

IMPLEMENTATION OF CONTEXT ADAPTIVE VARIABLE LENGTH CODING AND DECODING ALGORITHM FOR H.264 VIDEO CODEC USING MATLAB

Harisha G C¹, Dr. Murugendrappa N¹

Manjushree B S², Kunjana C M², Lakshmi Sai C², Mahima B L²

¹Assistant Professor, Dept. of ECE, G M Institute of Technology, Davanagere.

²Students, Dept. of ECE, G M Institute of Technology, Davanagere,
harisha0710@gmail.com, murugiele@gmail.com

ABSTRACT

This paper proposes the implementation of Context Adaptive Variable Length Coder for H.264 video encoder. Context-Adaptive Variable Length Coding, a specially designed method of coding the transform coefficients, in which different sets of variable-length codes are chosen depending on the statistics of recently-coded coefficients, using context adaptation. Context Adaptive Variable Length Coding is used to encode residual, scan ordered blocks of transform coefficients. The quantized coefficient inputs to the context adaptive variable length coder is generated by the integer transform and quantization processor. The implementation is capable of bringing about compression of video sequences and is capable of processing high resolution pictures of sizes of up to 1024 x 768 pixels, encoding at a real time frame rate of 25 fps. The compression achieved by the implementation is over 10 and the reconstructed picture quality is better than 35 dB.

KEYWORDS: Context Adaptive Variable Length Coding (CAVLC) and Decoding (CAVLD), Zig-zag scan, H.264/AVC.

I. INTRODUCTION

To improve the performance of the existing applications and to enable the applicability of video compression to new real-time applications, a new international standard for video compression is developed. This offers significantly better video compression efficiency than previous video compression standards. It is developed with the collaboration of ITU and ISO standardization Organizations, called with two names, H.264 and MPEG4 Part 10. The video compression efficiency achieved in H.264 standard is not a result of any single feature but rather a combination of a number of encoding tools [1]. One of these tools is the Context Adaptive Variable Length Coding (CAVLC) algorithm used in the baseline profile of H.264 standard [3]. This provides better entropy coding performance in comparison to an entropy coding algorithm using a single VLC table, and improves the entropy coding performance by using coding techniques such as run-level and trailing ones coding, that are designed to take advantage of the characteristics of the 4x4 blocks of transformed and quantized residual provide the major thrust for this new standard. CAVLC module, which assigns variable length codes to get the desired data. The consensus among the major players of the communications and video industry on H.264, might compress the bit stream.

The quantized coefficients are a 4x4 block of data which is scanned in a zig-zag manner and to exploit the number of zeros, a zig-zag scan of the matrix is used. Zig-zag scan allows all the DC coefficients and lower frequency AC coefficients to be scanned first. The DC coefficients are encoded using differential encoding and AC coefficients are encoded using run-length encoding and then a 4x4 block of data, which is scanned in a zig-zag manner, is then fed to the Golomb or CAVLC processor. The CAVLC decoding is faster than the conventional one with less complexity and enhances the

performance and saves the time required to decode the desired value from the lengthier bit stream. It has more bit reduction than to the Golomb. CPU execution time is more for CAVLC then compared to Golomb. CAVLC coding scheme itself determines that it is context adaptive in nature. [2]

The basic architecture of CAVLC is that has one degree of parallelism in terms of reading one residue or coding one symbol. When one MB starts to be processed, the MB information is translated into code words by the Exp-Golomb coding unit. Afterwards, the quantized transform residues are coded by the CAVLC unit. The MB is divided into 4x4 blocks, and the 4*4 blocks are processed one after another in the defined order. Each 4x4 block is processed through two phases, the scanning phase and the coding phase. In the scanning phase, the residues are read from the residue buffer in the backward zig-zag order. The selection of VLC tables within a class is according to the related statistics and the previously transmitted symbols. Different from the traditional fixed VLC tables, CAVLC utilizes the inter symbol correlation to further reduce the statistical redundancy. The proposed entropy coding engine with dual-block pipelined architecture, zero skipping technique and a 2-Kb bit stream buffer is implemented by using cell-based design flow and 0.18-UMC/Artisan cell library. Three types of memories are required. The coefficient memory and bit stream memory are used as input and output buffers for system consideration. The upper 4x4 block total coefficient memory is used to store the 4x4 block total coefficients required by following blocks. The entropy coding engine requires about 500 cycles for high-quality applications (QP=10–20) and about 200 cycles for low-bit-rate applications (QP=30–45). [3]

