Quantization and Entropy Coding in the Versatile Video Coding (VVC) Standard

Heiko Schwarz, Muhammed Coban, Marta Karczewicz, Tzu-Der Chuang, Frank Bossen, *Senior Member, IEEE,* Alexander Alshin, Jani Lainema, and Christian R. Helmrich, *Senior Member, IEEE*

***Abstract*—The paper provides an overview of the quantization and entropy coding methods in the Versatile Video Coding (VVC) standard. Special focus is laid on techniques that improve coding efficiency relative to the methods included in the High Efficiency Video Coding (HEVC) standard: The inclusion of trellis-coded quantization, the advanced context modeling for entropy coding transform coefficient levels, the arithmetic coding engine with multi-hypothesis probability estimation, and the joint coding of chroma residuals. Beside a description of the design concepts, the paper also discusses motivations and implementation aspects. The effectiveness of the quantization and entropy coding methods specified in VVC is validated by experimental results.**

***Index Terms*—Versatile Video Coding (VVC), quantization, entropy coding, transform coefficient coding, video coding.**

I. INTRODUCTION

THE Versatile Video Coding (VVC) standard [1], [2] is the most recent joint video coding standard of the ITU-T and ISO/IEC standardization organizations. It was developed by the Joint Video Experts Team (JVET), a partnership between the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). VVC was technically finalized in July 2020 and will be published as ITU-T Rec. H.266 and ISO/IEC 23090-3 (MPEG-I Part 3).

The primary objective of the new VVC standard is to provide a significant increase in compression capability compared to its predecessor, the *High Efficiency Video Coding* (HEVC) standard [3]. At the same time, VVC includes design features that make it suitable for a broad range of video applications. In addition to conventional video applications, it particularly addresses the coding of video with high dynamic range and wide color gamut, computer-generated video (e. g., for remote screen sharing or gaming), and omnidirectional video and it supports adaptive streaming with resolution switching, scalable coding, and tile-based streaming for immersive applications. Despite the rich set of coding tools and functionalities, particular care was taken to enable decoder implementations with reasonable complexity in both hardware and software.

Similar to all previous video coding standards of the ITU-T and ISO/IEC since H.261 [4], the VVC design follows the general concept of block-based hybrid video coding. The video pictures are partitioned into rectangular blocks and each block is predicted by intra- or inter-picture prediction. The resulting prediction error blocks are coded using transform coding, which consists of an orthogonal transform, quantization of the transform coefficients, and entropy coding of the resulting quantization indexes. Quantization artifacts are attenuated by applying so-called in-loop filters to reconstructed pictures before they are output or used as references for inter-picture prediction of following pictures.

Although VVC uses the same coding framework as its predecessors, it includes various improvements that eventually result in a substantially improved compression performance. One of the most prominent changes in comparison to HEVC is the very flexible block partitioning concept [5] that supports non-square blocks for coding mode selection, intra-picture prediction, inter-picture prediction, and transform coding and, thus, impacts the design of many other aspects. In the present paper, we describe modifications to quantization and entropy coding. The coding efficiency improvements in this area can be mainly attributed to the following four features:

- the support of trellis-coded quantization (TCQ);
- the advanced entropy coding of quantization indexes suitable for both TCQ and scalar quantization;
- the binary arithmetic coding engine with multi-hypothesis probability estimation;
- the support of joint chroma residual coding.

Theses changes in quantization and entropy coding together with a block-adaptive transform selection [6] eventually led to a substantially increased efficiency of the transform coding design in VVC compared to that of HEVC.

The paper is organized as follows. Section II describes the quantization in VVC with special focus on the TCQ design. The entropy coding of quantization indexes including context modeling is presented in Section III. Section IV discusses the improvements of the core binary arithmetic coding engine. The joint coding of chroma prediction errors is described in Section V. Experimental results validating the effectiveness of the quantization and entropy coding tools are provided in Section VI, and Section VII concludes the paper.

Manuscript uploaded August 22, 2020.

H. Schwarz and C. R. Helmrich are with the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, 10587 Berlin, Germany. H. Schwarz is also with the Institute of Computer Science, Free University of Berlin, 14195 Berlin, Germany (e-mail: heiko.schwarz@hhi.fraunhofer.de; christian.helmrich@hhi.fraunhofer.de).

M. Coban and M. Karczewicz are with Qualcomm Technologies Inc., San Diego, CA 92121, USA (e-mail: mcoban@qti.qualcomm.com; martak@qti.qualcomm.com).

T.-D. Chuang is with MediaTek Inc., Hsinchu 30078, Taiwan (e-mail: peter.chuang@mediatek.com).

F. Bossen is with Sharp Electronics of Canada Ltd., Mississauga, ON L4Z 1W9, Canada (e-mail: fbossen@sharpsec.com).

A. Alshin is with Intel Russia, Moscow, 121614 (e-mail: alexander.alshin@intel.com).

J. Lainema is with Nokia Technologies, 33101 Tampere, Finland (e-mail: jani.lainema@nokia.com).

II. Quantization

Quantization is an irreversible mapping of input values to output values. For the specification in image and video coding standards, it is split into a non-normative encoder mapping of input samples to integer *quantization indexes*, which are also referred to as *levels* and are transmitted using entropy coding, and a normative decoder mapping of the quantization indexes to reconstructed samples. The aim of quantization is to decrease the bit rate required for transmitting the quantization indexes while maintaining a low reconstruction error.

In hybrid video coding, quantization is generally applied to transform coefficients that are obtained by transforming prediction error blocks (also referred to as *residual blocks*) using an approximately orthogonal transform. The transforms used have the property that, for typical residual blocks, the signal energy is concentrated into a small number of transform coefficients. This has eventually the effect that simple scalar quantizers are more effective in the transform domain than in the original sample space [7]. In particular for improving the coding efficiency for screen content [8], where residual blocks often have different properties, VVC also provides a *transform skip* (TS) mode, in which no transform is applied, but the residual samples are quantized directly.

Similarly as in AVC (*Advanced Video Coding*) [9] and HEVC, the quantizer design in VVC is based on scalar quantization with *uniform reconstruction quantizers*. But VVC also includes two extensions that can improve coding efficiency at the cost of an increased encoder complexity.

A. Basic Design: Uniform Reconstruction Quantizers

In scalar quantization, the reconstructed value t'_k of each input coefficient (or sample) t_k depends only on the associated quantization index q_k. Uniform reconstruction quantizers (URQs) are a simple variant, in which the set of admissible reconstruction values is specified by a single parameter, called *quantization step size* Δ_k. The decoder operation is given by a simple *scaling*, $t'_k = \Delta_k\, q_k$. Similar as previous ITU-T and ISO/IEC video coding standards, VVC supports quantization weighting matrices by which the quantization step size can be varied across the transform coefficients of a block. Conceptually, the step size for a coefficient t_k is given by $\Delta_k = \alpha_k \Delta$, where α_k is a weighting factor that depends on the location of the coefficient t_k inside the transform block and Δ is a quantization step size, which can be selected on a block basis among a pre-defined set of candidates. The chosen Δ is indicated by a non-negative integer value referred to as *quantization parameter* (QP). VVC uses an exponential relationship between Δ and QP, which was originally introduced in AVC. When neglecting rounding operations, the reconstruction of transform coefficients can be written as

$$t'_k = \alpha_k \cdot 2^{(\mathrm{QP}-4)/6} \cdot q_k. \tag{1}$$

An increase of the quantization parameter by one corresponds to an increase of about 12% for the quantization step size.

For avoiding reconstruction mismatches, the entire VVC decoding process is specified using exact integer operations (similar to AVC and HEVC). In comparison to the idealized case with orthogonal transforms, the inverse transform for a $W{\times}H$ block includes an additional scaling by $\sqrt{WH}\cdot 2^{B-15}$, where B represents the bit depth of the color component in bits per sample. Consequently, the scaling in the decoder has to approximately generate reconstructed coefficients

$$t'_k = \alpha_k \cdot 2^{(\mathrm{QP}-4)/6} \cdot 2^{15-B} \cdot (WH)^{-1/2} \cdot q_k, \tag{2}$$

which are then used as input values to the inverse transform. With $\beta = \lceil \frac{1}{2}\log_2 WH \rceil$, $\gamma = 2\beta - \log_2 WH$, $p = \lfloor \mathrm{QP}/6 \rfloor$, and $m = \mathrm{QP}\,\%\,6$, where $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ denote the ceiling and floor functions, respectively, and $\%$ denotes the modulus operator, the mapping $q_k \mapsto t''_k$ can be rewritten according to

$$t'_k = \left(2^4 \alpha_k\right) \cdot \left(2^{(32+3\gamma+m)/6}\right) \cdot 2^p \cdot 2^{5-\beta-B} \cdot q_k. \tag{3}$$

Since both the width W and the height H of a transform block are integer powers of two, $\gamma \in \{0, 1\}$ is a binary parameter.

For obtaining a realization with integer operations, the two terms in parenthesis are rounded to integer values and the multiplication with $2^{5-\beta-B}$ is approximated by a bit shift. The VVC standard specifies the reconstruction according to

$$t'_k = \big(w_k \cdot (a[\gamma][m] \ll p) \cdot q_k + ((1 \ll b) \gg 1)\big) \gg b, \tag{4}$$

where $\ll$ and $\gg$ denote bit shifts to the left and right (in two's complement arithmetic), respectively, and $b = B + \beta - 5$. The $2{\times}6$ array $a[\gamma][m]$ specifies integer values that approximate the terms $2^{(32+3\gamma+m)/6}$. It is given by $a = \{\{40, 45, 51, 57, 64, 72\}, \{57, 64, 72, 80, 90, 102\}\}$. The integer values $w_k = \mathrm{round}(2^4\alpha_k)$, with $w_k \in [1; 255]$, are called *scaling list*. As further detailed in Section II-F, scaling lists for different block types can be specified in a corresponding high-level data structure. If scaling lists are not used, the values w_k are inferred to be equal to 16, which corresponds to $\Delta_k = \Delta$.

In transform skip mode, no inverse transform is applied and, hence, no additional scaling factor has to be included in the reconstruction process of residual samples r'_k. Furthermore, the concept of scaling lists is not applicable. An integer realization of the reconstruction $r'_k = \Delta \cdot q_k$ is obtained by using (4) with $w_k = 16$, $\gamma = 0$, and $b = 10$, which yields

$$r'_k = \big((a[0][m] \ll (p+4)) \cdot q_k + 512\big) \gg 10. \tag{5}$$

B. Quantization Improvements

If one only considers scalar quantization, the restriction to URQs has no negative impact on coding efficiency. When combined with a suitable entropy coding and encoder decision, URQs can achieve virtually the same rate-distortion efficiency as optimal scalar quantizers for typical distributions of transform coefficients [10], [11]. However, even for statistically independent transform coefficients, the usage of scalar quantization results in an unavoidable loss in coding efficiency relative to the fundamental rate-distortion bound. This gap can only be reduced by using vector quantizers (VQs) [12].

