Entropy Coding of Syntax Elements Related to Block Structures and Transform Coefficient Levels in HEVC

Tung Nguyen\textsuperscript{a}, Philipp Helle\textsuperscript{a}, Martin Winken\textsuperscript{a}, Detlev Marpe\textsuperscript{a}, Heiko Schwarz\textsuperscript{a}, and Thomas Wiegand\textsuperscript{a,b}

\textsuperscript{a}Image and Video Coding Group – Image Processing Department, Fraunhofer Institute for Telecommunications – Heinrich Hertz Institute, Einsteinufer 37, 10587 Berlin, Germany; \textsuperscript{b}Image Communication Chair – School of EE & CS, Berlin Institute of Technology, Einsteinufer 17, 10587 Berlin, Germany

ABSTRACT

The most recent video compression technology is High Efficiency Video Coding (HEVC). This soon to be completed standard is a joint development by Video Coding Experts Group (VCEG) of ITU-T and Moving Picture Experts Group (MPEG) of ISO/IEC. As one of its major technical novelties, HEVC supports variable prediction and transform block sizes using the quadtree approach for block partitioning. In terms of entropy coding, the Draft International Standard (DIS) of HEVC specifies context-based adaptive binary arithmetic coding (CABAC) as the single mode of operation. In this paper, a description of the specific CABAC-based entropy coding part in HEVC is given that is related to block structures and transform coefficient levels. In addition, experimental results are presented that indicate the benefit of the transform-coefficient level coding design in HEVC in terms of improved coding performance and reduced complexity.

Keywords: Transform coding, entropy coding, CABAC, HEVC, video compression

1. INTRODUCTION

Just like its successor H.264/AVC, High Efficiency Video Coding (HEVC) is a hybrid video-compression scheme consisting of block-based prediction followed by block-based transform coding of the prediction residual.\textsuperscript{1} In HEVC, however, the block-based paradigm has undergone an evolution in terms of increased flexibility of block partitioning both for prediction and transform coding by incorporating proposals which are based on suitable quadtree-based techniques.\textsuperscript{2–5}

HEVC specifies context-based adaptive binary arithmetic coding (CABAC) as the single entropy coding mode. This is in contrast to H.264/AVC, where CABAC has a coexistence with the less computationally complex but also less efficient context-adaptive variable-length coding (CAVLC). Similar to what is specified in H.264/AVC, CABAC in HEVC involves three successive stages which are, from an encoder perspective, binarization, context modeling, and binary arithmetic coding (BAC).\textsuperscript{6} The BAC engine of H.264/AVC has been adopted for HEVC without any modifications, including probability estimation and the low-complexity bypass mode. However, with transform block sizes of up to 32×32, adjustments to the original binarization and context modeling stages of CABAC in H.264/AVC were inevitable. These adjustments were primarily made in order to reduce complexity in many aspects such as, e.g., reduced number of context models, coherent logic for all transform block sizes, and increased throughput. Moreover, the use of local templates for context modeling is intended to improve coding efficiency relative to a straightforward extension of the CABAC scheme in H.264/AVC for transform blocks larger than 8×8.

The aim of the presented paper is to give a detailed description of entropy coding of syntax elements related to block structures and transform coefficient levels in HEVC. Basis for this description is the Draft International
Standard (DIS) of HEVC. For that, Section 2 briefly outlines CABAC in HEVC followed by Section 3 with a description of block structures and CABAC-based coding of the related structural data. Section 4 describes the transform-coefficient level coding in depth. Finally, Section 5 presents experimental results. Section 6 concludes this paper.

2. ENTROPY CODING

The core of CABAC is the binary arithmetic coding engine (BAC) also known as M Coder. While the BAC engine in HEVC is the same as specified in H.264/AVC, binarization and context modeling have been adjusted for transform coefficient levels. In contrast to the fixed binarization scheme in H.264/AVC, the binarization process for transform coefficient levels in HEVC is adaptive. Furthermore, transform blocks are decomposed into smaller sub-blocks and local templates are employed for context modeling.

Binarization is the decomposition of non-binary symbols into a sequence of bins (bin string). Examples for binarization schemes are fixed-length codes, Golomb-Rice codes, and Exp-Golomb codes. Note that the so-called (truncated) unary code is exactly the 0-th order Golomb-Rice code. Concatenation of different binarization schemes can be applied. An example is the binarization process of transform coefficient levels in HEVC. This process involves three different schemes: truncated unary, Golomb-Rice codes, and Exp-Golomb codes.

