Overview of the High Efficiency Video Coding (HEVC) Standard

Gary J. Sullivan, *Fellow IEEE*, Jens-Rainer Ohm, *Member IEEE,* Woo-Jin Han, *Member IEEE,* and Thomas Wiegand, *Fellow IEEE*

Abstract— **High-Efficiency Video Coding (HEVC) is currently being prepared as the newest video coding standard of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). The main goal of the HEVC standardization effort is to enable significantly improved compression performance relative to existing standards – in the range of 50% bit rate reduction for equal perceptual video quality. This article provides an overview of the technical features and characteristics of the HEVC standard.**

Index Terms—**Video compression, Standards, HEVC, JCT-VC, MPEG, VCEG, H.264, MPEG-4, AVC.**

I. INTRODUCTION

HEVC, the High Efficiency Video Coding standard, is the most recent joint video project of the ITU-T VCEG and ISO/IEC MPEG standardization organizations, working together in a partnership known as the Joint Collaborative Team on Video Coding (JCT-VC) [1]. The first edition of the HEVC standard is expected to be finalized in January 2013, resulting in an aligned text that will be published by both ITU-T and ISO/IEC. Additional work is planned to extend the standard to support several additional application scenarios including professional uses with enhanced precision and color format support, scalable video coding, and 3D / stereo / multiview video coding. In ISO/IEC, the HEVC standard will become MPEG-H Part 2 (ISO/IEC 23008-2) and in ITU-T it is likely to become ITU-T Recommendation H.265.

Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 [2] and H.263 [3], ISO/IEC produced MPEG-1 [4] and MPEG-4 Visual [5], and the two organizations jointly produced the H.262/MPEG-2 Video [6] and H.264/MPEG-4 AVC [7] standards. The two standards that were jointly produced have had a particularly strong impact and have found their way into a wide variety of products that are increasingly prevalent in our daily lives. Throughout this evolution, continued efforts have been made to maximize compression capability and improve other characteristics such as data loss robustness, while considering the computational resources that were practical for use in products at the time of anticipated deployment of each standard.

The major video coding standard directly preceding the HEVC project was H.264/MPEG-4 Advanced Video Coding (AVC), which was initially developed during 1999–2003, and then was extended in several important ways during 2003–2009. H.264/MPEG-4 AVC was an enabling technology for digital video in almost every area that was not previously covered by H.262/MPEG-2 Video, and has substantially displaced the older standard within its existing application domain. It is widely used for many applications, including broadcast of high definition (HD) TV signals over satellite, cable, and terrestrial transmission systems, video content acquisition and editing systems, camcorders, security applications, Internet and mobile network video, Blu-ray discs, and real-time conversational applications such as video chat, video conferencing, and telepresence systems.

However, an increasing diversity of services, the growing popularity of HD video, and the emergence of beyond-HD formats (e.g. 4k×2k or 8k×4k resolution) are creating even stronger needs for coding efficiency superior to H.264/MPEG-4 AVC's capabilities. The need is even stronger when higher resolution is accompanied by stereo or multi-view capture and display. Moreover, the traffic caused by video applications targeting mobile devices and tablet-PCs, as well as the transmission needs for video on demand services, are imposing severe challenges on today's networks. An increased desire for higher quality and resolutions is also arising in mobile applications.

HEVC has been designed to address essentially all existing applications of H.264/MPEG-4 AVC and to particularly focus on two key issues: increased video resolution and increased use of parallel processing architectures. The syntax of HEVC is generic and should also be generally suited for other applications that are not specifically mentioned above.

As has been the case for all past ITU-T and ISO/IEC video coding standards, in HEVC only the bitstream structure and syntax is standardized, as well as constraints on the bitstream and its mapping for the generation of decoded pictures. The mapping is given by defining the semantic meaning of syntax elements and a decoding process such that every decoder conforming to the standard will produce the same output when

Manuscript received May 25, 2012.

G. J. Sullivan is with Microsoft Corporation, Redmond, WA 98052 USA (e-mail: garysull@microsoft.com).

J.-R. Ohm is with the Institute of Communications Engineering, RWTH Aachen University, Aachen 52056, Germany (e-mail: ohm@ient.rwth-aachen.de).

W.-J. Han is with the Dept. of Software Development and Management, Gachon University, Seongnam, 461-701, Korea (corresponding author to provide phone: +82-31-750-8668; fax: +82-31-750-5662; e-mail: hurumi@gmail.com).

T. Wiegand is jointly affiliated with the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, Einsteinufer 37, and the Berlin Institute of Technology, Einsteinufer 35, both in 10587 Berlin, Germany (e-mail: twiegand@ieee.org)

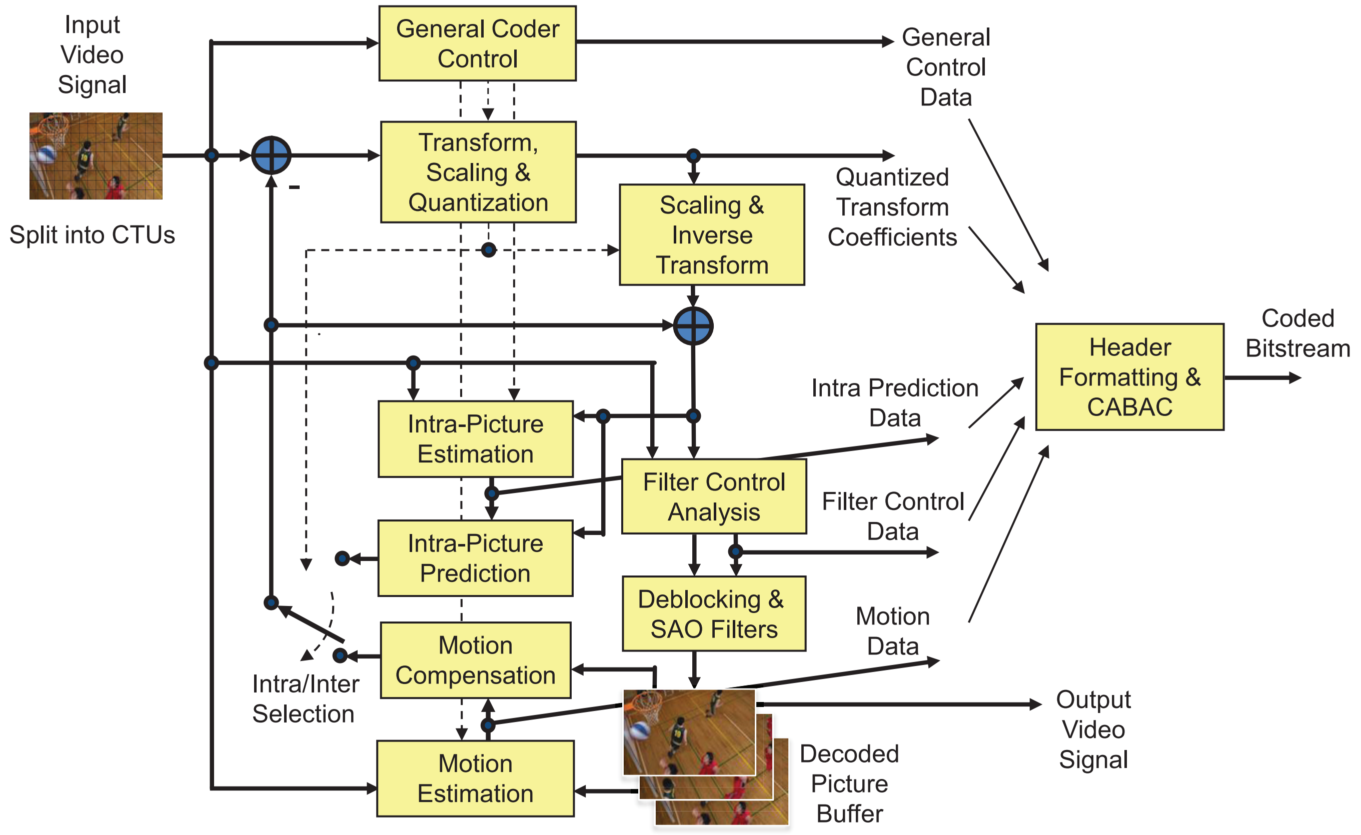


Fig. 1. Typical HEVC video encoder.

given a bitstream that conforms to the constraints of the standard. This limitation of the scope of the standard permits maximal freedom to optimize implementations in a manner appropriate to specific applications (balancing compression quality, implementation cost, time to market, etc.). However, it provides no guarantees of end-to-end reproduction quality, as it allows even crude encoding techniques to be considered conforming.

To assist the industry community in learning how to use the standard, the standardization effort not only includes the development of a text specification document, but also reference software source code as an example of how HEVC video can be encoded and decoded. The draft reference software has been used as a research tool for the internal work of the committee during the design of the standard, and can also be used as a general research tool and as the basis of products. A standard test data suite is also being developed for testing conformance to the standard.

This paper is organized as follows. Section II highlights some key features of the HEVC coding design. Section III explains the high-level syntax and the overall structure of HEVC coded video data. The HEVC video coding technology is then described in greater detail in Section IV. Section V explains the profile, tier and level design of HEVC, including its Main profile in particular. Since writing an overview of a technology as substantial as HEVC involves a substantial amount of summarization, the reader is referred to [1] for any omitted details. The history of the HEVC standardization effort is discussed in Section VI.

II. HEVC CODING DESIGN AND FEATURE HIGHLIGHTS

The HEVC standard is designed to achieve multiple goals: coding efficiency, transport system integration and data loss resilience, as well as implementability using parallel processing architectures. The following sub-sections describe at a glance the key elements of the design by which these goals are achieved, and the typical encoder operation which would generate a valid bitstream. (More details about the associated syntax and decoding process of the different elements are provided in sections III and IV.)

A. Video coding layer

The video coding layer of HEVC employs the same "hybrid" approach (inter-/intra-picture prediction and 2D transform coding) used in all video compression standards since H.261. Fig. 1 depicts the block diagram of a hybrid video encoder, which could create a bitstream conforming to the HEVC standard.

An encoding algorithm producing an HEVC compliant bitstream would typically proceed as follows. Each picture is split into block-shaped regions, with the exact block partitioning being conveyed to the decoder. The first picture of a video sequence (and the first picture at each "clean" random access point into a video sequence) is coded using only intra-picture prediction (which uses some prediction of data spatially from region-to-region within the same picture but has no dependence on other pictures). For all remaining pictures of a sequence or between random access points, inter-picture temporally-predictive coding modes are typically used for most blocks. The encoding process for inter-picture prediction consists of choosing motion data comprising the selected reference picture and motion vector (MV) to be applied for predicting the samples of each block. The encoder and decoder generate identical inter prediction signals by applying motion compensation (MC) using the MV and mode decision data, which are transmitted as side information.

The residual signal of the intra or inter prediction, which is the difference between the original block and its prediction, is transformed by a linear spatial transform. The transform coefficients are then scaled, quantized, entropy coded, and transmitted together with the prediction information.

The encoder duplicates the decoder processing loop such that both will generate identical predictions for subsequent data. Therefore, the quantized transform coefficients are constructed by inverse scaling and are then inverse transformed to duplicate the decoded approximation of the residual signal. The residual is then added to the prediction, and the result of that addition may then be fed into one or two loop filters to smooth out artifacts induced by the block-wise processing and quantization. The final picture representation (which is a duplicate of the output of the decoder) is stored in a *decoded picture buffer* to be used for the prediction of subsequent pictures. In general, the order of the encoding or decoding processing of pictures often differs from the order in which they arrive from the source; necessitating a distinction between the *decoding order* (a.k.a. *bitstream order*) and the *output order* (a.k.a. *display order*) for a decoder.

Video material to be encoded by HEVC is generally expected to be input as *progressive scan* imagery (either due to the source video originating in that format or resulting from de-interlacing prior to encoding). No explicit coding features are present in the HEVC design to support the use of interlaced scanning, as interlaced scanning is no longer used for displays and is becoming substantially less common for distribution. However, metadata syntax has been provided in HEVC to allow an encoder to indicate that interlace-scanned video has been sent by coding each field (i.e. the even or odd numbered lines of each video frame) of interlaced video as a separate picture or that it has been sent by coding each interlaced frame as an HEVC coded picture. This provides an efficient method of coding interlaced video without burdening decoders with a need to support a special decoding process for it.

