High-Efficiency Video Compression Framework Based on Flexible Unit Representation

Sunil Lee*, Woo-Jin Han*, Junghye Min*, Il-Koo Kim*, Elena Alshina*, Alexander Alshin*,
Tammy Lee*, Jianle Chen*, Vadim Seregin*, Yoon-Mi Hong*, Min-Su Cheon*,
Nikolay Shlyakhov*, Ken McCann†, Thomas Davies‡, and Jeong-Hoon Park*
* DMC R&D Center, Samsung Electronics, Suwon, Korea
E-mail: sunil.lee@samsung.com Tel: +82-31-279-8829
† ZetaCast, 35 Bathwick Street, Bath, BA2 6NZ, UK
E-mail: ken@zetacast.com Tel: +44-(0)-1962-625347
‡ British Broadcasting Corporation R&D, UK
E-mail: Thomas.Davies@bbc.co.uk

***Abstract*—This paper proposes a high-efficiency video compression framework based on a highly flexible hierarchy of unit representation which includes three block concepts: coding unit (CU), prediction unit (PU), and transform unit (TU). This separation of the block structure into three different concepts allows each to be optimized according to its role; the CU is a macroblock-like unit which supports region splitting in a manner similar to a conventional quadtree, the PU supports non-square motion partition shapes for motion compensation, while the TU allows the transform size to be defined independently from the PU. Several other coding tools are extended to arbitrary unit size to maintain consistency with the proposed design, e.g. transform size is extended up to 64×64 and intra prediction is designed to support an arbitrary number of angles for variable block sizes. The video codec described in this paper was a candidate in the competitive phase of the High-Efficiency Video Coding (HEVC) standardization work. Compared to H.264/AVC, it demonstrated bit-rate reductions of around 40% based on objective measures and around 60% based on subjective testing with 1080p sequences. It has been partially adopted into the first standardization model of the collaborative phase of the HEVC effort.**

I. Introduction

With the ever increasing popularity of high-definition (HD) video content, video compression technologies which can provide higher coding efficiency than existing video coding standards, e.g. state-of-the-art H.264/AVC [1], have received increased attention. To meet the growing demand, recently, ITU-T SG16 Q.6 (VCEG) and ISO/IEC JTC1/SC29/WG11 (MPEG) created a Joint Collaborative Team on Video Coding (JCT-VC), and jointly issued a Call for Proposals (CfP) on video compression technology called the High-Efficiency Video Coding (HEVC) [2]. The coding performance reported in a few responses to the CfP was already significantly higher than that of the H.264/AVC, and the video codec described in this paper was one of the best performing candidates in terms of both objective and subjective performance measures [3]. The proposed video codec could achieve high coding efficiency by using highly flexible unit representation and the novel coding tools adapted to the flexibility. In this paper, we focus on the flexible unit representation which is the key concept of the proposed codec. The interested readers can refer to our response to the CfP [4], for the details of other coding tools.

Motion compensation is one of the key modules in hybrid video codec. Fixed-size block motion compensation (FSBMC) methods assume that each block of pixels possesses the same translational motion and compensate the motion of a frame block by block with same size. However, since the boundaries of the moving objects seldom coincide with the block boundaries in a real image sequence, these methods result in poor prediction performance. Variable-size block motion-compensation (VSBMC) techniques were proposed to alleviate this problem [5]–[7]. In the VSBMC techniques, a frame is divided into fixed-size blocks, and then each block is further split and merged iteratively. The VSBMC techniques usually utilize a quadtree to represent the variable-size block structure. The quadtree structure for the blocks is generated by considering the tradeoff between prediction error and overhead for side information [8].

Early video coding standards such as MPEG-1 [9] and MPEG-2 [10] employ FSBMC method with a macroblock size of 16×16. In the H.264/AVC [1], this macroblock size is retained but with the addition of a depth-2 quadtree, which significantly improves coding efficiency. However, it has been observed that the maximum 16×16 block size is too small when applied to high resolution video and causes inefficiency [11], [12]. To solve this problem, recent VCEG proposals have extended the maximum motion block size up to 64×64 by adding additional large motion block modes on top of the conventional 16×16 H.264/AVC macroblock syntax [13]–[15].

Although those approaches suggested that the coding efficiency for the high resolution video materials can be improved by providing a better tradeoff between prediction performance and side information, they simply tried to add the large size motion partition while maintaining the existing 16×16 macroblock sizes, which imposes restrictions on the flexibility of the quadtree structure.

This paper describes a video compression framework where the whole codec has been designed to exploit flexible block

Proceedings of the Second APSIPA Annual Summit and Conference, pages 623–631,
Biopolis, Singapore, 14-17 December 2010.

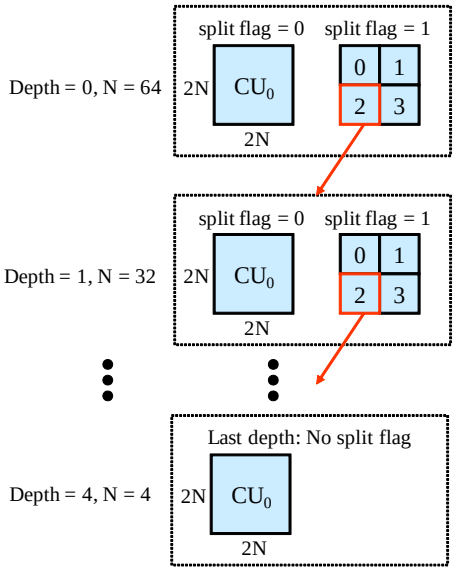

Fig. 1. Illustration of recursive CU structure (LCU size = 128, maximum hierarchical depth = 5).