Bins are coded either in regular or low-complexity bypass mode of BAC. A binary probability model has to be specified when coding a bin in regular mode. This so-called context model is adaptive and the internal probability representation is updated after the coding of each bin. In general, different sets of context models are used for different syntax elements S. For the remaining description in this paper, \( \chi_S \) shall denote the context model of a given set of context models for the syntax element S.

3. BLOCK STRUCTURES

The block-based partitioning concept in HEVC is based on the division of an input picture into regular square blocks called coding tree blocks (CTBs). Each CTB serves as the root for the coding quadtree whose leaves are referred to as coding blocks (CBs) as shown on the left in Figure 1. Each CB determines the prediction mode, i.e., either spatial (intra) prediction or temporal (inter) prediction. CBs serve as starting point for the prediction block (PB) structure as well as for the transform block (TB) structure, the so-called residual quadtree (RQT), see the examples on the right in Figure 1.

Separate prediction parameters are transmitted for each PB such as, e.g., intra prediction modes or motion parameters. The PB structure is such that a CB can consist of one PB, two rectangular PBs or four square PBs. In contrast to that, the RQT structure is a multi-level quadtree similar to the coding quadtree, and thus more flexible than the PB partitioning with its maximum depth of one. The leaves of the RQT are the transform blocks and a separable 2-D transform of the same size as the edge length of the given TB is applied. A Z-scan defines the processing order for all structures, i.e., it consists of a left-to-right and depth-first traversal as shown in Figure 1 by ascending numbers.

All color components use the single transmitted tree syntax in the current HEVC design. Due to this fact, each partition type forms a unit describing the usage for the luma as well as for both chroma components. Therefore, all CTBs of luma and chroma samples together with their corresponding syntax structure are collectively referred to as coding tree units (CTUs). Correspondingly, PBs and TBs of luma and chroma samples together with their associated syntax structures are called prediction units (PUs) and transform units (TUs), respectively.

The coding tree structure is specified by the syntax element split coding unit flag \( \omega_{\text{splitCu}} \). When \( \omega_{\text{splitCu}} \) is equal to one, the current prediction block is further subdivided using the quadtree approach. No further subdivision is specified with \( \omega_{\text{splitCu}} = 0 \). This syntax element is coded in regular mode of BAC and the context model selection involves the local neighborhood of the corresponding block. Figure 1 shows an example with the current CB has the number 23. Consequently, the above neighbor and the right neighbor are CB 19 and CB 14. The depth of the current block is compared to the depths of the left and the above neighbors. A different context model is selected if one of the neighbors has a larger depth, i.e., the neighboring CU is smaller than the current block. Furthermore, another context model is used when both neighbors have larger depths.
Figure 1. Left Block: Example for a subdivision of a $64 \times 64$ coding tree block (CTB) into coding blocks (CBs) by using the corresponding coding tree. Furthermore, the left block shows the concept of the local template for context model selection depending on CBs lying above and left relative to the current CB. The block denoted with number 23 (CB 23) shall be the current CB, then CB 14 is the left and CB 19 is the above neighbor. Right Blocks: As an example of further subdivisions, the $32 \times 32$ CB 7 is partitioned into two prediction blocks (above). In contrast, the residual quadtree (RQT) divides the prediction residual signal of CB 7 into variable transform block sizes (below). Z-scan with depth-first traversal defines the processing order for all structures. Ascending numbers used as labels highlight this aspect.

In contrast to the coding tree, the PB structure can be split only once. For this reason, the split flag and the shape type are combined into a single value called partition mode. The binarization of a partition mode $\omega_{\text{Part}}$ depends on the prediction mode and can consist of up to four bins. $\omega_{\text{Part}}$ is a split flag for intra predicted CBs as the PB structure can be either a PB with the same size as the CB or four square PBs. On the other hand, up to eight combinations of the split flag and the shapes are possible for inter predicted CBs due to rectangular shapes. In summary, each bin of $\omega_{\text{Part}}$ employs a separate context model except the last bin, which is coded in the low-complexity bypass mode of BAC.