In the following, the various features involved in hybrid video coding using HEVC are highlighted:

- **Coding Tree Units and Coding Tree Block structure:** The core of the coding layer in previous standards was the *macroblock*, containing a 16×16 block of luma samples and, in the usual case of 4:2:0 color sampling, two corresponding 8×8 blocks of chroma samples; whereas the analogous structure in HEVC is the *coding tree unit* (CTU), which has a size selected by the encoder and can be larger than a traditional macroblock. The CTU consists of a luma *coding tree block* (CTB) and the corresponding chroma CTBs and syntax elements. The size $L \times L$ of a luma CTB can be chosen as L = 16, 32, or 64 samples, with the larger sizes typically enabling better compression. HEVC then supports a partitioning of the CTBs into smaller blocks using a tree structure and quadtree-like signaling [8].
- **Coding Units and Coding Blocks:** The quadtree syntax of the CTU specifies the size and positions of its luma and chroma *coding blocks* (CBs). The root of the quadtree is associated with the CTU. Hence, the size of the luma CTB is the largest supported size for a luma CB. The splitting of a CTU into luma and chroma CBs is signaled jointly. One luma CB and ordinarily two chroma CBs, together with associated syntax, form a Coding Unit (CU). A CTB may contain only one CU or may be split to form multiple CUs, and each CU has an associated partitioning into prediction units (PUs) and a tree of transform units (TUs).
- **Prediction Units and Prediction Blocks:** The decision whether to code a picture area using inter-picture or intra-picture prediction is made at the CU level. A *prediction unit* (PU) partitioning structure has its root at the CU level. Depending on the basic prediction type decision, the luma and chroma CBs can then be further split in size and predicted from luma and chroma *prediction blocks* (PBs). HEVC supports variable PB sizes from 64×64 down to 4×4 samples.
- **Transform Units and Transform Blocks**: The prediction residual is coded using block transforms. A *transform unit* (TU) tree structure has its root at the CU level. The luma CB residual may be identical to the luma *transform block* (TB) or may be further split into smaller luma TBs. The same applies to the chroma TBs. Integer basis functions similar to those of a *discrete cosine transform* (DCT) are defined for the square TB sizes 4×4, 8×8, 16×16, and 32×32. For the 4×4 transform of intra-picture prediction residuals, an integer transform derived from a form of *discrete sine transform* (DST) is alternatively specified.
- **Motion vector signaling:** Advanced motion vector prediction (AMVP) is used, including derivation of several most probable candidates based on data from adjacent PBs and the reference picture. A "merge" mode for MV coding can be also used, allowing the inheritance of MVs from neighboring PBs. Moreover, compared to H.264/MPEG-4 AVC, improved "skipped" and "direct" motion inference are also specified.
- **Motion compensation:** Quarter-sample precision is used for the MVs, and 7-tap or 8-tap filters are used for interpolation of fractional-sample positions (compared to 6-tap filtering of half-sample positions followed by bi-linear interpolation of quarter-sample positions in H.264/MPEG-4 AVC). Similar to H.264/MPEG-4 AVC, multiple reference pictures are used. For each PB, either one or two motion vectors can be transmitted, resulting either in uni-predictive or bi-predictive coding, respectively. As in H.264/MPEG-4 AVC, a scaling and offset operation may be applied to the prediction signal(s) in a manner known as *weighted prediction*.
- **Intra-picture prediction:** The decoded boundary samples of adjacent blocks are used as reference data for spatial prediction in PB regions when inter-picture prediction is not performed. Intra prediction supports 33 directional

modes (compared to 8 such modes in H.264/MPEG-4 AVC), plus planar (surface fitting) and DC (flat) prediction modes. The selected intra prediction modes are encoded by deriving most probable modes (e.g. prediction directions) based on those of previously-decoded neighboring PBs.

- **Quantization control:** As in H.264/MPEG-4 AVC, uniform reconstruction quantization (URQ) is used in HEVC, with quantization scaling matrices supported for the various transform block sizes.
- **Entropy coding:** *Context adaptive binary arithmetic coding* (CABAC) is used for entropy coding. This is similar to the CABAC scheme in H.264/MPEG-4 AVC, but has undergone several improvements to improve its throughput speed (especially for parallel-processing architectures) and its compression performance, and to reduce its context memory requirements.
- **In-loop deblocking filtering (DF):** A *deblocking filter* (DF) similar to the one used in H.264/MPEG-4 AVC is operated in the inter-picture prediction loop. However, the design is simplified in regard to its decision-making and filtering processes, and is made more friendly to parallel processing.
- **Sample adaptive offset (SAO):** A non-linear amplitude mapping is introduced in the inter-picture prediction loop after the deblocking filter. The goal is to better reconstruct the original signal amplitudes by using a look-up table that is described by a few additional parameters that can be determined by histogram analysis at the encoder side.

B. *High-level syntax architecture*

A number of design aspects new to the HEVC standard improve flexibility for operation over a variety of applications and network environments and improve robustness to data losses. However, the high-level syntax architecture used in the H.264/MPEG-4 AVC standard has generally been retained, including the following features:

- **Parameter set structure:** Parameter sets contain information that can be shared for the decoding of several regions of the decoded video. The parameter set structure provides a secure mechanism for conveying data that are essential to the decoding process. The concepts of sequence and picture parameter sets from H.264/MPEG-4 AVC are augmented by a new *video parameter set* (VPS) structure.
- **NAL unit syntax structur**e: Each syntax structure is placed into a logical data packet called a *network abstraction layer* (NAL) unit. Depending on the content of a two-byte NAL unit header, it is possible to readily identify the purpose of the associated payload data.
- **Slices**: A *slice* is a data structure that can be decoded independently from other slices of the same picture, in terms of entropy coding, signal prediction, and residual signal reconstruction. (This describes ordinary slices; an alternative form known as dependent slices is discussed below.) A slice can either be an entire picture or a region of a picture. One of the main purposes of slices is re-synchronization in the event of data losses. In the case of packetized transmission, the maximum number of payload bits within a slice is typically restricted, and the number of CTUs in the slice is often varied to minimize the packetization overhead while keeping the size of each packet within this bound.
- **SEI and VUI metadata**: The syntax includes support for various types of metadata known as *supplemental enhancement information* (SEI), *video usability information* (VUI). Such data provides information about the timing of the video pictures, the proper interpretation of the color space used in the video signal, 3D stereoscopic frame packing information, other "display hint" information, etc.

C. *Parallel decoding syntax and modified slice structuring*

Finally, four new features are introduced in the HEVC standard to enhance parallel processing capability or modify the structuring of slice data for packetization purposes. Each of them may have benefits in particular application contexts, and it is generally up to the implementer of an encoder or decoder to determine whether and how to take advantage of these features.

- **Tiles**: The option to partition a picture into rectangular regions called *tiles* has been specified. The main purpose of tiles is to increase the capability for parallel processing rather than provide error resilience. Tiles are independently-decodable regions of a picture that are encoded with some shared header information. Therefore, they could additionally be used for the purpose of random access to local regions of video pictures. A typical tile configuration of a picture consists of segmenting the picture into rectangular regions with approximately equal numbers of CTUs in each tile. Tiles provide parallelism at a more coarse level (picture/sub-picture) of granularity, and no sophisticated synchronization of threads is necessary for their use.
- **Wavefront parallel processing**: When *wavefront parallel processing* (WPP) is enabled, a slice is divided into rows of CTUs. The first row is processed in an ordinary way; the second row can begin to be processed after only a few decisions have been made in the first row; the third row can begin to be processed after only a few decisions have been made in the second row; etc. The context models of the entropy coder in each row are inferred from those in the preceding row with a small fixed processing lag. WPP provides a form of processing parallelism at a rather fine level of granularity, i.e. within a slice. WPP may often provide better compression performance than tiles (and avoid some visual artifacts that may be induced by tiles).

TABLE I
NAL UNIT TYPES, MEANINGS AND TYPE CLASSES.

Type	Meaning	Class
0	Unspecified	non-VCL
1, 2	non-TSA, non-STSA trailing picture (reference and non-reference)	VCL
3, 4	Temporal sub-layer access (TSA) picture (reference and non-reference)	VCL
5, 6	Step-wise TSA (STSA) picture (reference and non-reference)	VCL
7, 8, 9	Broken link access (BLA) picture (with DLP and TFD, with DLP only, and without LP)	VCL
10, 11	Instantaneous decoder refresh (IDR) picture (with and without DLP)	VCL
12	Clean random access (CRA) picture	VCL
13	Decodable leading picture (DLP)	VCL
14	Tagged for discard (TFD) picture	VCL
15–20	Reserved	VCL
21–24	Reserved	non-VCL
25	Video parameter set (VPS)	non-VCL
26	Sequence parameter set (SPS)	non-VCL
27	Picture parameter set (PPS)	non-VCL
28	Access unit delimiter	non-VCL
29	End of sequence	non-VCL
30	End of bitstream	non-VCL
31	Filler data	non-VCL
32	SEI message	non-VCL
33–47	Reserved	non-VCL
48–63	Unspecified	non-VCL

- **Dependent slices**: A structure called a *dependent slice* allows data associated with a particular wavefront entry point or tile to be carried in a separate NAL unit, and thus potentially makes that data available to a system for fragmented packetization with lower latency than if it were all coded together in one slice. A dependent slice for a wavefront entry point can only be decoded after at least part of the decoding process of another slice has been performed. Dependent slices are mainly useful in low-delay encoding, where other parallel tools might penalize compression performance.

In the following two sections, a more detailed description of the key features is given.

III. HIGH-LEVEL SYNTAX

The high-level syntax of HEVC contains numerous elements that have been inherited from the *network abstraction layer* (NAL) of H.264/MPEG-4 AVC. The NAL provides the ability to map the *video coding layer* (VCL) data that represents the content of the pictures onto various transport layers including RTP/IP, ISO MP4 and H.222.0/MPEG-2 Systems, and provides a framework for packet loss resilience. For general concepts of the NAL design such as NAL units, parameter sets, access units, the byte stream format, and packetized formatting, please refer to [9][10][11].

NAL units are classified into VCL and non-VCL NAL units according to whether they contain coded pictures or other associated data, respectively. In the HEVC standard, several VCL NAL unit types identifying categories of pictures for decoder initialization and random-access purposes are included. TABLE I lists the NAL unit types and their associated meanings and type classes in the HEVC standard.

The following subsections briefly present a description of the new capabilities supported by the high-level syntax.

A. *Random access and bitstream splicing features*

The new design supports special features to enable random access and bitstream splicing. In H.264/MPEG-4 AVC, a bitstream must always start with an *instantaneous decoding refresh* (IDR) access unit. An IDR access unit contains an *intra* picture – a coded picture that can be decoded without decoding any previous pictures in the NAL unit stream, and the presence of an IDR access unit indicates that no subsequent picture in the bitstream will require reference to pictures prior to the intra picture that it contains in order to be decoded. The IDR picture is used within a coding structure known as “closed GOP” (in which “GOP” stands for “group of pictures”).

The new *clean random access* (CRA) picture syntax specifies the use of an intra picture at the location of a *random access point* (RAP), i.e. a location in a bitstream at which a decoder can begin successfully decoding pictures without needing to decode any pictures that appeared earlier in the bitstream, which supports an efficient temporal coding order known as “open GOP” operation. Good support of random access is critical for enabling channel switching, seek operations, and dynamic streaming services. Some pictures that follow a CRA picture in decoding order and precede it in display order may contain inter-picture prediction references to pictures that are not available at the decoder. These non-decodable pictures must therefore be discarded by a decoder that starts its decoding process at a CRA point. For this purpose, such non-decodable pictures are identified as *tagged for discard* (TFD) pictures. The location of splice points from different original coded bitstreams can be indicated by *broken link access* (BLA) pictures. A bitstream splicing operation can be performed by simply changing the NAL unit type of a CRA picture in one bitstream to the value that indicates a BLA picture and concatenating the new bitstream at the position of a RAP picture in the other bitstream. A RAP picture may be an IDR, CRA, or BLA picture, and both CRA and BLA pictures may be followed by TFD pictures in the bitstream (depending on the particular value of the NAL unit type used for a BLA picture). Any TFD pictures associated with a BLA picture must always be discarded by the decoder, as they may contain references to pictures that are not actually present in the bitstream due to a splicing operation. The other type of picture that can follow a RAP picture in decoding order and precede it in output order is the *decodable leading picture* (DLP), which cannot contain references to any pictures that precede the RAP picture in decoding order. TFD and DLP pictures are collectively referred to as *leading pictures* (LPs). Pictures that follow a RAP picture in both decoding order and output order, which are known as *trailing pictures*, cannot contain references to LPs for inter-picture prediction.