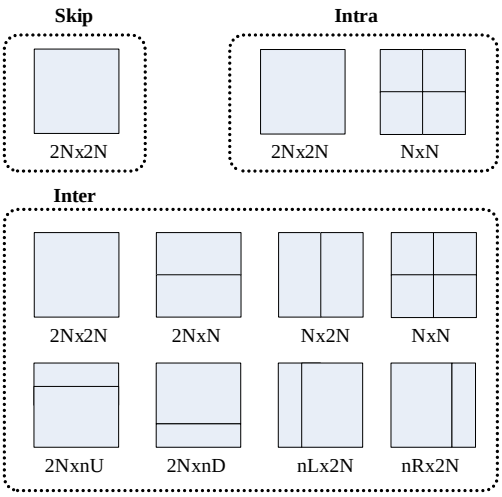

Fig. 2. Supported PU splittings according to each prediction mode.

sizes. A highly flexible hierarchical structure based on the generic quadtree scheme is defined by three independent block concepts: coding unit (CU), prediction unit (PU), and transform unit (TU). CU is the basic unit of region splitting. CU is analogous to the concept of macroblock, but it does not restrict the maximum size and it allows recursive splitting into four equal size CUs to improve the contents-adaptivity. PU is the basic unit of inter/intra prediction and it may contain multiple arbitrary shape partitions in a single PU to effectively code irregular image patterns. TU is the basic unit of transform. It can be defined independently from the PU, however, its size is limited to the CU which the TU belongs to. This separation of the block structure into three different concepts allows each to be optimized according to its role, which results in the improved coding efficiency.

From these unit definitions, all coding processes can be defined in a very consistent way; large block sizes are supported in both motion compensation and intra prediction schemes, prediction mode switching is allowed at any block sizes and spatial transforms larger than 16×16 are also supported in both inter and intra coded blocks. Due to this consistency, every syntax element can be defined in exactly the same way independent of the unit sizes, which makes the specification and parsing process much simpler than would be the case with an extension scheme on top of a conventional 16×16 macroblock syntax.

The rest of this paper is organized as follows. Section II provides the details of the proposed flexible unit representation. Section III briefly introduces the coding tools of the proposed video codec which include those for inter/intra-frame prediction, spatial transform, in-loop filtering, and entropy coding. Section IV summarizes the coding efficiency of the proposed codec based on the objective and subjective results. Finally, Section V concludes the paper.

II. Flexible Unit Representation

For efficient and flexible representation of video content with various resolutions, a triplet of coding unit (CU), prediction unit (PU), and transform unit (TU) are introduced. The following subsections describe the details of each unit.

A. Coding unit (CU)

The picture is divided into slices, with each slice composed of a sequence of largest coding units (LCUs). The LCU size is specified in the Sequence Parameter Set (SPS) or slice header; the optimum LCU size is dependent on the application.

The LCU can be divided into four CUs by a 1-bit syntax, split_flag. Each split CU can be divided into four CUs recursively up to the maximum allowed hierarchical depth, also specified in the SPS or slice header. For example, if LCU size is equal to 128 and the maximum hierarchical depth is equal to 5, then 5 kinds of CU sizes are possible: 128×128 (LCU), 64×64, 32×32, 16×16, and 8×8 which is the smallest CU (SCU). If LCU size is equal to 16 and the maximum hierarchical depth is equal to 2, then 16×16 (LCU) and 8×8 (SCU) are possible; this is a block structure similar to the H.264/AVC. Fig. 1 shows an example of the recursive CU structure where the LCU size is 128 and the maximum hierarchical depth is 5. In the proposed video codec, the three syntax elements, LCU size, maximum hierarchical depth and split_flags, completely specify the generic multi-depth quadtree.

B. Prediction unit (PU)

Coupled with CU, this paper introduces a basic unit for prediction: the prediction unit (PU). The PU is defined only for the leaf node CU which is not split in quadtree structure, and the size of the PU is limited to that of the CU. The prediction method is specified by the prediction mode and the PU splitting. As with H.264/AVC, the prediction mode can be one of the values among skip, intra, and inter. Fig. 2 shows

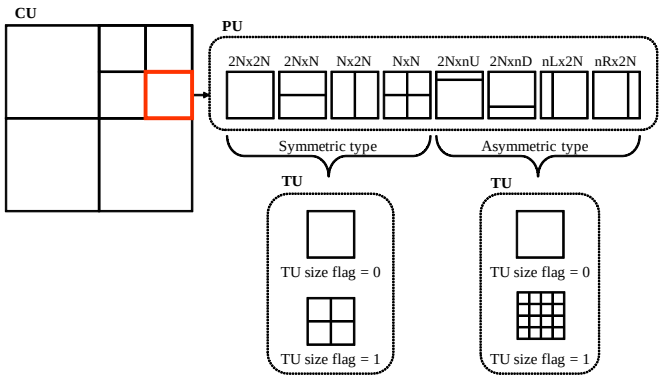

Fig. 3. Relationship between CU, PU, and TU.

different PU splitting strategy for a CU of size 2N×2N, which is dependent on the property of each prediction mode.

Skip mode can be thought of as a special inter mode in which no motion vector and residue information are transmitted. Typically it is used in an area of the image with smooth motion, thus no further splitting is necessary and only 2N×2N partition is used. For the normal inter mode, a set of 8 splitting options are defined to cover possible motion boundaries inside the PU. Given a PU of size 2N×2N, 4 symmetric splittings (2N×2N, 2N×N, N×2N, N×N) and 4 asymmetric splittings (2N×nU, 2N×nD, nL×2N, nR×2N) are defined. The asymmetric PU splittings are included in the design to improve the coding efficiency for the irregular object boundaries, which otherwise would be constrained to being represented less efficiently by further CU splittings to the deeper depth. For the intra mode, only two possible splittings 2N×2N and N×N are defined.

All information related to the prediction is signaled on a PU basis, for example, the prediction mode and PU splitting type are specified for each PU. Once the prediction mode and PU splitting is given, motion vector difference, intra prediction direction, reference indices for each PU partition are specified on a PU partition basis.