On the contrary to the PB structure, the partitioning for the transform using the RQT is flexible as for the coding tree. Further subdivisions into smaller TBs are controlled by the split transform flag $\omega_{\text{splitRqt}}$. Note that in HEVC, transforms are ranged from $4 \times 4$ dyadically up to $32 \times 32$. $\omega_{\text{splitRqt}} = 0$ results in a leaf that forms a TB while $\omega_{\text{splitRqt}} = 1$ specifies a further subdivision. This syntax element uses four context models, each coupled with a different TB size.

Although the generic quadtree approach of the RQT is not limited, additional structure parameters are transmitted in the sequence parameter set header. These parameters are the maximum depth, the minimum allowed transform size and the maximum allowed transform size. In addition to these structure parameters, insignificance flags are transmitted. These insignificance flags specify the non-existence of non-zero transform coefficient levels. The next section describes their relationship to the coding of transform coefficient levels.

4. TRANSFORM COEFFICIENT LEVELS

Further subdivisions of TBs into $4 \times 4$ sub-blocks (SBs) form the core concept of transform-coefficient level coding in HEVC. This approach enables the use of the same logic and processing steps for all TB sizes.

After the decomposition into SBs, scan patterns are employed to specifying the sequential processing order of the transform coefficient levels. For that, two scan patterns are used with the first scan pattern specifies the processing order of the SBs and the second scan pattern specifies the processing order of the frequency positions.
within each SB. In HEVC, both scan patterns are equal and each SB is processed completely, i.e., the complete level information are transmitted for a SB before processing the next SB within a TB.

A scan pattern is basically a mapping from the 2D-matrix to a 1D-vector. This mapping defines the sequential processing order of the 2D-matrix. The so-called bottom-left to up-right diagonal scan defines the processing order of SBs as well as frequency positions within SBs (on the right in Figure 2). But unlike H.264/AVC, reverse diagonal scan pattern is used in HEVC, i.e. flipping the direction from up-right to bottom-left. As a special case, predefined intra prediction modes employ the horizontal or the vertical scan pattern. These intra directional modes are ranged from 22 to 30 for the horizontal scan pattern and from 6 to 14 for the vertical scan pattern, both inclusively (on the left in Figure 2). However, these scan patterns are specified for 4×4 and 8×8 luma TBs only. In general, smaller TB sizes have better spatial support and lower frequency resolution. This leads to the effect of having a concentration of levels next to the edges of small intra-predicted TBs. For this reason, the directional scan patterns are used as they can reduce the number of coded transform coefficient levels equal to zero.

Additional side information specify the insignificance in different hierarchy levels are transmitted. Particularly, an insignificance specifies the existence of zero sample values, i.e. prediction residuals or transform coefficient levels. In the highest hierarchy level, i.e. the RQT root, the no residual syntax flag $\omega_{\text{noRes}}$ is coded. $\omega_{\text{noRes}} = 1$ specifies that all residual sample values covered by the CB are zero. This flag is inferred to be zero when using intra prediction as the probability for $\omega_{\text{noRes}} = 1$ in that case is extremely low. For each RQT leaf, the coded block flag $\omega_{\text{cbf}}$ specifies insignificance for the whole TB. An exceptional case is given for the chroma components where $\omega_{\text{cbf}}$ is coded interleaved with $\omega_{\text{splitRqt}}$. Before processing a TB, the last significant scan position coded as $(x, y)$ offset from the DC frequency position specifies the insignificance for a partial area of the TB. And finally in the lowest hierarchy level, the coded sub-block flag $\omega_{\text{csf}}$ specifies insignificance for a SB. To this end, the significance map specifies insignificant frequency positions within each SB. For that, the significance flag $\omega_{\text{sig}}$ specifies whether a frequency position has non-zero level $\omega_{\text{sig}} = 1$ or not $\omega_{\text{sig}} = 0$.

