Indo-Iranian Journal of Scientific Research (IIJSR)
Volume 2, Issue 1, Pages 74-82, January-March 2018

A Novel Dual Stage Pipeline Method of H.264 CAVLC Video Coding Using FPGA

M.Balasubramani[1] and S.Priya[2]

[1]*Assistant Professor, Department of ECE, Vivekanandha College of Engineering for Women (Autonomous), Tiruchengode, Tamilnadu, India.*
[2]*PG Scholar, Department of ECE, Vivekanandha College of Engineering for Women (Autonomous), Tiruchengode, Tamilnadu, India.*

Article Received: 28 November 2017 *Article Accepted: 31 January 2018* *Article Published: 06 February 2018*

ABSTRACT

Modern video coding standards used in H.264 CAVLC consist of an efficient entropy coding technique. It removes the redundancy of transformed quantized coefficients and significantly reduces the quantized coefficients. In the existing method, CAVLC implementation code the Chroma and Luma blocks this lead to low throughput in the encoded blocks. In this paper, the Luma and Chroma blocks are processed simultaneously and generate the output codes respectively. In the proposed parallel algorithm of CAVLC increase the encoding blocks throughput at high bit rates. This reduces the time generation by 40%. The proposed architecture synthesized with 100MHz clock using Xilinx.

Keywords: Video coding, Entropy coding and CAVLC.

1. INTRODUCTION

Video compression technique plays an important role in today's digital world. It is widely applied in the field of entertainment, multimedia short messages, storage, videophone, video conference, internet streaming and HDTV broadcasting. The efficient transmission and multimedia data storage we are going for the CAVLC with Advance video coding technique. In comparison with the traditional method of MPEG-2, MPEG-4 VISUAL where H.264 gives the better image quality at the same bit rate. H.264 video encoder process with prediction, transform and encoding to produce the output in the form of compressed bit stream. The decoder block carried out with decoding, inverse transform, and reconstruction of original sequence to produce the perfect video.

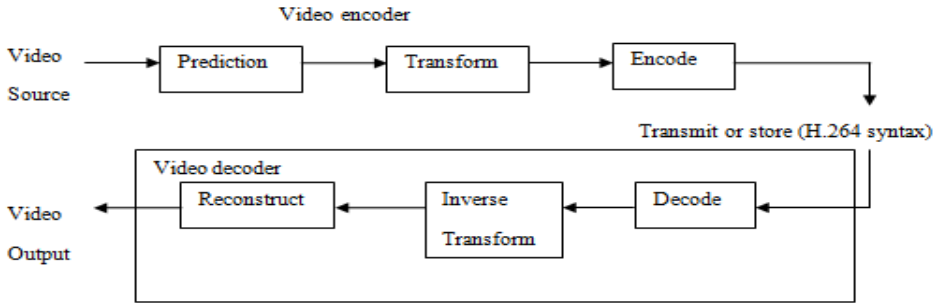

Fig.1. H.264 video coding and decoding process.

To achieve the higher compression ratio, Context-Based Adaptive Variable Length Coding (CAVLC) is adopted as one of entropy encoder in H.264 standard [1]. Compared with the traditional entropy encoder, CAVLC can achieve better coding efficiency, but the algorithm complexity is higher. On the other hand, because of the data dependency in CAVLC, its hardware implementation will be in complex form. The block diagram of H.264 encoder is shown in Fig.2.

In H.264 Encoder, the residue of Luma sub-macroblock (of size4x4) is resulted from subtracting the predicted sub-macroblock (of size 4x4) and the original sub-macroblock (of size 4x4). This residue

is transformed by integer Discrete Cosine Transform (integer DCT). Then this transformed residue is quantized. Finally, the quantized-transformed residue is encoded by CAVLC ". The same steps will carryout on the Chroma blocks.

