Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard

Team Members:

Dharmendra Baruah

UIN 932003069

baruah.dharmendra@tamu.edu

Swarna Prabhu

UIN 932003046

swarnatind-abc88@tamu.edu

CONTENTS TABLE

TITLE	PAGE NO
Abstract	**3**
Introduction	**4**
CABAC Framework	**5-10**
Detailed Description of CABAC	**10-18**
Results & Analysis	**18-19**
Conclusion	**20**
Reference	**21**

ABSTRACT

One of the most popular entropy coding methods used in the ISO Standard for video encoding is CABAC. In entropy coding the primary idea is to use fixed tables of the Variable length codes where the transform coefficients computed from a block of a frame is scanned to represent in one D list which is later encoded using run length and variable coding. The paper represents the design for the context based adaptive Binary Arithmetic coding process that is a part of H.264 Video coding standard which ultimately resolves some of the drawbacks of the first arithmetic entropy coder for a hybrid block based video encoder. This is because CABAC defines an adaptive probability model where the context of the symbol changes. Efficiency and performance of CABAC is improvised by adopting a table based probability estimation technique , thereby removing the overhead due to multiplication. The report reviews how adaptive coding techniques along with context modeling can be used for efficient video compression in the AVC standards.

1. INTRODUCTION

Context-based adaptive binary arithmetic coding (CABAC) is a lossless compression algorithm that is widely used in video coding standards such as H.264/AVC and HEVC. CABAC has been most effective in achieving high compression rate compared to the previous standards MPEG -4. CABAC works by converting the input non binary symbols into a series of binary symbols and encoding each symbol using a probability model that is adapted dynamically based on the context of the symbol. The context of a symbol refers to the set of symbols that precede it in the data stream. By **using context-based modeling, CABAC is able to achieve higher compression rates than other coding methods that use probability models that are static.**

CABAC uses arithmetic coding to encode each binary symbol. **In arithmetic coding, each symbol is assigned a probability range that is proportional to its frequency of occurrence in the input data**. The probability range is then divided into smaller subranges for each possible symbol value, and the binary code for the symbol is selected based on which subrange it falls into.

Prior to the release and usage of H.264 video standard, Annex E of H.263 was used for encoding hybrid blocks and had several disadvantages. The probability distribution model in H.263 was not designed to get updated dynamically based on the statistics of the source data. Secondly, the arithmetic coder used in Annex E was m-ary which means it had to encode m symbols from the source and hence encode m-probabilities which introduces large computational complexity and memory overhead.
The partitioning of the source alphabet size into different ranges was introduced in H.26L standardization, significantly reducing the size of the input alphabet. This method was introduced in CABAC by the process of binarization where any given non binary symbol is converted into a series of binary based bins. Therefore motivation for the design of CABAC has been to overcome the drawbacks of previously used H.263 standard and use the efficient methods from H.26L.

The report is divided into 4 sections where the first section below gives an overall brief description of the CABAC process , followed by a section detailing each and every step. Experimental results , conclusions and references are included in the report that demonstrates the performance of CABAC ,with any future scope of improvement.

2. OVERVIEW OF CABAC DESIGN

CABAC Overview

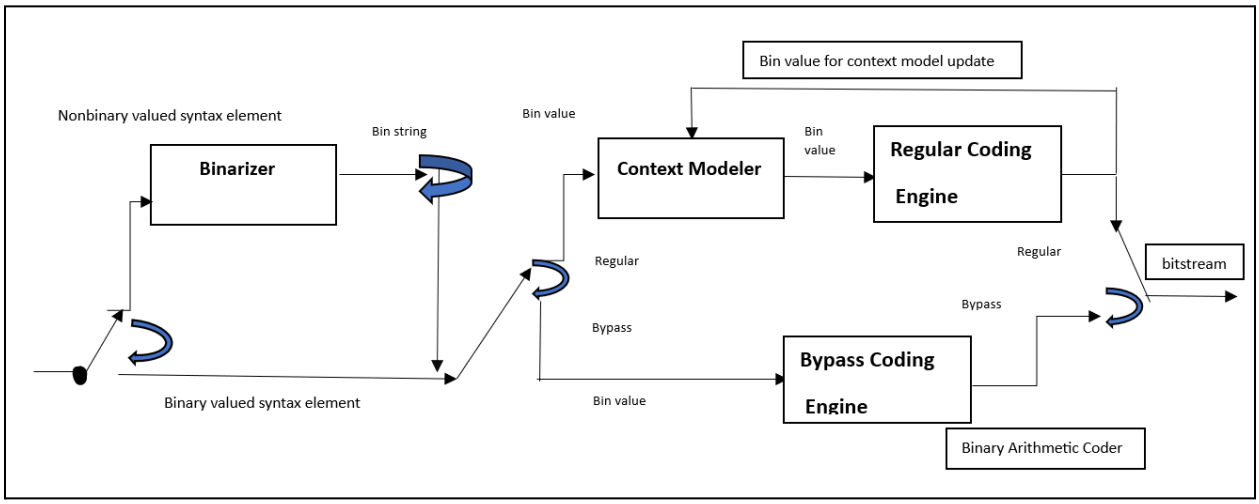

Fig. 1. Overview of CABAC process

The three major steps used in the design of CABAC is
A>Binarization process
B> Context Modeling process
C> Binary arithmetic encoder

2.1 Binarization

In the first step , any given symbol that is not binary based is converted to a symbol sequence containing only 0s and 1s forming a bin string. Binarization step is bypassed however if the given input symbol sequence is binary itself from the source level.

