Block Partitioning Structure in the HEVC Standard

Il-Koo Kim, Junghye Min, Tammy Lee, Woo-Jin Han, and JeongHoon Park

***Abstract*—High Efficiency Video Coding (HEVC) is the latest joint standardization effort of ITU-T WP 3/16 and ISO/IEC JTC 1/SC 29/WG 11. The resultant standard will be published as twin text by ITU-T and ISO/IEC; in the latter case, it will also be known as MPEG-H Part 2. This paper describes the block partitioning structure of the draft HEVC standard and presents the results of an analysis of coding efficiency and complexity. Of the many new technical aspects of HEVC, the block partitioning structure has been identified as representing one of the most significant changes relative to previous video coding standards. In contrast to the fixed size 16 × 16 macroblock structure of H.264/AVC, HEVC defines three different units according to their functionalities. The coding unit defines a region sharing the same prediction mode, e.g., intra and inter, and it is represented by the leaf node of a quadtree structure. The prediction unit defines a region sharing the same prediction information. The transform unit, specified by another quadtree, defines a region sharing the same transformation. This paper introduces technical details of the block partitioning structure of HEVC with an emphasis on the method of designing a consistent framework by combining the three different units together. Experimental results are provided to justify the role of each component of the block partitioning structure and a comparison with the H.264/AVC design is performed.**

***Index Terms*—Advanced video coding (AVC), H.264, High Efficiency Video Coding (HEVC), Joint Collaborative Team on Video Coding (JCT-VC), standards, video.**

I. Introduction

DUE TO THE ever-increasing demand for bit rate to support higher resolution video, there is a requirement to develop video compression technologies which would provide significantly higher coding efficiency than the current generation of video coding standards. The Joint Collaborative Team on Video Coding (JCT-VC), a joint activity of ITU-T WP 3/16 and ISO/IEC JTC 1/SC 29/WG 11, was set up in April 2010 to address these requirements for a next generation video coding standard. The resultant standard, called High Efficiency Video Coding (HEVC), is expected to be completed in January 2013.

Over the past decades, video coding standards such as MPEG-1 Video [1], MPEG-2 Video [2], MPEG-4 Visual [3], and H.264/advanced video coding (AVC) [4] played an important role in enabling multimedia applications. The basic ingredients of these standards are block-based motion compensation and spatial transforms. The current state-of-the-art, H.264/AVC provides approximately double the coding efficiency of the earlier MPEG-2 standard, using more flexible macroblock and submacroblock partitioning and variable transforms sizes of 4×4 and 8×8. However, due to some restrictions in the design, e.g., the fixed size of macroblock, limited depth of block partitioning and limited adaptivity between inter and intra prediction schemes, the coding efficiency is still not sufficient to cope with the ever increasing demands for storage and transmission of video content.

To overcome these problems, several papers tried to investigate the effect of relaxing the restrictions and using more flexible block partitioning. One direction was to add larger size blocks on top of the existing block structure [6]–[10], including 16×16, 16×8, and 8×16 transforms. Another direction utilized the more general quadtree structure for block partitioning [11]–[17] in addition to enlarging the size of blocks. In these approaches, up to 128×128 block size was allowed and more flexible motion and transform block (TB) partitioning structures were utilized.

The emerging HEVC standard represents one of the more advanced versions of the second approach. In the main profile of HEVC, a slice is partitioned into multiple coding tree units (CTU) which are allowed to have sizes from 8×8 up to 64×64. For comparison, prior video coding standards typically support a maximum block size of 16×16. Inside the CTU, a quadtree structure is built to allow more flexibility for partitioning of the CTU while maintaining consistent design, even when the CTU size is larger than 16×16. Each leaf node of the coding tree is called a coding unit (CU); this specifies how the prediction should be done between spatial and temporal schemes. The CU can have multiple prediction units (PU) and transform units (TU); these define regions sharing the same prediction-related information and the same transformation, respectively. The shape of the PU is specified by the splitting type, as in H.264/AVC whereas that of TU is represented by another quadtree, called the transform tree.

This paper explains the issues with H.264/AVC motivating HEVC development in Section II. Technical details of the block partitioning structure of HEVC are presented in Section III. Section IV provides the experimental results and Section V concludes this paper.

II. H.264/AVC Block Partitioning Structure

The block partitioning structure of the H.264/AVC is designed to provide more flexibility compared with the prior standards such as MPEG-2 or MPEG-4 Visual. However,

Manuscript received April 16, 2012; revised July 18, 2012; accepted August 20, 2012. Date of publication October 5, 2012; date of current version January 8, 2013. This work was supported by the Gachon University Research Fund of 2012 under Grant GCU-2011-R257. This paper was recommended by Associate Editor J. Ridge. (*Corresponding author: W.-J. Han.*)

I.-K. Kim, J. Min, T. Lee, and J. H. Park are with Samsung Electronics, Suwon 442-742, Korea (e-mail: ilkoo.kim@samsung.com; jh643.min@samsung.com; tammy.lee@samsung.com; jeonghoon@samsung.com).

W.-J. Han is with Gachon University, Seongnam 461-701, Korea (e-mail: hurumi@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2012.2223011

TABLE I
RELATIVE AREA OF BLOCKS WHICH HAVE SIZE OF 16 × 16 AND 64 × 64 ACCORDING TO VARIOUS VIDEO RESOLUTIONS

Video resolution	Relative area of 16 × 16 block (%)	Relative area of 64 × 64 block (%)
CIF (352 × 288)	0.253	4.040
SD (720 × 480)	0.074	1.185
720p (1280 × 720)	0.028	0.444
1080p (1920 × 1080)	0.012	0.198
2K (2560 × 1440)	0.007	0.111
4K (3840 × 2160)	0.003	0.049

there still remain several issues that could be addressed to improve the coding efficiency further. This section summarizes those issues which have been tackled during the HEVC standardization.

A. Size of Macroblock

The basic building units of H.264/AVC are macroblocks, which consist of 16 × 16 of luma samples and two corresponding blocks of chroma samples. The size of 16 × 16 has been considered to give a reasonable tradeoff between memory requirements and coding efficiency since the development of MPEG-1. More recently, it was found that the use of larger block sizes could increase coding efficiency significantly for high-resolution video, since the size of 16×16 is not sufficient to capture the increased spatial correlation coming from the higher resolution content [6]–[9]. Table I shows the relative area of blocks which have sizes of 16 × 16 and 64 × 64 according to several video resolutions. As shown in the table, the relative area of a macroblock in 4K resolution is decreased to about 1/82 of that of a macroblock in CIF resolution.

B. Limited Depth of Block Partitioning

In inter prediction of H.264/AVC, each macroblock can be processed in the following two-stage hierarchical process. A macroblock can be predicted either as one 16 × 16 partition, two 16 × 8 partitions, two 8 × 16 partitions, or four 8 × 8 partitions. If the 8 × 8 partition mode is selected, each of four 8 × 8 submacroblocks in the macroblock can be predicted either as one 8 × 8 partition, two 8 × 4 partitions, two 4 × 8 partitions, or four 4 × 4 partitions. Unlike inter prediction, only same sizes of partitions, which can be either 4×4, 8×8, or 16 × 16 partitions, are allowed in a macroblock for intra prediction. Fig. 1 shows the block partitioning structure for prediction and transform in H.264/AVC. Although there are several nonsquare partitions, the block partitioning structure of H.264/AVC can be roughly related to a three-level quadtree structure which supports nodes from 4 × 4 to 16 × 16.