C. Transform unit (TU)

The transform unit (TU), for transform and quantization, is the third basis unit. Although, in principle, TU can be defined in a similar way to PU to support arbitrary shape transforms, only two TU partitions are used in this paper considering the implementation complexity.

When the tu_size_flag is equal to zero, the TU size is set equal to that of the CU which it belongs to. When tu_size_flag is equal to one, the TU size is set as N×N for symmetric PU splittings and N/2×N/2 for asymmetric PU splittings, respectively. This ensures that the transform which is not applied across motion boundaries can be tested in the rate-distortion optimization process for asymmetric PU partitions.

It should be noted that the transform may be applied to the residue generated by multiple PU partitions with different motion vectors. This approach is especially beneficial for coding semi-random residuals generated by complex texture areas [4].

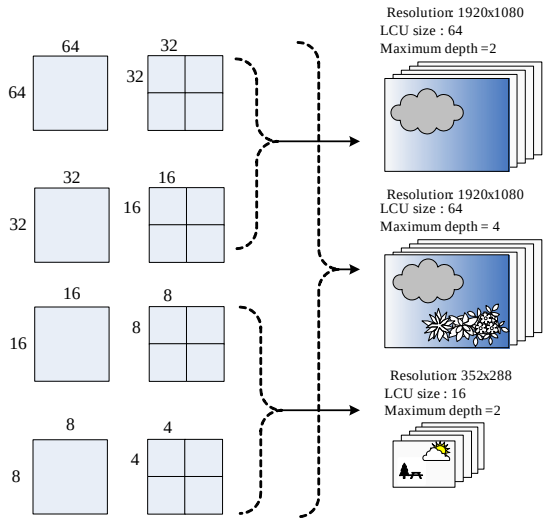

Fig. 4. Example of LCU size and maximum depth combinations for various resolutions.

D. Relationship between CU, PU, and TU

Fig. 3 shows the relationship between the three different unit concepts, CU, PU, and TU. Given a CU of any size, a series of split flags is used to specify smaller CUs inside the CU. For every CU which does not split any more, a PU is defined specifying how the prediction can be generated. Note that the PU does not need to be a square shape, and virtually any kind of new PU splitting can be added without changing the overall structure. TU size is specified by tu_size_flag and PU splitting as explained in Section II-C.

E. Advantages of the flexible unit representation

The proposed unit representation provides several major benefits. The first benefit comes from the easy support of CU sizes larger than the conventional 16×16 macroblock. As mention in previous works [13], [16], [17], this feature can dramatically reduce the side information when applied to HD video. Additionally, the proposed method introduces the separate PU and TU conception to further improve the coding performance.

Furthermore, since the LCU size and maximum hierarchical depth are specified in syntax, the unit sizes can be adaptively optimized for various content, applications, and devices. Compared to the use of fixed size macroblock or super-macroblock, support of various LCU sizes is one of the major advantages of the proposed codec. By choosing LCU size and maximum hierarchical depth appropriately, the hierarchical block structure can be optimized in a better way for the targeted application. From the aspect of the standardization, the range of LCU sizes could be specified in the Profiles and Levels section to match the requirements more specifically. Fig. 4 shows examples of CU size combinations for several resolutions.

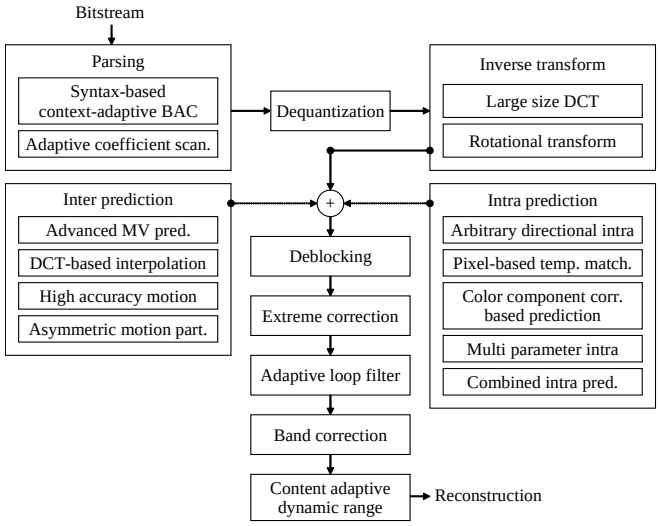

Fig. 5. All building blocks to decode the input bit-stream.

Finally, the proposed unit definitions allow us to design a size-independent syntax representation which specifies all syntax elements in a consistent way independent of the unit size. By contrast, in H.264/AVC and its extension in [13]–[15], block level syntax elements such as prediction_mode, transform_flag, coded_block_flag and intra_pred_mode are coded differently depending on block size. This size-independent method of representation allows syntax elements to be specified generically for all the remaining coding tools. This property can greatly simplify the specification effort as well as the actual parsing process, especially if a large number of hierarchical depths is allowed.

III. Coding tools

Fig. 5 shows all proposed building blocks of the decoder. All the coding tools are extended to support the proposed flexible architecture. Intra prediction is extended to support an arbitrary number of angles rather than the conventional 9-modes defined in H.264/AVC. Fast integer transforms larger than 8×8 are developed to support large TU sizes. Edge definition for the deblocking filter is modified according to the CU, PU, and TU concepts. Partition information in quadtree-based adaptive loop filter (QALF) [18] is replaced with CU splitting information. The details of each coding tool are briefly introduced in the following sections.

A. Intra prediction

1) Arbitrary directional intra (ADI): In H.264/AVC intra coding, 9 intra prediction modes are supported for 4×4 and 8×8 blocks but only 4 intra prediction modes for 16×16 blocks. Intra-coded 16×16 blocks were intended to code homogeneous regions in low resolution video. However, even relatively large blocks ($\geq$ 16×16) cover only a small part of objects or background in high resolution video, which may contain strong directional patterns.

