others. While partition/splitting directions may indicate the horizontal splitting or vertical splitting or diagonal splitting or others. One partition is denoted by its partition tree type and partition direction.
(110) QT, BT, TT, UQT, or EQT may refer to "QT split", "BT split", "TT split", "UQT split", "EQT split", respectively.
(111) In the following discussion, "split" and "partitioning" have the same meaning. The proposed methods may be also applicable to existing partition trees.
(112) Definitions of Proposed Partition Types
(113) 1. Quinary-Tree (QUI-T) partitioning is proposed. With QUI-T, a block with dimensions W×H is split into five smaller blocks. When the indication of using such partitioning is true, such a block is directly split into five smaller ones (a.k.a. split child blocks). The smaller one may be treated as a coding unit/a prediction unit/a transform unit. Each dimension of the smaller block may be denoted by W.sub.i×H.sub.i (i being 0 . . . 4, indicating the partition index) and W.sub.i, H.sub.i are all integers. a. In one example, each smaller block may be further split into even smaller blocks, such as in a recursive way. b. In one example, all the dimensions are in the form of power of 2. i. For example, W0=2N0, W1=2N1, W2=2N2, W3=2N3, W4=2N4, H0=2M0, H1=2M1, H2=2M2, H3=2M3, H4=2M4. c. In one example, QUI-T split one block in both horizontal and vertical directions. Such case is named as mixed direction. FIGS. **10**A, **10**B and **10**D give some examples. i. For example, at least one of Wi is unequal to W. ii. For example, at least one of Hi is unequal to H. d. In one example, QUI-T only splits one block in vertical direction. i. For example, H0=H1=H2=H3=H4=H. FIG. **10**C gives an example. e. In one example, QUI-T only splits one block in horizontal direction. i. For example, W0=W1=W2=W3=W4=W. FIG. **10**E gives an example. f. In one example, one of the partitions (with size equal W.sub.x×H.sub.x) to has different block sizes compared to others. i. In one example, the other four partitions are with same sizes. ii. In one example, W.sub.x=½ W and W.sub.y=⅛ W (y!=x) with x being a value within the range [0, 4]. Alternatively, furthermore, x is equal to 1 or 3. 1. FIG. **10**C gives an example wherein W.sub.3=½ W and W.sub.y=⅛ W (y!=3). iii. In one example, H.sub.x=½ H and H.sub.y=⅛ H (y!=x). Alternatively, furthermore, x is equal to 1 or 3. iv. In one example, W.sub.x=W−((W/5)<<2) and W.sub.y=W/5 (y!=x) with x being a value within the range [0, 4]. v. In one example, W.sub.x=W−((W/M)<<2) and W.sub.y=W/M (y!=x) wherein M is an integer value, such as 8, 16, 32, 64. g. In one example, two of the partitions may have equal size (with size equal W.sub.x×H.sub.x), the others are with equal sizes but different from these two. i. In one example,

W.sub.x=⅛ W and W.sub.y=¼ W (for all y that y!=x). Alternatively, furthermore, x being equal to 0 and 3. Alternatively, furthermore, x being equal to 0 and 4. Alternatively, furthermore, x being equal to 1 and 3. Alternatively, furthermore, x being equal to 1 and 4. 1. FIG. **10**C gives an example wherein W.sub.1=W.sub.3=⅛ W and W.sub.y=¼ W (y!=3 && y!=1). ii. In one example, H.sub.x=⅛ H and H.sub.y=¼ H (for all y that y!=x). Alternatively, furthermore, x being equal to 0 and 3. Alternatively, furthermore, x being equal to 0 and 4. Alternatively, furthermore, x being equal to 1 and 3. Alternatively, furthermore, x being equal to 1 and 4. h. In one example, two of the partitions may have equal size (with size equal W.sub.x×H.sub.x), the other three may have different sizes. i. In one example, W.sub.x= 1/16 W for the two with equal sizes; and W.sub.i= 2/16 W; W.sub.3= 4/16 W and W.sub.k= 8/16 W, wherein i, j, k are unequal to x. 1. FIG. **10**C gives an example wherein W.sub.0=W4= 1/16 W; W.sub.1= 2/16 W; W.sub.2= 4/16 W and W.sub.3= 8/16 W ii. In one example, H.sub.x= 1/16H for the two with equal sizes; and H.sub.i= 2/16W; H.sub.j= 4/16 W and H.sub.k= 8/16 W, wherein i, j, k are unequal to x. iii. Alternatively, two of the five partitions may have equal size (with size equal W.sub.x×H.sub.x), two of the remaining three are with the same size. i. In the above and below description, 1/16 W or ⅛ W mean 1/16×W or ⅛×W, or noted as W/16 or W/8. j. In one example, W.sub.0=W.sub.4=W/8 and W.sub.1=W.sub.2=W.sub.3=W/4, H.sub.0=H.sub.1=H.sub.2=H.sub.3=H.sub.4=H. k. In one example, H.sub.0=H.sub.4=H/8 and H.sub.1=H.sub.2=H.sub.3=H/4, W.sub.0=W.sub.1=W.sub.2=W.sub.3=W.sub.4=W. l. In one example, one or multiple splits are not allowed for one or more child blocks when the current block is split by quinary-split. i. For example, if the current block is split by quinary-split into five child-block: B0: W/8×H, B1: W/4×H, B2: W/4×H, B3: W/4×H, B4: W/8×H, then: 1. In one example, Bx is not allowed to be split by vertical BT, where x may be one or some of 0, 1, 2, 3, 4; 2. In one example, Bx is not allowed to be split by vertical TT, where x may be one or some of 0, 1, 2, 3, 4; 3. In one example, Bx is not allowed to be split by horizontal BT, where x may be one or some of 0, 1, 2, 3, 4; 4. In one example, Bx is not allowed to be split by horizontal TT, where x may be one or some of 0, 1, 2, 3, 4; 5. In one example, Bx is not allowed to be split by QT, where x may be one or some of 0, 1, 2, 3, 4; ii. For example, if the current block is split by quinary-split into five child-block: B0: W×H/8, B1: W×H/4, B2: W×H/4, B3: W×H/4, B4: W×H/8, then: 1. In one example, Bx is not allowed to be split by horizontal BT, where x may be one or some of 0, 1, 2, 3, 4; 2. In one example, Bx is not allowed to be split by horizontal TT, where x may be one or some of 0, 1, 2, 3, 4; 3. In one example, Bx is not allowed to be split by vertical BT, where x may be one or some of 0, 1, 2, 3, 4; 4. In one example, Bx is not allowed to be split by vertical TT, where x may be one or some of 0, 1, 2, 3, 4; 5. In one example, Bx is not allowed to be split by QT, where x may be one or some of 0, 1, 2, 3, 4; m. The above methods may be extended to other Senary-, Septenary-, Octonary-Tree partitions (SnT, StT, OctT) wherein one block may be split to 6, 7, or 8 smaller blocks. Examples of senary-tree partitions are shown in FIGS. **11**A and **11**B. n. Some exemplary partitions are depicted in FIGS. **10**A-**10**E. o. The coding order (denoted by PIdx 0 . . . 4) may be different from that defined in FIGS. **10**A-**10**E. i. The coding order for one QUI-T pattern may be pre-defined. ii. Alternatively, multiple coding orders may be pre-defined for one QUI-T pattern, and one block may choose one from them, such as via signaling the indication of selected coding order or derivation at the decoder side.

