Signal Processing: Image Communication 44 (2016) 108–123

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

An enhanced entropy coding scheme for HEVC

Min Gao [a], Xiaopeng Fan [a,*], Debin Zhao [a], Wen Gao [a,b]

[a] *Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China*
[b] *School of Electronic Engineering and Computer Science, Peking University, Beijing 100000, China*

ARTICLE INFO

Article history:
Received 28 September 2015
Received in revised form
10 March 2016
Accepted 27 March 2016
Available online 29 March 2016

Keywords:
HEVC
CABAC
Entropy coding
Context modeling
Binary arithmetic coding

ABSTRACT

This paper presents a modification to Context-based Adaptive Binary Arithmetic Coding (CABAC) in High Efficiency Video Coding (HEVC), which includes an improved context modeling for transform coefficient levels and a binary arithmetic coding (BAC) engine with low memory requirement. In the improved context modeling for transform coefficient levels, the context model index for significance map is dependent on the number of the significant neighbors covered by a local template and its position within transform block (TB). To limit the total number of context models for significance map, TBs are split into different regions based on the coefficient position. The same region in different TBs shares the same context model set. For the first and second bins of the truncated unary scheme of absolute level minus one, their context model indices depend on the neighbors covered by a local template of the current transform coefficient level. Specifically, the context model index for the first bin is determined by the number of neighbors covered by the local template with absolute magnitude equal to 1 and larger than 1; for the second bin, its context model index is determined by the number of neighbors covered by the local template with absolute magnitude larger than 1 and larger than 2. Moreover, TB is also split into different regions to incorporate the coefficient position in the context modeling of the first bin in luma component. In the BAC engine with low memory requirement, the probability is estimated based on a multi-parameter probability update mechanism, in which the probability is updated with two different adaption speeds and use the average as the estimated probability for the next symbol. Moreover, a multiplication with low bit capacities is used in the coding interval subdivision to substitute the large look-up table to reduce its memory consumption. According to the experiments conducted on HM14.0 under HEVC main profile, the improved context modeling for transform coefficient levels achieves 0.8%, 0.6% and 0.4% bitrate reduction on average for all intra (AI), random access (RA) and low delay (LD) configurations, respectively; the BAC engine with low memory requirement achieves 0.7%, 0.6% and 0.5% bitrate reduction on average for AI, RA and LD configurations, respectively; the overall bitrate reduction achieved by the proposed two techniques is 1.4%, 1.1% and 0.9% on average for AI, RA and LD configurations, respectively.

1. Introduction

High Efficiency Video Coding (HEVC) [1] is the latest video coding standard. It provides approximately two times the compression efficiency of its predecessor H.264/AVC [2] without any detectable loss in visual quality [3]. One of the fundamental coding tools adopted in HEVC is the quadtree block partitioning structure [4], which is a flexible mechanism for splitting a picture into different processing units for prediction and residual coding. The basic processing unit in HEVC is coding tree unit (CTU). Inside CTU, a quadtree structure is built for partitioning of the CTU, and the leaf node of the quadtree is called a coding unit (CU). Each CU uses either intra or inter prediction mode and is subdivided into prediction units (PU) and the shape of PU is specified by the splitting type. After prediction, the CU residual is partitioned into transform units (TU) by another quadtree referred to residual quadtree (RQT) [5]. The CUs, PUs, and TUs encapsulate coding blocks (CBs), prediction blocks (PBs), and transform blocks (TBs), respectively, as well as the associated syntax elements. The size of TB can range from 4×4 to 32×32 for luma and from 4×4 to 16×16 for chroma.

For the transform coefficient levels (quantization indices associated with the quantized transform coefficients) in one TB, a sub-block based coding scheme [5,6] is designed in HEVC. In this coding scheme, a TB is partitioned into non-overlapped 4×4 blocks, which are referred to as sub-blocks (SBs) [5]. The scan of SBs inside a TB and the scan of coefficients within a SB are both

* Corresponding author.
E-mail addresses: mgao@hit.edu.cn (M. Gao), fxp@hit.edu.cn (X. Fan), dbzhao@hit.edu.cn (D. Zhao), wgao@jdl.ac.cn (W. Gao).

http://dx.doi.org/10.1016/j.image.2016.03.008
0923-5965/

diagonal [7]. Horizontal and vertical scans may also be used in the intra prediction case for 4×4 and 8×8 TBs.

Up to five scan passes [8,9] are applied to code the syntax elements for the transform coefficient levels within a given SB. In the first scan pass, the significance map is signaled to indicate the significance for a scanning position, which involves the syntax element *significant_coeff_flag*. In the second and third scan passes, the first and second bins of truncated unary scheme of the absolute level minus one are signaled to indicate whether the absolute magnitude of a transform coefficient level is larger than 1 and 2, respectively. The syntax elements *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* are involved in the second and third scan passes, respectively. In the fourth scan pass, the syntax element *coeff_sign_flag* is coded to signal the sign information of a transform coefficient level. In the final scan pass, the syntax element *coeff_abs_level_remaining* is coded if the absolute value of a transform coefficient level is larger than that is coded in the previous scan passes.

HEVC only specifies context-based adaptive binary arithmetic coding (CABAC) as the entropy coding method to code the defined syntax elements. CABAC includes three basic processing steps: binarization, context modeling and binary arithmetic coding (BAC). The BAC engine also known as M-coder [22] includes two operation modes: a regular mode and a bypass mode. The regular mode uses adaptive probability estimator to estimate the probability of a binary source, while the bypass mode uses a probability model with probability of 0.5. The context modeling stage provides the context models for bins coded with regular mode. In the rest of the section, we firstly present the related work to CABAC in HEVC; then we provide the motivation behind our work.

1.1. The related work to CABAC

In this subsection, we will present the related work to CABAC in HEVC from two respects: context modeling for transform coefficient levels and the probability estimator in BAC engine.

1.1.1. Context modeling for transform coefficient levels

As described in [11], a position-based context modeling for *significant_coeff_flag* was introduced in CABAC within H.264/AVC, in which the context model index is dependent on the coefficient position within 4×4 transform blocks. The position based context modeling was partly inherited for coding *significant_coeff_flag* within 4×4 TBs in HEVC with some modifications, which include grouping the coefficient positions according to their frequencies and sharing the same context model for the significant flags within a group [12]. A template-based context modeling was designed in [13,14] for *significant_coeff_flag* by considering the characteristics of larger TBs. In this scheme, the context model index for *significant_coeff_flag* is determined by the neighbors covered by a predefined template. Compared with the position based context modeling, the template based context modeling can provide better coding efficiency. However, it does not allow a high degree of parallelism, since the context model index derivation depends on the significance of neighbors immediately preceding the current transform coefficient level. The context modeling for *significant_coeff_flag* in the TBs larger than 4×4 in HEVC is both position and template based [15,16]. In these methods, a template is selected for the current SB based on the information from the neighboring lower and right SBs and the specific context is finally determined by the coefficient position within the current SB using the selected template.

For *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag*, the context modeling in the early stage is partially inherited from H.264/AVC with some modifications, which takes into account the local properties in larger TBs [13]. Later, a context modeling based on a predefined local template was proposed in [14], in which the context model index depends on the number of the significant neighbors and the absolute sum of the neighbors covered by the local template. Thus, it is required to code all syntax elements in each coding step for a transform coefficient level. Therefore, the value of the current syntax element determines the type of the next syntax element to be processed. For example, if *coeff_abs_greater1_flag* is equal to 1, then the next syntax element is *coeff_abs_greater2_flag*. Otherwise, the next syntax element is *coeff_sign_flag*. This kind of dependency will increase the speculative calculations in the parallel context processing [17], since various combinations must be taken into account. This is one of the reasons why HEVC applies up to five scan passes to a SB and separate the syntax elements into different scan passes. In the context modeling in HEVC, the context model index is dependent on the number of *coeff_abs_greater1_flag* equal to 1 in the preceding SB [18], the number of trailing ones and the number of coefficients with magnitude larger than 1 in the current SB.

In the recently published literature [19] on the context modeling for transform coefficient levels, the concept of the local template in [14] is still utilized. Several modifications are introduced to address large TBs and enhance the parallelism by breaking the context dependencies between different scan passes. More specifically, for *significant_coeff_flag*, the number of the significant neighbors covered by a local template is used as a context instead of the absolute sum of the neighbors covered by the local template. In addition, the TB size is used as an additional context of *significant_coeff_flag* to capture the characteristics of TBs with different size. Instead of the number of the significant neighbors and the absolute sum of the neighbors covered by the local template, the number of neighbors covered by the local template with absolute magnitude larger than 1 and 2 are used in the context model selection for *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag*, respectively.

1.1.2. Probability estimator in binary arithmetic coding engine

The context modeling in CABAC divides the syntax elements into several non-stationary binary sources with different statistical properties [21]. The probability of each non-stationary binary source is estimated by M-coder [22] through one state machine. However, in view of the coding efficiency, it is better to find a good trade-off between adaption speed and the precision of the probability estimation for each binary source [21]. This problem can be solved by utilizing several state machines with different adaption speeds and precision of probability estimation [23]. However, the method in [23] introduced some additional look-up tables and thus increases the memory consumption. Some look-up table free solutions were proposed in [20,21] based on virtual sliding window, in which a specific window length is selected according to the statistical properties of the corresponding binary source and the probability estimation is calculated based on the selected window length. Another look-up table free approach namely multi-parameter probability update was proposed in [24]. The main idea of this approach is to use two probability estimations with different adaption speeds and combine them as weighted average for next bin probability prediction. Compared with the methods in [20,21], the multi-parameter probability update in [24] can achieve more accurate probability estimation according to our experiments.

1.2. The motivation of our work

Compared with the prior video coding standards, HEVC greatly enhance the coding efficiency. However, there is still requirement to further improve the coding efficiency to establish the next generation video coding standard succeeding HEVC [10]. Toward

this goal, literatures [19] and [24] proposed a context modeling for transform coefficient levels and a probability estimator in BAC engine, respectively.

However, for the context modeling in [19], the number of the context models will increase as well as the memory consumption, since the TB size is taken into account in the context model selection for *significant_coeff_flag*. In addition, only the information from the current scan pass is used in [19] for the context model selection of *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag*, thus their statistical redundancies are not fully exploited. For the BAC engine in [24], a 288 Kb look-up table was introduced in the coding interval subdivision, thus the memory consumption is increased significantly. As described in [21], decreasing memory consumption of CABAC is also very important issue for the hardware implementation and this is why a number of literatures [25–27] for HEVC aim at minimizing the memory consumption.

To address these two problems, this paper proposes an improved context modeling for transform coefficient levels and a BAC engine with low memory requirement. In the improved context modeling for transform coefficient levels, the contexts of *significant_coeff_flag* consist of the number of significant neighbors covered by a local template and coefficient position within TB. To limit the total number of the context models used for *significant_coeff_flag*, TBs are split into different regions based on the coefficient position. The same region from different TBs shares the same context model set. The context model index of *coeff_abs_greater1_flag* is determined by the number of neighbors covered by a local template with absolute magnitude equal to 1 and larger than 1. Moreover, the coefficient position is incorporated in the context model selection of *coeff_abs_greater1_flag* in luma component. For *coeff_abs_greater2_flag*, its context model index is determined by the number of neighbors covered by a local template with absolute magnitude larger than 1 and larger than 2. Although the proposed context modeling for *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* introduces the context dependency between different scan passes, this context dependency can be reduced as much as possible by designing the context modeling in an appropriate way. In the BAC engine with low memory requirement, the multi-parameter probability update mechanism in [24] is used to estimate the probability for the next symbol. To reduce the memory consumption, a multiplication with low bit capacities is used in the coding interval subdivision instead of the large look-up table, since the multiplication cost can be comparable with the cost of using a look-up table in modern architectures [28].