Arbitrary directional intra (ADI) is a coding tool designed to extend the directional intra prediction scheme used in H.264/AVC to give more accurate representation of directions and better suit larger prediction units. In the ADI scheme, up to 30 angles are generated using the integer pair (dx, dy) to represent the direction of lines which each mode uses for context pixel extrapolation.

Due to the large variety of the possible unit sizes, it is beneficial to adjust the number of angles according to the unit size. For example, PU partitions with 64×64 and 128×128 sizes are usually used for homogeneous areas which can be generated by using only some prediction modes, such as DC and plane modes. In this case, the number of supported prediction modes of the current PU partition may differ from that of the neighboring PU partitions. Intra prediction modes of the neighboring PU partitions are converted to the 9 most frequently used modes to predict the current intra prediction mode in a unified way.

2) Multi-parameter intra (MPI): Directional intra prediction sometimes results in non-smooth and unnatural patterns which are not transform-friendly. This problem can be alleviated by applying a smoothing filter to the prediction signal obtained by the ADI scheme. Multi-parameter intra (MPI) allows the used of the 4-point filter for each pixel to increase the probability of generating a prediction signal well-suited to the following transform stages. For each PU partition, mpi_flag is used to specify whether or not the MPI scheme is applied.

3) Color Component correlation based prediction (CCCP): There have been some studies on predicting chrominance pixels using the correlation with luminance [19]. Although most approaches try to predict the pixel values of one color channel by another color channel, the color component correlation based prediction (CCCP) scheme derives an object segmentation map of the chrominance channel from the already coded luma channel, and applies it to intra prediction.

The CCCP procedure is as follows. Firstly, the size of the reconstructed luma samples including the neighboring luma pixels are adjusted to match that of the chroma samples according to the given chroma format. Then, simple thresholding using the mean value is exploited to generate the binary segmentation map. After that, two mean values are computed from the neighboring chroma pixels according to the segmentation map and assigned to the inner part of the chroma block. Finally, a 3×3 averaging filter is applied to improve the smoothness of the prediction.

4) Pixel based template matching (PTM): Whilst directional extrapolation based intra prediction tools are very efficient for regions with directional patterns, they are not well-suited to regions with repeated regular patterns. To overcome this limitation, several intra techniques using some form of template matching based on the already decoded pixels have been proposed [20], [21]. Although the reported schemes were proven to provide good coding gains for the targeted areas, a major drawback was the high complexity at the decoder side.

In this paper, a simple pixel based template matching (PTM) approach is proposed. PTM is designed to apply template

matching at the pixel level to maximize the degrees of freedom. PTM uses only a small number of search points and both decoded and previously predicted pixels as template and candidate to minimize the complexity.

5) Combined intra prediction (CIP): Combined intra prediction (CIP) is a coding tool for providing improved prediction, especially in large blocks where a directional linear prediction may not work well, even with the large range of angles available to ADI. CIP predictions comprise a weighted combination of the prediction samples provided by other intra coding tools together with a pixel-by-pixel mean prediction. It provides pixel-by-pixel adaptation but is a simpler tool than, for example, local template matching approaches.

B. Inter prediction

1) High accuracy motion (HAM): Even though higher motion accuracy results in less prediction error, it does not guarantee improved coding efficiency, since it leads to an increase in the bit-rate to convey the motion information. For example, 1/8-pel motion accuracy may often result in a reduction in the overall coding efficiency, except for particular sequences [22].

In this paper, high accuracy motion (HAM), which adaptively refines the motion information to an arbitrary accuracy, is proposed. In the HAM method, motion information at quarter-pel accuracy is followed by a refinement flag and optional refinement information at 1/12-pel accuracy. The refinement flag is set equal to 1 only when the refinement provides the gain in the rate-distortion optimization sense, otherwise, the flag is set to 0 and no refinement information is signaled. Although the refinement information can represent any accuracy, the current design uses 1/12-pel, since refinement in 1/8-pel accuracy results in the redundancy in the refinement information.

2) DCT-based interpolation filter (DCT-IF): In H.264/AVC standard, the prediction values at half-pel positions are obtained using a 6-tap Wiener filter, while those at quarter-pel positions are obtained by bi-linear combination of the samples at integer and half-pel positions [1]. This paper proposes a DCT-based interpolation filter (DCT-IF), which can directly provide the interpolated value at the desired fractional accuracy from the samples at integer-pel positions. Since the DCT-IF can provide the interpolated value without cascaded filtering, the motion compensation process can be simplified and the complexity can also be reduced for samples at quarter-pel accuracy. Furthermore, the DCT-IF provides a unified way to generate the interpolation filters supporting motion accuracy higher than quarter-pel, e.g. 1/8-pel, 1/12-pel, etc. For example, Table I shows the coefficients of the 6-tap filters generated for 1/12-pel motion accuracy. Note that the filter coefficients are scaled by 256.

The proposed DCT-IF is a non-cascaded interpolation scheme, i.e. only one spatial filtering is performed for all the possible positions. It can be implemented as a multiplication-free design by optimizing the filter coefficients. Thus, the 6-tap DCT-IF has the smaller number of operations compared to the interpolation filter of H.264/AVC since samples at the quarter-pel position can be generated by one 6-tap filtering operation whereas H.264/AVC uses a combination of 6-tap and bi-linear filters. The 12-tap DCT-IF filter requires twice as many operations as the 6-tap DCT-IF, however it provides similar performance to the well-known adaptive interpolation filter (AIF) scheme [23] while being less complex and more suitable for hardware implementation.

TABLE I
FILTER COEFFICIENTS OF 6-TAP DCT-IF.