VVC includes two advanced techniques for quantizing transform coefficients that are referred to as *sign data hiding* and *trellis-coded quantization*. Both have properties of VQs, but also represent simple extensions of URQs and require only minimal changes of the decoding process. Since these

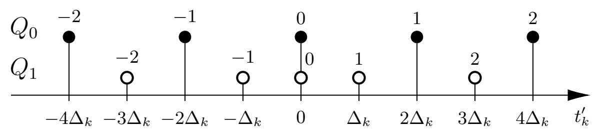

Fig. 1. Scalar quantizers Q_0 and Q_1. The circles indicate the reconstruction levels and the labels represent the associated quantization indexes.

approaches cannot utilize statistical dependencies of the input data, the basic concept of transform coding is not modified. The dependencies between residual samples are exploited by applying the quantization in the transform domain and by using an appropriate entropy coding method.

C. Sign Data Hiding

Sign data hiding (SDH) [13]–[15] is a technique that is already included in HEVC and hasn't been modified in the context of VVC. Consider a block of reconstructed transform coefficients $\{t'_k\}$ that is represented by a corresponding block of quantization indexes $\{q_k\}$, with $t'_k = \Delta_k q_k$. The basic idea of SDH is to omit the coding of the sign for one nonzero index in $\{q_k\}$ and instead derive it from the parity of the sum of absolute values $|q_k|$. In comparison to scalar quantization with the same step sizes Δ_k, SDH saves about 1 bit per block, which for suitably large blocks outweighs the average increase in distortion. But note that an encoder has to carefully select quantization indexes $\{q_k\}$ that obey the sign hiding condition in order to achieve coding efficiency improvements.

In HEVC and VVC, SDH is applied on the basis of so-called *coefficient groups* (CGs), which represent groups of successive levels q_k in coding order (see Section III); in most cases, they include 16 levels. If the difference between the scan indexes of the last and first nonzero level (in coding order) inside a CG is greater than 3, the sign for the last nonzero level of the CG is not coded but derived based on the sum of absolute values, $\sum_{k\in\mathrm{CG}} |q_k|$, where odd sums indicate negative values. At the decoder side, SDH does not require any change of the scaling in (4), only the entropy coding of sign flags is modified.

D. Trellis-Coded Quantization

The second improvement [16], [17] employs the concept of trellis-coded quantization (TCQ), first described in [18]. Since the reconstruction process specified in the standard does not use trellis structures, the TCQ design included in VVC is also referred to as *dependent quantization*. TCQ was well studied in the 1990s and it was demonstrated that it can significantly outperform the best scalar quantizers [18]–[21]. Due to its simple structure, it can be applied for quantizing vectors of arbitrary dimensions.

From a decoder perspective, TCQ specifies two scalar quantizers and a procedure for switching between these quantizers. The two scalar quantizers Q_0 and Q_1 used in VVC are illustrated in Fig. 1. Similar to URQs, the reconstruction levels of both quantizers represent integer multiples of a quantization step size Δ_k. The quantizer Q_0 includes the even multiples of Δ_k and the quantizer Q_1 includes the odd multiples of Δ_k and, in addition, the value of zero. Note that both quantizers are symmetric and include the reconstruction level equal to zero. This deviation from conventional TCQ designs improves the coding efficiency at low and medium rates without requiring significant changes of the entropy coding (in comparison to using URQs). For both quantizers Q_0 and Q_1, the selected reconstruction levels t'_k are indicated by integer quantization indexes q_k, as illustrated by the labels in Fig. 1.

TABLE I
STATE TRANSITION AND QUANTIZER SELECTION

current	quantizer	next state s_{k+1}	
state s_k	used for t'_k	$(q_k \,\&\, 1) = 0$	$(q_k \,\&\, 1) = 1$
0	Q_0	0	2
1	Q_0	2	0
2	Q_1	1	3
3	Q_1	3	1

In contrast to scalar quantization, the transform coefficients of a block have to be reconstructed in a pre-defined order, which shall be indicated by the index k. The reconstruction order is chosen to be equal to the coding order of quantization indexes q_k, which additionally enables the exploitation of certain TCQ properties in the entropy coding (see Section III). Given the reconstruction order, the procedure for switching between the two quantizers Q_0 and Q_1 can be specified by a state machine with 2^K states ($K \geq 2$), where the state s_k for a current coefficient t_k uniquely determines the quantizer used. The state s_{k+1} for the next coefficient t_{k+1} is determined by the current state s_k and the parity $p_k = (q_k \,\&\, 1)$ of the current quantization index q_k (the operator $\&$ represents a bit-wise and in two's complement arithmetic). Even though the achievable coding efficiency increases with number of states [18], [22], the TCQ design in VVC uses the minimal number of 4 states for limiting the required encoder complexity. The state transition and quantizer selection are given in Table I. The initial state s_0 is always set equal to zero.

The reconstruction of transform coefficients t'_k is specified as follows: First, the quantization indexes q_k for a block are mapped to integer multiplication factors q^*_k for the corresponding quantization step sizes Δ_k. And then, the multiplication $t'_k = q^*_k \Delta_k$ is approximated as in the conventional URQ case. For a block with N transform coefficients, the calculation of the factors q^*_k can be specified by the following algorithm, where stateTransTable represents the 4×2 state transition table given in Table I and $\mathrm{sgn}(\cdot)$ denotes the signum function

```
$s_0 = 0$
for $k = 0$ to $N - 1$ do
    $q^*_k = 2 \cdot q_k - (s_k \gg 1) \cdot \mathrm{sgn}(q_k)$
    $s_{k+1} = \mathrm{stateTransTable}[\, s_k \,][\, q_k \,\&\, 1 \,]$
end for
```

The distance between two neighboring reconstruction levels in the quantizers Q_0 and Q_1 is in most cases $2\Delta_k$, and not Δ_k as for URQs. For obtaining approximately the same reconstruction quality for a given QP, regardless of whether TCQ is enabled, the quantization step sizes Δ_k have to be scaled for TCQ. As verified experimentally, a scaling factor of $2^{-5/6}$, corresponding to a QP decrease of 5, represents

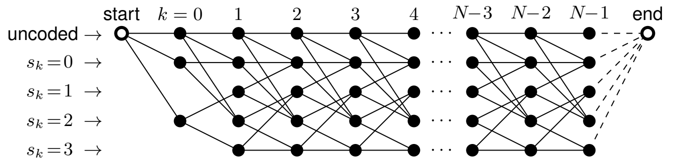

Fig. 2. Trellis structure used in the VTM reference encoder [28].

a suitable choice. Hence, when TCQ is enabled, the scaling $t'_k = q^*_k\Delta_k$ is specified by re-using (4), but with q_k being replaced with q^*_k and modified parameters $b = B + \beta - 4$, $p = \lfloor(\text{QP} + 1)/6\rfloor$, and $m = (\text{QP} + 1)\%6$.

For supporting TCQ at the decoder side, only three changes are required: (1) An additional mapping from levels q_k to multiplication factors q^*_k; (2) a modification of the scaling parameters b, p, and m; and (3) a state-dependent context selection, which will be described in Section III.

E. Encoder Operation

Even though the quantization process at the encoder, i. e., the algorithm for selecting levels q_k, is outside the scope of the standard, it has a significant impact on coding efficiency. State-of-the-art video encoders often use algorithms that select the levels $\boldsymbol{q} = \{q_k\}$ for a block by minimizing a Lagrangian function $J(\boldsymbol{q}) = D(\boldsymbol{q}) + \lambda R(\boldsymbol{q})$ of the MSE distortion $D(\boldsymbol{q})$ and the number of bits $R(\boldsymbol{q})$ required for transmitting the levels [23]–[25]. The Lagrange multiplier λ determines the operating point and is typically set depending on a base QP. These approaches take into account dependencies between levels q_k that are introduced in the entropy coding and are referred to as *rate-distortion optimized quantization* (RDOQ). An RDOQ algorithm suitable for URQs and the entropy coding design in HEVC and VVC is described in [26]. This algorithm is also implemented in the reference encoders HM [27] and VTM [28] for HEVC and VVC, respectively.

1) Sign Data Hiding: When SDH is enabled, an encoder has to ensure that the sign hiding condition (the parity of the sum of absolute levels correctly indicates the sign of the last nonzero index) is met for all CGs. This is typically achieved as follows [15]. First the RDOQ algorithm for URQs is applied. Then, in a second step, for all CGs for which the sign condition is violated, one of the levels q_k is increased or decreased by one. The corresponding level as well as the direction of the change are selected by minimizing the Lagrange cost $J(\boldsymbol{q})$.

2) Trellis-Coded Quantization: The quantizer switching in TCQ introduces dependencies, which have to be taken into account for achieving a good coding efficiency. The potential transitions between the quantizers Q_0 and Q_1 can be elegantly represented by a trellis with 4 states per coefficient [18]. The selection of indexes $\boldsymbol{q}$ for a block is then equivalent to finding the path with minimum $J(\boldsymbol{q})$ through the trellis.

For a better consideration of certain entropy coding aspects (coding of last position), the algorithm [17] implemented in the VTM software uses a trellis with 5 states, as shown in Fig. 2. In addition to the states 0–3, it includes an "uncoded" state, which represents levels equal to 0 that precede the first nonzero level in coding order. Note that the start and end states s_{k-1} and s_k, respectively, of a connection between two nodes uniquely determine the quantizer used and the parity of the associated level q_k. For each connection, the candidate level q_k that minimizes the difference $|t_k - t'_k(q_k)|$ between the original and reconstructed coefficients is determined first. Then, the final levels $\boldsymbol{q} = \{q_k\}$ for a block are selected among these candidates by applying the Viterbi algorithm[1] [29]. The cost assigned to a connection represents the contribution $D_k(q_k) + \lambda R_k(q_k|\cdot)$ of the associated candidate q_k to the overall cost $J(\boldsymbol{q})$. Preceding levels in the trellis paths are taken into account for calculating the rate terms $R(q_k|\cdot)$.

There are several possibilities for speeding up the encoding process, for example, by approximating the Lagrange costs or pruning unlikely connections. The VTM reference encoder uses a simple method that significantly reduces the encoder run time for typical video bit rates, at which most high-frequency coefficients are small compared to the quantization step size. In an initial step, the first original coefficient t_i in coding order with $|t_i| > \Delta_i/2$ is determined. The levels for all coefficients that precede this coefficient in coding order are set equal to zero and the Viterbi algorithm starts at $k = i$.

F. Quantization Control

As described above, VVC supports three quantizer designs (URQs, SDH, and TCQ) with different trade-offs between achievable coding efficiency and encoder complexity. An encoder can select the one that best suits the application requirements. The choice is indicated in the slice header.

For enabling both block-based rate control algorithms and perceptually optimized encoding approaches (e. g., [30]), the QPs can be selected on a block basis. The corresponding blocks are called *quantization groups* (QGs); their sizes are indicated in the picture header. The QPs for the luma component are coded differentially. For each QG that contains nonzero levels, the difference between the QP used and a prediction derived from spatially neighboring QGs is transmitted. For the chroma components, the QPs are derived from the luma QP of the co-located block via look-up tables. There are three different tables, one for the Cb component, one for the Cr component, and another one that is explicitly used for the JCCR modes with $|m| = 2$ (see Section V). For supporting a wide range of transfer functions and color formats, an encoder has the freedom to choose suitable look-up tables. They are defined by piece-wise linear mapping functions that are coded in the sequence parameter set. VVC supports QP values in the range from 0 to $63 + 6(B - 8)$, inclusive, where B denotes the bit depth of the corresponding color component.