In H.264/AVC, *mb_type* is a collective syntax element to specify whether the macroblock of size 16 × 16 should be split and how the prediction should be done. Additionally, *sub_mb_type* specifies whether submacroblock of size 8 × 8 should be split more. However, this kind of split and prediction combination would be inefficient in HEVC due to the large number of combinations, so a unified and consistent syntax is adopted across all sizes.

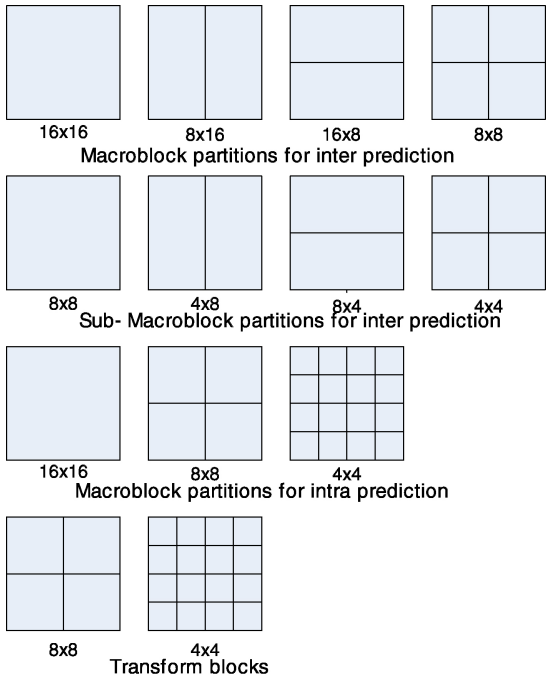


Fig. 1. Block partitioning structure for prediction and transform in H.264/AVC.

C. Adaptivity Between Inter and Intra Prediction Schemes

In H.264/AVC, it is possible to specify which spatial or temporal prediction scheme should be used for each macroblock. However, H.264/AVC does not allow adaptation between prediction schemes for partitions smaller than the 16 × 16 area.

D. Dependency Between Prediction and Transform

Two sizes of TB, 4 × 4 and 8 × 8, are supported in H.264/AVC as shown in Fig. 1. Two transform sizes can be adaptively used for each macroblock with an additional syntax element *transform_size_8x8_flag*; however, it is highly dependent on how the block is split.

When the inter prediction scheme is used, 4 × 4 transform should be used for all blocks in the macroblock if at least one block is smaller than 8×8. Even in the other case, all transform sizes should be the same within one macroblock. When the intra prediction scheme is used, the size of the transform should be equal to the block size except in Intra_16×16 mode. In Intra_16 × 16 mode, a Hadamard transform of 4 × 4 size is applied to DC values after applying sixteen 4 × 4 transforms to the 16 × 16 area. This strong dependency between prediction mode and transform size and the dependency on block size make the overall design more complex to implement, especially if were to be applied to a design such as HEVC that allows more variety of block sizes.

III. BLOCK PARTITIONING STRUCTURE IN HEVC

The draft HEVC standard has adopted a highly flexible and efficient block partitioning structure by introducing four different block concepts: CTU, CU, PU, and TU, which are defined to have clearly separated roles. The terms coding

tree block (CTB), coding block (CB), prediction block (PB), and TB are also defined to specify the 2-D sample array of one color component associated with the CTU, CU, PU, and TU, respectively. Thus, a CTU consists of one luma CTB, two chroma CTBs, and associated syntax elements. A similar relationship is valid for CU, PU, and TU.

Although use of a quadtree structure in video compression is not a new concept [19]–[22], the coding tree approach in HEVC can bring additional coding efficiency benefits by incorporating PU and TU quadtree concepts for video compression.

Leaf nodes of a tree can be merged or combined [22] in a general quadtree structured video coding scheme. After the final quadtree is formed, motion information is transmitted at the leaf nodes of the tree. L-shaped or rectangular-shaped motion partition is possible through merging and combination of nodes. However, in order to make such shapes, the merge process should be followed using smaller blocks after further splitting is occurred. In the HEVC block partitioning structure, such cases are taken care of by the PU [15]. Instead of splitting one depth more for merging and combination, predefined partition modes such as PART_$2N\times2N$, PART_$2N\times N$, and PART_$N\times2N$ are tested and the optimal partition mode is selected at the leaf nodes of the tree. It is worthwhile mentioning that PUs still can share motion information through merging mode in HEVC. Although a general quadtree structure without PU concept was investigated by removing the symmetric rectangular partition modes (PART_$2N\times N$ and PART_$N\times2N$) from the syntax and replaced by corresponding merge flags [23], both coding efficiency and complexity was proved inferior to the current design.

Another difference is the transform tree. Even though variable block size transforms were used for quadtree structured motion compensation, their usage was rather restricted. For example, transform size was strictly combined with motion compensation block size. Even though multiple transform size could be utilized, it was usual to use same size transform in a motion compensated block. In HEVC, the motion compensated residual can be transformed with a quadtree structure, and the actual transform is performed at leaf nodes. Since the transform tree is rooted from the leaf nodes of coding tree, this creates a nested quadtree. This kind of nested quadtree exists since the transform tree is started from the CU regardless of partition modes, i.e., PU shapes [16]. This is a way to construct a nested quadtree even though we have PU concepts that differ from a general quadtree structure.

Another noticeable aspect is the full utilization of depth information for entropy coding. For example, entropy coding of HEVC is highly reliant on the depth information of quadtree. For syntax elements such as *inter_pred_idc*, *split_transform_flag*, *cbf_luma*, *cbf_cb* and *cbf_cr*, depth dependent context derivation is heavily used for coding efficiency. It has been demonstrated that this can break the dependency with neighboring blocks with less line buffer requirement in hardware implementations because information of above CTU does not need to be stored.

In the following sections, the block partitioning structures in the HEVC standard are presented in conjunction with a detailed explanation of those unit definitions.

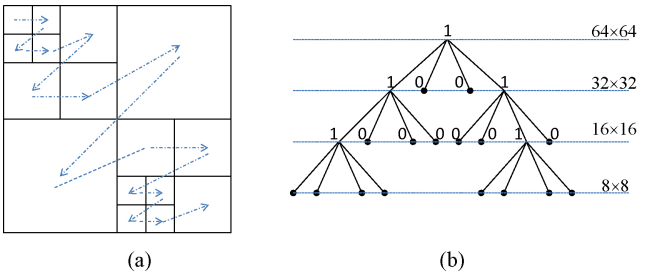


Fig. 2. Example of CTU partitioning and processing order when size of CTU is equal to 64 × 64 and minimum CU size is equal to 8 × 8. (a) CTU partitioning. (b) Corresponding coding tree structure.

A. Coding Tree Unit

A slice contains an integer multiple of CTU, which is an analogous term to the macroblock in H.264/AVC. Inside a slice, a raster scan method is used for processing the CTU.