Based on H.264 entropy decoding algorithm and measured complexity profiles, we propose complexity models that relate the complexity of each particular coding mode (I, P, or B, BAC or VLC) with the bit rate. Using these models, one can accurately derive the complexity for an arbitrary GOP structure, using either BAC or VLC. The current model requires separate model parameters for different coding mode. It is needed to explore how to simplify the proposed models so that fewer parameters are required. We will also investigate the relation between model parameters and video content [4].

In [5], authors analyze the computational complexity of software based H.264/AVC [1] baseline profile decoder by its decoding sub functions, and estimate the time complexity on DSP and general purpose computer via the frequency of use of decoding sub functions. CABAC is adopted in H.264/AVC and inherited in SVC as an alternative variable length coding method, which is more efficient, but also more complex than CAVLC. CABAC not only converts symbols to a binary code (the same as VLC, denoted as binarization), but also encode the resulting binary bits using arithmetic coding based on its “content model” or “probability model”. As shown in Fig.3, in CABAC (content-adaptive binary arithmetic decoding), the context variables and decoding engine are initialized, followed by parsing the input bits. These captured bits are interpreted based on the context model and inverse decoded, to yield syntax symbols [6]. The framework of H.264/AVC still belongs to block-based motion-compensated transform coding similar to previous standards.

The better compression performance mainly comes from the prediction and entropy coding tools [7]. The new features cause not only much higher computational complexity but also have a great impact on the traditional architectures for low-cost and high-performance considerations [8-9]. Entropy coding involves many bit-level operations that cannot be efficiently executed by general purpose processors. Some hardware accelerating solutions have been proposed for the decoding part. The directly unfolded CAVLC engine results in large area and long critical path [10]. The video compression efficiency achieved in H.264 standard is not a result of any single feature but rather a combination of a number of encoding tools. In H.264 Encoder one of the tools, is the Context Adaptive Variable Length Coding (CAVLC) algorithm used in the baseline profile of H.264 standard [11]. This hardware is designed to be used as part of a complete low power H.264 video coding system for portable applications [12]. One critical disadvantage of this method is the unnecessary iterations even for decoding one syntax element. And also in order to solve the drawbacks of the conventional method, several methods were proposed in the literature [13]. H.264/AVC is proposed with good tradeoffs between area and throughput [14].

II. METHODOLOGY

2.1 Context-Adaptive Variable Length Coding, a specially-designed method of coding transform coefficients in which different sets of variable-length codes are chosen depending on the statistics of recently-coded coefficients.