As noted above, the quantization of transform coefficients can be additionally controlled by weighting matrices, which are specified using scaling lists. The main motivation is that the usage of frequency-dependent quantization step sizes can help an encoder to better take the contrast sensitivity behavior of human vision into account. In total, VVC includes 28 scaling lists, each defining weighing factors for a 2×2, 4×4, or 8×8

[1]Due to complicated dependencies in the entropy coding, the Viterbi algorithm does not yield the optimal solution, but it still provides a very good trade-off between coding efficiency and implementation complexity.

array of coefficients. The scaling lists can be transmitted in a high-level syntax structure referred to as *adaptation parameter set*; similarities between the different lists are exploited using predictive coding. The list that is used for a transform block is determined by the color component, the prediction mode, and the maximum of the width and height of the block. For block sizes not equal to 2×2, 4×4, or 8×8, the weighting matrices are resampled using nearest neighbor interpolation.

III. Transform Coefficient Coding

Similarly as HEVC, VVC employs context-based adaptive binary arithmetic coding (CABAC) for entropy coding of all low-level syntax elements. Non-binary syntax elements are mapped to binary codewords. The bijective mapping between symbols and codewords, for which typically simple structured codes are used, is called *binarization*. The binary symbols, also called *bins*, of both binary syntax elements and codewords for non-binary data are coded using binary arithmetic coding. The core coding engine, which is further discussed in Section IV, supports two operating modes: A *regular mode*, in which the bins are coded with adaptive probability models, and a less complex *bypass mode* that uses fixed probabilities of $1/2$. The adaptive probability models are also called *contexts* and the assignment of probability models to individual bins is referred to as *context modeling*. Note that both the binarization and the context modeling used have a significant impact on coding efficiency. The required encoder and decoder complexities primarily increase with the number of context-coded bins (i. e., bins coded in regular mode). But they are also affected by other aspects such as the degree of dependencies between successive bins, the complexity of the context modeling used, or the frequency with which a switching between the regular and bypass modes of the arithmetic coding engine occurs.

The entropy coding of quantization indexes for transform blocks is commonly referred to as *transform coefficient coding*. Since, at typical video bit rates, transform coefficient levels consume the major part of the total bit rate, it is important to find a reasonable trade-off between coding efficiency and implementation complexity. The basic concept of the transform coefficient coding in VVC is similar to the coefficient coding specified in HEVC [15]:

1) A coded block flag (CBF) indicates whether a transform block includes any nonzero levels;
2) For blocks with CBF equal to 1, the x and y coordinate of the last nonzero level in forward scan order is transmitted;
3) Starting from the indicated last position, the levels are transmitted in reverse scan order, organized into so-called coefficient groups (CGs). The bins for a CG are coded in multiple passes, where all bypass-coded bins are grouped together in order to enable efficient implementations.

Since VVC supports a larger range of transform sizes than HEVC, some aspects of the transform coefficient coding were generalized. In contrast to HEVC, the scan order does not depend on the intra prediction mode as such a mode-dependent scan was found to provide only negligible improvements and would unnecessarily complicate the design. Moreover, the context modeling for the bins representing levels is independent of the block size; there are no exceptions for certain block shapes. But instead, the context dependency restrictions found in HEVC are relaxed and local statistical dependencies between levels are utilized for increasing coding efficiency. For enabling the exploitation of certain TCQ properties, the binarization for levels includes a parity bin and all context-coded bins of a CG are coded in a single pass. VVC uses a transform block based restriction on the number of context-coded bins to keep a similar worst-case complexity as HEVC.

TABLE II
Coefficient group sizes for $W\times H$ transform blocks

	width W						
height H	1	2	4	8	16	32	64
1	–	–	–	–	16×1	16×1	16×1
2	–	2×2	2×2	8×2	8×2	8×2	8×2
4	–	2×2	4×4	4×4	4×4	4×4	4×4
8	–	2×8	4×4	4×4	4×4	4×4	4×4
16	1×16	2×8	4×4	4×4	4×4	4×4	4×4
32	1×16	2×8	4×4	4×4	4×4	4×4	4×4
64	1×16	2×8	4×4	4×4	4×4	4×4	4×4

A. Coded Block Flag

The coded block flag (CBF) is coded in the regular mode of the coding engine. In total, 9 contexts are used (4 for luma, 2 for Cb, and 3 for Cr). One context per component is reserved for blocks coded in BDPCM mode (a special variant of the transform skip mode, see [8]). For luma, two contexts are used only for transform blocks coded in the intra sub-partitioning mode (see [31]); here, the chosen context depends on the CBF of the preceding luma transform block inside the same coding unit. In order to exploit statistical dependencies between the CBFs of the chroma components, the context for Cr blocks not coded in BDPCM mode is selected depending on the CBF of the co-located Cb block.

B. Coefficient Groups and Scan Order

The transform coefficient levels $\{q\}$ of a $W\times H$ transform block are arranged in a $W\times H$ matrix. For enabling a harmonized processing across all block sizes (see also [15]), but also for increasing coding efficiency for transform blocks, in which the signal energy is concentrated into transform coefficients that correspond to low horizontal or low vertical frequencies, transform blocks are partitioned into *coefficient groups* (CGs). As further detailed in Section III-D, the levels for each CG are coded in a unified manner using multiple scan passes. Since VVC also supports block sizes with widths and heights less than 4, the shape of CGs depends on the transform block size as shown in Table II. For transform blocks with at least 16 coefficients, the CGs always include 16 levels; for smaller blocks, CGs of 2×2 levels are used.

The coding order of CGs is given by the reverse diagonal scan illustrated in Fig. 3. Independent of the CG size, the CG diagonals are processed from the bottom right to the top left of a transform block, where each diagonal is scanned in down left direction. For limiting the worst-case decoder complexity, high-frequency coefficients of large transforms are forced to be equal to zero [6]. Nonzero quantization indexes can only

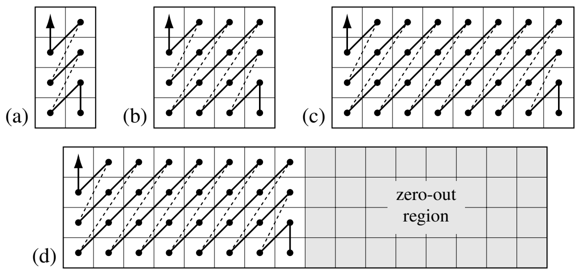

Fig. 3. Illustration of the reverse diagonal scan: Coding order of CGs in (a) 8×16 blocks, (b) 16×16 blocks, (c) 32×16 blocks, and (d) 64×16 blocks. The scan shown in (b) also illustrates the coding order of levels in 4×4 CGs.

TABLE III
BINARIZATION FOR COORDINATES OF LAST COEFFICIENT POSITION

coordinate	prefix (TU)	suffix (FL)	
0	(0)	–	← 1
1	1 (0)	–	← 2
2	1 1 0	–	
3	1 1 1 (0)	–	← 4
4 – 5	1 1 1 1 0	x_0	
6 – 7	1 1 1 1 1 (0)	x_0	← 8
8 – 11	1 1 1 1 1 1 0	x_0x_1	
12 – 15	1 1 1 1 1 1 1 (0)	x_0x_1	← 16
16 – 23	1 1 1 1 1 1 1 1 0	$x_0x_1x_2$	
24 – 31	1 1 1 1 1 1 1 1 1	$x_0x_1x_2$	← 32

be present in a $\max(W, 32) \times \max(H, 32)$ region at the top-left of a transform block[2]. Hence, CGs outside this region are not coded and thus excluded from the scan as is illustrated in Fig. 3d. The coding order of levels inside CGs is specified by the same reverse diagonal scan.

C. Last Significant Coefficient Position

Similar as in HEVC, the explicit coding of zero quantization indexes for coefficients related to high-frequency components is eliminated by transmitting the position of the *last nonzero level* in forward scan order (which is the first nonzero level in coding order). This does not only increase coding efficiency, but also reduces the number of context-coded bins.

The x and y coordinates corresponding to the column and row number, respectively, in the matrix of coefficient levels are transmitted independently of each other. As shown in Table III, each component is represented by a combination of a prefix codeword and a (possibly empty) suffix codeword. The prefix part specifies an interval of values. It is binarized using truncated unary (TU) binarization and the bins are coded in regular mode. The prefix part indicating the last interval of the non-zero-out region of a transform block is truncated. That means, the zero bins in parenthesis shown in Table III are not coded if $\max(W, 32)$, for the x coordinate, or $\max(H, 32)$, for the y coordinate, is equal to the number in the last table column. In particular, the coding of a coordinate is completely skipped if the corresponding block width or height is equal to 1. The suffix part represents the offset inside the interval indicated by the prefix part. It is binarized using fixed length (FL) binarization and coded in bypass mode. Only x and y coordinates with values greater than 3 have a suffix part.

[2]Although VVC specifies larger zero-out regions for transforms other than the DCT-II, see [6], this does not impact the transform coefficient coding; the syntax elements specifying the transform used are coded after the levels and they are conditioned on the presence of nonzero levels in certain regions.

TABLE IV
CONTEXT INDICES FOR PREFIX BINS OF LAST COEFFICIENT COORDINATES

	transform dimension	bin index								
		0	1	2	3	4	5	6	7	8
luma	2	0	1							
	4	0	1	2						
	8	3	3	4	4	5				
	16	6	6	7	7	8	8	9		
	32	10	10	11	11	12	12	13	13	14
	64	15	15	16	16	17	17	18	18	19
chroma	2	20	21							
	4	20	21	22						
	8	20	20	21	21	22				
	16	20	20	20	20	21	21	21		
	32	20	20	20	20	21	21	21	21	22

At the decoder side, the values of the x and y coordinates of the last significant level are derived as follows. Let v_{pre} be the number of bins equal to 1 in the prefix codeword. Then, the number n_{suf} of suffix bins to be decoded is derived by

$$n_{\text{suf}} = \max\left(0, \left\lfloor \frac{v_{\text{pre}}}{2} \right\rfloor - 1\right). \tag{6}$$

With v_{suf} being the value specified by the suffix codeword (in binary representation), the decoded coordinate value *last* is calculated according to

$$\textit{last} = \begin{cases} 2^{n_{\text{suf}}} \cdot (2 + (v_{\text{pre}} \,\&\, 1)) + v_{\text{suf}}, & n_{\text{suf}} > 0 \\ v_{\text{pre}}, & \text{otherwise.} \end{cases} \tag{7}$$

The prefix part for the x coordinate is signaled first followed by that for the y coordinate. For grouping bypass-coded bins, the suffix parts are coded after the prefix codewords. The prefix bins of the x and y coordinates are coded using separate sets of context models. Table IV lists the context offsets that indicate the probability model used inside a set. The model chosen depends on whether a luma or chroma block is coded, the width or height of the transform block, and the bin number inside the prefix codeword. Note that for large transform blocks, where zero-out is present, the transform dimension (and not the dimension of the non-zero-out region) is used to derive the context offset. In total, 46 contexts (40 for luma and 6 for chroma) are used for coding the last coefficient position.