Interaction with Other Partition Types 2. A block which is split into child blocks by QUI-T, may be split from a parent block by one or some specific kinds of split methods. a. A block which may allow QUI-T partitions, may be a block generated by QT or BT or TT or QUI-T partitions. b. For example, a block which is split into child blocks by QUI-T, can only be split from a parent block by QT. c. A block which may allow QUI-T partitions, may be a root block. 3. A block which is split from a parent block by QUI-T, may be further split into child blocks by one or multiple other partition types (such as QT, BT, TT, QUI-T, UQT). a. For example, a block which is split from a parent block by QUI-T, may be further split into child blocks by BT and/or TT. b. For example, a block which is split from a parent block by QUI-T, may be further split into child blocks by BT and/or TT, and/or QUI-T, but not QT. c. For example, a block which is split from a parent block by QUI-T, may be further split into child blocks by QUI-T and/or QT, but not BT/TT. d. For example, a block which is split from a parent block by QUI-T, cannot be further split into child blocks by QT. e. Alternatively, QUI-T split blocks may be not further split into child blocks. 4. When a parent block is split into child blocks by QUI-T, the split depth of the child block may be derived from the split depth of the parent block. a. In one example, the splitting due to QUI-T may be used to update the QT/BT/TT/QUI-T/MTT depth. i. In one example, the QT depth of one or all of the child blocks is equal to the QT depth of the parent block added by 1. ii. In one example, the BT depth of one or all of the child blocks is equal to the BT depth of the parent block added by 1. iii. In one example, the TT depth of one or all of the child blocks is equal to the TT depth of the parent block added by 1. iv. In one example, the QUI-T depth of one or all of the child blocks is equal to the QUI-T depth of the parent block added by 1. v. In one example, the MTT depth of one or all of the child block is equal to the MTT depth of the parent block added by 1. 1. For example, the MTT depth of the child block is equal to the MTT depth of the parent block added by 1 if the parent block is split into child blocks by BT. 2. For example, the MTT depth of the child block is equal to the MTT depth of the parent block added by 1 if the parent block is split into child blocks by TT. b. In one example, the QUI-T/BT/TT/QT/MTT depth increasement for different child block may be different. i. The depth increasement is dependent on the ratio of a child block compared to the parent block. Restrictions of Usage of QUI-T 5. In one example, the maximum/minimum block size that could allow QUI-T partitions and/or the maximum bit depth and/or maximum depth that could allow QUI-T partitions may be signaled in SPS/PPS/VPS/APS/sequence header/picture header/slice header/tile group header/CTU row/regions, etc. a. The maximum/minimum block size that could allow QUI-T partitions and/or the maximum depth that could allow QUI-T partitions may be derived from other values, such as depth for MTT or depth of QT. b. The maximum block that allows QUI-T partitions, may be the largest coding block (coding tree block or coding tree unit). c. For example, the maximum block that allows QUI-T partitions, may be the virtual pipeline data unit (VPDU). d. In one example, the maximum/minimum block size that could allow QUI-T partitions and/or the maximum depth that could allow QUI-T partitions may be dependent on profile/level/tier of a standard. e. In one example, the maximum/minimum block size that could allow QUI-T partitions and/or the maximum depth that could allow QUI-T partitions may be derived, such as to be the same as that for QT partitions. f. In one example, the maximum/minimum block size that could allow QUI-T partitions and/or the maximum depth that could allow QUI-T partitions may be dependent on tile group tile/slice type/color component/dual tree is enabled or not. g. In one example, the maximum/minimum block size that could allow QUI-T partitions and/or the maximum depth that could allow QUI-T partitions may be different for different QUI-T patterns. h. When one block is split according to QUI-T, the corresponding depth of QUI-T of one smaller block may be adjusted (e.g., increased by 1) accordingly. i. Alternatively, the corresponding depth of a certain partition (e.g., QT) of one smaller block may be adjusted (e.g., increased by 1) accordingly. ii. Alternatively, the corresponding depth of MTT of one smaller block may be adjusted (e.g., increased by 1) accordingly. iii. The adjustment of corresponding depth of different smaller blocks may be done in the same way (e.g., increase by 1) 1. Alternatively, the adjustment of corresponding depth of different smaller blocks may be done in the different way (e.g., increase by 1). For example, the adjustment is dependent on block dimension of the smaller block. 6. QUI-T is not allowed if a split child block cross more than one Virtual pipeline data units (VPDUs). a. Alternatively, QUI-T is still allowed, however, such child block is forced to be further split until no child block crosses more than one VPDU. 7. QUI-T is not allowed if the width/height of the current block (or any of the split child block) satisfy some conditions. (Suppose the width and height of the current block are W and H, T1, T2 and T are some integers) a. QUI-T is not allowed if W>=T1 and H>=T2; b. QUI-T is not allowed if W>=T1 or H>=T2; c. QUI-T is not allowed if W<=T1 and

H<=T2; d. QUI-T is not allowed if W<=T1 or H<=T2; e. QUI-T is not allowed if W×H<=T; f. QUI-T is not allowed if W×H>=T; g. Horizontal QUI-T is not allowed if H<=T; For example, T=16. h. Horizontal QUI-T is not allowed if H>=T; For example, T=128. i. Vertical QUI-T is not allowed if W<=T; For example, T=16. j. Vertical QUI-T is not allowed if W>=T; For example, T=128. k. T1, T2 and T may be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile header. l. T1, T2 and T may depend on color components. For example, T1, T2 and T may be different for luma and chroma components. i. In one example, the signaled thresholds such as T1, T2 and/or T may be shared by QUI-T and TT. ii. In one example, the signaled thresholds such as T1, T2 and/or T may be shared by QUI-T and BT. m. T1, T2 and T may depend on whether luma coding tree and chroma coding tree are separated. For example, T1, T2 and T may be different for luma and chroma components if luma coding tree and chroma coding tree are separated. n. Alternatively, when the transform is not supported for at least one child block due to QUI-T, QUI-T split is invalid. o. Alternatively, when the depth of one block exceeding the allowed depth for QUI-T splitting, QUI-T split is invalid. p. Alternatively, when any of a child block size is smaller than the allowed block size due to QUI-T splitting, QUI-T split is invalid. 