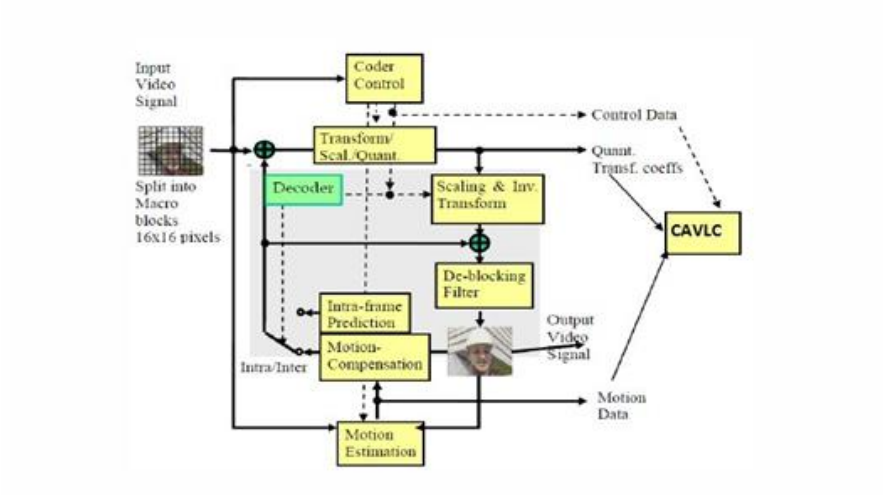

Fig.2. Block diagram of H.264 Encoder.

In this paper, an efficient implementation of CAVLC encoder for H.264 AVC video coding applications is proposed. The main concepts is to increase the throughput by reducing the waste time consumed during the input quantized transformed residual data of the 4x4 block and its output coded data. The proposed method exploits this waste time to coding the quantized transformed residual data of the 2x2 block.

The proposed paper is organized as follows: the H.264 CAVLC algorithm is presented in section 2. Then the related works are discussed in section 3. Our proposed hardware implementation architecture of CAVLC is described in Section 4. The implementation and simulation results are discussed in Section 5. The conclusions are presented in the last section.

2. CAVLC OF H.264 ALGORITHM OVERVIEW.

In order to encode one macroblock, consisting of Luma and chroma coefficients, the Luma coefficients and Chroma coefficients will be processed respectively. CAVLC algorithm is used to encode the transformed and quantized residual luminance then chrominance coefficients of a macroblock. All the transformed and quantized 4x4 and 2x2 blocks for a macro block are given as inputs to CAVLC algorithm. CAVLC algorithm processes each 4x4 block in zig-zag scan order and each 2x2 block in raster scan order.

The important parameters to be encoded are:

l) Number of nonzero coefficients (Total Coeff) and Trailing ones (T 1).

2) The pattern of Trailing ones (T 1).

3) The nonzero coefficient (Levels).

4) Number of zeros embedded in the non zero coefficients (Total zeros).

5) The location of those embedded zeros (Run_before).

Therefore, the CAVLC has five steps to encode one 4x4 block. CAVLC is entropy encoding used to encode residual data in 4×4 or 2×2-coefficient blocks. The coding techniques are applied according to statistical characteristic of the block: The blocks of quantized transform coefficients contain mostly zero coefficients. Thus, run-length coding is applied to encode the zero strings. In the zigzag order, non-zero coefficients are gradually lower. The last non-zero coefficients of the blocks are normally +1 or −1. The last consecutive coefficients with magnitude 1 are called trailing ones. The maximum number of trailing ones is three. These coefficients are encoded in a special way. The other non-zero coefficients are called "level". Levels are encoded in inversed zigzag order because in this inversed order, the magnitude of the previous level is used to predict the value of the next level. This prediction is then used to select the appropriate coding table. Because the numbers of non-zero coefficients in neighboring blocks are correlated, these numbers in upper and left blocks are used to predict the current block. The ITU-T recommendation has introduced the five syntax elements to be encoded in the CAVLC. They are coefficient token (coeff_token), trailing ones' signs, and level, total zero, and run before.

Coefficient Token encoding

Coefficient token (coeff_token) is a syntax element presenting a pair of numbers: the number of non-zero coefficients and the number of trailing ones in a block. The VLC table selection depends on the number of non-zero coefficients in the upper and the left blocks. If the number of non-zero coefficients in the upper block is nU and the number of non-zero coefficients in the left block is nL, then, the parameter nC used to decide which VLC table will be selected is calculated as follows:

$$nC = ceiling(\frac{nU + nL}{2})$$

The coeff_token of a 4×4 luma block is encoded using the table VLC0 if nC is less than 2. The table VLC1 is selected if nC is greater than 1 and less than 4. If nC is greater than 3 and less than 8, the VLC2 is selected. If nC is greater than 7, the Fixed Length Coding (FLC) table is used. We also have a special table for encoding the coeff_token of 2×2 chroma DC blocks.

Trailing One Sign flag encoding

Each Trailing One is encoded in one bit presenting its sign. If the coefficient is 1, the sign bit is zero ('0'). If it is −1, the sign bit is one ('1'). The Trailing One signs are encoded in the bit stream in inversed zigzag order.

