Transform Coefficient Coding in HEVC

Joel Sole, Rajan Joshi, Nguyen Nguyen, Tianying Ji, Marta Karczewicz,
Gordon Clare, Félix Henry, and Alberto Dueñas

***Abstract*—This paper describes transform coefficient coding in the draft international standard of High Efficiency Video Coding (HEVC) specification and the driving motivations behind its design. Transform coefficient coding in HEVC encompasses the scanning patterns and coding methods for the last significant coefficient, significance map, coefficient levels, and sign data. Special attention is paid to the new methods of last significant coefficient coding, multilevel significance maps, high-throughput binarization, and sign data hiding. Experimental results are provided to evaluate the performance of transform coefficient coding in HEVC.**

***Index Terms*—High Efficiency Video Coding (HEVC), high throughput entropy coder, transform coefficient coding, video coding.**

I. Introduction

ISO/IEC MPEG and ITU-T VCEG formed the Joint Collaborative Team on Video Coding (JCT-VC) to establish a new standardization activity on video coding, referred to as High Efficiency Video Coding (HEVC). A call for proposals was issued in January 2010 and the responses were reviewed at the first JCT-VC meeting in April 2010. The best performing proposals formed the basis of the initial HEVC test model under consideration [1]. The first HEVC test model (HM1.0) was made available in October 2010. Since then, it has undergone several refinements. This paper describes the transform coefficient coding in the draft international standard (DIS) of the HEVC specification [2].

HEVC is a successor to the H.264/AVC video coding standard [3]. One of its primary objectives is to provide approximately two times the compression efficiency of its predecessor without any detectable loss in visual quality. HEVC adheres to the hybrid video coding structure; it uses spatial and temporal prediction, transform of the prediction residual, and entropy coding of the transform and prediction information.

Manuscript received April 16, 2011; revised July 19, 2012; accepted August 21, 2012. Date of publication October 5, 2012; date of current version January 8, 2013. This paper was recommended by Associate Editor A. Kaup.

J. Sole, R. Joshi, and M. Karczewicz are with Qualcomm, San Diego, CA 92121 USA (e-mail: joels@qti.qualcomm.com; rajanj@qti.qualcomm.com; martak@qti.qualcomm.com).

N. Nguyen and T. Ji are with Research In Motion Ltd., Waterloo, ON N2L 5Z5, Canada (e-mail: nnguyen@rim.com; tiji@rim.com).

G. Clare and F. Henry are with Orange Labs, Issy-les-Moulineaux 92794, France (e-mail: gordon.clare@orange.com; felix.henry@orange.com).

A. Dueñas was with Cavium, San Jose, CA 95131 USA. He is now with NGcodec, San Jose, CA 95126 USA.

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

Digital Object Identifier 10.1109/TCSVT.2012.2223055

Fundamental difference between HEVC and previous video coding standards is that HEVC uses a quadtree structure. The quadtree structure is a flexible mechanism for subdividing a picture into different block sizes for prediction and residual coding. In HEVC, a block is defined as an array of samples. A unit encapsulates up to three blocks (e.g., one luma component block and its two corresponding chroma component blocks), and the associated syntactical information required to code these blocks. The basic processing unit is a coding tree unit (CTU) that is a generalization of the H.264/AVC concept of a macroblock. A CTU encapsulates up to three coding tree blocks (CTBs) and the related syntax. Each CTU has an associated quadtree structure that specifies how the CTU is subdivided. This subdivision yields coding units (CUs) that correspond to the leaves of the quadtree structure. A CU uses either intra or inter prediction and is subdivided into prediction units (PUs). For each PU, specific prediction parameters (i.e., intra prediction mode or motion data) are signaled. A nested quadtree, referred to as the residual quadtree (RQT), partitions a CU residual into transform units (TUs). The CUs, PUs, and TUs encapsulate coding blocks (CBs), prediction blocks (PBs), and transform blocks (TBs) respectively, as well as the associated syntax. The quadtree structure gives an encoder greater freedom to select the block sizes and coding parameters in accordance with the statistical properties of the video signal being coded. The reader is referred to [4] for additional details on the use of the quadtree structure in video coding.

After the quadtree structure appropriately determines the TBs, a coded block flag signals whether a TB has any significant (i.e., nonzero) coefficient. If a TB contains significant coefficients, the residual coding process signals the position and value of each nonzero coefficient in the TB. This paper describes the methods used to code this information, with a focus on the transform coefficient coding methods for square TBs. A TB can range in size from 4×4 to 32×32 for luma and from 4×4 to 16×16 for chroma. Non-square TBs are not explicitly discussed because they are coded in the same way as the 16×16 and 32×32 TBs and they are not a part of the HEVC main profile

Transform coefficient coding in HEVC is comprised of five components: scanning (Section III), last significant coefficient coding (Section IV), significance map coding (Section V), coefficient level coding (Section VI), and sign data coding (Section VII). Section II details the design principles that guided the development of the transform coefficient coding algorithms, Section VIII provides experimental results, and Section IX concludes this paper.

II. Transform Coefficient Coding Design

HEVC has a single entropy coding mode based on the context adaptive binary arithmetic coding (CABAC) engine that was also used in H.264/AVC. Unlike H.264/AVC, the context adaptive variable length coding (CAVLC) mode is not supported in HEVC. A thorough treatment of the inner workings of the CABAC engine in H.264/AVC can be found in [5].

The CABAC engine has two operating modes, regular mode and bypass mode. Regular mode has a context modeling stage, in which a probability context model is selected. The engine uses the selected context model to code the binary symbol (bin). After each bin is coded, the context model is updated. Bypass mode assumes an equiprobable model. This mode is simpler and allows a coding speedup and easier parallelization because it does not require context derivation and adaptation.

CABAC is highly sequential and has strong data dependences, which makes it difficult to exploit parallelism and pipelining in video codec implementations. The serial nature of CABAC stems from its feedback loops at the arithmetic coding and context modeling stages, particularly on the decoder side. Context selection for regular bins has two dependences; the context selection and the value of the context (CABAC state) may depend on previous bins. Due to the large percentage of bins devoted to residual coding, it is especially important that transform coefficient coding design limits these dependencies to enable high-throughput implementations.

HEVC introduces several new features and tools for the transform coefficient coding to help improve upon H.264/AVC, such as larger TB sizes, mode dependent coefficient scanning, last significant coefficient coding, multilevel significance maps, improved significance flag context modeling, and sign data hiding. HEVC followed a development process in which it was iteratively refined to improve coding efficiency and suitability for hardware and software implementation. Core experiments (CEs) were an important part of HEVC's iterative development process. A CE is a set of focused tests defined to gain a better understanding of the proposed techniques in JCT-VC. In a CE, new techniques had the opportunity to mature and be refined before their adoption. Over the course of the HEVC standardization process, several CEs on transform coefficient coding were conducted (e.g., [6]).

The following practical issues played key roles in the general design considerations.

1) *Hardware Area*: How many logic gates are required to implement the codec (particularly the decoder) in hardware? Is the size of the hardware reasonable, so as to limit the power consumption, operating temperature, and required footprint in a real device?
2) *SIMD Implementation*: Can single instruction multiple data (SIMD) instruction sets be used to exploit data-level parallelism?
3) *Throughput*: There is a significant complexity difference between the processing of the regular and bypass coded bins. Multiple bypass bins grouped together can be coded in a single cycle. Regular coded bins generally need to be processed sequentially, as the output of one bin may affect the following bin. In contrast, in CAVLC methods, all the data associated with a coefficient can be computed in parallel. CAVLC is traditionally used in applications that require high throughput because it is easier to parallelize its operation, but it leads to a penalty in compression efficiency compared to CABAC. How many regular mode bins are required to code each transform coefficient on average and in the worst-case? How many transform coefficients can be (de)coded per unit time?
4) *Parallelism, Pipelining, and Speculative Computation*: The length of the data dependency path determines the level of parallelism and pipelining. It is possible to use speculative methods to predict the values of multiple bins in parallel. Speculative computation is a look-ahead technique that pre-calculates a tree of several future bins. The proper branch of the tree is selected when the actual bins become available. On average, these methods allow coding of more than one regular bin per cycle. However, in some scenarios, none of the pre-calculated values match the actual bins, leading to problematic corner cases. Hence, speculative computation may not guarantee improvement in throughput in the worst-case. Furthermore, additional hardware resources are required to compute multiple cases in parallel, leading to a larger and more complex design. Can the design facilitate these performance optimizations to improve the overall speed (especially at the decoder)?

Transform coefficient coding in HEVC strives to achieve a balance between coding efficiency and practicality. As such, features and tools addressing the practical issues were adopted only if they improve coding efficiency or at worst, resulted in a slight degradation. With this in mind, the key principles in the design of transform coefficient coding in HEVC can be summarized as follows.