B. *Temporal sub-layering support*

Similar to the temporal scalability feature in the H.264/MPEG-4 AVC *scalable video coding* (SVC) extension [12], HEVC specifies a temporal identifier in the NAL unit header, which indicates a level in a hierarchical temporal prediction structure. This was introduced to achieve temporal scalability without the need to parse parts of the bitstream other than the NAL unit header.

Under certain circumstances, the number of decoded temporal sub-layers can be adjusted during the decoding process of one coded video sequence. The location of a point in the bitstream at which switching is possible to begin some higher temporal layers can be indicated by the presence of *temporal sub-layer access* (TSA) pictures and *step-wise TSA* (STSA) pictures. At the location of a TSA picture, it is possible to switch from decoding a lower temporal sub-layer to decoding any higher temporal sub-layer, and at the location of an STSA picture, it is possible to switch from decoding a lower temporal sub-layer to decoding only one particular higher temporal sub-layer (but not the further layers above that, unless they also contain STSA or TSA pictures).

C. *Additional parameter sets*

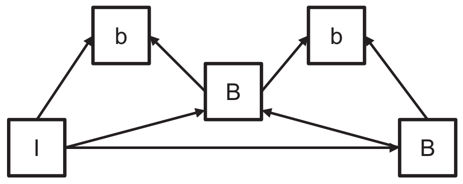
The *video parameter set* (VPS) has been added as metadata to describe the overall characteristics of coded video sequences, including the dependencies between temporal sub-layers. The primary purpose of this is to enable the compatible extensibility of the standard in terms of signaling at the systems layer, e.g. when the base layer of a future extended scalable or multi-view bitstream would need to be decodable by a legacy decoder, but for which additional information about the bitstream structure that is only relevant for the advanced decoder would be ignored.

D. *Reference picture sets and reference picture lists*

For multiple-reference picture management, a particular set of previously-decoded pictures needs to be present in the *decoded picture buffer* (DPB) for the decoding of the remainder of the pictures in the bitstream. To identify these pictures, a list of *picture order count* (POC) identifiers is transmitted in each slice header. The set of retained reference pictures is called the *reference picture set* (RPS). Fig. 2 shows POC values, decoding order and RPS for an example temporal prediction structure.

As in H.264/MPEG-4 AVC, there are two lists that are constructed as lists of pictures in the DPB, and these are called reference picture list 0 and list 1. An index called a *reference picture index* is used to identify a particular picture in one of these lists. For uni-prediction, a picture can be selected from either of these lists. For bi-prediction, two pictures are selected – one from each list. When a list contains only one picture, the reference picture index implicitly has the value 0 and does not need to be transmitted in the bitstream.

The high-level syntax for identifying the RPS and establishing the reference picture lists for inter-picture prediction is more robust to data losses than in the prior H.264/MPEG-4 AVC design, and is more amenable to such operations as random access and "trick mode" operation (e.g. fast-forward, smooth rewind, seeking, adaptive bitstream switching, etc.). A key aspect of this improvement is that the syntax is more explicit, rather than depending on inferences from the stored internal state of the decoding process as it decodes the bitstream picture-by-picture. Moreover, the associated syntax for these aspects of the design is actually simpler than it had been for H.264/MPEG-4 AVC.



POC	0	1	2	3	4
Decoding order	1	4	3	5	2
RPS	-	[0,2]	[0,4]	[2,4]	[0]

Fig. 2. An example of a temporal prediction structure and the POC values, decoding order, and RPS content for each picture.

IV. HEVC VIDEO CODING TECHNIQUES

As in all prior ITU-T and ISO/IEC JTC 1 video coding standards since H.261 [2], the HEVC design follows the classic block-based hybrid video coding approach (as depicted in Fig. 1). The basic source-coding algorithm is a hybrid of inter-picture prediction to exploit temporal statistical dependencies, intra-picture prediction to exploit spatial statistical dependencies, and transform coding of the prediction residual signals to further exploit spatial statistical dependencies. There is no single coding element in the HEVC design that provides the majority of its significant improvement in compression efficiency in relation to prior video coding standards. It is, rather, a plurality of smaller improvements that add up to the significant gain.

A. *Sampled representation of pictures*

For representing color video signals, HEVC typically uses a tri-stimulus *YCbCr* color space with 4:2:0 sampling (although extension to other sampling formats is straightforward, and is planned to be defined in a subsequent version). This separates a color representation into three components called *Y*, *Cb*, and *Cr*. The *Y* component is also called *luma*, and represents brightness. The two *chroma* components *Cb* and *Cr* represent the extent to which the color deviates from gray toward blue and red, respectively. Because the human visual system is more sensitive to luma than chroma, the "4:2:0" sampling structure is typically used, in which each chroma component has one fourth of the number of samples of the luma component (half the number of samples in both the horizontal and vertical dimensions). Each sample for each component is typically represented with 8 or 10 bits of precision, and the 8-bit case is the more typical one. In the remainder of the paper, we focus our attention on the typical use: *YCbCr* components with 4:2:0 sampling and 8 bits per sample for the representation of the encoded input and decoded output video signal.

The video pictures are typically progressively sampled with rectangular picture sizes $W \times H$, where W is the width and H is the height of the picture in terms of luma samples. Each chroma component array, with 4:2:0 sampling, is then $W/2 \times H/2$. Given such a video signal, the HEVC syntax partitions the pictures

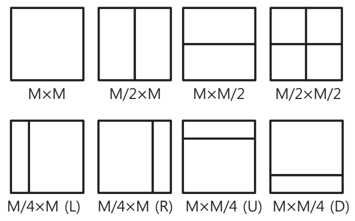


Fig. 3. Modes for splitting a CB into PBs in case of inter prediction. In the case of intra prediction, only M×M and M/2×M/2 (when M/2 = 4) are supported.

further as described below.

B. Division of the picture into coding tree units

A picture is partitioned into coding tree units (CTUs), which each contain luma CTBs and chroma CTBs. A luma CTB covers a rectangular picture area of $L \times L$ samples of the luma component and the corresponding chroma CTBs cover each $L/2 \times L/2$ samples of each of the two chroma components. The value of L may be equal to 16, 32, or 64 as determined by an encoded syntax element specified in the SPS. Compared with the traditional macroblock using a fixed array size of 16×16 luma samples, as used by all previous ITU-T and ISO/IEC JTC 1 video coding standards since H.261 (which was standardized in 1990), HEVC supports variable-size CTBs selected according to the needs of encoders in terms of memory and computational requirements. The support of larger CTBs than in previous standards is particularly beneficial when encoding high-resolution video content. The luma CTB and the two chroma CTBs together with the associated syntax form a coding tree unit (CTU). The CTU is the basic processing unit used in the standard to specify the decoding process.

C. Division of the coding tree block into coding blocks

The blocks specified as luma and chroma CTBs can be directly used as coding blocks (CBs) or can be further partitioned into multiple CBs. The partitioning is achieved using tree structures. The tree partitioning in HEVC is generally applied simultaneously to both luma and chroma, although exceptions apply when certain minimum sizes are reached for chroma.

The CTU contains a quadtree syntax that allows for splitting the CBs to a selected appropriate size based on the signal characteristics of the region that is covered by the CTB. The quadtree splitting process can be iterated until the size for a luma CB reaches a minimum allowed luma CB size that is selected by the encoder using syntax in the SPS and is always 8×8 or larger (in units of luma samples).

The boundaries of the picture are defined in units of the minimum allowed luma CB size. As a result, at the right and bottom edges of the picture, some CTUs may cover regions that are partly outside the boundaries of the picture. This condition is detected by the decoder, and the CTU quadtree is implicitly split as necessary to reduce the CB size to the point where the entire CB will fit into the picture.

D. Prediction blocks and units

The prediction mode for the CU is signaled as being *intra* or *inter*, according to whether it uses intra-picture (spatial) prediction or inter-picture (temporal) prediction. When the prediction mode is signaled as *intra*, the block size at which the intra prediction mode is established is the same as the CB size for all block sizes except for the smallest CB size that is allowed in the bitstream. For the latter case, a flag is present that indicates whether the CB is split into four quadrants that each have their own intra prediction mode. The reason for allowing this split is to enable distinct intra prediction mode selections for blocks as small as 4×4 in size. When the luma intra prediction operates with 4×4 blocks, the chroma intra prediction also uses 4×4 blocks (each covering the same picture region as four 4×4 luma blocks). The actual region size at which the intra prediction operates depends on the residual coding partitioning that is described below.

When the prediction mode is signaled as *inter*, it is specified whether the luma and chroma CBs are split into one, two, or four prediction blocks (PBs). The splitting into four PBs is allowed only when the CB size is equal to the minimum allowed CB size, using an equivalent type of splitting as could otherwise be performed at the CB level of the design rather than at the PB level. When a CB is split into four PBs, each PB covers a quadrant of the CB. When a CB is split into two PBs, various types of this splitting are possible. The partitioning possibilities for inter-predicted CBs are depicted in Fig. 3. The upper partitions illustrate the cases of not splitting the CB of size $M \times M$, of splitting the CB into two PBs of size $M \times M/2$ or $M/2 \times M$, or splitting it into four PBs of size $M/2 \times M/2$. The lower four partition types in Fig. 3 are referred to as *asymmetric motion partitioning*. One PB of the asymmetric partition has the height or width $M/4$ and width or height M, respectively, and the other PB fills the rest of the CB by having a height or width of $3M/4$ and width or height M. Each inter-coded PB is assigned one or two motion vectors and reference picture indices. To minimize worst-case memory bandwidth, PBs of size 4×4 are not allowed for inter prediction, and PBs of sizes 4×8 and 8×4 are restricted to uni-predictive coding. The inter prediction process is further described below.

The luma and chroma PBs together with the associated prediction syntax form the prediction unit (PU).

E. Tree-structured partitioning into transform blocks and units

For residual coding, a CB can be recursively partitioned into transform blocks (TBs). The largest possible TB size is equal to the CB size. The partitioning itself is signaled by a residual quadtree.

Only square partitioning is specified, where a block can be recursively split into quadrants as illustrated in Fig. 4. For a given luma CB of size $M \times M$, a flag signals whether it is split into four blocks of size $M/2 \times M/2$. If further splitting is possible, as signaled by a maximum depth of the residual quadtree indicated in the SPS, each quadrant is assigned a flag that

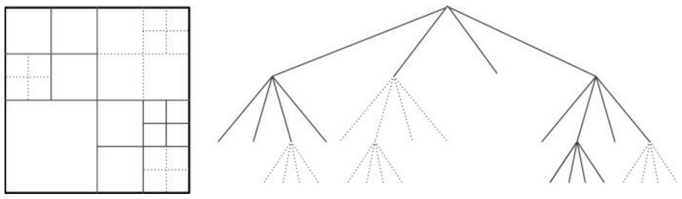


Fig. 4. Subdivision of a CTB into CBs (and Transform Block (TBs). Solid lines indicate CB boundaries and dotted lines indicate TB boundaries. Left: the CTB with its partitioning, right: the corresponding quadtree.

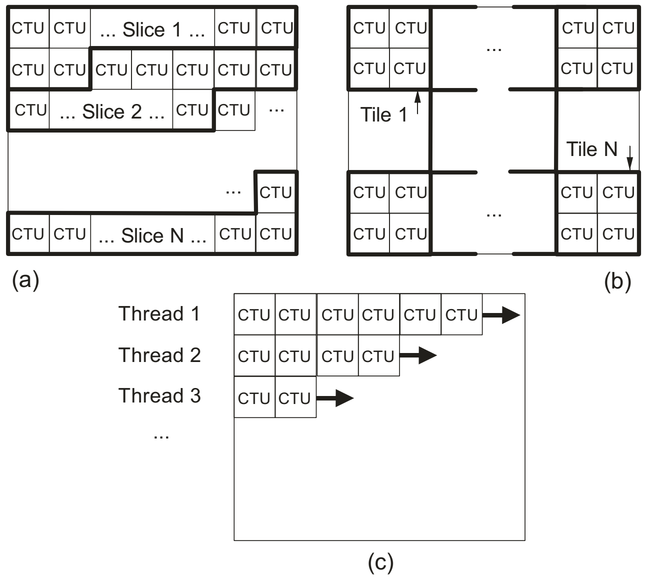


Fig. 5. Subdivision of a picture into (a) slices and (b) tiles, and illustration of wavefront parallel processing (c).

indicates whether it is split into four quadrants. The leaf node blocks resulting from the residual quadtree are the transform blocks that are further processed by transform coding. Since the minimum transform block size is 4×4, splitting that would result in a transform size smaller than this is not supported. In the case of intra-predicted CUs, the decoded samples of the nearest neighboring TBs (within or outside the CB) are used as reference data for intra prediction.