Next, a probability based model is created based on the context of the previous bins wrt to the current bin which is fed to the binary arithmetic encoder. Just like the first step, in order to get higher speed (sometimes in the current context if the probability of a symbol is 0.5 then for the next symbol will assume it is similar) some binarized bins may bypass the context modeling process.

The different ways and techniques in CABAC Binarization Schemes

1. ***Unary and Truncated Unary Binarization Scheme***

In CABAC, unary and truncated unary binarization schemes are used to represent unsigned integer-valued symbols.For a particular symbol y then unary binarization is given by y number of 1s and then we end it with zero. For example, the unary code word for y = 4 would be

"11110".The truncated unary code is only defined for y where y belongs to the range y <=S AND y >= 0 where S is a constant. TU code will be similar to unary code until y is less than S. Later, for y = S, 0 bit in the end is ignored and only 1s are retained in the binary codeword. so the TU code for x = S is given by a code word consisting of S "1" bits. For example, if S = 5, then the TU code word for x = 5 would be "11111".
The major advantage of the TU code over the unary code is that it requires fewer bits to represent larger values of x, since the "0" bit terminating the code is not needed for values of x up to S. This reduces the overall number of bits needed for encoding the sequence.

2. kth order Exp-Golomb Binarization Scheme

The kth order Exp-Golomb Binarization Scheme is a type of prefix-free coding scheme used for encoding integers. It is based on the Golomb coding scheme, which is optimal for geometrically distributed sources. The kth order Exp-Golomb code word for a given unsigned symbol y which is integer can be developed by adding a prefix part to a suffix part. The prefix part of the scheme is nothing but the unary code for the value l(x) = log (x / 2^k +1), which is the quotient of x divided by 2, plus one[1].
As far as the suffix part is concerned for k-th order Exp-Golomb it is floor(log2(x/ 2^k + 1)). The suffix part is computed as the binary representation of x + 2^k - 2^floor(log2(x / 2^k + 1)) using k + floor(log2(x/ 2^k + 1)) significant bits[1].
For example, let's say we want to encode the value z= 5 using the 1st order Exp-Golomb binarization scheme (k = 1). The prefix part of the code word would be floor(log2(5 / 2^1 + 1)) = floor(log2(3)) = 1, so the prefix would be 10. The suffix part would be the binary representation of 5 + 2^1 -2^floor(log2(5 / 2^1 + 1)) = 5 + 2 - 2 = 5, using 1 + floor(log2(5 / 2^1 + 1)) = 1 + floor(log2(3)) = 2 significant bits, so the suffix would be 01. Therefore, the complete bin string for x = 5 using the 1st order Exp-Golomb binarization scheme would be 1001.

The advantage of truncating 2 binarization schemes such as the kth order Exp-Golomb Binarization Scheme is highly beneficial when we want to encode residual data that can have transform coefficients. The value of k is typically set to 1 or 2, and is determined based on the probability distribution of the syntax element values[1]

3. Fixed-length Binarization scheme:

The Fixed-Length (FL) Binarization Scheme is a method of converting syntax elements into a binary representation of a fixed length. In this scheme, it is assumed that the syntax element has a finite alphabet of values, and the value of the syntax element denoted by y, is in the range $0 \leq y < S$. Here, S is the size of the alphabet.
The binary representation of symbol y using l = log2(S) bits is the binarized value for a symbol y in FL scheme. That is, each value of y is represented by a binary string of l bits.

Hence in CABAC , based on the source statistics we can accordingly select the appropriate binarization scheme and therefore Fixed length is well suited for binarizing those probability distributions that are constant and uniform[1]

4. ***Concatenation of Basic Binarization Schemes:***

In order to have better efficiency in binarization , CABAC introduces a method where the given syntax element or symbol can be binarized effectively by combining different binarization schemes like Unary, Truncated Unary, kth order Exp-Golomb etc.

In the paper [1] it is mentioned that the addition of binarization schemes of the Truncated Unary (TU) and the k-th order Exp-Golomb (EGk) is beneficial for encoding motion vectors that is derived for current frame using a reference frame and also for coding the absolute values of transform coefficient levels (residual data).

The major advantage of using the unary code in concatenated binarization technique is the large number of strings of 1s can be modeled quickly and easily since the context is going to be uniform. Generally Exponential Golomb code is more suited to encode for larger values. Hence for those symbols encoded using unary as prefix and EGk as suffix part improves the speed as the unary prefix bins can be bypassed while only suffix bins enter the regular mode for CABAC context modeling.

2.2 Context Modeling

Context Modeling uses the binarized bins to create a conditional probability model distribution based on the stream of symbol bins and the context wrt to a particular symbol from the source. The main motivation behind the use of Context Modeling in CABAC is to make use of the statistical dependencies and ensure that any changes to the model are automatically updated.