8. QUI-T is allowed if the width/height of the current block (or any of the split child block) satisfy some conditions. (Suppose the width and height of the current block are W and H, T1, T2 and T are some integers) a. QUI-T is allowed if W>=T1 and H>=T2; b. QUI-T is allowed if W>=T1 or H>=T2; c. QUI-T is allowed if W<=T1 and H<=T2; d. QUI-T is allowed if W<=T1 or H<=T2; e. QUI-T is allowed if W×H<=T; f. QUI-T is allowed if W×H>=T; g. Horizontal QUI-T is allowed if H<=T; For example, T=64. h. Horizontal QUI-T is allowed if H>=T; For example, T=32. i. Vertical QUI-T is allowed if W<=T; For example, T=64. j. Vertical QUI-T is allowed if W>=T; For example, T=32. k. T1, T2 and T may be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile header. i. In one example, the signaled thresholds such as T1, T2 and/or T may be shared by QUI-T and TT. ii. In one example, the signaled thresholds such as T1, T2 and/or T may be shared by QUI-T and BT. l. T1, T2 and T may depend on color components. For example, T1, T2 and T may be different for luma and chroma components. m. T1, T2 and T may depend on whether luma coding tree and chroma coding tree are separated. For example, T1, T2 and T may be different for luma and chroma components if luma coding tree and chroma coding tree are separated. 9. QUI-T is not allowed if the depth of the current block satisfy some conditions. The depth of the current block may refer to QT depth, BT depth, TT depth, QUI-T depth or MTT depth. a. QUI-T is not allowed if the split depth<=T; b. QUI-T is not allowed if the split depth>=T; c. QUI-T is not allowed if the QT split depth<=T; d. QUI-T is not allowed if the QT split depth>=T; e. QUI-T is not allowed if the BT split depth>=T; f. QUI-T is not allowed if the BT split depth<=T; g. QUI-T is not allowed if the TT split depth>=T; h. QUI-T is not allowed if the TT split depth>=T; i. QUI-T is not allowed if the QUI-T split depth<=T; j. QUI-T is not allowed if the QUI-T split depth>=T; k. QUI-T is not allowed if the MTT split depth<=T; l. QUI-T is not allowed if the MTT split depth>=T; m. T may be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile header. n. T may depend on color components. For example, T1, T2 and T may be different for luma and chroma components. o. T may depend on whether luma coding tree and chroma coding tree are separated. For example, T1, T2 and T may be different for luma and chroma components if luma coding tree and chroma coding tree are separated. 10. QUI-T is allowed if the depth of the current block satisfy some conditions. The depth of the current block may refer to QT depth, BT depth, TT depth, QUI-T depth or MTT depth. a. QUI-T is allowed if the split depth<=T; b. QUI-T is allowed if the split depth>=T; c. QUI-T is allowed if the QT split depth<=T; d. QUI-T is allowed if the QT split depth>=T; e. QUI-T is allowed if the BT split depth>=T; f. QUI-T is allowed if the BT split depth<=T; g. QUI-T is allowed if the TT split depth>=T; h. QUI-T is allowed if the TT split depth>=T; i. QUI-T is allowed if the QUI-T split depth<=T; j. QUI-T is allowed if the QUI-T split depth>=T; k. QUI-T is allowed if the MTT split depth<=T; l. QUI-T is allowed if the MTT split depth>=T; m. T may be signaled from the encoder to the decoder in VPS/SPS/PPS/picture header/slice header/tile group header/tile header. n. T may depend on color components. For example, T1, T2 and T may be different for luma and chroma components. o. T may depend on whether luma coding tree and chroma coding tree are separated. For example, T1, T2 and T may be different for luma and chroma components if luma coding tree and chroma coding tree are separated. 11. Whether and how to use QUI-T may depend on the position of the current block. For example, whether and how to use QUI-T may depend on the whether the current block crosses the picture/tile/tile group border or not. a. In one example, vertical QUI-T is not allowed if the current block crosses the picture/tile/tile group bottom border. b. In one example, horizontal QUI-T is not allowed if the current block crosses the picture/tile/tile group bottom border. c. In one example, vertical QUI-T is not allowed if the current block crosses the picture/tile/tile group right border. d. In one example, horizontal QUI-T is not allowed if the current block crosses the picture/tile/tile group right border. e. In one example, mixed QUI-T may be not allowed if the current block crosses the picture/tile/tile group right border. f. In one example, mixed QUI-T may be not allowed if the current block crosses the picture/tile/tile group bottom border. g. In one example, if a child block split by QUI-T is totally out of the picture/tile/tile group, the child block may be omitted in the encoding/decoding process. h. In one example, if a child block split by QUI-T is partially out of the picture/tile/tile group, the following may apply i. The part out of the picture may be omitted in the encoding/decoding process. ii. The part inside the picture may be further split. iii. The part inside the picture may be coded as a CU. 1. Whether the part inside the picture is coded as a CU may depend on the width (w) and height (h) of the part. a. In one example, the part inside the picture may be coded as a CU if w=2n.sub.w, h=2n.sub.h, where n.sub.w and n.sub.h are integers. i. In one example, if any child block split by QUI-T is partially/fully out of the picture/tile/tile group, QUI-T is disallowed. 12. When QUI-T or certain QUI-T pattern is disallowed, the signaling of indication of the usage of the pattern may be also skipped. a. Alternatively, it may be still signaled but is constrained to be false in a conformance bitstream. 13. When a child block is split from QUI-T, the child block may not be allowed to be further split with one or more splitting methods as: a. QT b. horizontal BT c. vertical BT d. horizontal TT e. vertical TT f. horizontal UQT g. vertical UQT h. QUI-T It is proposed to QUI-T may be only applied to the leaf nodes, e.g., when one block is not further split according to other partitions. i. In one example, a flag may be signaled for the leaf node whether to use QUI-T or not. i. Alternatively, furthermore, indications of which kind QUI-T may be further signaled. j. Alternatively, indications of disabling QUI-T or which kind QUI-T may be signaled for the leaf node.