In contrast to previous standards, the HEVC design allows a TB to span across multiple PBs for inter-predicted CUs to maximize the potential coding efficiency benefits of the quadtree structured TB partitioning.

F. *Slices and tiles*

Slices are a sequence of CTUs that are processed in the order of a raster scan. A picture may be split into one or several slices as shown in Fig. 5(a), so that a picture is a collection of one or more slices. Slices are self-contained in the sense that, given the availability of the active sequence and picture parameter sets, their syntax elements can be parsed from the bitstream and the values of the samples in the area of the picture that the slice represents can be correctly decoded (except with regard to the effects of in-loop filtering near the edges of the slice) without the use of any data from other slices in the same picture. This means that prediction within the picture (e.g. intra-prediction or prediction of motion vectors) is not performed across slice boundaries. Some information from other slices may, however, be needed to apply the in-loop filtering across slice boundaries. Each slice can be coded using different coding types as follows:

- **I slice:** A slice in which all CUs of the slice are coded using only intra-picture prediction.
- **P slice:** In addition to the coding types of an I slice, some CUs of a P slice can also be coded using inter-picture prediction with at most *one* motion-compensated prediction signal per PB (a.k.a. uni-prediction). P slices only use reference picture list 0.
- **B slice:** In addition to the coding types available in a P slice, some CUs of the B slice can also be coded using inter-picture prediction with at most *two* motion-compensated prediction signals per PB (a.k.a. bi-prediction). B slices use both reference picture list 0 and list 1.

The main purpose of slices is resynchronization after data losses. Furthermore, slices are often restricted to use a maximum number of bits, e.g. for packetized transmission, and therefore slices may often contain a highly-varying number of CTUs per slice in a manner dependent on the activity in the video scene. In addition to slices, HEVC also defines *tiles*, which are self-contained and independently-decodable rectangular regions of the picture. The main purpose of tiles is to enable the use of parallel processing architectures for encoding and decoding. Multiple tiles may share header information by being contained in the same slice. Alternatively, a single tile may contain multiple slices. A tile consists of a rectangular arranged group of CTUs (typically, but not necessarily, with all of them containing about the same number of CTUs), as shown in Fig. 5(b).

To assist with the granularity of data packetization, *dependent slices* are additionally defined. Finally, with *wavefront parallel processing* (WPP), a slice is divided into rows of CTUs. The decoding of each row can be begun as soon a few decisions that are needed for prediction and adaptation of the entropy coder have been made in the preceding row. This supports parallel processing of rows of CTUs by using several processing threads in the encoder or decoder (or both). An example is shown in Fig. 5(c). For design simplicity, WPP is not allowed to be used in combination with tiles (although these features could, in principle, work properly together).

G. *Intra-picture prediction*

For intra prediction, previously decoded boundary samples from adjacent PUs must be used. Directional prediction with 33 different directional orientations is defined for (square) PU sizes from 4×4 up to 32×32. The possible prediction directions are shown in Fig. 6; alternatively, planar prediction (assuming an amplitude surface with a horizontal and vertical slope derived from the boundaries) and DC prediction (a flat surface with a value matching the mean value of boundary) can also be used. For chroma, the horizontal, vertical, planar, and DC prediction modes can be explicitly signaled, or the chroma prediction mode can be indicated to be the same as the luma prediction mode (and, as a special case to avoid redundant signaling, when one of the first four choices is indicated and is the same as the luma prediction mode, the Intra_Angular[34] mode is applied instead).

Each CB can be coded by one of several coding types, depending on the slice type. Similar to H.264/MPEG-4 AVC, intra coding is supported in all slice types. HEVC supports various intra coding methods referred to as Intra_Angular, Intra_Planar and Intra_DC. The following subsections present a brief further explanation of these and several techniques to be applied in common.

1) *Prediction block partitioning*

An intra-coded CB of size $M \times M$ may have one of two types of PB partitions referred to as PART_2N×2N and PART_N×N,

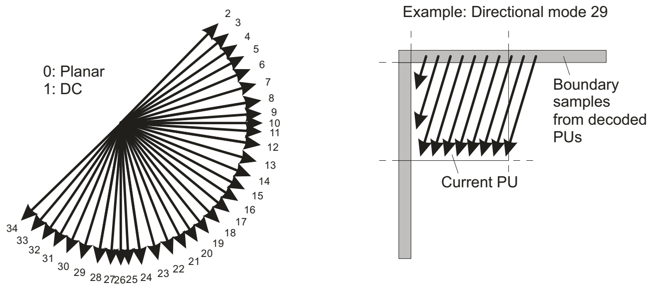


Fig. 6. Modes and directional orientations for intra-picture prediction.

the first of which indicates that the CB is not split and the second indicates that the CB is split into four equal size PBs. (Conceptually, in this notation, $N = M/2$.) However, it is possible to represent the same regions specified by four PBs by using four smaller CBs when the size of the current CB is larger than the minimum CU size. Thus, the HEVC design only allows the partitioning type PART_N×N to be used when the current CB size is equal to the minimum CU size. This means that the PB size is always equal to the CB size when the CB is coded using an intra prediction mode and the CB size is not equal to the minimum CU size.

2) *Intra_Angular prediction*

Spatial-domain intra prediction has previously been successfully used in H.264/MPEG-4 AVC. The intra prediction of HEVC is also based on spatial-domain intra prediction, but is extended significantly – mainly due to the increased size of the PB and an increased number of selectable prediction directions. Compared with 8 directional intra predictions of H.264/MPEG-4 AVC, HEVC supports a total of 33 directional intra predictions denoted as Intra_Angular[k] where k is a mode number from 2 to 34. The angles are intentionally designed to provide denser coverage for near-horizontal and near-vertical angles and coarser coverage for near-diagonal angles to reflect the observed statistical prevalence of the angles and the effectiveness of the signal prediction processing.

When using an Intra_Angular mode, each PB is predicted directionally from spatially neighboring samples which are reconstructed (but not yet filtered by the in-loop filters) before being used for this prediction. For a PB of size $N×N$, a total of $4N+1$ spatially neighboring samples may be used for the prediction, as shown in Fig. 6. When available from preceding decoding operations, samples from lower left PBs can be used for prediction in HEVC in addition to samples from PBs at the left, above, and above-right of the current PB.

The prediction process of the Intra_Angular modes can involve extrapolating samples from the projected reference sample location according to a given directionality. To remove sample-by-sample switching between the reference row and column buffers, all extrapolations in a PB refer to a single reference row or column depending on the mode number. For Intra_Angular[k] with k in the range of 2 to 17, the samples located at the left column are used for the extrapolation, and the samples located at the top row are used when k is in the range of 18 to 34.

To improve the intra prediction accuracy, the projected reference sample location is computed with 1/32 sample accuracy. Bi-linear interpolation is used to obtain the value of the projected reference sample using two closest reference samples located at integer positions.

The prediction process of the Intra_Angular modes is consistent across all block sizes and prediction directions, whereas H.264/MPEG-4 AVC uses different methods for its supported block sizes of 4×4, 8×8 and 16×16. This design consistency is especially desirable since HEVC supports a greater variety of PB sizes and a significantly increased number of prediction directions compared to H.264/MPEG-4 AVC.

3) *Intra_Planar and Intra_DC prediction*

In addition to Intra_Angular prediction which targets regions having strong directional edges, HEVC supports two alternative prediction methods, Intra_Planar, and Intra_DC, for which similar modes were specified in H.264/MPEG-4 AVC. While Intra_DC prediction uses an average value of reference samples for the prediction, average values of two linear predictions using four corner reference samples are used in Intra_Planar prediction to prevent discontinuities along the block boundaries. The Intra_Planar prediction mode is supported at all block sizes in HEVC while H.264/MPEG-4 AVC only supports plane prediction when the luma prediction block size is 16×16, and its plane prediction operates somewhat differently from the planar prediction in HEVC.

4) *Reference sample smoothing*

In HEVC, the reference samples used for the intra prediction are sometimes filtered by a 3-tap [1 2 1]/4 smoothing filter, in a manner similar to what was used for 8×8 intra prediction in H.264/MPEG-4 AVC. However, HEVC applies this smoothing operation more adaptively according to the directionality and the block size. As in H.264/MPEG-4 AVC, the smoothing filter is not applied for 4×4 blocks. For 8×8 blocks, only the diagonal directions, Intra_Angular[k] with k = 2, 18, or 34, use the reference sample smoothing. For 16×16 blocks, the reference samples are filtered for most directions except the near-horizontal and near-vertical directions, k in the range of 9 to 11 and 25 to 27. For 32×32 blocks, all directions except the exactly-horizontal ($k = 10$) and exactly-vertical ($k = 26$) directions use the smoothing filter.

The Intra_Planar mode also uses the smoothing filter when the block size is equal or greater than 8×8, and the smoothing is not used (or useful) for the Intra_DC case.

5) *Boundary value smoothing*

To remove discontinuities along block boundaries, in three modes, Intra_DC (mode 1) and Intra_Angular[k] with k = 10 or 26 (exactly-horizontal or exactly-vertical), the boundary samples inside the prediction block are replaced by filtered values. For Intra_DC mode, both the first row and column of samples in the PB are replaced by the output of a 2-tap [3 1]/4 filter fed by their original value and the adjacent reference sample. In horizontal (Intra_Angular[10]) prediction, the boundary samples of the first column of the PB are modified such that half of the difference between their neighbored reference sample and the top-left reference sample is added. This makes the prediction signal more smooth when large variations in the vertical direction are present. In vertical (Intra_Angular[26]) prediction, the same is applied to the first row of samples.

6) Reference sample substitution

The neighboring reference samples are not available at the slice or tile boundaries. In addition, when a loss-resilience feature known as *constrained intra prediction* is enabled, the neighboring reference samples inside any inter-coded PB are also considered not available in order to avoid letting potentially-corrupted prior decoded picture data propagate errors into the prediction signal. While only Intra_DC prediction mode is allowed for such cases in H.264/MPEG-4 AVC, HEVC allows the use of other intra prediction modes after substituting the non-available reference sample values with the neighboring available reference sample values.

7) Mode coding

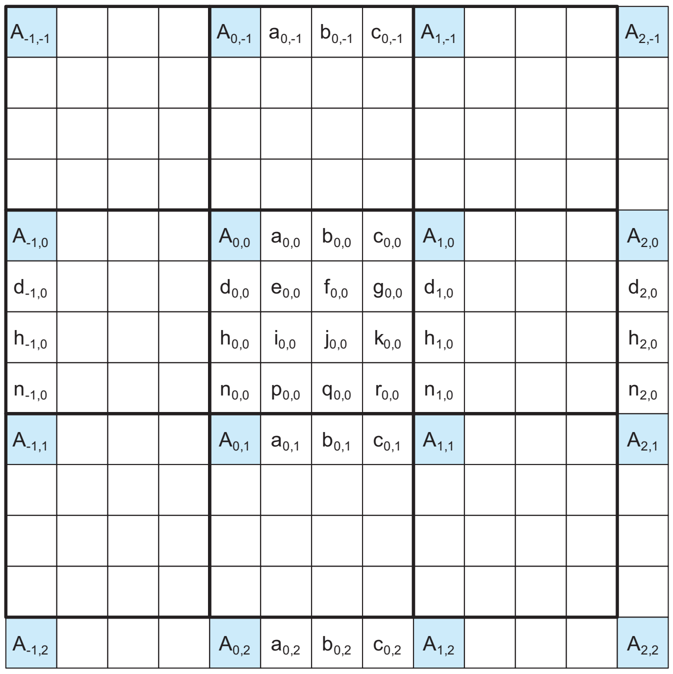
HEVC supports a total of 33 Intra_Angular prediction modes as well as Intra_Planar and Intra_DC prediction modes for luma intra prediction of all block sizes. Due to the increased number of directions, HEVC considers three "most probable modes" (MPMs) when coding the luma intra prediction mode predictively, rather than the one most probable mode considered in H.264/MPEG-4 AVC.

Among the three most probable modes, the first two are initialized by the luma intra prediction modes of the above and left PBs if those PBs are available and are coded using an intra prediction mode. Any unavailable prediction mode is considered to be Intra_DC. The PB above the luma CTB is always considered to be unavailable, in order to avoid the need to store a line buffer of neighboring luma intra prediction modes.

When the first two most probable modes are not equal, the third most probable mode is set equal to Intra_Planar, Intra_DC or Intra_Angular[26] (vertical), according to which of these modes, in this order, is not a duplicate of one the first two modes. When the first two most probable modes are the same, if this first mode has the value Intra_Planar or Intra_DC, the second and third most probable modes are assigned as Intra_Planar, Intra_DC, or Intra_Angular[26], according to which of these modes, in this order, are not duplicates. When the first two most probable modes are the same and the first mode has an Intra_Angular value, the second and third most probable modes are chosen as the two angular prediction modes that are closest to the angle (i.e. the value of k) of the first.

