Block Partitioning Structure in the VVC Standard

Yu-Wen Huang, Jicheng An, Han Huang, Xiang Li, *Senior Member, IEEE*, Shih-Ta Hsiang, Kai Zhang, *Senior Member, IEEE*, Han Gao, *Graduate Student Member, IEEE*, Jackie Ma, and Olena Chubach

(Invited Paper)

***Abstract*— Versatile Video Coding (VVC) is the latest video coding standard jointly developed by ITU-T VCEG and ISO/IEC MPEG. In this paper, technical details and experimental results for the VVC block partitioning structure are provided. Among all the new technical aspects of VVC, the block partitioning structure is identified as one of the most substantial changes relative to the previous video coding standards and provides the most significant coding gains. The new partitioning structure is designed using a more flexible scheme. Each coding tree unit (CTU) is either treated as one coding unit or split into multiple coding units by one or more recursive quaternary tree partitions followed by one or more recursive multi-type tree splits. The latter can be horizontal binary tree split, vertical binary tree split, horizontal ternary tree split, or vertical ternary tree split. A CTU dual tree for intra-coded slices is described on top of the new block partitioning structure, allowing separate coding trees for luma and chroma. Also, a new way of handling picture boundaries is presented. Additionally, to reduce hardware decoder complexity, virtual pipeline data unit constraints are introduced, which forbid certain multi-type tree splits. Finally, a local dual tree is described, which reduces the number of small chroma intra blocks.**

***Index Terms*— Block partitioning structure, binary tree, CTU dual tree, H.266, local dual tree, MPEG-I Part 3, multi-type tree, picture boundary handling, quaternary tree, ternary tree, versatile video coding, virtual pipeline data unit, VVC.**

I. Introduction

TECHNOLOGY advances of video coding become more critical as video data occupy higher traffic on the Internet and video applications demand increasing spatial and temporal resolutions. In early 2013, the first edition of High Efficiency Video Coding (HEVC) | ITU-T Rec. H.265 | ISO/IEC 23008-2 (MPEG-H Part 2) [1], [2] was finalized, and the HEVC Test Model (HM) software [3] was developed by the Joint Collaborative Team on Video Coding (JCT-VC) of Video Coding Experts Group (VCEG, ITU-T SG16 Q.6) and Moving Picture Experts Group (MPEG, ISO/IEC JTC1/SC29). In late 2015, VCEG and MPEG formed a Joint Video Exploration Team to explore future video coding technologies. A Joint Exploration Model (JEM) [4]–[6] was developed and showed substantial coding gains over the HM. In October 2017, VCEG and MPEG issued a joint Call for Proposals (CfP) on video compression with capabilities beyond HEVC [7]. In April 2018, VCEG and MPEG received 22, 12, and 12 responses in the Standard Dynamic Range (SDR), High Dynamic Range (HDR), and 360° categories, respectively [8]. Many proposals contributed as responses to the CfP, e.g., [9]–[14], showed significant coding gains over the HM and even over the JEM. The Joint Video Exploration Team was renamed as the Joint Video Experts Team (JVET), which then officially started the standardization process of Versatile Video Coding (VVC) | ITU-T Rec. H.266 | ISO/IEC 23090-3 (MPEG-I Part 3). After a bit more than two years of standardization work, in July 2020, the first edition of VVC [15] was finalized.

For more than two decades, macroblocks have been utilized as a basic compression unit for video coding. In Advanced Video Coding (AVC) | ITU-T Rec. H.264 | ISO/IEC 14496-10 (MPEG-4 Part 10) [16] and all video coding standards before AVC, each picture is partitioned into non-overlapping macroblocks. Switching between inter and intra prediction is performed at the macroblock level. One or multiple prediction blocks and one or multiple transform blocks are specified inside each macroblock.

As popular video picture resolution reached full high-definition (i.e., FHD or 1080p) and kept increasing, the macroblock-based design became less efficient. Adding further improvements on top of it did not provide any good tradeoff between coding gain and increased complexity. Consequently, after AVC, instead of building new coding tools on top of the old macroblock-based design, a new recursive block partitioning structure [17] was adopted in HEVC. This recursive block partitioning structure was regarded as one of the most important breakthroughs in HEVC. It provided the most significant coding gains over the AVC reference encoder compared to other coding tools.

Meanwhile, popular video picture resolutions continued to increase further and reached ultra-high definition (UHD), i.e., 4K UHD and 8K UHD. As in the case of AVC macroblocks,

Manuscript received August 18, 2020; revised February 8, 2021 and April 16, 2021; accepted May 28, 2021. Date of publication June 11, 2021; date of current version October 4, 2021. This article was recommended by Associate Editor G. J. Sullivan. *(Corresponding author: Yu-Wen Huang.)*

Yu-Wen Huang, Shih-Ta Hsiang, and Olena Chubach are with MediaTek Inc., Hsinchu 30078, Taiwan (e-mail: yuwen.huang@mediatek.com; shih-ta.hsiang@mediatek.com; olena.chubach@mediatek.com).

Jicheng An is with Alibaba Group, Beijing 100102, China (e-mail: jicheng.ajc@alibaba-inc.com).

Han Huang is with Qualcomm Technologies Inc., San Diego, CA 92121 USA (e-mail: hanhuang@qti.qualcomm.com).

Xiang Li is with Tencent, Palo Alto, CA 94306 USA (e-mail: xlxiangli@tencent.com).

Kai Zhang is with Bytedance Inc., San Diego, CA 92122 USA (e-mail: zhangkai.video@bytedance.com).

Han Gao is with Huawei Technologies, 80992 Munich, Germany (e-mail: han.gao@tum.de).

Jackie Ma is with the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, 10587 Berlin, Germany (e-mail: jackie.ma@hhi.fraunhofer.de).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2021.3088134.

Digital Object Identifier 10.1109/TCSVT.2021.3088134

adding new coding tools on top of HEVC block partitioning became a bottleneck in the further enhancement of video compression capabilities during the JEM development. Thus, more advanced flexible block partitioning structures [18]–[21] were researched and developed, which later became a fundamental part of VVC.

The rest of this paper is organized as follows. In Section II, the HEVC block partitioning structure is reviewed. In Section III, the VVC block partitioning structure is described. In Section IV, performance impacts from several aspects of the VVC block partitioning structure are illustrated through experimental results. Finally, Section V concludes this paper.

II. HEVC Block Partitioning Structure

A brief review of the HEVC block partitioning structure is presented in this section. More related details can be found in [17]. In HEVC, the basic compression unit is called the coding tree unit (CTU). For non-monochromatic color formats, one picture is divided into non-overlapping CTUs, and each CTU contains one luma coding tree block (CTB) and two corresponding chroma CTBs. In the following, the luma and two chroma components are respectively noted as Y, Cb, and Cr. Unlike the macroblock size, which is fixed to be equal to 16 × 16, the CTU size is flexible and can be specified in the sequence parameter set (SPS) as 64 × 64, 32 × 32, or 16 × 16 in units of luma samples. The 64 × 64 CTU size is suggested to be used for larger video resolutions, allowing to achieve the best coding efficiency of HEVC. This CTU size is commonly used in many practical HEVC applications. Smaller CTU sizes, i.e., 32 × 32 or 16 × 16, can be used to reduce encoding complexity at the cost of coding efficiency compared to the 64 × 64 CTU size. The concept of CTUs can be regarded as a concept of extended macroblocks when the CTU size is not 16 × 16.

The characteristics of video picture content may be different and vary significantly, even within one CTU, so flexible partitioning adaptation within each CTU is essential for efficient compression. In HEVC, the CTU is the basic compression unit and the root of a coding tree. A recursive quaternary tree (QT) split may be applied to each CTU, where pioneer researches on QT can be found in [22] and [23]. Accordingly, every CTU is either treated as one coding unit (CU) or split into multiple CUs. An encoder generally signals a CU split indication flag at each QT node to specify whether the current QT node is partitioned or not. If the encoder decides to split, the QT node is further divided into four square child QT nodes of equal size. When the QT node reaches the allowed minimum CU size signaled in the SPS, the CU split indication flag is not signaled and is inferred to be 0. Each CU contains one luma coding block (CB) and two corresponding chroma CBs. The CU split indication flag is shared between luma and chroma at each QT node. All QT leaf nodes are treated as CUs. The CU size can be 64 × 64, 32 × 32, 16 × 16, or 8 × 8. The processing order of CUs within a CTU follows the depth-first order of a QT. In HEVC, the prediction scheme is specified at the CU level. The following modes can be applied to a CU: skip mode inter prediction, non-skip mode inter prediction, and intra prediction.

In HEVC, each CU can be either treated as one prediction unit (PU) or further divided into multiple PUs, and the PU is used as the basic unit for prediction within a CU. Only one type of prediction process can be applied inside each PU, and relevant prediction information is signaled at the PU level. In the rest of this paragraph, the CU size is denoted as 2N × 2N. For CUs of skip mode inter prediction, the PU size is fixed as 2N × 2N, and the merge scheme is applied at the PU level. In this case, a merge index is conditionally signaled. This index indicates which one of the spatially or temporally neighboring blocks is used to copy all motion information, or identifies whether a combined bi-predictive merge candidate or a zero motion vector merge candidate is applied. Motion information is represented by a prediction indicator (list 0 prediction, list 1 prediction, or bi-prediction), reference indexes, and motion vectors. For a CU of non-skip mode inter prediction, the following eight PU split modes are possible: one 2N × 2N PU, two 2N × N PUs, two N × 2N PUs, four N × N PUs, one 2N × 0.5N PU and one 2N × 1.5N PU, one 2N × 1.5N PU and one 2N × 0.5N PU, one 0.5N × 2N PU and one 1.5N × 2N PU, or one 1.5N × 2N PU and one 0.5N × 2N PU. The last four modes here are used for asymmetric motion partitioning (AMP). In the non-skip mode inter prediction, either merge or non-merge scheme can be applied for each PU. In this case, the prediction indicator, reference indexes, motion vector prediction (MVP) indexes, and motion vector differences (MVDs) may be signaled in the case of a non-merge scheme. The N × N PU mode is only permitted when the CU size equals the allowed minimum CU size. This limitation allows avoiding redundancy between using four N × N PUs for a 2N × 2N CU and splitting the 2N × 2N CU into four child N × N CUs without any further partitioning. The inter 4 × 4 PU is forbidden in HEVC to reduce the worst-case motion compensation (MC) memory bandwidth. Thus, to support the N × N PU mode, the allowed minimum CU size must be larger than 8 × 8. In the case of intra prediction, there are two possible PU split modes for CUs: one CU can contain either one 2N × 2N PU or four N × N PUs. Similarly, to avoid redundancy, the N × N PU mode is only permitted when the CU size equals the allowed minimum CU size. In this case, because the intra 4 × 4 PU is supported in HEVC, there is no requirement for the allowed minimum CU size to be larger than 8 × 8. In general, one PU contains one luma prediction block (PB) and two corresponding chroma PBs. For inter prediction CUs, the numbers of the Cb PBs and the Cr PBs are the same as those of the Y PBs. The chroma PB size and motion information of the chroma PBs are derived from the corresponding luma PB according to the chroma sampling format. At the same time, for intra prediction CUs, regardless of the selected PU mode for luma (one 2N × 2N luma PB or four N × N luma PBs), the chroma PB size is kept the same as the chroma CB size without any further partitioning. Moreover, while the Cb PB and the Cr PB do not always have to follow the luma intra prediction mode, they always share one signaled chroma intra prediction mode.

In HEVC, each CU can be either treated as one transform unit (TU) or further divided into multiple TUs. The TU is

used as the basic unit for performing transform, quantization, and residual coding. Similar to splitting one CTU into one or multiple CUs, a recursive QT split may be applied at the CU level to form TUs. To differentiate from the QT split of a CTU, the latter QT split is often called a residual quaternary tree (RQT). At each RQT node, an encoder can select either non-split or split, and a corresponding TU split indication flag is signaled to the decoder. If the encoder decides to split, the RQT node is further divided into four square child RQT nodes of equal size. The maximum RQT depth is signaled in the SPS and is set to 2 in the common test conditions (CTCs) used for HEVC standardization. When the RQT node reaches the maximum RQT depth, the TU split indication flag is not signaled and is inferred to be 0. Thus, for one 2N × 2N CU, RQT may result in either one TU with a size equal to 2N × 2N (RQT depth 0) or multiple TUs with dimensions equal to N × N (RQT depth 1) or 0.5N × 0.5N (RQT depth 2). When the RQT node indicates that the TU size is larger than the allowed maximum TU size in HEVC (i.e., 32 × 32), the TU split indication flag is not signaled and is inferred to be 1. In general, one TU contains one luma transform block (TB) and two chroma TBs. At each RQT node, the TU split indication flag is shared between luma and chroma. In HEVC, one exception appears when the minimum TB size is set to 4 × 4 for both luma and chroma. In this case, when one 8 × 8 RQT node is split, four 4 × 4 luma TBs are formed, while only one 4 × 4 Cb TB and one 4 × 4 Cr TB are created.