Indications of usage of QUI-T 14. Whether to apply QUI-T and/or which kind QUI-T is applied may be signaled from encoder to decoder. a. In one example, it may be signaled in VPS/SPS/PPS/sequence header/picture header/slice header/tile group header/tile header to indicate whether QUI-T can be applied. b. In one example, it may be signaled in VPS/SPS/PPS/sequence header/picture header/slice header/tile group header/tile header to indicate which kinds of QUI-T can be applied. c. In one example, it may be signaled in a block to indicate whether QUI-T is used to split that block. d. In one example, it may be signaled in a block to indicate which kind of QUI-T is used to split that block. e. In one example, different QUI-T sets may be designed for different block shapes/sizes. f. In one example, different QUI-T sets may be designed for pictures/tiles/slices with different temporal layers. g. In one example, whether or how to apply QUI-T may depend on the video resolution/picture resolution/coded modes/video characteristics (screen content or camera captured sequence or mixed content)/slice type/picture type/tile group type/low delay check flag. 15. One syntax element may be signaled to indicate no split or partition (including partition tree type and split directions). a. Alternatively, one syntax element may be firstly signaled to indicate whether to split or not; and another syntax element may be signaled to indicate the partition. 16. Indication of partition may be represented by two syntax element: selected partition tree type may be firstly signaled, followed by splitting direction if needed. a. In one example, an index of partition tree type may be signaled in a block to indicate whether a block is split by QT, or QUI-T or non-split. i. Alternatively, furthermore, the splitting direction (horizonal/vertical/mixed direction) and/or splitting patterns may be further signaled. b. In one example, an index of partition tree type may be signaled in a block to indicate whether a block is split by BT, or TT, or QUI-T. i. For example, this index may be conditionally signaled, such as only when at least one of BT, TT and QUI-T is valid for this block. ii. Alternatively, furthermore, the splitting direction (horizonal/vertical) and/or splitting patterns may be further signaled. c. Alternatively, indication of splitting direction may be firstly signaled, followed by partition tree type (such as QT, TT, QUI-T). i. In one example, a flag is signaled in a block to indicate whether a block is vertical split or horizontal split. The vertical split may be BT vertical split, TT vertical split or QUI-T vertical split. The horizontal split may be BT horizontal split, TT horizontal split or QUI-T horizontal split. ii. For example, this flag is signaled only when the block is split by BT, or TT, or QUI-T. iii. For example, this flag is signaled only when both vertical split and horizontal split are valid for this block. 1. If only vertical split is valid, the flag is not signaled, and horizontal split is inferred to be used. 2. If only horizontal split is valid, the flag is not signaled, and vertical split is inferred to be used. d. In one example, a binarized code is signaled in a block to indicate which kind of split (BT, TT, or a kind of QUI-T) is used. In following examples, X represents 0 or 1 and Y=X (Y=1 if X=0 and Y=0 if X=1). i. In one example, the candidate BT, TT or QUI-Ts to be signaled are all vertical splits or horizontal splits depending on previously signaled or derived information. ii. In one example, a first flag is signaled to indicate whether QUI-T is used. For example, the binarized codewords orderly to represent BT, TT, QUI-T1, QUI-T2, QUI-T3 and QUI-T4 are XX, XY, YXX, YXY, YYX, YYY. 1. In an alternative example, the binarized codewords orderly to represent BT, TT, QUI-T1 are XX, XY, Y. iii. In one example, truncated unary code is applied. For example, the binarized codewords orderly to represent BT, TT, QUI-T1, QUI-T2, QUI-T3 and QUI-T4 are X, YX, YYX, YYYX, YYYYX, YYYYY. iv. In one example, a first flag is signaled to indicate whether BT is used. If BT is not used, then a second flag is signaled to indicate whether QUI-T is used. If QUI-T is used, which kind of QUI-T is used is further signaled. For example, the binarized codewords orderly to represent BT, TT, QUI-T1, QUI-T2, QUI-T3 and QUI-T4 are X, YX, YYXX, YYXY, YYYX, YYYY. 17. In one example, how to signal which kind of partitions is used in a block may depend on which kinds of partitions (including partition tree type and/or partition directions) are valid for the block. In following examples, X represents 0 or 1 and Y=X (Y=1 if X=0 and Y=0 if X=1). a. In one example, the candidate BT, TT or QUI-Ts to be signaled are all vertical splits or horizontal splits depending on previously signaled or derived information. b. For example, the non-allowed or invalid split cannot be signaled from the encoder to the decoder, i.e. there is no codeword to represent the non-allowed or invalid split. c. In one example, if there is only one kind of split from BT, TT and QUI-Ts is valid, then the binarized code to indicate which kind of split (BT, TT, or a kind of QUI-T) is used is not signaled. d. In one example, if there are only two kinds of split from BT, TT and QUI-Ts are valid, then a flag is signaled to indicate which one of the two valid splits is used. e. In one example, the code to indicate which kind of split (BT, TT, or a kind of QUI-T) is binarized as a truncated unary code. i. For example, the maximum value of the truncated unary code is N−1, where N is the number of valid splits (BT, TT and QUI-Ts). ii. For example, no codeword represents an invalid split. In other words, the invalid split is skipped when building the codeword table. f. In one example, if no QUI-T is valid, the flag indicating whether QUI-T is used is not signaled and inferred to be false. For example, the binarized codewords orderly to represent BT and TT are X and Y. g. In one example, if only one kind of QUI-T is valid and QUI-T is signaled to be used, then no further information is signaled to indicate which QUI-T is used. The valid QUI-T is used implicitly. h. In one example, if only two kinds of QUI-T are valid and QUI-T is signaled to be used, then a flag is signaled to indicate which QUI-T is used. i. In one example, if only three kinds of QUI-T are valid and QUI-T is signaled to be used, then a message is signaled to indicate which QUI-T is used. For example, the binarized codewords

orderly to represent the three QUI-Ts are X, YX, YY. j. In one example, the binarization and/or signaling method is not changed according to which kinds of split is valid in the block. An invalid split cannot be chosen in a conformance bit-stream. 18. Indications of partition may be coded by arithmetic coding with one or multiple contexts. a. In one example, only partial bins of a bin string may be coded with contexts and remaining bins may be coded with bypass mode (i.e., no context is utilized). b. Alternatively, all bins of a bin string may be coded with contexts. c. Alternatively, all bins of a bin string may be coded with bypass mode. d. For a bin coded with context, one or multiple contexts may be used. e. The context may depend on: i. The position or index of the bin. ii. The partitioning of spatial/temporal neighboring blocks. iii. The current partition depth (e.g., QT depth/BT depth/TT depth/QUI-T depth/MTT depth) of current block. iv. The partition depth (e.g., QT depth/BT depth/TT depth/QUI-T depth/MTT depth) of spatial/temporal neighboring blocks and/or spatial/temporal non-adjacent blocks. v. The coding modes of spatial/temporal neighboring blocks. vi. The width/height of spatial/temporal neighboring blocks. vii. The width/height of the current block viii. Slice types/picture types/tile group type ix. Color component x. Statistical results of partition types from previously coded blocks 19. Whether and/or how to use QUI-T may depend on color format (such as 4:4:4 or 4:2:0) and/or color components. a. Whether and how to use QUI-T may depend on whether luma and chroma coding trees are separated. b. In one example, QUI-T can only be applied on luma component when luma and chroma coding trees are separated. 20. The above methods may be also applicable to SnT, StT, OctT, UQT.
