Parallelization of CABAC Transform Coefficient Coding for HEVC

Vivienne Sze, Madhukar Budagavi
Texas Instruments
Video, Imaging and Vision Lab
Systems and Applications R&D Center
{sze, madhukar}@ti.com

***Abstract*—Data dependencies in CABAC make it difficult to parallelize and thus limits its throughput. The majority of the bins processed by the CABAC are used to represent the prediction error/residual in terms of quantized transform coefficients. This paper provides an overview of the various improvements to context selection and scans in transform coefficient coding that enable HEVC to potentially achieve higher throughput relative to AVC/H.264. Specifically, it describes changes that remove data dependencies in significance map and coefficient level coding. Proposed and adopted techniques up to HM-4.0 are discussed. This work illustrates that accounting for implementation cost when designing video coding algorithms can result in a design that can enable higher processing speed and reduce hardware cost, while still delivering high coding efficiency.**

I. Introduction

High Efficiency Video Coding (HEVC) is currently being developed by the Joint Collaborative Team for Video Coding (JCT-VC). It is expected to deliver up to a 50% higher coding efficiency compared to its predecessor AVC/H.264. In order to improve coding efficiency, HEVC uses larger block and transform sizes, additional loops filters, and highly adaptive entropy coding. While high coding efficiency is important for reducing the transmission and storage cost of video, processing speed and area cost also need to be considered in the development of next generation video coding in order to handle the increasing demand for higher resolution and frame rates.

Context-Adaptive Binary Arithmetic Coding (CABAC) [1] is a form of entropy coding used in AVC/H.264 [2] and also in HEVC [3]. While CABAC provides high coding efficiency, its data dependencies cause it to be a throughput bottleneck for AVC/H.264 video codecs [4]. The throughput of CABAC is determined based on the binary symbols (bins) that it can process per second. Syntax elements of the transform coefficient data, which represent the residual of the prediction error, accounts for a significant portion of the bin workload. For instance, under common conditions [5], with quantization parameter (QP) ranging from 22 to 37, transform coefficient data accounts for 60% to 90% of the total bins for All Intra sequences, 30% to 80% of the total bins for Low Delay, and 20% to 90% of the total bins for Random Access. At the same time, the transform coefficients also account for a significant portion of the total bits of a compressed video, and as a result the compression of transform coefficients significantly impacts the overall coding efficiency. Thus, transform coefficient coding with CABAC must be carefully designed in order to balance coding efficiency and throughput demands. Accordingly, as part of the HEVC standardization process, a core experiment on coefficient scanning and coding was established to investigate tools related to transform coefficient coding [6].

This paper focuses on tools that enable parallel processing of transform coefficients with CABAC for HEVC. It will discuss how CABAC transform coefficient coding has evolved from AVC/H.264 to the HM-4.0 version of the HEVC test model [3], [7] as well as describe additional improvements under consideration for HEVC. Section II provides an overview of CABAC entropy coding to explain the cause of the throughput bottleneck. Section III describes the key steps in transform coefficient coding using CABAC. Section IV and Section V describes changes to transform coefficient coding that enable increased throughput and higher coding efficiency for HEVC.

II. CABAC Throughput

Entropy coding is a form of lossless compression used at the last stage of video encoding (and first stage of video decoding), after the video has been reduced to a series of syntax elements. Arithmetic coding is a type of entropy coding that can achieve compression close to the entropy of a sequence by effectively mapping the symbols (i.e. syntax elements) to codewords with non-integer number of bits. In H.264/AVC, CABAC provides a 9 to 14% improvement over the Huffman-based CAVLC [1]. In HM-3.0, CABAC provides a 5 to 9% improvement over CAVLC [8].

CABAC involves three main functions: binarization, context modeling and arithmetic coding. Binarization maps syntax element to binary symbols (bins). Context modeling estimates the probability of the bins and arithmetic coding compresses the bins to bits.