In the case that the current luma intra prediction mode is one of three MPMs, only the MPM index is transmitted to the decoder. Otherwise, the index of the current luma intra prediction mode excluding the three MPMs is transmitted to the decoder by using a 5-bit fixed length code.

For chroma intra prediction, HEVC allows the encoder to select one of five modes: Intra_Planar, Intra_Angular[26] (vertical), Intra_Angular[10] (horizontal), Intra_DC, and Intra_Derived. The Intra_Derived mode specifies that the chroma intra prediction uses the same angular direction as the luma intra prediction. With this scheme, all angular modes specified for luma in HEVC can, in principle, also be used in the chroma intra prediction, and a good trade-off is achieved between prediction accuracy and the signaling overhead. The selected chroma intra prediction mode is coded directly (without using an MPM prediction mechanism).



$A_{-1,-1}$ $A_{0,-1}$ $a_{0,-1}$ $b_{0,-1}$ $c_{0,-1}$ $A_{1,-1}$ $A_{2,-1}$
$A_{-1,0}$ $A_{0,0}$ $a_{0,0}$ $b_{0,0}$ $c_{0,0}$ $A_{1,0}$ $A_{2,0}$
$d_{-1,0}$ $d_{0,0}$ $e_{0,0}$ $f_{0,0}$ $g_{0,0}$ $d_{1,0}$ $d_{2,0}$
$h_{-1,0}$ $h_{0,0}$ $i_{0,0}$ $j_{0,0}$ $k_{0,0}$ $h_{1,0}$ $h_{2,0}$
$n_{-1,0}$ $n_{0,0}$ $p_{0,0}$ $q_{0,0}$ $r_{0,0}$ $n_{1,0}$ $n_{2,0}$
$A_{-1,1}$ $A_{0,1}$ $a_{0,1}$ $b_{0,1}$ $c_{0,1}$ $A_{1,1}$ $A_{2,1}$
$A_{-1,2}$ $A_{0,2}$ $a_{0,2}$ $b_{0,2}$ $c_{0,2}$ $A_{1,2}$ $A_{2,2}$

Fig. 7. Integer and fractional sample positions for luma interpolation.

H. Inter-picture prediction

1) Prediction block partitioning

Compared with intra-coded CBs, HEVC supports more PB partition shapes for inter-coded CBs. The partitioning modes of PART_2N×2N, PART_2N×N and PART_N×2N indicate the cases when the CB is not split, split into two equal-size PBs horizontally and split into two equal-size PBs vertically, respectively. PART_N×N specifies that the CB is split into four equal-size PBs, but this mode is only supported when the CB size is equal to the smallest allowed CB size. Additionally, there are four partitioning types that support splitting the CB into two PBs having different sizes: PART_2N×nU, PART_2N×nD, PART_nL×2N and PART_nR×2N. These types are known as "asymmetric motion partitions".

2) Fractional sample interpolation

The samples of the PB for an inter-coded CB are obtained from those of a corresponding block region in the reference picture identified by a reference picture index, which is at a position displaced by the horizontal and vertical components of the motion vector. Except for the case when the motion vector has an integer value, fractional sample interpolation is used to generate the prediction samples for non-integer sampling positions. As in H.264/MPEG-4 AVC, HEVC supports motion vectors with units of one quarter of the distance between luma samples. For chroma samples, the motion vector accuracy is determined according to the chroma sampling format, which for 4:2:0 sampling results in units of one eighth of the distance between chroma samples.

The fractional sample interpolation for luma samples in HEVC uses separable application of an 8-tap filter for the half-sample positions and a 7-tap filter for the quarter-sample positions. This contrasts with the process used in H.264/MPEG-4 AVC, which applies a two-stage interpolation

TABLE II
FILTER COEFFICIENTS FOR LUMA FRACTIONAL SAMPLE INTERPOLATION

index	−3	−2	−1	0	1	2	3	4
hfilter[i]	−1	4	−11	40	40	−11	4	1
qfilter[i]	−1	4	−10	58	17	−5	1	

process by first generating the values of one or two neighboring samples at half-sample positions using 6-tap filtering, rounding the intermediate results, and then averaging two values at integer or half-sample positions. HEVC instead uses a single consistent separable interpolation process to generate all fractional positions without intermediate rounding operations, which improves precision and simplifies the architecture of the fractional sample interpolation. The interpolation precision is also improved in HEVC by using longer filters – i.e. 7-tap or 8-tap filtering rather than the 6-tap filtering used in H.264/MPEG-4 AVC. Using only 7 taps rather than the 8 used for half-sample positions was sufficient for the quarter-sample interpolation positions since the quarter-sample position are relatively close to integer-sample positions, so the most distant sample in an 8-tap interpolator would effectively be farther away than in the half-sample case (where the relative distances of the integer-sample positions are symmetric). The actual filter tap values of the interpolation filtering kernel were partially derived from DCT basis function equations.

In Fig. 7, the positions labeled with upper-case letters, $A_{i,j}$, represent the available luma samples at integer sample locations, whereas the other positions labeled with lower-case letters represent samples at non-integer sample locations, which need to be generated by interpolation.

The samples labeled $a_{0,j}$, $b_{0,j}$, $c_{0,j}$, $d_{0,0}$, $h_{0,0}$, and $n_{0,0}$ are derived from the samples $A_{i,j}$ by applying the 8-tap filter for half-sample positions and the 7-tap filter for the quarter-sample positions as follows:

$$a_{0,j} = (\textstyle\sum_{i=-3..3} A_{i,j}\ \text{qfilter}[\ i\ ]) >> (B-8)$$
$$b_{0,j} = (\textstyle\sum_{i=-3..4} A_{i,j}\ \text{hfilter}[\ i\ ]) >> (B-8)$$
$$c_{0,j} = (\textstyle\sum_{i=-2..4} A_{i,j}\ \text{qfilter}[\ 1-i\ ]) >> (B-8)$$
$$d_{0,0} = (\textstyle\sum_{j=-3..3} A_{0,j}\ \text{qfilter}[\ j\ ]) >> (B-8)$$
$$h_{0,0} = (\textstyle\sum_{j=-3..4} A_{0,j}\ \text{hfilter}[\ j\ ]) >> (B-8)$$
$$n_{0,0} = (\textstyle\sum_{j=-2..4} A_{0,j}\ \text{qfilter}[\ 1-j\ ]) >> (B-8)$$

where the constant $B \geq 8$ is the bit depth of the reference samples (and typically $B = 8$ for most applications) and the filter coefficient values are given in TABLE II. In these formulas, ">>" denotes an arithmetic right shift operation.

The samples labeled $e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $i_{0,0}$, $j_{0,0}$, $k_{0,0}$, $p_{0,0}$, $q_{0,0}$, and $r_{0,0}$ can be derived by applying the corresponding filters to samples located at vertically adjacent $a_{0,j}$, $b_{0,j}$ and $c_{0,j}$ positions as follows:

$$e_{0,0} = (\textstyle\sum_{j=-3..3} a_{0,j}\ \text{qfilter}[\ j\ ]) >> 6$$
$$f_{0,0} = (\textstyle\sum_{j=-3..3} b_{0,j}\ \text{qfilter}[\ j\ ]) >> 6$$
$$g_{0,0} = (\textstyle\sum_{j=-3..3} c_{0,j}\ \text{qfilter}[\ j\ ]) >> 6$$
$$i_{0,0} = (\textstyle\sum_{j=-3..4} a_{0,j}\ \text{hfilter}[\ j\ ]) >> 6$$
$$j_{0,0} = (\textstyle\sum_{j=-3..4} b_{0,j}\ \text{hfilter}[\ j\ ]) >> 6$$

TABLE III
FILTER COEFFICIENTS FOR CHROMA FRACTIONAL SAMPLE INTERPOLATION

index	−1	0	1	2
filter1[*i*]	−2	58	10	−2
filter2[*i*]	−4	54	16	−2
filter3[*i*]	−6	46	28	−4
filter4[*i*]	−4	36	36	−4

$$k_{0,0} = (\textstyle\sum_{j=-3..4} c_{0,j}\ \text{hfilter}[\ j\ ]) >> 6$$
$$p_{0,0} = (\textstyle\sum_{j=-2..4} a_{0,j}\ \text{qfilter}[\ 1-j\ ]) >> 6$$
$$q_{0,0} = (\textstyle\sum_{j=-2..4} b_{0,j}\ \text{qfilter}[\ 1-j\ ]) >> 6$$
$$r_{0,0} = (\textstyle\sum_{j=-2..4} c_{0,j}\ \text{qfilter}[\ 1-j\ ]) >> 6$$

The interpolation filtering is separable when B is equal to 8, so the same values could be computed in this case by applying the vertical filtering before the horizontal filtering. When implemented appropriately, the motion compensation process of HEVC can be performed using only 16-bit storage elements.

It is at this point in the process that weighted prediction is applied when selected by the encoder. Whereas H.264/MPEG-4 AVC supported both temporally-implicit and explicit weighted prediction, in HEVC only explicit weighted prediction is applied, by scaling and offsetting the prediction with values sent explicitly by the encoder. The bit depth of the prediction is then adjusted to the original bit depth of the reference samples. In the case of uni-prediction, the interpolated (and possibly weighted) prediction value is rounded, right-shifted, and clipped to have the original bit depth. In the case of bi-prediction, the interpolated (and possibly weighted) prediction values from two PBs are added first, and then rounded, right-shifted and clipped.

In H.264/MPEG-4 AVC, up to 3 stages of rounding operations are required to obtain each prediction sample (for samples located at quarter-sample positions. If bi-prediction is used, the total number of rounding operations is then seven in the worst case. In HEVC, at most two rounding operations are needed to obtain each sample located at the quarter-sample positions, thus five rounding operations are sufficient in the worst case when bi-prediction is used. Moreover, in the most common case, where the bit depth B is 8 bits, the total number of rounding operations in the worst case is further reduced to three. Due to the lower number of rounding operations, the accumulated rounding error is decreased and greater flexibility is enabled in regard to the manner of performing the necessary operations in the decoder.

The fractional sample interpolation process for the chroma components is similar to the one for the luma component, except that the number of filter taps is 4 and the fractional accuracy is 1/8 for the usual 4:2:0 chroma format case. HEVC defines a set of 4-tap filters for eighth-sample positions as given in TABLE III for the case of 4:2:0 chroma format. (In H.264/MPEG-4 AVC, only 2-tap filtering was applied.)

Filter coefficient values denoted as filter1[*i*], filter2[*i*], filter3[*i*] and filter4[*i*] with $i = -1..2$ are used for interpolating the 1/8th, 2/8th, 3/8th and 4/8th fractional positions for the chroma samples, respectively. Using symmetry for the

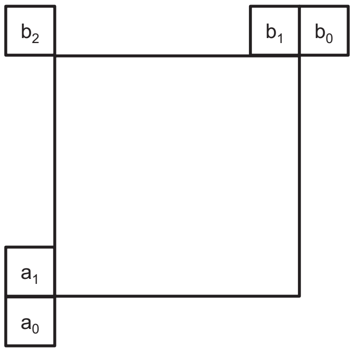


Fig. 8. Positions of spatial candidates of motion information

5/8th, 6/8th and 7/8th fractional positions, the mirrored values of filter3[1−*i*], filter2[1−*i*] and filter1[1−*i*] with $i = -1..2$ are used, respectively.

3) *Merge mode*

Motion information typically consists of the horizontal and vertical motion vector displacement values, one or two reference picture indices, and, in the case of prediction regions in B slices, an identification of which reference picture list is associated with each index. HEVC includes a merge mode to derive the motion information from spatially or temporally neighboring blocks. It is denoted as merge mode since it forms a merged region sharing all motion information.

The merge mode is conceptually similar to the direct and skip modes in H.264/MPEG-4 AVC; however, there are two important differences. Firstly, it transmits index information to select one out of several available candidates, in a manner sometimes referred to as a motion vector "competition" scheme. It also explicitly identifies the reference picture list and reference picture index, whereas the direct mode assumes that these have some pre-defined values.

The set of possible candidates in the merge mode consists of spatial neighbor candidates, a temporal candidate and generated candidates. Fig. 8 shows the positions of five spatial candidates. For each candidate position, the availability is checked according to the order { a_1, b_1, b_0, a_0, b_2 }. If the block located at the position is intra-coded or the position is outside of the current slice or tile, it is considered as unavailable.