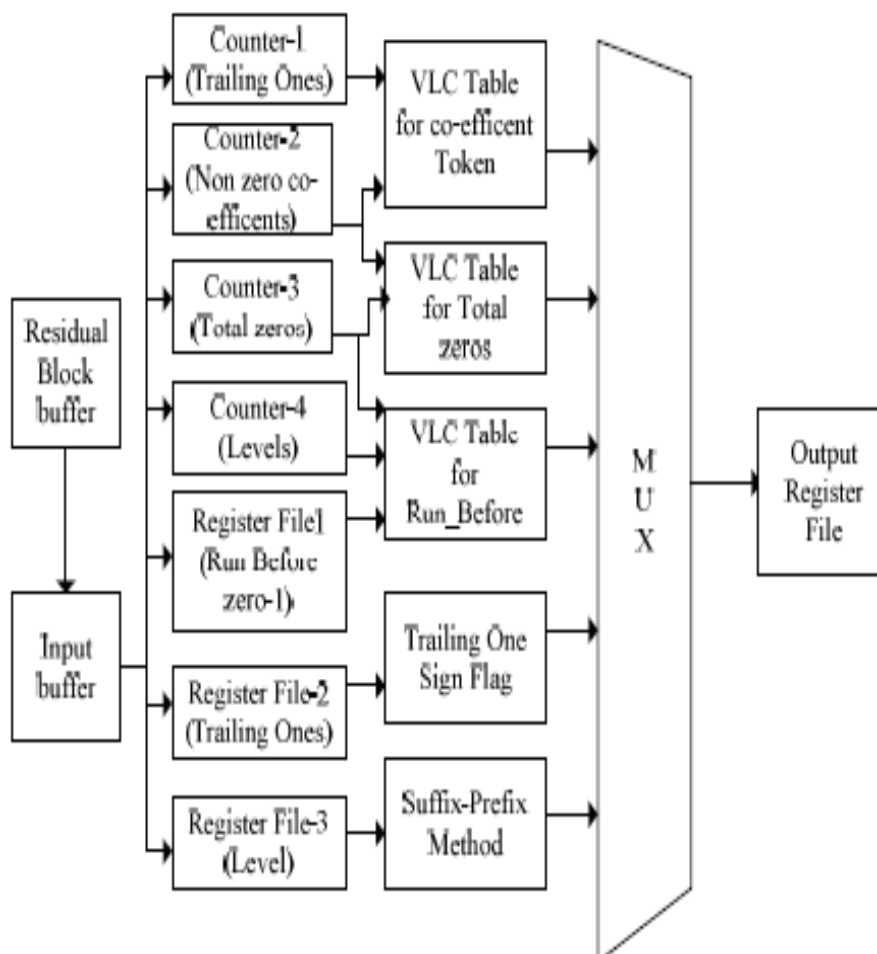


Fig 1: Architecture of CAVLC process

The output quantized values are encoded with the use of CAVLC in order to produce a bit stream. These values are accumulated in their respective registers. With help of values, the numbers of coefficients, trailing ones, zeros are calculated. The standard VLC tables are build up for encoding processing. FPGA implementation has the advantage of storing these standard values in look up table. An efficient CAVLC architecture is shown in Fig 1. The encoded stream is written in the output register file. Both blocks are implemented on FPGA to validate the functionality. Both blocks are implemented on FPGA to validate the functionality [15]. The implementation of functional blocks depicted in the Fig. 1 is carried out using Matlab.