1) Improve coding efficiency, as this is one of the primary goals of HEVC.
2) Reduce the number of coded bins on average and in the worst-case guarantee a minimum throughput.
3) Increase the percentage of bypass bins and group bypass bins together for higher throughput.
4) Reduce the dependency in context derivation of the current bin on previously coded bins.
5) Avoid interleaving syntax elements; a serial dependency exists if a syntax element depends on the value of the previous syntax element.
6) Reduce the number of contexts: in H.264/AVC, the number of contexts used for coefficient coding is a high percentage of the total number of contexts [5].
7) Simplify scans for the hardware and SIMD implementation.
8) Simplify and modularize the coding of large TBs.

III. Scans

There are two distinct concepts in scanning. A scan pattern converts a 2-D block into a 1-D array and defines a processing order for the samples or coefficients. A scan pass is an iteration over the transform coefficients in a block (as per the selected scan pattern) in order to code a particular syntax element.

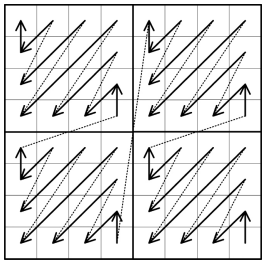

Fig. 1. Diagonal scan pattern in 8×8 TB: the diagonal scan of a 4×4 TB is used within each 4×4 subblock of larger blocks.

A. Scan Patterns

In H.264/AVC, a zigzag scan is used. A zigzag scan interacts poorly with the template-based context models [7] in which the context for a coefficient depends on the previous coefficients whenever the scan moves from one diagonal to another. A diagonal scan starts in the top right corner and proceeds to the bottom left corner. The diagonal scan reduces the data dependency, that allows for a higher degree of parallelism in context derivation. In HEVC, although the context model for the significance of a coefficient has been refined to remove the data dependency problem, however, a diagonal scan is still used instead of a zigzag scan.

In HEVC, the scan in a 4×4 TB is diagonal. The scan in a larger TB is divided into 4×4 subblocks and the scan pattern consists of a diagonal scan of the 4×4 subblocks and a diagonal scan within each of the 4×4 subblocks [8]. This is possible because in HEVC, the dimensions of all TBs are a multiple of 4. Fig. 1 shows the diagonal scan pattern in an 8×8 TB, that splits into 4 subblocks. One reason for dividing larger TBs into 4×4 subblocks is to allow for modular processing, that is, for harmonized subblock based processing across all block sizes. Additionally, the implementation complexity of a scan for the entire TB is much higher than that of a scan based on 4×4 subblocks, both in software (SIMD) implementations and hardware (the estimated gate count for the subblock scan is one half [9]).

Horizontal and vertical scans may also be applied in the intra case for 4×4 and 8×8 TBs. The horizontal and vertical scans are defined by row-by-row and column-by-column scans, respectively, within the 4×4 subblocks. The scan over the 4×4 subblocks is same as that used within the subblock.

B. Scan Passes

A coefficient group (CG) is defined as a set of 16 consecutive coefficients in a scan order. Given the scan patterns in HEVC, a CG corresponds to a 4×4 subblock. This is illustrated in Fig. 2, where each color corresponds to a different CG. A 4×4 TB consists of exactly one CG. TBs of size 8×8, 16×16, 32×32 are partitioned into nonoverlapping 4×4 CGs.

Scanning starts at the last significant coefficient in a block and proceeds to the DC coefficient in the reverse scanning order defined in Section III-A. CGs are scanned sequentially. Up to five scan passes are applied to a CG [10] and all the scan passes follow the same scan pattern [11]. Each scan pass

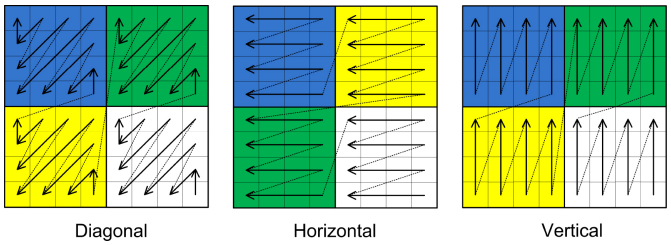

Fig. 2. Coefficient groups for 8×8 TB.

codes a syntax element for the coefficients within a CG, as follows.

1) *significant_coeff_flag*: significance of a coefficient (zero/nonzero).
2) *coeff_abs_level_greater1_flag*: flag indicating whether the absolute value of a coefficient level is greater than 1.
3) *coeff_abs_level_greater2_flag*: flag indicating whether the absolute value of a coefficient level is greater than 2.
4) *coeff_sign_flag*: sign of a significant coefficient (0: positive, 1: negative).
5) *coeff_abs_level_remaining*: remaining value for absolute value of a coefficient level (if value is larger than that coded in previous passes).

In each scan pass, a syntax is coded only when necessary as determined by the previous scan passes. For example, if a coefficient is not significant, the remaining scan passes are not necessary for that coefficient. The bins in the first three scan passes are coded in a regular mode, while the bins in scan passes 4 and 5 are coded in bypass mode, so that all the bypass bins in a CG are grouped together [12]. In certain scenarios, scan passes 2 and 3 may be terminated early. In these cases, the remaining flags are not coded, and any information signaled by these flags is instead signaled by the syntax element *coeff_abs_level_remaining*, thus shifting more bins to bypass mode [13].

Data processing is localized within a CG and once a CG is fully processed, its coefficient levels can be reconstructed before proceeding to the next one. With this syntax-plane coding approach, syntax elements are separated into different scan passes, thus helping speculative coding algorithms [14], since the next syntax element to be processed within a scan pass is known. In contrast, in H.264/AVC, all the syntax elements specifying the level information of a significant coefficient are coded before proceeding to the next coefficient. Therefore, the value of the current syntax element determines the type of the next syntax element to be processed. For instance, if *coeff_abs_level_greater1_flag* equals 1, then the next syntax element is *coeff_abs_level_greater2_flag*. Otherwise, the next element is *coeff_sign_flag*. This kind of dependency is considerably reduced with the new design.

C. Mode Dependent Coefficient Scanning

A new tool in HEVC that improves coding efficiency is mode dependent coefficient scanning (MDCS) [15]. For intra coded blocks, the scanning order of a 4×4 TB and a 8×8 luma TB is determined by the intra prediction mode. Each of the 35 intra modes uses one of the three possible scanning patterns:

diagonal, horizontal, or vertical. A look-up table maps the intra prediction mode to one of the scans.

This tool exploits the horizontal or vertical correlation of the residual depending on the intra prediction mode. For example, for a horizontal prediction mode, transform coefficient energy is clustered in the first few columns, so a vertical scan results in fewer bins being entropy coded. Similarly for a vertical prediction, a horizontal scan is beneficial. Experiments showed that including horizontal and vertical scans for large TBs offers little compression efficiency, so the application of these scans is limited to the two smaller TBs. A detailed performance evaluation of MDCS is provided in Section VIII.

IV. Last Significant Coefficient

A. Last Significant Coefficient Flag

Signaling the last significant coefficient typically reduces the number of coded bins by saving the explicit coding of trailing zeros toward the end of a forward scan. In H.264/AVC, the significance map coding is carried out by interleaving a bin indicating the significance of the coefficient (*significant_coeff_flag*) and a bin indicating whether the coefficient is the last significant within the block (*last_significant_coeff_flag*) when the *significant_coeff_flag* equals 1.

The drawback of the H.264/AVC approach is that the decision regarding coding of the *last_significant_coeff_flag* for a particular transform coefficient is dependent on the value of the *significant_coeff_flag*. As discussed above, this increases the complexity of speculative coding substantially. This is especially critical for the significance map as it may account for more than 40% of the bins.

B. Last Significant Coefficient Coordinates Signaling

In HEVC, the coding of the significance flag is separated from the coding of the last significant coefficient flag. To achieve this, the position of the last significant coefficient in a TB following the forward scan order is coded first, and then, the *significant_coeff_flags* are coded. The position of the last significant coefficient in a block is coded by explicitly signaling its (X, Y)-coordinates [16]. Coordinate X indicates the column number and Y the row number.

The coordinates are binarized in two parts, a prefix and a suffix. The first part represents an index to an interval (syntax elements *last_significant_coeff_x_prefix* and *last_significant_coeff_y_prefix*). This prefix has a truncated unary representation and the bins are coded in regular mode. The second part (*last_significant_coeff_x_suffix* and *last_significant_coeff_y_suffix*) has a fixed length representation and is coded in bypass mode [17]. The suffix represents the offset within the interval. For certain values of the prefix, the suffix is not present and is assumed to be zero.

Let T be the transform size. The number of intervals is $N + 1$, where $N = 2\log_2(T) - 1$. The truncated unary code for size N is used to code the interval index, i.e., the *prefix*. The interval lengths are shared across all transform sizes and the binarization is also shared (except when the unary code is truncated). The suffix is coded only when the interval length is larger than one, that is, when $prefix > 3$. The *suffix* is represented by a fixed length binary code using b bits specifying the offset within interval, where

$$b = \max(0, \lfloor prefix/2 \rfloor - 1) \tag{1}$$

and the suffix range is

$$suffix = \{0, \ldots, 2^b - 1\}. \tag{2}$$

The fixed length code is signaled starting with the most significant bit.