For inter prediction CUs, PUs and TUs are completely decoupled and can be independently chosen by an encoder. However, this is not the case for intra prediction CUs, where no PB boundaries can appear inside any TB. Therefore, for intra prediction CUs, the root TU split indication flag may be skipped and inferred to be 1. For example, when the 4 × 4 luma intra prediction mode is selected for one 8 × 8 CU, it results in four 4 × 4 luma intra PBs. In this case, the TU split indication flag at the 8 × 8 block level is not signaled and is inferred to be 1. Besides the normative relationship between PUs and TUs mentioned earlier, in the case of an intra coded CU, information is signaled on a PU basis. At the same time, prediction is performed on a TU basis. For example, for one 8 × 8 CU, an encoder can select 8 × 8 luma intra prediction (resulting in one 8 × 8 luma intra PB) and set the TU split indication flag to 1 (resulting in four 4 × 4 luma TBs). In this case, only one luma intra mode is signaled for this 8 × 8 CU, while corresponding luma 4 × 4 TBs are intra predicted and transformed sequentially, using the same luma intra mode.

III. VVC Block Partitioning Structure

In this section, the VVC block partitioning structure is described in detail. Methods to improve the coding efficiency of the block partitioning structure are introduced in Subsections III-A, III-C, and III-D. Methods to reduce the hardware decoder implementation cost and increase the hardware decoder processing throughput are described in Subsections III-B and III-E, respectively. Split signaling, configurations, and default settings of the VVC Test Model (VTM) are presented in Subsection III-F. The encoding algorithm and speedup methods in VTM are presented in Subsection III-G.

A. Quaternary Tree Plus Multi-Type Tree

In VVC, as in HEVC, one picture is partitioned into multiple non-overlapping CTUs. A CTU size in VVC can be set up to 128 × 128 in units of luma samples, while in HEVC, as mentioned previously, it can be set up to 64 × 64. Increasing the maximum allowed CTU size can often improve coding efficiency, and it is especially effective when encoding UHD sequences. The result of setting a CTU size to 64 × 64 instead of 128 × 128 will be shown in Section IV. As shown later, more significant losses appear for higher spatial resolution videos, which experimentally demonstrates that increasing the maximum allowed CTU size from 64 × 64 in HEVC to 128 × 128 in VVC helps to achieve better performance for higher spatial resolution videos. However, just enlarging a CTU size to 128 × 128 causes a significant cost increase, and therefore does not provide a good tradeoff between coding gain and additional complexity. As described in the previous section, in HEVC, a recursive QT split is applied to each CTU, resulting in one or multiple CUs, all having square shapes. In VVC, rectangular CUs are supported together with square CUs, which allows a better fit to local picture content characteristics. This better fit is achieved by adopting a binary tree (BT) split, initially introduced in the quaternary tree plus binary tree (QTBT) design [18]–[20] in the JEM, and a ternary tree (TT) split, initially introduced in [21], in addition to the QT and BT splits.

The QTBT design was initially adopted into JEM3.0 and then used as the JEM block partitioning structure. According to the QTBT design, a recursive QT split is applied first to each CTU, followed by a recursive BT split applied to each QT leaf node. A general idea of the QTBT is to use BT in addition to QT so that more CU split options are provided, allowing to capture the local picture content characteristics better. The idea behind applying BT only to QT leaf nodes is to allow a quick adaptation to small local changes while still using less additional side information. Otherwise, if both QT and BT are permitted for every coding tree node, the amount of additional side information will vastly increase, leading to a significant overhead of bits needed for signaling adaptation. At the same time, it was experimentally shown that if QT and BT are allowed to interleave, i.e., further allowing recursive QT at BT leaf, the additional coding efficiency improvement is minimal and not worth the complexity increase.

The TT was introduced later on top of the QTBT. The general idea behind the TT is as follows: when a small object is located in the middle region of a coding tree node, neither QT nor BT can provide a good partitioning adaptation, while TT can. Note that when either BT or TT is applied, both the width and height of all resulting blocks are kept equal to the power of 2. Therefore, no new transform size is introduced by BT or TT.

In VVC, each CTU is either treated as one CU or split into multiple CUs by a recursive QT, followed by a recursive binary-ternary tree (BTT). The BTT, also called a multi-type tree (MTT), is described below. Each CTU is first partitioned

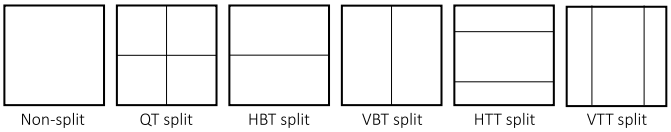

Fig. 1. Illustration of splitting types in QT + MTT.

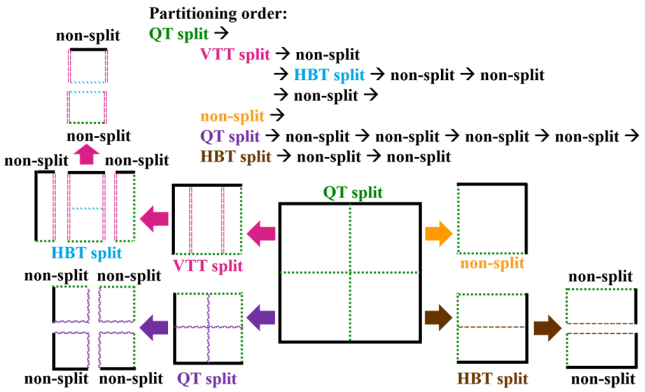

Fig. 2. Example of CU partitioning in one CTU of VVC. Each CU is partitioned recursively, left to right, top to bottom.

by a recursive QT split. Every QT node can either be non-split (becoming a QT leaf) or be QT split into four child QT nodes. Each child QT node has a square shape, and its size is equal to a quarter of its parent's node size. At every QT leaf, a recursive MTT split is further applied. It is important to note that once the MTT is applied to the QT leaf, only the BT and TT are further allowed, while the QT is disallowed for all subsequent nodes. Each MTT node can be either non-split (as MTT leaf) or one of the following: horizontal binary tree (HBT) split into two child MTT nodes, vertical binary tree (VBT) split into two child MTT nodes, horizontal ternary tree (HTT) split into three child MTT nodes, vertical ternary tree (VTT) split into three child MTT nodes. In the BT case, two MTT child nodes have the same size, each equal to half of the parent MTT node. In the TT case, the splitting ratio is 1:2:1, and three MTT child nodes are of a quarter, half, and quarter-size of the parent MTT node, respectively. Fig. 1 illustrates all splitting types of QT + MTT, and Fig. 2 shows an example of the CU partitioning in one CTU, where QT + MTT split decisions in the depth-first order are provided with the drawing. In this example, a QT split is first applied to the CTU, resulting in four QT nodes, where each QT node is of the quarter CTU size. The following splits are further used for the four QT nodes: a VTT split and subsequent splits of the VTT split are applied to the top-left QT leaf, a non-split is applied to the top-right QT leaf, a QT split and subsequent splits of the QT split are applied to the bottom-left QT node, and an HBT split and subsequent splits of the HBT split are applied to the bottom-right QT leaf.

As mentioned earlier, in HEVC, the PU and TU are further specified for every CU, and each CU can contain one or more PUs and one or more TUs. In VVC, CU, PU, and TU concepts are unified. In general, the PU and TU are of the same size as the corresponding CU. This unification simplifies the

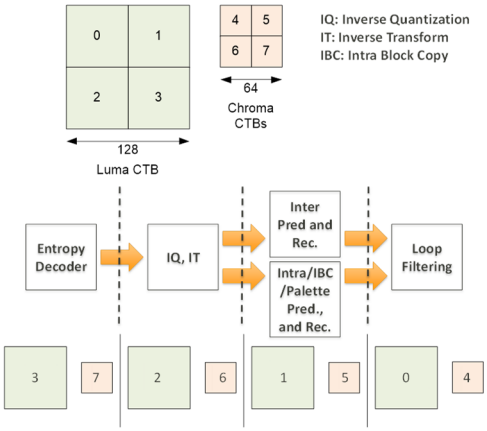

Fig. 3. Example of a pipelined hardware video decoder with the VPDU size equal to 64 × 64-L/32 × 32-C.

VVC block partitioning structure and significantly reduces signaling overhead for PUs and TUs without sacrificing coding gain. Some exceptions are yet defined in VVC. For example, when a subblock-based temporal motion vector prediction (SbTMVP) or a decoder-side motion vector refinement (DMVR) is applied to a CU, this CU is conditionally split into multiple subblocks. Such a split depends on the CU size and does not require signaling any side information to define the subblock size. When affine MC is applied to a CU, this CU is split into multiple subblocks without signaling any side information to determine the subblock size. In another example, an intra subpartitions (ISP) mode is applied to an intra prediction CU. This CU is then partitioned into either two or four transform block subpartitions. No additional side information for defining the subpartition size is signaled in this case. When a subblock transform (SBT) is applied to an inter prediction CU, this CU is split into two TUs, where one TU has residual data, and the other TU does not. In this case, the transform type is decided implicitly; however, signaling additional syntax elements is still required to define the size of obtained TUs. In yet another example, when a CB is larger than the maximum allowed transform size signaled in the SPS, this CB is implicitly split into multiple TBs without signaling any side information for defining the TB size.

B. Virtual Pipeline Data Unit Constraints

In this Subsection, the concept of virtual pipeline data units (VPDUs) [24], [25] is introduced, which is very important for hardware video decoder architectures. VPDUs are non-overlapping M × M-luma(L)/N × N-chroma(C) units of a picture. In hardware video decoders, successive VPDUs are processed by multiple pipeline stages simultaneously, and different stages process different VPDUs simultaneously. Fig. 3 shows an example of a 4-stage pipelined hardware video decoder with M = 64 and N = 32, where the corresponding four pipeline stages simultaneously process four VPDUs. In most pipeline stages, the VPDU size is roughly proportional to the buffer size requirement, so it is crucial to keep it small.

In most HEVC hardware decoders, the VPDU size is set to the maximum allowed TU size. Increasing the maximum

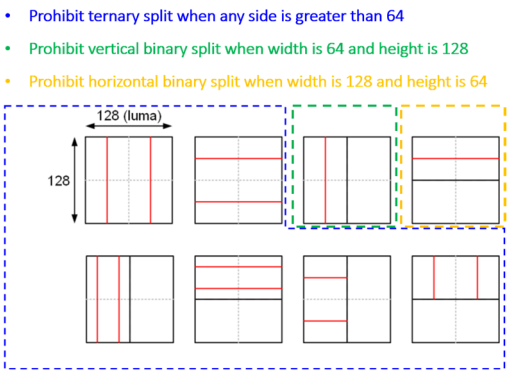

Fig. 4. Examples of BT or TT splits that are inefficient for 64 × 64-L/32 × 32-C pipelining and the three VPDU constraints to disallow these examples.

allowed TU size from 32 × 32-L/16 × 16-C, as in HEVC, to 64 × 64-L/32 × 32-C in VVC, may increase coding gains, yet it will result in four times the size of the VPDU (i.e., 64 × 64-L/32 × 32-C) when compared to HEVC. Meanwhile, on top of the QT split, the BT and TT splits are adopted in VVC. Applying BT or TT recursively to 128 × 128-L/64 × 64-C CTUs will result in 16 times the size of the VPDU (i.e., 128 × 128-L/64 × 64-C) when compared to HEVC. Examples of the BT or TT splits that are harmful to 64 × 64-L/32 × 32-C pipelining are shown in Fig. 4. To reduce the VPDU size back to 64 × 64-L/32 × 32-C for VVC, when processing CUs in the CTU, it is not allowed to revisit a 64 × 64-L/32 × 32-C VPDU after leaving it. Also, the following two conditions must be satisfied for each CTU.

[Condition 1] For each 64 × 64-L/32 × 32-C VPDU containing one or multiple CUs, all CUs must be contained entirely in a 64 × 64-L/32 × 32-C VPDU.

[Condition 2] For each CU containing one or multiple 64 × 64-L/32 × 32-C VPDUs, all 64 × 64-L/32 × 32-C VPDUs must be contained entirely in the CU.