(114) The examples described above may be incorporated in the context of the method described below, e.g., method **1300**, which may be implemented at a video decoder/encoder.
(115) FIG. **13** illustrates a flowchart of an exemplary method for video processing. The method **1300** comprises, at step **1310**, determining, for a current video block, whether a first partition mode is applicable to the current video block in responsive to at least one condition, wherein the current video block is split into M sub-blocks in the first partition mode, and M>4; and at step **1320**, performing a conversion for the current video block based on the determination.
(116) FIG. **14** illustrates a flowchart of an exemplary method for video processing. The method **1400** comprises: determining, at step **1410**, for a video block, whether and/or how to apply a first partition mode to the video block based on an indication, wherein the video block is split into M portions in the first partition mode, M>4; and performing a conversion of the video block based on the determination.
(117) Some embodiments and techniques related to methods **1300** and **1400** may be described using the following examples.
(118) In an example, there is disclosed a method for video processing, comprising: determining, for a current video block, whether a first partition mode is applicable to the current video block in responsive to at least one condition, wherein the current video block is split into M sub-blocks in the first partition mode, and M>4; and performing a conversion for the current video block based on the determination.
(119) In an example, M=5, and the current video block is split into five sub-blocks using Quinary-Tree(QUI-T) in the first partition mode.
(120) In an example, the at least one condition depends on at least one of a maximum block size, a minimum block size, and a maximum depth allowed for the first partition mode.
(121) In an example, the maximum depth comprises at least one of a maximum bit depth and a maximum split depth.
(122) In an example, the at least one condition is determined from an indication signaled in a video unit level wherein the video unit comprises at least one of sequence, video, picture, slice, tile group, a coding tree unit (CTU) row or a CTU region.
(123) In an example, the at least one condition is determined from an indication signaled in at least one of a sequence parameter set (SPS), a video parameter set (VPS), a picture parameter set (PPS), an adaptation parameter set (APS), a sequence header, a picture header, a slice header, a tile group

header.
(124) In an example, the at least one condition is derived from at least one of a depth of a multiple-type tree (MTT) or a depth of quadtree(QT).
(125) In an example, the maximum block size is a size of a largest coding block/unit or a size of a virtual pipeline data unit (VPDU).
(126) In an example, the largest coding block/unit is a coding tree block/unit (CTB/CTU).
(127) In an example, the at least one condition depends on at least one of a profile, level and tier of a coding standard.
(128) In an example, at least one of the maximum block size, minimum block size, and maximum depth is derived in a same way as that of QT partition mode.
(129) In an example, the at least one condition depends on at least one of a tile group, a tile, a slice type, a color component and an activation of a dual tree
(130) In an example, the first partition mode has a plurality of partition patterns by which one block can be split into M sub-blocks in different ways, and the at least one condition differs between different partition patterns.
(131) In an example, all sub-blocks are leaf nodes, and the method further comprises: splitting at least one sub-block into a plurality of portions singly or recursively.
(132) In an example, the at least one sub-block is split in at least one of the first partition mode, a QT partition mode and an MTT partition mode.
(133) In an example, a depth of the at least one sub-block is adjusted based on a depth of the current video block.
(134) In an example, the depth of the at least one sub-block is equal to the depth of the current video block plus 1.
(135) In an example, depths of all sub-blocks are adjusted in a same way or depths of different sub-blocks are adjusted in different ways.
(136) In an example, a depth of the at least one sub-block is adjusted based on a size of the at least one sub-block.
(137) In an example, the at least one condition depends on a position of each sub-block.
(138) In an example, the first partition mode is not applicable to the current video block if at least one sub-block crosses a border of a VPDU.
(139) In an example, the method further comprises: splitting any sub-block which crosses a border of a VPDU into a plurality of portions singly or recursively until no portion crosses the border of the VPDU.
(140) In an example, the at least one condition depends on a dimension of the current video block.
(141) In an example, the first partition mode is applicable to the current video block if the dimension of the current video block satisfies at least one of: $W>=T1$; $H>=T2$; $W\times H>=T3$; wherein W, H represent a width and height of the current video block respectively, and T1, T2 and T3 represents first to third thresholds respectively.
(142) In an example, the first partition mode is applicable to the current video block if the dimension of the current video block satisfies at least one of: $W<=T1'$; $H<=T2'$; $W\times H<=T3'$; wherein W, H represent a width and height of the current video block respectively, and T1′, T2′ and T3′ represents first to third thresholds respectively.