One of the main reasons that a CABAC engine has limited throughput is due to the data dependencies for context selection. Often the context selection for a bin depends on the value of a previously encoded/decoded bin. This dependency makes parallelism difficult and costly to achieve, particularly at the decoder, if multiple bins are to be decoded at the same time. The closer the bins are in terms of time, the tighter the dependency, since closer, or consecutive, bins are likely to

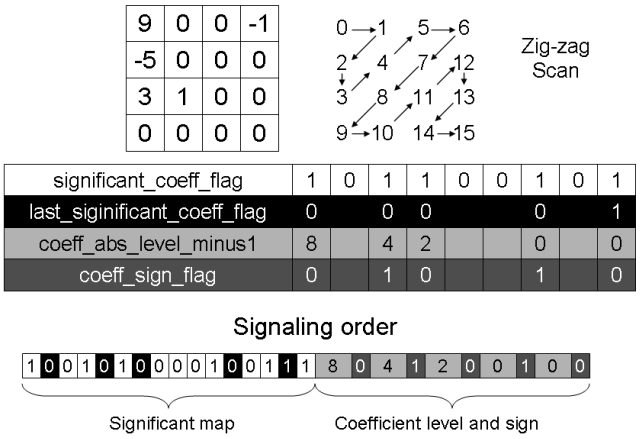


Fig. 1: Example of transform coefficient coding for a 4x4 TU in AVC/H.264.

be processed in parallel within the same cycle. In the worst case, the current bin depends on the immediate preceding bin. If the context of a bin depends on the value of another bin being decoded in parallel, then speculative computations are required which increases area cost and critical path delay [9]. The amount of speculation grows exponentially with the number of parallel bins which limits the throughput that can be achieved [10]. For HEVC, improvements have been made to the context selection and coefficient scanning in order to reduce the amount of speculation required to process multiple bins in parallel [11].

III. Overview of Transform Coefficient Coding

In video coding, both intra and inter prediction are used to reduce the amount of data that needs to be transmitted. Rather than sending the pixels, the prediction error is transmitted. This prediction error is transformed from spatial to frequency domain to leverage energy compaction properties, and after quantization, it can be represented in terms of a few coefficients. The method of signaling the value and the frequency position of these coefficients is referred to as transform coefficient coding.

In CABAC, the position of the coefficients is transmitted in the form of a *significance map*. Specifically, the significance map indicates the location of the non-zero coefficients. The *coefficient level* information is then only transmitted for the coefficients with values greater than one, while the coefficient sign is transmitted for all non-zero coefficients. An example of transform coefficient coding in AVC/H.264 is shown in Fig. 1.

IV. Significance Map

In AVC/H.264, the significance map is signaled by transmitting a significant_coeff_flag (SCF) for each position to indicate whether the coefficient is non-zero. The positions are processed in an order based on a zig-zag scan. After each non-zero SCF, an additional flag called last_significant_coeff_flag (LSCF) is immediately sent to indicate whether it is the last non-zero SCF; this prevents unnecessary SCF from being signaled. Different contexts are used depending on the position within the 4x4 and 8x8 transform unit (TU), and whether the bin represents an SCF or LSCF. Since SCF and LSCF are interleaved, the context selection of the current bin depends on the immediate preceding bin. The dependency of LSCF on SCF results in a strong bin to bin dependency for context selection for significance map in the AVC/H.264.

In HM-1.0, additional dependencies are introduced between SCF to improve coding efficiency. Additional TU sizes of 16x16 and 32x32 are used in HEVC. The context selection for SCF in these larger TU depends on the number of non-zero neighbors to give coding gains between 1.4 to 2.8% [12]. Specifically, the context of SCF depends on up to 10 neighbors as shown in Fig. 2a.

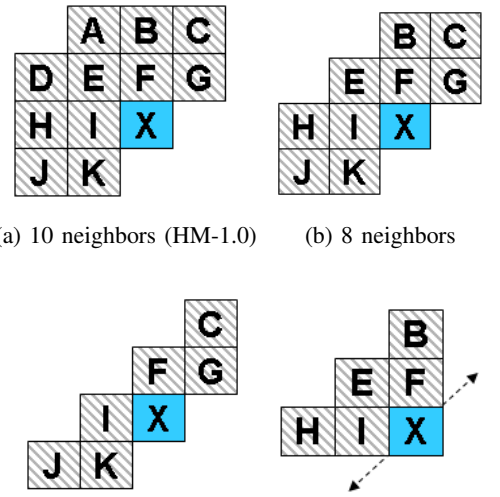
A. *significant_coeff_flag (SCF)*

Simplification by reducing the neighbor dependencies was proposed in [13]. Fewer neighbors reduces the context selection logic complexity, storage cost, and has potential throughput benefits. Reducing number of neighbors had minimal cost to coding efficiency. For instance, using only a maximum of 8 neighbors (removing neighbors A and D as shown in Fig. 2b) had negligible impact on coding efficiency, while using only 6 neighbors (removing neighbors A, B, D, E and H as shown in Fig. 2c) results in a coding loss of only 0.2%. This was further extended in [14] where only a maximum of 5 neighbors is used by removing dependencies on positions G and K, as shown in Fig. 2d, since those neighbors pertain to the most recently processed SCF. This simplification was adopted into HM-3.0. It should be noted that the neighbor dependencies are inverted from top-left to bottom-right in HM-4.0 to accommodate reverse significance map scanning (from high frequency to low frequency) [15].