To comply with the above principles, the following three VPDU constraints are included in the VVC specification text as normative syntax constraints. 1) Prohibit applying TT split to any coding tree node with a width or height greater than 64 luma samples. 2) Prohibit applying VBT split to any coding tree node of 64 × 128 luma samples. 3) Prohibit applying HBT split to any coding tree node of 128 × 64 luma samples. With these VPDU constraints, significant hardware decoder cost savings can be achieved. At the same time, it was reported in [25] that coding efficiency loss caused by this design is only 0.00%, 0.15%, 0.06% in terms of luma Bjøntegaard Delta rate (BD-rate) [26], [27] for the all intra (AI), random access (RA), and low-delay B (LB) VTM CTCs, respectively.

C. CTU Dual Tree

In HEVC, the coding tree of a CTU is shared by Y, Cb, and Cr components, so one CU consists of one luma CB and two chroma CBs. In VVC, this single tree structure is retained for P and B slices. However, in I slices, the luma and chroma components' spatial characteristics can differ. This assertion is utilized in VVC, as described below. In Fig. 5, an exemplary

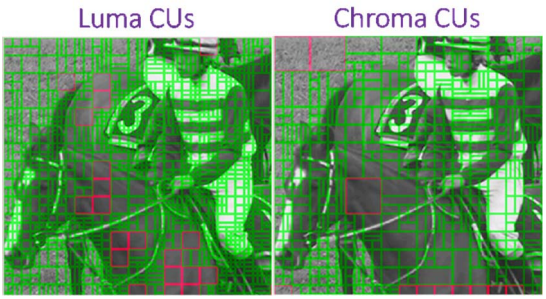

Fig. 5. Illustration of CU partitions when encoding luma and chroma separately. QT split CUs are marked with red, MTT split CUs are marked with green.

CU partitioning of a coded picture is shown, where luma and chroma components are encoded separately. It can be seen that luma mostly has a finer texture than chroma, which results in a higher amount of smaller CUs in luma than those in chroma. Thus, it is reasonable to use separate coding trees for the luma and chroma components in I slices. In this case, a luma CTU (containing only one luma CTB of the original CTU) forms one coding tree, and a chroma CTU (containing only two chroma CTBs of the original CTU) forms a chroma separate tree (CST). In VVC, this CST design in I slices is also called CTU dual tree.

Starting CST partitioning from the CTU level requires signaling luma CTB first, followed by signaling chroma CTBs. Therefore, a decoder must process and store a 128 × 128-L block before processing the corresponding 64 × 64 chroma CTBs. Such processing order results in four times the buffer size compared to that without the CTU dual tree. To reduce the buffer requirement in VVC, as suggested in [28], a CST starts at the maximum TU level instead of the CTU level.

In the VTM CTCs, every CTU of 128 × 128-L/64 × 64-C in I slices is first implicitly QT split into four 64 × 64-L/32 × 32-C coding tree nodes. Two separate coding trees (one is the luma coding tree, and the other is the CST) start at each of the four 64 × 64-L/32 × 32-C coding tree nodes. Fig. 6 demonstrates an example of starting a dual tree at the 64 × 64-L/32 × 32-C level. Here, a QT split is inferred at the CTU level, which allows the successful processing of multiple 64 × 64-L/32 × 32-C VPDUs in parallel by a pipelined VVC hardware decoder.

When the cross-component linear model (CCLM) mode is selected as a chroma intra prediction mode for the current chroma CBs, the following samples are involved in the process of generating chroma prediction samples: 1) the reconstructed samples from the chroma neighboring blocks of the current chroma CBs; 2) the reconstructed samples from the corresponding luma neighboring blocks and the luma collocated block of the current chroma CBs. In a single tree with CCLM chroma CBs, partitions of luma and chroma components are aligned. The reconstruction of any chroma sample in the current Cb/Cr CB can be performed right after reconstructing the required luma samples. Thus, as it is shown in Example 1 of Fig. 7, if the same CU partitions are applied to the 64 × 64 luma coding tree node and the

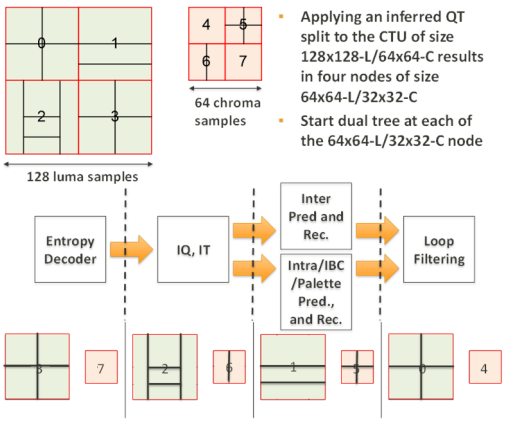

Fig. 6. Example of starting dual tree at 64 × 64-L/32 × 32-C level for a pipelined VVC hardware decoder with the VPDU size equal to 64 × 64-L/32 × 32-C.

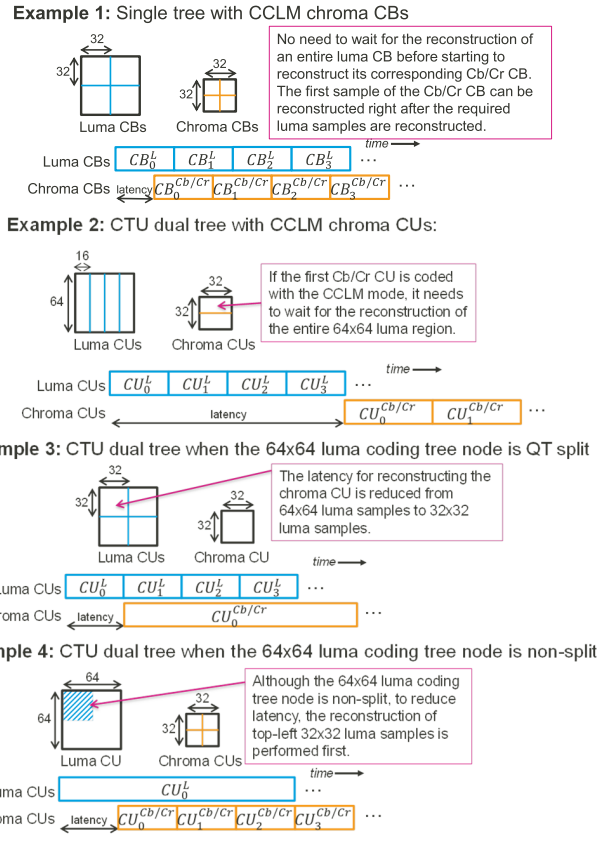

Fig. 7. Examples of processing latency for hardware decoding caused by interaction between CTU dual tree and CCLM.

corresponding 32 × 32 chroma coding tree node, there is no need to wait for the reconstruction of an entire luma CB before starting to reconstruct its corresponding Cb/Cr CB. However, when a CTU dual tree is applied, the CU partitions between the luma and chroma components might differ, resulting in different processing orders of the luma and chroma CUs. Once the CCLM mode is also applied, a long processing latency issue occurs in hardware decoding. Example 2 in Fig. 7 demonstrates the worst case when a decoder needs to reconstruct all four 16 × 64 luma CUs for the reconstruction of

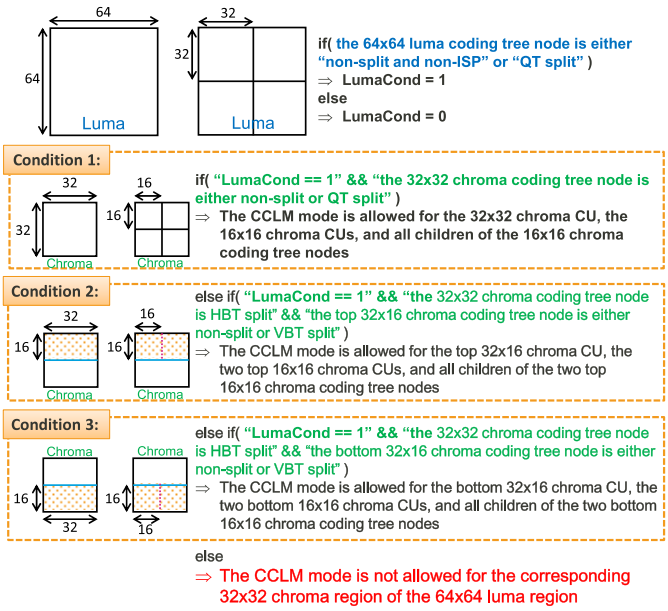

Fig. 8. Checks to reduce processing latency for hardware decoding caused by interaction between CTU dual tree and CCLM. If Conditions 1, 2, and 3 are all false, the CCLM is not allowed.

the first 32 × 16 chroma CU. This case requires a processing latency of 4096 (4 × 16 × 64) luma samples.

Several proposals allowing to reduce the processing latency were submitted and studied during the VVC standardization, e.g., [29]–[32]. The solution adopted in VVC restricts the usage of the CCLM mode by checking the luma CU partitioning and chroma CU partitioning. The high-level concept of the solution is to allow the CCLM mode only for those partitions, which result in a processing latency between luma and chroma samples of no longer than 1024 luma samples when the CCLM mode is used. As shown in Fig. 8, in the CTU dual tree, a variable, LumaCond, is calculated first by checking the luma CU partitioning. The computation is as follows. If a 64 × 64 luma coding tree node is either "non-split and non-ISP" or "QT split," then LumaCond is equal to 1; otherwise, it is equal to 0. Then Conditions 1, 2, and 3 are checked to determine whether the CCLM mode is allowed for particular chroma CUs within the corresponding 32 × 32 chroma region.

[Condition 1] If LumaCond is equal to 1 and the 32 × 32 chroma coding tree node is either non-split or QT split, the CCLM mode can be applied to 32 × 32 chroma CU, the 16 × 16 chroma CUs, and all children of the 16 × 16 chroma coding tree nodes.

[Condition 2] If LumaCond is equal to 1, and the 32 × 32 chroma coding tree node is HBT split, and the top 32 × 16 chroma coding tree node is either non-split or VBT split, the CCLM mode is allowed for the top 32 × 16 chroma CU, the two top 16 × 16 chroma CUs, and all children of the two top 16 × 16 chroma coding tree nodes.

[Condition 3] If LumaCond is equal to 1, and the 32 × 32 chroma coding tree node is HBT split, and the bottom 32 × 16 chroma coding tree node is either non-split or VBT split, the CCLM mode is allowed for the bottom 32 × 16 chroma CU, the two bottom 16 × 16 chroma CUs, and all children of the two bottom 16 × 16 chroma coding tree nodes.

The CCLM mode is not allowed for the corresponding 32 × 32 region of the 64 × 64 luma region in all other cases.

When any of Condition 1, 2, or 3 is satisfied, the additional processing latency needed for generating the CCLM chroma prediction samples is no longer than 32 × 32 luma samples. For example, as shown in Example 3 of Fig. 7, the 64 × 64 luma coding tree node is split into four 32 × 32 luma CUs, and the corresponding 32 × 32 chroma coding tree node is non-split. In this case, the decoder only needs to wait for the first 32 × 32 luma CU to be reconstructed before reconstructing the first chroma sample. In Example 4 of Fig. 7, the 64 × 64 luma coding tree node is non-split, and the corresponding 32 × 32 chroma coding tree node is split into four 16 × 16 chroma CUs. For generating the prediction samples of the first 16 × 16 chroma CU, the reconstruction of the collocated 32 × 32 luma samples can be performed first, and the required latency is kept as 32 × 32 luma samples. It is reported that the coding efficiency loss caused by the constrained CCLM is only 0.02%, 0.39%, and 0.35% in terms of Y, Cb, and Cr BD-rates under the RA VTM CTCs, respectively, which means that roughly 98% of the original CCLM coding gain is preserved.

Another case of processing latency for chroma components appears when the luma mapping with chroma scaling (LMCS) [33], [34] and the CTU dual tree are both applied. In this case, the decoded chroma residual is scaled by a particular factor. During the LMCS development, the chroma residual scaling factor was derived initially based on the average value of the corresponding luma PB (the result of motion compensation in inter mode or the intra predictor in intra mode). Such definition, however, resulted in a similar latency issue as in the case of CCLM. To reduce the latency in LMCS, one single chroma residual scaling factor is derived for the entire VPDU instead of defining an individual scaling factor for each PB. Also, this factor is not based on the values of the current VPDU. Instead, it is derived based on the average value of the neighboring luma reconstruction samples of the current VPDU. As shown in Fig. 9, for each VPDU, the average value of the M left neighboring luma reconstruction samples and the M top neighboring luma reconstruction samples is calculated and denoted as *avgY*, where the M = min(the luma CTB width, 64). The chroma residual scaling factor is derived using a table lookup based on the *avgY*, even before the current VPDU is decoded. This way, there is no dependency between the luma and chroma components when CTU dual tree is applied, and therefore the processing latency does not appear.