(143) In an example, the first partition mode is applicable to the current video block in a horizontal direction if the dimension of the current video block satisfies at least one of: $H<=T4$; $H>=T5$; wherein T4 and T5 represents fourth and fifth thresholds respectively.
(144) In an example, $T4=64$, and $T5=32$.
(145) In an example, the first partition mode is applicable to the current video block in a vertical direction if the dimension of the current video block satisfies at least one of: $W<=T6$; $W>=T7$; wherein T6 and T7 represents sixth and seventh thresholds respectively.
(146) In an example, $T6=64$, and $T7=32$.

(147) In an example, the first partition mode is not applicable to the current video block in a horizontal direction if the dimension of the current video block satisfies at least one of: H<=16; H>=128.
(148) In an example, the first partition mode is not applicable to the current video block in a vertical direction if the dimension of the current video block satisfies at least one of: W<=16; W>=128.
(149) In an example, the at least one condition depends on a split depth of the current video block.
(150) In an example, the first partition mode is applicable to the current video block if the split depth of the current video block satisfies at least one of: D<=D1; D>=D2; wherein D, D1 and D2 represent the split depth of the current video block, a first depth threshold, and a second depth threshold respectively.
(151) In an example, the split depth of the current video block comprises at least one of QT split depth, binary tree (BT) split depth, ternary tree(TT) split depth, MTT split depth and split depth in the first partition mode.
(152) In an example, the at least one condition depends on a position of the current video block.
(153) In an example, the first partition mode is not applicable to the current video block in at least one direction if the current video block crosses a border of at least one of a picture, tile, and tile group comprising the current video block.
(154) In an example, the border comprises at least one of a bottom border and a right border.
(155) In an example, the at least one direction comprises a vertical direction, a horizontal direction, and a mixed direction including both vertical and horizontal directions.
(156) In an example, the method further comprises: skipping any sub-block, which is located outside one of a picture, tile, and tile group comprising the current video block, in a subsequent conversion.
(157) In an example, if any sub-block has first and second portions which are located outside and inside one of a picture, tile and tile group respectively, the method comprises skipping the first portion in a subsequent conversion.
(158) In an example, the second portion is split into a plurality of sub-portions.
(159) In an example, the second portion is converted as a coding unit.
(160) In an example, at least one of width and height of the second portion is equal to a power of 2.
(161) In an example, the first partition mode is not applicable to the current video blocks if any sub-block is partial or fully outside at least one of a picture, tile, and tile group.
(162) In an example, the first partition mode is not applicable to the current video block if one of the following is satisfied: the maximum depth allowed for the first partition mode is reached for at least one sub-block; the minimum block size allowed for the first partition mode is reached for at least one sub-block; a block size for which a transform can be supported is reached for at least one sub-block;
(163) In an example, at least one of thresholds is signaled in a sequence parameter set (SPS), a video parameter set (VPS), a picture parameter set (PPS), a picture header, a slice header, a tile group header or a tile header.
(164) In an example, at least one of thresholds is signaled in a video unit level wherein the video unit comprises at least one of sequence, video, picture, slice, tile group, a coding tree unit (CTU) row or a CTU region.
(165) In an example, the signaled at least one of thresholds is shared by the ternary tree (TT) partition and the first partition mode or shared by the binary tree(BT) partition and the first partition mode.
(166) In an example, at least one threshold depends on sample components of the current video block.
(167) In an example, the sample components comprise at least one of color components, luma component and chroma components.

(168) In an example, at least one threshold differs between the luma component and the chroma components if luma and chroma coding trees of the current video block are separated from each other.
(169) In an example, at least one of the following partition modes is not available for at least one of sub-blocks: QT partition; BT partition in a horizontal direction; BT partition in a vertical direction; TT partition in a horizontal direction; TT partition in a vertical direction; unsymmetrical quadtree(UQT) partition in a horizontal direction; UQT partition in a vertical direction; and the first partition mode.
(170) In an example, the at least one condition depends on whether the current video block belongs to a leaf node to which any other partition modes is not applicable.
(171) In an example, the method further comprises: determining a first indication indicating whether the first partition mode is applicable to the leaf node.
(172) In an example, the first partition mode has one or more partition patterns, and the method comprises: determining a second indication indicating which partition pattern is used.
(173) In an example, the at least one condition depends on at least one of color format, luma component and chroma components.
(174) In an example, the first partition mode is only applicable to luma components of the current video block if luma and chroma coding trees of the current video block are separated from each other.
(175) In another aspect, there is disclosed a method for video processing, comprising: determining, for a video block, whether and/or how to apply a first partition mode to the video block based on an indication, wherein the video block is split into M portions in the first partition mode, M>4; and performing a conversion of the video block based on the determination.
(176) In an example, M=5, and the video block is split into five portions using Quinary-Tree(QUI-T) in the first partition mode.
(177) In an example, the first partition mode has one or more partition patterns.
(178) In an example, the indication is signaled in one of a sequence parameter set (SPS), a video parameter set (VPS), a picture parameter set (PPS), a sequence header, a picture header, a slice header, a tile group header or a tile header.
(179) In an example, the indication is signaled in a video unit level wherein the video unit comprises at least one of sequence, video, picture, slice, tile group, a coding tree unit (CTU) row or a CTU region
(180) In an example, the indication is signaled in a bitstream representation of the video block.
(181) In an example, multiple partition patterns are designed based on a shape or size of the video block.
(182) In an example, multiple partition patterns are designed based on one of pictures, tiles and slices with different temporal layers.
(183) In an example, the indication depends on at least one of a video resolution, picture resolution, coding mode, video content, slice type, picture type, tile group type, low delay check flag.
(184) In an example, the video content comprises at least one of a screen content, a camera captured sequence or mixed content.
(185) In an example, the indication is represented by one or more syntax elements, wherein the one or more syntax elements comprise a first syntax element which indicates whether a split is performed on the video block.
(186) In an example, the one or more syntax elements further comprises a second syntax element which indicates a partition tree and a partition direction to be used in the split.
(187) In an example, the indication is represented by one or more syntax elements, wherein the one or more syntax elements comprises a first syntax element which indicates an index of a type of a partition tree to be applied to the video block.
(188) In an example, the type of the partition tree belongs to at least one of: BT, TT, QT, the first

partition and non-split.
(189) In an example, the one or more syntax elements further comprises a second syntax element which indicates at least one of a partition direction and a partition pattern.