The magnitude of the last position, denoted by *last*, can be derived from the *prefix* and *suffix* as

$$last = \begin{cases} 2^b(2 + \mathrm{mod}(prefix, 2)) + suffix, & \text{if } prefix > 3 \\ prefix, & \text{otherwise} \end{cases} \tag{3}$$

where $mod(.)$ is the modulus after division operation.

The maximum length of the truncated unary code (which is also the number of regular coded bins) for one coordinate is 3, 5, 7, and 9 for transform sizes of 4, 8, 16, and 32, respectively. The maximum number of bins for coding one coordinate (regular and bypass) is 3, 6, 9 and 12, respectively. As an example, Table I shows the binarization for $T = 32$.

In order to group bypass bins, the prefix of the X coordinate is signaled first, followed by the prefix of Y. After that, the suffixes for X and Y are signaled. Coordinates X and Y have separate set of contexts. Different bins within the truncated unary part with similar statistics share contexts in order to reduce the total number of contexts. The number of contexts for the prefix of one coordinate is 18 (15 for luma and 3 for chroma), so the total number of contexts for last position coding is 36. Table II shows the context assignment for different bins for a given coordinate across all transform sizes, luma, and chroma components.

This method has no performance penalty [16] with respect to the interleaved signaling of significance map and last flags in H.264/AVC. At the same time it has the following advantages.

1) On average, the total number of bins for the last position is reduced.
2) In the worst-case, the number of bins are significantly reduced. For an $N \times N$ TB, the maximum number of bins with the H.264/AVC method is $N \times N - 1$. For example, in case of a 32×32 TB, the worst-case is 1023 bins in regular mode, whereas in HEVC, the worst-case is reduced to 24 bins: 12 bins per coordinate when each value is equal or larger than 24 (see last row in Table I).
3) Some of the bins are coded in bypass mode and grouped.
4) Interleaving of significance map and last flags is eliminated.
5) In H.264/AVC, the scan pass for the significance map is in the forward direction due to the method of signaling the last coefficient. For the next scan pass the position of the last coefficient is known, and hence, coefficient levels are scanned in reverse order enabling the usage of

TABLE I
LAST POSITION BINARIZATION FOR T = 32. FOR THE SUFFIX X IS 0 OR 1 AND THE MOST SIGNIFICANT BITS ARE SIGNALED FIRST

Position	***prefix***	***suffix***	Suffix Range
	Truncated unary (regular mode)	Fixed length (bypass mode)	
0	0	–	–
1	10	–	–
2	110	–	–
3	1110	–	–
4–5	11110	X	0 to 1
6–7	111110	X	0 to 1
8–11	1111110	XX	0 to 3
12–15	11111110	XX	0 to 3
16–23	111111110	XXX	0 to 7
24–31	111111111	XXX	0 to 7

TABLE II
LAST POSITION CONTEXT INDEX FOR EACH TRUNCATED UNARY CODE BIN AND T

Bin Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
T	Luma								
4	0	1	2						
8	3	3	4	4	5				
16	6	6	7	7	8	8	9		
32	10	10	11	11	12	12	13	13	14
T	Chroma								
4	15	16	17						
8	15	15	16	16	17				
16	15	15	15	15	16	16	16		

efficient context models [5]. The HEVC method allows all the scan passes to use the same reverse scan order.

V. SIGNIFICANCE MAP

In HEVC, as in H.264/AVC, a coded block flag (CBF) signals the significance of the entire TB, i.e., it signals whether a TB contains nonzero transform coefficients. There are separate CBFs for luma and each of the two chroma components. A CBF is coded using CABAC in regular mode. Context selection for a CBF depends on the depth of the block in the quadtree hierarchy and whether it is comprised of luma or chroma samples. There are two contexts for luma and three for chroma. The contexts are defined in such a way for simplicity and due to the disparity between their statistics. Indeed, luma and chroma blocks have different properties as do blocks at different levels of the RQT. All of the transform coefficients in the TB are zero when a CBF equals zero. On the other hand, when a CBF equals one, the last significant coefficient and the significance map, which identifies the positions of the nonzero coefficients in the TB, are coded.

The significance map is coded in the first scan pass over the residual data. Since the last coefficient is already known to be significant, the significance map scan pass starts at the coefficient before the last coefficient in the scan order, and continues backwards until the top-left coefficient of the CG is reached. The scan pass then proceeds to the next CG in reverse scan order, continuing in this manner until the entire TB has been processed.

A. Multilevel Significance

Effective intra and inter frame prediction methods in HEVC reduce the energy of the prediction residual. The strong energy compaction property of the discrete cosine and sine transforms concentrates this energy in a small number of coefficients. Then, quantization may adjust certain coefficients to zero. As a result, the significance map is often sparse. In order to exploit this sparsity, a new approach introduced in HEVC is to code the significance map within a TB in two levels [18]. The idea is to group together coefficients and code the significance of coefficient groups before coding the significance of the coefficients contained within them. As such, this design is highly compatible with the subblock and CG based scans described in Section III-A. Benefits come from the fact that if the significance of a CG is known to be zero, then the coefficients in that CG do not need to be coded since they can be inferred to be zero.

Let p_0 denote the probability that a given CG is comprised entirely of coefficients that are zero, and let N denote the number of coefficients in that CG. Then, if

$$p_0 + (N+1) \times (1 - p_0) < N \tag{4}$$

the coding complexity measured in terms of the number of explicitly coded bins is reduced. This in turn increases the average throughput as per the design principles. Experimental results (see Section VIII-B) indeed show that the approach in [18] reduces the average number of coded bins, in addition to improving the coding efficiency.

In HEVC, the significance information is coded at multiple levels. The CBF signals the significance of the entire TB, while within a TB, level L_1 corresponds to the significance of CGs, and level L_0 corresponds to the significance of individual coefficients.

At L_1, the significance of a CG is defined to be 1 if at least one coefficient in that CG is nonzero, and 0 otherwise. The significance of a CG is signaled using the syntax element *coded_sub_block_flag* (CSBF). The flag of the CG containing the last significant coefficient is not coded, since it is known to be 1. Similarly, the CSBF of all subsequent CGs (in forward scan order) are not coded because they are known to be 0. To improve coding efficiency, the CSBF of the CG containing the DC coefficient is not coded but instead, implicitly set to be 1 at both the encoder and decoder. This is because the probability of this CG being entirely comprised of coefficients that are 0 is itself nearly 0.

The CSBF is coded using CABAC in regular mode. The context model is based on a template of neighboring CGs, the basic premise being that the significance of neighboring CGs can be used to make a good prediction about the significance of the current CG. The context c_g for the CSBF of a given CG g can be 0 or 1 and is derived as follows:

$$c_g = \min(1, s_r + s_l) \tag{5}$$

where s_r and s_l are equal to the CSBF of the neighboring right and lower CGs, respectively. If the neighboring CG falls outside the boundary of the TB, its CSBF is assumed to be 0.

Fig. 3. Significance flag context assignment in the 4×4 TB. Assignment is based on the position within the block.

At L_0, the significance of the individual coefficients are signaled using the *significant_coeff_flag*. L_1 is leveraged to avoid having to explicitly code this flag in many instances. A *significant_coeff_flag* in the CG containing the DC coefficient is always coded since the CSBF of that CG is always 1. Otherwise, a *significant_coeff_flag* is not coded if:

1) the corresponding coefficient is in a CG with CSBF = 0, in which case the *significant_coeff_flag* is inferred to be 0, or
2) the corresponding coefficient is the last in reverse scan order in a CG with CSBF = 1, and all other coefficients in that CG are 0, in which case the *significant_coeff_flag* is inferred to be 1.

B. Coefficient Significance

A significance flag is coded using a context model for each coefficient between the last one in scanning order (which is excluded) and the DC coefficient. For 4×4 TBs, the context depends on the position of the coefficient within the TB, as in H.264/AVC. Coefficient positions are grouped according to their frequency [19] and the significance flags within a group are coded using the same context. Fig. 3 shows the context modeling for a 4×4 TB. High-frequency coefficients with similar statistical distributions share the same context, while a separate context is assigned to each of the lower frequency coefficients. Luma and chroma components are treated in the same way for simplicity, even though significance information in chroma blocks requires less modeling.

The position-based context modeling approach is simple to implement and allows for a high degree of parallelism in context derivation. On the other hand, a template-based context modeling approach, in which a context is determined using a causal neighborhood [7], provides a higher coding efficiency in large TBs. However, this approach does not allows for a high degree of parallelism because in certain cases, a context is dependent on the significance of coefficients immediately preceding it in the scan order.