D. Picture Boundary Handling

In this paper, a CTU or coding tree node is called a partial CTU or partial coding tree node when this CTU or coding tree node is located either at the right or the bottom picture boundary and contains samples outside of the current picture. In HEVC, each partial coding tree node is forced to be split using QT, and CUs located entirely outside of the current picture are not coded. Such a scheme may result in rows and/or columns of small square CUs along picture boundaries, leading to poor coding efficiency for partial CTUs. An example of

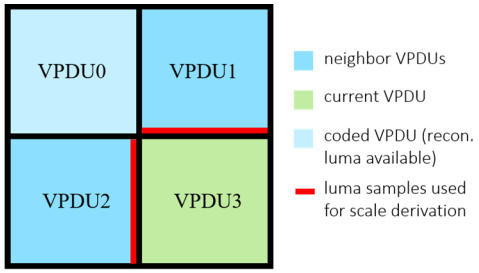

Fig. 9. VPDU neighboring samples used in chroma scaling factor derivation.

Fig. 10. CTU split at picture boundaries; (a) HEVC-like forced QT split at the bottom picture boundary; (b) forced split using QT or BT split at the bottom picture boundary. QT split CUs are marked with red, MTT split CUs are marked with green.

a cropped coded picture is shown in Fig. 10 (a) and (b). In the first case, a picture of 1080p resolution is partitioned using HEVC-like forced QT split, which leads to many small blocks in partial coding tree nodes at the bottom picture boundary. During the VVC standardization process, several contributions, e.g., [35]–[37], etc., advocated that a partial coding tree node can be split into child CUs using either QT or BT split, as shown in Fig. 10 (b). The CU partition signaling procedure for partial CTUs in VVC is presented below.

First of all, for each partial CTU, split_cu_flag (indicating whether this coding tree node is a coding tree leaf) is not signaled and is inferred to be 1. Also, assuming that any one of the following conditions is satisfied, a QT split is enforced to the current coding tree node (i.e., split_qt_flag, indicating whether the QT split is applied, is not signaled and inferred to be 1): 1) the current partial coding tree node crosses both the right and the bottom picture boundary, and the size of the current coding tree node is greater than the minimum allowed QT size (the minimum QT leaf CB size for luma, MinQtSizeY, in cases of the single tree and the

dual tree for luma, or the minimum QT leaf CB size for chroma, MinQtSizeC, in case of the dual tree for chroma); 2) the size of the current partial coding tree node is greater than the maximum allowed BT size (the maximum CB size for luma to apply the BT split, MaxBtSizeY, in cases of the single tree and the dual tree for luma, or the maximum CB size for chroma to apply the BT split, MaxBtSizeC, in case of the dual tree for chroma). Otherwise, if the MTT is enabled and any one of the following conditions is true, then a BT split is enforced to the current partial coding tree node (i.e., split_qt_flag is not signaled and inferred to be 0, and the flag which specifies whether a CU is split using a binary or ternary split, mtt_split_cu_binary_flag, is not signaled and inferred to be 1): 1) the hierarchy depth of the MTT splitting of the current CB, i.e., the MTT depth (the maximum hierarchical MTT depth for luma, MaxMttDepthY, in cases of the single tree or the dual tree for luma, or the maximum hierarchical MTT depth for chroma, MaxMttDepthC, in case of the dual tree for chroma) of the current partial coding tree node is greater than 0; 2) the size of the current partial coding tree node is less than or equal to the minimum allowed QT size (MinQtSizeY in cases of the single tree and the dual tree for luma, or MinQtSizeC in case of the dual tree for chroma). Otherwise, if the MTT is enabled and none of the previous conditions apply, either a QT or BT split can be selected by an encoder for the current coding tree node (i.e., split_qt_flag is signaled). If the QT split is not chosen by the encoder (split_qt_flag is 0), then mtt_split_cu_binary_flag is not signaled and inferred as 1.

When a BT split is applied to the current partial coding tree node, mtt_split_cu_vertical_flag (identifies whether a coding unit is split horizontally or vertically) is not signaled and inferred, as described below. If the BT split is applied to the current coding tree node, which crosses the bottom picture boundary, an HBT split is enforced to this coding tree node (i.e., mtt_split_cu_vertical_flag is inferred to be 0). Otherwise, a VBT split is enforced to this coding tree node (i.e., mtt_split_cu_vertical_flag is inferred to be 1). The above processes are recursively applied until the coding tree node represents a CU located entirely within a picture. During the splitting process of a partial CTU, the maximum allowed MTT depth for the current coding tree node is increased by an accumulated number of BT splits of the corresponding ancestor partial coding tree nodes.

To prevent the picture boundary handling process from violating the 64 × 64-L/32 × 32-C VPDU principles, an additional VPDU constraint is required at the picture boundary [38]. This constraint and the motivation behind it are described below. The left side of Fig. 11 shows one example of a violation caused by the picture boundary handling scheme described in the previous paragraphs. The dotted lines are 64 × 64-L/32 × 32-C VPDU boundaries. The left 48 × 128-L/24 × 64-C region of the 128 × 128-L/64 × 64-C partial CTU is located within a picture and marked in white. The right 80 × 128-L/40 × 64-C region of the 128 × 128-L/64 × 64-C partial CTU is located outside of the picture and marked in grey. Three VBT splits are applied recursively at three MTT depths, resulting in one 32 × 128-L/16 × 64-C CU and one 16 × 128-L/8 × 64-C CU, which are both located entirely within the picture. Since the CU height is larger than 64-L/32-C, TU splits are inferred for the two CUs. The processing order of the four TUs is as follows (marked with numbers 0-3 on the left side of Fig. 11): the top-left 32 × 64-L/16 × 32-C TU, the bottom-left 32 × 64-L/16 × 32-C TU, the top-right 16 × 64-L/8 × 32-C TU, and the bottom-right 16 × 64-L/8 × 32-C TU. This order violates the VPDU principles defined in Subsection III-B. To solve this issue, an additional VPDU constraint needs to be enforced at picture boundaries. Specifically, if the CTU size is set to 128 × 128-L/64 × 64-C, then the QT split is always applied to the root coding tree node of any partial CTU. This additional VPDU constraint is demonstrated on the right side of Fig. 11. The QT split is first applied to the partial CTU, and then two VBT splits are applied recursively at two MTT depths to each of the two QT child nodes that are not totally outside the picture. As a result, four CUs are generated, and the processing order becomes as follows (marked with numbers 0-3 on the right side of Fig. 11): the top left 32 × 64-L/16 × 32-C CU, the top right 16 × 64-L/8 × 32-C CU, the bottom left 32 × 64-L/16 × 32-C CU, and the bottom right 16 × 64-L/8 × 32-C CU. This order satisfies all VPDU principles. An example of the bottom picture boundary is omitted in this paper since it can be easily derived from an analogy.

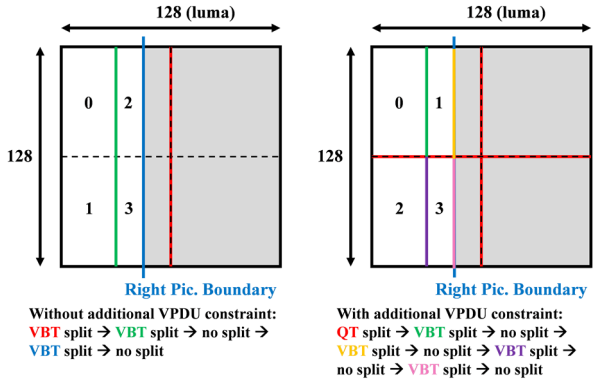

Fig. 11. Illustration of additional VPDU constraint at a picture boundary.

The following corner case of a coding tree split [39] is also considered in VVC: when a coding tree node crosses a picture boundary, and no splitting mode is allowed, the QT split is inferred to be applied to this coding tree node. For example, let us consider a case when the maximum allowed hierarchical MTT depth (MaxMttDepthY in cases of the single tree and the dual tree for luma, or MaxMttDepthC in case of the dual tree for chroma) is set to 0, and the size of a current partial coding tree node is equal to the minimum allowed QT size (MinQtSizeY in cases of the single tree and the dual tree for luma, or MinQtSizeC in case of the dual tree for chroma). In this case, any QT split is not permitted because the size of a current coding tree node reaches the minimum allowed QT size; BT and TT splits are not allowed because the maximum hierarchical MTT depth is equal to 0. Yet, in VVC, the current coding tree node is inferred to be further split by a QT split, even though no splitting mode is allowed.

E. Local Dual Tree

When too many small blocks are present in the coded picture, the average processing throughput drops in most hardware video decoders. Intra blocks cannot be processed in parallel and must be decoded sequentially. Thus, the worst-case processing throughput is dominated by the smallest intra block size. Besides that, intra prediction processes in VVC become much more complex than those in HEVC after adopting the CCLM mode, the 4-tap interpolation filters, and the position dependent prediction combination (PDPC) mode. It was identified that the worst-case hardware decoder processing throughput occurs when the CTU dual tree is switched off, and the coded CTU is full of the smallest intra CUs, where the smallest luma CB size is 4×4, and the smallest chroma CB size is 2×2 [40]–[42]. It was also noted that when the CTU dual tree is on, the smallest luma CB size is 4×4, and the smallest chroma CB size is either 4×4 or 8×2. Therefore, the worst-case scenario cannot occur when using the CTU dual tree. To improve the worst-case hardware decoder processing throughput, a local dual tree for the single tree structure is adopted into VVC. As a general rule, a chroma CB with an area less than 16 samples or/and a width less than 4 samples are disallowed in VVC.

Details of the local dual tree are elaborated in this paragraph. In the single tree, scanned from the root toward leaves, if any further split of this coding tree node will result in a chroma CB with an area less than 16 samples or width less than 4 samples, then the current coding tree node is called the smallest chroma intra prediction unit (SCIPU). All CUs within one SCIPU have to be either non-inter (intra, intra block copy (IBC), or palette) coded or inter coded. There cannot be any mixture of inter and non-inter CUs within one SCIPU. Three examples of the SCIPU are shown in Fig. 12, where the three SCIPU sizes are 8×8-L/4×4-C, 16×8-L/8×4-C, and 8×16-L/4×8-C, respectively. When one SCIPU is non-inter coded, its chroma CBs are not further split together with luma, and as a result, a local dual tree is formed. This way, the smallest luma CB size is 4×4, and the smallest chroma CB size is either 4×4 or 8×2, which alleviates the bottleneck issue of the processing throughput. For each SCIPU, a syntax element, non_inter_flag, is conditionally signaled by an encoder, indicating whether this SCIPU is all non-inter coded or all inter coded. If the current slice is I slice (always non-inter coded) or if the current SCIPU has any 4×4 luma CB (there is no inter 4×4 in VVC, so the 4×4 luma CB must be intra or IBC coded), non_inter_flag is not signaled and inferred to be 1. Otherwise, non_inter_flag is signaled before sending CUs in the SCIPU.

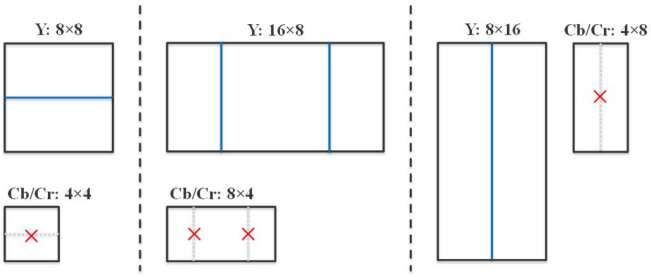

Fig. 12. Examples of local dual tree.

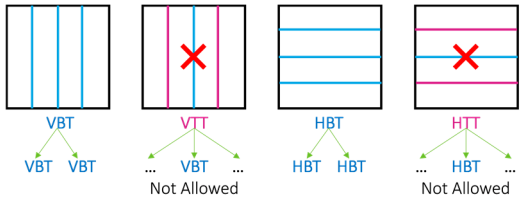

Fig. 13. Removal of redundant partitions between BT and TT.

F. Split Signaling, Configurations, and Default Settings in the VTM

In the following subsection, syntax elements used for the split signaling of a coding tree are described, together with configurations and default values in the VTM.