(190) In an example, the first syntax element is signaled only if a corresponding partition tree is valid for the video block.
(191) In an example, the partition direction comprises at least one of a vertical direction, a horizontal direction, and a mixed direction including both vertical and horizontal directions.
(192) In an example, the second syntax element is signaled prior to or subsequent to the first syntax element.
(193) In an example, the second syntax element is signaled only if a corresponding partition direction is valid for the video block.
(194) In an example, the indication is represented as a binarized codeword which comprises a first bin to indicate whether the first partition mode is applied to the video block.
(195) In an example, if the first bin indicates that the first partition mode is not applied, the binarized codeword further comprises a second bin to indicate whether BT or TT partition is applied to the video block.
(196) In an example, if the first bin indicates that first partition mode is applied, the binarized code further comprises one or more bins to indicate which partition pattern is applied to the video block.
(197) In an example, the indication is represented as a binarized codeword, wherein the binarized codeword uses a first bin to indicate one of the BT and TT partitions; the binarized codeword uses first and second bins to indicate the other one of the BT and TT partitions; and the binarized codeword uses first, second, and one or more subsequent bins to indicate which partition pattern of the first partition mode.
(198) In an example, the indication is represented as a binarized codeword, wherein the binarized codeword uses a first bin to indicate the first partition mode; the binarized codeword uses first and second bins to indicate the BT and TT partitions.
(199) In an example, the binarized codeword is a truncated unary code.
(200) In an example, the indication is signaled for one or more partition modes which are valid for the video block, and the one or more partition modes comprise at least one of the BT, TT and first partition modes.
(201) In an example, the one or more partition modes use only one partition direction which is previously signaled or derived.
(202) In an example, no indication is signaled for any partition mode which is invalid for the video block or for only one valid partition mode of the video block.
(203) In an example, if there are only two partition modes valid for the video block, the indication comprises a flag to indicate which partition mode is used for the video block.
(204) In an example, if there are more than two partition modes valid for the video block, the indication comprises a binarized codeword to indicate which partition mode is used for the video block.
(205) In an example, the binarized codeword is a truncated unary code, and the truncated unary code has a maximum value equal to N−1, wherein N represents an amount of the partition modes valid for the video block.
(206) In an example, if the first partition mode or at least one partition pattern of the first partition mode is invalid for the video block, a signaling of the at least one partition pattern is skipped or a flag for the signaling is constrained to be false.
(207) In an example, if the first partition mode is applied to the video block, the indication further indicates which partition pattern is applied to the video block.
(208) In an example, if the first partition mode has only one partition pattern valid for the video block, the partition pattern is implicitly used without being signaled.
(209) In an example, if the first partition mode has two partition patterns valid for the video block,

the indication comprises a flag to indicate which partition pattern is used for the video block.
(210) In an example, if the first partition mode has more than two partition patterns valid for the video block, the indication comprises a binarized codeword to indicate which partition pattern is used for the video block.
(211) In an example, the binarized codeword is coded by an arithmetic coding with at least one context or with a bypass mode.
(212) In an example, partial bins of the binarized codeword are coded with the at least one context, and remaining bins are coded with the bypass mode.
(213) In an example, all bins of the binarized codeword are coded with the at least one context; or all bins of the binarized codeword are coded with the bypass mode.
(214) In an example, the at least one context depends on at least one of: a position or index of a bin in the binarized codeword; characteristic of at least one of spatial neighbouring block, temporal neighbouring block, and the video block; statistical results of partition types from previously coded blocks; a type of at least one of slice, picture and title group covering the video block; and color component.
(215) In an example, the characteristic of at least one of spatial block, temporal neighbouring block, and the video block comprises at least one of: a partition mode; a partition depth; a coding mode; a width; and a height.
(216) In an example, M=5, 6, 7, or 8.
(217) In an example, the first partition mode comprises a unsymmetrical quad-tree (UQT) partition.
(218) In an example, the conversion includes encoding the current video block into the bitstream representation of a video and decoding the current video block from the bitstream representation of the video.
(219) In an example, there is disclosed an apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of examples described above.
(220) In an example, there is disclosed a computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of examples described above.
5. Example Implementations of the Disclosed Technology
(221) FIG. **12** is a block diagram of a video processing apparatus **1200**. The apparatus **1200** may be used to implement one or more of the methods described herein. The apparatus **1200** may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus **1200** may include one or more processors **1202**, one or more memories **1204** and video processing hardware **1206**. The processor(s) **1202** may be configured to implement one or more methods (including, but not limited to, method **1200**) described in the present document. The memory (memories) **1204** may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware **1206** may be used to implement, in hardware circuitry, some techniques described in the present document.
(222) In some embodiments, the video coding methods may be implemented using an apparatus that is implemented on a hardware platform as described with respect to FIG. **12**.
(223) From the foregoing, it will be appreciated that specific embodiments of the presently disclosed technology have been described herein for purposes of illustration, but that various modifications may be made without deviating from the scope of the invention. Accordingly, the presently disclosed technology is not limited except as by the appended claims.
(224) Implementations of the subject matter and the functional operations described in this patent document can be implemented in various systems, digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Implementations of the subject matter described in this specification can be implemented as one or more computer program

products, i.e., one or more modules of computer program instructions encoded on a tangible and non-transitory computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more of them. The term “data processing unit” or “data processing apparatus” encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.
(225) 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.
(226) The processes and logic flows described in this specification can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).
(227) Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of nonvolatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.
(228) It is intended that the specification, together with the drawings, be considered exemplary only, where exemplary means an example. As used herein, the singular forms “a”, “an” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. Additionally, the use of “or” is intended to include “and/or”, unless the context clearly indicates otherwise.
(229) 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.
(230) 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.
(231) 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

1. A method of processing video data, comprising: determining, for a conversion between a video comprising multiple regions comprising one or more video blocks including a current video block and a bitstream of the video, whether a first partition mode is applied to the current video block, wherein in the first partition mode, the current video block is directly split into five sub-blocks using Quinary-Tree (QUI-T), wherein a maximum or a minimum block size for a region that allows the one or more video blocks of the region to be split into the five sub-blocks using the QUI-T is determined according to whether dual tree partitioning is enabled for the region; and performing the conversion based on the determining, wherein the first partition mode is not applied to the current video block in response to a sub-block of the five sub-blocks crossing more than one virtual pipeline data unit (VPDU).
