International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Mathematical Modelling of Engineering Problems
Vol. 9, No. 2, April, 2022, pp. 458-462
Journal homepage: http://iieta.org/journals/mmep

Optimized Context-Adaptive Binary Arithmetic Coder in Video Compression Standard Without Probability Estimation

S.T. Mrudula[1*], K.E. Srinivasa Murthy[2], M.N. Giri Prasad[3]

[1] Department of ECE, JNTUA, Anantapuramu 515002, A.P, India
[2] Ravindra College of Engineering for Women, Kurnool 518452, A.P, India
[3] Academic Audit, JNTUA, Anantapuramu 515002, A.P, India

Corresponding Author Email: sunkaramrudula@gmail.com

https://doi.org/10.18280/mmep.090222

Received: 11 August 2021
Accepted: 20 January 2022

Keywords:
CABAC, BAC, M-Coder, re-norm, Probability Estimation

ABSTRACT

CABAC is a Context Adaptive Binary Arithmetic Coder utilized in novel AVC/H.264 of video standard. AC (arithmetic coding) permits important enhancement in the compression. However, the complexity of implementation is main drawback because of slowness and hardware cost. In this paper, we propose the implementation of MPEG4/H-264 AVC against M-decoder without PE (Probability Estimation). Furthermore, in order to estimate an algorithm, we have compared many existing methods, and the comparison takes place based on power dissipation and device utilization.

1. INTRODUCTION

Throughout An entropy coding is a lossless coding technique which compress the data using its statistical information. CABAC (Context- Based Adaptive-Binary Arithmetic-Coding) is greatly efficient entropy coding technique. It has been by recent standard of video compression H.264 and plays eminent role in improvising its coding-efficiency. Compared with UVLC (Universal variable-length coding), CBAC takes full benefits of the feature of arithmetic codes and significance statistical properties of video streams that develops an efficiency of coding.

CABAC is higher performance of entropy coding technique utilized in an AVC/H.264. The main baseline of UVLC method is based on an Exp-Golmb codes and BAC (Binary arithmetic-coder) [1]. The context-based method carries out complex binarization symbols in the efficient manner by unary tree. The BAC (Binary arithmetic-coder) [2, 3] compresses the binary symbols attaining bit rates nearer to entropy limit.

AC (Arithmetic coding) united with an efficient context modelling provides high compression rations than compression techniques such as Golomb-Rice and Huffman [2]. In an AVC context, CABAC outcomes are up-to twenty percent than those whose achieved with the coder of baseline entropy [4]. Other sources insist that the CABAC overcomes UVLC by 30 to 40 percent [5]. Later, the CABAC provides clear benefits over compression techniques implemented in the existing methods. Higher compression method comes with the price. Both AC and context modelling need higher number of memory and operations accesses. Henceforth, the efficient implementations are essential.

Moreover, the analysis of rate-distortion significantly maximizes number of the operations. An image is encoded by following techniques (half-pixel, intra, inter pixel-MC and unchanged) an optimal technique is selected based on minimizing the distortion for minimal maximization in bit rate. When the analysis of rate distortion is carried out with CABAV instead of the UVLC. Hence, the hardware acceleration is required as an amount of data to process maximizes with quality settings and size of image.

In this work, we represent new architecture for CABAC. The efficient mechanism for the context managing is introduced that enhances the throughput and reduces workload based on the mixed design of software and hardware. A novel architecture of AC is represented that takes the benefits of novel characteristics discovered in the CABAC. Thus, fast implementation is represented is represented capable to process the symbols at speed demanded by higher quality of video encoding. This paper is organized in such a way that section-2 represents CABAC, section-3 represents Background of BAC, section 4 represents result analysis and finally conclude our work.