Level encoding

Levels are encoded in inversed zigzag order. In the level encoding, seven VLC tables are used. The next VLC table is selected according to the current VLC table and current magnitude of level. The first level is encoded using VLC0. Then the VLC number is increased if the magnitude of the current level is larger than the correlate threshold

of the current VLC. One exception is that if there are more than 10 nonzero coefficients and less than three trailing ones, the first level will be coded using table VLC1. Other exception is that if there are less than three trailing ones, the first level coded with the magnitude is decreased by 1.

Total Zero encoding

Total Zero is the number of zero coefficients standing before the last non-zero coefficient in the zigzag order. Total zero is encoded using 15 VLC tables selected by the number of non-zero coefficients in the luma blocks. Three other tables are used for encoding chroma DC blocks.

Run_Before encoding

Run_before is a sequence of numbers of zero coefficients standing before levels in zigzag order. However, run_befores are encoded in inversed zigzag order. The VLC table selection is done based on "zero left" information, that is, the number of zero left after each run_before is encoded. Next zero left is equal to current zero left minus current run_before. There are 7 VLC tables used for run_before encoding. Finally, the syntax structure of output bitstream for one block data is in the following order: coeff_token, Trailing One signs, Levels, Total Zero, and run_before.

A conventional CAVLC encoder is composed of five encoders to generate five syntax elements of the current block. Each encoder contains several look-up tables to store VLC tables. Table selection is done by the previous coded syntax elements with some exceptional cases.

3. RELATED WORKS

In the previous works had presented the implementation of CAVLC encoder. Most of them tried to implement a high throughput CAVLC encoder, using pipelining techniques [2]. Two-stage pipeline architecture requires double buffer size to store all the statistic information of one block before the data is encoded. Therefore, Arun et al [6] introduced a mechanism that scans the coefficients in inversed zigzag order and proposed a special buffer structure to maintain the buffer size at the size of one block. Parallel symbols encoding is an efficient method in terms of performance, however, it obviously double the area cost of symbol encoders. In the paper [6] also needs to handle the data dependency, that is, with two symbols encoded in parallel, the encoding of the later symbol depends on the previous one.

Some authors presented an efficient method to overcome the bottleneck at the scan phase by scanning coefficients in parallel. This method halves the required time of the scan phase. Parallel coding of level and run-before is also applied. Finally, Chuan-Yung Tsai [7] has implemented a low power CAVLC encoder which can reduce up to 69% the power consumption but the total gate count is somewhat high.

4. PROPOSED HARDWARE ARCHITECTURE

In a real time, video encoder stores the residual data after quantization in a two separated buffers one for luminance and the other for chrominance [2]. During encoding of residual data, the two chrominance (Cr,Cb) transformed quantized residual data is encoded at the same time of the luminance transformed quantized residual data is encoded. The two chain of calculations works at the same time. The proposed input/output architecture is shown in Fig.3

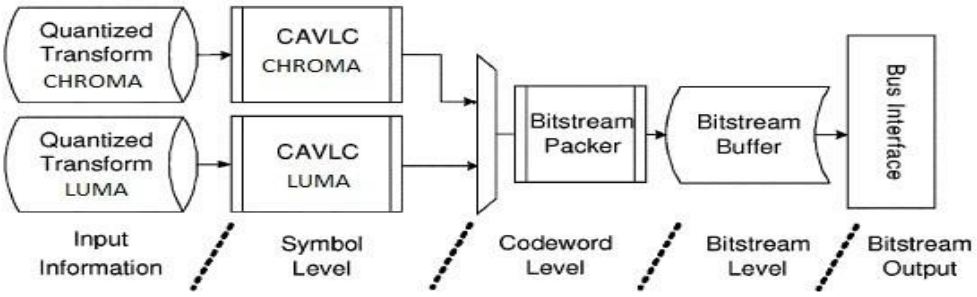

Fig.3. Proposed method

our proposal architecture encodes one macroblock by dividing it into two chains one for the luminance blocks and the other for chrominance blocks and encodes them simultaneously. This method takes only 5 slots of time as seen in Fig.4.