In main profile, the minimum and the maximum sizes of CTU are specified by the syntax elements in the sequence parameter set (SPS) among the sizes of 8×8, 16×16, 32×32, and 64×64. Due to this flexibility of the CTU, HEVC provides a way to adapt according to various application needs such as encoder/decoder pipeline delay constraints or on-chip memory requirements in a hardware design. In addition, the support of large sizes up to 64×64 allows the coding structure to match the characteristics of the high definition video content better than previous standards; this was one of the main sources of the coding efficiency improvements seen with HEVC.

B. Coding Unit

The CTU is further partitioned into multiple CU to adapt to various local characteristics. A quadtree denoted as the coding tree is used to partition the CTU into multiple CUs.

1) *Recursive Partitioning from CTU:* Let CTU size be $2N\times2N$ where N is one of the values of 32, 16, or 8. The CTU can be a single CU or can be split into four smaller units of equal sizes of $N\times N$, which are nodes of coding tree. If the units are leaf nodes of coding tree, the units become CUs. Otherwise, it can be split again into four smaller units when the split size is equal or larger than the minimum CU size specified in the SPS. This representation results in a recursive structure specified by a coding tree.

Fig. 2 illustrates an example of CTU partitioning and the processing order of CUs when the size of CTU is equal to 64 × 64 and the minimum CU size is equal to 8 × 8. Each square block in Fig. 2(a) represents CU. In this example, a CTU is split into 16 CUs which have different sizes and positions. Fig. 2(b) shows corresponding coding tree structure representing the structure of the CTU partitioning in Fig. 2(a). Numbers on the tree represent whether the CU is further split. In Fig. 2(a), CUs are processed by following the dotted line. This processing order of CUs can be interpreted as a depth-first traversing in the coding tree structure [24]. If CTU size of 16 × 16 and the minimum CU size of 8 × 8 are used, the resultant structure is roughly similar to that of H.264/AVC.

HEVC utilizes CU as a unit to specify which prediction scheme is used for intra and inter predictions. Since the

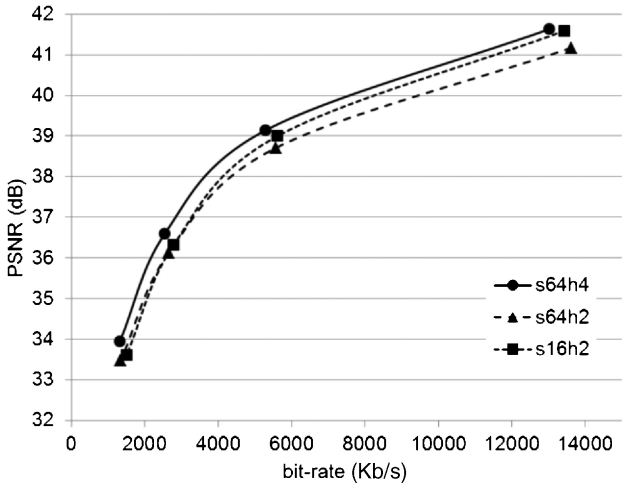


Fig. 3. Rate-distortion curves of several combinations of size of CTU and maximum coding tree depth for *Traffic* sequences (2560 × 1600). The size of CTU is represented by character "s" and maximum coding tree depth is represented by character "h." Each curve shows the result when s64h4, s16h2, and s64h2 are used, respectively.

TABLE II
SIMPLIFIED FORM OF CODING TREE SYNTAX TABLE

coding_tree(x0, y0, log2CbSize, cbDepth) {
split_coding_unit_flag[x0][y0]
if(split_coding_unit_flag[x0][y0]) {
coding_tree(x0, y0, log2CbSize − 1, cbDepth + 1)
coding_tree(x1, y0, log2CbSize − 1, cuDepth + 1)
coding_tree(x0, y1, log2CbSize − 1, cbDepth + 1)
coding_tree(x1, y1, log2CbSize − 1, cbDepth + 1)
} else {
coding_unit(x0, y0, log2CbSize)
}
}

minimum CU size can be 8 × 8, the minimum granularity for switching different prediction schemes is 8 × 8, which is smaller than the macroblock size of H.264/AVC.

2) *Benefits of Flexible CU Partitioning Structure:* This kind of flexible and recursive representation provides several major benefits. The first benefit comes from the support of CU sizes greater than the conventional 16 × 16 size. When the region is homogeneous, a large CU can represent the region by using a smaller number of symbols than is the case using several small blocks.

Fig. 3 shows rate-distortion curves of several combinations of size of CTU and maximum coding tree depth for Traffic 2560×1600@30 Hz sequence which is specified in [31]. The results are obtained using HM-6.0 Main profile using low delay constraint of the common test condition of HEVC. The size of CTU is represented by character "s" and maximum coding tree depth is represented by character "h" in the figure. Each curve shows the result when s64h4, s16h2, and s64h2 are used, respectively. There is a big gap of coding efficiency about 13.7% in Bjøntegaard delta bitrate [29], [30] between s64h4 and s16h2. This result illustrates that adding large size CU is an effective means to increase coding efficiency for higher resolution content.

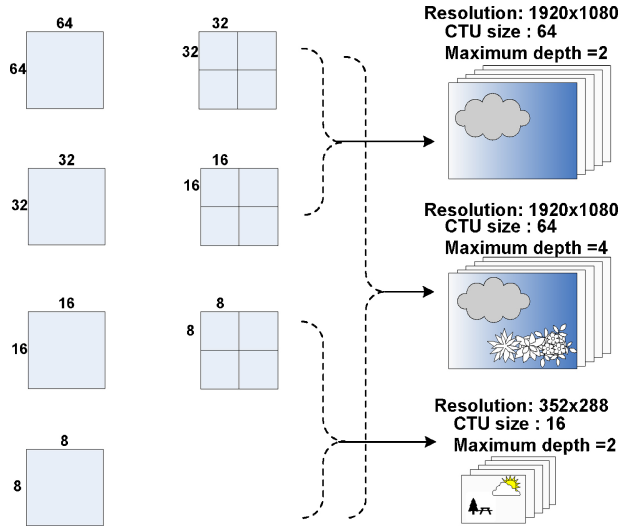


Fig. 4. Example of CTU size and various CU sizes for various resolutions. The figure is taken from JCTVC-A124 [13].

Coding efficiency difference between s64h4 and s64h2 is about 19.5% and it is also noticeable that coding efficiency difference between s64h2 and s16h2 is similar at low bit rate, but s16h2 shows better coding efficiency at high bit rate because smaller size blocks cannot be utilized for s64h2, where minimum CU size is 32 × 32. These results can be interpreted as showing that large size CU is important to increase coding efficiency in general but still small size CU should be used together to cover regions which large CU cannot be applied to successfully.