2. OPTIMIZED CONTEXT-BASED ADAPTIVE-BINARY ARITHMETIC-CODING

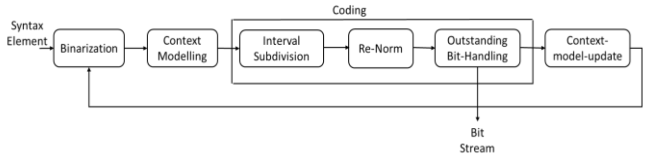

Figure 1. Basic diagram of CABAC

Optimized Context- Based Adaptive-Binary Arithmetic-Coding is CABAC utilized in an AVC/H.264 [1]. CABAC has two parts. At first, parameters, events and coefficients made by the help of video-encoder can be converted to the binary symbol. Each symbol is assigned to the specific context. Afterwards, binary symbols can be compressed with AC utilizing context information. The figure above represents the basis diagram of CABAC. This work represents context information managing and BAC (binary arithmetic-coder). All elements highlighted in Figure 1.

2.1 Context-based-compression

The context-based compression is very crucial method in order to improvise Markov-sources compression. An efficiency of the entropy compressors relies on the adequate context-managing. The division of optimal context assigns various symbols in various context without any fail in the context atomization. Such as, four-bit numbers can be encoded as four-binary symbols utilizing the context for all of them. Anyways, it could occur 2 less important bits that does not profit through this context assignation. Thus, encoding bits within similar context could improve and simplify the compression.

CABAC assumes 701 various contexts. All of them are not utilized in various operation of encoder modes. Such as, when encoding process is not interlaced the images, 240 contexts render are not used. The process symbol of converting generic to the binary ones contains more than one evaluation conditions. Here, we have represented 2 cases. The syntactical elements produce only one binary symbol and all symbols are encoded per frame. Afterwards, contributes to smaller part of arithmetic coder-workload.

Some syntactical elements produce many binary symbols in the predictable fashion and encoded per frame 100 times. Mainly, these syntactical elements transform significance and coefficient maps and contribute to the workload of arithmetic coder. The first element classes are well suited to processed in the software. Second elements of syntactical class may require hardware-acceleration. Furthermore, those symbols are generated in the predictable manner, the efficient implementation can enhance memory access and boost the performance. As it is represented in Figure 1, the context model may be updated after each access. Because of irregular access patterns, the memory bandwidth can commit the encoder throughput. Henceforth, the context managing is implemented in the optimized manner to maximize the data reuse, enable fast update and permit pre-fetching. An effective throughput is improved so that the requirements of speed are met.

2.2 Arithmetic coding

AC [2] contains iterative interval division based on various probability symbols. The CABAC implements the binary coder, the particular case permits substantial reduction of complexity with higher compression efficiency. Specifically, the arithmetic coder of CABAC is associated to the family of Q-Coder [3]. By calling range and low to lower point and current interval length, an encoding equation are given as:

$$\begin{array}{l} \text{MPS=Most-Probable Symbol} \\ \qquad L_{new} = L \\ \qquad R_{new} = R - rLPS \\ \text{LPS=Least-Probable Symbol} \\ \qquad L_{new} = L + R + rLPS \\ \qquad R_{new} = rLPS \end{array} \tag{1}$$

The $rLPS$ value depends of current context encoding-state and range value. Thus, the dependence of recursive exists in Figure 1.

Figure 2 represents encoding process. At any time, lower values keep track in which the symbols were encoded. The range length minimizes faster for the LPS. Small range values require high precision for the encoding that means more bits. To utilize an integer arithmetic, the value of range is normalized after each iteration.

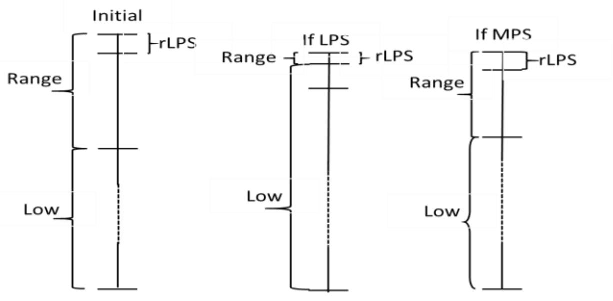

Figure 2. Example of Binary Arithmetic-coding

Furthermore, the CABAC contains mode for encoding of equally-probable-symbols. In this mode, the context accessing is not required. It also permits fast implementation in the software. The equally-probable symbols can be encoded utilizing similar circuit by the selection of appropriate operands. The encoding termination is implemented by the help of special encoding-process. Afterwards, the state of internal encoder is encoded by utilizing similar circuit by appropriate operands-selection. The encoding termination can also have implemented by the help of special process of encoding. Then, the state of internal encoder is flushed towards an output.