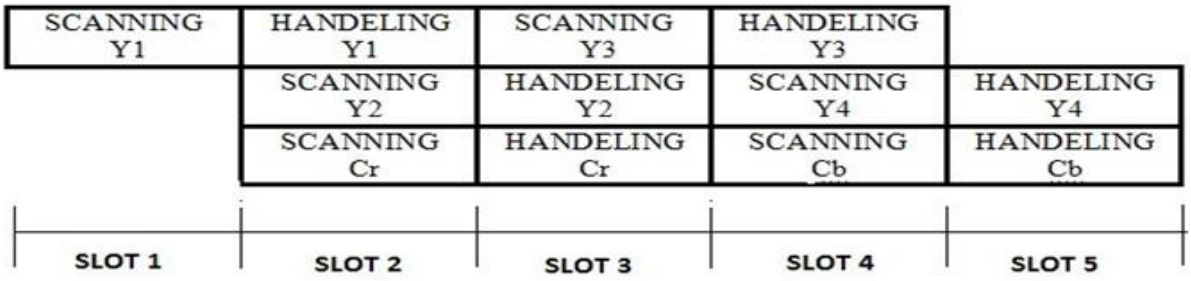

Fig.4. Number of time slots to process 4 luma and 2 chroma

In our implementation, there are two clocks used for calculation (clkl) and (clk2), where (clk2= 2clkl). The fIrst clock (clk2) is used during input coefficient data and the second clock (clkl) is used when output coded data.

In the Fig.5 , the fIrst stage (scan stage), the 4x4 luminance block is scanned by 16 cycles of clk2 for calculating and storing the corresponding information then used it during second encoding process.

In the second stage (statistical stage), about (8) cycles of clk2 for calculating all the coefficients are needed. In the final stage, the output bit stream generated during 8 cycles of clkl (or 16 clk2). So the total number of cycles used for encoding 4x4 blocks is about 44 cycles of c1k2. The 2x2 luminance block, is scanned then calculated, the total number of cycles used for encoding 2x2 blocks is about 21 cycles of c1k2.

So the proposed method can coding 2 chrominance blocks during 1 luminance block. This method will lead to reduce the time of calculation or increase the throughput of coding CAVLC.

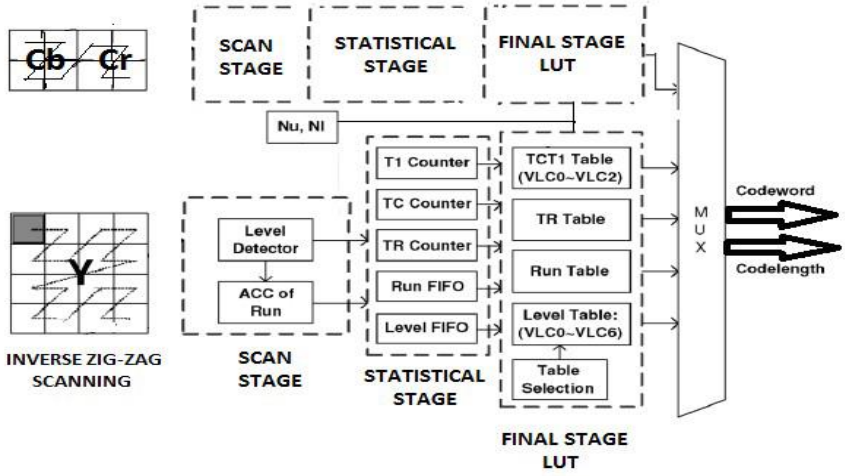

Fig.5. The CAVLC calculation stages

The proposed architecture of CAVLC contains a number of counters and register files to store the information for a block that will be encoded by variable length codes. Non-Zero Coefficients counter is used to store the number of non-zero coefficients (Total Coeft). Trailing Ones counter is used to store the number of trailing ± 1 values (Trailing Ones). Total Zeros counter is used to store the total number of zeros before the last non-zero coefficient (Total Zeros). Level counter is used to store the number of non- zero coefficients other than the Trailing Ones. Trailing Ones register file is used to store the sign of each Trailing One. Level register file is used to store the level (sign and magnitude) of each non-zero coefficient other than the Trailing Ones. Run _Befores register file is used to store the number of zeros preceding each non-zero coefficient. The finite state machine represents the steps of calculation are shown in Fig.6.

CAVLC hardware begins the encoding for a 4x4 or 2x2 block by reading the coefficients from the input buffer in reverse zig-zag order. In each cycle, it reads one coefficient from the input buffer analyzes the coefficient and updates the information stored in the related counter and register file. At the end of this process, the counters and register files mentioned above contain all the information for the current block that will be encoded with variable length codes. Reverse zig-zag scanning enables us to determine the necessary information for encoding a 4x4 block

by reading and analyzing each coefficient only once. This reduces the power consumption by reducing the switching activity on the input buffer address and data signals.