In VVC, a syntax element, split_cu_flag, is signaled or inferred at each coding tree node. If split_cu_flag is 0, a coding tree leaf is reached, and this coding tree node is treated as a CU; otherwise, a syntax element, split_qt_flag, is signaled or inferred to select between the QT split and the MTT split. If split_qt_flag is equal to 1, then the QT split is applied; otherwise, a syntax element, mtt_split_cu_vertical_flag, is signaled or inferred to select between the horizontal split direction and the vertical split direction; and a syntax element, mtt_split_cu_binary_flag, is signaled or inferred to choose between the BT split and the TT split. When signaling syntax elements related to the coding tree, some additional conditions allow to reduce signaling redundancy. For example, if split_qt_flag is 0 for a CU of size not larger than the maximum CU size for the BT split but larger than the maximum CU size for the TT split, mtt_split_cu_binary_flag is not signaled and inferred to be 1. Another case is related to inferring these four coding tree split information flags depending on syntax elements in either SPS or picture header (PH). For example, the maximum allowed CU size for BT split and the maximum allowed CU size for the TT split can be derived from the related parameters in either SPS or PH. Besides decreasing syntax redundancy using SPS and PH syntax elements, syntax redundancy between BT and TT is also reduced in VVC. As shown in Fig. 13, two redundant partitioning patterns are removed by syntax constraints. If a parent node is split by TT and the to-be-split node is the second child TT node, then prohibit the BT partitioning along the same direction as that of the parent node. In this case, mtt_split_cu_binary_flag is not signaled and is inferred to be 0. Nevertheless, certain syntax redundancy between QT and BT is retained in VVC. This syntax redundancy is preserved on purpose to reduce coding tree parsing complexity without sacrificing significant coding efficiency.

The abovementioned four syntax elements are all context coded by the context-based adaptive binary arithmetic coding (CABAC). It is expected that the partitioning decisions for a coding tree node may be closely related to the partitioning structures of the current and neighboring

coding tree nodes. Therefore, in VVC, 9, 6, 5, and 4 context variables are assigned respectively for entropy coding split_cu_flag, split_qt_flag, mtt_split_cu_vertical_flag, and mtt_split_cu_binary_flag. To specify how to derive the context index increment, *ctxInc*, the following variables are defined.

- $A_{sum} = 2A_{QT} + A_{BH} + A_{BV} + A_{TH} + A_{TV} - 1$, where A_{QT}, A_{BH}, A_{BV}, A_{TH}, and A_{TV} denote the availability information for QT, HBT, VBT, HTT, VTT, respectively. For example, if a QT split is allowed for a coding tree node, $A_{QT} = 1$; otherwise, $A_{QT} = 0$.
- *widthC*, *widthA*, *heightC*, and *heightL* denote the width of the current CB, the width of the above CB, the height of the current CB, and the height of the left CB, respectively.
- *qtDepthC*, *qtDepthA*, and *qtDepthL* denote the QT depth of the current CB, the QT depth of the above CB, and the QT depth of the left CB, respectively.
- $d_w = widthC \,/\, widthA$
- $d_h = heightC \,/\, heightL$
- $d = (d_w == d_h)\ ?\ 0 :\ (d_w < d_h\ ?\ 1 :\ 2)$
- *mttDepthC* denotes the MTT depth of the current CB

The variable *ctxInc* for each of the four syntax elements is derived as follows:

- For split_cu_flag, *ctxInc* in the range of 0 to 8, inclusive, is derived as follows:
  $ctxInc = ((widthC > widthA)\ ?\ 1 :\ 0) + ((heightC > heightL)\ ?\ 1 :\ 0) + 3 * ((A_{sum} == 0)\ ?\ 0 :\ (A_{sum} - 1) \gg 1)$.
- For split_qt_flag, *ctxInc* in the range of 0 to 5, inclusive, is derived as follows:
  $ctxInc = ((qtDepthA > qtDepthC)\ ?\ 1 :\ 0) + ((qtDepthL > qtDepthC)\ ?\ 1 :\ 0) + 3 * ((qtDepthC < 2)\ ?\ 0 :\ 1)$.
- For mtt_split_cu_vertical_flag, *ctxInc* in the range of 0 to 4, inclusive, is derived as follows:
  $ctxInc = ((A_{BH} + A_{TH}) == (A_{BV} + A_{TV}))\ ?\ d :\ (((A_{BH} + A_{TH}) < (A_{BV} + A_{TV}))\ ?\ 3 :\ 4)$.
- For mtt_split_cu_binary_flag, *ctxInc* in the range of 0 to 3, inclusive, is derived as follows: $ctxInc = 2 *$ mtt_split_cu_vertical_flag $+ (mttDepthC <= 1\ ?\ 1 :\ 0)$.

Table I shows the related syntax elements to the coding tree in the SPS and their corresponding values used in the VTM CTCs [43], [44]. In VVC, the following settings are allowed:

- The CTB size for luma (CtbSizeY) can be 32, 64, or 128.
- The minimum CB size for luma (MinCbSizeY) must be less than or equal to min(64, CtbSizeY).
- The maximum luma transform size (MaxTransformSize) can be 32 or 64.
- MinQtSizeY must be in the range between MinCbSizeY and min(64, CtbSizeY).
- MaxMttDepthY must be in the range between 0 and 2 *(CtbLog2SizeY − MinCbLog2SizeY), inclusive, where CtbLog2SizeY is the base 2 logarithm of the CtbSizeY and MinCbLog2SizeY is equal to the base 2 logarithm of MinCbSizeY.
- MaxBtSizeY must be in the range between MinQtSizeY and min(64, CtbSizeY) inclusive if intra slice with CST is enabled; in the range of MinQtSizeY to CtbSizeY, inclusive, otherwise.

TABLE I
CODING TREE RELATED SYNTAX IN SPS

seq_parameter_set_rbsp( ) {	**CTC values**
…	
sps_log2_ctu_size_minus5	2 // CtbSizeY = 128
…	
if(ChromaArrayType != 0)	
sps_qtbtt_dual_tree_intra_flag	1 // enabling dual tree in intra slices
sps_log2_min_luma_coding_block_size_minus2	0 // MinCbSizeY = 4
sps_partition_constraints_override_enabled_flag	1 // Enabling override in PH
sps_log2_diff_min_qt_min_cb_intra_slice_luma	1 // MinQtSizeY = 8 for intra slices
sps_max_mtt_hierarchy_depth_intra_slice_luma	3 // MaxMttDepthY = 3 for intra slices
if(sps_max_mtt_hierarchy_depth_intra_slice_luma != 0) {	
sps_log2_diff_max_bt_min_qt_intra_slice_luma	2 // MaxBtSizeY = 32 for intra slices
sps_log2_diff_max_tt_min_qt_intra_slice_luma	2 // MaxTtSizeY = 32 for intra slices
}	
sps_log2_diff_min_qt_min_cb_inter_slice	1 // MinQtSizeY = 8 for inter slices
sps_max_mtt_hierarchy_depth_inter_slice	3 // MaxMttDepthY = 3 for inter slices
if(sps_max_mtt_hierarchy_depth_inter_slices != 0) {	
sps_log2_diff_max_bt_min_qt_inter_slice	4 // MaxBtSizeY = 128 for inter slices
sps_log2_diff_max_tt_min_qt_inter_slice	3 // MaxTtSizeY = 64 for inter slices
}	
if(sps_qtbtt_dual_tree_intra_flag) {	
sps_log2_diff_min_qt_min_cb_intra_slice_chroma	0 // MinQtSizeC = 4 for CST
sps_max_mtt_hierarchy_depth_intra_slice_chroma	3 // MaxMttDepthC = 3 for CST
if (sps_max_mtt_hierarchy_depth_intra_slice_chroma != 0) {	
sps_log2_diff_max_bt_min_qt_intra_slice_chroma	4 // MaxBtSizeC = 64 for CST
sps_log2_diff_max_tt_min_qt_intra_slice_chroma	3 // MaxTtSizeC = 32 for CST
}	
}	
if(CtbSizeY > 32)	
sps_max_luma_transform_size_64_flag	1 // MaxTransformSize = 64
…	

- The maximum CB size for luma to apply the TT split (MaxTtSizeY) must be in the range between MinQtSize and min(64, CtbSizeY), inclusive.

For CST, the following settings are allowed in VVC:

- MinQtSizeC must be in the range between MinCbSizeY and min(64, CtbSizeY).
- MaxMttDepthC must be in the range between 0 and 2 *(CtbLog2SizeY − MinCbLog2SizeY), inclusive, where CtbLog2SizeY is the base 2 logarithm of the CtbSizeY and MinCbLog2SizeY is the base 2 logarithm of MinCbSizeY;
- MaxBtSizeC must be in the range between 0 and min (64, CtbSizeY) − MinQtSizeC, inclusive.
- The maximum CB size for chroma to apply TT split (MaxTtSizeC) must be in the range between 0 and min(64, CtbSizeY) − MinQtSizeC, inclusive.

Table II shows the corresponding coding tree-related syntax elements in the PH. {MinQtSizeY, MaxMttDepth, MaxBtSizeY, MaxTtSizeY} for intra slices, {MinQtSizeC, MaxMttDepthC, MaxBtSizeC, MaxTtSizeC} for CST, and {MinQtSizeY, MaxMttDepth, MaxBtSizeY, MaxTtSizeY} for inter slices. Initially, they are set in the SPS but can be overridden by the corresponding 12 syntax elements in the PH. These 12 parameters are critical for the rate-distortion (RD) performance and encoding time. In the VTM CTCs, the corresponding four parameters for intra slices, four parameters for CST, and MaxMttDepthY for inter slices are fixed as signaled in SPS and are not overridden in the PH. At the same time, to achieve a good tradeoff between the RD performance and the encoding time, MaxBtSizeY for inter slices can be adjusted in the PH. When an average CU size of the previous picture, which has the same quantization parameter (QP) layer index of the group of pictures (GOP) structure, is larger than a predefined threshold, then MaxBtSizeY for the current inter slice is set to a larger value and vice versa. Here, the QP layer index of the i-th picture is defined as the smallest non-negative

TABLE II
CODING TREE RELATED SYNTAX IN PH

```
picture_header_rbsp( ) {
  ...
  if( sps_partition_constraints_override_enabled_flag ) {
    ph_partition_constraints_override_flag
  if( ph_intra_slice_allowed_flag ) {
    if( ph_partition_constraints_override_flag ) {
      ph_log2_diff_min_qt_min_cb_intra_slice_luma
      ph_max_mtt_hierarchy_depth_intra_slice_luma
      if( ph_max_mtt_hierarchy_depth_intra_slice_luma != 0 ) {
        ph_log2_diff_max_bt_min_qt_intra_slice_luma
        ph_log2_diff_max_tt_min_qt_intra_slice_luma
      }
      if( sps_qtbtt_dual_tree_intra_flag ) {
        ph_log2_diff_min_qt_min_cb_intra_slice_chroma
        ph_max_mtt_hierarchy_depth_intra_slice_chroma
        if( ph_max_mtt_hierarchy_depth_intra_slice_chroma != 0 ) {
          ph_log2_diff_max_bt_min_qt_intra_slice_chroma
          ph_log2_diff_max_tt_min_qt_intra_slice_chroma
        }
      }
    }
    ...
  }
  if( ph_inter_slice_allowed_flag ) {
    if( ph_partition_constraints_override_flag ) {
      ph_log2_diff_min_qt_min_cb_inter_slice
      ph_max_mtt_hierarchy_depth_inter_slice
      if( ph_max_mtt_hierarchy_depth_inter_slice != 0 ) {
        ph_log2_diff_max_bt_min_qt_inter_slice
        ph_log2_diff_max_tt_min_qt_inter_slice
      }
    }
    ...
  }
  ...
}
```

integer value k for which there is an integer y such that $i = y^{*}2^{(m-k)}$, where m is base two logarithm of half of the GOP size.

G. Encoding Algorithm and Speedup Methods in the VTM

This subsection describes the related VTM encoding algorithm. A recursive function xCompressCU() starts from the root of the coding tree, i.e., CTU, and visits other coding tree nodes for all nonzero coding tree depths. The following options are tested inside the xCompressCU(): 1) one coding tree node of non-split, and 2) child coding tree nodes from the HBT split, VBT split, HTT split, VTT split, and QT split, with 2, 2, 3, 3, and 4 times of recursive calls of the xCompressCU() function, respectively. The best split decision for a coding tree node is updated by finding the minimum RD cost among the abovementioned variants. Depending on the parameters defined in the previous subsection, some of the split options may become invalid and be skipped. On top of the described encoding flow, to reduce the encoding time, six encoder speedup methods are adopted in the VTM. These speedup methods are summarized below.

[Method 1] The minimum and maximum QT depths are derived from the QT depths of the neighboring CUs at left, bottom-left, above, above-right positions relative to the current coding tree node. If the derived minimum QT depth minus m is greater than the QT depth of the current coding tree node, then the QT split is enforced (i.e., HBT, VBT, HTT, and VTT splits of the current coding tree node are skipped). Here, $m = 1$, if the derived minimum QT depth is greater than 0, and $m = 0$, otherwise. If the derived maximum QT depth plus n is less than or equal to the QT depth of the current coding tree node, then the QT split is prohibited. Here, $n = 1$, if the derived maximum QT depth is less than the maximum allowed QT depth, and $n = 0$, otherwise.