In HEVC, context modeling for significance flags in 8×8, 16×16 and 32×32 TBs is both position and template-based. The key is that the template is designed to avoid data dependencies within a CG [20]. As shown in Fig. 4, a context is selected for a significance flag depending on a template of the neighboring right and lower CSBF, s_r and s_l respectively, and on the position of the coefficient within the current CG. There are 4 patterns, corresponding to the 4 combinations of s_r and s_l, with each pattern assigning different contexts (represented by distinct numbers in Fig. 4) to the different positions in the CG [21]. For example, if $s_r = 0$, $s_l = 1$ and the coefficient is in the top leftmost position of the CG, the context

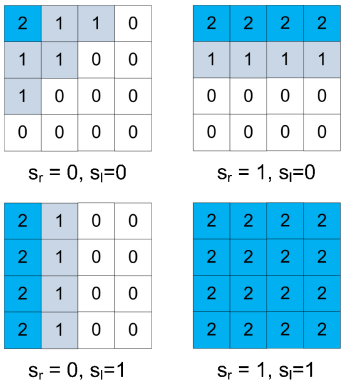

Fig. 4. Significance flag context assignment in the 8×8, 16×16, and 32×32 TBs. A context depends on the significance of the neighboring right (s_r) and lower (s_l) CGs and on the coefficient position in the current CG.

for its significance flag is "2." This design sacrifices some of the coding gain of a context model based on a template of neighboring coefficients, but allows the determination of up to 16 contexts in parallel. This tradeoff is an example of how transform coefficient coding in HEVC achieves a balance between coding efficiency and practicality.

TBs are split into two regions: the top leftmost subblock is region 1, and the rest of subblocks make up region 2. Both regions use the context selection method described above, but have different sets of contexts for luma to account for the different statistics of the low and high frequencies. For chroma blocks, contexts for regions 1 and 2 are shared. The DC component has a single dedicated context and it is shared across all TB sizes. TBs of size 16×16 and 32×32 share contexts to limit the total number of contexts.

VI. Coefficient Level

H.264/AVC codes the absolute level as a truncated unary code in the regular mode for bins 1 to 14. If the level is larger than 14, then a suffix is appended to the truncated unary code. The suffix is binarized with a 0^{th}-order Exp-Golomb code (EG0) and coded in bypass mode.

Coefficient level coding in HEVC is partly inherited from H.264/AVC. Several modifications are introduced to address large TBs [7] and to enhance throughput by encoding more bins in bypass mode [13], [22]. The absolute level of a significant coefficient is coded in the second, third and fifth scanning passes in a CG. The corresponding syntax elements are *coeff_abs_level_greater1_flag*, *coeff_abs_level_greater2_flag* and *coeff_abs_level_remaining*.

A. Greater Than One and Two Flags

In order to improve throughput, the second and third passes may not process all the coefficients in a CG [13]. The first eight *coeff_abs_level_greater1_flag*s in a CG are coded in regular mode. After that, the values are left to be coded in bypass mode in the fifth pass by the syntax *coeff_abs_level_remaining*. Similarly, only the *coeff_abs_level_greater2_flag* for the first

TABLE III
CONTEXT SET DEFINITION FOR COEFFICIENT LEVEL CODING

Context Sets for a CG				
	Luma		Chroma	
# of *coeff_abs_level_greater1_flag* = 1 in previous CG	0	> 0	0	> 0
Region 1 (top left CG)	0	1	4	5
Region 2 (other CGs)	2	3	4	5

TABLE IV
CONTEXT MODELS IN EACH CONTEXT SET FOR *coeff_abs_level_greater1_flag*

Context Model	Description
0	1 or more larger than 1
1	Initial—no trailing ones
2	1 trailing one
3	2 or more trailing ones

coefficient in a CG with magnitude larger than 1 is coded. The rest of coefficients with magnitude larger than 1 of the CG use *coeff_abs_level_remaining* to code the value. This method limits the number of regular bins for coefficient levels to a maximum of 9 per CG: 8 for the *coeff_abs_level_greater1_flag* and 1 for *coeff_abs_level_greater2_flag*. There is no performance impact by introducing this method in the HEVC design as demonstrated in [13].

For coefficient level flags, a context set is selected depending on whether there is a *coeff_abs_level_greater1_flag* equal to 1 in the previous CG [23] and whether the DC coefficient is part of the CG [11], i.e., if the current CG is in region 1 or 2. For chroma, the context set assignment does not depend on the CG location. Therefore, there are 4 different contexts sets for luma and 2 for chroma, as shown in Table III. Each set has 4 context models for *coeff_abs_level_greater1_flag* and 1 context for *coeff_abs_level_greater2_flag*, so the number of contexts for these syntax elements is 24 and 6, respectively. The specific context within a context set for *coeff_abs_level_greater1_flag* is selected depending on the number of trailing ones and the number of coefficient levels larger than 1 in the current CG (Table IV). The same logic and contexts are applied to all TB sizes.

B. Remaining Absolute Level

After the level flags are coded, the fifth and last scan pass codes the syntax element *coeff_abs_level_remaining*, which specifies the remaining absolute value of the coefficient level. Let the *baseLevel* of a coefficient be defined as

$$\begin{aligned} baseLevel = {} & significant\_coeff\_flag \\ & + coeff\_abs\_level\_greater1\_flag \\ & + coeff\_abs\_level\_greater2\_flag \end{aligned} \tag{6}$$

where a flag has a value of 0 or 1 and is inferred to be 0 if not present. Then, the absolute value of the coefficient is simply

$$absCoeffLevel = baseLevel + coeff\_abs\_level\_remaining. \tag{7}$$

The syntax element *coeff_abs_level_remaining* is present in the bitstream if a coefficient level is greater than 2 or whenever the maximum number of *coeff_abs_level_greater1_flag* or *coeff_abs_level_greater2_flag* per CG is reached. *Coeff_abs_level_remaining* is binarized using Golomb–Rice codes and Exp-Golomb codes [24].

Golomb–Rice codes are a subset of Golomb codes and represent a value $n >= 0$, given a tunable Rice parameter m, as a quotient q and a remainder r

$$q = \lfloor n/m \rfloor \tag{8}$$

$$r = n - q \times m \tag{9}$$

where m is a power of 2. The quotient q is the prefix and has a unary code representation. The remainder r is the suffix and has a fixed length representation. Golomb–Rice codes are attractive here for several reasons. Firstly, they are optimal for geometrically distributed sources such as the residual coefficients. Secondly, since m is a power of 2, division and multiplication can be efficiently implemented using shift operations. Finally, the fixed length part is coded with exactly $log_2(m)$ bins, which simplifies reading from the bitstream.

Exp-Golomb codes, like Golomb–Rice codes, have implementation and speed advantages. They are also very efficient for geometric distributions, but more robust to changes in the source distribution. The code structure is similarly formed by a unary prefix followed by a fixed length suffix, but the number of codewords in the suffix part doubles after each bit in the unary code. Therefore, Exp-Golomb codes have a slower growth of the codeword length. By using Exp-Golomb codes, the maximum codeword length for *coeff_abs_level_remaining* is kept within 32 bits.

The syntax element *coeff_abs_level_remaining* is coded in bypass mode in order to increase throughput. HEVC employs Golomb–Rice codes for small values and switches to an Exp-Golomb code for larger values. The transition point between the codes is when the unary code length equals 4. Table V shows the binarization for Rice parameter $m = 0$ and $m = 1$.

The Rice parameter is set to 0 at the beginning of each CG and it is conditionally updated depending on the previous value of the parameter and the current absolute level as follows:

$$\text{if } absCoeffLevel > 3 \times 2^m, \quad m = \min(4, m + 1). \tag{10}$$

The parameter update process allows the binarization to adapt to the coefficient statistics when large values are observed in the distribution. Fig. 5 summarizes the absolute level binarization processes of H.264/AVC and HEVC.

VII. SIGN DATA

In HEVC, the sign of each nonzero coefficient is coded in the fourth scan pass in bypass mode, assuming that these symbols are equiprobable and uncorrelated. Sign flags *coeff_sign_flag* represent a substantial proportion of a compressed bitstream (around 15–20% depending on the configurations). It is difficult to directly compress this information.

TABLE V
BINARIZATION FOR THE REMAINING LEVEL

Value	***prefix***	***suffix***	Suffix Range
m=0	unary	fixed length	
0	0	–	–
1	10	–	–
2	110	–	–
3	1110	–	–
4–5	11110	X	0 to 1
6–9	111110	XX	0 to 3
10–17	1111110	XXX	0 to 7
18–33	11111110	XXXX	0 to 15
...	...	...	...
m=1	unary	fixed length	
0–1	0	X	0 to 1
2–3	10	X	0 to 1
4–5	110	X	0 to 1
6–7	1110	X	0 to 1
8–11	11110	XX	0 to 3
12–19	111110	XXX	0 to 7
20–35	1111110	XXXX	0 to 15
36–67	11111110	XXXXX	0 to 31
...	...	...	...

The top of the table is for Rice parameter equals 0 and the bottom is for Rice parameter equals 1.