Compared with CABAC in HEVC under main profile, the two techniques presented in this paper can achieve 1.4%, 1.1% and 0.9% bit rate reduction on average for all intra (AI), random access (RA) and low delay (LD) configurations, respectively. More specifically, the proposed context modeling for transform coefficient levels can achieve 0.8%, 0.6% and 0.4% bitrate reduction on average for AI, RA and LD configurations, respectively; the BAC engine with low memory requirement achieves 0.7%, 0.6% and 0.5% bitrate reduction on average for AI, RA and LD configurations, respectively.

The rest of the paper is organized as follows. Section 2 presents an overview of the entropy coding scheme in HEVC, which includes the context modeling for transform coefficient levels and the BAC engine. Section 3 describes the proposed modification to CABAC within HEVC in detail. In Section 4, the extensive experimental results are provided. Finally, Section 5 concludes the paper.

2. The entropy coding scheme in HEVC

In this section, we will describe the transform coefficient level coding and BAC engine in HEVC. For the transform coefficient level coding, we first give the overview of the coding procedure and then describe the context modeling for *significant_coeff_flag*, *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* in detail. For the BAC engine, the probability update mechanism and the coding interval subdivision are described.

2.1. Transform coefficient level coding in HEVC

2.1.1. Overview

The transform coefficient level coding in HEVC is based on the concept of sub-block [5,6], in which TBs larger than 4×4 are partitioned into non-overlapped SBs. Fig. 1 illustrates the partition of 8×8 TB into 4 SBs under different scan passes, in which different colors correspond to different SBs.

In HEVC, the effective intra and inter prediction techniques are used to remove the spatial and temporal redundancy, generating the prediction residual. Then the energy of the prediction residual is concentrated into a small number of transform coefficients. After quantization, a large portion of transform coefficients are quantized into zero. In other word, the quantized transform coefficients, also denoted as transform coefficient levels, within TB are often sparse. There are not any significant transform coefficient levels within entire TB in some situations. To exploit the features of the transform coefficient levels within TB, the syntax element *coded_block_flag* (*CBF*) is first coded to indicate the significance of the entire TB. For significant TB, the position of the last significant transform coefficient level in a TB is coded to indicate the significance for the partial area of TB [5]. The position of the last significant transform coefficient level is obtained following the forward scan order and is coded by explicitly signaling its (X, Y) coordinate relative to the top-left (DC) coefficient.

Next, the transform coefficient levels are coded. Since reverse scanning of the transform coefficient levels allows a more reliable estimation of the statistics [11], reverse scan patterns are used for the scans of SBs within a TB and the coefficients within a SB. Here, the scan of SBs within a TB is the same as that of coefficients within a SB, which includes diagonal, horizontal and vertical scans, as shown in Fig. 1. In order to exploit the sparse property of TB, the syntax element *coded_sub_block_flag* (*CSBF*) is first coded to indicate the significance of a SB. The *CSBF* for a SB is defined to be 1 if at least one transform coefficient level in that SB is non-zero, and 0 otherwise [6]. The *CSBF* for the SB containing the last significant transform coefficient level is not coded, since it is known to be 1. The *CSBF* for the SB containing the DC coefficient is also not coded, since it is equal to 1 with high probability. Then, for a SB with *CSBF* equal to 1, the syntax element *significant_coeff_flag* is coded to indicate the significance of a scanning position. For the significant transform coefficient levels, the syntax elements *coeff_abs_greater1_flag*, *coeff_abs_greater2_flag*, and *coeff_abs_level_remaining* are coded to indicate the absolute magnitude of the transform coefficient levels. Finally, the syntax element *coeff_sign_flag* is coded to indicate the sign flag for significant transform coefficient levels. As described in the previous section, these syntax elements are coded in different scan passes. To improve the throughput, only the first eight *coeff_abs_greater1_flag* in a SB are coded in regular mode. After that, the values are left to be coded in bypass mode with *coeff_abs_level_remaining*. The similar method is also used for *coeff_abs_greater2_flag*. That is only the *coeff_abs_greater2_flag* for the first transform coefficient level with absolute magnitude larger than 1 is coded in regular mode, and the rest of the transform coefficient levels with absolute magnitude larger than 1 are also coded in bypass mode with *coeff_abs_level_remaining*.

2.1.2. Context modeling scheme for transform coefficient levels

In this paper, we will focus on the coding of the syntax elements *significant_coeff_flag*, *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag*, thus the context modeling for these syntax elements are presented in this subsection. For the context modeling of other syntax elements, refer to [5,6] for the details.

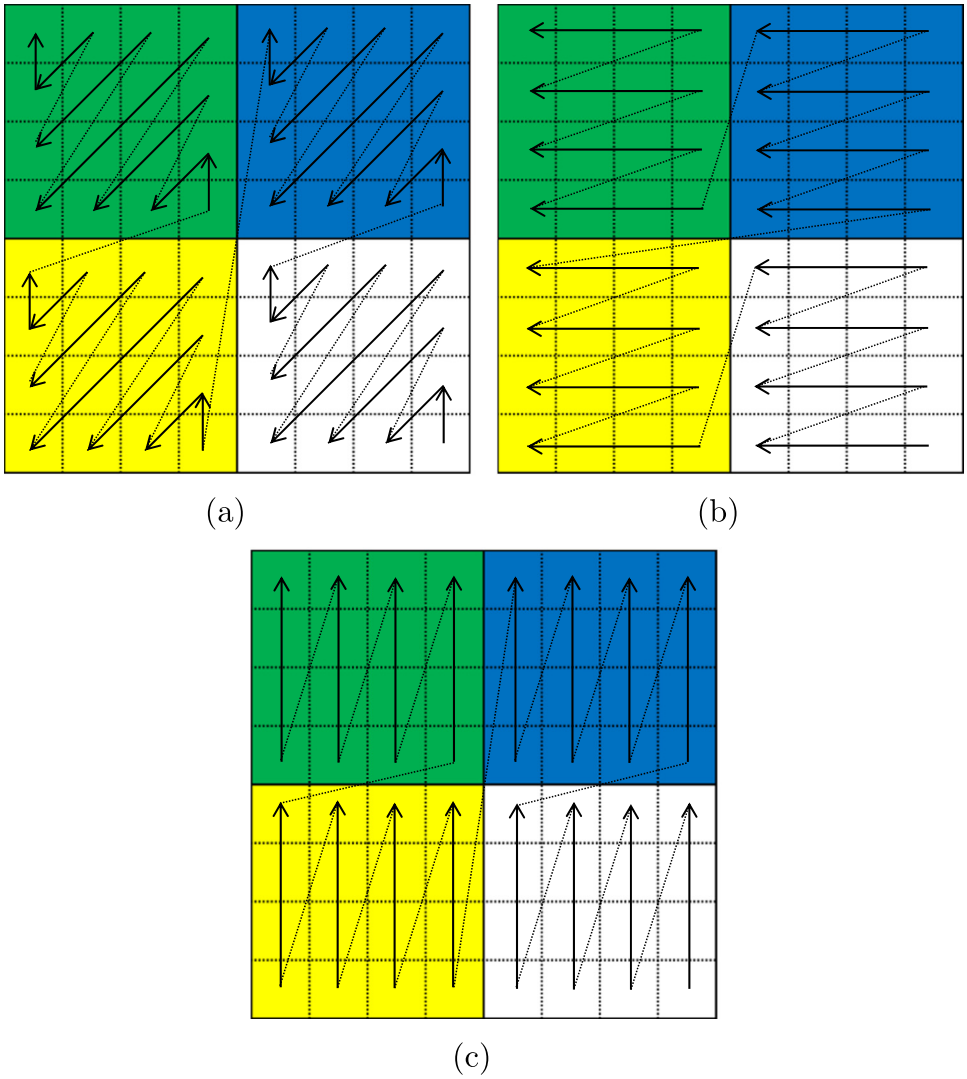

Fig. 1. Sub-block (SB) partition within 8×8 TB: (a) diagonal, (b) horizontal, (c) vertical.

For coding *significant_coeff_flag*, the position-based context modeling shown in Fig. 2 is used for 4×4 TB, in which the context model selection is dependent on the position of the transform coefficient level within TB. The position and template based context modeling is used for TBs larger than 4×4. In this approach, a template is first selected according to *CSBF*s of the neighboring lower and right SBs s_l and s_r; the specific context model is finally determined by the position of the transform coefficient level within that SB using the selected template. There are 4 templates in HEVC corresponding to 4 combinations of s_l and s_r, shown in Fig. 3.

TBs in luma component are split into two regions, in which top-left SB is region 1 and other SBs are region 2. The context model selections of *significant_coeff_flag* for luma and chroma are summarized in Algorithms 1 and 2, respectively. In Algorithms 1 and 2, Ctx_{Sig} denotes the context model index of *significant_coeff_flag*, (x_{SB}, y_{SB}) is the position of the current SB, (x_{InSB}, y_{InSB}) is the position of the current transform coefficient level within the current SB, $CtxInc_{4\times4}[\cdot][\cdot]$ is the context model index assignment in 4×4 TB shown in Fig. 2 and $Temp[\cdot][\cdot]$ is the selected template for SBs in TBs larger than 4×4 shown in Fig. 3.

Algorithm 1. The context model selection of *significant_coeff_flag* for luma.

If the current coefficient is DC, then
$Ctx_{Sig} = 0$

0	1	5	7
2	3	5	7
4	4	8	8
6	6	8	8

Fig. 2. Context model index assignment for *significant_coeff_flag* in 4×4 TB.