2.2. Context Adaptive Variable Length Coding Flow chart

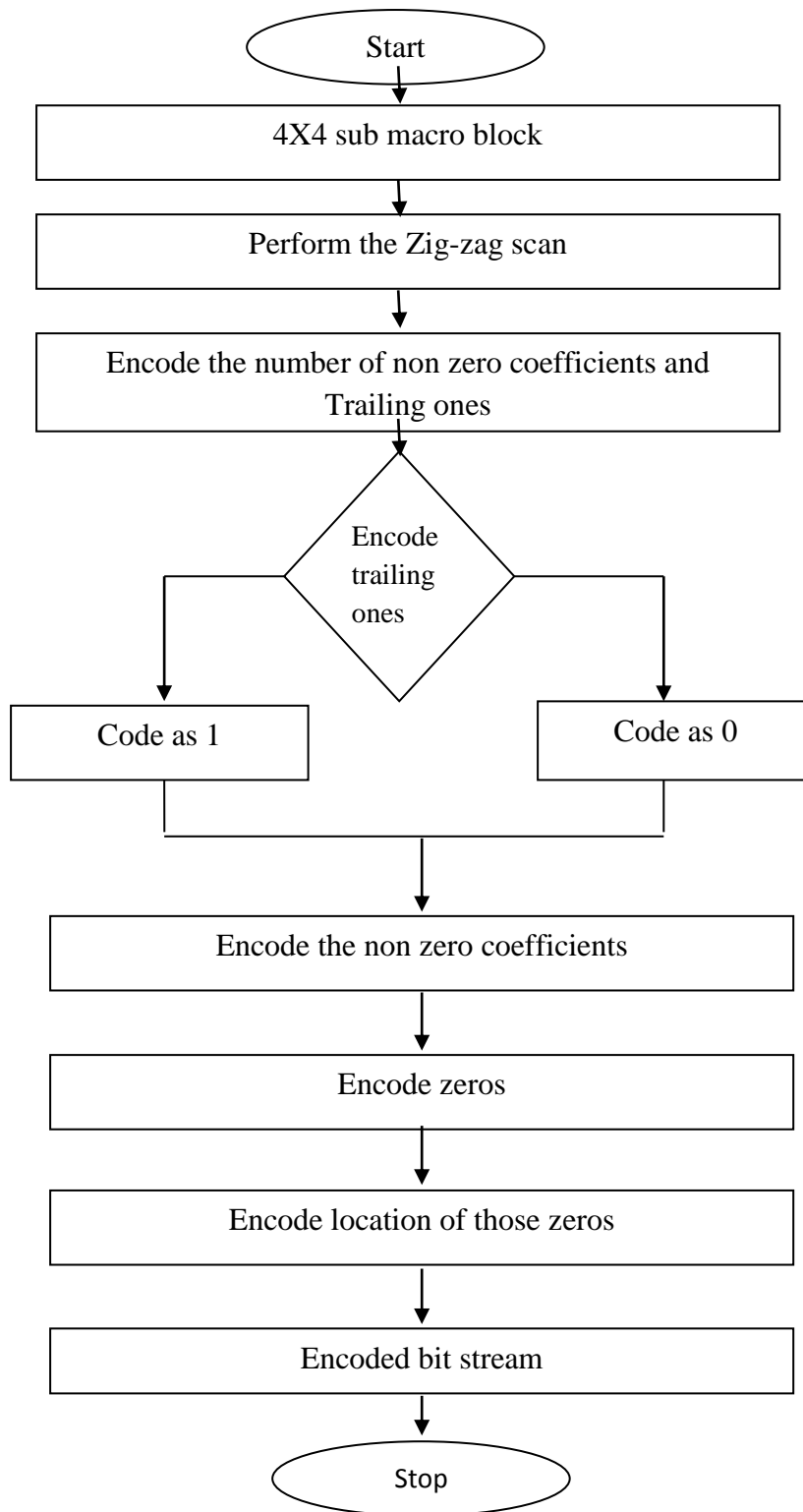


Fig 2: Flowchart for CAVLC

First CAVLC is done on 4x4 quantized blocks to check its correctness. Then this algorithm is applied to entire image and to video also. The flowchart for CAVLC encoding has been depicted in the fig 2. Following are the steps to perform CAVLC.

1. Encode the number of coefficients and trailing ones.
2. Encode the sign of each trailing ones.
3. Encode the levels of the remaining no-zero coefficients.

4. Encode the total number of zeros before the last coefficients.
5. Encode each run of zeros.

Consider the following Example,

Let the input sequence be 0, 3,-1,0,0,-1,1,0,1,0,0,0,0,0,0

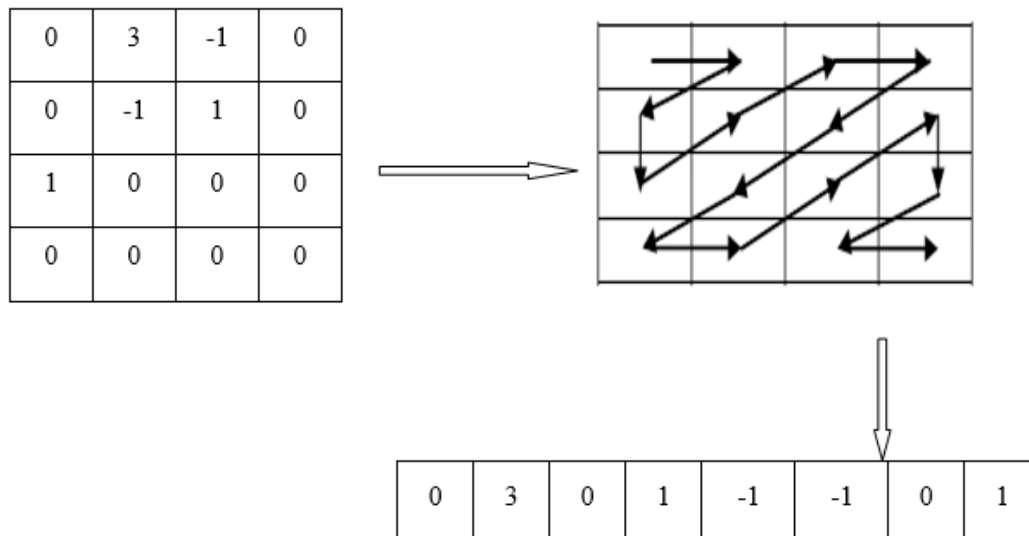


Fig 3: Zig-zag scan pattern

From Fig 3, After the Zig-Zag scan of 4x4 block the obtained sequence is:

Coefficients: 0, 3, 0, 1, -1, -1, 0, 1

Nonzero Coefficients: 3, 1,-1,-1, 1

2.2.1 Encode the number of coefficients and trailing ones.

- TotalNonZero_Coeff = 5
- Total Zeros = 3
- Trailing ones = 3
- Let $n_{C_L} = 1$

Code1: 0000100 (Table 3) is the code generated.