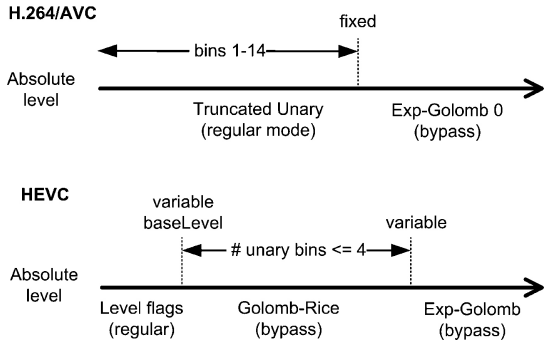

Fig. 5. Binarization of absolute value of the transform coefficients for H.264/AVC and HEVC.

However, HEVC provides a mechanism to reduce the number of coded signs, called sign data hiding (SDH), introduced in [25] and [26].

For each CG, and depending on a criterion, encoding the sign of the last nonzero coefficient (in reverse scan order) is simply omitted when using SDH. Instead, the sign value is embedded in the parity of the sum of the levels of the CG using a predefined convention: even corresponds to "+" and odd to "-." The criterion to use SDH is the distance in scan order between the first and the last nonzero coefficients of the CG. If this distance is equal or larger than 4, SDH is used. This value of 4 was chosen because it provides the largest gain on HEVC test sequences. Also, this value is fixed in the standard, because experiments could not establish that additional compression gain can be obtained by allowing this threshold to vary for different sequences or pictures [27]. Having a fixed value simplifies hardware implementation and bitstream conformance testing.

On the encoder side, if the criterion is met and if the parity of the sum of the levels in the CG matches the sign to omit, there is no additional process, and one bit of information is saved by avoiding the signaling of this sign. If the parity does not match the sign to omit, the encoder has to change the value of one of the quantized coefficients in the CG so that the parity matches the sign. This is an encoder choice and it is up to the implementer to decide which coefficient to modify and how. Of course, it is preferable to make the change that least affects the rate-distortion (RD) performance. References [25] and [26] show that such change can be found with a satisfying tradeoff between complexity and compression.

A first approach [25] relies on rate-distortion optimized quantization (RDOQ) being used during encoding. RDOQ [28], [29] is an encoder-only method that adjusts the quantized values of the coefficients to minimize a joint RD cost function. RDOQ tests alternate quantization values of the coefficients and selecting them if they provide a better RD tradeoff compared to the inital ones. When SDH is used, the RD costs computed during the RDOQ phase are also used to identify the parity change that least degrades RD performance. This can be performed without computing new RD costs in addition to the ones already computed by RDOQ and, therefore, very little additional complexity is needed.

A second approach [26] has been proposed when RDOQ is not used, such as in low-complexity encoders. Here, it is desirable to avoid computing RD costs, mostly because of the complexity incurred by simulating the encoding of alternate quantization values by CABAC. Therefore, for each coefficient in a CG, only the difference between the original coefficient and its dequantized value is computed. The coefficient that yields the largest difference magnitude in its CG has its quantized value increased by one (if the difference is positive) or decreased by one (if the difference is negative), thus providing the parity change. Since the coefficient with the largest difference magnitude is also the closest to its alternate quantization value, this process ensures that the impact of the parity change is small. The computation of the difference can be simply derived from the usual quantization formula. Therefore, the impact on the encoder complexity is modest.

On decoder side, if the SDH criterion is met, the sign of the last nonzero coefficient of each CG is not decoded. Instead, it is inferred from the parity of the sum of the levels in the CG. The advantage of hiding the sign of the last coefficient in scan order (instead of the first, for instance) is clear; when the last coefficient is reached, the information needed to process its sign when SDH is used (such as obtaining the parity of the sum of the quantized coefficients in order to infer the sign) is already available.

The rationale behind SDH, as shown on Fig. 6 (left diagram), resides in the fact that, in about 50% of the CGs where it is used, a full bit is saved. In the other 50%, when a change in one of the quantization levels is needed, the RD loss is moderate because there exist quantization solutions of the CG that are close to the optimal solution and have the opposite parity, as shown in Fig. 6 (right diagram). In this case, embedding the sign bit in a CG gives enough of a chance to find a quantized coefficient that causes moderate RD loss when modified. Furthermore, when there are more sign bits to hide (e.g., with small quantization steps), there is more residual data where to embed them.

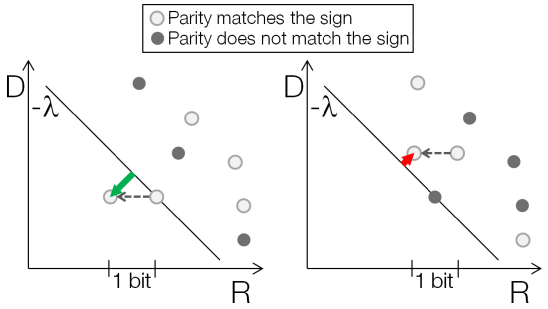

Fig. 6. Example of RD cost modification when the parity matches the sign (left, large RD gain) and when it does not (right, small RD loss). Dashed arrow: rate gain of one bit provided by omitting the signaling of one sign.

HEVC includes several tools that are aimed at facilitating lossless encoding at the CU level. Since SDH usually requires some change in the quantization levels, it is inherently a lossy algorithm. Therefore, the standard does not allow the use of SDH in CUs where lossless tools are activated [30]. Also, a specific syntax element is provided to activate SDH at picture level during encoding, so an encoder can choose simply not to use SDH if it does not match its efficiency-complexity target.

VIII. Experimental Results

This section provides performance results for transform coefficient coding in HEVC. Individual results for the new features of MDCS, multilevel significance map, and SDH are first reported. Then, HEVC transform coefficient coding and H.264/AVC transform coefficient coding are compared as a whole with respect to coding efficiency, bin usage, and throughput. This is done by comparing a realization of HEVC, HM8.0, and a model of H.264/AVC. In this model, H.264/AVC transform coefficient coding has been extended in a straightforward manner, as in [31], to deal with transforms larger than 8×8 and integrated into HM8.0. The main profile does not include non-square transforms, so no further extension to cover them is necessary. However, RDOQ has been disabled in order to measure the performance of the coefficient coding methods directly without the influence of this encoder-only technique.

The conducted experiments follow the JCT-VC common test conditions as described in [32] using HEVC main profile in HM8.0. The test set has twenty sequences split in five classes depending on the resolution: class A (4K), class B (1080p), class C (WVGA), class D (WQVGA), and class E (720p). Additionally, there is a class F (not included in the performance averages) composed of screen content sequences. Ten seconds of each sequence are encoded. The three coding configurations in common test conditions are all-intra (AI), low delay (LD), and random access (RA) with a hierarchical GOP structure of size 8 and refresh points every second. Common test conditions do not include results for the highest resolution sequences (class A) in the LD configuration and for the video-conference content (class E) in the RA configuration. The quantization parameter is set to $QP = \{22, 27, 32, 37\}$. Coding efficiency results are presented as the percentage of bit-rate savings (BD-rate [33]) with respect to the main profile anchor. BD-rate computes an average of the bit-rate savings over the four QP points. Positive numbers indicate BD-rate losses. Results include the encoding and decoding times of the methods as a percentage of the tested anchor.

TABLE VI
BD-Rate [%] of MDCS

	All Intra			Random Access			Low Delay		
	Y	U	V	Y	U	V	Y	U	V
Class A	0.4	0.5	0.4	0.2	0.1	0.0	–	–	–
Class B	0.5	0.7	0.9	0.3	0.3	0.4	0.1	0.1	0.1
Class C	1.0	1.5	1.5	0.4	0.7	0.6	0.2	0.4	0.2
Class D	1.0	1.6	1.7	0.4	0.7	0.8	0.1	0.9	0.0
Class E	1.4	0.6	0.6	–	–	–	0.3	−0.7	0.3
Overall	**0.8**	**1.0**	**1.0**	**0.3**	**0.5**	**0.5**	**0.1**	**0.2**	**0.2**
Class F	0.7	1.1	1.0	0.3	0.4	0.3	0.4	0.6	0.7
Enc Time	100%			100%			100%		
Dec Time	100%			100%			100%		

TABLE VII
BD-Rate [%] of Multilevel Significance Map

	All Intra			Random Access			Low Delay		
	Y	U	V	Y	U	V	Y	U	V
Class A	1.1	2.2	2.2	1.0	0.6	0.7	–	–	–
Class B	1.1	2.1	2.2	1.0	1.6	1.8	1.1	1.7	1.6
Class C	0.6	1.0	0.9	0.9	0.8	0.9	1.4	1.1	0.6
Class D	0.6	1.0	0.8	0.8	0.6	0.3	1.0	0.2	0.0
Class E	1.4	1.8	1.8	–	–	–	1.2	0.5	2.6
Overall	**1.0**	**1.6**	**1.6**	**0.9**	**1.0**	**1.0**	**1.2**	**0.9**	**1.1**
Class F	0.4	0.4	0.3	0.6	0.4	0.5	1.0	1.0	0.5
Enc Time	101%			99%			100%		
Dec Time	100%			100%			100%		

A. MDCS Performance

The performance of MDCS is tested by disabling the method. Table VI compares MDCS on and off cases. Positive numbers show the loss incurred by disabling MDCS. For the AI configuration, MDCS shows an average BD-rate gain of 0.8%. Since MDCS is applied only to intra predicted blocks, the gains are lower for the configurations that make use of inter prediction. For RA, the gain is 0.3%, and for LD, which uses less intra prediction than RA, the gain is 0.1%. Encoding and decoding times are essentially unchanged.