D. Binarization and Coding Order

Starting with the CG containing the last nonzero level (as indicated by the x and y coordinates), the CGs are transmitted in coding order (given by the reverse diagonal scan). The first syntax element coded for a CG is the *sb_coded_flag*. If this flag is equal to 0, it indicates that the CG contains only zero levels. For the first CG (which contains the last nonzero level) and the last CG (which contains the DC level), this flag is not coded, but inferred to be equal to 1. The *sb_coded_flag* is coded in regular mode. The chosen context depends on whether the CG to the right or the CG below contain any nonzero levels, where separate context sets are specified for

TABLE V
BINARIZATION OF THE ABSOLUTE LEVEL VALUES

$\lvert q \rvert$	0	1	2	3	4	5	6	7	8	9	10	11	$\cdots$
sig	0	1	1	1	1	1	1	1	1	1	1	1	$\cdots$
gt1	–	0	1	1	1	1	1	1	1	1	1	1	$\cdots$
par	–	–	0	1	0	1	0	1	0	1	0	1	$\cdots$
gt3	–	–	0	0	1	1	1	1	1	1	1	1	$\cdots$
rem	–	–	–	–	0	0	1	1	2	2	3	3	$\cdots$

luma and chroma. In total, 4 contexts (2 for luma and 2 for chroma) are used. For CGs with *sb_coded_flag* equal to 1, the level values are coded as described in the following.

The binarization of coefficient levels and the coding order of bins were chosen to support an efficient entropy coding for both TCQ and conventional quantization. Due to the different structures of the two scalar quantizers Q_0 and Q_1 used in TCQ (see Fig. 1), the probability that a level is equal to 0 highly depends on the quantizer used. For exploiting this effect in context modeling (Section III-E) and, at the same time, grouping the context- and bypass-coded bins, the binarization includes a dedicated parity flag that is used for determining the TCQ state during entropy coding [32]. By additionally taking into account the number of context-coded bins required for achieving a good coding efficiency [33], [34] as well as the dependencies between successive bins [35], the binarization shown in Table V was chosen. The absolute values $|q|$ of the quantization indexes are mapped to the bins *sig* (significance), *gt1* (greater than 1), *par* (parity), *gt3* (greater than 3), and the non-binary remainder *rem*.

The syntax elements for a CG are coded in multiple passes over the scan positions. Unlike HEVC, where a single syntax element per coefficient is coded per scan pass, VVC codes up to 4 syntax elements per coefficient in a single pass. In the first pass, the context-coded bins *sig*, *gt1*, *par*, and *gt3* are coded in an interleaved manner (i.e., all bins for a scan position are coded before proceeding to the next scan position). Note that the parity bin driving the TCQ state machine is included in the first pass for enabling an efficient coding of the *sig* bin for the TCQ case. For scan positions for which the *sig* bin can be inferred to be equal to 1 (e. g., for the last significant position), it is not signaled. The presence of the *gt1*, *par*, and *gt3* bins is controlled as specified in Table V. The non-binary remainders *rem* are coded in a second scan pass. They are binarized using similar parametric codes as in HEVC and the resulting bins are coded in the bypass mode of the coding engine.

In order to increase the worst-case throughput, the number of context-coded bins that can be coded in the first pass is restricted [33], [35]. For allowing a suitable distribution of context-coded bins across CGs, the limit is specified on a transform block basis. With N being the number of transform coefficients in the non-zero-out region of a transform block, the maximum allowed number of context-coded bins is set to $1.75 \times N$. This would correspond to 28 bins per CG if the bin budget was distributed equally among CGs, which is only slightly higher than the limit specified in HEVC (25 bins). The limit on context-coded bins is enforced as follows. If, at the start of a scan position, the total number of already

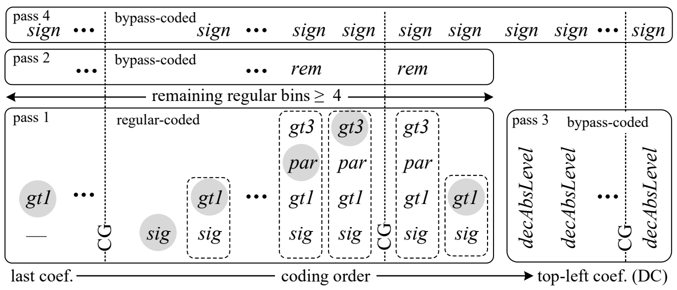

Fig. 4. Illustration of the scan passes (shaded bins have zero values). Before the limit for the number of regular-coded bins is reached, the absolute levels for a CG are coded in passes 1 and 2. After reaching the limit, the remaining absolute values are coded in pass 3 only. The signs are coded in the last pass.

coded *sig*, *gt1*, *par*, and *gt3* bins for the transform block exceeds $1.75 \times N - 4$, i. e., less than 4 bins are remaining in the budget, the first coding pass is terminated. In that case, the absolute values $|q|$ for the remaining scan positions are coded in a third scan pass. They are represented by syntax elements *decAbsLevel*, which are completely coded in bypass mode.

Finally, in the fourth and last pass, the signs for all nonzero levels of a CG are coded in bypass mode. If SDH is enabled and the difference between the scan indexes of the last and first nonzero level inside the CG is greater than 3, the sign for the last nonzero level is not signaled. Fig. 4 illustrated the organization of level data into the different scan passes.

E. Context Modeling

In order to efficiently utilize conditional statistics for arithmetic coding, VVC uses a rather large set of context models for coding the bins *sig*, *gt1*, *par*, and *gt3*. Beside the TCQ state[3], the context modeling also exploits statistical dependencies between spatially neighboring quantization indexes, similar to the approaches described in [36]–[38].

The context for the *sig* bin depends on the associated TCQ state s_k, the diagonal position $d = x + y$ of the coefficient in transform block, and the sum of partially reconstructed absolute levels q^* inside the local template T illustrated in Fig. 5a. The partially reconstructed absolute levels are given by already coded bins for neighboring scan positions and can be calculated according to

$$q^* = \mathit{sig} + \mathit{gt1} + \mathit{par} + 2 \cdot \mathit{gt3}. \tag{8}$$

For luma blocks, the context index $c^{\text{sig}}_{\text{lum}}$ indicating the adaptive probability model used is derived according to

$$c^{\text{sig}}_{\text{lum}} = 12 \cdot \max(0, s_k - 1) + f_{\text{sig}}(T) + \begin{cases} 8, & d < 2, \\ 4, & 2 \le d < 5, \\ 0, & d \ge 5, \end{cases} \tag{9}$$

with

$$f_{\text{sig}}(T) = \min\Big(3, \Big(1 + \sum_T q^*\Big) \gg 1\Big) \tag{10}$$

being a function of the partially reconstructed levels q^* inside the local template T. For chroma blocks, only two classes of

[3]It was observed [32] that the probability distribution for the *sig* bin actually depends on the TCQ state and not only on the quantizer used.

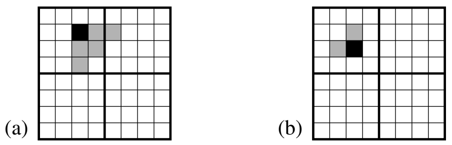

Fig. 5. Locate template T (gray) around a current scan position (black): (a) for transform coefficient coding; (b) for transform skip residual coding.

diagonal positions ($d < 2$ and $d \geq 2$) are used. The context index $c_{\text{chr}}^{\text{sig}}$ is derived by

$$c_{\text{chr}}^{\text{sig}} = 8 \cdot \max(0, s_k - 1) + f_{\text{sig}}(T) + \begin{cases} 4, & d < 2, \\ 0, & d \geq 2. \end{cases} \tag{11}$$

When TCQ is not enabled, the value of the TCQ state s_k is set equal to 0. In total, 60 context model are used for coding the *sig* bin (36 for luma and 24 for chroma).

The probability models chosen for *gt1*, *par*, and *gt3* do not depend on the TCQ state, as it was found to provide only a very minor benefit. A single shared context offset is computed to select the probability model for these syntax elements. They are chosen based on the diagonal position d of the coefficient (4 classes for luma and 2 for chroma) and the sum of the values $\max(0, q^* - 1)$ inside the local template T. With

$$f(T) = \min\Big(4, \sum_T \max(0, q^* - 1)\Big) \tag{12}$$

being another function of the partially reconstructed levels q^* inside the local template T, the context indexes c_{lum} and c_{chr} for luma and chroma blocks, respectively, are given by

$$c_{\text{lum}} = f(T) + \begin{cases} 1, & d \geq 10, \\ 6, & 3 \leq d < 10, \\ 11, & 0 < d < 3, \\ 16, & d = 0, \end{cases} \tag{13}$$

$$c_{\text{chr}} = f(T) + \begin{cases} 1, & d > 0, \\ 6, & d = 0. \end{cases} \tag{14}$$

In addition, for the last coefficient position a separate context (given by $c_{\text{lum}} = 0$ and $c_{\text{chr}} = 0$) is used. For each of the *gt1*, *par*, and *gt3* bins, 32 probability models (21 for luma and 11 for chroma) are used in total.

F. Binarization of Bypass-Coded Level Data

The syntax elements *rem* coded in the second pass represent remainders for absolute levels. They are only transmitted for a scan position if the associated *gt3* bin is equal to 1. With q^* being a partially reconstructed level according to (8), the absolute value $|q|$ of the level is given by

$$|q| = q^* + 2 \cdot \textit{rem}. \tag{15}$$

The remainders *rem* and the syntax elements *decAbsLevel*, which represent absolute levels coded in the third pass, are binarized using a combination of truncated Rice (TR) and Exp-Golomb (EG) codes, similar to remainder values in HEVC. The resulting bins are coded in the bypass mode of the coding engine. Unlike HEVC, the Rice parameter for the TR codes is derived based on the sum of absolute level values $|q|$ in a local template T. The local template T used is the same as the template used for context index derivation in the first coding pass. The Rice parameter m is given by

$$m = \begin{cases} 0, & s_T < 7, \\ 1, & 7 \leq s_T < 14, \\ 2, & 14 \leq s_T < 28, \\ 3, & s_T \geq 28, \end{cases} \quad \text{with } s_T = \sum_T |q| - 5\, z_0, \tag{16}$$

where z_0 is set equal to 4 for coding the remainders *rem*, and it is set equal to 0 for the coding *decAbsLevel*. The reason for this difference is that the values of *decAbsLevel* specify complete absolute levels, while the remainders *rem* represent differences $\textit{rem} = (|q| - q^*)/2$, which have smaller values.

For each Rice parameter *m*, values less than $v_{\max} = 2^m \cdot 6$ are coded using only TR codes of order m (TR$_m$); this corresponds to codes with a unary prefix of length 6. For values greater than or equal to $v_{\max}$, the TR$_m$ codes are concatenated with Exp-Golomb codes of order $m+1$ (EG$_{m+1}$). Table VI shows the binarization for Rice parameters $m = 0$ and $m = 3$ with a concatenation of TR$_m$ and EG$_{m+1}$ codes. Bold bins in the table correspond to the TR$_m$ portion of the binarization. When the combined code length would exceed 32 bins, the binarization is slightly modified [39]. In this case, the length of the Exp-Golomb prefix is limited to 11 bins (see underlined entry for $m = 0$ in Table VI) and the remaining 15 bins of the 32 bit budget are used to represent the suffix part.