As an outcome of encoding process, low value maximizes while range value reduces. To keep both operands size under control the range value is normalized in each cycle. The low value can be shifted as consequences; its value is shortened. The shortened bits constitute an outcome of encoding process. They are also packed into the bytes and send to an output.

3. BACKGROUND OF BAC

Here, first we represent basic principle of BAC (Binary Arithmetic Coding) with the particular focus on the implementation related features. In BAC, it is convenient to distinguish among 2 symbols of binary alphabet not by utilizing their actual symbols “0” and “1” values but rather by denoting to their evaluated probability-values. By differentiating among MPS and LPS and keep track the symbol value of MPS ($valMPS$) as probability of LPS ($pLPS$), the simple parameterization is achieved by underlying the given binary alphabet’s probability model.

On the basis of given settings, BAC is carried out by dividing initially in given interval that is represented by the help of its LB (lower bound) $\mathcal{L}$ and range of width $\mathcal{R}$ into the 2 given disjoint sub-intervals: first width interval $\mathcal{R}_{LPS} = \mathcal{R}.p_{LPS}$ that is connected with LPS and second width interval $\mathcal{R}_{MPS} = \mathcal{R} - \mathcal{R}_{LPS}$ is allocated to MPS. Based on binary value to encode, recognized as MPS and LPS, corresponding sub-interval is selected as novel coding interval. Byiterativelyapplying this, the scheme of interval-subdivision to every element x_b of the given sequence $(x_1, x_2, \dots\dots, x_3)$ of the binary symbols to encode, the BAC finally defines a value of z_x in sub-interval $\left[\mathcal{L}^{(\mathcal{N})}, \mathcal{L}^{(\mathcal{N})} + \mathcal{R}^{(\mathcal{N})}\right]$ that outcomes after $\mathcal{N}th$ process of interval sub-division. The binary representation z_x is an arithmetic code-word of an input sequence x.

In our previous work [6], we proposed M-ABRC (Modified-Adaptive BAC). This algorithm reduces bit-capacity of multiplication through its architecture of VLSI, introduced algorithm utilizes Look-Up-table (LUT) based on Virtual-sliding window (VSW) for PE (Probability Estimation). In order to obtain high compression rate, our algorithm has been implemented, this in terms gives good adoption probability in the encoding phase and also provides an absolute estimation of low-entropy binary sources (EBS). The conventional-binary arithmetic-encoding of the symbol value is represented in paper [6]. In this paper, we propose the implementation of MPEG4/H-264 AVC against M-decoder without PE (Probability Estimation).

Algo-1: Procedure of M-decoder without PE

```
Step-1: $\mathcal{R}_{LPS} = \mathcal{R}TAB[m][\mathcal{R} \gg 2^2 + 2)\&3$
Step-2: $\mathcal{R}_{MPS} = \mathcal{R} - \mathcal{R}_{LPS}$
Step-3: $If\ (V < \mathcal{R}_{MPS})$
Step-4: $R = \mathcal{R}_{MPS}, \quad val = valMPS$
Step-5: Else
Step-6: $V = V - \mathcal{R}_{MPS}, val = !\, valMPS$
Step-7: $\mathcal{R} = \mathcal{R}_{LPS}$
Step-8: while ($R < 256$)
Step-9: $\mathcal{R} = \mathcal{R} \ll 1$
Step-10: $V = V \ll 1$
Step-11: $V = V|\ Read\_1\_Bit()$
```

where, $[m]$ is defined as fixed probability state.