α	$2M = 6$ (6-tap filter)						
-1/12	{	-4,	19,	254,	-19,	8,	-2 }
1/12	{	4,	-16,	252,	22,	-8,	2 }
2/12	{	6,	-28,	242,	48,	-17,	5 }
3/12	{	9,	-37,	227,	75,	-25,	7 }
4/12	{	11,	-42,	208,	103,	-33,	9 }
5/12	{	12,	-44,	184,	132,	-39,	11 }
6/12	{	11,	-43,	160,	160,	-43,	11 }
7/12	{	11,	-39,	132,	184,	-44,	12 }
8/12	{	9,	-33,	103,	208,	-42,	11 }
9/12	{	7,	-25,	75,	227,	-37,	9 }
10/12	{	5,	-17,	48,	242,	-28,	6 }

3) Advanced motion vector prediction (AMVP): In H.264/AVC standard, motion vectors are predicted from a median of the motion vectors of the spatially adjacent blocks [1]. To further improve the efficiency of the motion vector prediction, a few competition-based prediction methods have been proposed [24], [25], where the best predictor is selected from a given set through rate-distortion optimization.

This paper proposes the advanced motion vector prediction (AMVP) method, which is adapted to the proposed flexible unit representation. It allows the selection of the best predictor from the set which consists of three spatially adjacent motion vectors, their median, and a temporal motion vector. The overhead for signaling the index of the best predictor is reduced by re-organizing the set of candidate predictors. First, the predictors in the candidate set are reordered based on the PU splitting, so that the most probable motion predictor appears first in the set. Then, duplicate predictors and the predictors which can be excluded based on the parsed MVD values at the decoder side are eliminated from the set [25]. Note that, when only one predictor remains as a result of the re-organization, no overhead is signaled, since it can be also derived at the decoder.

C. Spatial Transforms

It is well known that transforms larger than 8×8 provides better energy compaction and less quantization error for smooth and coarse textured video areas at the expense of computational complexity [17], [26]. In this paper, 16×16, 32×32 and 64×64 fast integer transforms are presented in addition to the conventional 4×4 and 8×8 transforms to support large TUs. In addition, a rotational transform (ROT) tool, which can adaptively compact the energy into low frequency components

by rotating the transform basis, is also proposed as a second transform.

1) Large integer transform (16×16, 32×32 and 64×64): The discrete cosine transform (DCT) is independent of input signal while it still provides comparable decorrelation ability to the Karhunen-Loève Transform (KLT), which is highly signal-dependent. Moreover, many fast DCT algorithms have been developed. All large transforms proposed in this paper are scaled integer DCTs based on Chen's fast factorizations [27]. Chen's fast DCT has a regular butterfly structure which is beneficial to hardware implementation.

Similarly to the H.264/AVC 4×4 core transform, each multiplication factor in the flow-graph of the fast algorithm is scaled and approximated by some fixed precision values p, i.e. each constant is approximated by the dyadic rational of the form $q/2^p$, which can be implemented with only additions and bit-shifting operations. The precision value p can be chosen according to the dynamic range limitation and coding efficiency. As all multiplication factors are approximated, the core transform may not be truly orthogonal. However, this minor non-orthogonality does not result in any perceptible negative effect on the compression performance, whilst the complexity can be significantly reduced. Furthermore, it is still possible to achieve perfect reconstruction using a lifting scheme that can be applied additionally to the butterfly structure [28].

After all multiplication factors are determined, the forward 2-D transform is performed separately in vertical and horizontal direction as follows:

$$Y = C_f(C_f X^T)^T = C_f X C_f^T \quad (1)$$

where X is N×N residual data, C_f is the core N-point forward transform, and Y is N×N transformed coefficients. The calculation of $C_f X^T$ is performed from left to right in the flow-graph of the fast algorithm. For the inverse transform, the computation order is reversed from right to left. Since the core transform C_f is scaled DCT, the remaining scaling factors should be integrated into the quantization stage.

2) Rotational transform (ROT): While DCT is the most widely used block transform for video and image codecs, the DCT basis functions are not optimal for some types of residual signal. For example, a residual signal having strong diagonal components cannot be represented efficiently with the DCT basis vectors. Typically, directional transform schemes are proposed to exploit this problem such as Mode Dependent Directional Transform (MDDT) [29], [30].

However, the direct use of directional transform is not appropriate to the proposed flexible unit definitions since it is difficult to design the directional transform by a fast butterfly structure such as DCT. In addition, the number of transforms would be almost doubled because both DCT and the directional transform for various TU sizes would be kept in the codec design.

In this paper, rotational transform (ROT) is proposed as a second transform that could be applied after the core integer transform instead of introducing completely new transform cores. The main idea of ROT is to change the coordinate system of the transform basis, instead of direct rotation of the input residual. The rotation matrices of horizontal and vertical directions can be obtained from a set of angles predefined in the ROT dictionary. The ROT dictionary was first generated by Monte Carlo method using Lehmer's pseudo-random numbers for angles [31], [32]. Then the 4 most frequent elements of rotational transform dictionary were selected in order to minimize encoder-side complexity. The index of rotational angles in the dictionary is signaled to the decoder explicitly.

Since ROT is a second transform, not all of the DCT coefficients are processed. For TUs larger than 4×4, only 8×8 low-frequency areas are processed. It has been found that this restriction does not degrade the coding efficiency since most of the coefficients are already compacted into these low-frequency areas.

D. In-loop Filtering

1) Deblocking filter: The same process as used in H.264/AVC is employed for the deblocking filter. However, the use of a large size transform can effectively reduce the complexity of the deblocking filter, by reducing the number of deblocked pixels. It was reported that the number of deblocked pixels was reduced by 43% on average [4] for the 1080p sequences used in [2].

2) CU-synchronized adaptive loop filter (CU-ALF): Quadtree based adaptive loop filter (QALF) [18] is known to provide significant coding gain by effectively combining both the Wiener filter approach and the quadtree based on/off selections. In QALF, along with filter coefficients, the information representing the quadtree structure is coded independently with the prediction quadtree structure, which results in large amount of side information in the slice header.