B. Multilevel Significance Map Performance

The performance of the multilevel significance map is measured by disabling level L_1 significance. Tables VII and VIII report the BD-rate and number of bins of having the multilevel significance map enabled versus disabled. In Table VIII, the columns labeled T, R, and S correspond to the total bins per pixel, regular coded bins per pixel, and significance map bins per pixel, respectively. In both tables, positive numbers show the loss incurred by disabling the L_1 significance.

Table VIII shows average savings of significance bins per pixel of 5.5% in AI, 8.2% in RA and 16.2% in LD. The main reason for differences in the three configurations is that inter predicted blocks have sparser residue than intra predicted blocks, since inter prediction is typically more accurate. As explained in Section V-A, the multilevel significance map is

TABLE VIII
AVERAGE BINS PER PIXEL SAVINGS [%] OF MULTILEVEL SIGNIFICANCE MAP

	All Intra			Random Access			Low Delay		
	T	R	S	T	R	S	T	R	S
Class A	1.8	3.0	2.4	3.2	4.9	8.9	–	–	–
Class B	2.8	4.2	3.2	4.4	5.8	10.3	5.6	7.0	12.8
Class C	1.4	2.2	1.5	3.1	4.5	8.5	4.2	5.7	10.8
Class D	1.3	2.1	2.3	2.0	3.0	5.3	2.1	2.8	4.3
Class E	5.8	9.2	18.0	–	–	–	10.7	13.7	36.7
Overall	**2.6**	**4.1**	**5.5**	**3.2**	**4.5**	**8.2**	**5.7**	**7.3**	**16.2**
Class F	1.0	1.7	2.5	1.5	2.2	4.1	2.3	3.3	6.7

designed to take advantage of sparse residual. Table VII shows that the average BD-rate gain provided by the method is between 1.0% and 1.2%, without a noticeable effect on encoding and decoding times. Disabling L_1 increases the number of bins and changes the significance map statistics, which negatively affects other significance map methods tuned to work along with L_1. The total bin savings does not directly translate into corresponding BD-rate gains because the significance flag bins eliminated by a zero CSBF have relatively low entropy

C. SDH Performance

An encoder can choose whether to activate or deactivate the SDH tool. Table IX shows the loss incurred over the different test configurations when SDH is disabled. In these experiments, SDH is applied with the encoder approaches described in Section VII. The first approach is used together with RDOQ (which is a part of the main profile common test conditions), and the second approach is used for the case when RDOQ is disabled. The average gain provided by SDH is 0.8% when used on top of RDOQ. When RDOQ is disabled, transform coefficients tend to have larger magnitude, and therefore SDH is activated more often, and an average gain of 1.4% is observed. Since SDH requires additional computation on the encoder side to determine the right place to hide the sign bit, the encoding time is decreased by an average of 2% when SDH is deactivated. The decoding time is essentially unchanged because there is very little additional decoding complexity due to SDH and the number of bins is reduced slightly. The encoder strategies may be modified to further improve the gains achieved by SDH. For instance, since coefficients are heavily interdependent for entropy coding, an encoder may chose to test several combined coefficient changes in order to obtain the desired parity, instead of changing just one. This type of approach could potentially provide compression gains at the expense of additional encoder complexity.

D. Coding Efficiency

Transform coefficient coding in HEVC and the H.264/AVC model are compared in terms of bit-rate savings. Table X shows that on average, the HEVC method reduces the BD-rate by approximately 4.5% for AI and 3.5% for RA and LD when compared to the H.264/AVC model. Coding efficiency improvement is greater when more residual data is present i.e., at high bit-rates and for intra coding. Most of the gain comes from MDCS, multilevel significance map and SDH. Some gain comes from the careful selection of a reduced and meaningful set of context models for the syntax elements. Table XI provides a summary of the number of contexts used for the syntax elements related to coefficient coding in HEVC and H.264/AVC.

TABLE IX
BD-RATE [%] OF SIGN DATA HIDING FOR COMMON TEST CONDITIONS (TOP-TABLE) AND WITH RDOQ DISABLED (BOTTOM-TABLE)

	All Intra			Random Access			Low Delay		
	Y	U	V	Y	U	V	Y	U	V
Class A	1.1	1.5	1.7	1.0	0.3	0.5	–	–	–
Class B	0.9	1.6	1.6	0.7	0.9	1.0	0.9	1.1	1.3
Class C	0.8	1.1	1.2	0.6	0.5	0.7	0.9	0.8	0.9
Class D	0.9	1.3	1.2	0.7	0.5	0.6	0.8	1.6	0.6
Class E	0.5	1.5	1.6	–	–	–	0.2	0.6	1.5
Overall	**0.9**	**1.4**	**1.4**	**0.8**	**0.5**	**0.7**	**0.7**	**1.1**	**1.1**
Class F	0.7	0.6	0.4	0.5	0.0	−0.1	0.8	0.3	−0.1
Enc Time	97%			100%			100%		
Dec Time	98%			101%			102%		
RDOQ off	All Intra			Random Access			Low Delay		
	Y	U	V	Y	U	V	Y	U	V
Class A	1.7	1.4	1.4	2.0	0.0	0.4	–	–	–
Class B	1.4	1.2	1.1	1.6	0.9	1.2	1.3	0.0	0.0
Class C	1.4	1.4	1.3	1.4	0.8	1.1	1.1	0.2	−0.2
Class D	1.5	1.3	1.3	1.6	1.1	1.0	1.3	0.3	0.3
Class E	1.2	0.8	0.8	–	–	–	0.6	−0.5	0.4
Overall	**1.5**	**1.2**	**1.2**	**1.6**	**0.7**	**1.0**	**1.1**	**0.0**	**0.1**
Class F	1.0	1.0	0.7	1.1	1.0	0.4	1.0	0.5	−0.3
Enc Time	96%			97%			98%		
Dec Time	101%			100%			101%		

TABLE X
BD-RATE [%] OF HEVC COEFFICIENT CODING COMPARED TO H.264/AVC MODEL (RDOQ DISABLED)

RDOQ off	All Intra			Random Access			Low Delay		
	Y	U	V	Y	U	V	Y	U	V
Class A	5.0	6.0	6.2	4.4	1.8	2.8	–	–	–
Class B	4.2	5.8	5.9	3.1	3.5	3.6	3.3	3.3	2.8
Class C	4.3	4.8	4.9	3.1	2.8	3.0	4.1	3.0	2.7
Class D	4.6	5.4	5.3	3.0	2.8	2.4	3.7	1.8	2.9
Class E	4.9	6.0	6.2	–	–	–	3.3	2.4	3.2
Overall	**4.6**	**5.6**	**5.7**	**3.4**	**2.8**	**3.0**	**3.6**	**2.7**	**2.9**
Class F	2.4	2.5	2.4	2.0	1.9	1.8	2.8	2.7	2.7
Enc Time	98%			98%			100%		
Dec Time	97%			99%			99%		

TABLE XI
HEVC AND H.264/AVC COEFFICIENT CONTEXTS PER SYNTAX ELEMENT

Syntax	**HEVC**		**H.264/AVC**	
	Y	U/V	Y	U/V
Last (X&Y)	30	6	53	17
CSBF	2	2	–	–
Significance	27	15	59	17
Greater than 1	16	8	20	10
Greater than 2	4	2	20	9
Total	**79**	**33**	**152**	**53**

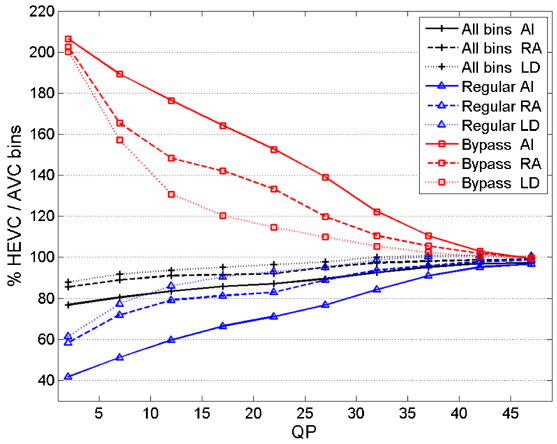

Fig. 7. Average HEVC to H.264/AVC model bin ratio (in %) for AI, RA, and LD configurations in HM8.0 (RDOQ disabled).