While encoding a given symbol x, a conditional probability -P(x| F(z)) is defined which defines the probability that we encode x given a function F that maps T -> C where T is the set of previous symbols and C is a related set C={0,1…C-1} of contexts[1]. Once x is encoded using the estimated conditional probability p(x|F(z)), the probability model is updated with the value of the encoded symbol x. Thus, p(x|F(z)) is estimated on the go by tracking the actual source statistics. Due to the model cost, increase in context set C ,can lead to overfitting of the model if we consider a large number of neighboring symbols as context where computation of conditional probabilities can go incorrect[1].

This problem is resolved by ensuring that only few binarized bins enter the context modeling stage as few encoded bins can be bypassed to binary arithmetic coder hence reducing the

complexity and computation overhead. Other way to take care of this issue is to ensure only a small predefined set of past symbols(neighboring symbols) are taken into consideration for

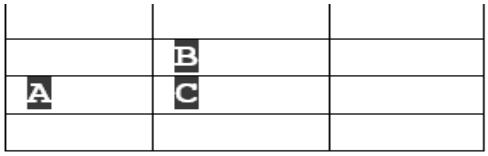

Fig. 2. Current symbol C with A and B as context template [1]

modeling so that the size of the context set C is drastically reduced. In CABAC there are majorly 4 different approaches to context modeling and these are briefly explained in the below section. In the first approach for a given symbol x to be encoded the past 2 neighbor symbols are taken into consideration as shown in Fig 2 and these symbols' context for choosing which type of neighbors can be from any syntax element defined in the below table taken from [1]

Syntax element	Slice type		
	SI/I	P/SP	B
mb_type	0/3-10	14-20	27-35
mb_skip_flag		11-13	24-26
sub_mb_type		21-23	36-39
mvd (horizontal)		40-46	40-46
mvd (vertical)		47-53	47-53
ref_idx		54-59	54-59
mb_qp_delta	60-63	60-63	60-63
intra_chroma_pred_mode	64-67	64-67	64-67
prev_intra4x4_pred_mode_flag	68	68	68
rem_intra4x4_pred_mode	69	69	69
mb_field_decoding_flag	70-72	70-72	70-72
coded_block_pattern	73-84	73-84	73-84
coded_block_flag	85-104	85-104	85-104
significant_coeff_flag	105-165, 277-337	105-165, 277-337	105-165, 277-337
last_significant_coeff_flag	166-226, 338-398	166-226, 338-398	166-226, 338-398
coeff_abs_level_minus1	227-275	227-275	227-275
end_of_slice_flag	276	276	276

TABLE I - REPRESENTATION OF THE INDEX FOR VARIOUS SYNTAX ELEMENTS [1]

Here, syntax elements in the table refers to the different components of a video based coding syntax that describes the video content. This can be any parameter that is used for encoding such as macroblock , motion vectors, coefficients of the transform matrices etc.

In the second approach to CM, the model is selected based on previously coded b0, b1,b2…bi-1 bins for a given bin bi with ith index and this is used only for mb_type and ,b_sub_type syntax element. The third and fourth approach is more suitable for residual data(difference between the original frame and predicted) where we consider the position during the scanning and the total number of encoded levels with a specific value before the current encoded symbol respectively[1].

CABAC keeps track of different syntax elements that are used in video encoding H.264 standard and allocates a specific unique index referred to as context index γ as shown in the above table

I.Each element is further indexed depending upon the slice type of the frame - Independent slice, Predictive slice and Bi directional Slice. Using context index γ the number of neighboring symbol bins b0….bi-1 is decided based on the context index value and hence the probability model is designed in a way depending on which category the symbol belongs to an efficient context index γ is used so that compression efficiency is high overall.From Table I it is evident a pair of values - a 6 bit probability index σγ and a binary value of the most probable symbol which is used to represent the probability is mainly decided from the context index.

In this section the context indices and their range from table I is briefly described. Context index lambda which belongs to range 0 to 72 is calculated using the formula $\gamma = \Gamma + \chi s$ where χs is the context index increment for a given element S and Γ denoting the context index offset[1]. This range is typically used to address prediction modes (spatial and temporal) , submacroblock , macroblock etc. The next context indices range of interest is 73- 398 which denotes the syntax elements -coded block pattern , coded block flag, significant coefficient flag and last significant coefficient flag . Table I also shows 2 rows of indices for coefficient flag syntax elements based on the direction in which scanning is performed for frame or field coded macroblock. Here the lower indices represent the field based coding mode.

The formula for context index γ for residual syntax elements (Excluding coded block pattern) is given by $\gamma = \Gamma s + \Delta$ ctx cat $+ \chi s$ where Γs denotes the context index offset and Δ ctx cat denotes the offset on the context category chosen based on delta as shown in table[1] . Therefore an overview of Context Modeling section gives how the context index increment is calculated for various syntax elements based on the context template that is added to offset to give the overall index from which the probability model distribution table is selected during encoding.