For increasing the coding efficiency for completely bypass-coded levels [33], the values of *decAbsLevel* do not represent the absolute level values $|q|$ directly, but are derived as

$$\textit{decAbsLevel} = \begin{cases} \text{pos}_0, & |q| = 0, \\ |q| - 1, & 0 < |q| \leq \text{pos}_0, \\ |q|, & |q| > \text{pos}_0. \end{cases} \tag{17}$$

TABLE VI
BINARIZATION FOR SYNTAX ELEMENTS *rem* AND *decAbsLevel*

value range	prefix (unary)	suffix (fixed length)
Rice parameter $m = 0$		
0	**0**	
1	**10**	
2	**110**	
3	**1110**	
4	**11110**	
5	**111110**	
$[6; 7]$	**111111**0	x
$[8; 11]$	**111111**10	xx
...	...	...
$[2052; 4099]$	**111111**11111111110	xxxx xxxx xxx
$[4100; 32768]$	**111111**$\underline{11111111111}$	xxxx xxxx xxxx xxx
Rice parameter $m = 3$		
$[0; 7]$	**0**	**xxx**
$[8; 15]$	**10**	**xxx**
$[16; 23]$	**110**	**xxx**
$[24; 31]$	**1110**	**xxx**
$[32; 39]$	**11110**	**xxx**
$[40; 47]$	**111110**	**xxx**
$[48; 63]$	**111111**0	xxxx
$[64; 95]$	**111111**10	xxxx x
...	...	...
$[8224; 16415]$	**111111**1111111110	xxxx xxxx xxxxx
$[16416; 32768]$	**111111**11111111110	xxxx xxxx xxxxx x

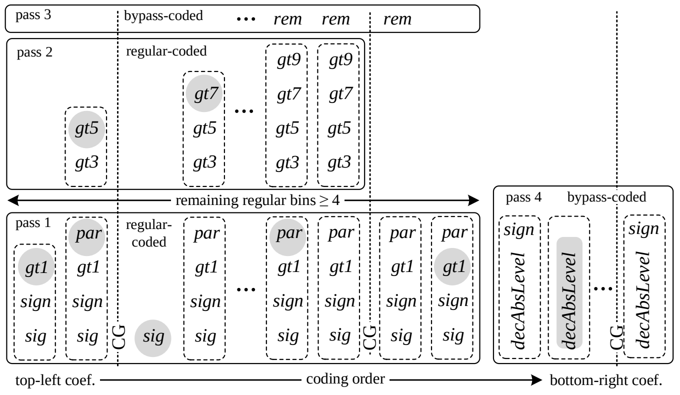

Fig. 6. Illustration of coding passes in transform skip residual coding (shaded elements have zero values).

These values are coded using the same binarization as for the remainders *rem*. Note that the parameter pos_0 basically specifies the position of the codeword for $|q| = 0$ in a reordered codeword table. It is derived based on the Rice parameter m and the TCQ state s_k according to

$$\text{pos}_0 = 2^m \cdot \begin{cases} 1, & s_k < 2, \\ 2, & s_k \geq 2. \end{cases} \tag{18}$$

G. Transform Skip Residual Coding

In addition to the regular residual coding (RRC) for transform coefficients described above, VVC also includes a dedicated entropy coding for quantization indexes in transform skip mode, which is referred to as *transform skip residual coding* (TSRC). It was mainly designed for improving coding efficiency for screen content and can be enabled on a slice level. When enabled, the TSRC scheme is used for coding quantization indexes of transform skip blocks; when not enabled, the quantization indexes of transform skip blocks are coded with the regular residual coding.

In contrast to the regular residual coding, the position of the last nonzero level is not transmitted and the levels are coded in forward scan order, i. e., starting from the top-left coefficient and proceeding to the bottom-right coefficient. Similar to RRC, the syntax elements for a CG are coded in multiple passes over the scan positions, and the same limit for the number of context-coded bins is applied. As long as this limit is not reached, the levels are coded using three passes, as shown in Fig. 6. In the first pass, the bins of *sig*, *sign*, *gt1*, and *par* are interleaved and context-coded using adaptive probability models. A local template, as shown in Fig. 5b, is also applied in TSRC for deriving the context indexes, but it only includes two neighboring coefficient positions. Since, in transform skip blocks, successive signs have often similar values, the sign flags are included into the first pass and are coded in the regular mode of the coding engine. If the limit of regular-coded bins is still not reached after the first pass for a CG, up to four greater-than-x flags (*gt3*, *gt5*, *gt7*, and *gt9*) per coefficient are coded in a second pass. These bins are also context-coded. Finally, in a third pass, the remainders for absolute levels (*rem*) are coded in bypass mode. Note that the remainders can have different meanings, depending on whether the bin limit was reached for a scan position during the second pass (and, thus, no *gt3* bin could be coded). For all scan positions for which no data were transmitted in the first pass, the complete absolute values (*decAbsLevel*) as well as the associated sign flags are coded in bypass mode in a fourth pass. The Rice parameter m for both *rem* and *decAbsLevel* is always set equal to 1. For more details on the design of TSRC, the reader is referred to [8].

IV. Binary Arithmetic Coding

Context-based adaptive binary arithmetic coding (CABAC) [40] was originally introduced in AVC, as one of two supported entropy coding methods. Due its superior coding efficiency compared to conventional variable-length coding, it is the only entropy coding supported in both HEVC and VVC. But while AVC and HEVC share the same core coding engine, VVC introduces a new engine for the regular coding mode that is designed to be more flexible and efficient.

In binary arithmetic coding, the coding engine consists of two elements: Probability estimation and codeword mapping. The purpose of probability estimation is to determine the likelihood of the next binary symbol having the value 1. This estimation is based on the history of symbol values coded using the same context and typically uses an exponential decay window [41]. Given a sequence of binary symbols $x(t)$, with $t \in \{1, \cdots, N\}$, the estimated probability $p(t+1)$ of $x(t+1)$ being equal to 1 is given by

$$p(t+1) = p(1) + \sum_{k=1}^{t} \alpha \cdot (1-\alpha)^{t-k} \cdot (x(t) - p(1)), \tag{19}$$

where $p(1)$ is an initial probability estimate and α is a base determining the rate of adaptation. Alternatively, this can be expressed in a recursive manner as

$$p(t+1) = p(t) \cdot (1-\alpha) + x(t) \cdot \alpha. \tag{20}$$

The engine of AVC and HEVC implements such an exponential smoothing estimator using a single finite state machine with 128 states. VVC also uses such an estimator, but with some key differences:

- VVC maintains two estimates for each context, where each estimate uses it own base α. The probability that is actually used for coding is the average of the two estimates. The reason for using multiple estimates is to improve compression performance;
- VVC defines a different pair of bases for each context to improve compression performance;
- VVC does not use a state machine but arithmetically derives the probability estimates using the recursive function described above.

More details on the rationale for using two estimates and per-context customized bases are provided in Section IV-A.

In VVC, the initial estimate $p(1)$ is derived for each context using a linear function of the quantization parameter QP, as is also done in AVC/HEVC. The main difference lies in the fact that, in VVC, the so derived value represents an actual probability (linear space), whereas in AVC/HEVC, it represents a state of the state machine (logarithmic space).

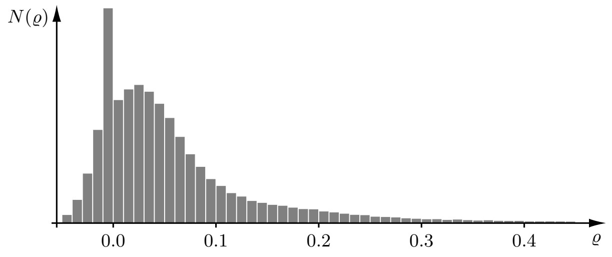

Fig. 7. Histogram for auto-correlation coefficients ϱ.

For codeword mapping, a current interval is split into two subintervals, each corresponding to one of the possible values of a binary symbol. The range of each subinterval is obtained by multiplying the range r of the current interval with the corresponding probability estimate. In AVC/HEVC, the multiplication is approximated using a lookup table, which determines the range r_{LPS} of the subinterval associated with the least probable symbol (LPS). In VVC, a direct multiplication is used instead while using the same LPS convention. Once r_{LPS} is determined, the AVC/HEVC and VVC engines operate in identical manners.

A. Multi-Hypothesis Probability Estimation

Consider a binary source $x(t)$ and let p_0 be the marginal probability of a symbol being equal to 1. When using the initial estimate $p(1) = p_0$, the expected value of the exponential smoothing estimator is given by

$$E[\,p(t)\,] = p_0, \tag{21}$$

which means that the estimator is unbiased. Assuming that the source is uncorrelated, i. e.,

$$E[\,(x(t_i) - p_0)(x(t_k) - p_0)\,] = 0, \quad \text{if } t_i \neq t_k, \tag{22}$$

the variance of the prediction error is

$$E\big[\,(p(t) - p_0)^2\,\big] = \alpha(p_0 - p_0^2)/(2 - \alpha). \tag{23}$$

This implies that the optimal value of α should be 0. However, this is not observed in practice, where larger values of α are found to be optimal. The assumption that the source is uncorrelated is therefore incorrect. Fig. 7 shows the distribution of first-order auto-correlation coefficients ϱ for data collected from a set of VVC bitstreams, where the correlation coefficients were estimated on chunks of 4096 symbols for each context. The range $[-0.05, 0.45]$ and distribution are clearly biased towards positive auto-correlation coefficients.

The combination of multiple estimators using averaging was proposed in [42]. It is given by

$$\begin{aligned} p_0(t+1) &= p_0(t) \cdot (1 - \alpha_0) + x(t) \cdot \alpha_0, \\ p_1(t+1) &= p_1(t) \cdot (1 - \alpha_1) + x(t) \cdot \alpha_1, \\ p(t+1) &= (\,p_0(t+1) + p_1(t+1)\,)\,/\,2. \end{aligned} \tag{24}$$

For this two-parameter estimator, the relationship between the auto-correlation coefficient ϱ of the source and the estimation error was investigated in [43]. In particular, considering an first-order auto-regressive source model,

$$E[\,(x(t_i) - p_0)(x(t_k) - p_0)\,] = \varrho^{|t_i - t_k|}\sigma^2, \tag{25}$$

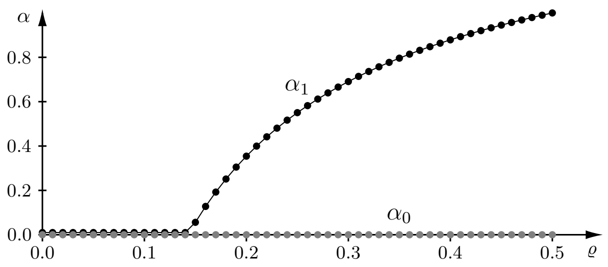

Fig. 8. Optimal values for the parameters α_0 and α_1 as function of the correlation coefficient ϱ.

with $0 < \varrho < 1$, optimal values for α_0 and α_1 were derived as a function of the correlation coefficient ϱ. As shown in Fig. 8, the first parameter α_0 should be equal to 0, and the second parameter α_1 should be chosen as a function of ϱ. Using these optimal parameters, it was further shown in [43] that the two-parameter estimator outperforms the traditional one-parameter estimator for a wide range of correlation coeffcients ϱ.