In order to ensure that the registers with y bits precision are more sufficient to present $\mathcal{L}^{(y)}$and $\mathcal{R}^{(y)}$ for y, the operation of re-norm is needed, whenever $\mathcal{R}^{(y)}$ decreases certain limit after one and multiple process of sub-division. Moreover, by renormalizing $\mathcal{L}^{(y)}$and $\mathcal{R}^{(y)}$ the leading arithmetic code-word bits z_x can output as soon as they are unambiguously recognized.

Recently, novel design of multiplication free BAC has been represented in 8 and 10 lines. Its eminent innovative features are produced by table-based interval sub-division combined with PE based FSM (Finite-state machine) as well as the fast bypass of coding mode. This is also called M coder (Modulo) family of the BAC methods provides parameterizable trade-off among memory needs requirements and coding efficiency for underlying the LUT. Actually, the design of M-coder can be assumed as the generalization of Q-coder, latter can be resulted through M-coder incarnation that belonging to simplest parameter choice.

Another, more elaborated selection of M-coder has been acquired by ISO/IEC and ITU-T as the normative part of the standard of video coding MPEG4-AVC/H.264 [7]. It provides good trade-off among complexity and the performance of compression, as represented in paper [8, 9].

3.1 M-Coder

The major element of lower-complexity M-coder method of an interval sub-division is quantizing an admissible domain $\mathfrak{D} = \left[2^{\mathscr{b}-2}, 2^{\mathscr{b}-1}\right]$ for range of $\mathbb{R}$ register bring by the help of re-norm into smaller number of C cells. Inorder to simplify, we considered $\mathfrak{D}$ as uniform quantization to be applied, resulting the representative equispaced sets with range values $\mathbb{Q}_0, \mathbb{Q}_1, \ldots\ldots, \mathbb{Q}_{C-1}$, where C is defined as constrained to the power of 2 such as $C = 2^C$ for an integer value $C \geq 0$. By discretisation of LPS related probability-range values $\mathbb{P}_{LPS} \in \left[0, \frac{1}{2}\right]$, the representative probabilities sets $\mathbb{P} = \{\wp_0, \wp_1, \ldots\ldots, \wp_{M-1}\}$ can be reconstructed organized with set of the corresponding transition-rules for the FSM based PE. Both $\mathbb{P}$ and $\mathbb{Q}$ enable the operation of multiplication approximation $\mathbb{P}_{LPS} \times \mathcal{R}$ for an interval sub-division by RTAB which consists $\mathcal{M} \times C$ pre-computed products values like $\{\wp_M \times \mathbb{Q}_C 0 \leq \mathfrak{m} < M; 0 \leq c < C\}$ in selected integer precision. An entity can be addressed by utilizing $\mathbb{m}$ index of state and c index of quantization cell to $\mathcal{R}$ value. The computation c is performed by bit-shift concatenation and the operation of bit masking applied to $\mathcal{R}$, where latter is interpreted as the modulo operation-based operand $C = 2^C$, hence the proposed codes is defined as:

$$c = \left(\mathcal{R} \gg (\mathscr{b} - 2 - c)\right)\&(2^c - 1)$$

For M-coder realization, $\mathscr{b}$ and c are fixed, therefore both operands are given as the fixed values in above equation. By selecting the value of $c = 0$, to the linear, where all $\mathcal{R}$ values have single representation-value is utilized for $\wp_{\mathbb{m}} \times \mathcal{R}$. This type of case is an equivalent to the operation of sub-interval carried out in Q-Coder and its corresponding derivate.

Anyways, for presentation clarity and without generality, we restrict ourselves in specific case with MPEG4/H.264-AVC-compliant M-Coder that corresponds to $c = 2$ and specification of the LUT (Lookup-table). For further simplification, we neglect LUT operations needed to adapt probability state $\mathbb{m}$ during every single decoding/encoding cycle.