2.3 Binary Arithmetic Coding

In the regular way of implementing binary arithmetic coding, we encode 2 symbols namely Least probable symbol(LPS) and Most probable symbol(MPS) using successive subdivision of the interval based on the incoming symbol. Let R represent the range and L the lower limit of the LPS, then R lps = R * pLPS where pLPS denotes the probability of the LPS. The Range for the MPS is then easily calculated using R - R lps = R mps. The final range after successive interval division along with the specified precision bits will give the encoded bitstream from the encoder. The only disadvantage of this computation is that at each stage multiplication operation has to be performed which can cause overhead and memory issues depending on the source to be encoded. CABAC solves this problem by adopting a modulo coder (M coder) where multiplication operation is completely removed by using a table that has probability precomputed for different Quantized interval Range R and state index. Hence a given range R is divided into k levels Q0, Q1…Qk-1 and probability is categorized as P0, P1 ..PN-1 where for every state index σ and the product of this is calculated in prior and stored in the 2D table. Once the context modeling and

the index for a syntax element is found, the corresponding probability distribution is used for encoding of the source symbols. In this way the binary arithmetic process is speeded up to give efficient correct results with less overhead and less memory burden on hardware.
Further speedup is guaranteed by skipping the binary arithmetic coding for symbols that seem to have the same probability distribution. The detailed description for the implementation of table based binary arithmetic is given in the next section.

3. DETAILED DESCRIPTION OF CABAC

3.1 Coding for Macroblock Type, Prediction Mode and Control Information

In order to evaluate coding of blocks which are macroblock and sub macroblock type, the syntax element mb_skip_flag type is used which denotes if the current macroblock (Backward or predictive slice) is skipped. If the skip flag value is zero it is not skipped and if the value is set to 1 it is skipped. In case if macroblock is not skipped then sub_mb_type syntax is used to denote for the sub macroblock of size 8 * 8 for a predictive or backward slice. The Context index increment for macroblock skip flag denoted by χMbSkip context taken for current symbol C is based on skip flag values of left neighbor A and top neighbor B[1].
If any of the two neighboring macroblocks A or B or both are not present (e.g. because they are outside of the current slice), the corresponding mb_skip_flag(X) value is kept to 0 (not skipped). The binarization for mb and sub-mb type is done using a binary decision tree where the terminal node represents the different symbol values for a given syntax element and binary string while traversing through the tree from root to terminal node gives the bin string associated with each symbol.
In this section ,Coding of Prediction mode is explained where the difference between current and reference frame used as prediction error is coded and context index is calculated for the prediction.
Intra Prediction Modes for Luma 4×4- Coding of this luminance block is based on syntax element -prev_intra4x4_pred_mode_flag and the mode indicator rem_intra4x4_pred_mode…If prev intra 4*4 pred_mode_flag is zero then consider rem_intra4*4 pred_mode. A separate probability model is used for coding the flag value and coding the binary bins of rem_intra4*4_pred_mode element[1].
Intra Prediction Modes for Chrominance 4×4 is given by the below formula where ChPredInDcMode is a binary value that denotes the prediction type is DC mode where we take the average of the top and left pixel for prediction. Here the context chosen is based on the DcMode flag for neighbors A and B.
Therefore when syntax element IntraChromaPred mode is encoded , it is initially binarized using the TU method and the flag value ChPredInDcMode is encoded in the initial bin. Therefore the 3 probability model generated from the above formula is used to get the probability of only the first bin of the syntax element.

The coding of the syntax element which belongs to a reference frame or block is mentioned below.In order to code the first bin of ref_index type syntax element we refer to the ref_index value of neighbors A and B by using a binary value flag named RefIdxZeroFlag.
Here the first bin after binarization of syntax containing reference index element is coded using unary coding and remaining bins using 2 additional probability models.
Motion vectors are derived for video coding the current frame with respect to the reference frame. Here mvd motion vector difference component is used which is a value that denotes the motion vector component in a horizontal or vertical direction for a specific macro or submacro block X . Context index increment for partition C is derived using mvd values for macroblock or submacroblock for neighbors A and B using the formula:

$$\chi_{Mvd}(C,cmp) = \begin{cases} 0, \text{if } e(A,B,cmp) < 3 \\ 1, \text{if } 3 \le e(A,B,cmp) \le 32 \\ 2, \text{if } e(A,B,cmp) > 32 \end{cases}$$
$$\text{with } e(A,B,cmp) = |mvd(A,cmp)| + |mvd(B,cmp)|,$$

Fig. 3. mvd context index increment formula [1]

This is mainly used to get the probability model for the first bin value of the mvd which is derived from UEG3 binarization technique. Remaining bins are encoded by different method including exp Golomb code Macroblock and submacroblock contain syntax elements named - y mb_qp_delta, end_of_slice_flag, and mb_field_decoding_flag which are also referred to as Control Information that denotes some information for a given block such as flag values, mode values etc.Mb_qB_delta: this syntax element is mainly used for macroblock which has coded block pattern obtained from quantization. The syntax element δ(Z) is coded by binarization of δ+ (Z) using unary binarization where δ+ (Z) = 2| δ(Z) | − ((δ(Z) > 0) : 1 ? 0). For encoding the first bin we see if the δ(X) !=0 where X is the previous macroblock of Z . Hence 2 models are used for encoding the first bin depending on the value of δ(P). End of slice flag : is encoded using a special probability mode which is not dynamic and is fixed since the end of slice is used to mark the end of the current slice being processed.
Macroblock pair field or frame flag is similarly coded using the mb_field_decoding flag from neighbors A and B.