After validating the spatial candidates, two kinds of redundancy are removed. If the candidate position for the current PU would refer to the first PU within the same CU, the position is excluded, as the same merge could be achieved by a CU without splitting into prediction partitions. Furthermore, any redundant entries where candidates have exactly the same motion information are also excluded.

For the temporal candidate, the right bottom position just outside of the collocated PU of the reference picture is used if it is available. Otherwise, the center position is used instead. The way to choose the collocated reference picture and the collocated PU is similar with that of prior standards, but HEVC allows more flexibility by transmitting an index to specify which reference picture list is used for the collocated reference picture.

One issue related to the use of the temporal candidate is the amount of the memory to store the motion information of the reference picture. This is addressed by restricting the granularity for storing the temporal motion candidates to only the resolution of a 16×16 luma grid, even when smaller PB structures are used at the corresponding location in the reference picture. Additionally, a PPS-level flag allows the encoder to disable the use of the temporal candidate, which is useful for applications in error-prone transmission.

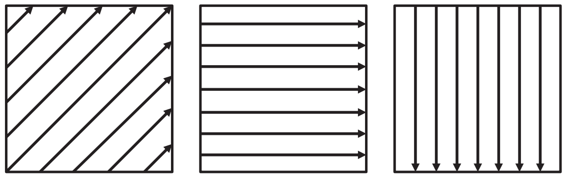


Fig. 9. Three coefficient scanning methods in HEVC; (a) diagonal up-right scan (b) horizontal scan and (c) vertical scan

The maximum number of merge candidates *C* is specified in the slice header. If the number of merge candidates found (including the temporal candidate) is larger than *C*, only the first *C*−1 spatial candidates and the temporal candidate are retained. Otherwise, if the number of merge candidates identified is less than *C*, additional candidates are generated until the number is equal to *C*. This simplifies the parsing and makes it more robust, as the ability to parse the coded data is not dependent of merge candidate availability.

For B slices, additional merge candidates are generated by choosing two existing candidates according to a pre-defined order for reference picture list 0 and list 1. For example, the first generated candidate uses the first merge candidate for list 0 and the second merge candidate for list 1. HEVC specifies a total of twelve pre-defined pairs of two in the following order in the already constructed merge candidate list as (0, 1), (1, 0), (0, 2), (2, 0), (1, 2), (2, 1), (0, 3), (3, 0), (1, 3), (3, 1), (2, 3) and (3, 2). Among them, up to five candidates can be included after removing redundant entries.

When the slice is a P slice or the number of merge candidates is still less than *C*, zero motion vectors associated with reference indices from zero to the number of reference pictures minus one are used to fill any remaining entries in the merge candidate list.

In HEVC, the skip mode is treated as a special case of the merge mode when all coded block flags are equal to zero. In this specific case, only a skip flag and the corresponding merge index are transmitted to the decoder. The B-direct mode of H.264/MPEG-4 AVC is also replaced by the merge mode, since the merge mode allows all motion information to be derived from the spatial and temporal motion information of the neighboring blocks with residual coding.

4) *Motion vector prediction for non-merge mode*

When the inter-coded CB is not coded in the skip or merge modes, the motion vector is differentially coded using a motion vector predictor. Similar to the merge mode, HEVC allows the encoder to choose it among multiple predictor candidates. The difference between the predictor and the actual motion vector, and the index of the candidate are transmitted to the decoder.

Only two spatial motion candidates are chosen according to the availability among five candidates in Fig. 8. The first spatial motion candidate is chosen from the set of left positions {a_0, a_1}

and the second one from the set of above positions {b_0, b_1, b_2} according to their availabilities, while keeping the searching order as indicated in the two sets.

HEVC only allows a much lower number of candidates to be used in the motion vector prediction process for the non-merge case, since the encoder can send a coded difference to change the motion vector. Further, the encoder needs to perform motion estimation, which is one of the most computationally expensive operations in the encoder, and complexity is reduced by allowing less candidates.

When the reference index of the neighboring PU is not equal to that of the current PU, a scaled version of the motion vector is used. The neighboring motion vector is scaled according to the temporal distances between the current picture and the reference pictures indicated by the reference indices of the neighboring PU and the current PU, respectively. When two spatial candidates have the same motion vector components, one redundant spatial candidate is excluded.

When the number of motion vector predictors is not equal to two and the use of temporal MV prediction is not explicitly disabled, the temporal MV prediction candidate is included. This means that the temporal candidate is not used at all when two spatial candidates are available. Finally, a zero motion vector is included repeatedly until the number of motion vector prediction candidates is equal to two, which guarantees that the number of motion vector predictors is two. Thus, only a coded flag is necessary to identify which motion vector prediction is used in the case of non-merge mode.

I. Transform, scaling, and quantization

HEVC uses transform coding of the prediction error residual in a similar manner as in prior standards. The residual block is partitioned into multiple square TBs, as described in section IV.E. Herein, the possible transform block sizes are 4×4, 8×8, 16×16 and 32×32.

1) Core transform

Two-dimensional transforms are computed by applying one-dimensional transforms in both the horizontal and vertical directions. The elements of the core transform matrices were derived by approximating scaled discrete cosine transform (DCT) basis functions, under considerations such as limiting the necessary dynamic range for transform computation and maximizing the precision and closeness to orthogonality when the matrix entries are specified as integer values.

For simplicity, only one integer matrix for the length of 32 points is specified, and sub-sampled versions are used for other sizes. For example, the matrix for the length-16 transform is given as:

$$H = \begin{bmatrix}
64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\
90 & 87 & 80 & 70 & 57 & 43 & 25 & 9 & -9 & -25 & -43 & -57 & -70 & -80 & -87 & 90 \\
89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 & -89 & -75 & -50 & -18 & 18 & 50 & 75 & 89 \\
87 & 57 & 9 & -43 & -80 & -90 & -70 & -25 & 25 & 70 & 90 & 80 & 43 & -9 & -57 & -87 \\
83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 & 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\
80 & 9 & -70 & -87 & -25 & 57 & 90 & 43 & -43 & -90 & -57 & 25 & 87 & 70 & -9 & -80 \\
75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 & -75 & 18 & 89 & 50 & -50 & -89 & -18 & 75 \\
70 & -43 & -87 & 9 & 90 & 25 & -80 & -57 & 57 & 80 & -25 & -90 & -9 & 87 & 43 & -70 \\
64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\
57 & -80 & -25 & 90 & -9 & -87 & 43 & 70 & -70 & -43 & 87 & 9 & -90 & 25 & 80 & -57 \\
50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 & -50 & 89 & -18 & -75 & 75 & 18 & -89 & 50 \\
43 & -90 & 57 & 25 & -87 & 70 & 9 & -80 & 80 & -9 & -70 & 87 & -25 & -57 & 90 & -43 \\
36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 & 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\
25 & -70 & 90 & -80 & 43 & 9 & -57 & 87 & -87 & 57 & -9 & -43 & 80 & -90 & 70 & -25 \\
18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 & -18 & 50 & -75 & 89 & -89 & 75 & -50 & 18 \\
9 & -25 & 43 & -57 & 70 & -80 & 87 & -90 & 90 & -87 & 80 & -70 & 57 & -43 & 25 & -9
\end{bmatrix}.$$

The matrices for the length-8 and length-4 transforms can be derived by using the first 8 entries of rows 0, 2, 4, ..., and using the first four entries of rows 0, 4, 8, ..., respectively. Although the standard specifies the transform simply in terms of the value of a matrix, the values of the entries in the matrix were selected to have key symmetry properties that enable fast "partially-factored" implementations with far fewer mathematical operations than an ordinary matrix multiplication, and the larger transforms can be constructed by using the smaller transforms as building blocks.

Due to the increased size of the supported transforms, limiting the dynamic range of the intermediate results from the first stage of the transformation is quite important. HEVC explicitly inserts a 7-bit right shift and 16-bit clipping operation after the first one-dimensional inverse transform stage of the transform (the vertical inverse transform stage), to ensure that all intermediate values can be stored in 16-bit memory (for 8-bit video decoding).

2) Mode-dependent alternative transform

For the transform block size of 4×4, an alternative integer transform derived from a discrete sine transform (DST) is applied to the luma residual blocks for intra prediction modes, with the transform matrix

$$H = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix}.$$

With the basis functions of the DST, the property is better modeled that the residual amplitudes tend to increase as the distance from the boundary samples which are used for prediction becomes larger. In terms of complexity, the 4×4 DST-style transform is not much more computationally demanding than the 4×4 DCT-style transform, and it provides approximately 1% bit rate reduction in intra-predictive coding.

The usage of the DST type of transform is restricted to only 4×4 luma transform blocks, since for other cases the additional coding efficiency improvement for including the additional transform type was found to be marginal.

3) Scaling and quantization

Since the rows of the transform matrix are close approximations of values of uniformly-scaled basis functions of the orthonormal DCT, the pre-scaling operation that is incorporated in the dequantization of H.264/MPEG-4 AVC is not needed in HEVC. This avoidance of frequency-specific basis function scaling is useful in reducing the intermediate memory size – especially when considering that the size of the transform can be as large as 32×32.

For quantization, HEVC uses essentially the same uniform-reconstruction quantization (URQ) scheme controlled by a *quantization parameter* (QP) as in H.264/MPEG-4 AVC. The range of the QP values is defined from 0 to 51, and an increase by 6 doubles the quantization step size, such that the mapping of QP values to step sizes is approximately logarithmic. Quantization scaling matrices are also supported.

To reduce the memory needed to store frequency-specific scaling values, only quantization matrices of sizes 4×4 and 8×8 are used. For the larger transformations of 16×16 and 32×32 sizes, an 8×8 scaling matrix is sent and is applied by sharing values within 2×2 and 4×4 coefficient groups in frequency sub-spaces – except for values at DC positions, for which distinct values are sent and applied.

J. Entropy coding

HEVC specifies only one entropy coding method, context adaptive binary arithmetic coding (CABAC) [13] rather than two as in H.264/MPEG-4 AVC. The core algorithm of CABAC is unchanged, and the following subsections present several aspects of how it is used in the HEVC design.

1) Context modeling

Appropriate selection of context modeling is known to be a key factor to improve the efficiency of CABAC coding. In HEVC, the splitting depth of the coding tree or transform tree is exploited to derive the context model indices of various syntax elements in addition to the spatially neighboring ones used in H.264/AVC. For example, the syntax element *skip_flag* specifying whether the CB is coded as inter-skipped and the syntax element *split_coding_unit_flag* specifying whether the CB is further split are coded by using context models based on the spatially neighboring information. The syntax element *split_transform_flag* specifying whether the TB is further split and three syntax elements specifying non-zero transform coefficients for each color component, *cbf_luma*, *cbf_cb* and *cbf_cr* are coded based on the splitting depth of the transform tree. Although the number of contexts used in HEVC is substantially less than in H.264/MPEG-4 AVC, the entropy coding design actually provides better compression than would a straightforward extension of the H.264/MPEG-4 AVC scheme. Moreover, more extensive use is made in HEVC of the bypass-mode of CABAC operation, to increase throughput by reducing the amount of data that needs to be coded using CABAC contexts. Dependencies between coded data are also carefully considered to enable further throughput maximization.

2) Adaptive coefficient scanning

Coefficient scanning is performed in 4×4 sub-blocks for all TB sizes (i.e. using only one coefficient region for the 4×4 TB size, and using multiple 4×4 coefficient regions within larger transform blocks). Three coefficient scanning methods, diagonal up-right, horizontal and vertical scans as shown in Fig. 9, are selected implicitly for coding the intra-coded transform coefficients of 4×4 and 8×8 TB sizes. The selection of the coefficient scanning order depends on the directionalities of the intra prediction. When the direction is close to the horizontal direction, the vertical scan is used and the horizontal scan is used when the direction is close to the vertical direction. For other prediction directions, the diagonal up-right scan is used.

For the transform coefficients in inter prediction modes of all block sizes and for the transform coefficients of 16×16 or 32×32 intra prediction, the 4×4 diagonal up-right scan is exclusively applied to sub-blocks of transform coefficients.

3) Coefficient coding

Similar to H.264/MPEG-4 AVC, HEVC transmits the position of the last non-zero transform coefficient, a significance map, sign bits and levels for the transform coefficients. However, various changes for each part have been made, especially for better handling of the significantly increased size of the TBs.

Firstly, the *i* and *j* coordinate positions of the last non-zero coefficient are coded for the TB before sending the significance maps of 4x4 sub-blocks that indicate which other transform coefficients have non-zero values, rather than sending a series of last-coefficient identification flags which are interleaved with the significance map as done in H.264/MPEG-4 AVC.