2.2.2 Encode the sign of each trailing ones.

Trailing One's = 1, -1, -1

Along with encoding trailing ones its position also included which will be helpful in decoding process. In fact there are four trailing ones but only three can be encoded as a 'special case'.

T1's	Sign_code	Position_code	Code_2
1	0	0111	00111
-1	1	0101	10101
-1	1	0100	10100

2.2.3. Level encoding

For any level, there will be <Prefix>, <Suffix>

- <Prefix will have <Zeros 1>
- <Suffix> will have sign bit as LSB, the remaining bits will be derived from the non zero coefficient. Number of bits of suffix is called suffix length.

There are two levels in the present 4x4 block {1, 3}. Let 'a' holds level value.

*Iteration 1:

- a = 1
- There is no Suffix
- Prefix = $2*(abs(a)-1) + sign$
Prefix = $2*(1-1) + 0 = 0$

- Suffix length = 1 (Table 4)

Prefix	Suffix
<<1>	No

*Iteration 2:

- a = 3, prev_SL = 1
- As 'a' is positive, LSB of Suffix = 0
- Remaining Suffix_bits = (Prev_SL - 1) LSBs from binary (a-1)
Remaining Suffix_bits = 0 LSBs from '10'
- Prefix = remaining MSBs value i.e. 2

Prefix	Suffix
<00><1>	0

Code_3 = 10010

2.2.4. Total zero's encoding

Total zero's = 3.

Total zero's	Code_4 (Table 5)
3	111

2.2.5. Encode each runs of zero's

NonZeroCoeff's	ZeroLeft	RunBefore	Code_5 (Table 6)
1	3	1	10
-1	2	0	1
-1	2	0	1
1	2	1	01
3	1	1	-

Output bit stream for 4x4 block example:

Code_1 =< 0000100>, Code_2 = <0 0111 1 0101 10100>

Code_3 =<1>< 0010>, Code_4 = <111>, Code_5 = <101101>

Transmitted bit stream= "000010000111101011010010010111101101"

So the actual number of bits required is (4*4*8) = 128 bits.

After applying CAVLC it is reduced to 36 bits.

At the receiver side, bit streams and its length for entire frame is received and **context adaptive variable length decoding** algorithm i.e. a reverse process of CAVLC, in which reconstruction of original 4x4 block is done.

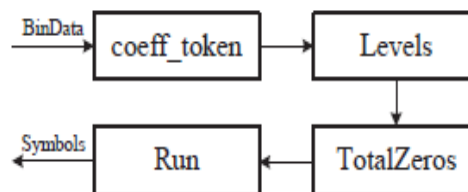


Fig 4: CAVLD

In fig 4, the coeff token is first parsed to get TotalCoeff and Trailing Ones. Then Levels of coefficients, Total Zeros and Run (representing the zeros before non-zero coefficient) are parsed successively. These are then combined to construct coefficient vectors. During the parsing processing, the different VLC tables defined in standard are adjusted adaptively according to the value of parsed data.