Furthermore, supporting arbitrary sizes of CTU enables the codec to be readily optimized for various content, applications, and devices. Compared to the use of fixed size macroblock, support of various sizes of CTU is one of the strong points of HEVC in terms of coding efficiency and adaptability for contents and applications. This ability is especially useful for low-resolution video services, which are still commonly used in the market. By choosing an appropriate size of CTU and maximum hierarchical depth, the hierarchical block partitioning structure can be optimized to the target application. Fig. 4 shows examples of various CTU sizes and CU sizes suitable for different resolutions and types of content. For example, for an application using 1080p content that is known to include only simple global motion activities, a CTU size of 64 and depth of 2 may be an appropriate choice. For more general 1080p content, which may also include complex motion activities of small regions, a CTU size of 64 and maximum depth of 4 would be preferable.

Finally, by eliminating the distinction between macroblock and submacroblock and using only CU, the multilevel hierarchical quadtree structure can be specified in a very simple and elegant way. Together with the size-independent syntax representation, syntax items of one general size may be specified for the remaining coding tools.

Table II shows the recursive part of the coding tree syntax in simplified form [5]. For a detailed explanation about the notations used in the syntax table, please refer to [5]. As shown

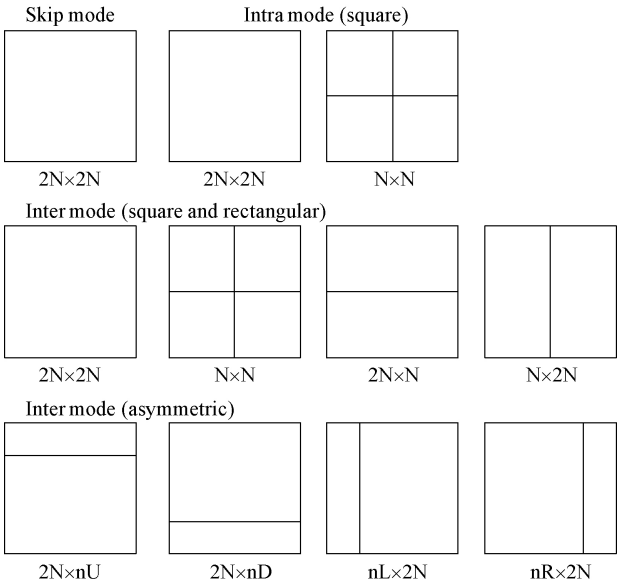


Fig. 5. Illustration of PU splitting types in HEVC.

in the table, the splitting process of coding tree can be specified recursively and all other syntax elements can be represented in the same way regardless of the size of CU. This kind of recursive representation is very useful in terms of reducing parsing complexity and improving clarity when the quadtree depth is large.

C. Prediction Unit

One or more PUs are specified for each CU, which is a leaf node of coding tree. Coupled with the CU, the PU works as a basic representative block for sharing the prediction information. Inside one PU, the same prediction process is applied and the relevant information is transmitted to the decoder on a PU basis.

A CU can be split into one, two or four PUs according to the PU splitting type. HEVC defines two splitting shapes for the intra coded CU and eight splitting shapes for inter coded CU. Unlike the CU, the PU may only be split once.

1) *PU Splitting Type:* Similar to prior standards, each CU in HEVC can be classified into three categories: skipped CU, inter coded CU, and intra coded CU. An inter coded CU uses motion compensation scheme for the prediction of the current block, while an intra coded CU uses neighboring reconstructed samples for the prediction. A skipped CU is a special form of inter coded CU where both the motion vector difference and the residual energy are equal to zero.

For each category, PU splitting type is specified differently as shown in Fig. 5 when the CU size is equal to $2N\times2N$. As shown in the figure, only PART_$2N\times2N$ PU splitting type is allowed for the skipped CU. For the intra coded CU, two possible PU splitting types of PART_$2N\times2N$ and PART_$N\times N$ are supported. Finally, total eight PU splitting types are defined as two square shapes (PART_$2N\times2N$, PART_$N\times N$), two rectangular shapes (PART_$2N\times N$ and PART_$N\times2N$), and four asymmetric shapes (PART_$2N\times nU$, PART_$2N\times nD$, PART_$nL\times2N$, and PART_$nR\times2N$) for inter coded CU. Although more sophisticated partitioning was considered as in [18], but current PU splitting types were chosen as a good tradeoff between encoding complexity and coding efficiency.

Note that all information related to the prediction scheme is specified on a PU basis. For instance, the most probable mode index and intra prediction mode for intra coded CU or merge flag, merge index, inter prediction flag, motion vector prediction index, reference index, and motion vector difference for inter coded CU are unique per PU.

For most cases, PU partitioning of chroma block shares the same splitting of luma component; however, when the CU size is equal to 8×8 and PART_$N\times N$ is used for the PU splitting type, PART_$2N\times2N$ is used for the chroma block to prevent the block size from being less than 4×4.

2) *Constraints According to CU Size:* In PART_$N\times N$, CU is split into four equal-sizes PUs, which is conceptually similar with the case of four equal-size CUs when the CU size is not equal to the minimum CU size. Thus, HEVC disallows the use of PART_$N\times N$ except when the CU size is equal to the minimum CU size. It was observed that this design choice can reduce the encoding complexity significantly while the coding efficiency loss is marginal [25].

To reduce the worst-case complexity, HEVC further restricts the use of PART_$N\times N$ and asymmetric shapes. In case of inter coded CU, the use of PART_$N\times N$ is disabled when the CU size is equal to 8×8. Moreover, asymmetric shapes for inter coded CU are only allowed when the CU size is not equal to the minimum CU size.

D. Transform Unit

Similar with the PU, one or more TUs are specified for the CU. HEVC allows a residual block to be split into multiple units recursively to form another quadtree which is analogous to the coding tree for the CU [14], [16], [26]. The TU is a basic representative block having residual or transform coefficients for applying the integer transform and quantization. For each TU, one integer transform having the same size to the TU is applied to obtain residual coefficients. These coefficients are transmitted to the decoder after quantization on a TU basis.

1) *Residual Quadtree:* After obtaining the residual block by prediction process based on PU splitting type, it is split into multiple TUs according to a quadtree structure. For each TU, an integer transform is applied. The tree is called transform tree or residual quadtree (RQT) since the residual block is partitioned by a quadtree structure and a transform is applied for each leaf node of the quadtree.

Similar to the coding tree, which is represented by a series of *split_coding_unit_flag*, RQT is also structured by successive signalling of the syntax element *split_transform_flag* in a recursive manner. RQT can be classified into two cases having square shape and nonsquare shape, and they are denoted as square residual quadtree (SRQT) and nonsquare residual quadtree (NSRQT), respectively. The NSRQT was adopted temporarily, but excluded in the final draft text specification [5]. In this paper, NSRQT is investigated for better understating of current HEVC block structure design. Table III shows a syntax table for the recursive structure of RQT [5].