[Method 2] Skip testing the HTT split of the current coding tree node if the HBT split has been already tested for this coding tree node, and the best split decision of this coding tree node is non-split with zero residual. Similarly, skip checking the VTT split of the current coding tree node if the VBT split has already been tested for this coding tree node, and the best split decision of this coding tree node is non-split with zero residual.

[Method 3] Skip testing the HBT, VBT, and QT splits of the current coding tree node if the current slice is either non-intra or allows IBC, and the skip mode is the best mode of non-split for the current coding tree node, the parent coding tree node, and the grandparent coding tree node.

[Method 4] Skip testing the QT split of the current coding tree node if: a) the HBT and VBT splits of this coding tree node have been tested, and the best split decision of this coding tree node does not result in a large BT depth; b) the CU size represented by the current coding tree node does not exceed 64×64 in units of luma samples.

[Method 5] Test the QT split of the current coding tree node right after testing non-split for this coding tree node if the following conditions apply:

- at least one of the two conditions below is true:
  - at least one CU on the left or above the current coding tree node has a QT depth larger than the QT depth of the current coding tree node;
  - the CU width represented by the current coding tree node is greater than or equal to S in units of luma samples, where $S = 32/64/128$ for QP layer index 0/1/2;
- the QT split of the current coding tree node is allowed.

The HBT, VBT, HTT, and VTT splits are skipped if the following conditions apply:

- the QT split is tested right after testing non-split of the current coding tree node;
- the best split decision for a current coding tree node is QT split.

[Method 6] As shown in Fig. 14, the same CU may be tested multiple times when testing different partitionings. Therefore, to reduce computational overhead, some of the previously obtained encoding results can be reused as follows. If a current coding tree node was previously tested, the following two rules could be conditionally applied. 1) Always test the same split option, equal to the best split option of the current coding tree node. 2) If the best split option of the current coding tree node tested earlier is non-split, skip testing the HBT, VBT, HTT, VTT, and QT splits; otherwise, skip testing non-split.

A more detailed analysis of these encoder speedup methods can be found in [45]. The presented above algorithms are included in the paper for reference, as those were already considered by the JVET experts and included in the VTM software. These six methods should not be regarded as an

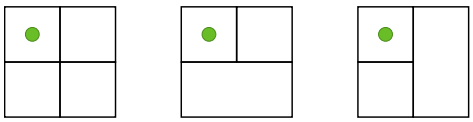

Fig. 14. Illustration of testing the same CU multiple times.

TABLE III
RESULTS OF SETTING CTU SIZE TO 64 × 64 COMPARED TO 128 × 128

	All Intra				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	0.37%	3.03%	2.58%	102%	99%
Class A2	0.62%	1.01%	0.36%	101%	103%
Class B	0.39%	0.14%	0.89%	101%	106%
Class C	0.31%	0.36%	0.79%	100%	105%
Class E	0.39%	-0.19%	2.25%	100%	108%
Overall	0.41%	0.76%	1.29%	101%	104%
Class D	0.12%	-0.09%	0.50%	98%	102%
Class F	-1.12%	-1.37%	-0.95%	88%	107%
	Random Access				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	3.15%	4.53%	5.07%	77%	98%
Class A2	3.17%	4.77%	4.98%	83%	92%
Class B	1.70%	3.72%	4.93%	87%	100%
Class C	0.38%	1.39%	2.02%	96%	101%
Overall	1.93%	3.47%	4.19%	86%	98%
Class D	0.09%	0.41%	0.75%	102%	98%
Class F	0.19%	0.51%	1.43%	86%	108%
	Low Delay B				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class B	1.64%	3.78%	6.09%	85%	92%
Class C	0.26%	0.63%	1.19%	94%	101%
Class E	2.41%	4.21%	4.45%	76%	103%
Overall	1.37%	2.84%	4.05%	86%	97%
Class D	-0.18%	-1.00%	0.22%	99%	110%
Class F	1.16%	1.73%	2.63%	85%	108%

extensive list of all possible encoder speedups. Any encoder complexity depends on the implementation and the speedup methods used.

IV. EXPERIMENTAL RESULTS

In the VTM CTCs [44], there are 3, 3, 5, 4, 4, 3, and 4 test sequences in Classes A1, A2, B, C, D, E, and F, respectively. Corresponding spatial resolutions of Classes A1, A2, B, C, D, and E in units of luma samples are 3840 × 2160, 3840 × 2160, 1920 × 1080, 832 × 480, 416 × 240, and 1280 × 720, respectively. The spatial resolution of Class F ranges from 832 × 480 to 1920 × 1080. Class F contains test sequences with screen content and mixed content, while the other classes of test sequences include camera-captured content only. Class E is skipped in the RA condition. Classes A1 and A2 are omitted in the LB condition. The overall average does not include Classes D and F. In this paper, only the class averages and the overall average are shown to avoid too much data. Y BD-rates, Cb BD-rates, Cr BD-rates, encoding time ratios, and decoding time ratios over the anchor for all conducted experiments are provided in the following paragraphs.

Table III shows the VTM9.0 results of setting a CTU size to 64 × 64 against 128 × 128. More significant losses are observed for sequences in Classes A1 and A2. These results experimentally demonstrate that increasing the maximum CTU size from 64 × 64 to 128 × 128 allows for achieving better performance for larger sequences. However, this level of coding gain improvement is much lower than that of the VVC block partitioning, as shown in the next paragraph.

TABLE IV
DIFFERENT SETTINGS OF CU SPLITTING METHODS AND CTU DUAL TREE

Setting	QT Split	BT Split	TT Split	CTU Dual Tree
(Anchor) Setting 1	On	Off	Off	Off
Setting 2	On	On	Off	Off
Setting 3	On	Off	On	Off
Setting 4	On	On	On	Off
(VTM Default) Setting 5	On	On	On	On

TABLE V
RESULTS OF SETTING 2 (QT SPLIT ON, BT SPLIT ON, TT SPLIT OFF, CTU DUAL TREE OFF) COMPARED AGAINST SETTING 1 (QT SPLIT ON, BT SPLIT OFF, TT SPLIT OFF, CTU DUAL TREE OFF)

	All Intra				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-3.49%	-5.27%	-7.33%	456%	110%
Class A2	-4.12%	-9.53%	-8.61%	561%	110%
Class B	-5.09%	-9.42%	-10.13%	600%	110%
Class C	-7.41%	-9.05%	-9.49%	659%	127%
Class E	-8.16%	-10.51%	-9.83%	526%	108%
Overall	-5.69%	-8.84%	-9.22%	566%	113%
Class D	-7.22%	-9.27%	-9.36%	581%	132%
Class F	-12.56%	-14.22%	-14.88%	536%	119%
	Random Access				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-6.78%	-8.68%	-11.03%	360%	97%
Class A2	-8.97%	-13.29%	-12.36%	335%	93%
Class B	-7.94%	-12.89%	-13.38%	335%	94%
Class C	-9.23%	-10.91%	-11.61%	325%	91%
Overall	-8.26%	-11.60%	-12.23%	337%	93%
Class D	-8.87%	-12.22%	-12.23%	291%	92%
Class F	-14.18%	-15.18%	-15.95%	262%	90%
	Low Delay B				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class B	-7.00%	-13.13%	-13.91%	316%	105%
Class C	-7.36%	-11.07%	-12.10%	286%	100%
Class E	-8.21%	-13.22%	-13.13%	233%	100%
Overall	-7.42%	-12.46%	-13.11%	283%	102%
Class D	-7.46%	-11.12%	-12.28%	235%	91%
Class F	-14.11%	-17.37%	-19.27%	216%	97%

To demonstrate the benefits of the BT split, the TT split, and the CTU dual tree in the VTM, five settings are shown in Table IV. They were simulated using VTM9.0, where Setting 1 is treated as the anchor, and Setting 5 is the same as the VTM default setting. Tables V, VI, VII, and VIII summarize the results of comparing Settings 2, 3, 4, and 5 against Setting 1, respectively. It is worth noting that if comparing Tables V, VI, and VII, gains from BT and TT do not entirely add up. One of the reasons for that may be fast algorithms applied during encoding, which affect decisions made by an already very complex encoding algorithm. Since different partitioning options are checked recursively, even if one additional option is added, the complexity increases exponentially, not linearly. Thus, the increase will be even more without any speedups than those shown in Tables V, VI, and VII. According to Tables V, VI, VII, and VIII, evidence is reported that the BT split, the TT split, and the CTU dual tree provide significant coding gains in all tested cases. For example, in Table VIII, Setting 5 (the VTM default setting) is compared against Setting 1 (QT split only, no BT split, no TT split, no CTU dual tree). And Setting 1 is similar to (but not

TABLE VI

RESULTS OF SETTING 3 (QT SPLIT ON, BT SPLIT OFF, TT SPLIT ON, CTU DUAL TREE OFF) COMPARED AGAINST SETTING 1 (QT SPLIT ON, BT SPLIT OFF, TT SPLIT OFF, CTU DUAL TREE OFF)

	All Intra				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-1.53%	-2.19%	-3.15%	285%	107%
Class A2	-2.28%	-5.37%	-4.68%	326%	116%
Class B	-2.75%	-5.77%	-6.40%	321%	108%
Class C	-4.04%	-5.62%	-5.74%	314%	116%
Class E	-5.00%	-7.77%	-7.49%	289%	103%
Overall	-3.13%	-5.41%	-5.61%	309%	110%
Class D	-3.66%	-5.56%	-5.14%	277%	110%
Class F	-8.19%	-9.77%	-10.62%	294%	111%
	Random Access				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-3.35%	-4.21%	-5.51%	250%	97%
Class A2	-5.94%	-8.83%	-8.06%	249%	95%
Class B	-4.28%	-7.20%	-7.45%	243%	96%
Class C	-5.49%	-6.43%	-6.59%	239%	93%
Overall	-4.75%	-6.72%	-6.95%	244%	95%
Class D	-4.15%	-6.06%	-5.81%	204%	97%
Class F	-9.47%	-10.41%	-10.90%	188%	96%
	Low Delay B				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class B	-3.99%	-7.81%	-8.14%	248%	109%
Class C	-4.38%	-6.86%	-7.23%	230%	99%
Class E	-3.96%	-6.49%	-7.02%	187%	99%
Overall	-4.11%	-7.16%	-7.56%	225%	103%
Class D	-3.32%	-6.08%	-6.18%	186%	99%
Class F	-9.89%	-12.43%	-13.07%	183%	99%

TABLE VII

RESULTS OF SETTING 4 (QT SPLIT ON, BT SPLIT ON, TT SPLIT ON, CTU DUAL TREE OFF) COMPARED AGAINST SETTING 1 (QT SPLIT ON, BT SPLIT OFF, TT SPLIT OFF, CTU DUAL TREE OFF)

	All Intra				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-4.07%	-5.87%	-8.38%	869%	110%
Class A2	-5.01%	-11.44%	-10.32%	1174%	110%
Class B	-6.07%	-11.26%	-12.14%	1261%	115%
Class C	-8.76%	-11.38%	-11.92%	1495%	121%
Class E	-9.66%	-13.20%	-12.71%	1088%	108%
Overall	-6.76%	-10.74%	-11.26%	1187%	114%
Class D	-8.30%	-11.43%	-11.48%	1298%	118%
Class F	-14.52%	-16.87%	-18.01%	1098%	116%
	Random Access				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-8.10%	-10.34%	-13.13%	670%	103%
Class A2	-11.12%	-16.29%	-15.20%	608%	100%
Class B	-9.85%	-15.54%	-16.05%	615%	102%
Class C	-11.60%	-13.95%	-14.84%	656%	99%
Overall	-10.22%	-14.23%	-14.97%	635%	101%
Class D	-11.02%	-15.34%	-15.81%	545%	106%
Class F	-16.77%	-18.50%	-18.93%	459%	99%
	Low Delay B				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class B	-8.95%	-15.90%	-16.48%	608%	105%
Class C	-9.71%	-14.44%	-15.54%	575%	99%
Class E	-10.36%	-15.74%	-16.44%	336%	95%
Overall	-9.55%	-15.37%	-16.16%	514%	101%
Class D	-9.86%	-15.17%	-15.68%	441%	94%
Class F	-17.38%	-21.73%	-22.62%	361%	99%

the same as) HEVC block partitioning structure under RA condition. Results are as follows: 10.33% Y, 17.49% Cb, and 18.27% Cr BD-rate savings, with 617% encoding and 94% decoding runtimes. In [18], it is reported that compared to the HEVC reference software HM13.0 default setting, the preliminary development of the new block partitioning structure built on top of HM13.0 with QT split on, BT split on, TT split off (TT was not proposed at that time), and CTU dual tree on,