3.2 Coding of Residual Information

3.2.1 *Encoding process for the Residual information*

The CABAC encoding process for residual data in H.264/AVC involves several steps, as illustrated in Fig. 5, which deals with a single block of transform coefficients. The steps in the encoding process include:

1. Coded Block Pattern: The coded_block_pattern symbol is used to signal which of the six 8x8 blocks (4 for luminance and 2 for chrominance) contains significant or non-zero transform coefficients. For binarization , it uses concatenation of Fixed length for 4 bit and Truncated Unary

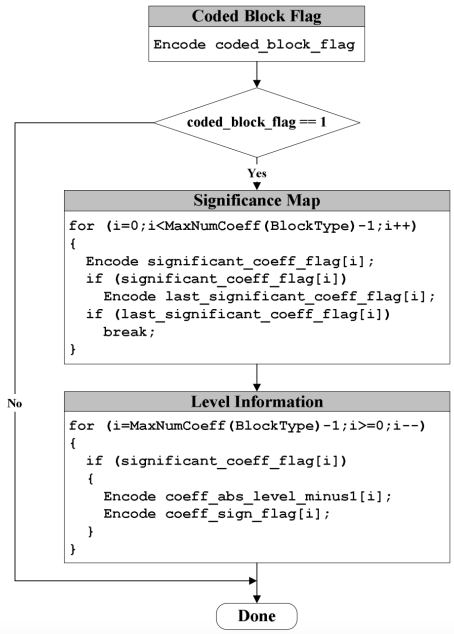

Fig. 4. Flow diagram of the CABAC encoding scheme for a block of transform coefficients. [1]

Binarization, considering 2 as the cut off value.

Scanning position	1	2	3	4	5	6	7	8	9
Transf. coefficient levels	9	0	-5	3	0	0	-1	0	1
significant_coeff_flag	1	0	1	1	0	0	1	0	1
last_significant_coeff_flag	0		0	0			0		1

TABLE II EXAMPLE FOR ENCODING THE SIGNIFICANCE MAP [1]

2. Coded Block Flag: Similar to the coded block pattern, this flag signals if any significant transform coefficients are present inside transform coefficients based block.
3. Scanning of Transform Coefficients:Zig zag scanning is used to convert the two dimensional block of significant non zero coefficients into a single one dimensional list .

4. Significance Map Encoding: Here two flags named significant_coefficient flag and last significant coefficient flag are used that represent the starting point from where non zero coefficient is present and the last position where non zero coefficient is present respectively. Table III in the reference provides an example of the significance map encoding procedure[1]
5. Encoding of Coefficient Levels: When the absolute magnitude and sign is encoded it is scanned in reverse order and this is referred to as the encoding of the coefficient levels

Level Information:

Level information encoding represents the values and signs of the non-zero transform coefficients which are quantized inside a block . This information is encoded using in the below two coding symbols:

1. coeff_abs_level_minus1: This symbol is obtained by taking the absolute value of the coefficient and subtracting 1. The UEG0 (Unary Exponential Golomb with k = 0) binarization scheme, as shown in Table I of the reference, is used to encode the values of coeff_abs_level_minus1. This scheme provides efficient encoding for a wide range of level values.
2. coeff_sign_flag: This is a single bit symbol where the sign of the levels is given, with a value of 1 for negative coefficients and 0 for positive coefficients.

BlockType	MaxNumCoeff	Context category (*ctx_cat*)
Luma DC block for Intra16x16	16	Luma-Intra16-DC: 0
Luma AC block for Intra16x16	15	Luma-Intra16-AC: 1
Luma block for Intra 4x4	16	Luma-4x4: 2
Luma block for Inter	16	
U-Chroma DC block for Intra	4	Chroma-DC: 3
V-Chroma DC block for Intra	4	
U-Chroma DC block for Inter	4	
V-Chroma DC block for Inter	4	
U-Chroma AC block for Intra	15	Chroma-AC: 4
V-Chroma AC block for Intra	15	
U-Chroma AC block for Inter	15	
V-Chroma AC block for Inter	15	

TABLE III BASIC BLOCK TYPES WITH NUMBER OF COEFFICIENTS AND ASSOCIATED CONTEXT CATEGORIES [1]

The levels are transmitted in reverse scanning order, starting with the last significant coefficient of the block. This approach allows for the use of context models that are adapted based on the previously transmitted coefficients. By adapting the context models based on the statistical dependencies among the coefficients, the CABAC encoding process can achieve better compression efficiency[1].

3.2.2 *Context Modeling for the Residual Data*

There are a total of 12 transform coefficients that are used in encoding the residual data but since this increases the complexity of the context modeling since every one of 12 will have to be referenced to

different probability model in CABAC , it is categorized into 5 categories, as shown in the right column of Table IV.