Despite reducing the neighbors in HM-3.0, dependency on the most recently processed SCF still existed for the positions at the edge of the transform as shown in Fig. 3a. In order to address this, in HM-4.0, a diagonal scan was introduced to replace the zig-zag scan [16] as shown in Fig. 3b. Changing from zig-zag to diagonal scan had negligible impact on coding efficiency, but removed dependency on recently processed SCF for all positions in the TU.

B. *last_significant_coeff_flag (LSCF)*

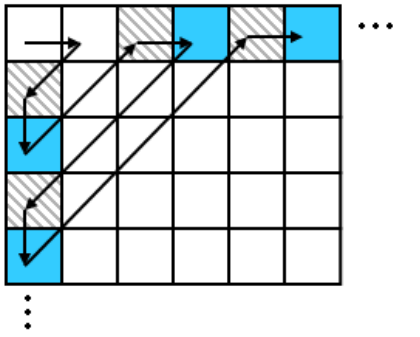
As mentioned earlier, there are strong data dependencies between SCF and LSCF. The concept of parallel context processing (PCP) is introduced in [17] to address this concern. To reduce interleaving of SCF and LSCF, significance map PCP technique parallelizes significance map coding by transmitting a LSCF only once per N number of SCF. If all of the N SCF are zero, LSCF is not transmitted. [18] avoids all interleaving of SCF and LSCF altogether. Specifically, the X, Y position of the last non-zero SCF (last_significant_coeff_x and last_significant_coeff_y) is sent rather than LSCF. For instance, in the example of shown in Fig. 1, last_significant_coeff_x equal to 3 and last_significant_coeff_y equal to 0 is sent rather than



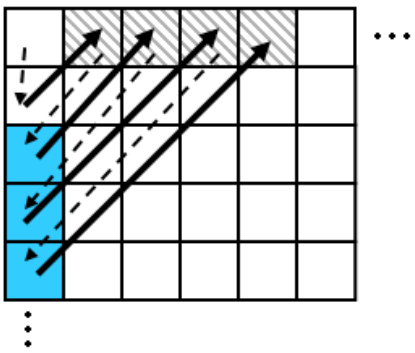
(a) 10 neighbors (HM-1.0) (b) 8 neighbors

(c) 6 neighbors (d) 5 neighbors (HM-3.0)

Fig. 2: Neighbor dependencies for SCF context selection. X in blue represents the current position of the bin being processed.



(a) Zig-zag scan



(b) Diagonal scan

Fig. 3: Diagonal scan is used to avoid dependency on most recently processed SCF. Context selection for blue positions are affected by values of the neighboring grey positions.

Fig. 4: Grouping bypass bins to increase throughput. **b1=coeff_abs_level_greater1_flag, b2=coeff_abs_level_greater2_flag, s=sign_flag, GR=coeff_abs_level_minus3**

last_significant_coeff_flag. Signaling the X, Y position of the last non-zero SCF was adopted into HM-3.0.

V. Coefficient Level and Sign

In AVC/H.264, the coefficient level is composed of two parts. The first 14 bins, generated with truncated unary binarization, are context coded (i.e. require context selection). The remaining bins, generated by exp-golomb binarization, are bypass coded bins, which means that a fixed equal probability of 0.5 is assumed and thus do not require context selection. After each coefficient level is signaled, the sign is signaled with one bypass bin. It is important to note that bypass coded bins can be processed in parallel much easier than context coded bins [19].

Parallel context processing (PCP) for coefficient level and sign was proposed for HM-1.0 in [20]. As with significance map, reducing the interleaving of bins coded with different contexts reduces the data dependencies for context selection. If the context switches less from bin to bin, then fewer speculative computations for context selection are required. Grouping the first bin of coefficient levels together as well as grouping the sign bins together was adopted into HM-1.0.