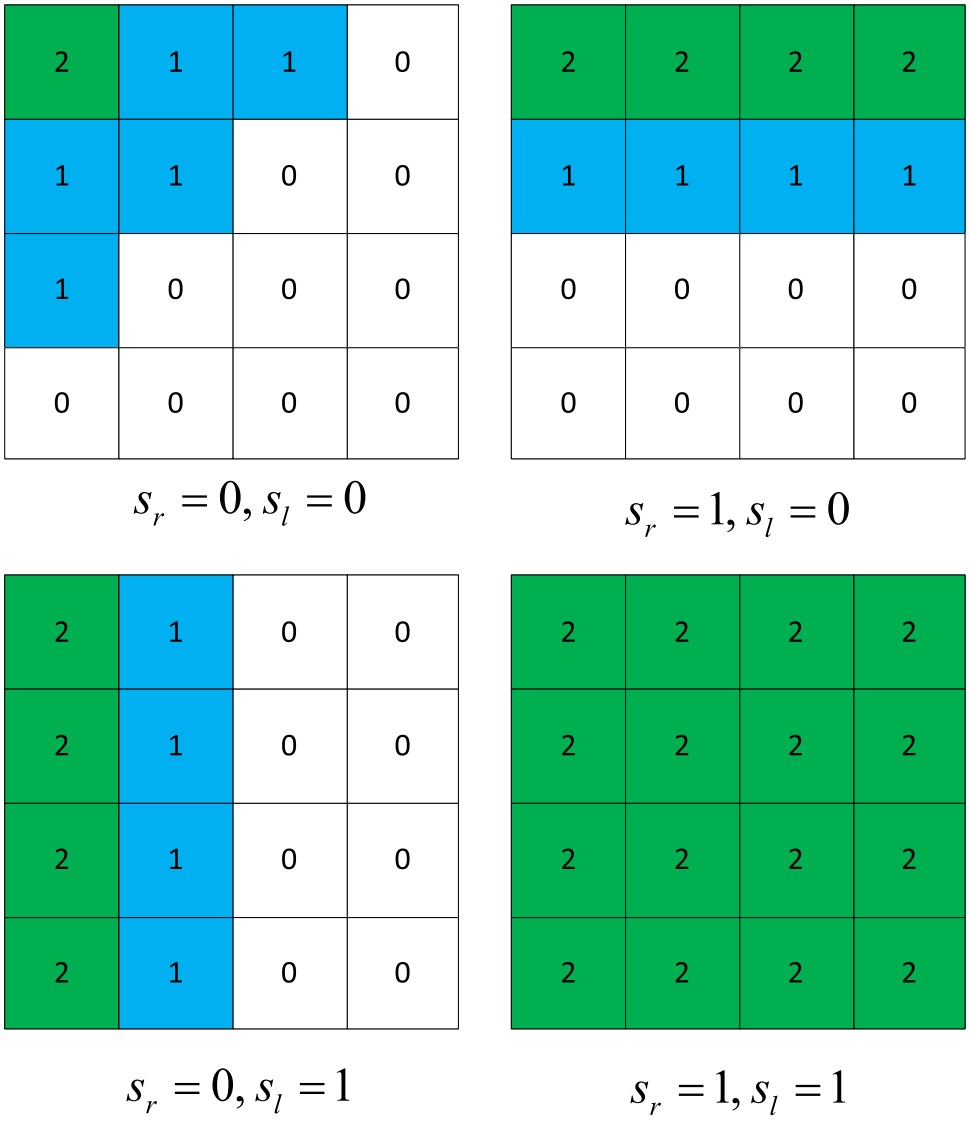

Fig. 3. Templates for *significant_coeff_flag* in TBs larger than 4×4.

Else If TB size is equal to 4×4, then

$Ctx_{Sig} = CtxInc_{4\times4}[x_{InSB}][y_{InSB}]$

Else If TB size is equal to 8×8 and scan pattern is diagonal, then

$$Ctx_{Sig} = \begin{cases} 9 + Temp[x_{InSB}][y_{InSB}], & \text{if } x_{SB} + y_{SB} == 0 \\ 12 + Temp[x_{InSB}][y_{InSB}], & \text{otherwise} \end{cases}$$

ElseIf TB size is equal to 8×8 and scan pattern is horizontal or vertical, then

$$Ctx_{Sig} = \begin{cases} 15 + Temp[x_{InSB}][y_{InSB}], & \text{if } x_{SB} + y_{SB} == 0 \\ 18 + Temp[x_{InSB}][y_{InSB}], & \text{otherwise} \end{cases}$$

Else

$$Ctx_{Sig} = \begin{cases} 21 + Temp[x_{InSB}][y_{InSB}], & \text{if } x_{SB} + y_{SB} == 0 \\ 24 + Temp[x_{InSB}][y_{InSB}], & \text{otherwise} \end{cases}$$

Algorithm 2. The context model selection of *significant_coeff_flag* for chroma.

If the current coefficient is DC, then

$Ctx_{Sig} = 0$

Else If TB size is equal to 4×4, then

$Ctx_{Sig} = CtxInc_{4\times4}[x_{InSB}][y_{InSB}]$

Else If TB size is equal to 8×8, then

$$Ctx_{Sig} = \begin{cases} 9 + Temp[x_{InSB}][y_{InSB}], & \text{if } x_{SB} + y_{SB} == 0 \\ 12 + Temp[x_{InSB}][y_{InSB}], & \text{otherwise} \end{cases}$$

Else

$$Ctx_{Sig} = \begin{cases} 15 + Temp[x_{InSB}][y_{InSB}], & \text{if } x_{SB} + y_{SB} == 0 \\ 18 + Temp[x_{InSB}][y_{InSB}], & \text{otherwise} \end{cases}$$

For coding *coeff_abs_greater1_flag*, a context model set is first

selected depending on whether there is *coeff_abs_greater1_flag* equal to 1 in the preceding SB. Separate context model sets are used for luma and chroma; another context model set is used for the top-left SB in luma. In each context model set, there are 4 context models and the specific context model is determined by the number of the previously coded transform coefficient levels with absolute magnitude equal to 1 *NumEqu*1 and larger than 1 *NumGre*1 in the current SB. For coding *coeff_abs_greater2_flag*, the selection of context model set is the same as that for *coeff_abs_greater1_flag*, but there is only one context model in each context model set. The context model selection for *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* are summarized in Algorithms 3 and 4, respectively. In Algorithms 3 and 4, $c1$ is defined to 1 if there is *coeff_abs_greater1_flag* equal to 1 in the preceding SB; $Ctx_{greater1}$ and $Ctx_{greater2}$ denote the context model index for *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag*, respectively.

Algorithm 3. The context model selection of *coeff_abs_greater1_flag*.

Luma Component:

$$CtxSet = \begin{cases} 0 & \text{If } x_{SB} + y_{SB} == 0 \text{ and } c1 == 0 \\ 1 & \text{If } x_{SB} + y_{SB} == 0 \text{ and } c1 == 1 \\ 2 & \text{If } x_{SB} + y_{SB} \neq 0 \text{ and } c1 == 0 \\ 3 & \text{If } x_{SB} + y_{SB} \neq 0 \text{ and } c1 == 1 \end{cases}$$

$$Ctx_{greater1} = \begin{cases} \min(NumEqu1, 2) + 4_{*}CtxSet, & \text{if } NumGre1 == 0 \\ 3 + 4_{*}CtxSet, & \text{otherwise} \end{cases}$$

Chroma Component:

$$CtxSet = \begin{cases} 0, & \text{If } c1 == 0 \\ 1, & \text{If } c1 == 1 \end{cases}$$

$$Ctx_{greater1} = \begin{cases} \min(NumEqu1, 2) + 4_{*}CtxSet, & \text{if } NumGre1 == 0 \\ 3 + 4_{*}CtxSet, & \text{otherwise} \end{cases}$$

Algorithm 4. The context model selection of *coeff_abs_greater2_flag*.

Luma Component:

$$Ctx_{greater2} = \begin{cases} 0 & \text{If } x_{SB} + y_{SB} == 0 \text{ and } c1 == 0 \\ 1 & \text{If } x_{SB} + y_{SB} == 0 \text{ and } c1 == 1 \\ 2 & \text{If } x_{SB} + y_{SB} \neq 0 \text{ and } c1 == 0 \\ 3 & \text{If } x_{SB} + y_{SB} \neq 0 \text{ and } c1 == 1 \end{cases}$$

Chroma Component:

$$Ctx_{greater2} = \begin{cases} 0, & \text{If } c1 == 0 \\ 1, & \text{If } c1 == 1 \end{cases}$$

2.2. Binary arithmetic coding engine in HEVC

The binary arithmetic coding engine in HEVC is M-coder [22], in which the implementation of the probability update is based on the state machine approach. In M-coder, the probability of a symbol to be coded is represented by (p_{LPS}, V_{MPS}), where p_{LPS} is the probability of the least probable symbol (LPS) and V_{MPS} is the value of the most probable symbol (MPS). The range of p_{LPS} is projected into a set of representative probabilities $S_p = \{p_0, p_1, \ldots, p_{63}\}$ according to Eq. (1):

$$p_s = \alpha \cdot p_{s-1}, \quad \text{for all } s = 1, 2, \ldots, 63 \tag{1}$$

where s denotes the state defined the estimation of p_{LPS}, $\alpha = \left(\frac{0.01875}{0.5}\right)^{1/63}$ and $p_0 = 0.5$. The probability p_{63} is always used to code the terminal bit, thus the probability update in M-coder shown in Eq. (2) can be implemented with look-up tables $TransStateLPS[s]$ and $TransStateMPS[s]$, which contain the state corresponding to the next probability after compressing the current symbol with state s:

$$p_{new} = \begin{cases} \max(\alpha \cdot p_{old}, p_{62}), & \text{if MPS occurs} \\ \alpha \cdot p_{old} + (1 - \alpha), & \text{if LPS occurs} \end{cases} \tag{2}$$

where p_{old} is the given LPS probability and p_{new} is the updated counterpart.

Suppose the coding interval is represented with (L, R), where L is the lower bound of the coding interval and R is the length of the coding interval. The coding interval is subdivided into two parts after obtaining p_{LPS}, shown as follows:

$$R_{LPS} = p_{LPS} \cdot R, \; R_{MPS} = R - R_{LPS} \tag{3}$$

where R_{LPS} and R_{MPS} are the coding interval lengths for LPS and MPS, respectively. The coding interval length R satisfies the following inequality after adopting the renormalization procedure in [29]:

$$\frac{1}{2} \cdot 2^{b-1} \leq R < 2^{b-1} \tag{4}$$

where b is the precision of the register storing R. To remove the multiplication in the coding interval subdivision, the interval $\left[\frac{1}{2} \cdot 2^{b-1}, 2^{b-1}\right)$ is uniformly quantized into four cells, so the multiplication can be implemented with the look-up table $TabLPSRange[s][\Delta]$, where s is the state corresponding to p_{LPS} and Δ, $\Delta \in \{0, 1, 2, 3\}$, is the interval cell index. Therefore, the implementation of M-coder is summarized in Algorithm 5.

Algorithm 5. The procedure for encoding the binary symbol x_t with M-coder.

$\Delta = (R \gg 6)\&3$
$R_{LPS} = TabLPSRange[s][\Delta]$
$R = R - R_{LPS}$
If x_t is LPS, then
  $L = L + R$
  $R = R_{LPS}$
  $s = TransStateLPS[s]$
Else
  $s = TransStateMPS[s]$
End If
Call Renormalization procedure.

3. The proposed entropy coding scheme for HEVC

In this section, we will present the improved context modeling for transform coefficient levels and the BAC engine with low memory requirement. In the description of the improved context modeling for transform coefficient levels, we first illustrate the statistical features of the transform coefficient levels that guide the context modeling design in this work, and then present the proposed context modeling in detail. For the BAC engine with low memory requirement, we first describe the multi-parameter probability estimation, and then present the multiplication with low bit capacity used in the coding interval subdivision.