For each of these categories, a specific set of context models is used for all syntax elements related to residual data, with the exception of coded_block_pattern. This classification approach helps to efficiently manage the context models and adapt them according to the statistical properties of different block types, ultimately improving the compression performance of the H.264/AVC encoding process[1].

For coded_block_pattern, probability models depend on the bin index. Separate context assignment rules exist for luminance and chrominance bins, which result in a total of 8 additional probability models.

Coded block flag is encoded by considering the top and the left neighbor's block value for the flag.Since there are total 5 categories each category is further coded using 4 different probability models.Hence overall 20 different models are used for the coded_block_flag bit.

Scanning position based context modeling is used for encoding flags such as significant_coeff_flag and the last_significant_coeff thereby using a total probability model upto 61.

Level information

Level information is encoded using 2 parameters namely the NumT1 which denotes the total number of encoded trailing 1's and NumLgt1which denotes the total number of levels with value higher than 1. Initially the 2 above parameters are kept 0 at the start of the reverse scanning process.For encoding the first bin, the context index increment χAbsLFirstB(i) at scanning position i is determined as follows:

$$\chi\text{AbsLFirstB}(i) = \begin{cases} 4, & \text{if NumLgt1}(i) > 0 \\ \max(3, \text{NumT1}(i)), & \text{otherwise} \end{cases} \quad [1]$$

1 to 13 bin indices are encoded , using the formula below where the context index increment is calculated using NumLgt1:

$$\chi\text{AbsLRemB}(i) = \min(4, \text{NumLgt1}(i))$$

It is observed that the total number of different probability models for encoding the level information is 49[1].

3.3 Probability estimation:

In CABAC, the probability estimation for a given symbol is efficiently designed to avoid the multiplication process between interval range and symbol probability as seen in binary arithmetic encoder. To reduce the computational overhead, CABAC utilizes a predefined table that represents the probability of Least Probable Symbol (LPS) using 64 levels(index) and its

corresponding Most Probable Symbol. The formula used to derive the probability of each index is as below in Fig 5:For example the probability for index 3, is given by P3 = α P2 where α is the scaling factor. Here each LPS probability, is represented by a MPS ϖ which is either a 0 or 1 ,therefore we can have a total of 128 probability states for the estimation. Since the index number 63 is used for encoding the last string a total of 126 states are effectively utilized, each represented using 7 bits[1].

$$p_\sigma = \alpha \cdot p_{\sigma-1} \quad \text{for all } \sigma = 1,\ldots,63$$
$$\text{with } \alpha = \left(\frac{0.01875}{0.5}\right)^{1/63} \text{ and } p_0 = 0.5.$$

Probability calculation based on state index [1]

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

Updation of probability [1]

The adaptive design in CABAC , ensures that the probability model can be updated based on the probability and value of the most recently encoded symbol. Hence the current state index and the value of the previously encoded symbol is significant. After every update the probability of the least probable symbol keeps changing.

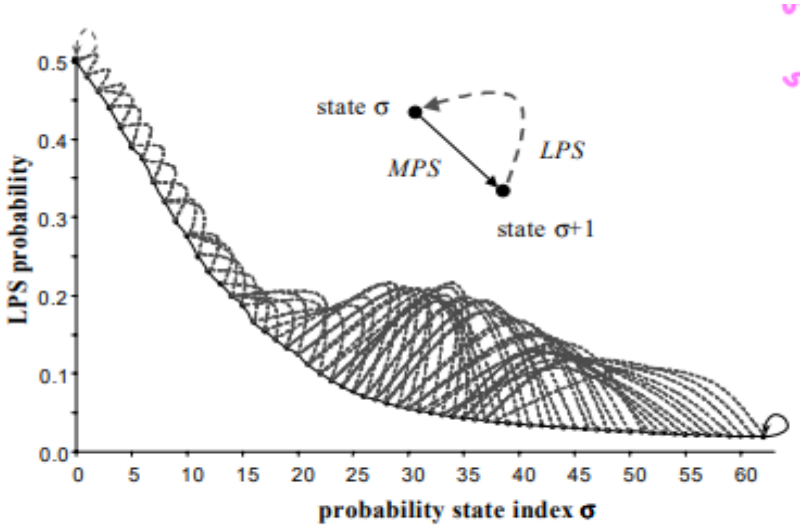

Fig. 5. LPS Probability vs state index [1]

The above figure 5 illustrates the values of the estimates for the least probable symbol for different state indexes. While encoding if the next occurring symbol is a Most probable symbol, we simply increment the index to 1 since a higher index represents higher probability values. If the index reaches the last state 62, only when an LPS symbol is seen the index goes back down to a lower index value by a specific value ,else if MPS is seen we retain the same index state. In the paper it is assumed that only for the first index where σ = 0 at the start , the index will not be

changed; however only the value of the LPS and MPS will be exchanged. The new probability estimate is calculated based on the current probability (p old) and α[1].

In this section, Initialization and reset of probability states, for video coding where information is encoded using slice, at every slice boundary the probability model cannot be reused since it can represent wrong/outdated information on the context. Hence this calls for the need to know some prior information before we start encoding a first frame every time. CABAC uses a technique to pre-initialize the probability model by consuming the statistical information of the source data which is called the initialization process at 2 levels.