3.2 Re-norm procedure

In terms of the implementation costs, the re-norm part of M-coder still suffers through bit-by-bit output/input and as far as the concerns of encoder side also through the bitwise carry over the handling. In the implementation of encoder computationally critical parts can be attributed to bitwise loop of operating re-norm and conditional branching inside this loop is represented in algo-1 line 11.

Although from the decoder perspective, an issue appears to be marginally lessened when comparing Re-norm parts in algo-1, there is still substantial CO (computational overhead) convoluted in the implementation of conventional M-coder because of its sequential bits reading from bit-stream.

4. EXPERIMENTAL RESULT

Our methodology is evaluated by comparing optimized CABAC with the existing methodology. Our methodology is simulated by using Xilinx-version 14.7 and code is written by utilizing the VDHL. For the evaluation of optimized CABAC, we have compared with several existing methodologies, which is mentioned below, and various parameters and constraint has been considered and each case our methodology outperforms than existing methodologies. The result section is divided into 2 section such as device utilization and dynamic and static power. Below Table 1 represents device utilization and Table

2 represents power dissipation.

4.1 Device-utilization

Table 1. Comparison of various coder

Architecture	Technology	Logic cell (no.)	Memory (Kbit)
MQ Coder Dyer [10]	Altera Stratix	1596	8192
MQ Coder Dyer [11]	Altera Stratix	761	2675
MQ Coder [12]	Xilinx Virtex 4-LX80	15,692	4.17
MQ Coder Kai [13]	Xilinx Virtex 4-XC4VL	6974	4269
ABRC [14]	Xilinx Virtex 4-LX80	1688	0
ABRC Shcherbakov [15]	Xilinx Virtex 5-ML507	1544	552960
ABRC [16]	Altera Stratix	1296	0
Optimized CABAC	Xilinx Virtex 4-XC4VF	948	0

Table 1 above represents the outcome of optimized CABAC when FPGA implementation takes place and different architecture is compared. First column represents different architecture, whereas second, third and fourth represents technology, Logic cell and Memory. Number of logic cells is considered as one of the best parameters that has been used for different resources by FPGA technologies. So, maximization in image doesn't affect the hardware resources of optimized CABAC at fixed number of the block size. This methodology also achieves trade-off among adaption speed and precision of the probability of one's due to utilization of different window size. Therefore, it can be preferable for image and video coding standards and also for non-standardized codes. So it is very clear that optimized CABAC consumes lower amount of resources whenever compared to MQ Coder [12], MQ Coder Dyer [10, 11], MQ Coder Kai [13], ABRCShcherbakov [15], ABRC [14] and ABRC [16].

4.2 Power dissipation

Below Table 2 represents power dissipation for optimized CABAC in comparison to MQ-coder and ABRC. First column represents architecture whereas second, third and fourth row represents MQ-Coder, ABRC and optimized CABAC. As we can see that our optimized CABAC slightly outperforms better than other existing methodologies.

Table 2. Power dissipation for optimized CABAC

Architecture	MQ Coder	ABRC	Optimized CABAC
Frequency (MHz)	48.30	105.92	182.75
Normalized power (mW/MHz)	10.117	1.19	0.114
Dynamic power (mW)	488.67	127.05	20.77
Power density (μW/(MHz × Logic cell no.))	0.65	0.71	0.12

Static Power

Static power is defined as power consumed when there is no activity takes place in the circuit. It also generates leakage power and standby power. In order to prove the effectiveness of our model, we have considered this parameter that is given in Figure 3 below. For the comparison analysis, we have considered two models i.e., MQ-Coder kai and ABRC and it is compared with our optimized CABAC. Furthermore, the dissipations of static power 624.68 and 622.55 and optimized CABAC is 182.75 when compared with our optimized CABAC as we can see that our model slightly outperforms than other existing models.