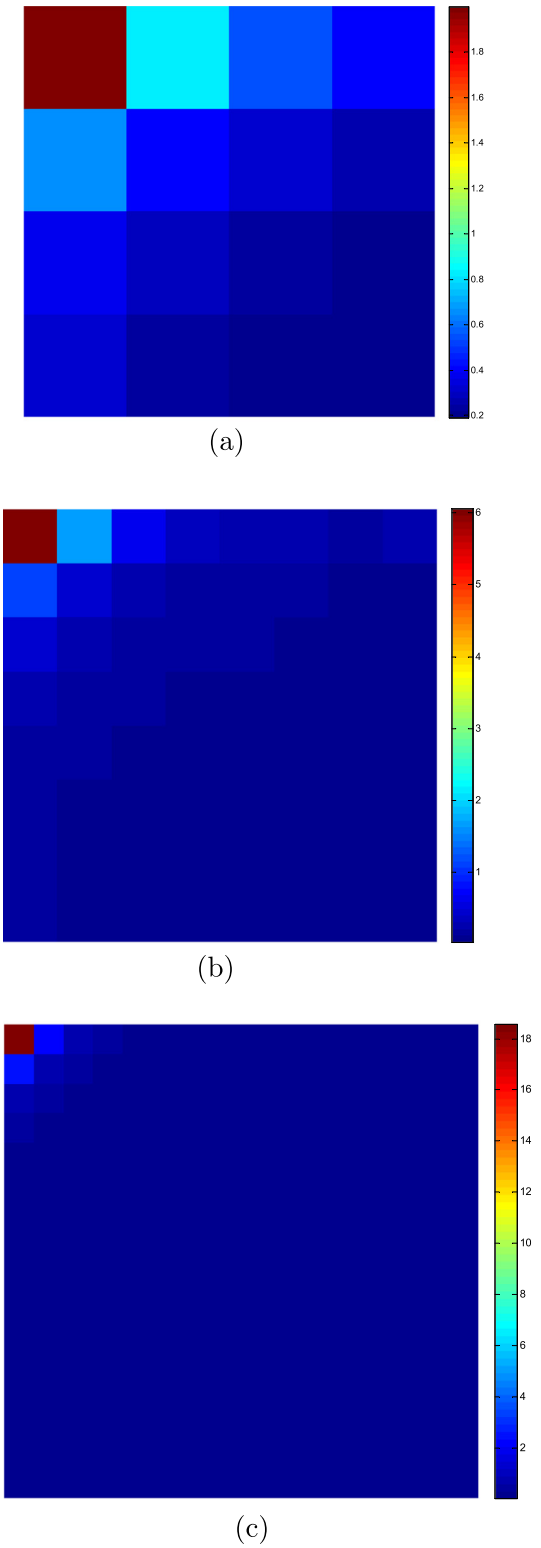

Fig. 4. The standard variance of transform coefficient level at each position within (a) 4×4, (b) 8×8 and (c) 16×16 TBs.

3.1. Improved context modeling for transform coefficient levels

3.1.1. Statistical features of transform coefficient levels

As we know, the statistical distributions of the transform coefficient levels at low frequency subbands have larger variance than those in the high frequency subbands, which is similar to the case in image coding as illustrated in [30]. Fig. 4 shows the standard variance of the transform coefficient levels at different positions in 4×4, 8×8 and 16×16 TBs, where the sequences in Class C are coded under $QP=22$. From Fig. 4, we can see that the variance of the transform coefficient levels in a TB decreases from the top left position to bottom right position and the transform coefficient levels at different positions have the similar variance in the high frequency region.

In addition, there are some correlations between the current transform coefficient level and its neighbors, which consist of the previously coded transform coefficient levels in the reverse scanning path. For example, the current transform coefficient level is more likely to be significant when there is a significant one among its neighbors. Here the mutual information $I(X;Y)$ is used to quantitatively evaluate the correlations between two random variables X and Y, which is calculated as follows:

$$I(X;Y)=\sum_{y\in Y}\sum_{x\in X}p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{5}$$

where $p(x,y)$ is joint probability distribution function of X and Y, $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y, respectively. Table 1 illustrates the empirical estimate of mutual information between *significant_coeff_flag* and *numSigs* for 8×8 TB, where *numSigs* is the number of the significant neighbors covered by a local template. Three local templates are evaluated for the current transform coefficient level x in Table 1, which are $\{x_0, x_1\}$, $\{x_0, \ldots, x_4\}$ and $\{x_0, \ldots, x_8\}$, as illustrated in Fig. 5. In Table 1, *temp*1, *temp*2 and *temp*3 denote $\{x_0, x_1\}$, $\{x_0, \ldots, x_4\}$ and $\{x_0, \ldots, x_8\}$, respectively; the position of the current transform coefficient level is represented as the (X, Y) coordinate relative to DC coefficient. Table 1 is based on the statistical data of 8×8 TBs, where the sequences in Class C are coded under $QP=22$.

As we know, if the mutual information $I(X;Y)$ between X and Y is larger, the correlation between X and Y is higher, thus it is more efficient to guide the context modeling of X using Y. Comparing *temp*1 column with *temp*2 column in Table 1, we can see that the empirical estimate of mutual information under *temp*2 is always larger. This demonstrates that the correlation will not be fully exploited if only a few neighboring transform coefficient levels are used. Comparing *temp*2 column with *temp*3 column in Table 1, it can also be seen that the empirical estimate of mutual information under *temp*2 is also larger in some cases. This indicates that the context dilution problem may be caused if too many neighboring transform coefficient levels are used [11]. So, the local template *temp*2 $\{x_0, \ldots, x_4\}$ is used in this paper to exploit the correlations between the current transform coefficient level and its neighbors, which is the same as that in [14].

3.1.2. Description of the proposed context modeling

To achieve higher coding efficiency, the statistical features of transform coefficient levels described in the previous subsection should be used to guide the context modeling design.

For *significant_coeff_flag*, the number of the significant transform coefficient levels in the local template *numSigs* and the

Table 1
Empirical estimate of mutual information between *significant_coeff_flag* and *numSigs* for different local templates.

Position	Empirical estimate of mutual information		
	temp1	*temp2*	*temp3*
(0, 0)	0.190	0.244	0.323
(0, 1)	0.113	0.143	0.145
(1, 0)	0.125	0.159	0.158
(3, 0)	0.092	0.127	0.135
(0, 3)	0.115	0.159	0.169
(5, 0)	0.102	0.141	0.118
(0, 5)	0.087	0.125	0.111

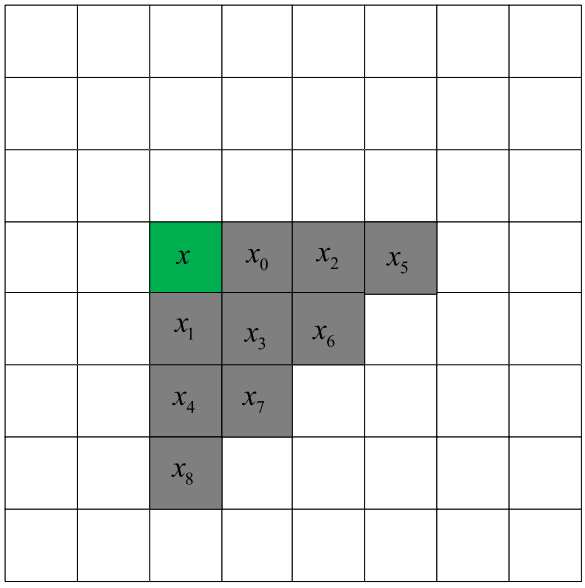

Fig. 5. Illustration of the different local templates for the current transform coefficient level x evaluated in Table 1.

position of the current transform coefficient levels within TB (x, y) are used in the context model selection. If *numSigs* and (x, y) are directly combined, the number of the context models is very large. Take 8×8 TB as an example, the number of the context models is up to 384 if directly combining *numSigs* and (x, y), since $0 \le numSigs \le 5$ and $0 \le x \le 7$, $0 \le y \le 7$. So many context models will consume more memory and also cause the context dilution problem. To address this problem, TBs are first split into different regions based on the position of the transform coefficient level. Then, the specific context model for *significant_coeff_flag* of the transform coefficient level is determined for each region by *numSigs* calculated with its neighbors.

More specifically, for TBs in luma component, the top-left SB is split into 3 regions based on Eq. (6), and other SBs are split into 2 regions based on Eq. (7); for TBs in chroma component, they are split into 2 regions based on Eq. (8):

$$RegIdx = \begin{cases} 0, & \text{if } x_{InSB} + y_{InSB} < 2 \\ 1, & \text{if } x_{InSB} + y_{InSB} \ge 2 \\ 2, & \text{if } x_{InSB} + y_{InSB} \ge 5 \end{cases} \quad \text{and} \quad x_{InSB} + y_{InSB} < 5 \tag{6}$$

$$RegIdx = \begin{cases} 3, & \text{if } x_{InSB} + y_{InSB} < 4 \\ 4, & \text{otherwise} \end{cases} \tag{7}$$

$$RegIdx = \begin{cases} 0, & \text{if } x + y < 2 \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

In the above equations, *RegIdx* denotes the region index and (x_{InSB}, y_{InSB}) is the position of the current transform coefficient level within the current SB, which can be calculated based on Eq. (9):

$$\begin{cases} x_{InSB} = (x - ((x \gg ; 2) \ll ; 2)) \\ y_{InSB} = (y - ((y \gg ; 2) \ll 2)) \end{cases} \tag{9}$$

Fig. 6 illustrates the splitting method for TBs in luma and chroma components, where the block circulated by red square is the top-left SB. Based on this TB partition method, the context model index for *significant_coeff_flag* is finally calculated by Eq. (10):

$$Ctx_{Sig} = numSigs + 6 \cdot RegIdx \tag{10}$$

where Ctx_{Sig} denotes the context model index of *significant_coeff_flag*. The context model selection procedure for *significant_coeff_flag* is presented in Algorithm 6.

Algorithm 6. Context model selection of *significant_coeff_flag* in the proposed method.

Luma Component:
If the current coefficient is in the top-left SB, then
  Calculate *RegIdx* based on Eq. (6).
Else

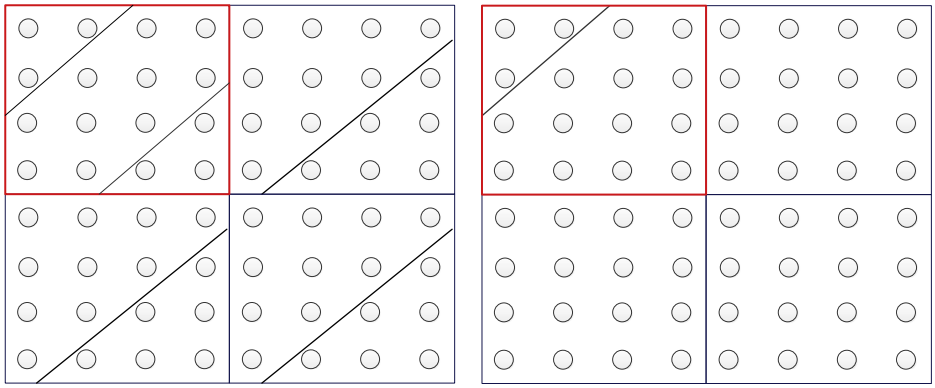

Fig. 6. Splitting method of TBs in (a) luma and (b) chroma components for coding *significant_coeff_flag*.

Fig. 7. Splitting method of TBs in luma for coding *coeff_abs_greater1_flag*.

Calculate *RegIdx* based on Eq. (7).
End If
Calculate the context model index for *significant_coeff_flag* with Eq. (10).
Chroma Component:
Calculate *RegIdx* based on Eq. (8).
Calculate the context model index for *significant_coeff_flag* with Eq. (10).

For *coeff_abs_greater1_flag*, the number of neighbors covered by the local template with absolute magnitude equal to 1 *NumEqu*1 and larger than 1 *NumGre*1 are used as the contexts. Moreover, the position (x, y) of the transform coefficient level within TB is also used as a context for luma to exploit the correlation between the transform coefficient level and its position. To use the position (x, y) in the context model selection, the TBs in luma component are split into 3 regions based on Eq. (11). Fig. 7 illustrates the TB splitting method for *coeff_abs_greater1_flag*:

$$RegIdx = \begin{cases} 0, & \text{If } x + y < 3 \\ 1, & \text{If } x + y \geq 3 \quad \text{and} \quad x + y < 10 \\ 2, & \text{If } x + y \geq 10 \end{cases} \tag{11}$$

After getting *RegIdx*, the context model index for *coeff_abs_greater1_flag* is calculated with Eq. (12):

$$Ctx_{greater1} = 7 \cdot RegIdx + \begin{cases} \min(NumGre1 - 1, 3), & \text{if } NumGre1 > 0 \\ \min(NumEqu1, 2) + 4, & \text{otherwise} \end{cases} \tag{12}$$

where $Ctx_{greater1}$ denotes the context model index of *coeff_abs_greater1_flag*.