TABLE III
SIMPLIFIED FORM OF TRANSFORM TREE SYNTAX TABLE

```
transform_tree( trafoDepth, blkIdx ) {
    no_residual_data_flag
    if( !no_residuual_data_flag ) {
        split_transform_flag[ x0 ][ y0 ][ trafoDepth ]
        if( split_transform_flag[ x0 ][ y0 ][ trafoDepth ] ) {
            transform _tree( trafoDepth+1, 0 )
            transform _tree( trafoDepth+1, 1 )
            transform _tree( trafoDepth+1, 2)
            transform _tree( trafoDepth+1, 3)
        } else {
            transform _unit( trafoDepth )
        }
    }
}
```

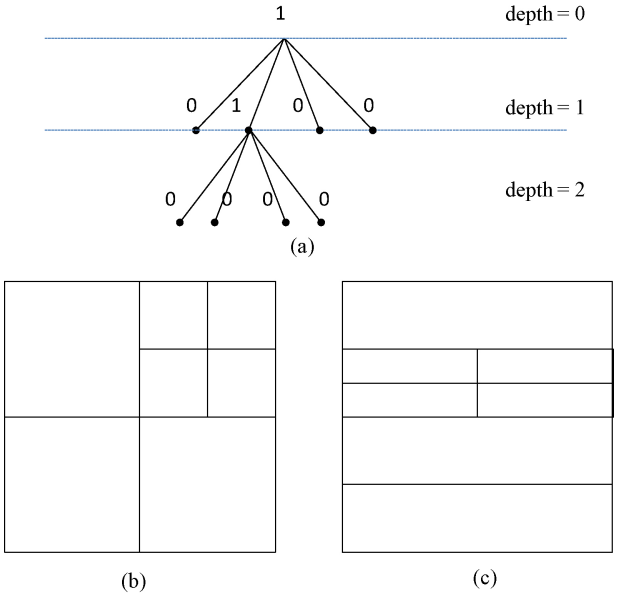


Fig. 6. Examples of transform tree and block partitioning. (a) Transform tree. (b) TU splitting for square-shaped PU. (c) TU splitting for rectangular or asymmetric shaped PU.

2) *Nonsquare Partitioning:* SRQT is constructed when PU splitting type is square shape while NSRQT is utilized for rectangular and asymmetric shapes. For NSRQT, transform shape is horizontal when the choice of the partition mode is horizontal type such as PART_$2N\times N$, PART_$2N\times nU$, and PART_$2N\times nD$. The same rule is applied to the vertical type case such as PART_$N\times 2N$, PART_$nL\times 2N$, and PART_$nR\times 2N$. Although the syntax of SRQT and NSRQT is the same, as depicted in Table III, the shapes of TUs at each transform tree depth are defined differently for SRQT and NSRQT. Fig. 6 illustrates an example of transform tree and corresponding TU splitting. Fig. 6(a) represents transform tree. Fig. 6(b) shows TU splitting when the PU shape is square. Fig. 6(c) shows TU splitting when the PU shape is rectangular or asymmetric. Although they share the same transform tree, the actual TU splitting is different depending on the PU splitting type.

3) *Transform Across Boundary:* In HEVC, both the PU size and the TU size can reach the same size of the corresponding CU. This leads to the fact that the size of TU may be larger than that of the PU in the same CU, i.e., residuals from different PUs in the same CU can be transformed together. For example, when the TU size is equal to the CU size, the transform is applied to the residual block covering the whole CU regardless of the PU splitting type. Note that this case exists only for inter coded CU, since the prediction is always coupled with the TU splitting for intra coded CU.

4) *Maximum Depth of Transform Tree:* The maximum depth of transform tree is closely related to the encoding complexity. To provide the flexibility on this feature, HEVC specifies two syntax elements in the SPS which control the maximum depth of transform tree for intra coded CU and inter coded CU, respectively.

The case when the maximum depth of transform tree is equal to 1 is denoted as implicit TU splitting since there is no need to transmit any information on whether the TU is split. In this case, the transform size is automatically adjusted to be fit inside the PU rather than allowing transform across the boundary. The coding efficiency loss of implicit TU partitioning is about from 0.7% to 1% compared to the cases RQT depth is equal to 2 [27], [28]. More quantitative results about the effectiveness of this approach will be provided in Section IV-D.

IV. EXPERIMENTAL RESULTS

A. Test Condition

To investigate the performance of various test cases following test conditions were used. Unless additional conditions are given, all tests were performed by setting different configurations in HM 6.0 reference software with the JCT-VC common test conditions [31]. Test conditions are summarized as follows.

1) HEVC main profile was used.
2) Four quantization parameters were used: 22, 27, 32, and 37.
3) All intra, random access, low-delay B configurations were used.
4) Bjøntegaard delta bitrate is computed using piece-wise cubic interpolation [29], [30].
5) Encoding and decoding runtime is measured on a Linux-based clustering system having CPUs with same capability.
6) A total of 20 video sequences in five classes were evaluated. These sequences cover a wide range of resolutions and image patterns. Video resolutions are 2560×1600, 1920×1080, 832×480, 416×240, and 1280×720, designated as class A, B, C, D and E, respectively. Class E contains video conferencing sequences.
7) CTU size, minimum CU size, maximum TU size, minimum TU size and both maximum depths of transform trees of intra coded CU and inter coded CU are set equal to 64×64, 8×8, 32×32, 4×4, 3 and 3, respectively.

TABLE IV
CODING EFFICIENCY LOSS ACCORDING TO DECREASED CTU SIZES

Class	All intra		Random access		Low delay B	
	32 × 32	16 × 16	32 × 32	16 × 16	32 × 32	16 × 16
Class A	0.7%	4.8%	4.1%	21.9%	–	–
Class B	0.8%	4.0%	2.9%	15.5%	3.4%	17.3%
Class C	0.5%	1.8%	1.0%	6.2%	1.5%	7.8%
Class D	0.3%	1.0%	0.5%	3.5%	0.9%	4.2%
Class E	1.2%	5.6%	–	–	6.8%	31.8%
Average	0.7%	3.4%	2.2%	12.0%	2.9%	14.4%
Enc. time	86%	70%	79%	56%	78%	57%
Dec. time	104%	114%	108%	126%	105%	122%

B. Experimental Results Related to CTU and CU

1) *Results of Various CTU Sizes:* Performances with various CTU sizes of 64 × 64, 32 × 32, and 16 × 16 were evaluated as in Table IV. The results of HM 6.0 using the common test condition have been used as a reference. The minimum CU size was kept unchanged to 8 × 8. When the CTU size is equal to 32 × 32, the average coding efficiency loss is about 0.7% in intraonly case whereas the loss is about 2.2% and 2.9% in random access and low-delay B cases. If CTU size is reduced to 16 × 16, equivalent to that in previous standards, the average coding efficiency loss is increased to 3.4% in intra-only case and 12.0% and 14.4% in random access and low-delay B cases. From these results, it can be concluded that the CTU size of 32 × 32 leads to significant coding efficiency improvements while the CTU size of 64 × 64 provides a marginal further coding efficiency improvement. Additional observations from these results are as follows.

1) The benefits from the use of larger CTU size become significant when the high-resolution video sequences such as Class A (2560 × 1600), Class B (1920 × 1080), and Class E (1280 × 720) are used.
2) The benefits from the use of large size CTU become significant for random access and low-delay B cases where the inter prediction scheme is applied rather than intra-only case.
3) Video conferencing sequences in Class E have significant coding efficiency loss when CTU sizes are reduced. It indicates that the large size CTU is especially useful for these kinds of sequences which have a lot of homogeneous regions with small motion activities.
4) When smaller CTU sizes are used, encoding time is significantly reduced, but the decoding time is increased. In particular, when a CTU size of 16 × 16 is used, the decoding time is increased by up to 26%. Although definitive conclusions should not be drawn from results using nonfully optimized reference software, it appears that the use of larger CTU sizes does not increase the decoding time, presumably because the number of bits to be parsed is reduced due to the higher coding efficiency. In addition, the lower memory bandwidth than for smaller blocks may contribute to a reduction of decoding time.