E. Throughput Analysis

In H.264/AVC, applications requiring higher throughput than those achieved by using CABAC could rely on CAVLC entropy coding. However, HEVC supports only one entropy coder based on CABAC. Hence, it is crucial that residual coding can achieve much higher throughput when compared to the H.264/AVC CABAC method [34].

HEVC substantially reduces the average number of coded bins of the H.264/AVC model. Fig. 7 shows the average ratio of HEVC to H.264/AVC model number of bins for the three coding configurations and a wide range of QP values. HEVC codes fewer bins and a higher percentage of them are coded in bypass mode. The average number of bins per pixel for HEVC are shown in Table XII. The highest bin-rate occurs for all-intra at low QP. For $QP = 0$ (left column), 6.9 bins per pixel are coded. In that scenario, HEVC reduces the average number of regular mode bins up to 2.5 times. The percentage of regular and bypass coded bins in HEVC are shown in Table XIII. The data is split for coefficient-related syntax elements. For AI and $QP = 0$, the average percentage of bypass bins is above 60%: most of them are grouped together in the sign and remaining level scan passes.

However, it is fundamental to focus on the worst-case when comparing the implementation complexity of different entropy coding methods as the worst-case defines the hardware requirements. In case of HEVC, the worst-case is greatly improved compared to H.264/AVC [35]. In H.264/AVC, more than 15 bins per coefficient may be coded with adaptive context modeling, while remaining bins are coded in bypass mode. Using this approach, all the residual data could use regular bins in the worst-case even with typical operational data rates. HEVC minimizes the maximum number of regular coded bins per coefficient and achieves a throughput closer to what would traditionally have been possible using CAVLC schemes. For a 4×4 TB, HEVC requires at most 30 regular coded bins: 6 for last coding, 15 for significance, 8 for the larger than one flag, and 1 for the larger than two flag. Thus, the worst-case is 1.875 regular bins/coefficient. The ratio is lower for larger TBs, due to the last significant coefficient coding method. In comparison, H.264/AVC requires 15.875 regular bins/coefficient for a 4×4 TB and 15.969 regular bins/coefficient for a 8×8 TB in the worst-case.

TABLE XII
AVERAGE NUMBER OF BINS PER PIXEL FOR AI, RA, AND LD CONFIGURATIONS AND DIFFERENT QP VALUES

	AI			RA		LD	
QP	0	22	37	22	37	22	37
Bins per pixel	**6.89**	**1.35**	**0.288**	**0.290**	**0.028**	**0.281**	**0.024**

TABLE XIII
PROPORTION [%] OF BINS USED FOR EACH SYNTAX ELEMENT

	AI			RA		LD	
QP	0	22	37	22	37	22	37
Last (prefix)	4.9	10.9	13.6	10.6	9.5	12.9	8.9
CSBF	0.4	1.0	0.9	1.3	0.6	1.2	0.6
Significance	18.9	29.5	25.4	30.7	16.6	33.1	14.6
Greater than 1	10.0	13.5	11.7	11.7	7.2	10.8	6.0
Greater than 2	1.2	1.5	1.2	1.4	0.8	1.0	0.7
All other syntax	2.2	9.0	22.0	18.3	40.8	22.2	49.0
% Regular	**37.6**	**65.5**	**74.7**	**74.0**	**75.6**	**81.3**	**79.8**
Last (suffix)	0.3	0.5	0.5	0.6	0.4	0.7	0.4
Sign	15.0	13.6	10.4	11.0	6.4	9.2	5.2
Remaining level	45.8	14.7	3.9	8.0	2.1	3.5	1.0
All other syntax	1.3	5.7	10.4	6.4	15.5	5.2	13.7
% Bypass	**62.4**	**34.5**	**25.3**	**26.0**	**24.4**	**18.7**	**20.2**

All other syntax refers to all the other syntax elements in HEVC.

The studies in [36]–[39] indicate that in practice, the parsing and decoding of the transform coefficients in HEVC, which account for over 50% of the compressed bitstream, takes at most the same amount of time as the motion compensation or deblocking filter stages.

IX. CONCLUSION

This paper described in detail the transform coefficient coding in the HEVC DIS specification. Transform coefficient coding in HEVC strives to strike a balance between high coding efficiency and practicality of implementation. It is comprised of five components: scanning, last significant position coding, significance map coding, coefficient level coding, and sign data coding. Since HEVC and H.264/AVC share the basic arithmetic coding engine, the HEVC transform coefficient coding design has sought to overcome the shortcomings in the H.264/AVC design with respect to its throughput capacity, which became apparent during the implementation phase of that standard. The new design is capable of high compression efficiency and delivering high throughput at the same time, leading to a single entropy coder that can address a wider range of applications.

From the point of view of coding efficiency, existing methods in significance map and level coding were improved while new schemes such as MDCS, multilevel significance map and SDH were introduced, leading to an overall average gain of 3.5% over the H.264/AVC-like transform coefficient coding. To summarize, HEVC transform coefficient coding improves coding efficiency while reducing the average and worst-case complexity.

REFERENCES

[1] ITU-T and ISO/IEC JTC1, *Test Model Under Consideration*, JCTVC-A205, 1st Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Dresden, Germany, Apr. 2010.

[2] B. Bross, W.-J. Han, J.-R. Ohm, G. Sullivan, and T. Wiegand, *High Efficiency Video Coding (HEVC) Text Specification Draft 8*, Joint Collaborative Team on Video Coding (JCTVC-J1003), Jul. 2012.

[3] ITU-T and ISO/IEC, *Advanced Video Coding for Generic Audiovisual Services*, ITU-T Rec. H.264 and ISO/IEC 14496-10, version 13, 2010.

[4] D. Marpe, H. Schwarz, S. Bosse, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, T. Nguyen, S. Oudin, M. Siekmann, K. Suhring, M. Winken, and T. Wiegand, "Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1676–1687, Dec. 2010.

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

[6] V. Sze, T. Nguyen, K. Panusopone, Y. Piao, and J. Sole, *CE11: Summary Report of Core Experiment on Coefficient Scanning and Coding*, JCTVC-H0041, 8th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, San Jose, CA, Feb. 2012.

[7] T. Nguyen, H. Schwarz, H. Kirchhoffer, D. Marpe, and T. Wiegand, "Improved context modeling for coding quantized transform coefficients in video compression," in *Proc. Picture Coding Symp.*, Dec. 2010, pp. 378–381.

[8] J. Sole, R. Joshi, and M. Karczewicz, *Non-CE11: Diagonal Sub-Block Scan for HE Residual Coding*, JCTVC-G323, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Nov. 2011.

[9] C. Auyeung and T. Suzuki, *CE11: Hardware Complexity of Large Zig-Zag Scan for Level-Coding of Transform Coefficients*, JCTVC-F597, 6th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Torino, Italy, Jul. 2011.

[10] J. Sole, R. Joshi, and M. Karczewicz, *CE11: Scanning Passes of Residual Data in HE*, JCTVC-G320, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Nov. 2011.

[11] J. Sole, R. Joshi, and M. Karczewicz, *CE11: Unified Scans for the Significance Map and Coefficient Level Coding in High Efficiency*, JCTVC-F288, 6th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Torino, Italy, Jul. 2011.

[12] V. Sze and M. Budagavi, *Parallel Context Processing of Coefficient Level*, JCTVC-F130, 6th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Torino, Italy, Jul. 2011.

[13] J. Chen, W.-J. Chien, R. Joshi, J. Sole, and M. Karczewicz, *Non-CE1: Throughput Improvement on CABAC Coefficients Level Coding*, JCTVC-H0554, 8th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, San Jose, CA, Feb. 2012.

[14] M. Budagavi and M. U. Demircin, *Parallel Context Processing Techniques for High Coding Efficiency Entropy Coding in HEVC*, JCTVC-B088, 2nd Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Jul. 2010.

[15] Y. Zheng, M. Coban, X. Wang, J. Sole, R. Joshi, and M. Karczewicz, *CE11: Mode Dependent Coefficient Scanning*, JCTVC-D393, 4th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Daegu, Korea, Jan. 2011.

[16] J. Sole, R. Joshi, and M. Karczewicz, *CE11: Parallel Context Processing for the Significance Map in High Coding Efficiency*, JCTVC-E338, 5th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Mar. 2011.

[17] W.-J. Chien, J. Sole, and M. Karczewicz, *Last Position Coding for CABAC*, JCTVC-G704, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Nov. 2011.

[18] N. Nguyen, T. Ji, D. He, G. Martin-Cocher, and L. Song, *Multi-Level Significant Maps for Large Transform Units*, JCTVC-G644, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Nov. 2011.

[19] G. Korodi, J. Zan, and D. He, *Encoding and Decoding Significant Coefficient Flags for Small Transform Units Using Partition Sets*, JCTVC-G657, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Nov. 2011.

[20] T. Kumakura and S. Fukushima, *Non-CE3: Simplified Context Derivation for Significance Map*, JCTVC-I0296, 9th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Apr.–May 2012.