The absolute level magnitudes and the signs are coded after the specification of the significance map. Up to three different syntax elements describing the remaining level magnitude are coded. Figure 3 gives an example for the five stages using for transform coefficient level coding of a 4×4 TB. In the following, detailed description is given on the context modeling and further aspects for each coding stage.
4.1 Coding of Insignificance Flags

The first insignificance flag $\omega_{\text{noRes}}$ uses one single context model. For $\omega_{\text{cbf}}$, the luma component and the chroma components use different context modeling as well as different context model sets. In the case of luma, $\chi_{\text{cbf}}$ is different for the RQT root and for the remaining transform tree depths. For the chroma components, $\chi_{\text{cbf}}$ is equal to the current tree depth. The context modeling of $\omega_{\text{last}}$ is summarized in Equation (1) with $i$ is the bin index of the bin string. Note that $\omega_{\text{last}}$ represents the offset in $x$ and then in $y$, respectively.

$$\chi_{\text{last}} = \frac{i}{2^n} \quad (1)$$

The quantization step size $q$ depends on the color component and is smaller for luma than for chroma. Furthermore, luma and chroma components use different context model sets. Finally, the last insignificance flag in the hierarchy $\omega_{\text{csf}}$ uses the local template based on already coded SBs (above left in Figure 4). The right and the below SBs are evaluated and equation (2) summarizes the rule.

$$\chi_{\text{csf}} = \max(1, \omega_{\text{csf}}^R + \omega_{\text{csf}}^B) \quad (2)$$

4.2 Coding of Significance Map

An $\omega_{\text{sig}} = 0$ specifies the insignificance for a frequency position while $\omega_{\text{sig}} = 1$ also specifies that an absolute level is at least one. The above center of Figure 3 shows an example for a significance map. Regarding the context modeling, a fixed assignment to context models is used depending on the spatial location relative to the top-left corner of the SB. The mapping for a $4 \times 4$ TB is shown in the above center of Figure 4. For TBs larger than $4 \times 4$, one out four mappings or patterns is selected. These four mappings are shown in Figure 4 denoted by $p = 0 \ldots 3$. 

![Table and Diagram](image-url)
The mapping selection depends on the already coded SBs as for $\omega_{csf}$. Let $p(\cdot)$ denotes the pattern and $p_{(i,j)}$ denotes the context model assignment at the local frequency position $(i,j)$. Furthermore, $\omega^R_{csf}$ denotes the coded sub-block flag for the right neighboring SB and $\omega^B_{csf}$ the bottom, respectively. Then, equation (3) summarizes the context modeling for $\omega_{sig}$.

$$\chi_{sig} = p_{(i,j)}(\omega^R_{csf} + 2 \cdot \omega^B_{csf})$$

4×4 TBs use a different context model set than 8×8 TBs while 16×16 and 32×32 TBs share one set of context models. In addition, different context sets are used for the luma and the chroma components as well as for 8×8 luma TBs using a directional scan pattern. Furthermore, the DC frequency position uses a single context model independent from the TB size. And finally, the SB covering the DC frequency position uses different sets of context models.

### 4.3 Coding of Magnitudes and Signs

The level magnitudes are transmitted using up to three different syntax elements. Firstly, the $\omega_{gr1}$ flags specify if significant scan positions are greater than one. Secondly, the $\omega_{gr2}$ flag specifies if the first significant scan position with a coded $\omega_{gr1} = 1$ is greater than two. Thirdly, the signs are coded in the low-complexity bypass mode of BAC. Finally, $\omega_{rem}$ specify the remaining level magnitudes using the low-complexity bypass mode of BAC.

Figure (3) shows a detailed example for the four processing stages: $\omega_{gr1}$ above right, $\omega_{gr2}$ below left, $\omega_{sign}$ below center, and $\omega_{rem}$ below right. The $\omega_{gr1}$ flags are specified for up to eight significant scan positions and the $\omega_{gr2}$ flag is coded for the first occurrence of $\omega_{gr1} = 1$ only. This is because of the adaptive binarization threshold resulting in a higher throughput. Consequently, $\omega_{rem}$ specifies the remaining absolute level and depending on whether or not $\omega_{gr1}$ and $\omega_{gr2}$ have been coded.

The context model of $\omega_{gr1}$ for the first scan position in a SB is zero. For each coded $\omega_{gr1} = 0$, $\chi_{gr1}$ is incremented by one but limited to three. When the first $\omega_{gr1} = 1$ has been coded, $\chi_{gr1} = 0$ for the remaining scan positions in the SB. For $\omega_{gr2}$, only one context model is used as this flag is coded once only. Context sets
Figure 5. Binarization process of transform coefficient levels using three different schemes with variable thresholds (above). The dynamic change of the first threshold depending on the coded levels $z$ for the used example in Figure 3. As the reverse diagonal scan pattern is used, the first value enters the binarization process is 1 and the last value is -12 in this example.

are derived for $\omega_{gr1}$ and $\omega_{gr2}$ in the same way. A different context set is selected when at least one $\omega_{gr1} = 1$ occurred in the previous SB. Furthermore, the SBs covering the DC frequency positions use different context sets.