The first level is based on the Quantization Parameter of the slice. In video encoding every frame or slice before its encoded will go through a quantization process and where the quantization step is for every frame is represented by Slice QP parameter[1]. Given the initial Slice QP parameter, regression is used to derive parameters (μγ, νγ) for a given model with context index γ which is then plugged into the formula in Fig 5. below to get pre state index σpre.

1. σ_{pre} =
   max(1, min(126, ((μ_γ * SliceQP) >> 4) + ν_γ))

2. if(σ_{pre} <= 63) {
   σ = 63 - σ_{pre}
   ϖ = 0
   } else {
   σ = σ_{pre} - 64
   ϖ = 1
   }

Initialization of state index from Slice QP parameter [1]

This way for every new slice, state index and most probable symbol is obtained in CABAC.

Slice dependent initialization: In order to further improve the video coding efficiency the initialization of the probability index is done based on the slice type if it is P or B type since every slice type will have its own statistical information. This means the encoder has to read the slice header to get the information(2 bit used to indicate which out of 3 context tables are used) to fetch the corresponding initialization table , and then repeat the process of using the Slice QP parameter to arrive at the probability state index and symbol for every new slice. The major downside of this process is memory overhead since the entire process is repeated for every slice[1].

Table Based Binary Arithmetic Coding: Binary Arithmetic coding employed in H.264/AVC is implemented in two ways -first way being the regular mode which is used to encode most of the symbols and second is bypassed mode which is particularly used to encode the symbols that have

equal probability of occurrence. The overall process of the BA in CABAC primarily consists of interval subdivision without multiplication , bypass coding mode, renormalization and the termination of the codeword.

A. Division of the interval to subinterval

In order to encode a binValue which we get after binarization , context modeling with probability state index σ (and value of MPS ϖ) , the first step is division of the current interval range R into sub interval which should always be in the range of 8-9 bit precision. In CABAC , range R is quantized into 4 different levels and the range chosen is based on a quantization index ρ, which is computed by right shifting current range R as given below:
Hence the index ρ and probability state index σ is used to perform a lookup on a predefined table TabRangeLPS (which has 64*4 distinct values) to fetch the next range of the LPS. The range for MPS is simply R- Rlps. As shown in the below flow diagram if the binVal is a MPS then state index is calculated for the next state. If its a LPS ,range R is updated to Rlps and Lower limit L is updated to L +R where R is R - Rlps (range for MPS). Then the state index σ is checked where if its zero (Starting state) we toggle the MSB else the index for next state is calculated.

B. Bypass mode:

Bypass mode used for equi probable symbols will have the Range R same for MPS/LPS. Here the interval length R and lower limit L is doubled instead of dividing to smaller subintervals. This reduces the computational overhead of doubling L and R again at the renormalization stage. In the paper, this method is compared to directly sending the codeword to bitstream without any encoding or context modeling and hence is also referred to as “lazy coding mode”

C. Renormalization and Carry Control:

Renormalization is a process used to ensure that the total range during interval subdivision does not exceed the required precision of 8- 9 bits since a low difference in range limit can result in inefficient coding and loss of information. This is avoided by increasing the range R and the lower limit L by a factor of 2[1]. This is done by propagating the extra bits of the bitstream out to ensure the total precision is in the required range and during this if any carry is generated , it is resolved.

Termination Codeword: The last index of the TabRangeLPS table is fixed for indicating the terminating symbol and hence the 7 bit in the table is reserved for this purpose. Therefore the ending syntax for any block or slice is usually encoded using the fixed value of Rlps. The renormalization is designed in such way that when it encounters this syntax the decoder stops decoding the next bits to indicate the termination.

4. RESULTS AND ANALYSIS

An experiment was done to find the efficiency of the working of the CABAC model for video encoding of the set of sequences of frames from a TV broadcast using the AVC standard. This experiment involved applying 2 B frames between every 2 Independent frames (which is inserted as IDR) in a gap of 500 milliseconds. Langragian coder control is used to derive the parameters for motion search , macroblock and picture based decisions. Quantization parameter Qp is used to design the bit rate for encoding of a complete sequence.

The experiment results showed that the average value for CABAC shows significant bit rate savings of 9 -14 percent compared to the entropy coding method for a SNR range of 30- 38 dB. In our analysis, in order to evaluate the performance of H.264 Video standard , a python script was written that fetches the codec library for H264 -CABAC and CAVLC standards and uses that with defined quality parameter (bit rate) and quality value to encode , decode the given video input. The function also calculates the compressed file size and ratio. Bit rate and the quality value has been used to calculate the PSNR and bit rate. The result for the experiment has given below in Fig9 and Fig 10