For *coeff_abs_greater2_flag*, the number of neighbors covered by the local template with absolute magnitude larger than 1 *NumGre*1 and larger than 2 *NumGre*2 are used as the contexts. With these two contexts, the context model index is calculated as follows:

$$Ctx_{greater2} = \begin{cases} 0, & \text{if } NumGre2 > 0 \\ 1, & \text{else if } NumGre1 > 0 \\ 2, & \text{otherwise} \end{cases} \tag{13}$$

where $Ctx_{greater2}$ denotes the context model index of *coeff_abs_greater2_flag*.

For the *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* of the transform coefficient level at the last significance scan position, a separate context model is used, and this separate context model is never selected for the transform coefficient levels at other scan positions. Thus the context model selection procedures for *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* are presented in Algorithms 7 and 8, respectively.

Algorithm 7. Context model selection of *coeff_abs_greater1_flag* in the proposed method.

Luma Component:
If the current coefficient is at the last significant scan position, then
Set the context model index as 21.
Else
Calculate *RegIdx* based on Eq. (11).
Calculate the context model index with Eq. (12).
End If
Chroma Component:
If the current coefficient is at the last significant scan position, then
Set the context model index as 7.
Else
Set *RegIdx* as 0.
Calculate the context model index with Eq. (12).
End If

Algorithm 8. Context model selection of *coeff_abs_greater2_flag* in the proposed method.

Luma Component:
If the current coefficient is at the last significant scan position, then
Set the context model index as 3.
Else
Calculate the context model index with Eq. (13).
End If
Chroma Component:
If the current coefficient is at the last significant scan position, then
Set the context model index as 3.
Else
Calculate the context model index with Eq. (13).
End If

3.2. The binary arithmetic coding engine with low memory requirement

A multi-parameter probability update was proposed in [24] for updating probability with different adaption speeds in BAC engine. In this method, the probability updating models with two different adaption speeds are applied, which are shown as follows:

$$p_{new} = (p^0_{new} + p^1_{new} + 1) \gg 1; \quad \text{with} \begin{cases} p^0_{new} = \bar{\alpha}_0 \cdot y + (1 - \bar{\alpha}_0) \cdot p^0_{old} \\ p^1_{new} = \bar{\alpha}_1 \cdot y + (1 - \bar{\alpha}_1) \cdot p^1_{old} \end{cases} \tag{14}$$

where y is equal to zero if the current symbol is 0, otherwise y is equal to 1; $\bar{\alpha}_0$ and $\bar{\alpha}_1$ are two adaption speeds, which are equal to $\frac{1}{16}$ and $\frac{1}{128}$, respectively. At the beginning, the model with $\bar{\alpha}_0$ is used to update the probability because it can converge to optimal value very fast. After stabilization near optimal value, the average value of the two probabilities obtained by $\bar{\alpha}_0$ and $\bar{\alpha}_1$ is used. So a counter of updates since last initialization is introduced to determine when to switch the probability update model.

The coding interval subdivision in Eq. (3) is achieved by the look-up table technique in [24], whose size is equal to 288 Kb. Such large look-up table will consume much area/memory in the

hardware implementation. If we multiply coding interval width R by probability p in the straightforward way in the coding interval subdivision, the multiplication with bit capacity 15×9 is required, since p is represented with 15 bits and R is represented with 9 bits in [24]. The bit capacity of the multiplication is also too high for hardware implementation.

To reduce the bit capacity of the multiplication, a multiplication with low bit capacity is proposed in this paper. In this method, the interval $[\frac{1}{2}\cdot 2^{b-1}, 2^{b-1})$ in Eq. (4) is uniformly quantized into 32 cells, thus the coding interval subdivision can be approximated as follows:

$$R = p\cdot R \approx \frac{(p \ll;(b-1)) + p\cdot\frac{1}{32}\cdot\Delta\cdot 2^{b-1}}{2} = \frac{(p \ll;(b-1)) + 2^{b-6}\cdot p\cdot\Delta}{2} \approx \frac{(p \ll;(b-1)) + 2^{b}\cdot(p \gg;6)\cdot\Delta}{2} \quad (15)$$

where p is represented by k bits and Δ is the interval cell index that is calculated by $\Delta = (R - 2^{b-2}) \gg; 3$. Same as that in [24], k and b in our implementation are equal to 15 and 10, respectively. To guarantee the accuracy of the approximation, the LPS/MPS concept is used, that is the probability p is represented by (p_{LPS}, V_{MPS}). With this approximation, the bit capacity of multiplication is reduced to 9×5, and the implementation of the proposed method is summarized in Algorithm 9.

Algorithm 9. The procedure for encoding the binary symbol x_t with the proposed BAC engine.

$\Delta = (R - 2^{b-2}) \gg; 3;$
$p_{LPS} = (p_0 + p_1 + 1) \gg; 1;$
$p_{LPS} = \min(((1 \ll; k) - p_{LPS}), p_{LPS});$
$R_{LPS} = ((p_{LPS} \ll;(b-1)) + ((\Delta\cdot(p_{LPS} \gg; 6)) \ll; b) + (1 \ll;(k-1))) \gg; k;$
$R_{LPS} = (R_{LPS} + 1) \gg; 1;$
$R = R - R_{LPS};$
If x_t is LPS, then
  $L = L + R;$
  $R = R_{LPS};$
  Call Renormalization procedure.
Else
  Call Renormalization procedure.
End If
If x_t is 1, then
  $p_0 = p_0 + (1 \ll;(k-4)) - (p_0 \gg; 4);$
  $p_1 = p_1 + (1 \ll;(k-7)) - (p_1 \gg; 7);$
Else
  $p_0 = p_0 - (p_0 \gg; 4);$
  $p_1 = p_1 - (p_1 \gg; 4);$
End If

4. Experimental results

This section provides the coding performance of the proposed modification to CABAC in HEVC. Individual results for the improved context modeling and the BAC engine with low memory requirement are first reported. Then, the proposed two techniques and the original entropy coding scheme in HEVC are compared as a whole.

The experiments are conducted on HM14.0 using main profile under three configurations, which include all intra (AI), low delay (LD) and random access (RA). There are twenty common test sequences, which are split into five classes depending on their resolution: Class A (2K), Class B (1080p), Class C (WVGA), Class D (WQVGA) and Class E (720p). There is an additional class consisting of the screen content sequences, which is named Class F. According to the common test conditions in [31], the results for LD configuration do not include the sequences in Class A, and the results for RA configuration do not include the sequences in Class E. In addition, we also evaluate the proposed two techniques on five UHD sequences (their resolution is 3840×2160) provided in [32] under each configuration. These UHD sequences include *Fountains*, *Runners*, *Rushour*, *Trafficflow* and *Campfireparty*. The quantization parameters are set to $QP = \{22, 27, 32, 37\}$. Coding efficiency is evaluated by BD-Rate that can be computed according to [33].

4.1. Improved context modeling scheme for transform coefficient levels

In this section, we evaluate the coding performance of the improved context modeling in this paper by comparing it with the original context modeling in HEVC and the context modeling in [19]. The overall coding performance of the improved context modeling is first provided as well as the number of the required context models. Then the coding performance of the proposed context modeling for significant flag (*significant_coeff_flag*) and level information (*coeff_abs_greater1_flag* and *coeff_abs_greater2_flag*) are provided separately. The row labeled **Overall** (**HD**) in following tables is the average BD-Rate gains calculated on the high resolution sequences in Classes A, B and UHD.

4.1.1. Overall coding performance of the improved context modeling

Table 2 shows the BD-Rate of the improved context modeling over the original context modeling in HEVC, which includes the context modeling for *significant_coeff_flag*, *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag*. Table 2 shows that, on average, the improved context modeling reduces the bit-rate by 0.8% for AI, 0.6% for RA and 0.4% for LD compared with the original context modeling in HEVC.

Table 3 shows the BD-Rate of the improved context modeling for the transform coefficient levels over that in [19]. From Table 3,

Table 2
BD-Rate (%) of the improved context modeling over that in HEVC.

	AI			*RA*			*LD*		
Sequences	*Y*	*U*	*V*	*Y*	*U*	*V*	*Y*	*U*	*V*
Class A	−0.9	−0.6	−0.7	−0.5	−0.3	−0.2			
Class B	−0.8	−0.6	−0.7	−0.5	−0.4	−0.6	−0.3	−0.1	−0.1
Class C	−0.6	−0.2	−0.3	−0.6	−0.4	−0.3	−0.3	0.4	0.2
Class D	−0.6	−0.4	0.0	−0.6	0.5	−0.4	−0.4	−1.2	−0.1
Class E	−1.0	−0.6	−0.5				−0.6	−1.0	−0.2
Overall	**−0.8**	**−0.5**	**−0.4**	**−0.6**	**−0.2**	**−0.4**	**−0.4**	**−0.4**	**0.0**
Class F	−1.5	−1.5	−1.5	−1.3	−1.3	−1.2	−1.3	−1.5	−1.8
UHD	−0.7	−0.8	−0.9	−0.5	−0.2	0.1	−0.3	−0.2	−0.1
Overall(HD)	**−0.8**	**−0.7**	**−0.8**	**−0.5**	**−0.3**	**−0.2**	**−0.3**	**−0.1**	**−0.1**
Enc time (%)	107			100			100		
Dec time (%)	101			100			100		

we can see that the average BD-Rate gains are 0.2%, 0.2% and 0.1% for AI, RA and LD, respectively. Moreover, more BD-Rate gains can be achieved for high resolution sequences and the average BD-Rate gains for high resolution sequences are 0.3%, 0.4% and 0.2% for AI, RA and LD, respectively. We will explain the reasons why more BD-Rate gains are achieved for high resolution sequences in the following two subsections.

Table 4 gives an overview of the number of the context models in the context modeling in HEVC, [19] and this paper. It can be seen that the number of the context models used for *significant_coeff_flag* in this paper is the same as that in HEVC and much lower than that in [19]. The number of the overall context models in this paper is higher than that in HEVC, but still lower than that in [19].

4.1.2. Coding performance of the context modeling for significant flag

Table 5 presents the BD-Rate of the proposed context modeling for *significant_coeff_flag* over that in HEVC. It can be seen that, on average, the proposed context modeling for *significant_coeff_flag* reduces the bit-rate by 0.4% for AI, 0.4% for RA and 0.3% for LD compared with the context modeling in HEVC. Table 6 lists the coding performance comparisons between the proposed context modeling for *significant_coeff_flag* and that in [19]. From Table 6, we can see that the proposed context modeling can achieve similar or better coding efficiency with fewer context models compared with that in [19]. In the following, we will make an analysis to the context modeling for *significant_coeff_flag* in HEVC, [19] and this paper.