2. The method of claim 1, wherein the determining depends on at least one of a maximum block size, a minimum block size, and a maximum depth allowed for the first partition mode, wherein the maximum depth comprises at least one of a maximum bit depth and a maximum split depth, and the maximum block size is a size of a largest coding block/unit or a size of a VPDU.
3. The method of claim 2, wherein at least one of the maximum block size, the minimum block size, and the maximum depth is derived in a same way as that of QT partition mode.
4. The method of claim 2, wherein the first partition mode is not applicable to the current video block in a case that one of the following is satisfied: the maximum depth allowed for the first partition mode is reached for at least one sub-block; the minimum block size allowed for the first partition mode is reached for at least one sub-block; a block size for which a transform can be supported is reached for at least one sub-block.
5. The method of claim 1, wherein the determining depends on at least one of a depth of a multiple-type tree (MTT) or a depth of quadtree (QT).
6. The method of claim 1, wherein the determining depends on a split depth of the current video block, wherein the split depth of the current video block comprises at least one of QT split depth, binary tree (BT) split depth, ternary tree (TT) split depth, MTT split depth or split depth in the first partition mode, and wherein the first partition mode is applicable to the current video block in a case that the split depth of the current video block satisfies at least one of: D<=D1; D>=D2; wherein D, D1 and D2 represent the split depth of the current video block, a first depth threshold, and a second depth threshold respectively.
7. The method of claim 6, wherein at least one of thresholds is signaled in a sequence parameter set (SPS), a video parameter set (VPS), a picture parameter set (PPS), a picture header, a slice header, a tile group header or a tile header, or the at least one of thresholds is signaled in a video unit level wherein the video unit level comprises at least one of sequence, video, picture, slice, tile group, a coding tree unit (CTU) row or a CTU region, wherein the at least one of thresholds that is signaled

is shared by ternary tree (TT) partition and the first partition mode or shared by binary tree (BT) partition and the first partition mode.
8. The method of claim 1, wherein the first partition mode is not applicable to the current video block in at least one direction in a case that the current video block crosses a border of at least one of a picture, tile, and tile group comprising the current video block, wherein the border comprises at least one of a bottom border and a right border.
9. The method of claim 1, wherein at least one of the following partition modes is not available for at least one of sub-blocks: QT partition; BT partition in a horizontal direction; BT partition in a vertical direction; TT partition in a horizontal direction; TT partition in a vertical direction; unsymmetrical quadtree (UQT) partition in a horizontal direction; or UQT partition in a vertical direction.
10. The method of claim 1, wherein the determining depends on whether the current video block belongs to a leaf node to which any other partition modes is not applicable, and the first partition mode has one or more partition patterns, wherein the method further comprises: determining a first indication indicating whether the first partition mode is applicable to the leaf node; and determining a second indication indicating which partition pattern is used.
11. The method of claim 1, wherein in a case that the first partition mode or at least one partition pattern of the first partition mode is invalid for the current video block, a signaling of the at least one partition pattern is skipped or a flag for the signaling is constrained to be false.
12. The method of claim 1, wherein the conversion includes encoding the current video block into the bitstream.
13. The method of claim 1, wherein the conversion includes decoding the current video block from the bitstream.
14. An apparatus for processing video data comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to: determine, for a conversion between a video comprising multiple regions comprising one or more video blocks including a current video block and a bitstream of the video, whether a first partition mode is applied to the current video block, wherein in the first partition mode, the current video block is directly split into five sub-blocks using Quinary-Tree (QUI-T), wherein a maximum or a minimum block size for a region that allows the one or more video blocks of the region to be split into the five sub-blocks using the QUI-T is determined according to whether dual tree partitioning is enabled for the region; and perform the conversion based on the determination, wherein the first partition mode is not applied to the current video block in response to a sub-block of the five sub-blocks crossing more than one virtual pipeline data unit (VPDU).
15. The apparatus of claim 14, wherein the determination depends on whether the current video block belongs to a leaf node to which any other partition modes is not applicable, and the first partition mode has one or more partition patterns, wherein the instructions further cause the processor to: determine a first indication indicating whether the first partition mode is applicable to the leaf node; and determine a second indication indicating which partition pattern is used.
16. The apparatus of claim 14, wherein at least one of the following partition modes is not available for at least one of sub-blocks: QT partition; BT partition in a horizontal direction; BT partition in a vertical direction; TT partition in a horizontal direction; TT partition in a vertical direction; unsymmetrical quadtree (UQT) partition in a horizontal direction; or UQT partition in a vertical direction.
17. The apparatus of claim 14, wherein the determining depends on at least one of a maximum block size, a minimum block size, and a maximum depth allowed for the first partition mode, wherein the maximum depth comprises at least one of a maximum bit depth and a maximum split depth, and the maximum block size is a size of a largest coding block/unit or a size of a VPDU.
18. A non-transitory computer-readable recording medium storing a bitstream of a video which is generated by a method performed by a video processing apparatus, wherein the method comprises:

determining, for a conversion between the video comprising multiple regions comprising one or more video blocks including a current video block and the bitstream of the video, whether a first partition mode is applied to the current video block, wherein in the first partition mode, the current video block is directly split into five sub-blocks using Quinary-Tree (QUI-T), wherein a maximum or a minimum block size for a region that allows the one or more video blocks of the region to be split into the five sub-blocks using the QUI-T is determined according to whether dual tree partitioning is enabled for the region; and performing the conversion based on the determining, wherein the first partition mode is not applied to the current video block in response to a sub-block of the five sub-blocks crossing more than one virtual pipeline data unit (VPDU).
19. The non-transitory computer-readable recording medium of claim 18, wherein at least one of the following partition modes is not available for at least one of sub-blocks: QT partition; BT partition in a horizontal direction; BT partition in a vertical direction; TT partition in a horizontal direction; TT partition in a vertical direction; unsymmetrical quadtree (UQT) partition in a horizontal direction; or UQT partition in a vertical direction.
20. The non-transitory computer-readable recording medium of claim 18, wherein the determining depends on at least one of a maximum block size, a minimum block size, and a maximum depth allowed for the first partition mode, wherein the maximum depth comprises at least one of a maximum bit depth and a maximum split depth, and the maximum block size is a size of a largest coding block/unit or a size of a VPDU.