In the next coding stage, the signs are coded in the low-complexity mode of BAC. When the optionally Sign Inference technique is enabled, the sign for the last scan position in reverse scan order may be inferred. For that, the distance between the first and the last significant scan position in a SB has to be checked. In the case of a difference in scan positions larger than three, the sign for the last scan position in reverse scan order is inferred from the parity of the absolute level sum.

Finally, the remaining absolute magnitudes $\omega_{rem}$ are coded in the low-complexity bypass mode of BAC. The syntax element $\omega_{rem}$ is basically the remaining part of the bin string resulting from the binarization process of transform coefficient levels. The relationship between the syntax elements and the bin string is described in the following subsection.

### 4.4 Binarization of Levels

The binarization process for transform coefficient levels consists of three concatenated schemes. Truncated unary is followed by Golomb-Rice and Exp-Golomb codes. A transform coefficient level $z$ lying in the range $[0, b_0]$ can be represented by a truncated unary code. If $z$ exceeds the threshold $b_0$, the value of $z - b_0$ is binarized with the second scheme. Furthermore, if the $z$ exceeds the threshold $b_1$, the value of $z - b_1$ is binarized using the third scheme (above in Figure 5).

The second variable threshold $b_1$ depends on the Golomb-Rice parameter $k$. The relation is $b_1 = 3 \cdot 2^k + b_0$. In contrast, the first variable threshold $b_0$ depends on already coded absolute level values. Starting with three before processing a SB, $b_0$ is decreased to two when the first value greater than two has been binarized. This threshold is further decreased to one when eight values greater than one have been binarized. As a result, the number of bins coded in the regular mode of BAC is decreased. For the Golomb-Rice scheme, the order $k$ for the binarization has to be specified. This parameter $k$ is zero before processing a SB and is increased by one after the binarization of a $z$ larger than $3 \cdot 2^k$. Furthermore, $k$ is limited to four and the Exp-Golomb order is always $k + 1$.

CABAC in H.264/AVC requires up to 22.5 bins in regular mode for one sample (4:2:0 subsampled chroma). Compared to that, this theoretical worst-case limit is reduced by 80% when setting the first bound $b_0$ equal
to three. Making $b_0$ adaptive, the theoretical worst-case limit is further reduced to 90% relative to the fixed binarization scheme of CABAC in H.264/AVC.

5. EXPERIMENTAL RESULTS

Simulations were conducted in order to evaluate the impact of the transform coefficient level coding design. The reference implementation of HEVC in version 6.0 was used as simulation software. Parameters were adjusted along the so-called Common Test Conditions. Three test configurations were simulated as described in the Common Test Conditions. In this section, $AI$ is the abbreviation for the All Intra test configuration, $RA$ the abbreviation for the Random Access test configuration, and $LD$ the abbreviation for the Low Delay test configuration. Results are presented in terms of the Bjøntegaard Delta bit rate (BD-rate) for the luma component.

Test sequences are divided into classes labeled from A to F. Each class contains test sequences with the same spatial resolution excepting class F. In contrast to the other classes, class F contains test sequences with screen and graphical content but may differing in their spatial resolution. Detailed description can be found in the Common Test Conditions.

The transform coefficient level coding specified for CABAC in H.264/AVC was extended to TBs larger than $8 \times 8$ in a straightforward manner. No change was required for the binarization process. Only the context modeling for $\omega_{\text{sig}}$ was changed as follows. CABAC in H.264/AVC assigns a separate context model for each scan position in a $4 \times 4$ TB. Four successive scan positions share one context model in a $8 \times 8$ TB. Both cases result in 16 context models for each TB size. Formula (4) summarizes the context modeling for $\omega_{\text{sig}}$ for $4 \times 4$ and $8 \times 8$. In (4), $i$ denotes the current scan position, $n = \log_2 N$ and $N$ is the current TB size.

$$\chi_{\text{sig}} = \left\lfloor \frac{i}{2^{2n-4}} \right\rfloor$$

The denominator is equivalent to a quantization step size. This is controlled by the TB size and the quantization step size. This number of successive scan positions sharing one context model. Thus, (4) can be applied directly to larger TBs resulting in 16 context models for each block size. Note that the context modeling for $\omega_{\text{last}}$ is exactly the same as for $\omega_{\text{sig}}$ in CABAC of H.264/AVC.