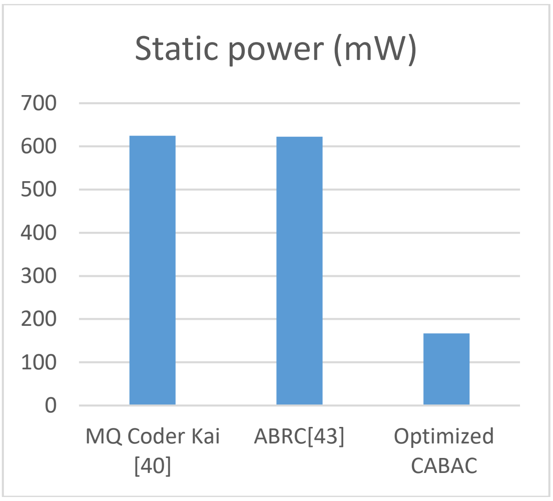

Figure 3. Comparison of static power

Dynamic Power

Dynamic power is defined as the power that is consumed when inputs are in an active state. The dissipation of dynamic power is considered one of the best key parameters while comparing the model. For the comparison analysis, we have considered two models i.e., ABRC and MQ-Coder kai and it is compared with our optimized CABAC. Furthermore, the dissipations of static power 69.81 and 18.15 and optimized CABAC is 20.77 when compared with our optimized CABAC as we can see in Figure 4 that our model slightly outperforms than other existing models.

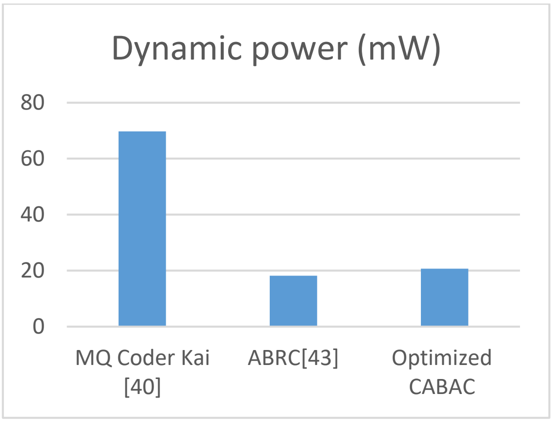

Figure 4. Comparison of Dynamic power

Total Power

The comparative analysis is based on the total power which needed to perform a task. For an efficient model, total power should be as lesser as possible. For the comparison analysis, we have considered two models i.e., ABRC and MQ-Coder kai and it is compared with our optimized CABAC. Furthermore, the dissipations of static power 694.49 and 640.7and optimized CABAC is 187.56 when compared with our optimized CABAC as we can see in Figure 5 that our model slightly outperforms than other existing models.

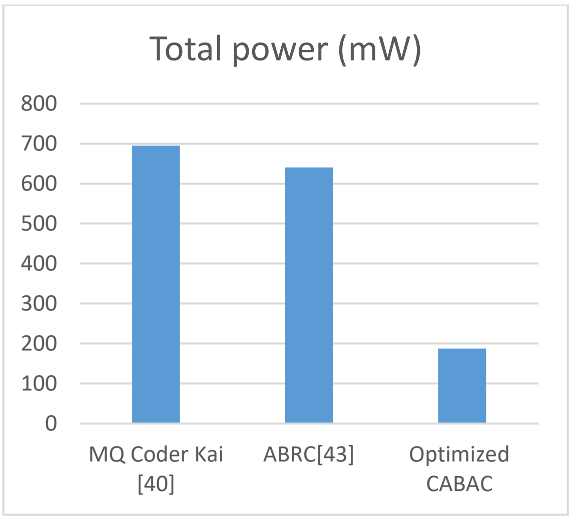

Figure 5. Comparison of Total power

5. CONCLUSIONS

This paper represented new architecture for CABAC EE (Entropy encoding). CABAC is main element in novel AVC/H.264 video standard as it provides 20% reduction of bit rate when compared to the entropy coder. CABAC is one type of CPU consuming application, which is suited to implement in the specialized hardware. A new architecture has been introduced that allows efficient and fast processing. Then, we propose optimized CABAC implementation of MPEG4/H-264 AVC against M-decoder without the PE (Probability Estimation). Our optimized CABAC obtains better results in terms of device utilization and power dissipation.