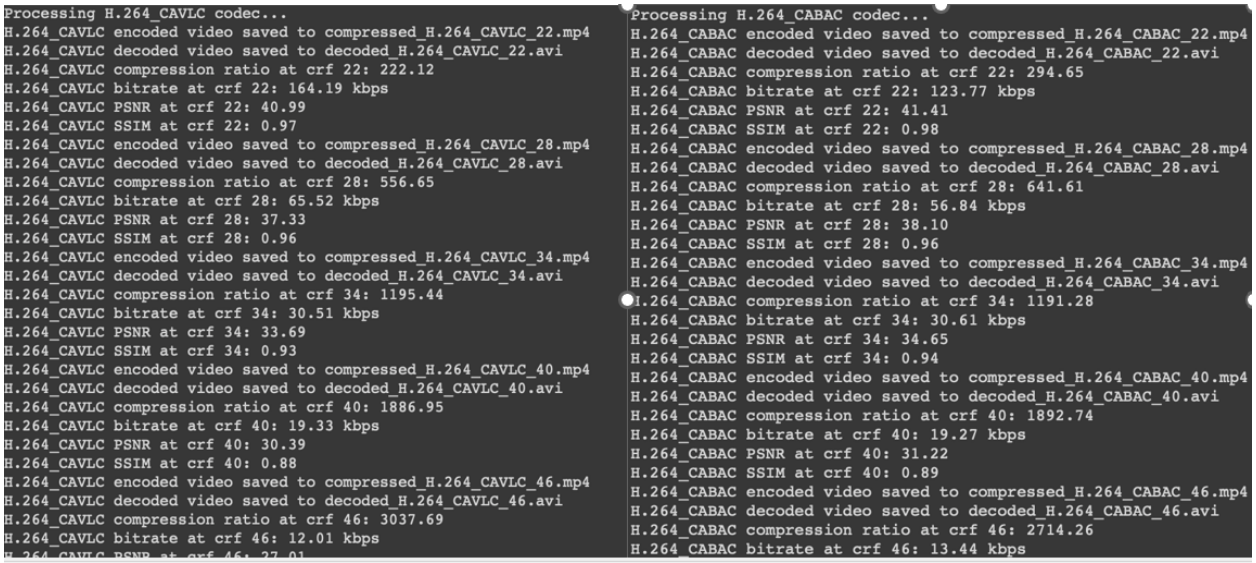

Fig. 6. Results for PSNR and SSIM for H264 CABAC and CAVLC that shows the performance comparison

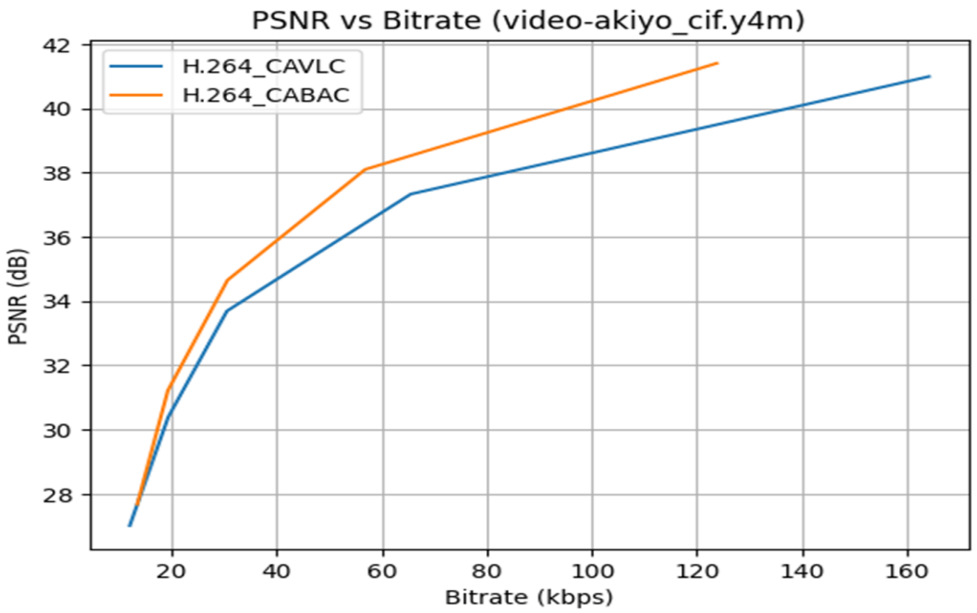

Fig. 7. Graph showing the PSNR(in db) vs bitrate for CABAC vs CAVLC.

From the above result it is clear that CABAC outperforms the CAVLC and hence is the most efficient standard for video compression.

5. CONCLUSION

The paper gives a detailed outline of the Context Adaptive Binary Arithmetic Encoding Process employed in the current H.264 video coding standard that guarantees high compression efficiency and efficiency due to the fact that the probability models are dynamically updated based on the previously coded symbols. Compared to the previous standards , multiplication operation in binary arithmetic has been removed by adopting table based arithmetic coding that decreased the computational complexity and cost overall. Another advantage is that this process simplifies encoding by using binary arithmetic with the help of a binarizer that converts any non binary syntax element to binary form.

H264 from the time it has been developed and published has been used in a wide variety of applications like video conferencing, streaming and broadcasting.

6. REFERENCES

[1] Detlev Marpe(Ed), " Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. X, NO. Y, MONTH 2003

[2] T. Wiegand (Ed.), "Draft ITU-T Recommendation H.264 and Draft ISO/IEC 14496-10 AVC," Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVT-G050, Pattaya, Thailand, March 2003.