5.1 Coding Performance

Table 1 summarizes the BD-rate performance for three different experiments. All BD-rate results are relative to a configuration using the straightforward extension as described before. The results headed by Local Templates were achieved by the default configuration without the additional Sign Inference and Adaptive Scan techniques. Hence, these results show the improvement of the design using SBs and local templates. Averaged bit rate savings up to 6.45% was observed (class F). For the second experiment, the Sign Inference technique was additionally enabled (results headed by Sign Inference). This technique further improves the coding performance for about 0.68% averaged over all three test configurations. Finally, the Adaptive Scan was additionally enabled (results headed by Adaptive Scan). This configuration is equivalent to the default configuration of the reference software. An additional gain of about 0.66% was observed averaged over all three test configurations. In summary, the most significant improvement comes from the local templates for context modeling. A gain of 2.84% averaged over all three test configurations was observed.

5.2 Complexity

Improvements in terms of complexity were achieved by three adjustments. Firstly, the decomposition into SBs allows the same processing steps applied to all TB sizes. Secondly, the context memory is significantly reduced relative to the straightforward extension. Context memory is an important factor affecting the area cost and the critical path length in hardware implementations. Thirdly, the number of regular coded bins is significantly reduced relative to the fixed binarization scheme of CABAC in H.264/AVC. A reduction of regular coded bins

*At the time of submission of this paper, the reference software that corresponds to the DIS status of HEVC is still under development. Presented results conform to the Committee Draft status of HEVC.*
Table 1. BD-rate performance of different techniques for transform coefficient level coding in HEVC relative to the straightforward extension of the scheme specified for CABAC in H.264/AVC extended to 16×16 and 32×32 TBs. Results headed by Local Templates show the improvement of the design in HEVC without the additional techniques Sign Inference and Adaptive Scan, i.e. using the decomposition into SBs and local templates for context modeling only. The inference of a sign is enabled additionally and the results are headed by Sign Inference. Finally, results headed by Adaptive Scan show the improvement when additionally enabling the Adaptive Scan.

<table>
<thead>
<tr>
<th></th>
<th>Local Templates</th>
<th>Sign Inference</th>
<th>Adaptive Scan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AI</td>
<td>RA</td>
<td>LD</td>
</tr>
<tr>
<td>Class A</td>
<td>-2.89</td>
<td>-1.92</td>
<td>-</td>
</tr>
<tr>
<td>Class B</td>
<td>-2.79</td>
<td>-2.28</td>
<td>-2.33</td>
</tr>
<tr>
<td>Class C</td>
<td>-2.01</td>
<td>-1.90</td>
<td>-2.39</td>
</tr>
<tr>
<td>Class E</td>
<td>-2.97</td>
<td>-</td>
<td>-2.23</td>
</tr>
<tr>
<td>Class F</td>
<td>-6.45</td>
<td>-5.51</td>
<td>-4.80</td>
</tr>
<tr>
<td>Average</td>
<td>-3.22</td>
<td>-2.62</td>
<td>-2.68</td>
</tr>
</tbody>
</table>

Table 2. Number of context models for transform coefficient level coding in CABAC: For the straightforward extension scheme (first row), for CABAC in HEVC (second row), relative reduction values (third row).

<table>
<thead>
<tr>
<th></th>
<th>$\omega_{\text{csf}}$</th>
<th>$\omega_{\text{last}}$</th>
<th>$\omega_{\text{sig}}$</th>
<th>$\omega_{\text{gr1}}$</th>
<th>$\omega_{\text{gr2}}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264/AVC</td>
<td>-</td>
<td>112</td>
<td>112</td>
<td>35</td>
<td>35</td>
<td>294</td>
</tr>
<tr>
<td>HEVC</td>
<td>4</td>
<td>36</td>
<td>42</td>
<td>24</td>
<td>6</td>
<td>112</td>
</tr>
<tr>
<td>Reduction</td>
<td>-</td>
<td>68%</td>
<td>62%</td>
<td>31%</td>
<td>83%</td>
<td>62%</td>
</tr>
</tbody>
</table>

resulting in enlargement of bypass coded bins. As the coding of bins in bypass mode is less complex than in regular mode, a higher amount of bins can be processed for a given time constant. Consequently, the throughput of CABAC is increased.