REFERENCES

[1] Marpe, D., Schwartz, H., Wiegand, T. (2003). Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. IEEE Transactions on Circuits and Systems for Video Technology, 13(7): 620-636. https://doi.org/10.1109/TCSVT.2003.815173

[2] Witten, H., Neal, R.M., Cleary, J.G. (1987). Arithmetic coding for data compression. Communications of the ACM, 30(6): 520-540.

[3] Pennebaker, W.B., Mitchel, J.L., Langdon, G.G., Arps. R.B. (1988). An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder. IBM Journal of Research and Development, 32(6): 717-726. https://doi.org/10.1147/rd.326.0717

[4] Saponara, S., Blanch, C., Denolf, K., Bormans, J. (2003). The JVT advanced video coding standard: Complexity and performance analysis on a tool-by-tool basis. In IEEE Packet Video.

[5] Marpe, D., Blattermann, G., Heising, G., Wiegand, T. (2001). Video compression using context-based arithmetic coding. Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205), 3: 558-561. https://doi.org/10.1109/ICIP.2001.958175

[6] Mrudula, S.T., Srinivasa Murthy, K.E., Giri Prasada, M.N. 2(019). M-ABRC (Adaptive Binary Range Coder) using Virtual Sliding Window technique and its VLSI implementation. Microprocessors and Microsystems, 71: 102901. https://doi.org/10.1016/j.micpro.2019.102901

[7] Rissanen, J. (1984). Universal coding information prediction and estimation. IEEE Trans. Inform. Theory, 30(4): 629-636. https://doi.org/10.1109/TIT.1984.1056936

[8] Bossen F. CABAC cleanup and complexity reduction. Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q. 2002 Oct;6. Taubman, D., Marcellin, M.W. (2002). JPEG2000 Image Compression: Fundamentals, Standards and Practice. Kluwer Academic Publishers.

[9] Dyer, M., Taubman, D., Nooshabadi, S., Gupta, A. (2006). Concurrency techniques for arithmetic coding in JPEG2000. IEEE Transactions on Circuits and Systems I: Regular Papers, 53(6): 1203-1213.

[10] Dvir, I., Allouche, A., Drezner, D., Ecker, A., Irony, D., Peterfreund, N., Yang, H.T., Zhou J.T. (2017). Trapezoidal block split using orthogonal C2 transforms for HEVC video coding. 017 25th European Signal Processing Conference (EUSIPCO), pp. 1016-1020. https://doi.org/10.23919/EUSIPCO.2017.8081361

[11] Liu, K., Zhou, Y., Li, Y.S., Ma, J.F. (2010). A high performance MQ encoder architecture in JPEG2000. Integration, 43(3): 305-317. https://doi.org/10.1016/j.vlsi.2010.01.001

[12] Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A. (2003). Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology, 13(7): 560-576. https://doi.org/10.1109/TCSVT.2003.815165

[13] Vasilache, A. (2017). Order adaptive Golomb rice coding for high variability sources. 2017 25th European Signal Processing Conference (EUSIPCO), pp. 1789-1793. https://doi.org/10.23919/EUSIPCO.2017.8081517

[14] Hou, J.H., Chau, L.P., Magnenat-Thalmann, N., He, Y. (2017). Sparse low-rank matrix approximation for data compression. IEEE Transactions on Circuits and Systems for Video Technology, 27(5): 1043-1054. https://doi.org/10.1109/TCSVT.2015.2513698

[15] Ding, J.R., Yang, J.F. (2007). Adaptive entropy coding with (5, 3) DWT for H.264 lossless image coding. TENCON 2007 - 2007 IEEE Region 10 Conference, pp. 1-4. https://doi.org/10.1109/TENCON.2007.4429056

[16] Said, A., Pearlman, W.A. (1996). A new fast and efficient image codec based on set partitioning in hierarchical trees. IEEE Transactions on Circuits and Systems for Video Technology, 6(3): 243-250. https://doi.org/10.1109/76.499834