The above assumes that the initial probability estimate is set equal to p_0. In practice, this is not achievable as p_0 may depend on the actual content of a slice. α_0 should therefore be set to a value larger than 0 such as to gradually disregard the initial estimate. The larger the value of α_0, the less impact has the initial estimate. In VVC, the parameters α_0 and α_1 were selected for each context using a training algorithm that jointly optimizes these parameters and the initial probability estimates [44].

B. Implementation Considerations

Arithmetic coding is an inherently serial process: Each symbol must be processed in sequence. Throughput, measured in the number of symbols processed per second, is a key complexity metric to be considered in the design of a coding engine. Another key complexity metric is the memory requirement. A combination of hardware and software considerations have been used to design the VVC coding engine.

For probability estimation, to simplify implementation and avoid multiplications, the bases α are limited to negative integer powers of 2, i. e., $\alpha = 2^{-\beta}$ with $\beta \in \mathbb{N}^+$. This enables implementations with bit shifting operations [45],

$$q(t+1) = q(t) - (q(t) \gg \beta) + x(t) \cdot ((2^b - 1) \gg \beta), \tag{26}$$

where $q(t)$ is an integer representation of $p(t)$ with b bits. The relationship between $p(t)$ and $q(t)$ is given by

$$p(t) = q(t) \cdot 2^{-b} + 2^{-b-1}. \tag{27}$$

The value of b for each estimator is selected based on coding efficiency and memory considerations. Memory requirements are driven by the product of two numbers: The number of contexts n and the number of bits m required to capture the state for each context. n depends on context modeling, as discussed in Section III, while m is equal to the sum $b_1 + b_2$ of the number of bits used for each estimator. As VVC uses two estimators with different adaptation rates, a smaller number of bits is typically required for the estimator with the faster adaptation rate. Hence, $b_2 = 10$ and $b_1 = 14$ bits are used for

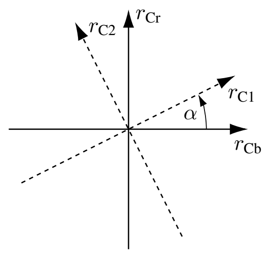

Fig. 9. Rotational transform of vectors $(r_{\text{Cb}}, r_{\text{Cr}})$ to $(r_{\text{C1}}, r_{\text{C2}})$ by an angle α.

TABLE VII
SUPPORTED VALUES OF α AND ASSOCIATED WEIGHTS OF ROTATION MATRIX APPLIED DURING JCCR PROCESSING IN A VVC DECODER

angular mode m	–3	–2	–1	1	2	3
value of α	–63.4°	–45°	–26.6°	26.6°	45°	63.4°
weights of T_α^{-1}	$\begin{matrix} \frac{1}{2} & 1 \\ -1 & \frac{1}{2} \end{matrix}$	$\begin{matrix} 1 & 1 \\ -1 & 1 \end{matrix}$	$\begin{matrix} 1 & \frac{1}{2} \\ -\frac{1}{2} & 1 \end{matrix}$	$\begin{matrix} 1 & -\frac{1}{2} \\ \frac{1}{2} & 1 \end{matrix}$	$\begin{matrix} 1 & -1 \\ 1 & 1 \end{matrix}$	$\begin{matrix} \frac{1}{2} & -1 \\ 1 & \frac{1}{2} \end{matrix}$

the faster and slower estimator, respectively, yielding a total of $m = 24$ bits per context. This amount is significantly higher than for HEVC (7 bits) but nevertheless remains reasonable.

Multiplications are thus avoided for probability estimation but not for subinterval range computation. While the bit width of the multiplier has typically no impact in software (latency and throughput are the same for 8-, 16-, and 32-bit multiplications), it does matter in hardware, where smaller multipliers are preferred. The size of the multiplier in VVC is thus limited to 5 by 4 bits, where 5 is the number of bits representing the probability estimate and 4 the number of bits representing the range of the current interval. Thus, r_{LPS} is computed as follows,

$$q(t) = q_1(t) + 16 \cdot q_2(t), \tag{28}$$

$$q_5(t) = (q(t) \gg 10) \oplus (63 \cdot (q(t) \gg 14)), \tag{29}$$

$$r_{\text{LPS}}(t) = ((q_5(t) \cdot (r(t-1) \gg 5)) \gg 1) + 4, \tag{30}$$

where $\oplus$ specifies the bit-wise "exclusive or" operator.

During the development of VVC, throughput of the coding engine was measured for optimized software implementations (see experiment 2 in [46]). In that experiment, the throughput of the VVC engine was determined to be about 7% lower than that of the AVC/HEVC engine (128.5 million symbols per second versus 137.8 million symbols per second).

V. JOINT CODING OF CHROMA RESIDUALS

The previous sections focused on the actual quantization and entropy coding of individual blocks of transform coefficients in the VVC standard. An efficient joint representation of multi-component residual block signals, however, was also addressed during the development of VVC. In addition to the cross-component linear model (CCLM) prediction of chroma samples from collocated luma samples [47], VVC provides a means for the joint coding of chroma residuals (JCCR), which is described in the following.

Digital images and video pictures are generally composed of multiple color components (for example, *red*, *green*, *blue* in RGB color formats and Y, Cb, Cr in the YCbCr color format). In natural pictures acquired via image sensors, a signal correlation can be observed between these color components, causing some redundancy to remain in the quantized residuals. JCCR [48], [49] exploits the correlation between chroma components (particularly, the Cb and Cr components in YCbCr coding) by allowing an encoder to transmit, on a transform unit (TU) basis, only one instead of two quantized residual signals, along with compact correlation angle and sign information. In the decoder, the transmitted *downmixed* joint residual block signal is then *upmixed* to the original color components, scaled according to the angle and sign information.

The following two subsections describe the JCCR mode in VVC in more detail. For further information on the fundamental concept behind the JCCR tool, the inter-component transformation (ICT), the reader is referred to [50].

A. Forward and Inverse Rotational Transform

The JCCR processing can be regarded as a switchable *inter*-component rotational transform applied in addition to conventional *intra*-component spatial transforms like the DCT-II [50], with the purpose of achieving increased compaction of residual energy into a single component on a block basis. As illustrated in Fig. 9, this rotational transform on two residual blocks r_{Cb} and r_{Cr} is controlled by an angle α. Conceptually, the forward and inverse transforms are given by

$$\begin{pmatrix} r_{\text{C1}} \\ r_{\text{C2}} \end{pmatrix} = T_\alpha \cdot \begin{pmatrix} r_{\text{Cb}} \\ r_{\text{Cr}} \end{pmatrix}, \quad \begin{pmatrix} r'_{\text{Cb}} \\ r'_{\text{Cr}} \end{pmatrix} = T_\alpha^{-1} \cdot \begin{pmatrix} r'_{\text{C1}} \\ r'_{\text{C2}} \end{pmatrix}, \tag{31}$$

with the forward transform matrix

$$T_\alpha = \beta_\alpha \cdot \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}. \tag{32}$$

In the JCCR modes supported in VVC, the samples of the second component r'_{C2} are enforced to be equal to zero. Hence, at the decoder side, both color components r'_{Cb} and r'_{Cr} are reconstructed from the transmitted *downmix* block signal r'_{C1}. At the encoder side, the rotation angle α can be selected from a predefined set of values. Typically, an encoder would select the angle α_{opt} that yields the lowest rate-distortion cost (when considering the reconstruction of the Cb and Cr components). Naturally, an encoder can also disable JCCR on a TU basis (for example, if it would decrease coding efficiency); then, the residual blocks for Cb and Cr are coded separately.

To enable efficient hardware and software implementations with integer arithmetic, the scaling parameter β_α is chosen as

$$\beta_\alpha = \max\big(\,|\cos(\alpha)|, |\sin(\alpha)|\,\big)\,. \tag{33}$$

VVC supports, in total, 6 different rotation angles α, which can be indicated by an angular mode m. The rotation angles α and the corresponding weights of T_α^{-1} for all supported modes m are shown in Table VII. Note that multiplications by $^1\!/_2$ can be realized efficiently using bit shifts to the right. Thus, JCCR *upmixing*, which follows the inverse spatial transforms in the decoder, is given by

$$r'_{\text{Cb}} = r'_{\text{C1}}, \quad r'_{\text{Cr}} = (c_{\text{sign}} \cdot r'_{\text{C1}}) \gg 1, \quad \text{for } |m| = 1, \tag{34}$$

$$r'_{\text{Cb}} = r'_{\text{C1}}, \quad r'_{\text{Cr}} = c_{\text{sign}} \cdot r'_{\text{C1}}, \quad \text{for } |m| = 2, \tag{35}$$

$$r'_{\text{Cr}} = r'_{\text{C1}}, \quad r'_{\text{Cb}} = (c_{\text{sign}} \cdot r'_{\text{C1}}) \gg 1, \quad \text{for } |m| = 3, \tag{36}$$

where $c_{\text{sign}} \in \{-1, 1\}$ represents the sign of the mode m or, equivalently, the sign of the rotation angle α.

B. Signaling of JCCR Usage and Rotation Parameters

The general usage of JCCR can be enabled on a picture level. When enabled, a flag *tu_joint_cbcr_residual_flag* is transmitted for every TU for which either or both chroma coded block flags, CBF_{Cb} and CBF_{Cr}, are equal to 1. If the *tu_joint_cbcr_residual_flag* is equal to 0, JCCR is not used for the TU and the chroma residual blocks are reconstructed in a conventional manner. Otherwise, if the flag is equal to 1, the absolute value of the JCCR mode m is derived by

$$|m| = \begin{cases} 1, & \text{if } \text{CBF}_{\text{Cb}} = 1 \text{ and } \text{CBF}_{\text{Cr}} = 0, \\ 2, & \text{if } \text{CBF}_{\text{Cb}} = 1 \text{ and } \text{CBF}_{\text{Cr}} = 1, \\ 3, & \text{if } \text{CBF}_{\text{Cb}} = 0 \text{ and } \text{CBF}_{\text{Cr}} = 1. \end{cases} \tag{37}$$

The value of c_{sign}, allowing the decoder to distinguish between $m < 0$ and $m > 0$, is conveyed via the picture header syntax element *ph_joint_cbcr_sign_flag*. This flag is transmitted on a picture level, since it was observed that its optimal value usually varies very little within a video frame. Note that, when JCCR is enabled for a TU and both chroma CBFs are equal to 1, no quantization indexes are sent for the second chroma component. Hence, in all cases, either the Cb residual r_{Cb} or the Cr residual r_{Cr} is replaced by the downmix component r_{C1}.

The support of JCCR does not require any modifications of the transform coding for residual blocks. But as noted in Section II-F, a separate look-up table can be specified for deriving the QP for JCCR modes with $|m| = 2$. For $|m| = 1$ and $|m| = 3$, the QPs for the Cb and Cr components, respectively, are used.

VI. Experimental Results

In the following, we provide experimental results evaluating the coding efficiency impact of the quantization and entropy coding modifications in VVC relative to HEVC. All results were obtained by running coding experiments according to the JVET Common Test Conditions (CTC) [51]. In addition to the test sequences specified in JVET's CTC, we also generated results for two other well-known test sets: The UHD-1 test set of the EBU [52] with 12 sequences in 2160p resolution and the 5 publicly available 1080p sequences of the SVT [53] test set. All sequences are given in the YCbCr 4:2:0 format with 8 or 10 bits per sample and frame rates of 24 Hz to 60 Hz.