The context modeling in HEVC, [19] and this paper all use the coefficient position as a context. However, the probability distribution of *significant_coeff_flag* at the same position in different TBs may be different due to the diversity of the prediction residual, such as the statistics of the prediction residuals in the homogeneous and inhomogeneous regions are very different. For a coefficient position in a given TB, it is required to use other information as additional contexts to distinguish the probability distributions at this position with different statistical properties. Toward this, HEVC uses neighboring lower and right SBs as the additional contexts, while the literature [19] and this paper use the neighbors covered by the local template of the current transform coefficient level as the additional contexts. Compared with the neighboring lower and right SBs, the neighbors covered by the local template are more close to the current transform coefficient level. In other words, the transform coefficient levels far away from the current transform coefficient level will not be used in the context modeling in [19] and this paper. This is illustrated in Fig. 8, where x is the current transform coefficient level, $\{x_0, \ldots, x_4\}$ are the neighbors covered by the local template of x, and *RightSB* and *LowerSB* are the neighboring right and lower SBs. Thus the neighbors covered by the local template have stronger ability to distinguish the probability distributions with different statistical

Table 3
BD-Rate (%) of the improved context modeling over that in [19].

	AI			RA			LD		
Sequences	Y	U	V	Y	U	V	Y	U	V
Class A	−0.2	−0.1	−0.1	−0.1	0.0	−0.1			
Class B	−0.4	−0.2	−0.2	−0.3	0.1	0.1	−0.2	0.3	0.4
Class C	−0.1	0.1	0.0	−0.1	−0.3	0.1	−0.1	0.2	−0.3
Class D	−0.1	0.1	0.2	−0.1	0.7	0.2	0.0	−1.0	−0.3
Class E	−0.2	0.2	0.0				0.0	−0.6	0.4
Overall	**−0.2**	**0.0**	**0.0**	**−0.2**	**0.1**	**0.1**	**−0.1**	**−0.2**	**−0.2**
Class F	0.1	0.1	0.0	0.1	0.1	0.0	−0.2	−0.5	0.0
UHD	−0.3	−0.1	−0.2	−0.6	0.0	0.1	−0.2	0.0	−0.1
Overall(HD)	**−0.3**	**−0.1**	**−0.2**	**−0.4**	**0.0**	**0.0**	**−0.2**	**0.0**	**−0.2**
Enc time (%)	101			100			100		
Dec time (%)	100			100			100		

Table 4
The number of context models in the context modeling in HEVC, [19] and this paper.

	HEVC		[19]		This paper	
Syntax elements	*Luma*	*Chroma*	*Luma*	*Chroma*	*Luma*	*Chroma*
significant_coeff_flag	27	15	54	12	30	12
coeff_abs_greater1_flag	16	8	16	6	22	8
coeff_abs_greater2_flag	4	2			4	4
Overall	72		88		80	

Table 6
BD-Rate (%) of the proposed context modeling for significant flag over that in [19].

	AI			RA			LD		
Sequences	Y	U	V	Y	U	V	Y	U	V
Class A	0.0	0.0	0.0	0.0	0.3	0.2			
Class B	−0.1	−0.1	0.0	−0.1	−0.1	0.1	−0.1	0.0	0.2
Class C	0.0	−0.1	0.0	0.0	0.1	0.0	−0.1	0.0	−0.3
Class D	0.0	−0.2	0.2	−0.2	0.6	0.1	−0.2	−0.2	−0.2
Class E	0.0	−0.1	−0.2				0.1	0.4	0.3
Overall	**0.0**	**−0.1**	**0.0**	**−0.1**	**0.3**	**0.1**	**−0.1**	**0.1**	**0.0**
Class F	0.1	0.1	0.0	−0.1	−0.1	0.0	0.3	0.3	0.2
UHD	−0.1	−0.1	0.0	0.1	−0.3	0.0	−0.1	0.0	−0.3
Overall(HD)	**−0.1**	**−0.1**	**0.0**	**0.0**	**0.1**	**0.0**	**−0.1**	**0.0**	**0.0**
Enc time (%)	100			100			101		
Dec time (%)	100			100			100		

Table 5
BD-Rate (%) of the proposed context modeling for significant flag over that in HEVC.

	AI			RA			LD		
Sequences	Y	U	V	Y	U	V	Y	U	V
Class A	−0.5	−0.1	−0.3	−0.4	−0.2	−0.3			
Class B	−0.5	−0.2	−0.4	−0.3	−0.3	−0.7	−0.2	−0.6	−0.2
Class C	−0.3	0.0	0.1	−0.4	−0.1	−0.3	−0.3	0.0	−0.1
Class D	−0.4	−0.2	0.1	−0.5	0.4	−0.4	−0.4	−0.4	−0.3
Class E	−0.5	−0.4	−0.3				−0.5	0.2	−0.3
Overall	**−0.4**	**−0.2**	**−0.5**	**−0.4**	**−0.1**	**−0.5**	**−0.3**	**−0.3**	**−0.2**
Class F	−1.5	−1.3	−1.6	−1.3	−1.5	−1.1	−1.3	−1.7	−1.7
UHD	−0.2	−0.4	−0.5	−0.2	0.2	−0.5	−0.2	−0.3	−0.5
Overall(HD)	**−0.4**	**−0.3**	**−0.4**	**−0.3**	**−0.2**	**−0.5**	**−0.2**	**−0.5**	**−0.3**
Enc time (%)	102			102			101		
Dec time (%)	101			100			100		

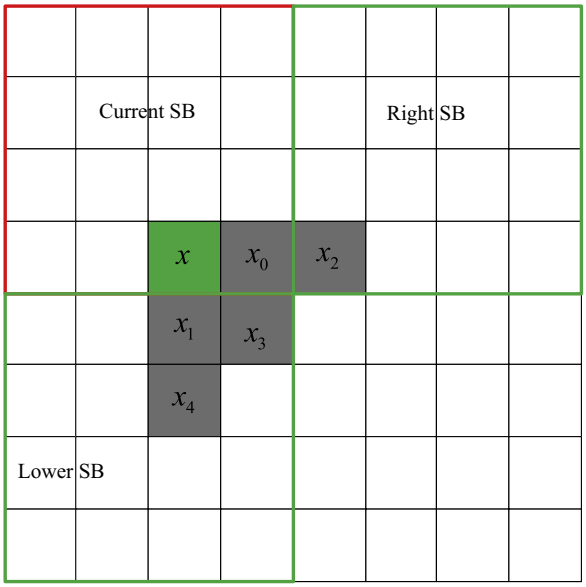

Fig. 8. Illustration of the neighbors used in this paper and HEVC.

properties. So the context modeling in [19] and this paper can achieve better coding performance over that in HEVC.

To capture the characteristics in TBs with different size, TB size is also used as an additional context in [19], which means separate context model set are used for the same position from TBs with different size. On one hand, this can improve the accuracy of context modeling in [19], since it classifies the probability distributions of *significant_coeff_flag* in a more refined way. On the other hand, this will increase the number of the context models and may cause the context dilution problem in some cases, since it requires more samples to accurately estimate context model parameters. Different from [19], the proposed context modeling uses a different TB partitions and the same region from different TBs shares the same context model set to limit the total number of the context models. So the proposed context modeling can achieve better coding performance than that in [19] in some cases by avoiding the context dilution problem.

To verify the above analysis, we estimate the bits $\hat{R}$ used for coding random variable X as follows:

$$\hat{R} = \sum_{y} p(y) \cdot H(X|y) \tag{16}$$

where y is the context model index, $p(y)$ is the occurrence probability of y, $H(X|y)$ is the conditional entropy of X under context model index y, which is calculated as follows:

$$H(X|y) = -\sum_{x} p(X = x|y) \cdot \log(p(X = x|y)) \tag{17}$$

where $p(X = x|y)$ is the conditional probability and $\log(\cdot)$ is the logarithm based on 2.

Table 7 provides the estimated bits for coding *significant_coeff_flag* at different positions in 8×8 TBs on some sequences, where the position is represented as (X, Y) coordinate relative to DC coefficient and these sequences are all coded at $QP = 22$ under RA configuration. From Table 7, we can see that the estimated bits for the context modeling in [19] and this paper are smaller than that in HEVC for all positions. Compared with [19], the estimated bits in this paper are smaller at position $(1, 1)$ and larger at position $(5, 5)$. For position $(1, 1)$, the transform coefficient levels at this position from TBs with different size share context models, thus there are more samples to be used for training the context model parameters, so the estimated bits in this paper are smaller. For position $(5, 5)$, the TB size will increase the accuracy of the context modeling in [19], since the statistics of the transform coefficient levels at this position in TBs with different size are different. Therefore, the estimated bits in [19] are smaller.

Table 7
The estimated bits for coding *significant_coeff_flag* at different positions in 8×8 TBs on some sequences.

Sequences	Position	HEVC	[19]	This paper
*BQMall*_832 × 480	(1, 1)	0.940	0.768	0.763
	(5, 5)	0.743	0.679	0.711
*BasketballDrill*_832 × 480	(1, 1)	0.940	0.909	0.907
	(5, 5)	0.735	0.689	0.702
*ChinaSpeed*_1024 × 768	(1, 1)	0.887	0.749	0.745
	(5, 5)	0.749	0.694	0.709

4.1.3. Coding performance of the context modeling for level information

Table 8 presents BD-Rate of the proposed context modeling for *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* over that in HEVC. As shown in Table 8, the proposed context modeling can achieve average bitrate reduction of 0.4% in AI, 0.2% in RA and 0.1% for LD compared with that in HEVC.

Table 9 shows BD-Rate of the proposed context modeling for *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* over that in [19]. On average, the BD-Rate gains of the proposed context modeling is 0.2%, 0.1% and 0.0% for AI, RA and LD configurations, respectively. For high resolution sequences, the average BD-rate gains are 0.3%, 0.3% and 0.2% for AI, RA and LD configurations, respectively. More specifically, the proposed context modeling can achieve better coding performance than that in [19] under AI and RA configurations and on the high resolution sequences (Class B and UHD) under LD configuration. It achieves worse coding performance than that in [19] on Classes D, E and F under LD configuration.

Since the portion of *coeff_abs_greater1_flag* in the bitstream is much larger than that of *coeff_abs_greater2_flag*, we only make an analysis in the following to the context modeling for *coeff_abs_greater1_flag* in HEVC, [19] and this paper to explain the experimental results.

In HEVC, a context model set is first selected depending on whether there is a *coeff_abs_greater1_flag* equal to 1 in the preceding SB, and then a specific context model is selected with *NumEqu*1 and *NumGre*1 of the coded transform coefficient levels in the current SB. In the proposed context modeling, the context model is selected with *NumEqu*1 and *NumGre*1 of the neighbors covered by the local template of the current transform coefficient level. Take Fig. 8 as an example, x is the current transform coefficient level and the preceding SB is either *RightSB* (in horizontal scanning pass) or *LowerSB* (in vertical and diagonal scanning passes). Thus, as shown in Fig. 8, the neighbors covered by the local template are more close to x and the neighbors far away from x will not be used in the proposed context modeling. So the proposed context modeling has stronger ability to distinguish the probability distributions with different statistical properties for *coeff_abs_greater1_flag* at a position in one TB.

In [19], the context model for *coeff_abs_greater1_flag* is selected only with *NumGre*1 of the neighbors covered by the local template of the current transform coefficient level. Compared with [19], the number of context models for *coeff_abs_greater1_flag* in this paper is higher, since the information from the previous scan pass *NumEqu*1 is used as the context in addition to *NumGre*1 in the current scan pass. There are only a few significant transform coefficient

Table 8
BD-Rate (%) of the proposed context modeling for level information over that in HEVC.