Table 2 shows the number of context models for transform coefficient level coding in CABAC. Interestingly, only one third context models are required resulting in a coding gain for about 2.84% as shown before.

<table>
<thead>
<tr>
<th>Regular Bins per Sample</th>
<th>Bypass Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H.264/AVC</td>
</tr>
<tr>
<td>Class A</td>
<td>4.49</td>
</tr>
<tr>
<td>Class B</td>
<td>4.47</td>
</tr>
<tr>
<td>Class C</td>
<td>5.35</td>
</tr>
<tr>
<td>Class D</td>
<td>5.88</td>
</tr>
<tr>
<td>Class E</td>
<td>3.22</td>
</tr>
<tr>
<td>Class F</td>
<td>3.15</td>
</tr>
<tr>
<td>Average</td>
<td>4.47</td>
</tr>
</tbody>
</table>

Table 3. Improvement from the adaptive Golomb-Rice binarization scheme gained through setting the quantization parameter equal to zero. The columns headed by H.264/AVC show the results for the straightforward extension. The columns headed by Golomb-Rice show the results when using adaptive Golomb-Rice binarization with a fixed threshold $b_0 = 3.8$. Finally, the columns headed by Adaptive $b_0$ show results for the adaptive threshold $b_0$ as described in Section 4.4. Regular Bins per Sample: Averaged number of regular coded bins per sample. Bypass Ratio: The ratio between bypass and regular coded bins.
The impact of the adaptive Golomb-Rice binarization was verified practically. An experiment was conducted in which the quantization parameter is set equal to zero. Only the AI test configuration was simulated. The highest bit rates can be generated by such a configuration. Therefore, this type of experiment shows the worst-case scenario for the used test set. Table 3 and 4 summarize the impact from the adaptive Golomb-Rice binarization. The results show a reduction of regular coded bins as well as a reduction of the total number of bins. Interestingly, the result for the adaptive threshold $b_0$ is extremely close to the theoretical bound. In contrast, the reduction of the total number of bins mainly comes from the adaptive binarization using Golomb-Rice codes. Note that the results include bins for the insignificance flags.

<table>
<thead>
<tr>
<th></th>
<th>Reduction Regular Bins</th>
<th>Reduction Total Bins</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H.264/AVC</td>
<td>Golomb-Rice</td>
</tr>
<tr>
<td>Class A</td>
<td>100.0%</td>
<td>39.19%</td>
</tr>
<tr>
<td>Class B</td>
<td>100.0%</td>
<td>36.90%</td>
</tr>
<tr>
<td>Class C</td>
<td>100.0%</td>
<td>33.52%</td>
</tr>
<tr>
<td>Class D</td>
<td>100.0%</td>
<td>30.80%</td>
</tr>
<tr>
<td>Class E</td>
<td>100.0%</td>
<td>52.48%</td>
</tr>
<tr>
<td>Class F</td>
<td>100.0%</td>
<td>35.69%</td>
</tr>
<tr>
<td>Average</td>
<td>100.0%</td>
<td>38.10%</td>
</tr>
</tbody>
</table>

Table 4. Further results for the conducted experiment where the quantization parameter is set equal to zero. Reduction Regular Bins: The relative number of regular coded bins reduced relative to the straightforward extension scheme. Reduction Total Bins: The relative total number of bins reduced relative to the straightforward extension scheme.

6. CONCLUSION

This overview paper has highlighted two aspects of the entropy coding stage in HEVC. Firstly, the coding efficiency has been improved relative to a straightforward extension of the CABAC scheme in H.264/AVC. This is mainly due to the use of local templates for context modeling. Further improvements are achieved by an adaptive scan and the inference of transform coefficient level signs. Secondly, the computational complexity has been reduced. The decomposition of transform blocks into sub-blocks allows the same logic and procedures to be applied for all block sizes. The number of context models could be reduced by two third while maintaining the overall coding performance. And finally, the number of regular coded bins could be decreased significantly using the adaptive binarization for transform coefficient levels. All these complexity-reducing methods result in a higher throughput of CABAC in HEVC.

REFERENCES