Since the proposed unit representations provides a generic quadtree structure similar to the original design of QALF, this information can be re-utilized for the adaptive loop filter process. CU-synchronized adaptive loop filter (CU-ALF) is based on the QALF approach; however, it uses the already coded CU hierarchy as the on/off control for the adaptive loop filter process instead of coding separate quadtree information in the slice header.

It has been reported that the proposed CU-ALF provides comparable coding gain to QALF, while eliminating the need for any side information related to the quadtree partition in the slice header and reducing the encoder complexity [4].

3) Extreme correction (EXC) and band correction (BDC): Extreme correction (EXC) is designed to compensate the distortion according to the identified pixel classes. Each pixel is categorized as local maxima, local minima, or object edge by simple computation based on the four neighboring pixel values: left, above, right, and bottom. For each pixel class, the mean difference between the original and the reconstructed signals is computed and transmitted to the decoder. A fuller description is given in [4]. Band correction (BDC) is another form of pixel classification, based on pixel value ranges. The main motivation is to equalize the different probability distribution functions of the reconstructed signal with the

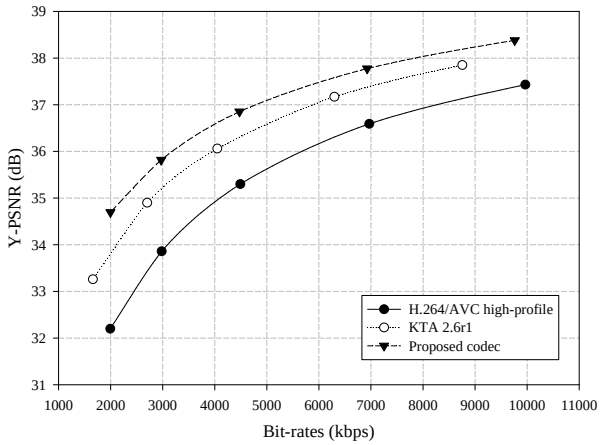

Fig. 6. Rate distortion curve of BasketballDrive sequence in Class B (1080p).

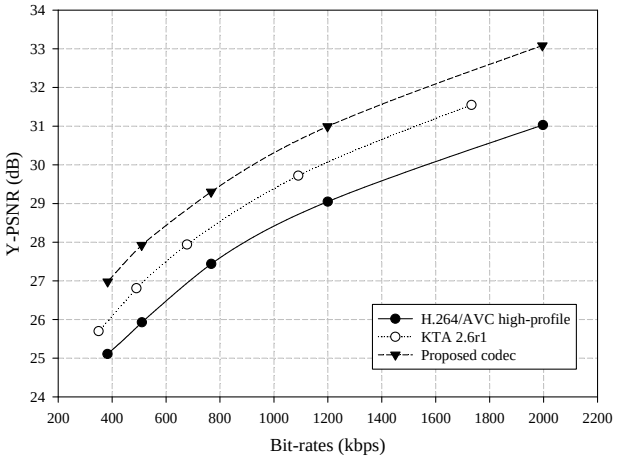

Fig. 7. Rate distortion curve of PartyScene sequence in Class C (WVGA).

original. In this paper, up to 16 value ranges are defined; the mean difference for each band is computed and transmitted to the decoder.

4) Content adaptive dynamic range (CADR): Internal bit depth increase (IBDI) is a technique to reduce the rounding error by enlarging the internal data precision [33]. Typically, IBDI is implemented by increasing the source bit-depth, which makes the encoder implementation more difficult. Content adaptive dynamic range (CADR) is related to IBDI, but is a more general form of source scaling. Using the CADR, a pixel value x is adaptively scaled to $F(x)$ as follows

$$F(x) = (2^{\text{bit}} - 1)(x - \text{Min})/(\text{Max} - \text{Min}) \qquad (2)$$

where bit, Max and Min are the bit-depth, the maximum value and the minimum value of the input source, respectively. It should be noted that the parameters Min and Max can be adjusted at sequence, slice, or even block level. CADR can be used to reduce the rounding error without requiring increased bit-depth in the encoder.

E. Entropy Coding

1) Syntax-based context-adaptive binary arithmetic coding (SBAC): As with the context-adaptive binary arithmetic coding (CABAC) of H.264/AVC [34], the SBAC consists of four major steps: syntax-based binarization, context modeling, probability estimation, and binary arithmetic coding. The syntax-based binarization and the context modeling are adapted to the syntax elements obtained from the flexible unit representation and the extended coding tools. The probability estimation and binary arithmetic coding are based on an improved version of the arithmetic coding method specified in Annex D of the JPEG still image coding standard [35]. Specifically, the table-based probability update and the multiplication-free design are improved, and the bypass mode for equi-probable binary symbols is added to reduce complexity [4].

2) Adaptive coefficient scanning (ACS): Since large transforms frequently have very sparse coefficient distributions, a fixed zigzag scanning method may result in long coefficient scanning lists including many zeroes, resulting in inefficient coding. Adaptive coefficient scanning (ACS), which adaptively chooses the best scanning method for each TU, is used to avoid this problem. Given the quantized transform coefficients, the best scanning pattern is chosen from conventional zigzag, horizontal, and vertical scanning patterns, and then its index is explicitly signaled in each TU. Note that the ACS index is signaled only when non-zero AC coefficients exist in the TU, since there is no difference between the patterns when the TU includes only the DC coefficient. When the horizontal or vertical scanning methods are chosen as the best scanning method, the mapping process from the coefficient array indices to the zigzag scanning indices is not necessary. Thus, the complexity for the coefficient indexing can be reduced and also the memory bandwidth to access the zigzag mapping table can be saved.

IV. Performance Analysis

To verify the performance of the proposed coding tools, they were implemented into one complete video codec software which is available at [4]. This section provides both objective performance based on the peak signal-to-noise ratio (PSNR) measure and the subjective performance based on the mean opinion score (MOS) values for the proposed video codec.