	AI			*RA*			*LD*		
Sequences	*Y*	*U*	*V*	*Y*	*U*	*V*	*Y*	*U*	*V*
Class A	−0.5	−0.6	−0.5	−0.2	−0.2	0.0			
Class B	−0.4	−0.3	−0.4	−0.2	0.1	−0.4	−0.1	0.0	0.2
Class C	−0.3	−0.4	−0.4	−0.1	0.1	−0.1	0.0	0.1	0.3
Class D	−0.2	−0.4	−0.1	−0.1	0.7	−0.1	−0.1	−0.1	−0.1
Class E	−0.5	−0.6	−0.4				−0.3	0.3	0.3
Overall	**−0.4**	**−0.4**	**−0.3**	**−0.2**	**0.2**	**−0.2**	**−0.1**	**0.0**	**0.2**
Class F	−0.1	−0.2	−0.3	0.1	0.1	0.2	0.0	−0.1	−0.3
UHD	−0.5	−0.5	−0.6	−0.3	0.3	0.2	−0.2	0.1	−0.1
Overall(HD)	**−0.5**	**−0.4**	**−0.5**	**−0.3**	**0.1**	**−0.1**	**−0.1**	**−0.1**	**0.1**
Enc time (%)	105			100			101		
Dec time (%)	102			100			100		

Table 9
BD-Rate (%) of the proposed context modeling for level information over that in [19].

	AI			*RA*			*LD*		
Sequences	*Y*	*U*	*V*	*Y*	*U*	*V*	*Y*	*U*	*V*
Class A	−0.3	−0.3	−0.1	−0.2	0.0	−0.1			
Class B	−0.3	−0.1	−0.2	−0.2	0.2	−0.2	−0.2	0.0	−0.3
Class C	−0.1	−0.3	0.0	−0.1	−0.1	−0.2	0.0	−0.2	−0.3
Class D	0.0	0.0	−0.1	0.0	1.5	−0.1	0.2	0.1	−0.2
Class E	−0.3	−0.1	−0.3				0.1	0.8	1.9
Overall	**−0.2**	**−0.2**	**−0.1**	**−0.1**	**0.3**	**0.1**	**0.0**	**0.1**	**0.1**
Class F	0.0	0.0	−0.3	−0.2	0.0	−0.2	0.2	0.3	0.0
UHD	−0.3	−0.2	−0.3	−0.6	−0.2	−0.3	−0.1	−0.2	−0.1
Overall(HD)	**−0.3**	**−0.2**	**−0.2**	**−0.3**	**0.0**	**−0.2**	**−0.2**	**−0.1**	**−0.2**
Enc time (%)	100			100			101		
Dec time (%)	100			100			100		

levels in TB for low resolution sequences (Class D) or video-conference sequences (Class E) under LD configuration, thus the context dilution problem may occur due to the limitation of the samples that are used to accurately estimate context model parameters. So the proposed context modeling achieves worse coding performance than that in [19]. For AI and RA configurations or high resolution sequences, there are enough samples to accurately estimate context model parameters, so the proposed context modeling achieves better coding performance than that in [19]. Since the target of the next generation video coding standard is to process high resolution video sequences, the proposed context modeling will be appropriate for the next generation video coding standard.

Here we also provide the estimated bits for coding *coeff_abs_greater1_flag* at some positions within a TB according to Eq. (16). Table 10 lists the estimated bits for coding *coeff_abs_greater1_flag* at different positions in 8×8 TBs on some sequences, where the position is represented as the (X, Y) coordinate relative to DC coefficient. The sequences *BasketballDrill*_832 × 480 and *Kimono*_1920 × 1080 are coded at $QP=22$ under RA configuration, and *RaceHorses*_416 × 240 is coded at $QP=27$ under LD configuration. From Table 10, it can be seen that the estimated bits in HEVC is smaller than that in [19] for some positions on *BasketballDrill*_832 × 480 and *Kimono*_1920 × 1080, i.e. (1, 1) on *BasketballDrill*_832 × 480, (1, 1) and (3, 3) on *Kimono*_1920 × 1080. This demonstrates the efficiency of the context *NumEqu*1 from the previous scan pass in the context modeling for *coeff_abs_greater1_flag*. For position (1, 1) on *RaceHorses*_416 × 240, the estimated bits in [19] are smaller than that in this paper, since context dilution is occurred in this paper due to the limitation of training samples. For other positions besides (1, 1) on *RaceHorses*_416 × 240, the estimated bits in this paper is smaller than that in [19], since *NumEqu*1 can help to discriminate the probability distributions with different statistical properties.

Table 10
The estimated bits for coding *coeff_abs_greater1_flag* at different positions in 8×8 TBs on some sequences.

Sequences	Position	HEVC	[19]	This paper
*RaceHorses*_416 × 240	(1, 1)	0.724	0.708	0.712
	(3, 3)	0.487	0.475	0.462
*BasketballDrill*_832 × 480	(1, 1)	0.860	0.873	0.846
	(3, 3)	0.620	0.618	0.591
*Kimono*_1920 × 1080	(1, 1)	0.753	0.758	0.711
	(3, 3)	0.383	0.397	0.339

In regard of context dependency, the proposed context modeling introduces the context dependency between different scan passes, since it uses the information from the previous scan pass as the contexts of *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag*. However, as shown in Eqs. (12) and (13), *NumEqu*1 is used for *coeff_abs_greater1_flag* only when *NumGre*1 is equal to zero, and *NumGre*1 is used for *coeff_abs_greater2_flag* only when *NumGre*2 is equal to zero. In other words, the context modeling for *coeff_abs_greater1_flag* can be performed without referring *NumEqu*1 if *NumGre*1 is greater than zero, and the context modeling for *coeff_abs_greater2_flag* can also be executed without referring *NumGre*1 if *NumGre*2 is greater than zero. Thus this context dependency can be reduced as much as possible. Compared with the context modeling for *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* in [19], the proposed context modeling can further improve the coding performance, especially for high resolution sequences. So, in our opinions, the proposed context modeling for *coeff_abs_greater1_flag* and *coeff_abs_greater2_flag* is a good balance between coding performance and parallelism.

4.2. The binary arithmetic coding engine with low memory requirement

In this section, the coding efficiency of the BAC engine with low memory requirement is presented. We adopt the M-coder in HEVC, the BAC engine in [24] with look-up table and the BAC engine in [24] with straightforward multiplication (multiplication with bit capacity 15×9) as the competing methods.

The coding performance of the BAC engine with low memory requirement is listed in Table 11, in which the M-coder in HEVC is used as the benchmark to compute BD-Rate. Table 11 shows that on average, the BAC engine with low memory requirement reduces the bitrate by 0.7% for AI, 0.6% for RA and 0.5% for LD

Table 11
BD-Rate (%) the proposed BAC engine over that in HEVC.

	AI			*RA*			*LD*		
Sequences	*Y*	*U*	*V*	*Y*	*U*	*V*	*Y*	*U*	*V*
Class A	−0.8	−1.2	−0.8	−0.8	−0.9	−0.5			
Class B	−0.7	−0.8	−0.6	−0.7	−0.3	−0.2	−0.7	−0.2	0.2
Class C	−0.7	−0.8	−0.7	−0.5	−0.4	−0.4	−0.4	0.0	0.2
Class D	−0.6	−0.7	−0.4	−0.4	−0.5	−0.3	−0.4	−0.6	0.5
Class E	−0.7	−1.3	−0.8				−0.5	−1.0	0.1
Overall	**−0.7**	**−0.9**	**−0.6**	**−0.6**	**−0.5**	**−0.3**	**−0.5**	**−0.4**	**0.3**
Class F	−0.5	−0.5	−0.4	−0.2	0.0	0.2	−0.4	−0.8	−0.9
UHD	−0.8	−0.7	−0.7	−0.8	−0.1	−0.4	−0.8	−0.3	−0.6
Enc Time (%)	108			103			103		
Dec Time (%)	102			101			100		

Table 12
BD-Rate (%) of the proposed BAC engine over that in [24] with look-up table.

	AI			*RA*			*LD*		
Sequences	*Y*	*U*	*V*	*Y*	*U*	*V*	*Y*	*U*	*V*
Class A	−0.01	−0.01	−0.01	−0.01	−0.01	−0.01			
Class B	−0.02	−0.02	−0.02	−0.02	−0.02	−0.02	−0.02	−0.02	−0.02
Class C	−0.01	−0.01	−0.01	−0.01	−0.01	−0.01	−0.02	−0.02	−0.02
Class D	−0.03	−0.02	−0.02	−0.02	−0.02	−0.02	−0.03	−0.03	−0.03
Class E	−0.04	−0.03	−0.03				−0.03	−0.03	−0.03
Overall	**−0.02**	**−0.02**	**−0.02**	**−0.02**	**−0.02**	**−0.01**	**−0.02**	**−0.02**	**−0.02**
Class F	−0.01	−0.01	−0.01	−0.01	−0.01	−0.01	0.00	0.00	0.00
UHD	−0.01	0.00	0.00	−0.01	−0.01	−0.01	−0.01	−0.01	−0.01
Enc time (%)	93			98			99		
Dec time (%)	96			99			99		

compared with M-coder in HEVC. This is mainly because the probability update with two different adaption speeds can get more accurate probability in probability estimation module. As shown in Table 11, the encoding and decoding time of the proposed BAC engine are similar as that of M-coder in HEVC. The improvement in encoding time is mainly from RDO (Rate Distortion Optimization) procedure, in which a counter is used to switch the probability model for each binary symbol.

Table 12 provides the coding performance comparisons between the proposed BAC engine and the BAC engine in [24] with look-up table. Compared with the BAC engine in [24] with look-up table, the proposed BAC engine can achieve a little better coding performance. This demonstrates the accuracy of the multiplication with low bit capacity used in the coding interval subdivision. It can also be seen that the encoding time and decoding time of the proposed BAC engine are lower than that in [24] with look-up table from Table 12. This is because the size of the table in [24] is too large and the processor cannot pre-fetch all entries of the table into the cache, thus the processor will carry the entry from memory to cache when the accessed entry is not in the cache and this process will consume more time. Since only intra prediction is available under AI configuration and the RDO process is also simple in this configuration, there are more significant transform coefficient levels under AI configuration and the proportion of time consumption on CABAC is also larger compared with RA and LD configurations. So the time reduction on encoding/decoding time under AI configuration is more than that under RA and LD configurations.

We also make an analysis on the memory requirement of CABAC in [24] to state the significance of such memory reduction. According to [34], there are 154 context models in HEVC and each context model requires 8bits to store the initial value and 7 bits to store the probability. In addition, HEVC requires a line buffer size of 1536 bits for $4k \times 2k$ sequences if the minimum CU size is equal to 8×8. So the CABAC in [24] with look-up table will require 38 times memory more than the CABAC within HEVC, and the look-up table used in the coding interval subdivision accounts for nearly 98% of the memory requirement of the CABAC in [24]. Therefore, the multiplication with low bit capacity will reduce the memory size of CABAC in [24] by 98%.

Table 13 shows the coding performance comparisons between the proposed BAC engine and the BAC engine in [24] with straightforward multiplication. The proposed BAC engine achieves the similar coding performance as that in [24] with straightforward multiplication from Table 13, thus the multiplication with low bit capacity can achieve the similar accuracy as the straightforward multiplication.