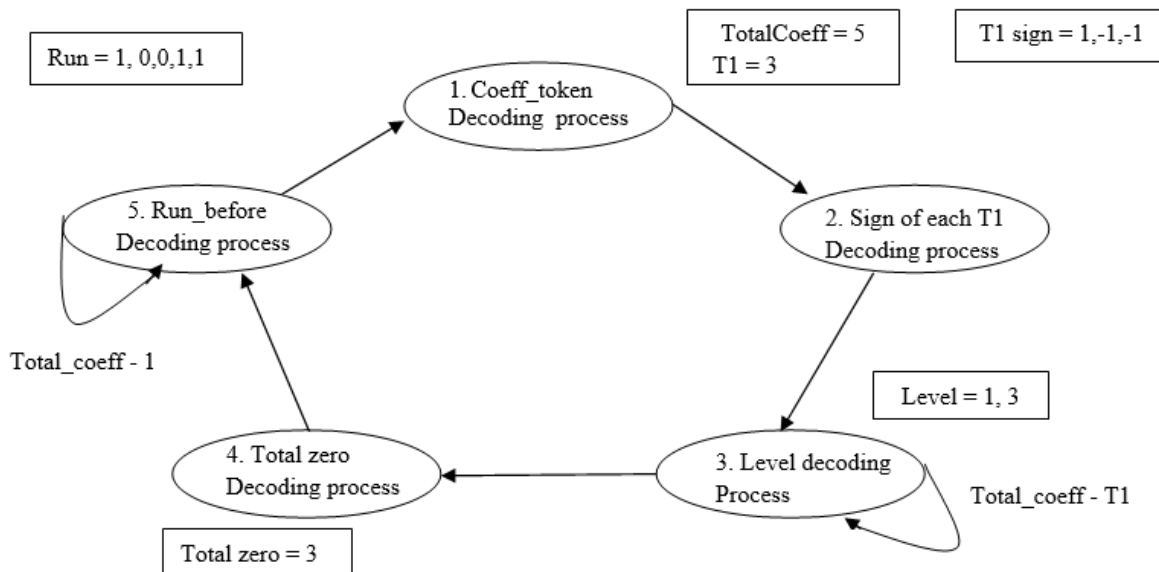


Fig 5. Decoding flow example of CAVLC

Through Fig 5, we can illustrate the decoding process flow of CAVLC, with an example. After CAVLD, inverse transformation & quantization is done as explained earlier, up sampling is done and the matrices are rearranged to form an YCBCR image, and further YCBCR image is converted to true color RGB image. Once all the frames are reconstructed, we combine all the frames back to form video. The compression achieved in this implementation is better than the existing system.

III. RESULTS & DISCUSSIONS

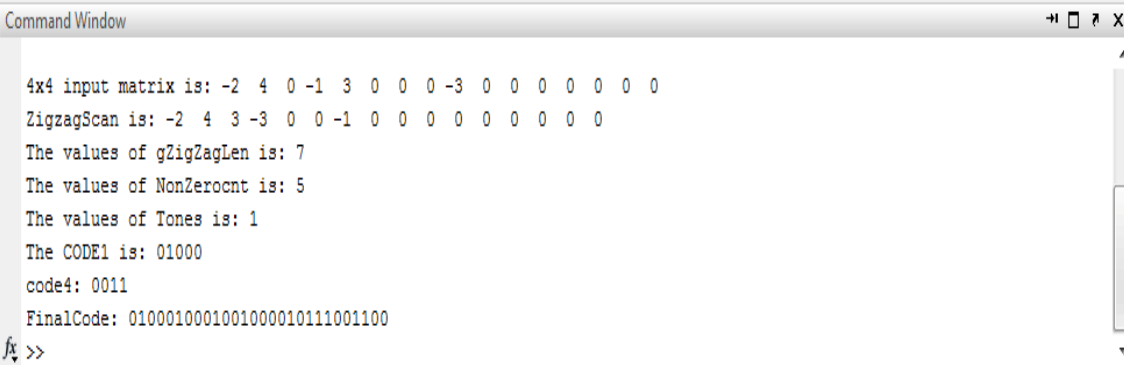
The proposed system improves the entropy coding performance by using coding techniques such as run-level and trailing ones coding that are designed to take advantage of the characteristics of the 4x4 blocks of transformed and quantized residual data. This provides better entropy coding performance in comparison to an entropy coding algorithm using a single VLC table 1.

Table 1: Quality of reconstructed block of an image and compression achieved for different QP.

SL. No.	Quantization Parameter (QP)	Compression Ratio (CR)	PSNR
1	16	4.9231	51.14dB
2	22	7.5294	50.08dB
3	34	32.0	45.12dB

The results obtained for the MATLAB CAVLC/CAVLD Code for different values of quantization parameter values are shown in Table I. The given quantization parameter value of 16, the compression ratio achieved is 4.9231 with a PSNR of 51.14dB.