PCP can be further leveraged with the new binarization scheme for coefficient levels that was introduced in HM-3.0 [21]. In HM-3.0, the coefficient level is composed of three parts. Only the first two bins (coeff_abs_level_greater1_flag and coeff_abs_level_greater2_flag) are context coded and the remaining bins, generated with Golomb-Rice binarization, are bypass coded (coeff_abs_level_minus3).

In [22], PCP is applied to the second bin of the coefficient level. As a result, the all bypass coded bins are grouped together which maximizes the throughput advantages of bypass bins. coeff_abs_level_minus3 across multiple coefficients are grouped together and sign bins are grouped together as shown in Fig. 4. To reduce storage cost, the sign bins are signaled before coeff_abs_level_minus3 bins. This reordering of data to enable parallel context processing has no impact on coding efficiency and was adopted into HM-4.0.

VI. Summary

Methods such as parallel context processing and diagonal scans are used to reduce data dependencies in CABAC transform coefficient coding for HEVC. An example of transform coefficient coding in HM-4.0 is shown in Fig. 5.

Transform coefficient coding for HEVC has been carefully designed to enable it to deliver higher throughput and higher coding efficiency as compared with AVC/H.264. Dependencies

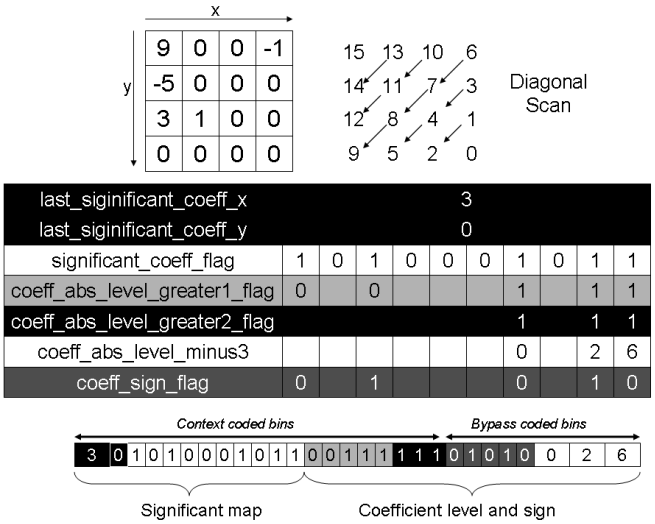


Fig. 5: Example of transform coefficient coding for a 4x4 TU in HEVC (HM-4.0).

Tool	HM	Benefit	Coding Gains
Neighbor based context selection for SCF [12]	1.0	coding gain	1.4% to 2.8%
PCP (sign) [20]	1.0	throughput	0.0%
Simplified neighboring dependency [14]	3.0	throughput	0.0% to 0.1%
Golomb-Rice Codes of coefficient level [21]	3.0	throughput	0.0% to 0.1%
Last position coding [18]	3.0	throughput	0.0% to 0.1%
PCP (SCF) [22]	4.0	throughput	0.0%
Diagonal Scan [16]	4.0	throughput	0.0% to 0.1%

TABLE I: Summary of throughput related transform coefficient coding tools adopted up to HM-4.0.

for context selection of consecutive bins has been reduced for significance map and coefficient level in order to enable multiple bins to be processed in parallel. At same time, new tools for improved coding efficiency have been simplified to meet this requirement. Table I summarizes the tools adopted up to HM-4.0. Parallel transform coefficient coding continues to be an active area of development in JCT-VC [23].

Similar approaches of grouping bypass bins has also been applied to syntax elements beyond transform coefficients to speed up CABAC and reduce hardware cost. This work shows that by accounting for implementation cost when designing video coding algorithms results in a design that can maximize processing speed and minimize area cost, while delivering high coding efficiency in the next generation video coding standard.

ACKNOWLEDGMENT

The work presented in this paper was carried out as a part of the core experiment on coefficient scanning and coding for HEVC standardization.