In hardware implementation, the multiplication is implemented based on the addition and shift operations. The multiplication of two binary numbers is generally implemented as follows: for each bit in the multiplier, shift the multiplicand left and add the shifted multiplicand to the product if this bit is equal to one. The multiplication with bit capacity 15×9 will require a 24-bits register to store the result and consume 9 shift operations and 9 24-bits addition operations to accomplish this multiplication; while the multiplication with bit capacity 9×5 requires a 14-bits register to store the result and consumes 5 shift operations and 5 14-bits addition operations to accomplish the multiplication. Compared with the multiplication with bit capacity 15×9, the multiplication with bit capacity 9×5 can reduce the hardware complexity and power consumption while keeping the same time consumption.

4.3. Overall coding efficiency of the proposed two techniques

The overall coding efficiency of the proposed modification to CABAC is evaluated with respective to the original entropy coding scheme in HEVC. Table 14 shows that on average, the proposed

Table 13
BD-Rate (%) of the proposed BAC engine over that in [24] with straightforward multiplication.

	AI			*RA*			*LD*		
Sequences	*Y*	*U*	*V*	*Y*	*U*	*V*	*Y*	*U*	*V*
Class A	0.00	0.00	0.00	0.00	0.00	0.00			
Class B	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class C	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Class D	−0.01	−0.01	−0.01	0.00	0.00	0.00	0.00	0.00	0.00
Class E	−0.01	0.00	−0.01				0.02	0.02	0.02
Overall	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Class F	0.00	0.00	0.00	0.01	0.01	0.01	0.03	0.03	0.03
UHD	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
Enc time (%)	95			97			99		
Dec time (%)	99			100			100		

Table 14
BD-Rate (%) of the proposed two techniques in this paper.

	AI			*RA*			*LD*		
Sequences	*Y*	*U*	*V*	*Y*	*U*	*V*	*Y*	*U*	*V*
Class A	−1.7	−1.7	−1.5	−1.4	−1.1	−1.0			
Class B	−1.5	−1.5	−1.5	−1.2	−0.9	−0.9	−1.0	−0.6	−0.4
Class C	−1.3	−1.1	−1.1	−1.1	−0.8	−0.8	−0.9	−0.1	−0.3
Class D	−1.2	−0.9	−0.6	−1.0	0.2	−0.8	−1.0	−0.6	−0.3
Class E	−1.7	−1.9	−1.5				−0.9	−1.9	−0.6
Overall	**−1.4**	**−1.4**	**−1.2**	**−1.1**	**−0.6**	**−0.9**	**−0.9**	**−0.7**	**−0.4**
Class F	−2.0	−2.0	−2.0	−1.9	−1.5	−1.3	−1.6	−1.8	−2.3
UHD	−1.4	−1.6	−1.6	−1.3	−0.2	−0.3	−1.2	−0.1	−0.3
Enc time (%)	119			106			107		
Dec time (%)	104			102			101		

two techniques can reduce the bitrate by 1.4% for AI, 0.9% for LD and 1.1% for RA when compared to the original entropy coding scheme in HEVC. This demonstrates that the gains of the improved context modeling for transform coefficient levels and the gains of the BAC engine with low memory requirement are additive.

5. Conclusion

In this paper, we propose a modification to CABAC within HEVC to improve its coding performance under low memory requirement, which consists of an improved context modeling for transform coefficient levels and a binary arithmetic coding (BAC) engine with low memory requirement. For the improved context modeling, the statistical features of the transform coefficient levels within HEVC are first presented, and then the coefficient position and the neighbors covered by a local template of the current transform coefficient level are used to design the context models. For *significant_coeff_flag*, TBs are split into different regions based on the frequency of the transform coefficient level. For each region, the specific context model is determined by the number of the significant neighbors covered by the local template of the current transform coefficient level. To limit the total number of the context models, the same region from different TBs shares the same context model set. For *coeff_abs_greater1_flag*, the number of neighbors covered by the local template with absolute magnitude equal to 1 and larger than 1 are used as the contexts. Moreover, the coefficient position is incorporated in the context model selection to capture the statistical features of the transform coefficient levels at different frequencies. The number of neighbors covered by the local template with absolute magnitude larger than 1 and larger than 2 are used as the contexts of *coeff_abs_greater2_flag*. For the BAC engine with low memory requirement, the multi-parameter probability update mechanism is used to update the probability. Moreover, the multiplication with low bit capacities is used in the coding interval subdivision instead of the large look-up table to reduce the memory consumption. Experiments conducted on HM14.0 under main profile demonstrate that each technique can improve the coding efficiency for HEVC, and the further coding efficiency improvement can be achieved by adopting the two techniques together.

Acknowledgments

This work was supported by the Major State Basic Research Development Program of China (973 Program 2015CB351804) and the National Science Foundation of China (NSFC) under Grant 61272386. The authors would like to thank the Media Lab in Shanghai Jiao Tong University (SJTU) for providing the UHD sequences and thank the reviewers for their constructive criticism.

References

[1] G.J. Sullivan, J. Ohm, W. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Trans. Circuits Syst. Video Technol. 22 (December (12)) (2012) 1649–1668.
[2] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Trans. Circuits Syst. Video Technol. 13 (July (7)) (2003) 560–576.
[3] J. Ohm, G.J. Sullivan, H. Schwarz, T.K. Tan, T. Wiegand, Comparison of the coding efficiency of video coding standards including high efficiency video coding (HEVC), IEEE Trans. Circuits Syst. Video Technol. 22 (December (12)) (2012) 1669–1684.
[4] I. Kim, J. Min, T. Lee, W. Han, J. Park, Block partitioning structure in the HEVC standard, IEEE Trans. Circuits Syst. Video Technol. 22 (December (12)) (2012) 1697–1706.
[5] T. Nguyen, P. Helle, M. Winken, B. Bross, D. Marpe, H. Schwarz, T. Wiegand, Transform coding techniques in HEVC, IEEE J. Sel. Top. Signal Process. 7 (December (6)) (2013) 978–989.
[6] J. Sole, R. Joshi, N. Nguyen, T. Ji, M. Karczewicz, G. Clare, F. Henry, A. Duenas, Transform coefficient coding in HEVC, IEEE Trans. Circuits Syst. Video Technol. 22 (December (12)) (2012) 1765–1777.
[7] J. Sole, R. Joshi, M. Karczewicz, Non-CE11: diagonal sub-block scan for HE residual coding, JCTVC-G323, in: 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, November 2011.
[8] J. Sole, R. Joshi, M. Karczewicz, CE11: scanning passes of residual data in HE, JCTVC-G320, in: 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, November 2011.
[9] J. Sole, R. Joshi, M. Karczewicz, CE11: unified scans for the significance map and coefficient level coding in high efficiency, JCTVC-F288, in: 6th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Torino, Italy, July 2011.
[10] Qualcomm Incorporated, Coding tools investigation for next generation video coding, ITU-T SG16 CONTRIBUTION 806, January 2015.
[11] D. Marpe, H. Schwarz, T. Wiegand, Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard, IEEE Trans. Circuits Syst. Video Technol. 13 (July (7)) (2003) 620–636.
[12] G. Korodi, J. Zan, D. He, Encoding and decoding significant coefficient flags for small transform units using partition sets, JCTVC-G657, in: 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, November 2011.
[13] T. Nguyen, H. Schwarz, H. Kirchhoffer, D. Marpe, T. Wiegand, Improved context modeling for coding quantized transform coefficients in video compression, in: Picture Coding Symposium (PCS), December 2010, pp. 378–381.
[14] T. Nguyen, D. Marpe, T. Wiegand, Non-CE11: proposed cleanup for transform coefficient coding, JCTVC-H0288, in: 8th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, San Jose, USA, February 2012.
[15] T. Kumakura, S. Fukushima, Non-CE3: simplified context derivation for significance map, JCTVC-I0296, in: 9th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, April 2012.
[16] J. Sole, R. Joshi, M. Karczewicz, Removal of $8 \times 2/2 \times 8$ coefficient groups, JCTVC-J0256, in: 10th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Stockholm, Sweden, July 2012.
[17] M. Budagavi, M.U. Demircin, Parallel context processing techniques for high coding efficiency entropy coding in HEVC, JCTVC-B088, in: 2nd Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, July 2010.
[18] Y. Piao, Y. Hong, I.-K. Kim, J.H. Park, Cross-check results for JCTVC-J0228, JCTVC-J0408, in: 10th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Stockholm, Sweden, July 2012.
[19] J. Chen, W. Chien, M. Karczewicz, X. Li, H. Liu, A. Said, L. Zhang, X. Zhao, Further improvements to HMKTA-1.0, VCEG-AZ07.

[20] E. Belyaev, M. Gilmutdinov, A. Turlikov, Binary arithmetic coding system with adaptive probability estimation by virtual sliding window, in: IEEE International Symposium on Consumer Electronics (ISCE), 2006, pp. 194–198.
[21] E. Belyaev, A. Turlikov, K. Egiazarian, M. Gabbouj, An efficient adaptive binary arithmetic coder with low memory requirement, IEEE J. Sel. Top. Signal Process. 7 (June (6)) (2013) 1053–1061.
[22] D. Marpe, T. Wiegand, A highly efficient multiplication-free binary arithmetic coder and its application in video coding, in: IEEE International Conference on Image Processing (ICIP), September 2003, pp. 263–266.
[23] H. Eeckhaut, B. Schrauwen, M. Christiaens, J. Campenhout, Tuning the M-coder to improve diracs entropy coding, WSEAS Trans. Inf. Sci. Appl. (2005) 1563–1571.
[24] A. Alshin, E. Alshina, Multi-parameter probability up-date for CABAC, JCTVC-F254, in: 6th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Torino, Italy, July 2011.
[25] V. Sze, A. Chandrakasan, Joint algorithm-architecture optimization of CABAC to increase speed and reduce area cost, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2011, pp. 1577–1580.
[26] K. Sugimoto, A. Minezawa, S. Sekiguchi, K. Asai, T. Murakami, Reduction of initialization tables for CABAC contexts, JCTVC-H0646, in: 8th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, San Jose, USA, February 2012.
[27] T. Chuang, C. Chen, Y. Huang, S. Lei, CABAC with a reduced LPS range table, JCTVC-F061, in: 6th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Torino, Italy, July 2011.
[28] E. Belyaev, K. Liu, M. Gabbouj, Y. Li, An Efficient adaptive binary range coder and its VLSI architecture, IEEE Trans. Circuits Syst. Video Technol. 25 (November (8)) (2014) 1435–1446.
[29] A. Moffat, R. Neal, I. Witten, Arithmetic coding revisited, ACM Trans. Inf. Syst. 16 (July) (1998) 256–294.
[30] E. Lam, J. Goodman, A mathematical analysis of the DCT coefficient distributions of images, IEEE Trans. Image Process. 9 (October (10)) (2000) 1661–1666.
[31] F. Bossen, Common HM test conditions and software reference configurations, JCTVC-E700, in: 5th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland, March 2011.
[32] L. Song, X. Tang, W. Zhang, X. Yang, P. Xia, The SJTU 4K video sequence dataset, in: International Workshop on Quality of Multimedia Experience (QoMEX), July 2013, pp. 34–35.
[33] G. Bjontegaard, Calculation of Average PSNR Differences Between RD-Curves, Doc. ITU-T VCEG (VCEG-M33), Austin, Texas, USA, April 2001.
[34] V. Sze, M. Budagavi, High throughput CABAC entropy coding in HEVC, IEEE Trans. Circuits Syst. Video Technol. 22 (December (12)) (2012) 1778–1791.