TABLE VIII

RESULTS OF SETTING 5 (QT SPLIT ON, BT SPLIT ON, TT SPLIT ON, CTU DUAL TREE ON) COMPARED AGAINST SETTING 1 (QT SPLIT ON, BT SPLIT OFF, TT SPLIT OFF, CTU DUAL TREE OFF)

	All Intra				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-4.71%	-14.55%	-10.97%	543%	111%
Class A2	-5.17%	-19.27%	-16.06%	798%	113%
Class B	-6.50%	-20.33%	-23.31%	843%	104%
Class C	-9.22%	-16.64%	-17.78%	1050%	115%
Class E	-9.79%	-20.52%	-21.09%	724%	97%
Overall	-7.13%	-18.40%	-18.44%	795%	108%
Class D	-8.67%	-15.94%	-15.84%	952%	120%
Class F	-14.88%	-21.25%	-23.30%	849%	107%
	Random Access				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-8.16%	-12.20%	-13.36%	660%	93%
Class A2	-11.17%	-19.69%	-17.30%	597%	94%
Class B	-9.96%	-20.67%	-22.80%	596%	95%
Class C	-11.80%	-15.83%	-17.01%	630%	93%
Overall	-10.33%	-17.49%	-18.27%	617%	94%
Class D	-11.16%	-17.86%	-18.28%	520%	97%
Class F	-16.91%	-20.37%	-21.79%	426%	92%
	Low Delay B				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class B	-8.97%	-16.98%	-18.32%	611%	110%
Class C	-9.76%	-14.78%	-16.28%	577%	100%
Class E	-10.51%	-18.92%	-19.68%	335%	95%
Overall	-9.62%	-16.73%	-17.98%	516%	103%
Class D	-9.88%	-16.74%	-17.41%	428%	89%
Class F	-17.34%	-22.21%	-24.14%	363%	94%

results in BD-rate reductions of 5.97% (Y), 13.89% (Cb), and 15.29% (Cr), with 158% encoding and 106% decoding runtimes under the RA condition. Y PSNR values of Setting 5 and Setting 1 for Tango2 sequence encoded with RA condition at various bitrates are shown in Fig. 15. Setting 5 and Setting 1 are plotted with squares and diamonds, respectively. As it can be seen, a more flexible partitioning (Setting 5) provides higher PSNR values at all tested bitrates. It is reported that Y, Cb, Cr BD-rate savings of Setting 5 compared against Setting 1 are 9.57%, 16.35%, and 15.06%, respectively, which is consistent with the curves in Fig. 15. Cb/Cr PSNR-bitrate curves of Tango2 and Y/Cb/Cr PSNR-bitrate curves of other sequences show the same trend, that Setting 5 is better than Setting 1 at all tested bitrates, so, for the sake of brevity, they are omitted in this paper. When comparing Tables VII and VIII, one can notice a substantial reduction in encoder runtime, particularly for the AI condition. It is caused by the CTU dual tree being switched off in Setting 4. Chroma CUs in the CTU dual tree tend to be larger than those in the CTU single tree, and testing small chroma CUs is often skipped by fast encoding algorithms. Also, it should be noted that the Class C sequences, in many cases, have the highest encoder complexity increase. This increase may be because the algorithms of the VTM were developed targeting more on encoding videos of UHD and higher resolutions. Thus, the encoding time for the smaller Class C sequences was less optimized since it is not a bottleneck of the computing resources, unlike sequences from Classes A and B.

Table IX summarizes the VVC picture boundary handling results against the HEVC picture boundary handling. VTM9.0 uses the VTM CTC default settings, and it is compared to the modified VTM9.0, where the QT splits are always applied to all partial CTUs. As expected, BD-rates are

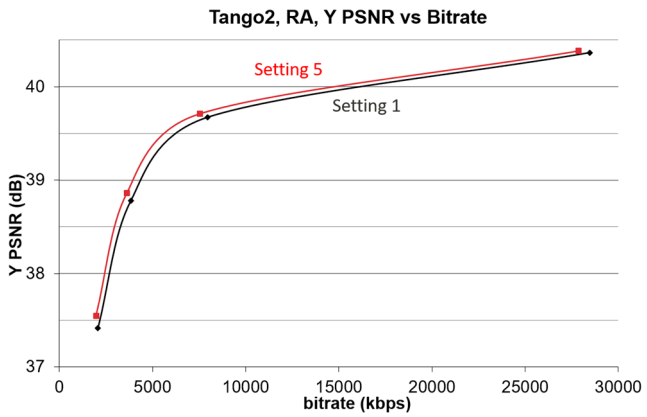

Fig. 15. Performance of Setting 5 (red squares) and Setting 1 (black diamonds) at various bitrates (RA CTC QPs 22, 27, 32 and 37) of Tango2 sequence.

TABLE IX

RESULTS OF VVC PICTURE BOUNDARY HANDLING USING VTM9.0 DEFAULT SETTING COMPARED AGAINST HEVC PICTURE BOUNDARY HANDLING

	All Intra				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-0.07%	-0.25%	-0.28%	99%	100%
Class A2	-0.03%	-0.15%	-0.08%	99%	94%
Class B	-0.16%	-0.69%	-0.66%	99%	92%
Class C	-0.01%	-0.05%	-0.04%	98%	94%
Class E	-0.11%	-0.17%	-0.18%	100%	93%
Overall	-0.08%	-0.30%	-0.28%	99%	94%
Class D	-0.11%	-0.20%	-0.22%	103%	97%
Class F	-0.14%	-0.53%	-0.68%	98%	95%
	Random Access				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	-0.38%	-0.48%	-0.57%	98%	107%
Class A2	-0.24%	-0.23%	-0.16%	97%	110%
Class B	-0.86%	-1.24%	-1.13%	98%	109%
Class C	-0.07%	-0.17%	-0.19%	98%	117%
Overall	-0.43%	-0.60%	-0.57%	98%	111%
Class D	-0.51%	-0.59%	-0.55%	104%	121%
Class F	-0.41%	-0.61%	-0.77%	98%	108%
	Low Delay B				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class B	-0.92%	-1.43%	-1.62%	102%	107%
Class C	-0.09%	-0.16%	0.07%	100%	91%
Class E	-1.12%	-1.12%	-1.00%	96%	87%
Overall	-0.69%	-0.93%	-0.90%	100%	96%
Class D	-0.72%	-1.37%	-1.44%	103%	87%
Class F	-0.61%	-1.25%	-1.56%	99%	92%

affected by spatial resolutions. The Class B (1920 × 1080) test sequences and the Class E (1280 × 720) test sequences have larger coding gains, and luma BD-rates decrease roughly by 0.9% and 1.1%, respectively, under non-AI conditions.

Performance results (the RA condition and the test sequences from Classes A and B) of the QT + MTT structure over HEVC are reported in [46]. The coding gains and the encoder complexity depend on the allowed maximum depths for QT, BT, and TT partitions. On average, for the Class A test sequences, the reported luma BD-rate saving of QT + MTT varies between 17.2% and 21.4%, with encoder runtime up to 314%.

Table X summarizes the results of enabling the local dual tree compared to disabling the local dual tree. The VTM CTC default settings are compared to the modified VTM9.0 without

TABLE X

RESULTS OF ENABLING LOCAL DUAL TREE COMPARED AGAINST DISABLING LOCAL DUAL TREE ON VTM9.0

	All Intra				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	0.00%	0.00%	0.00%	100%	100%
Class A2	0.00%	0.00%	0.00%	101%	101%
Class B	0.00%	0.00%	0.00%	101%	101%
Class C	0.00%	0.00%	0.00%	101%	99%
Class E	0.00%	0.00%	0.00%	101%	100%
Overall	0.00%	0.00%	0.00%	101%	100%
Class D	0.00%	0.00%	0.00%	101%	97%
Class F	0.00%	0.00%	0.00%	101%	100%
	Random Access				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class A1	0.05%	0.03%	0.19%	100%	100%
Class A2	0.10%	0.28%	0.27%	100%	100%
Class B	0.10%	-0.13%	0.01%	100%	101%
Class C	0.25%	-0.24%	-0.23%	99%	100%
Overall	0.13%	-0.05%	0.04%	100%	100%
Class D	0.37%	0.67%	0.58%	98%	99%
Class F	0.23%	0.29%	0.23%	97%	101%
	Low Delay B				
	Y BD-rate	Cb BD-rate	Cr BD-rate	EncT	DecT
Class B	0.14%	0.45%	0.24%	101%	100%
Class C	0.35%	0.39%	0.17%	102%	102%
Class E	0.15%	0.56%	0.10%	101%	103%
Overall	0.21%	0.46%	0.18%	101%	101%
Class D	0.35%	-0.01%	0.47%	101%	105%
Class F	0.70%	1.05%	-0.03%	97%	101%

using the local dual tree. Since the CTU dual tree is always used in the AI condition and the local dual tree affects only the single tree structure, all BD-rates are zero for AI. In the RA and LB conditions, luma BD-rates increase only by roughly 0.1% and 0.2%, respectively. It is claimed that a significant improvement in hardware decoder processing throughput completely justifies such minor coding efficiency loss caused by the local dual tree.

V. CONCLUSION

In this paper, the block partitioning structure in VVC is introduced. The block partitioning in VVC is more flexible than in HEVC and contributes significantly to coding efficiency and encoder run time. Several design concepts for improving coding efficiency and reducing hardware complexity are illustrated. Related experiments demonstrate the benefits of various aspects of the VVC block partitioning structure. It is reported that on average 10.33% Y BD-rate saving, 17.49% Cb BD-rate saving, and 18.27% Cr BD-rate savings are achieved for the RA condition with 6 times of encoding runtime in the VTM.

ACKNOWLEDGMENT

The authors would like to thank Chih-Wei Hsu, Ching-Yeh Chen, Tzu-Der Chuang, Chia-Ming Tsai, Zhi-Yi Lin, Shaw-Min Lei, Ru-Ling Liao, Adam Wieckowski, and many other experts in the JVET for their valuable comments, collaborative suggestions, and significant contributions to the VVC block partitioning structure and this article.

REFERENCES

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[2] *High Efficiency Video Coding (HEVC)*, document ITU-T Rec. H.265, ISO/IEC 23008-2 (MPEG-H Part 2), Apr. 2013.

[3] *High Efficiency Video Coding Test Model (HM) Reference Software of the JCT-VC of ITU-T VCEG and ISO/IEC MPEG*. Accessed: Jun. 13, 2021. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/

[4] J. Chen, E. Alshina, G. J. Sullivan, J.-R. Ohm, and J. Boyce, *Algorithm Description of Joint Exploration Test Model 7 (JEM 7)*, document JVET-G1001, Output Document of the 7^{th} JVET Meeting, Jul. 2017.

[5] *Joint Exploration Test Model (JEM) Reference Software of the JVET of ITU-T VCEG and ISO/IEC MPEG*. Accessed: Jun. 13, 2021. [Online]. Available: https://jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/

[6] J. Chen, M. Karczewicz, Y.-W. Huang, K. Choi, J.-R. Ohm, and G. J. Sullivan, "The joint exploration model (JEM) for video compression with capability beyond HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1208–1225, May 2020.

[7] A. Segall, V. Baroncini, J. Boyce, J. Chen, and T. Suzuki, *Joint Call for Proposals on Video Compression With Capability Beyond HEVC*, document JVET-H1002, Output Document of the 8^{th} JVET Meeting, Oct. 2017.

[8] V. Baroncini, J.-R. Ohm, and G. J. Sullivan, *Report of Results From the Call for Proposals on Video Compression With Capability Beyond HEVC*, document JVET-J1003, Output Document of the 10^{th} JVET meeting, Apr. 2018.

[9] J. Pfaff *et al.*, "Video compression using generalized binary partitioning, trellis coded quantization, perceptually optimized encoding, and advanced prediction and transform coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1281–1295, May 2020.

[10] X. Xiu *et al.*, "A unified video codec for SDR, HDR, and 360° video applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1296–1310, May 2020.

[11] Y.-W. Huang *et al.*, "A VVC proposal with quaternary tree plus binary-ternary tree coding block structure and advanced coding techniques," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1311–1325, May 2020.

[12] K. Choi *et al.*, "Video codec using flexible block partitioning and advanced prediction, transform and loop filtering technologies," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1326–1345, May 2020.

[13] W.-J. Chien *et al.*, "Hybrid video codec based on flexible block partitioning with extensions to the joint exploration model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1346–1360, May 2020.