[21] J. Sole, R. Joshi, and M. Karczewicz, *Removal of* 8×2 / 2×8 *Coefficient Groups*, JCTVC-J0256, 10th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Stockholm, Sweden, Jul. 2012.

[22] T. Nguyen, *CE11: Coding of Transform Coefficient Levels with Golomb-Rice Codes*, JCTVC-E253, 5th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Mar. 2011.

[23] Y. Piao, Y. Hong, I.-K. Kim, and J. H. Park, *Cross-Check Results for JCTVC-J0228*, JCTVC-J0408, 10th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Stockholm, Sweden, Jul. 2012.

[24] W.-J. Chien, M. Karczewicz, J. Sole, and J. Chen, *On Coefficient Level Remaining Coding*, JCTVC-I0487, 9th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Apr.–May 2012.

[25] G. Clare, F. Henry, and J. Jung, *Sign Data Hiding*, JCTVC-G271, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Nov. 2011.

[26] X. Yu, J. Wang, D. He, G. Martin-Cocher, and S. Campbell, *Multiple Sign Bits Hiding*, JCTVC-H0481, 8th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, San Jose, CA, Feb. 2012.

[27] X. Yu, J. Wang, D. He, G. Martin-Cocher, and S. Campbell, *Simplification of Multiple Sign Bits Hiding*, JCTVC-I0156, 9th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Apr. 2012.

[28] E.-H. Yang and X. Yu, "Rate distortion optimization for H.264 interframe coding: A general framework and algorithms," *IEEE Trans. Image Proces.*, vol. 16, no. 7, pp. 1774–1784, Jul. 2007.

[29] M. Karczewicz, Y. Ye, and I.-S. Chong, *Rate Distortion Optimized Quantization*, VCEG-AH21, Antalya, Turkey, Jan. 2008.

[30] G. Clare and F. Henry, *AHG 13: Proposed Bugfix for Tickets 410 and 470 Related to Lossless Coding* , JCTVC-I0529, 9th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Apr. 2012.

[31] K. McCann, W.-J. Han, I.-K. Kim, J.-H. Min, E. Alshina, A. Alshin, T. Lee, J. Chen, V. Seregin, S. Lee, Y.-M. Hong, M.-S. Cheon, and N. Shlyakhov, *Samsung's Response to the Call for Proposals on Video Compression Technology*, JCTVC-A124, 1st Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Dresden, Germany, Apr. 2010.

[32] F. Bossen, *HM8 Common Test Conditions and Software Reference Configurations*, JCTVC-J1100, 10th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Stockholm, Sweden, Jul. 2012.

[33] G. Bjøntegaard, *Improvements of the BD-PSNR Model*, ITU-T SG16 Q.6, document VCEG-AI11, Berlin, Germany, Jul. 2008.

[34] J. Lainema, K. Ugur, and A. Hallapuro, *Single Entropy Coder for HEVC With a High Throughput Binarization Mode*, JCTVC-G569, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Nov. 2011.

[35] A. Dueñas, *BoG report on High Throughput Binarization Schemes for CABAC*, JCTVC-H0728, 8th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, San Jose, CA, Feb. 2012.

[36] F. Bossen, *On Software Complexity*, JCTVC-G757, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Nov. 2011.

[37] K. McCann, J. Choi, and J. H. Park, *HEVC Software Player Demonstration on Mobile Devices*, JCTVC-G988, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, Nov. 2011.

[38] K. Veera, R. Ganguly, B. Zhou, N. Kamath, S. Chowdary, J. Du, I. S. Chong, and M. Coban, *A Real-Time ARM HEVC Decoder Implementation*, JCTVC-H0693, 8th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, San Jose, CA, Feb. 2012.

[39] F. Bossen, *On Software Complexity: Decoding 720p Content on a Tablet*, JCTVC-J0128, 10th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Stockholm, Sweden, Jul. 2012.

Joel Sole (M'01) received the M.Sc. degree in telecommunications from the Technical University of Catalonia (UPC), Barcelona, Spain, and Télécom ParisTech, Paris, France, and the Ph.D. degree from UPC in 2006.

From 2006 to 2010, he was with Thomson Corporate Research, Princeton, NJ, initially as a Post-Doctoral Fellow and later as a Staff Member and Senior Scientist. He is currently a Staff Engineer with Qualcomm, San Diego, CA.

Rajan Joshi (M'95) received the B.Tech. degree in electrical engineering and the M.Tech. degree in communications engineering from the Indian Institute of Technology Bombay, Mumbai, India, in 1988 and 1990, respectively, and the Ph.D. degree in electrical engineering from Washington State University, Pullman, in 1996.

In 1995, he was with the Xerox Palo Alto Research Center, Palo Alto, CA. From 1996 to 2006, he was a Senior Research Scientist with Eastman Kodak Company, Rochester, NY. From 2006 to 2008, he was a Senior Technical Staff Member with Thomson Corporate Research, Burbank, CA. He is currently a Senior Staff Engineer with Qualcomm, San Diego, CA. His current research interests include video and image coding, video processing, and information theory.

Nguyen Nguyen received the M.A.Sc. degree from the University of Waterloo, Waterloo, ON, Canada, in 2005.

In 2005, he joined SlipStream Data, Inc., Waterloo, as a Performance Engineer, focusing on optimized network protocols and lossless compression algorithms. In 2006, he joined Research in Motion Ltd. (RIM), Waterloo, ON, Canada, as a part of its acquisition of SlipStream Data, Inc. Between 2006 and 2010, he designed and developed compression algorithms tailored for the BlackBerry platform. He is currently a member of the Compression Research Group, RIM, focusing on the development of HEVC. His current research interests include video coding and compression, lossless compression, and network protocols.

Tianying Ji received the Bachelors, Masters, and Ph.D. degrees from the Department of Electrical Engineering, Tsinghua University, Beijing, China, in 1989, 1991, and 1995, respectively.

In 1995, he joined the University of Waterloo, Waterloo, ON, Canada, as a Post-Doctoral Fellow, where his research was focused on speech recognition. He has been with companies such as Lucent and LSI Logic in DSP and video processor design. In 2008, he joined Research in Motion Ltd. (RIM), Waterloo, ON, Canada. Initially, he optimized a real-time H.264 decoder on a BlackBerry device, and developed multimedia drivers for the BlackBerry operating system. He is currently a member of the Compression Research group at RIM, focusing on the development of HEVC. His research interests include video coding and compression, embedded systems, and DSP applications.

Marta Karczewicz received the M.Sc. and Ph.D. (Hons.) degrees from the Tampere University of Technology, Tampere, Finland, in 1994 and 1997, respectively.

From 1996 to 2006, she was with Nokia Research, Tampere. She is currently the Senior Director of the Multimedia Research and Development and Standards Group, Qualcomm, San Diego, CA.

Gordon Clare received the M.Sc. degree from the University of Auckland, Auckland, New Zealand, in 1984.

From 1985 to 1989, he was a Development Engineer with Rakon Computers, Sydney, Australia. From 1989 to 1991, he was a Team Leader with Software Development International, Sydney, creating a network management system for fault tolerant environments. From 1991 to 1997, he was with CISRA, a Canon Research Center, Sydney, focusing on real-time hardware and software image processing solutions. He has been with several international companies working on developing document management systems and search engines in France. From 2005 to 2010, he was a Consultant, developing H.264 and SVC solutions at Canon and Orange Labs, Issy-les-Moulineaux, France. He is currently with France Telecom Orange. His current research interests include HEVC standardization and development.

Félix Henry received the Dipl.-Ing. (M.Sc.) degree in telecommunications from Telecom ParisSud, Paris, France, in 1993, and the Ph.D. degree from Telecom ParisTech, Paris, in 1998 in the domain of wavelet image coding.

He started his career in 1995 with the Canon Research Center, France, where he worked on still image compression and video coding. He participated actively in JPEG2000 standardization and is a co-inventor of more than 100 patents in the domain of image and video processing. He is currently working as a Video Coding Project Leader with Orange Labs, Issy-les-Moulineaux, France. His current research interests include development of HEVC, transform coefficient coding, and high level parallel processing.

Alberto Dueñas received the M.Sc. degree in telecommunications from the Technical University of Madrid, Madrid, Spain, in 1995.

In 1996, he was with DMV-NDS-Tandberg Television, working on audio and video compression research, including development of AAC and MPEG-4 specifications. From 2000 to 2004, he developed a real-time MPEG-4 video compression product at PRODYS. From 2004 to 2005, he architected a high-performance systems-on-a-chip solution for imaging and vision applications at AnaFocus, Seville, Spain. From 2005 to 2008, he developed communications H.264/AVC based low latency products at W&W. From 2008 to 2012, he continued the development of wireless video compression chips and systems at Cavium, San Jose, CA. In 2012, he developed network-based panoramic cameras at Altia Systems and Co-Founded NGCodec to develop HEVC products. He is currently involved in the development of HEVC. His current research interests include high throughpout entropy coding, low latency, high level parallel processing, and error resilience.