A. Objective compression efficiency

The coding efficiency of the proposed system is verified using two sets of coding configurations defined in the Call for Proposals [2] — constraint set 1 (CS1) and constraint set 2 (CS2) which correspond to a random access case and a low delay case, respectively. Fig. 6 and Fig. 7 show typical examples of rate-distortion curves when video sequences are coded using CS1 configuration. As shown in the figures, the proposed codec consistently showed significant coding benefits over H.264/AVC High Profile. For comparison with other recent advances in the video coding area, the results of the KTA 2.6r1 [36] with the same test configurations are also included. This test configuration is same as in [37], where all

TABLE II
SUMMARIZED BD-RATE REDUCTIONS FOR VARIOUS RESOLUTIONS.

Class (Size)	Sequence name	CS1 (%)		CS2 (%)	
		BD-rate	Avg.	BD-rate	Avg.
A (4K)	Traffic	40.59	35.21	N/A	N/A
	PeopleOnStreet	29.84		N/A	
B (1080p)	Kimono	44.42	43.29	44.45	43.66
	ParkScene	32.96		31.18	
	Cactus	40.78		34.74	
	BasketballDrive	44.21		47.29	
	BQTerrace	54.06		60.62	
C (WVGA)	BasketballDrill	43.09	39.94	36.82	35.58
	BQMall	39.69		37.06	
	PartyScene	41.13		40.45	
	RaceHorses	35.86		27.99	
D (WQVGA)	BasketballPass	30.55	36.45	26.93	32.16
	BQSquare	54.19		51.87	
	BlowingBubbles	33.97		28.24	
	RaceHorses	27.08		21.59	
E (720p)	Vidyo 1	N/A	N/A	48.38	47.48
	Vidyo 3	N/A		46.26	
	Vidyo 4	N/A		47.79	
Total avg.		39.49		39.48	

TABLE III
SUMMARIZED MOS VALUES FOR 1080P SEQUENCES.

Sequence	H.264/AVC		Proposed		
	Bit-rates (kbps)	MOS	Bit-rates (kbps)	MOS	Reduction (%)
Kimono	4000	8.86	1600	8.56	60.0
ParkScene	4000	7.41	1600	7.78	60.0
Cactus	7000	8.61	3000	8.24	57.1
BasketballDrive	7000	7.44	3000	8.17	57.1
BQTerrace	10000	8.77	3000	9.17	70.0
Average	6400	8.22	2440	8.38	60.9

major tools of KTA including enhanced AIF, MV competition, MDDT, high precision filter, ALF, and extended block size are enabled. It should be noted that the improved coding efficiency can be observed with both high resolution and low resolution video material.

Table II summarizes the performance of the proposed codec relative to the H.264/AVC High Profile. The average compression performance improvement over all sequences is 39.49% in CS1 and 39.48% in CS2, respectively. Class B (1080p resolution) achieves 43.29% (CS1) and 43.66% (CS2) bit saving on average, with BQTerrace as the best sequence in both configurations, showing 54.06% (CS1) and 60.62% (CS2) performance improvement, respectively. RaceHorses in Class D (WQVGA resolution) shows the lowest performance improvement in both configurations, at 27.08% in CS1 and 21.59% in CS2, respectively. The performance summarized in Table II shows that the proposed codec consistently achieves the excellent coding performance regardless of the resolution of input video sequences and the coding configuration.

B. Subjective compression efficiency

To evaluate the submitted responses to the CfP on HEVC standard [2], a large scale subjective test was carried jointly by ISO/IEC JTC1/SC29/WG11 (MPEG) and ITU-T SG16 Q.6 (VCEG). The official report of subjective test results can be found in [3]. It was shown that the proposed codec provided excellent subjective improvements over H.264/AVC in all configurations and it gave the best overall results of all of the proposals that were submitted, based on MOS values.

Table III shows several pairs of MOS values for H.264/AVC High Profile and the proposed codec for the five 1080p sequences. Each pair has been selected to give similar MOS values, to verify the bit-rate reduction that the proposed codec provides for similar subjective visual quality. As shown in the table, the proposed codec operating at an average of 2.4Mbit/s provides similar visual quality to H.264/AVC High Profile operating at an average of 6.4Mbit/s. This corresponds to approximately 60% bit-rate savings, indicating that the proposed codec provides even greater improvement in subjective visual quality than would be expected from objective metrics such as PSNR.

V. CONCLUSION

In this paper, a high-efficiency video compression framework based on flexible unit representation was proposed. The proposed scheme provides a generic quadtree structure while defining the relationship between region splitting, prediction method, and transform sizes with minimal signaling overhead. The coding tools of the proposed codec were also extended in a very consistent way according to the flexible unit representation. For example, the number of directions in the directional intra prediction scheme was extended to arbitrary angles and the supported transform size was extended up to 64×64. By combining other novel coding tools designed under the proposed unit representation, the proposed codec showed about 40% performance improvement over the state-of-the-art H.264/AVC standard based on objective PSNR measurements. The large-scale subjective testing results show that an average of about 60% coding gain can be achieved for the 1080p sequences. The video codec described in this paper was a candidate in the competitive phase of the HEVC standardization work and it has been partially adopted into the first standardization model of the collaborative phase of the HEVC effort.

ACKNOWLEDGMENT

The authors would like to thank their colleagues including, H. K. Jung, J. C. Lee, S. R. Lee, D. Y. Kim, K. H. Lee, Y. M. Sohn, Y. Y. Lee, B. K. Lee, S. M. Park, H. S. Song, J. B. Choi, B. D. Choi, C. H. Lee, S. S. Jung, and D. S. Cho, for their invaluable contributions and continuous comments on this research.

REFERENCES

[1] ISO/IEC JTC1/SC29/WG11, "Final draft international standard of joint video specification," in ITU-T Recommendation H.264, ISO/IEC 14496-10, Mar. 2003.