[14] K. Misra, A. Segall, and F. Bossen, "Tools for video coding beyond HEVC: Flexible partitioning, motion vector coding, luma adaptive quantization, and improved deblocking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1361–1373, May 2020.

[15] *Versatile Video Coding (VVC)*, document ITU-T Rec. H.266, ISO/IEC 23090-3 (MPEG-I Part 3), Feb. 2021.

[16] *Advanced Video Coding (AVC)*, document ITU-T Rec. H.264 | ISO/IEC 14496-10 (MPEG-4 Part 10), Mar. 2005.

[17] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, "Block partitioning structure in the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1697–1706, Dec. 2012.

[18] J. An, Y.-W. Chen, K. Zhang, H. Huang, Y.-W. Huang, and S.-M. Lei, *Block Partitioning Structure for Next Generation Video Coding*, document MPEG m37524 and ITU-T SG16 COM16-C966, Input Document of MPEG and ITU-T SG16 Meeting, Oct. 2015.

[19] J. An, H. Huang, K. Zhang, Y.-W. Huang, and S.-M. Lei, *Quadtree Plus Binary Tree Structure Integration With JEM Tools*, document JVET-B0023, Input Document of the 2nd JVET Meeting, Feb. 2016.

[20] H. Huang, K. Zhang, Y.-W. Huang, and S.-M. Lei, *EE2.1: Quadtree Plus Binary Tree Structure Integration With JEM Tools*, document JVET-C0024, Input Document of the 3rd JVET Meeting, Jun. 2016.

[21] X. Li *et al.*, *Multi-Type Tree*, document JVET-D0117, Input Document of the 4th JVET Meeting, Oct. 2016.

[22] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *IEEE Trans. Image Process.*, vol. 3, no. 3, pp. 327–331, May 1994.

[23] G. M. Schuster and A. K. Katsaggelos, "An optimal quadtree-based motion estimation and motion-compensated interpolation scheme for video compression," *IEEE Trans. Image Process.*, vol. 7, no. 11, pp. 1505–1523, Nov. 1998.

[24] C.-W. Hsu, T.-D. Chuang, C.-Y. Chen, Y.-W. Huang, and S.-M. Lei, *CE1-Related: Constraint for Binary and Ternary Partitions*, document JVET-K0556, Input Document of the 11th JVET Meeting, Jul. 2018.

[25] C.-M. Tsai, C.-W. Hsu, T.-D. Chuang, C.-Y. Chen, Y.-W. Huang, and S.-M. Lei, *CE1.2.1: Constraint for Binary and Ternary Partitions*, document JVET-L0081, Input Document of the 12th JVET Meeting, Oct. 2018.

[26] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD Curves*, document VCEG-M33, Input Document of the 13th VCEG Meeting, Apr. 2001.

[27] G. Bjøntegaard, *Improvements of the BD-PSNR Model*, document VCEG-AI11, Input Document of the 35th VCEG Meeting, Jul. 2008.

[28] T.-D. Chuang, C.-Y. Chen, Y.-W. Huang, and S.-M. Lei, *CE1-Related: Separate Tree Partitioning at* 64×64*-Luma/*32×32*-Chroma Unit Level*, document JVET-K0230, Input Document of the 11th JVET Meeting, Jul. 2018.

[29] Y. Zhao and H. Yang, *CE2-Related: CCLM for Dual Tree With* 32×32 *Latency*, document JVET-O0196, Input Document of the 15th JVET Meeting, Jul. 2019.

[30] C.-M. Tsai, C.-W. Hsu, Y.-W. Huang, and S.-M. Lei, *CE2-Related: Luma-Chroma Latency Reduction for Chroma Separate Tree*, document JVET-O0273, Input Document of the 15th JVET Meeting, Jul. 2019.

[31] C.-M. Tsai, T.-D. Chuang, C.-W. Hsu, Y.-W. Huang, and S.-M. Lei, *Luma-Chroma Dependency Reduction for Chroma Separate Tree by Constraining CCLM Usage*, document JVET-O0274, Input Document of the 15th JVET Meeting, Jul. 2019.

[32] Y. Zhao *et al.*, *Draft Text for CCLM Restriction to Reduce Luma-Chroma Latency for Chroma Separate Tree*, document JVET-O1124, Input Document of the 15th JVET Meeting, Jul. 2019.

[33] M. Karczewicz *et al.*, "VVC in-loop filters," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Apr. 9, 2021, doi: 10.1109/TCSVT.2021.3072297.

[34] J. Chen *et al.*, *Non-CE2: Unification of Chroma Residual Scaling Design*, document JVET-O1109, Input Document 15th JVET Meeting, Jul. 2019.

[35] S.-T. Hsiang and S.-M. Lei, *CE1.2.0.10: CU Partitioning Along Picture Boundaries*, document JVET-K0224, Input Document of the 11th JVET Meeting, Jul. 2018.

[36] A. Wieckowski *et al.*, *CE1-Related: Joint Proposal for Picture Boundary Partitioning by Fraunhofer HHI and Huawei*, document JVET-K0554, Input Document of the 11th JVET Meeting, Jul. 2018.

[37] H. Gao, Z. Zhao, E. Steinbach, and J. Chen, "Improving picture boundary handling for video coding beyond HEVC," in *Proc. IEEE Int. Conf. Vis. Commun. Image Process. (VCIP)*, Taichung, Taiwan, Dec. 2018, pp. 1–4.

[38] C.-M. Tsai *et al.*, *CE1-Related: Picture Boundary CU Split Satisfying the VPDU Constraint*, document JVET-M0888, Input Document of the 13th JVET Meeting, Jan. 2019.

[39] R.-L. Liao *et al.*, *On Block Partitioning at Picture Boundary*, document JVET-Q0330, Input Document of the 17th JVET Meeting, Jan. 2020.

[40] Z.-Y. Lin, T.-D. Chuang, C.-Y. Chen, Y.-W. Huang, and S.-M. Lei, *CE3-Related: Shared Reference Samples for Multiple Chroma Intra CBs*, document JVET-M0169, Input Document of the 13th JVET Meeting, Jan. 2019.

[41] Z.-Y. Lin, T.-D. Chuang, C.-Y. Chen, Y.-W. Huang, and S.-M. Lei, *CE3-2.2: Shared Reference Samples for Multiple Chroma Intra CBs*, document JVET-N0081, Input Document of the 14th JVET Meeting, Mar. 2019.

[42] Z.-Y. Lin *et al.*, *CE3-2.1.1 and CE3-2.1.2: Removing* 2×2*,* 2×4*, and* 4×2 *Chroma CBs*, document JVET-O0050, Input Document of the 15th JVET Meeting, Jul. 2019.

[43] *Versatile Video Coding Test Model (VTM) Reference Software of the JVET of ITU-T VCEG and ISO/IEC MPEG*. Accessed: Jun. 13, 2021. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM

[44] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, *JVET Common Test Conditions and Software Reference Configurations for SDR Video*, document JVET-N1010, Output Document of the 14th JVET Meeting, Mar. 2019.

[45] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe, and T. Wiegand, "Fast partitioning decision strategies for the upcoming versatile video coding (VVC) standard," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Taipei, Taiwan, Sep. 2019, pp. 4130–4134.

[46] F. Le Leannec *et al.*, "Highly flexible coding structures for next-generation video compression standard," in *Proc. Data Compress. Conf. (DCC)*, Snowbird, UT, USA, Mar. 2019, pp. 280–289, doi: 10.1109/DCC.2019.00036.

Yu-Wen Huang received the B.S. degree in electrical engineering and the Ph.D. degree in electronics engineering from National Taiwan University, Taiwan, in June 2000 and December 2004, respectively.

He joined MediaTek Inc., Hsinchu, Taiwan, in December 2004, and is currently the Deputy Director of the Multimedia Technology Development Division. In 2006, he started attending international video coding standard meetings held by ITU-T VCEG and ISO/IEC MPEG and has been an active contributor since 2009. His current research interests include image/video coding and processing and related hardware architectures.

Jicheng An received the bachelor's degree in automation and the master's degree in control science and engineering from Central South University (CSU), Changsha, Hunan, China, in June 2006 and June 2009, respectively.

From 2009 to 2016, he was with MediaTek Inc., Beijing, China. From 2016 to 2018, he was with HiSilicon Ltd., Beijing. From 2009 to 2018, he was involved in developing new video coding algorithms for next generation video coding standards. He has contributed actively to the development of the ITU-T VCEG and ISO/IEC MPEG video coding standards HEVC and VVC. Since 2018, he has been a Senior Algorithm Engineer with Alibaba Group, Beijing, working on video coding for real-time communication. His research interests include image and video compression, block structure partitioning, and error resilient video coding.

Han Huang received the B.S. degree in computer engineering and the Ph.D. degree in signal and information processing from Beijing Jiaotong University, China, in June 2007 and November 2013, respectively.

He is currently a Senior Staff Engineer at Qualcomm Inc., San Diego, CA, USA. Before he joined Qualcomm, he was with MediaTek Inc., from November 2013 to May 2018. From September 2009 to August 2011, he was a Visiting Student with Rensselaer Polytechnic Institute, Troy, NY, USA. Since 2014, he has been contributing actively to international video coding standard meetings held by ITU-T VCEG and ISO/IEC MPEG. His current research interests include image and video coding and processing.

Xiang Li (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronic engineering from Tsinghua University, Beijing, China, and the Dr.-Ing. degree in electrical, electronic and communication engineering from the University of Erlangen-Nuremberg, Germany.

He is currently the Head of video coding standards at Tencent's Media Lab. Before joining Tencent, he was with Qualcomm, MediaTek, the Institute of Communications Engineering at RWTH Aachen University, and Siemens. He has been working in the field of video compression for years and is contributing actively to international video coding standards. He served as the chair and the co-chair in a number for *ad hoc* groups and core experiments, including the co-chair for JEM reference software and VVC reference software. He was a co-editor of MPEG-5 EVC. He has published over 50 journal articles, over 50 conference papers, more than 300 standard contributions, and holds more than 240 U.S. granted and pending patents. His research interests include video coding and processing.

Shih-Ta Hsiang received the B.S. degree in electrical engineering from National Cheng Kung University, Tainan, Taiwan, the M.S. degree in electrical engineering from the University of Florida, Gainesville, FL, USA, and the M.S. degree in mathematics and the Ph.D. degree in electrical, computer, and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA.

From 2002 to 2004, he was a Research Scientist with the Imaging Technology Department, Hewlett Packard Laboratories, Palo Alto, CA, USA. From 2004 to 2011, he was a Principal Researcher with the Multimedia Research Laboratory, Motorola Labs, Schaumburg, IL, USA. In 2011, he joined MediaTek, Taiwan, as a Senior Technical Manager, where he has been involved in research related to video coding standardization.

Kai Zhang (Senior Member, IEEE) received the B.S. degree in computer science from Nankai University, Tianjin, China, in 2004, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2011.

From 2011 to 2012, he was with Tencent Inc., Beijing. From 2012 to 2016, he was with MediaTek Inc., Beijing. In 2016, he joined Qualcomm Inc., San Diego, CA, USA. He is currently a Senior Research Scientist at Bytedance Inc., San Diego. His research interests include video/image compression, coding, processing and communication, and video coding standardization.

Han Gao (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering from Xidian University, Xi'an, China, in 2012, and the M.S. degree in electrical engineering and information technology from the Technical University of Munich (TUM), Munich, Germany, in 2015, where he is currently pursuing the Ph.D. degree. In 2016, he joined the Chair of Media Technology at TUM and the Audiovisual Technology Laboratory at Huawei Technologies, Munich. Since 2016, he has been actively involved in the development of the VVC standard that was jointly issued by ITU-T VCEG and ISO/IEC MPEG. His research focuses on image and video processing and traditional and neural network-based video compression and coding.

Jackie Ma received the M.S. degree in mathematics and the Dr. rer. nat. degree from the Technical University of Berlin, Berlin, Germany, in 2013 and 2016, respectively.

From 2012 to 2016, he was Visiting Researcher at ETH Zurich, Switzerland, the University of Cambridge, U.K., and The University of Hong Kong, Hong Kong. In 2017, he joined the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, Berlin, as a Project Manager. During the development of VVC, he has made several technical contributions and was the coordinator of the core experiments on partitioning. He is currently with the Machine Learning Group, Heinrich Hertz Institute, coordinating several projects on machine learning and medicine.

Olena Chubach received the M.S. degree in applied mathematics from Odesa National I.I. Mechnikov University, Ukraine, in July 2010, and the Ph.D. degree in electrical engineering from RWTH Aachen University, Germany, in September 2018.

She joined MediaTek Inc., San Jose, USA, in June 2018, and is currently a Staff Engineer with the Multimedia Technology Development Division. Her current research interests include video coding algorithms and related AI technologies.