REFERENCES

[1] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. on CSVT*, vol. 13, no. 7, pp. 620– 636, July 2003.
[2] "Recommendation ITU-T H.264: Advanced Video Coding for Generic Audiovisual Services," Tech. Rep., ITU-T, 2003.
[3] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, and T. Wiegand, "JCTVC-F803: WD4: Working Draft 4 of High-Efficiency Video Coding," Joint Collaborative Team on Video Coding (JCT-VC), July. 2011.
[4] Vivienne Sze, Madhukar Budagavi, and Mehmet Umut Demircin, "VCEG-AJ31: CABAC throughput requirements for real-time decoding," Video Coding Experts Group (VCEG), October 2008.
[5] F. Bossen, "JCTVC-D600: Common test conditions and software reference configurations," Joint Collaborative Team on Video Coding (JCT-VC), Jan. 2011.
[6] V. Sze, K. Panusopone, J. Chen, T. Nguyen, and M. Coban, "JCTVC-C511: Description of Core Experiment 11: Coefficient Scanning and Coding," Joint Collaborative Team on Video Coding (JCT-VC), Oct. 2010.
[7] "HEVC Test Model, HM 4.0," https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-4.0/.
[8] T. Davies and A. Fuldseth, "JCTVC-F162: Entropy coding performance simulations," Joint Collaborative Team on Video Coding (JCT-VC), July. 2011.
[9] Vivienne Sze, Madhukar Budagavi, Anantha P. Chandrakasan, and Minhua Zhou, "Parallel CABAC for Low Power Video Coding," in *IEEE Inter. Conf. on Image Processing,*, October 2008, pp. 2096–2099.
[10] V. Sze, "JCTVC-D244: Context selection complexity in HEVC CABAC," Joint Collaborative Team on Video Coding (JCT-VC), Jan. 2011.
[11] V. Sze, J. Chen, T. Nguyen, K. Panusopone, and J. Sole, "JCTVC-G041: CE11: Summary report of Core Experiment on coefficient scanning and coding," Joint Collaborative Team on Video Coding (JCT-VC), Nov. 2011.
[12] T. Nguyen, D. Marpe, H. Schwarz, and T. Wiegand, "JCTVC-D061: CE11: Evaluation of Transform Coding tools in HE configuration," Joint Collaborative Team on Video Coding (JCT-VC), Oct. 2010.
[13] V. Sze and M. Budagavi, "JCTVC-C227: Parallelization of HHI_TRANSFORM_CODING," Joint Collaborative Team on Video Coding (JCT-VC), Oct. 2010.
[14] A. Cheung and W. Lui, "JCTVC-D260: Parallel processing friendly simplified context selection of significance map," Joint Collaborative Team on Video Coding (JCT-VC), Jan. 2011.
[15] J. Sole, R. Joshi, and M. Karczewicz, "JCTVC-F288: CE11: Unified scans for the significance map and coefficient level coding in high efficiency," Joint Collaborative Team on Video Coding (JCT-VC), July 2011.
[16] V. Sze and M. Budagavi, "JCTVC-F129: CE11: Parallelization of HHI_TRANSFORM_CODING Fixed Diagonal Scan," Joint Collaborative Team on Video Coding (JCT-VC), July. 2011.
[17] M. Budagavi and M. U. Demircin, "JCTVC-B088: Parallel Context Processing techniques for high coding efficiency entropy coding in HEVC," Joint Collaborative Team on Video Coding (JCT-VC), July. 2010.
[18] J. Sole, R. Joshi, and M. Karczewicz, "JCTVC-E338: CE11: Parallel Context Processing for the significance map in high coding efficiency," Joint Collaborative Team on Video Coding (JCT-VC), March. 2011.
[19] Yao-Chang Yang and Jiun-In Guo, "High-Throughput H.264/AVC High-Profile CABAC Decoder for HDTV Applications," *IEEE Trans. on CSVT*, vol. 19, no. 9, pp. 1395 –1399, September 2009.
[20] M. Budagavi, "JCTVC-C062: TE8: TI parallel context processing (PCP) proposal," Joint Collaborative Team on Video Coding (JCT-VC), Oct. 2010.
[21] T. Nguyen, "JCTVC-E253: CE11: Coding of transform coefficient levels with Golomb-Rice codes," Joint Collaborative Team on Video Coding (JCT-VC), March 2011.
[22] V. Sze and M. Budagavi, "JCTVC-F130: Parallel Context Processing of Coefficient Level," Joint Collaborative Team on Video Coding (JCT-VC), July. 2011.
[23] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, and T. Wiegand, "JCTVC-H1103: High efficiency video coding (HEVC) text specification draft 6," Joint Collaborative Team on Video Coding (JCT-VC), Feb. 2012.