Fig. 7 shows decoded images of the CTU sizes of 64 × 64 and 16 × 16 for test sequence *Kimono* of Class B. The image of the CTU size 16 × 16 was captured from video encoded at 817 kbit/s resulting in a PSNR of 35.5 dB. A CTU size of 64 × 64 results in a bit-rate of 814 kbit/s and a PSNR of 36.5 dB. This demonstrates a gain in PSNR of 1.0 dB, which also corresponds to improved subjective visual quality at the same bitrate. Visual differences were observed around the eyes, nose, and coat string.

Fig. 7. Coding results for *Kimono*, left: 16 × 16, right: 64 × 64 of CTU size.

TABLE V
CODING EFFICIENCY LOSS ACCORDING TO INCREASED MINIMUM CU SIZES

Class	All intra		Random access		Low delay B	
	16 × 16	32 × 32	16 × 16	32 × 32	16 × 16	32 × 32
Class A	3.6%	9.3%	4.3%	15.9%	–	–
Class B	5.0%	15.1%	5.7%	19.1%	5.1%	18.8%
Class C	10.1%	26.0%	10.4%	36.0%	9.2%	34.9%
Class D	11.5%	29.0%	12.4%	44.2%	11.0%	44.7%
Class E	9.5%	26.6%	–	–	5.8%	21.8%
Average	7.7%	20.6%	8.0%	28.2%	7.7%	29.9%
Enc. time	59%	36%	68%	41%	69%	41%
Dec. time	83%	73%	90%	84%	88%	82%

2) *Results of Various Minimum CU Sizes:* Performance of various minimum CU sizes of 16 × 16 and 32 × 32 was investigated as in Table V while keeping a CTU size of 64 × 64. When the minimum CU size is increased from 8 × 8 to 16 × 16, the average coding losses are 7.7%, 8.0% and 7.7% for intra-only, random access and low-delay B cases. Furthermore, the use of 32 × 32 for the minimum CU size results in significant coding efficiency loss from 20.0% to 29.9% in the configurations tested. It can be observed that the coding efficiency loss from the use of increased minimum CU size becomes significant for lower resolution video sequences.

3) *Results of Optimized Block Partitioning Parameters:* Although the use of 64 × 64 CTU size and 8 × 8 minimum CU size appear to provide the best coding efficiency, there are some examples which justify the decision of allowing flexible specification of those two parameters. Table VI shows some examples when the CTU size and the minimum CU size are adjusted to obtain better tradeoffs between the coding efficiency and the computational complexity.

When the CTU size is reduced from 64 × 64 to 32 × 32 in *PeopleOnStreet* sequence, the coding efficiency loss is

TABLE VI
PERFORMANCE OF SEVERAL SEQUENCES BY OPTIMIZING BLOCK PARTITIONING PARAMETERS

Sequence	CTU size	Minimum CU size	BD rate (%)	Enc. time (%)
PeopleOnStreet (2560 × 1600)	32 × 32	8 × 8	0.5	73
Nebuta (2560 × 1600)	64 × 64	16 × 16	0.9	60
RaceHorses (832 × 480)	32 × 32	8 × 8	0.6	79

TABLE VII
CODING EFFICIENCY LOSS WHEN THE ADAPTION BETWEEN PREDICTION SCHEMES IS NOT ALLOWED FOR 8 × 8 CU

Class	Random access	Low-delay B
Class A	0.7%	–
Class B	3.4%	3.5%
Class C	1.7%	2.0%
Class D	1.6%	1.5%
Class E	–	0.5%
Average	1.9%	2.1%

TABLE VIII
CODING EFFICIENCY IMPROVEMENT OF RECTANGULAR PU SHAPE AND ASYMMETRIC PU SHAPE

Class	Random access		Low delay B	
	Rectangular	Rectangular + asymmetric	Rectangular	Rectangular + asymmetric
Class A	−2.0%	−2.5%	–	–
Class B	−2.0%	−2.6%	−2.1%	−2.9%
Class C	−3.3%	−4.2%	−3.6%	−4.5%
Class D	−4.1%	−4.9%	−4.8%	−5.7%
Class E	–	–	−3.2%	−4.5%
Average	−2.8%	−3.5%	−3.2%	−4.3%

only 0.5% while the encoding time is reduced by 27%. Similarly, the encoding times for *Nebuta* and *RaceHorses* sequences can be reduced by 40% and 21%, respectively, with less than 1.0% coding efficiency loss by adjusting the block partitioning parameters suitably. These results imply that the smart HEVC encoder may find better tradeoffs between the coding efficiency and computational complexity by controlling the block partitioning parameters according to the resolution and the characteristics of input video sequences. Although this property can be obtained by controlling the encoder to bypass mode estimation at certain sizes while keeping the CTU size of 64 × 64, the processing latency between CTUs and the required line buffer memory cannot be reduced by this way.

4) *Results of Inter and Intra Prediction Adaptivity:* Table VII shows the coding efficiency loss when the adaptation between inter and intra prediction schemes is only allowed for the CU sizes of 16 × 16 or above. That means minimum CU size for skipping is 16 × 16. Note that this would correspond to the case of H.264/AVC, which allows the inter and intra prediction adaptivity at macroblock level (size of 16 × 16). As shown in the table, average coding efficiency losses from disabling the inter and intra prediction adaptivity at 8 × 8 CU are about 1.9% and 2.1% for random access and low-delay B cases, respectively. Note that more than 3% coding efficiency loss is observed for Class B sequences while the loss becomes smaller for other classes.

TABLE IX
CODING EFFICIENCY LOSS WHEN MAXIMUM TU SIZE IS REDUCED

Class	All intra		Random access		Low delay B	
	16 × 16	8 × 8	16 × 16	8 × 8	16 × 16	8 × 8
Class A	2.7%	9.7%	4.1%	12.1%	–	–
Class B	1.7%	5.9%	2.4%	8.1%	2.8%	8.9%
Class C	0.2%	1.4%	0.5%	3.0%	1.0%	4.0%
Class D	0.1%	0.8%	−0.1%	0.5%	0.0%	1.1%
Class E	1.8%	6.9%	–	–	1.6%	7.2%
Average	1.3%	4.9%	1.7%	6.1%	1.4%	5.4%

TABLE X
CODING EFFICIENCY LOSS WHEN MAXIMUM TRANSFORM TREE DEPTH IS DECREASED

Class	All intra		Random access		Low delay B	
	2	1	2	1	2	1
Class A	0.1%	0.3%	0.3%	0.8%	–	–
Class B	0.1%	0.4%	0.3%	1.0%	0.3%	1.1%
Class C	0.1%	0.4%	0.2%	0.9%	0.3%	1.1%
Class D	0.1%	0.3%	0.1%	0.8%	0.2%	1.1%
Class E	0.1%	0.5%	–	–	0.2%	0.6%
Average	0.1%	0.4%	0.2%	0.9%	0.2%	1.0%

C. Experimental Results on PU

PU shapes can be classified into three categories: square, rectangular, and asymmetric. To verify the benefits of each category, rectangular PU shapes and asymmetric PU shapes were tested against the modified HM 6.0 without both PU shapes. For mode selection for PU shapes in encoder, we followed the behavior implemented in HM 6.0. Table VIII summarizes the experimental results.