Since a video coding standard like VVC specifies a combination of multiple coding tools and design concepts, it is difficult to assess the benefit of individual aspects. For example, the design of many tools is affected by the block partitioning, improvements in intra- and inter-picture prediction influence the effectiveness of all transform coding tools, and the non-normative encoding algorithm has a significant impact on all coding efficiency comparisons. In our coding experiments, we ran simulations with the VTM-9.1 reference software [28] and compared the following five versions:

1) VTM-9.1 configured according to CTC (enabling all tools that contribute to coding efficiency);
2) Version 1 with disabling JCCR;
3) Version 2 with additionally disabling TCQ but enabling SDH (already supported in HEVC) instead;
4) Version 3 with additionally replacing the arithmetic coder of VVC with that of AVC/HEVC (the same initialization tables are used, but with a mapping to initial states);
5) Version 4 with additionally replacing the VVC with the HEVC coefficient coding (for supporting all block shapes, the definition of CGs and the scan is not modified).

By comparing bitstreams generated with versions 1 and 5, we can estimate the coding efficiency benefit of the newly added features for quantization and entropy coding. The contribution of individual tools is assessed by comparing two successive versions in the list above. As measure for coding efficiency differences, we use the Bjøntegaard delta (BD) rate [54] with base QP values of 37, 32, 27, and 22, as specified in the JVET CTC. Note that negative numbers indicate corresponding average savings in bit rate for the same quality, measured as peak signal-to-noise ratio (PSNR).

The BD rates are measured for the three test scenarios *all intra* (AI), *random access* (RA), and *low delay* (LD) specified in CTC. The average results for the CTC sequence classes (A1 to F) and the two additional test sets (EBU and SVT) are summarized in Table VIII. For each scenario, the table lists two BD rate averages: An average over the sequences of classes A1, A2, B, C, and E as defined in JVET's CTC and an average over all tested HD and UHD sequences. It also reports increases in encoder and decoder run times (measured via geometric averages, see [51]), which give an indication of the impact on encoder and decoder complexity, respectively.

The simulation results indicate that the improvements of quantization and entropy coding in VVC relative to HEVC yield bit-rate savings of roughly 4% at reasonably small increases of encoding and decoding times, where somewhat higher gains (but also higher encoding times) are observed for intra-only coding. The contributions of the individual tools lie in a range of about 0.5–2%. The larger improvements for class F, which comprises sequences with screen content, can be attributed to the newly included *transform skip residual coding* (see also [8]). The decreased decoding times for enabling TCQ are caused by a slight shift of the quantizer's operating point towards lower bit rates. For the results shown in Table VIII, TCQ was compared to SDH, since the latter is already included in HEVC. When one compares TCQ to conventional quantization with URQs, the bit-rate savings increase by about 0.5–0.7%, which represents the gain of SDH. Due to their VQ properties, both quantization tools show larger gains for higher video qualities. In contrast to that, JCCR is more effective for lower bit rates. JCCR also yields larger benefits for non-4:2:0 color sampling formats, as these include more samples

TABLE VIII
AVERAGE BD RATES [%] FOR THE IMPROVEMENTS OF QUANTIZATION AND ENTROPY CODING IN VVC RELATIVE TO HEVC

		all improvements			**coefficient coding**			**arithmetic coding**			**TCQ** (vs SDH)			**JCCR**		
		Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
all intra	EBU (2160p)	−5.3	−3.2	−5.1	−1.2	−0.2	−0.6	−1.4	−1.3	−1.2	−2.6	0.0	0.9	−0.3	−1.6	−4.2
	SVT (1080p)	−4.5	−2.4	−4.5	−0.7	−0.5	−0.8	−1.1	−1.0	−0.9	−2.0	−2.8	−2.1	−0.8	2.0	−0.8
	class A1	−4.9	−2.4	−0.3	−1.2	−0.5	0.0	−1.3	−1.1	−1.1	−2.0	−1.5	−0.8	−0.6	0.8	1.6
	class A2	−6.1	−4.3	−3.0	−1.2	−0.9	−0.7	−1.3	−1.4	−1.2	−2.6	−1.0	−1.5	−1.2	−1.2	0.4
	class B	−3.9	−2.7	−3.4	−1.0	−0.3	−0.6	−1.1	−1.1	−1.1	−1.5	−1.0	−0.4	−0.3	−0.3	−1.3
	class C	−4.0	−4.6	−5.0	−1.5	−0.7	−0.6	−1.0	−1.2	−1.1	−1.1	−1.4	−1.2	−0.5	−1.5	−2.2
	class D	−3.9	−4.3	−4.5	−1.4	−0.5	−0.7	−1.0	−1.1	−0.9	−1.2	−1.1	−1.1	−0.5	−1.5	−1.9
	class E	−4.2	−3.1	−3.0	−1.6	−0.8	−0.5	−1.0	−1.3	−1.3	−1.2	−0.8	−0.6	−0.4	−0.4	−0.6
	class F	−7.3	−7.5	−8.1	−5.1	−3.5	−3.9	−1.2	−1.2	−1.1	−0.5	−1.2	−1.0	−0.7	−1.8	−2.3
	avg. CTC	**−4.5**	**−3.4**	**−3.1**	**−1.3**	**−0.6**	**−0.5**	**−1.1**	**−1.2**	**−1.2**	**−1.6**	**−1.1**	**−0.9**	**−0.6**	**−0.5**	**−0.6**
	avg. HD/UHD	**−4.9**	**−3.0**	**−3.9**	**−1.1**	**−0.4**	**−0.6**	**−1.3**	**−1.2**	**−1.1**	**−2.3**	**−1.0**	**−0.3**	**−0.5**	**−0.4**	**−2.0**
	(enc./dec. time)	(121% / 98%)			(105% / 102%)			(108% / 101%)			(105% / 95%)			(102% / 100%)		
random access	EBU (2160p)	−3.9	−1.6	−4.3	−0.9	−0.3	−1.7	−1.0	−0.6	0.2	−1.7	0.1	0.1	−0.3	−0.8	−2.9
	SVT (1080p)	−4.6	−1.0	−4.8	−0.7	−0.2	−1.0	−1.2	−0.8	−0.4	−2.2	−1.7	−1.5	−0.7	1.7	−1.9
	class A1	−3.9	−1.8	1.5	−0.8	−0.9	−0.2	−1.0	−0.9	−0.9	−1.4	−0.6	−0.8	−0.8	0.7	1.7
	class A2	−4.0	−3.5	−1.8	−0.7	−0.5	−0.7	−1.0	−0.7	−0.6	−1.6	−0.5	−0.8	−0.8	−1.8	0.2
	class B	−3.8	−2.4	−2.1	−0.8	−0.6	−0.9	−1.0	−0.9	−0.5	−1.8	−0.4	0.1	−0.3	−0.5	−0.8
	class C	−3.5	−2.9	−3.2	−1.2	−0.5	−0.8	−0.8	−0.8	−0.7	−1.1	−1.8	−1.0	−0.5	0.1	−0.6
	class D	−3.4	−3.4	−2.7	−1.3	−0.2	0.2	−0.7	−0.6	−0.8	−1.2	−1.8	−1.5	−0.4	−0.9	−0.7
	class F	−6.4	−6.1	−6.7	−4.1	−3.1	−3.4	−0.9	−1.0	−0.9	−0.6	−1.2	−1.3	−1.0	−0.9	−1.3
	avg. CTC	**−3.8**	**−2.6**	**−1.6**	**−0.9**	**−0.6**	**−0.7**	**−0.9**	**−0.8**	**−0.6**	**−1.5**	**−0.8**	**−0.2**	**−0.5**	**−0.4**	**−0.1**
	avg. HD/UHD	**−4.0**	**−1.9**	**−3.1**	**−0.8**	**−0.4**	**−1.2**	**−1.0**	**−0.7**	**−0.2**	**−1.8**	**−0.5**	**−0.2**	**−0.5**	**−0.2**	**−1.5**
	(enc./dec. time)	(110% / 99%)			(104% / 101%)			(103% / 100%)			(101% / 98%)			(101% / 100%)		
low delay	SVT (1080p)	−4.6	−2.3	−3.6	−0.3	−0.3	−1.1	−1.2	−0.6	0.3	−3.0	−1.8	0.0	−0.3	0.3	−2.9
	class B	−3.7	−2.9	−2.7	−0.6	−0.6	−1.4	−0.9	−0.3	0.2	−2.1	−0.1	0.6	−0.1	−2.0	−2.0
	class C	−3.2	−4.8	−6.0	−1.0	−0.5	−2.0	−0.8	−1.0	−0.1	−1.3	−1.2	−0.9	−0.2	−2.2	−3.0
	class D	−3.3	−6.4	−5.8	−1.1	−0.4	−1.6	−0.7	0.0	0.6	−1.4	−2.1	−1.8	−0.1	−3.8	−3.1
	class E	−2.6	−1.6	0.4	−0.8	0.2	−0.1	−0.7	−2.3	0.9	−1.1	1.8	1.5	0.0	−1.2	−1.9
	class F	−5.5	−6.1	−7.4	−3.6	−2.5	−2.8	−0.9	−1.1	−1.3	−0.7	−0.9	−0.8	−0.3	−1.6	−2.6
	avg. CTC	**−3.3**	**−3.2**	**−3.0**	**−0.8**	**−0.3**	**−1.3**	**−0.8**	**−1.0**	**0.3**	**−1.6**	**0.0**	**0.3**	**−0.1**	**−1.9**	**−2.3**
	avg. HD	**−4.2**	**−2.6**	**−3.2**	**−0.5**	**−0.4**	**−1.3**	**−1.1**	**−0.4**	**0.2**	**−2.5**	**−0.9**	**0.3**	**−0.2**	**−0.9**	**−2.4**
	(enc./dec. time)	(111% / 97%)			(105% / 101%)			(102% / 99%)			(102% / 98%)			(101% / 100%)		

TABLE IX
AVERAGE BD RATES [%] FOR ENABLING JCCR IN 4:4:4 SEQUENCES

	YCbCr 4:4:4			**RGB 4:4:4**		
	Y	Cb	Cr	G	B	R
all intra	−2.4	−0.2	−0.3	−3.0	−0.1	0.4
random access	−1.0	−0.9	−0.1	−3.3	−0.8	−0.3
low delay	−1.0	−1.0	−0.8	−1.4	−0.4	0.0

in the secondary color components. This is demonstrated by additional results shown in Table IX, which were obtained by running simulations for eight sequences in YCbCr 4:4:4 and RGB 4:4:4 formats according to the JVET Common Test Conditions for non-4:2:0 color formats [55].