The significance map is derived for significance groups relating to the fixed size 4×4 sub-blocks. For all groups having at least one coefficient preceding the last coefficient position, a significant group flag specifying a non-zero coefficient group is transmitted, followed by coefficient significance flags for each coefficient prior to the indicated position of the last significant coefficient. The context models for the significant coefficient flags are dependent on the coefficient position as well as the values of the right and the bottom significant group flags.

A method denoted as *sign data hiding* is used for further compression improvement. The sign bits are coded conditionally based on the number and positions of coded coefficients. If there are at least two non-zero coefficients in a 4×4 sub-block and the difference between the scan positions of the first and the last non-zero coefficients is greater than 3, the sign bit of the first non-zero coefficient is inferred from the parity of the sum of the coefficient amplitudes, otherwise, the sign bit is coded normally. At the encoder side, this can be implemented by selecting one coefficient with an amplitude close to the boundary of a quantization interval to be forced to use the adjacent quantization interval in cases where the parity would not otherwise indicate the correct sign of the first coefficient. This allows the sign bit to be encoded at a lower cost (in rate-distortion terms) than if it were coded separately – by giving the encoder the freedom to choose which transform coefficient amplitude can be altered with the lowest rate-distortion cost.

For each position where the corresponding significant coefficient flag is equal to one, two flags specifying whether the level value is greater than one or two are coded, and then the remaining level value is coded depending on those two values.

K. *In-loop filters*

In HEVC, two processing steps, namely a deblocking filter (DBF) followed by a sample adaptive offset (SAO) operation are applied to the reconstructed samples before writing them into the decoded picture buffer in the decoder loop. The DBF is intended to reduce the blocking artifacts due to block-based coding. The DBF is similar to the DBF of the H.264/MPEG-4 AVC standard, whereas SAO is newly introduced in HEVC. While the DBF which is only applied to the samples located at block boundaries, the SAO operation is applied adaptively to all samples satisfying certain conditions, e.g. based on gradient. During the development of HEVC, it had also been considered to operate a third processing step denoted as adaptive loop filter (ALF) behind SAO, which was however not included in the first version of HEVC.

1) *Deblocking filter*

The deblocking filter is applied to all samples adjacent to a PU or TU boundary except the case when the boundary is also a picture boundary, or when deblocking is disabled across slice or tile boundaries (which is an option that can be signaled by the encoder). It should be noted that both PU and TU boundaries should be considered since PU boundaries are not always aligned with TU boundaries in some cases of inter-coded CBs. For the slice and the tile boundaries, several syntax elements specified in the SPS and the slice header are used for allowing the deblocking filter to the samples adjacent to them.

Unlike H.264/MPEG-4 AVC, where the deblocking filter is applied on a 4×4 sample grid basis, HEVC only applies the deblocking filter to the edges which are aligned on an 8×8 sample grid, for both the luma and chroma samples. This restriction is especially helpful to reduce the worst-case computational complexity without noticeable degradation of the visual quality. It also improves parallel-processing operation by preventing cascading interactions between nearby filtering operations.

The strength of the deblocking filter is controlled by the values of several syntax elements similar to the scheme in H.264/MPEG-4 AVC, but only three strengths from 0 to 2 are used rather than five. Given that P and Q are two adjacent blocks with a common 8×8 grid boundary, the filter strength of 2 is assigned when one of the blocks is intra-coded. Otherwise, the filter strength of 1 is assigned if any of the following conditions is satisfied:

- P or Q has at least one non-zero transform coefficient.
- The reference indices of P and Q are not equal.
- The motion vectors of P and Q are not equal.
- The difference between each motion vector component of P and Q is equal or greater than one integer sample.

If none of the above conditions is met, the filter strength of 0 is assigned, which means that the deblocking process is not applied.

According to the filter strength and the average quantization parameter (QP) of P and Q, two thresholds, t_C and β, are determined from pre-defined tables. For luma samples, one of three cases, no filtering, strong filtering, and weak filtering, are chosen according to the experimental rules based on β. Note that this decision is shared across four luma rows or columns using the first and the last rows or columns to reduce the computational complexity.

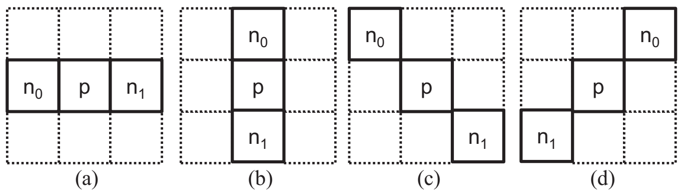


Fig. 10. Four gradient patterns used in SAO. Sample labeled "p" indicates a center sample to be considered. Two samples labeled "n0" and "n1" specify two neighboring samples along the gradient pattern (a) horizontal (sao_eo_class = 0) (b) vertical (sao_eo_class = 1) (c) 135° diagonal (sao_eo_class = 2) (d) 45° (sao_eo_class = 3).

TABLE IV
SAMPLE EDGEIDX CATEGORIES IN SAO EDGE CLASSES

EdgeIdx	Condition	Meaning
0	$p = n_0$ and $p = n_1$	flat area
1	$p < n_0$ and $p < n_1$	local min
2	$p < n_0$ and $p = n_1$ or $p < n_1$ and $p = n_0$	edge
3	$p > n_0$ and $p = n_1$ or $p > n_1$ and $p = n_0$	edge
4	$p > n_0$ and $p > n_1$	local max

There are only two cases, no filtering and normal filtering, for chroma samples. Normal filtering is applied only when the filter strength is greater than one. The actual filtering process for the samples is performed based on both values of t_C and β, a detailed description of the strong filter and the weak filter processes can be found in [1].

In HEVC, the processing order of the deblocking filter is defined as horizontal filtering for vertical edges for the entire picture first, followed by vertical filtering for horizontal edges. This specific order enables either multiple horizontal filtering or vertical filtering processes to be applied in parallel threads, and can still be implemented on a CTB by CTB basis with only small processing latency.

2) *Sample adaptive offset (SAO)*

SAO is a process which modifies the samples after the deblocking filter through a look-up table. It is performed on a region basis (adapted per CTB). Depending on the local gradient at the sample position, a certain offset value from a look-up table is added to the sample. A value of the syntax element sao_type_idx equal to 0 indicates that the SAO is not applied to the region, and sao_type_idx equal to 1 or 2 signals the use of band or edge offset types, respectively. In case of sao_type_idx equal to 2, sao_eo_class with values from 1 to 4 signals additionally whether the horizontal, the vertical or one of the two diagonal gradients is used for this purpose. Fig. 10 depicts four gradient patterns used in SAO.

In the edge offset mode, once a specific sao_eo_class is chosen for a CTB, all samples in the CTB are classified into five EdgeIdx categories by comparing the sample value located at p with two neighboring sample values located at n_0 and n_1 as shown in TABLE IV. Note that this classification is done for each sample at both encoder and decoder, so no additional signaling for the classification is required. For sample categories from 1 to 4, a certain offset value is specified for

TABLE V
LEVEL LIMITS FOR THE MAIN PROFILE

Level	Max luma picture size (samples)	Max luma sample rate (samples/sec)	Main Tier Max bit rate (1000 bits/s)	High Tier Max bit rate (1000 bits/s)	Min comp. ratio
1	36,864	552,960	128	–	2
2	122,880	3,686,400	1,500	–	2
2.1	245,760	7,372,800	3,000	–	2
3	552,960	16,588,800	6,000	–	2
3.1	983,040	33,177,600	10,000	–	2
4	2,228,224	66,846,720	12,000	30,000	4
4.1	2,228,224	133,693,440	20,000	50,000	4
5	8,912,896	267,386,880	25,000	100,000	6
5.1	8,912,896	534,773,760	40,000	160,000	8
5.2	8,912,896	1,069,547,520	60,000	240,000	8
6	33,423,360	1,069,547,520	60,000	240,000	8
6.1	33,423,360	2,005,401,600	120,000	480,000	8
6.2	33,423,360	4,010,803,200	240,000	800,000	6

each category, denoted as the edge offset, which is added to the sample value, thus a total of four edge offsets are estimated by the encoder and transmitted to the decoder for each CTB. To reduce the bit overhead for transmitting the four edge offsets which are originally signed values, HEVC allows only positive offset values for the categories 1 and 2 and only negative offset values for the categories 3 and 4, since these cover most relevant cases.

In the "banding type" operation specified by sao_type_idx equal to 1 the selected offset value directly depends on the sample amplitude. The whole relevant sample amplitude range is split into 32 bands and the sample values belonging to four consecutive bands are modified by adding the values denoted as band offsets. The main reason of the use of four consecutive bands lies in the fact that flat areas where banding artifacts could appear, most sample amplitudes in a CTB tend to be concentrated in only few bands. In addition, this design choice is unified with the edge offset types which also use four offset values.

The look-up table for the best gradient pattern and four corresponding offsets to be used must be determined by the encoder. For band offset use, the starting position of the bands is also determined by the encoder. The parameters must be explicitly encoded or can be inherited from the left CTB (in the latter case signaled by a special merge flag). In summary, SAO is a non-linear operation which allows additional minimization of the reconstruction error in a way that cannot be achieved by linear filters, and particularly is able to enhance edge sharpness. In addition, it was found that SAO is very efficient to suppress pseudo-edges referred to as "banding artifacts", as well as the "ringing artifacts" coming from the quantization errors of high frequency components in the transform domain.

L. *Special coding modes*

HEVC defines three special coding modes, which can be invoked at the CU level or the TU level:

- In I_PCM mode, prediction, transform, quantization and entropy coding are bypassed, and samples are directly represented by a pre-defined number of bits. Its main purpose is to avoid excessive consumption of bits when the signal characteristics are ill-posed and cannot be properly handled by hybrid coding (e.g. noise-like signals).
- In lossless mode, the transform, quantization and other processing that affects the decoded picture (SAO, loop and deblocking filters) are bypassed, and the residual signal from inter or intra prediction is directly fed into the entropy coder (using same neighborhood contexts that would usually be applied to the quantized transform coefficients). This allows mathematically-lossless reconstruction, which is achieved without defining any additional coding tools.
- In transform skipping mode, only the transform is bypassed. This mainly addresses the coding efficiency issue for specific video content such as computer generated images or graphics mixed with camera-view content (e.g. scrolling text). This mode can be applied to inter- and intra-coded TBs of 4×4 size only.

SAO, loop filtering and deblocking filtering is not executed across boundaries between regular CUs on one side and I_PCM or lossless CUs on the other side.

V. PROFILES, TIERS AND LEVELS

A. *Profile, level, and tier concepts*

Profiles, tiers and levels specify conformance points for implementing the standard in an interoperable way across various applications of the standard that have similar functional requirements. A *profile* defines a set of coding tools or algorithms that can be used in generating a conforming bitstream, whereas a *level* places constraints on certain key parameters of the bitstream, corresponding to decoder processing load and memory capabilities. Level restrictions are established in terms of maximum sample rate, maximum picture size, maximum bit rate, minimum compression ratio and capacities of the DPB and the *coded picture buffer* (CPB) that holds compressed data prior to its decoding for data flow management purposes. In the design of HEVC, it was determined that some applications existed that had requirements that differed only in terms of maximum bit rate and CPB capacities. To resolve this issue, two *tiers* were specified for some levels – a "Main" tier for most applications and a "High" tier for use in the most demanding applications.

A decoder conforming to a certain tier and level would generally be capable of decoding all bitstreams that conform to the same tier or the lower tier of that level or any level below it.

The decoders conforming to a specific profile must support all features in that profile. Encoders are not required to make use of any particular set of features supported in a profile, but are required to produce conforming bitstreams, i.e. bitstreams that obey the specified constraints that enable them to be decoded by conforming decoders.

B. *The HEVC Main profile and level definitions*

Currently, only one profile, called the "Main" profile is foreseen to be defined in the first version of HEVC (to be finalized by January 2013). It is possible that some other profiles of the standard will also be specified – either in the first version or soon thereafter. Minimizing the number of profiles provides a maximum amount of interoperability between devices, and is further justified by the fact that traditionally

TABLE VI
STRUCTURE OF CODING TOOLS ASSOCIATED WITH HIGH EFFICIENCY AND LOW COMPLEXITY CONFIGURATIONS OF HM 1

Functionality	High efficiency	Low complexity
CTB / CU structure	Tree structure from 8×8 to 64×64	
PU structure	Square and symmetric shapes	
TU structure	3-level tree structure	2-level tree structure
Transform	Integer transforms from 4 to 32 points	
Intra prediction	33 angular modes with DC mode	
Luma interpolation	12-tap separable	6-tap directional
Chroma interpolation	Bi-linear	
MV prediction	Spatial CU merging / AMVP	
Entropy coding	CABAC	Event-based VLC
Deblocking filter	Enabled	Enabled
Adaptive loop filter	Enabled	Disabled

separate services such as broadcast, mobile, streaming are converging to the point where most devices should become usable to support all of them. The drafted Main profile specification consists of the coding tools and high-layer syntax described in the earlier sections of the paper, except that it imposes the following specific restrictions:

- Only 8-bit video with 4:2:0 chroma sampling is supported.
- When an encoder encodes a picture using multiple tiles, it cannot also use wavefront parallel processing, and each tile must be at least 256 luma samples wide and 64 luma samples tall.