We can observe from the table 1, there is a tradeoff between compression ratio and PSNR i.e better the PSNR less the compression ratio and vice-versa.



```

Command Window
4x4 input matrix is: -2  4  0 -1  3  0  0  0 -3  0  0  0  0  0  0
ZigzagScan is: -2  4  3 -3  0  0 -1  0  0  0  0  0  0  0  0
The values of gZigZagLen is: 7
The values of NonZeroCnt is: 5
The values of Tones is: 1
The CODE1 is: 01000
code4: 0011
FinalCode: 0100010001001000010111001100
fx >>

```

Fig 6: Encoded bit stream for given 4x4 sub block coefficients & Decoded coefficients from encoded bit stream.

The CAVLC/CAVLD sequence of operation carried out to encode the coefficients of 4x4 sub block and vice versa is shown in figure 6.

IV. CONCLUSION

The paper represents significantly better video compression efficiency than obtained from single VLC video compression technique. The complete CAVLC/CAVLD was realized using MATLAB. The results show that the reconstructed 4x4 block is indistinguishable from the original in spite of achieving high compression. The compression can be changed according to the user's choice. The end user may tradeoff between compression and quality.

ACKNOWLEDGEMENT

We thank our colleagues from dept. of electronics and communication engineering, GMIT who provided insight and expertise that greatly assisted the research, although they may not agree with all of the interpretations/conclusions of this paper.

REFERENCES:

- [1]. EsraSahin and IlkerHamzaoglu "A high performance and low power hardware architecture for h.264 cavlc algorithm".
- [2]. SavitaNargundmath#, Archana Nandibewool2 "Entropy Coding of H.264/AVC using Exp-Golomb Coding and CA VLC Coding" IEEE trans on International Conference on Advanced Nanomaterials & Emerging Engineering Technologies July 2013.
- [3]. Tung-Chien Chen, Yu-Wen Huang, Chuan-Yung Tsai, Bing-Yu Hsieh, and Liang-Gee Chen, Fellow, IEEE "Architecture Design of Context-Based Adaptive Variable-Length Coding for H.264/AVC" IEEE transactions on circuits and systems—ii: express briefs, vol. 53, no. 9, september 2006.
- [4]. Zhan Ma, Zhongbo Zhang, Yao Wang "Complexity modeling of h.264 entropy decoding" IEEE trans on ICIP 2008.
- [5]. M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC Baseline Profile Decoder Complexity Analysis," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 704–716, July 2003.
- [6]. D. Marpe, H. Schwarz, and T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 620–636, July 2003.
- [7]. T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. on Circuits and Systems for Video Technology vol. 13, no. 7, pp. 560–576, July 2003.
- [8]. Y. Su and M.-T. Sun, "Fast multiple reference frame motion estimation for H.264," in Proc. ICME, 2004, pp. 695–698.
- [9]. Y.-W. Huang, B.-Y. Hsieh, T.-C. Chen, and L.-G. Chen, "Hardware architecture design for H.264/AVC intra frame coder," in Proc. ISCAS, 2004, pp. II269–II272.

- [10]. I. Amer, W. Badawy, and G. Jullien, "Towards MPEG-4 part 10 system on chip: a VLSI prototype for context-based adaptive variable length coding (CAVLC)," in Proc. SIPS, Oct. 2004, pp. 275–279.
- [11]. T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuits Syst.Video Technol., vol. 13, no. 7, pp. 560–576, Jul. 2003.
- [12]. Bhaskaran and K. Konstantinides, Image and Video Compression Standards: Algorithms and Architectures, Kluwer Academic Publishers, 2nd Edition, 1997.
- [13]. Jun-Young Lee, Jae-Jin Lee, and SeongMo Park, "New Lookup Tables and Searching Algorithms for Fast H.264/AVC CAVLC Decoding" , IEEE Transaction on circuits and systems for video Technology, Vol.20,NO.7, July 2010.
- [14]. Cristiano C. Thiele, Bruno B. Vizzotto, André L. M. Martins, Vagner S. da Rosa, Sergio Bampi," A Low-Cost and High Efficiency Entropy Encoder Architecture for H.264/AVC", 2012.
- [15]. Amit Mahesh Joshi, Dr. Vivekananda Mishra, Dr. R.M.Patrikar, "Low Complexity Hardware Implementation of Quantization and CAVLC for H.264 Encoder", 2014 IEEE International Conference on Computational Intelligence and Computing Research.