VII. CONCLUSION

Transform coding of prediction error blocks is one of the key components in hybrid video coding. This paper described the fundamental principles and implementation considerations behind the quantization and entropy coding design in the recently finalized Versatile Video Coding (VVC) standard. It introduced the trellis-coded quantization feature of VVC and highlighted the improvements, relative to the High Efficiency Video Coding (HEVC) standard, made in both the entropy coding scheme for quantized transform coefficients and the binary arithmetic coding engine. In addition, a newly integrated method for a block-wise joint coding of chroma residuals in color images and videos was discussed. A comprehensive performance evaluation, conducted by means of a large set of video sequences of varying resolution, confirmed the increased coding efficiency (measured in bit-rate reduction at the same peak signal-to-noise ratio) achieved by each of the aforementioned improvements, as well as by the combination of these coding tools. Beside the technology described in this paper, VVC includes a variety of other improvements such as the flexible block partitioning [5], block-adaptive transforms [6], various improvements of the intra- and inter-picture prediction, and new adaptive in-loop filters. By combining all these coding tools, VVC is able to outperform its predecessors, HEVC and AVC, in compression efficiency by a considerable margin and, thus, represents the new state-of-the-art in video coding.

REFERENCES

[1] ITU-T and ISO/IEC, "Versatile Video Coding," ITU-T Rec. H.266 and ISO/IEC 23090-3, to be published.

[2] B. Bross, J. Chen, J.-R. Ohm, and G. J. Sullivan, "Developments in international video coding standardization after AVC, with an overview of Versatile Video Coding (VVC)," *Proc. IEEE*, to appear.

[3] ITU-T and ISO/IEC, "High efficiency video coding," ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC), version 1, 2013.

[4] ITU-T, "Video codec for audiovisual services at $p \times 64$ kbit/s," ITU-T Rec. H.261, version 3, 1993.

[5] Y.-W. Huang, J. An, H. Huang, X. Li, S.-T. Hsiang, K. Zhang, H. Gao, and J. Ma, "Block partitioning structure in the VVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, this issue.

[6] X. Zhao, S.-H. Kim, Y. Zhao, H. E. Egilmez, M. Koo, S. Liu, J. Lainema, and M. Karczewicz, "Transform coding in the VVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, this issue.

[7] V. K. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 9–21, Sep. 2001.

[8] T. Nguyen, X. Xu, F. Henry, R.-L. Liao, M. Sarwer, M. Karczewicz, Y.-H. Chao, J. Xu, S. Liu, and G. J. Sullivan, "Overview of the screen content support in VVC: Applications, coding tools, and performance," *IEEE Trans. Circuits Syst. Video Technol.*, this issue.

[9] ITU-T and ISO/IEC, "Advanced video coding for generic audiovisual services," ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), 2003.

[10] G. J. Sullivan, "Efficient scalar quantization of exponential and Laplacian random variables," *IEEE Trans. Inf. Theory*, vol. 42, no. 5, pp. 1365–1374, Sep. 1996.

[11] H. Schwarz and T. Wiegand, "Video coding: Part II of fundamentals of source and video coding," *Found. and Trends in Signal Processing*, vol. 10, no. 1-3, pp. 1–346, Dec. 2016.

[12] T. D. Lookabaugh and R. M. Gray, "High-resolution quantization theory and the vector quantizer advantage," *IEEE Trans. Inf. Theory*, vol. 35, no. 5, pp. 1020–1033, Sep. 1989.

[13] G. Clare, F. Henry, and J. Jung, "Sign data hiding," Joint Collaborative Team on Video Coding (JCT-VC), doc. JCTVC-G271, Nov. 2011.

[14] X. Yu, J. Wang, D. He, G. Martin-Cocher, and S. Campell, "Multiple signs bits hiding," Joint Collaborative Team on Video Coding (JCT-VC), doc. JCTVC-H0481, Feb. 2012.

[15] J. Sole, et. al., "Transform coefficient coding in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1765–1777, Dec. 2012.

[16] H. Schwarz, T. Nguyen, D. Marpe, and T. Wiegand, "CE7: Transform coefficient coding and dependent quantization (tests 7.1.2, 7.2.1)," Joint Video Experts Team (JVET), doc. JVET-K0071, Jul. 2018.

[17] H. Schwarz, T. Nguyen, D. Marpe, and T. Wiegand, "Hybrid video coding with trellis-coded quantization," in *Proc. of Data Compression Conference (DCC)*, Mar. 2019.

[18] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources," *IEEE Trans. Commun.*, vol. 38, no. 1, pp. 82–93, Jan. 1990.

[19] T. R. Fischer and M. Wang, "Entropy-constrained trellis-coded quantization," *IEEE Trans. Inf. Theory*, vol. 38, no. 2, pp. 415–426, Mar. 1992.

[20] R. L. Joshi, V. J. Crump, and T. R. Fischer, "Image subband coding using arithmetic coded trellis coded quantization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 6, pp. 515–523, Dec. 1995.

[21] J. H. Kasner, M. W. Marcellin, and B. R. Hunt, "Universal trellis coded quantization," *IEEE Trans. Image Process.*, vol. 8, no. 12, pp. 1677–1687, Dec. 1999.

[22] H. Schwarz, S. Schmidt, P. Haase, T. Nguyen, D. Marpe, and T. Wiegand, "Additional support of dependent quantization with 8 states," Joint Video Experts Team (JVET), doc. JVET-Q0243, Jan. 2020.

[23] K. Ramchandran and M. Vetterli, "Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 700–704, Sep. 1994.

[24] M. Karczewicz, Y. Ye, and I. Chong, "Rate distortion optimized quantization," ITU-T SG16/Q6 (VCEG), doc. VCEG-AH21, Jan. 2008.

[25] E.-H. Yang and X. Yu, "Soft decision quantization for H.264 with main profile compatibility," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 1, pp. 122–127, Jan. 2009.

[26] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards — Including High Efficiency Video Coding (HEVC)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1669–1684, Dec. 2012.

[27] Joint Collaborative Team on Video Coding (JCT-VC), "HEVC test model software (HM)," software repository is available at https://vcgit.hhi.fraunhofer.de/jct-vc/HM.

[28] Joint Video Experts Team (JVET), "VVC test model software (VTM)," software repository is available at https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM.

[29] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[30] C. R. Helmrich, S. Bosse, M. Siekmann, H. Schwarz, D. Marpe, and T. Wiegand, "Perceptually optimized QP adaptation and associated distortion measure," in *Proc. of Data Compression Conference (DCC)*, Mar. 2019.

[31] J. Pfaff, A. Filippov, S. Liu, X. Zhao, J. Chen, S. de Luxán Hernández, V. Rufitskiy, A. K. Ramasubramonian, and G. van der Auwera, "Intra prediction and mode coding in the VVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, this issue.

[32] H. Schwarz, T. Nguyen, D. Marpe, and T. Wiegand, "Non-CE7: Alternative entropy coding for dependent scalar quantization," Joint Video Experts Team (JVET), doc. JVET-K0072, Jul. 2018.

[33] H. Schwarz, T. Nguyen, D. Marpe, T. Wiegand, M. Karczewicz, M. Coban, and J. Dong, "CE7: Transform coefficient coding with reduced number of regular-coded bins (tests 7.1.3a, 7.1.3b)," Joint Video Experts Team (JVET), doc. JVET-L0274, Oct. 2018.

[34] F. Bossen, "CE7-related: Modified binarization for reduced bin-to-bit ratio," Joint Video Experts Team (JVET), doc. JVET-L0325, Oct. 2018.

[35] T.-D. Chuang, S.-T. Hsiang, Z.-Y. Lin, C.-Y. Chen, Y.-W. Huang, and S.-M. Lei, "CE7-related: Constraints on context-coded bins for coefficient coding," Joint Video Experts Team (JVET), doc. JVET-L0145, Oct. 2018.

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

[37] T. Nguyen, D. Marpe, and T. Wiegand, "Non-CE11: Proposed cleanup for transform coefficient coding," Joint Collaborative Team on Video Coding (JCT-VC), doc. JCTVC-H0228, Feb. 2012.

[38] J. Chen, W.-J. Chien, M. Karczewicz, X. Li, H. Liu, A. Said, L. Zhang, and X. Zhao, "Further improvements to HMKTA-1.0," ITU-T SG16/Q6 (VCEG), doc. VCEG-AZ07, Jun. 2015.

[39] D. Flynn, D. Marpe, M. Naccari, T. Nguyen, C. Rosewarne, K. Sharman, J. Sole, and J. Xu, "Overview of the range extensions for the HEVC standard: Tools, profiles, and performance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 4–19, Jan. 2016.

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

[41] C. Holt, "Forecasting trends and seasonal by exponentially weighted moving averages," *O.N.R. Memorandum, Carnegie Inst. Of Technology*, no. 2, 1957.

[42] A. Alshin, E. Alshina, and H. Park, "CE1 (subset B): Multiparameter probability up-date for CABAC," Joint Collaborative Team on Video Coding (JCT-VC), doc. JCTVC-G0764, Nov. 2011.

[43] A. Alshin, E. Alshina, and H. Park, "High Precision Probability Estimation for CABAC," in *Proc. of Visual Communications and Image Processing (VCIP)*, Nov. 2013.

[44] F. Bossen, J. Dong, and H. Kirchhoffer, "Description of Core Experiment 1 (CE1): CABAC Initialization," Joint Video Experts Team (JVET), doc. JVET-N1021, Apr. 2019.

[45] F. Bossen, "CE5-related: Implementation considerations for entropy coding," Joint Video Experts Team (JVET), doc. JVET-K0273, Jul. 2018.

[46] H. Kirchhoffer and A. Said, "Description of Core Experiment 5 (CE5): Arithmetic Coding Engine," Joint Video Experts Team (JVET), doc. JVET-L1025, Jan. 2019.

[47] K. Zhang, J. Chen, L. Zhang, X. Li, and M. Karczewicz, "Enhanced Cross-Component Linear Model for Chroma Intra-Prediction in Video Coding," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3983–3997, Aug. 2018.

[48] J. Lainema, "CE7-related: Joint coding of chrominance residuals," Joint Video Experts Team (JVET), doc. JVET-M0305, Jan. 2019.

[49] C. Helmrich, C. Rudat, T. Nguyen, H. Schwarz, D. Marpe, and T. Wiegand, "CE7-related: Joint chroma residual coding with multiple modes," Joint Video Experts Team (JVET), doc. JVET-N0282, Mar. 2019.

[50] C. Rudat, C. Helmrich, J. Lainema, T. Nguyen, H. Schwarz, D. Marpe, and T. Wiegand, "Inter-component transform for color video coding," in *Proc. of Picture Coding Symposium (PCS)*, Nov. 2019.

[51] F. Bossen, J. Boyce, K. Sühring, X. Li, and V. Seregin, "JVET common test conditions and software reference configurations for SDR video," Joint Video Experts Team (JVET), doc. JVET-N1010, May 2019.

[52] European Broadcasting Union. UHD-1 test sequences. [Online]. Available: https://tech.ebu.ch/testsequences/uhd-1

[53] European Broadcasting Union. SVT high definition multi format test set. [Online]. Available: https://tech.ebu.ch/hdtv/hdtv_test-sequences

[54] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves," ITU-T SG16/Q6 (VCEG), doc. VCEG-M33, Apr. 2001.

[55] Y.-H. Chao, Y.-C. Sun, J. Xu, and X. Xu, "JVET common test conditions and software reference configurations for non-4:2:0 colour formats," Joint Video Experts Team (JVET), doc. JVET-R2013, Apr. 2020.