Currently, the definition of 13 levels is planned to be included in the first version of the standard as shown in TABLE V, ranging from levels that support only relatively small picture sizes such as a luma picture size of 176×144 (sometimes called quarter common intermediate format) to picture sizes as large as 7680×4320 (often called 8k×4k). The picture width and height are each required to be less than or equal to $\sqrt{8 \cdot \text{MaxLumaPS}}$, where MaxLumaPS is the maximum luma picture size as shown in TABLE V (to avoid the problems for decoders that could be involved with extreme picture shapes).

There are two tiers supported for eight of these levels (levels 4 and higher). The CPB capacity is equal to the maximum bit rate times 1 second for all levels except level 1, which has a (higher) CPB capacity of 350,000 bits. The specified maximum DPB capacity in each level is 6 pictures when operating at the maximum picture size supported by the level (including both the current picture and all other pictures that are held in the decoder at any point in time for reference or output purposes). When operating with a smaller picture size than the maximum size supported by the level, the DPB picture storage capacity can increase to as many as 16 pictures (depending on the particular selected picture size). Level-specific constraints are also specified for the maximum number of tiles used horizontally and vertically within each picture.

VI. HISTORY AND STANDARDIZATION PROCESS

After the finalization of the H.264/MPEG-4 AVC High profile in mid-2004, both ITU-T VCEG and ISO/IEC MPEG have been trying to identify when the next major advances in coding efficiency would become ready for standardization. VCEG began studying potential advances in 2004, began identifying certain Key Technology Areas (KTAs) for study in early 2005, and developed a common KTA software codebase for this work [14]. Various technologies were proposed and verified using the KTA software codebase, which was developed from the H.264/MPEG-4 AVC reference software known as the Joint Model (JM).

During 2005–2008, MPEG began exploration activities toward significant coding efficiency improvements as well, organized several workshops and issued a “Call for Evidence” [15] of such advances in April 2009. Expert viewing tests were conducted to evaluate submissions of responses to the Call.

From their respective investigations, it was agreed that there were sufficient technologies having the potential to improve the coding efficiency significantly compared to the existing video coding standards. The Joint Collaborative Team on Video Coding (JCT-VC) was planned to be established by both groups in January 2010, and a joint Call for Proposals (CfP) on Video Compression Technology [16] was issued by the same time to identify the initial technologies which would serve as basis of the future standardization activities.

At its first meeting in April 2010, the JCT-VC established the High Efficiency Video Coding (HEVC) project name, studied the proposals submitted in response to the CfP, and established the first version of a “Test Model under Consideration” (TMuC) [17], which was produced collectively from elements of several promising proposals. A corresponding software codebase was implemented after this meeting. The technology submitted in several of the key proposal contributions was previously discussed in a Special Section of the *IEEE Trans. Circuits and Systems for Video Technology* [18].

Although TMuC showed significant coding efficiency improvements compared to prior standards, there were a lot of redundant coding tools in each functional block of the video compression system mainly due to the fact that TMuC was a collective design from various contributions. During the 2nd JCT-VC meeting in July 2010, it was suggested to have a minimal set of coding tools per each functional block by testing each component of TMuC separately. For a detailed description of the experiments, please refer to [19].

Based on the reported results of the exhaustive component testing [20], an HEVC test model version 1 (HM 1) [21] and corresponding HEVC working draft specification version 1 (WD 1) [22] were produced as outputs of the 3rd JCT-VC meeting in October 2010. Compared with the prior TMuC design, HM 1 was simplified greatly by removing coding tools that showed only marginal benefits with respect to their computational complexities.

In several subsequent studies, the coding tools of the HM were classified into two categories: high efficiency and low complexity configurations. Two corresponding test scenarios for verifying future contributions in the JCT-VC were also established. TABLE VI summarizes the HM 1 coding tools for high efficiency and low complexity configurations.

From the 4th to 9th JCT-VC meetings, not just coding efficiency improvements, but many other aspects including computational complexity reduction, unification of various coding tools, and parallel friendly design, were investigated, and the HEVC design was updated accordingly, until the current status of draft standard, as described in the previous sections of this paper, was reached. In this context, it also turned out that the differentiation for low complexity and high efficiency was mostly no longer necessary, such that it became possible to define the unified Main profile. TABLE VII provides a summary of coding tools of the high efficiency configuration in HM 1 and the current specification of HEVC.

At the 8th JCT-VC meeting in February 2012, the draft version 6 of HEVC standard was produced, which was subsequently balloted as the ISO/IEC Committee Draft (CD) of the HEVC standard. The 10th JCT-VC meeting in July 2012 released the draft version 8 for a Draft International Standard (DIS) ballot, and the finalized text for Consent in ITU-T and Final Draft International Standard (FDIS) in ISO/IEC is expected to be produced in January 2013.

Future extensions of HEVC, which are already being explored and prepared by JCT-VC's parent bodies, are likely to include higher-fidelity formats with increased bit depth and enhanced color component sampling, scalable coding and 3D/stereo/multi-view video coding (the latter including the encoding of depth maps for use with advanced 3D displays).

TABLE VII
SUMMARY OF CODING TOOLS OF HIGH EFFICIENCY CONFIGURATION IN HM 1 AND HEVC

Functionality	HM1 High Efficiency	HEVC (draft 8)
CTU structure	Tree structure from 8×8 to 64×64	
PU structure	Square and symmetric	Square, symmetric and asymmetric (only square for intra)
TU structure	Tree structure of square TUs	Tree structure of square TUs
Core transform	Integer transforms from 4 to 32 points (Full factorization)	Integer transforms from 4 to 32 points (Partially factorable)
Alternative transform	n/a	Integer DST type for 4×4
Intra prediction	33 angular modes with DC mode	33 angular modes with planar and DC modes
Luma interpolation	12-tap separable	8-tap/7-tap separable
Chroma interpolation	Bi-linear	4-tap separable
MV prediction	AMVP	AMVP
MC region merging	Spatial CU merging	PU merging
Entropy coding	CABAC	CABAC
Deblocking filter	Non-parallel	Parallel
Sample adaptive offset	n/a	Enabled
Adaptive loop filter	Multiple shapes	n/a
Dedicated tools for parallel processing	Slices	Slices, dependent slices, tiles and wavefronts

VII. CONCLUSIONS AND CLOSING REMARKS

The emerging HEVC standard has been developed and standardized collaboratively by both the ITU-T VCEG and ISO/IEC MPEG organizations. HEVC represents a number of advances in video coding technology. Its video coding layer design is based on conventional block-based motion-compensated hybrid video coding concepts, but with some important differences relative to prior standards.

When used well together, the features of the new design provide approximately a 50% bit rate savings for equivalent perceptual quality relative to the performance of prior standards (especially for high-resolution video). For more details on compression performance, please refer to [23]. Implementation complexity analysis is outside the scope of this paper; however, the implementation complexity of HEVC overall is not a major burden (e.g., relative to H.264/MPEG-4 AVC) using modern processing technology. For more details on implementation complexity, please refer to [24].

Further information and documents of the project are available in the JCT-VC document management system (URL: http://phenix.int-evry.fr/jct/).

ACKNOWLEDGMENT

The authors thank the experts of ITU-T VCEG, ISO/IEC MPEG, and the ITU-T/ISO/IEC Joint Collaborative Team on Video Coding for their contributions.

REFERENCES

[1] B. Bross, W.-J. Han, G. J. Sullivan, J.-R. Ohm, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 8," ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-J1003, July 2012.
[2] ITU-T, "Video Codec for Audiovisual Services at px64 kbit/s," ITU-T Recommendation H.261, Version 1: Nov. 1990; Version 2: Mar. 1993.
[3] ITU-T, "Video coding for low bit rate communication," ITU-T Recommendation H.263, Nov. 1995 (and subsequent editions).
[4] ISO/IEC JTC 1, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s – Part 2: Video," ISO/IEC 11172-2 (MPEG-1), 1993.
[5] ISO/IEC JTC1, "Coding of audio-visual objects – Part 2: Visual," ISO/IEC 14496-2 (MPEG-4 Visual version 1), April 1999 (and subsequent editions).
[6] ITU-T and ISO/IEC JTC 1, "Generic coding of moving pictures and associated audio information – Part 2: Video," ITU-T Recommendation H.262 and ISO/IEC 13818-2 (MPEG 2 Video), Nov. 1994.
[7] ITU-T and ISO/IEC JTC 1, "Advanced Video Coding for generic audio-visual services," ITU T Rec. H.264 and ISO/IEC 14496-10 (AVC), May 2003 (and subsequent editions).
[8] H. Samet, "The quadtree and related hierarchical data structures," *Comput. Surv.* 16 (1984), pp. 187-260
[9] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. on Circuits and Syst. for Video Tech.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[10] S. Wenger, "H.264/AVC over IP," *IEEE Trans. on Circuits and Syst. for Video Tech.*, vol. 13, no. 7, pp. 645–656, July 2003.
[11] T. Stockhammer, M. M. Hannuksela, and T. Wiegand, "H.264/AVC in wireless environments," *IEEE Trans. on Circuits and Syst. for Video Tech.*, vol. 13, no. 7, pp. 657–673, July 2003.
[12] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H. 264/AVC standard, " *IEEE Trans. on Circuits and Syst. for Video Tech.*, vol.17, no. 9, pp. 1103-1120, Sep. 2007.
[13] D. Marpe, H. Schwarz, and T. Wiegand, "Context-adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 620–636, Jul. 2003.
[14] G. J. Sullivan, "Meeting report for 26th VCEG meeting," ITU-T SG16/Q6 document VCEG-Z01, Apr. 2005.
[15] ISO/IEC JTC 1/SC 29/WG 11, "Call for evidence on high-performance video coding (HVC)," ISO/IEC MPEG document N10553, Apr. 2009.
[16] ITU-T and ISO/IEC JTC1, "Joint Call for Proposals on Video Compression Technology," ITU-T VCEG-AM91 and ISO/IEC MPEG document N11113, Jan. 2010.

[17] ITU-T and ISO/IEC JTC1, "Test Model under Consideration," ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-A205, Apr. 2010.
[18] T. Wiegand, J.-R. Ohm, G. J. Sullivan, W.-J. Han, R. Joshi, T. K. Tan, and K. Ugur, "Special Section on the Joint Call for Proposals on High Efficiency Video Coding (HEVC) Standardization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, pp. 1661–1666, Dec. 2010.
[19] K. McCann, "Tool experiment 12: evaluation of TMuC tools," ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-B312, July 2010.
[20] K. McCann, "TE12: summary of evaluation of TMuC tools in TE12," ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-C225, Oct. 2010.
[21] K. McCann, B. Bross, S. Sekiguchi, and W.-J. Han, "Encoder-side description of HEVC Test Model (HM)," ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-C402, Oct. 2010.
[22] T. Wiegand, W.-J. Han, B. Bross, J.-R. Ohm, and G. J. Sullivan, "WD1: working draft 1 of high-efficiency video coding," ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-C402, Oct. 2010.
[23] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards – Including High Efficiency Video Coding (HEVC)", see pp. [INSERT PAGE NUMBER] of this issue.
[24] F. Bossen, B. Bross, K. Sühring, and D. Flynn, "HEVC Complexity and Implementation Analysis", see pp. [INSERT PAGE NUMBER] of this issue.

Gary J. Sullivan (S'83–M'91–SM'01–F'06). See biography on page [INSERT PAGE NUMBER] of this issue.

Jens-Rainer Ohm (M'92). See biography on page [INSERT PAGE NUMBER] of this issue.

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 at the department of Software Design and Management at the Gachon University. From 2003-2011, 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 appointed as the Editor of the HEVC video coding standard. His current research interests include high-efficiency video compression techniques, scalable video coding, multi-view synthesis, and visual contents understanding.

Thomas Wiegand (M'05–SM'08–F'11). See biography on page [INSERT PAGE NUMBER] of this issue.