[2] ——, "Joint call for proposals on video compression technology," in *91st MPEG meeting*, no. N11113, Kyoto, Japan, Apr. 2010.

[3] V. Baroncini, J. R. Ohm, and G. Sullivan, "Report of subjective test results of responses to the joint call for proposals (cfp) on video coding technology for high efficiency video coding (hevc)," in *1st JCT-VC meeting*, no. JCTVC-A204, Apr. 2010.

[4] 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," in *1st JCT-VC meeting*, no. JCTVC-A124, Apr. 2010.

[5] A. Puri, H. M. Hang, and D. Schilling, "An efficient block-matching algorithm for motion-compensated coding," in *Proc. ICASSP*, Aug. 1997.

[6] M. H. Chan, Y. B. Yu, and A. G. Constantinides, "Variable size block matching motion compensation with applications to video coding," in *Communications, Speech and Vision, IEE Proceedings I*, Aug. 1990.

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

[8] 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.*, no. 5, pp. 808 – 822, Aug. 1999.

[9] ISO/IEC JTC1/SC29/WG11, "Information technology – coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s: Video," in *ISO/IEC 11172-2*, Aug. 1993.

[10] ——, "Information technology – generic coding of moving pictures and associated audio information: Video," in *ISO/IEC 13818-2*, 1996.

[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," in *85th MPEG meeting*, no. M15580, Hannover, Germany, 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," in *86th MPEG meeting*, no. M15899, Busan, Korea, Oct. 2008.

[13] P. Chen, Y. Ye, and M. Karczewicz, "Video coding using extended block sizes," in *36th VCEG meeting*, no. VCEG-AJ23, San Diego, USA, Oct. 2008.

[14] T. Yoshino, S. Naito, and S. Sakazawa, "Preliminary response for draft call for evidence on high performance video coding," in *87th MPEG meeting*, no. M16082, Lausanne, Switzerland, Feb. 2009.

[15] S. Sekiguchi and S. Yamagishi, "On coding efficiency with extended block size for uhdtv," in *87th MCEG meeting*, no. M16019, Lausanne, Switzerland, Feb. 2009.

[16] S. Naito, A. Matsumura, and A. Koike, "Efficient coding scheme for super high definition video based on extending h.264 high profile," in *Proc. VCIP*, Jan. 2006.

[17] S. Ma and C.-C. J. Kuo, "High-definition video coding with super-macroblocks," in *Proc. VCIP*, Jan. 2007.

[18] T. Chujoh, N. Wada, and G. Yasuda, "Quadtree-based adaptive loop filter," in *ITU-T Study Group 16*, no. VCEG-C181, Jan. 2009.

[19] S. H. Lee and N. I. Cho, "Intra prediction method based on the linear relationship between the channels for yuv 4:2:0 intra coding," in *Proc. ICIP*, 2009.

[20] T. K. Tan, C. S. Boon, and Y. Sujuki, "Intra prediction by template matching," in *Proc. ICIP*, 2006.

[21] J. Balle and M. Wien, "Extended texture prediction for h.264/avc intra coding," in *Proc. ICIP*, 2007.

[22] T. Wedi, "Hybrid video coding based on high-resolution displacement vectors," in *Proc. VCIP*, Jan. 2001.

[23] T. Wedi and H. G. Musmann, "Motion- and aliasing-compensated prediction for hybrid video coding," *IEEE Trans. Circuits Syst. Video Technol.*, no. 7, pp. 577–586, Jul. 2003.

[24] J. Jung and G. Laroche, "Competition-based scheme for motion vector selection and coding," in *29th VCEG meeting*, no. VCEG-AC06, Klagenfurt, Austria, Jul. 2006.

[25] S.-D. Kim and J. B. Ra, "An efficient motion vector coding scheme based on minimum bitrate prediction," *IEEE Trans. Image Process.*, no. 8, pp. 1117–1120, Aug. 1999.

[26] J. Dong, K. N. Ngan, C. K. Fong, and W. K. Cham, "2-d order-16 integer transforms for hd video coding," *IEEE Trans. Circuits Syst. Video Technol.*, no. 10, pp. 1462 –1474, Oct. 2009.

[27] W.-H.Chen, C. H. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, no. 25, pp. 1004–1009, Sep. 1977.

[28] J. Liang and T. D. Tran, "Fast multiplierless approximations of the dct with the lifting scheme," *IEEE Trans. Signal Process.*, no. 12, pp. 3032–3044, Dec. 2001.

[29] Y. Ye and M. Karczewicz, "Improved intra coding," in *ITU-T Q.6/SG16, Contribution C257*, Geneva, Switzerland, Jun. 2007.

[30] ——, "Improved intra coding," in *VCEG Contribution VCEG-AG11*, no. 20, Shenzhen, China, Oct. 2007.

[31] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods*. London: Methuen, 1964.

[32] W. H. Payne, J. R. Rabung, and T. P. Bogyo, "Coding the lehmer pseudo-random number generator," *Comm ACM*, no. 2, pp. 85 – 86, Feb. 1969.

[33] T. Chujoh and R. Noda, "Internal bit depth increase for coding efficiency," in *31st VCEG meeting*, no. VCEG-AE13, Marrakech, MA, Jan. 2007.

[34] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, Jul. 2003.

[35] ISO/IEC JTC1/SC29/WG1, "Coding of still pictures," in ISO/IEC IS 10918-1, ITU-T Recommendation T.81, Sep. 1993.

[36] ITU-T VCEG, "Key technology area (KTA) software," in *http://iphome.hhi.de/suehring/tml/download/KTA/*, 2010.

[37] T. K. Tan, G. Sullivan, and T. Wedi, "Recommended simulation common condition for coding efficiency experiments revisio 1," in *VCEG Contribution VCEG-AJ10*, San Diego, USA, Jul. 2008.