As shown in table, the average coding efficiency improvements from rectangular PU shapes are about 2.8% and 3.2% for random access and low-delay B, respectively. By using the asymmetric PU shapes together, the average coding efficiency improvements reach to 3.5% and 4.3% for random access and low-delay B cases.

D. Experimental Results on TU

1) *Results of Maximum TU Size:* Maximum TU size can be adjusted by the syntax element specified in the SPS. When the maximum TU size is smaller than the CU size, the transform tree is automatically split until the tree node is fit to the maximum TU size. Table IX summarizes the average coding efficiency loss when the maximum TU sizes are adjusted to 16 × 16 and 8 × 8 rather than the default size of 32 × 32. The results show that restricting the maximum TU size results in a significant coding efficiency loss for the higher resolution video sequences.

E. Results of Maximum Transform Tree Depth

Maximum transform tree depth is another controllable parameter to be specified in the SPS. The main purpose of using different maximum transform tree depth is to provide more freedom to the encoder in terms of encoding complexity. Table X shows coding efficiency loss when the maximum transform tree depth is reduced from 3 to 2 and from 3 to 1. It can be seen that the coding efficiency loss due reducing

TABLE XI
CODING EFFICIENCY IMPROVEMENT OF NSRQT AND NSRQT WITH ASYMMETRIC PU SHAPE

Class	Random access		Low delay B	
	NSRQT	NSRQT + Asymmetric PU	NSRQT	NSRQT + Asymmetric PU
Class A	−0.1%	−0.8%	–	–
Class B	−0.4%	−1.2%	−0.9%	−1.9%
Class C	−0.3%	−1.4%	−0.7%	−1.9%
Class D	−0.3%	−1.2%	−0.6%	−1.9%
Class E	–	–	−1.2%	−2.8%
Average	−0.3%	−1.2%	−0.8%	−2.1%

the maximum transform tree depth is generally less severe than that due to reducing the maximum TU size.

The coding efficiency impact of transform over prediction boundaries can be estimated by disabling the TU when the TU has the size equal to CU size and PU shape is not PART_$2N \times 2N$. *split_transform_flag* is skipped for this case. Base on this experiment, the average coding efficiency improvement due to transform over prediction boundaries is about 0.3% and 0.4% for class B for random access and low-delay B cases, respectively.

1) *Results of NSRQT:* Coding efficiency improvement from NSRQT is summarized in Table XI. Since NSRQT is closely related to the asymmetric PU shape, two experimental results for NSRQT on top of HM 6.0 with and without asymmetric PU shape were conducted. As shown in the table, the average coding efficiency improvements from NSRQT are about 0.3% and 0.8% for random access and low-delay B cases, respectively. When NSRQT is used with asymmetric PU shapes, the average coding efficiency improvement is increased to 1.2% and 2.1%. By comparing the results of Table VIII, it can be seen that the benefits from NSRQT and asymmetric PU shape are not overlapped and are even synergistic.

It should be noted that the flexibility of HEVC block partitioning design allows us to simulate the block structure of H.264/AVC very closely. For example, CTU size of 16×16, 8×8 maximum TU size restriction, no inter/intra adaptivity at 8×8 CU level, no NSRQT and no asymmetric motion partition can be used for simulating the block structure specified in the H.264/AVC. Among them, disabling of inter/intra adaptivity at the 8×8 CU level cannot be achieved unless the relevant HEVC specification is changed.

V. CONCLUSION

This paper described the details of the block partitioning scheme of the draft HEVC standard, which is specified using the newly introduced CTU, CU, PU, and TU concepts. Along with the technical details, extensive experimental results were provided to highlight various aspects of the block partitioning scheme of HEVC.

Experimental results reveal that the proposed flexible block partitioning structure plays a major role in the substantial performance gains exhibited by HEVC relative to previous video coding standards. In addition to coding efficiency, the hierarchical structure provides elegant and efficient ways to optimize design to the expected resolutions and image patterns of specific applications. These flexible features provide essential functionalities to support the increasing demand for improved video quality across a wide range of applications.

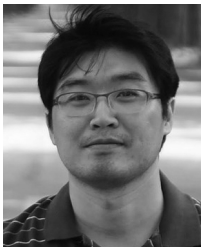
ACKNOWLEDGMENT

The authors would like to thank the experts of ITU-T VCEG, ISO/IEC MPEG, and the ITU-T/ISO/IEC Joint Collaborative Team on Video Coding as well as their colleagues including, but not limited to, H. K. Jung, M. S. Cheon, J. C. Lee, S. R. Lee, D. Y. Kim, and K. H. Lee for their contributions.

REFERENCES

[1] ISO/IEC, *Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbit/s: Video*, document ISO/IEC 11172-2, Aug. 1993.
[2] ISO/IEC, *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video*, document ISO/IEC 13818-2, 1996.
[3] ISO/IEC, *Information Technology—Coding of Audio-Visual Objects Part 2: Visual*, document ISO/IEC 14496-2, 2004.
[4] *Advanced Video Coding for Generic Audiovisual Services, ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Version 8*, document ITU-T and ISO/IEC, Jul. 2007.
[5] B. Bross, W.-J. Han, G. J. Sullivan, J.-R. Ohm, and T. Wiegand, *High Efficiency Video Coding (HEVC) Text Specification Draft 8*, document JCT-VC J1003, Joint Collaborative Team on Video Coding (JCT-VC), Stockholm, Sweden, Jul. 2012.
[6] S. Ma and C.-C. J. Kuo, "High-definition video coding with super-macroblocks," in *Proc. Visual Commun. Image Process.*, vol. 6508. Jan. 2007, p. 650816.
[7] P. Chen, Y. Ye, and M. Karczewicz, *Video Coding Using Extended Block Sizes*, VCEG document AJ23, 36th VCEG Meeting, Oct. 2008.
[8] T. Yoshino, S. Naito, and S. Sakazawa, *Preliminary Response for Draft Call for Evidence on High Performance Video Coding*, MPEG document M16082, 87th MPEG Meeting, Feb. 2009.
[9] S. Sekiguchi and S. Yamagishi, *On Coding Efficiency With Extended Block Size for UHDTV*, MPEG document M16019, 87th MPEG Meeting, Feb. 2009.
[10] M. Karczewicz, P. Chen, R. L. Joshi, X. Wang, W.-J. Chien, R. Panchal, Y. Reznik, M. Coban, and I. S. Chong, "A hybrid video coder based on extended macroblock sizes, improved interpolation, and flexible motion representation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1698–1708, Dec. 2010.
[11] K. H. Lee, E. Alshina, J. H. Park, W.-J. Han, and J. H. Min, *Technical Considerations for Ad-Hoc Group on New Challenges in Video Coding Standardization*, MPEG document M15580, 85th MPEG Meeting, Jul. 2008.
[12] E. Alshina, K. H. Lee, W.-J. Han, and J. H. Park, *Technical Considerations for Ad-Hoc Group on New Challenges in Video Coding Standardization*, MPEG document M15899, 86th MPEG Meeting, Oct. 2008.
[13] K. McCann, W.-J. Han, I.-K. Kim, J. H. Min, E. Alshina, A. Alshin, T. Lee, J. Chen, V. Seregin, S. Lee, Y. M. Hong, M. S. Cheon, and N. Shlyakhov, *Samsung's Response to the Call for Proposals on Video Compression Technology*, JCT-VC document A124, 1st JCT-VC Meeting, Apr. 2010.
[14] M. Winken, S. Bosse, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, D. Marpe, S. Oudin, M. Preiss, H. Schwarz, M. Siekmann, K. Suhring, and T. Wiegand, *Video Coding Technology Proposal by Fraunhofer HHI*, JCT-VC document A116, 1st JCT-VC Meeting, Apr. 2010.
[15] W.-J. Han, J. H. Min, I.-K. Kim, E. Alshina, A. Alshin, T. Lee, J. Chen, V. Seregin, S. Lee, Y. M. Hong, M. S. Cheon, N. Shlyakhov, K. McCann, T. Davies, and J. H. Park, "Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1709–1720, Dec. 2010.

[16] D. Marpe, H. Schwarz, S. Bosse, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, T. Nguyen, S. Oudin, M. Siekmann, K. Suhring, M. Winken, and T. Wiegand, "Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1676–1687, Dec. 2010.
[17] I.-K. Kim, J. H. Min, T. Lee, W.-J. Han, and J. H. Park, "Large and various shapes block processing in HEVC," in *Proc. Appl. Digital Image Process.*, vol. 8499. Aug. 2012, p. 84990Q.
[18] P. Bordes, E. François, and D. Thoreau, "Fast encoding algorithms for geometry-adaptive block partitioning," in *Proc. Int. Conf. Image Process.*, Sep. 2011, pp. 1205–1208.
[19] A. Puri, H. M. Hang, and D. Schilling, "An efficient block-matching algorithm for motion-compensated coding," in *Proc. Int. Conf. Acoustics Speech Signal Process.*, Aug. 1997, pp. 1063–1066.
[20] M. H. Chan, Y. B. Yu, and A. G. Constantinides, "Variable size block matching motion compensation with applications to video coding," in *Proc. IEE Commun., Speech Vision*, Aug. 1990, pp. 205–212.
[21] 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.
[22] J. J. Zhang, M. O. Ahmad, and M. N. Swamy, "Quadtree structured region-wise motion compensation for video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 5, pp. 808–822, Aug. 1999.
[23] H. Schwarz, D. Marpe, and T. Wiegand, *Investigations for Representing Rectangular Blocks Using the Merging Concept*, document JCT-VC C306, Joint Collaborative Team on Video Coding (JCT-VC), Guangzhou, China, Oct. 2010.
[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.
[25] S. Liu, Y.-W Huang, and S. Lei, *Remove Partition Size $N \times N$*, JCT-VC document D432, 4th JCT-VC Meeting, Jan. 2011.
[26] M. Winken, P. Helle, D. Marpe, H. Schwarz, and T. Wiegand, "Transform coding in the HEVC test model," in *Proc. Int. Conf. Image Process.* , Sep. 11–14, 2011, pp. 3693–3696.
[27] K. Panusopone, X. Fang, and L. Wang, *Proposal on RQT Root Location*, JCT-VC document E364, 5th JCT-VC Meeting, Mar. 2011.
[28] X. Zheng, Y. Yuan, and Y. He, *Non-CE2: Harmonization of Implicit TU, AMP and NSQT*, JCT-VC document G519, 7th JCT-VC Meeting, Nov. 2011.
[29] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document ITU-T SG16 Q.6, VCEG-M33, Austin, TX, Apr. 2001.
[30] G. Bjøntegaard, *Improvement of BD-PSNR Model*, document ITU-T SG16 Q.6, VCEG-AI11, Berlin, Germany, Jul. 2008.
[31] F. Bossen, *Common Test Conditions*, document JCT-VC H1100, Joint Collaborative Team on Video Coding (JCT-VC), San Jose, CA, Mar. 2012.



Il-Koo Kim received the B.S., M.S., and Ph.D. degrees from the School of Electrical Engineering, Seoul National University, Seoul, Korea, in 1999, 2001, and 2006, respectively.

He has been with the Digital Media and Communications Research and Development Center, Samsung Electronics, Suwon, Korea, since 2006, where he has been engaged in development of high-efficiency video codecs. Since 2010, he has been an active contributor to the Joint Collaborative Team on Video Coding for High Efficiency Video Coding standardization. His main research interests include video compression and transmission.

Junghye Min received the B.S. degree in mathematics from Ewha Women's University, Seoul, Korea, in 1995, and the M.E. and Ph.D. degrees in computer science and engineering from Pennsylvania State University, University Park, in 2003 and 2005, respectively.

From 1995 to 1999, she was a Software Engineer with Telecommunication Systems Business, Samsung Electronics, Suwon, Korea, where she is currently a Principal Engineer with the Multimedia Platform Laboratory, Digital Media and Communications Research and Development Center. Her current research interests include video content analysis, motion tracking, pattern recognition, and video coding.

Tammy Lee received the B.S. degree in mathematics education from Ewha Women's University, Seoul, Korea, in 2000, and the M.S. degree in computer science from New York University, New York, in 2005.

She is currently a Senior Engineer with the Multimedia Platform Laboratory, Digital Media and Communications Research and Development Center, Samsung Electronics, Suwon, Korea. Her current research interests include image and video compression.

Woo-Jin Han (M'02) received the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1997 and 2002, respectively.

He is currently a Professor with the Department of Software Design and Management, Gachon University, Seongnam, Korea. From 2003 to 2011, he was a Principal Engineer with the Multimedia Platform Laboratory, Digital Media and Communication Research and Development Center, Samsung Electronics, Suwon, Korea. Since 2003, he has contributed successfully to the ISO/IEC Moving Pictures Experts Group, Joint Video Team, and Joint Collaborative Team standardization efforts. In 2010, he was an editor of the HEVC Video Coding Standard. His current research interests include high-efficiency video compression techniques, scalable video coding, multiview synthesis, and visual contents understanding.

JeongHoon Park received the B.S. and M.S degrees in electrical engineering from Hanyang University, Seoul, Korea, in 1991, and 1994, respectively.

He is currently a Principal Engineer at the Multimedia Platform Laboratory, Digital Media and Communication R&D Center, Samsung Electronics, Suwon, Korea. In 1998, he was a Visiting Scholar at the University of California, Los Angeles, where he researched on error resilient video compression and transmission. Since then, he has contributed to the ITU-T Video Coding Experts Group and ISO/IEC Moving Pictures Experts Group. From 2002 to 2004, He has been involved in developing mobile broadcasting technology as an active member of WorldDMB forum. His research interests include audio and video compression technology, medical image processing, and multimedia systems, including video broadcasting and communications.