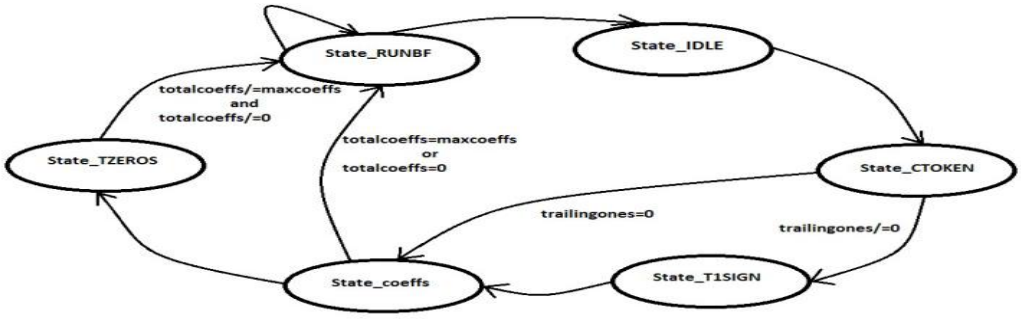

Fig.6. The FSM of the CAVLC.

5. SIMULATION AND IMPLEMENTATION

The proposed architecture engine with dual-block pipelined architecture is synthesized and simulated by using ISE 14.5 navigator tools and modelsim 6.5f. To achieve full hardware utilization by the dual-buffer architecture, two block statistic buffers are required and two types of memories are required.

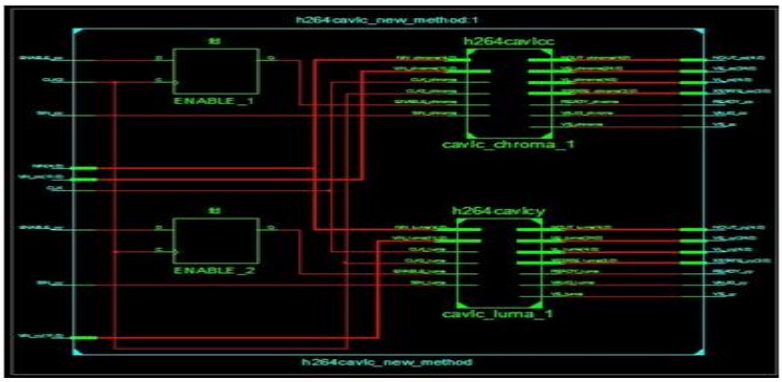

Fig.7. RTL schematic of proposed method.

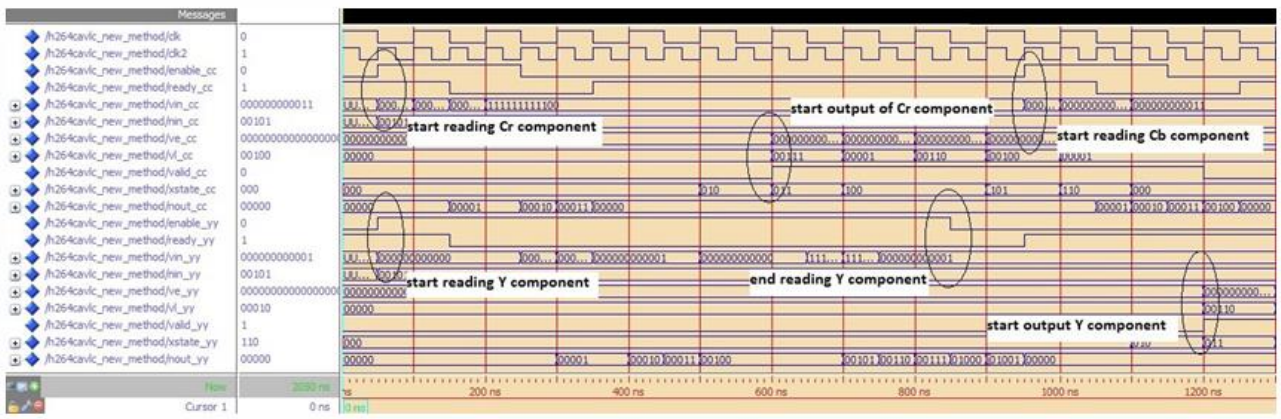

Fig.8. the behavioral simulation result.

The coefficient memory and bit stream memory are used as input and output buffers for system consideration. The upper 2 x 2 block for chrominance is used to process 2x2 block and the total coefficient memory is used to store. The lower 4x4 block for luminance is used to process the 4x4 block as shown in Fig.7 and Fig.8.

6. CONCLUSION

This proposed paper focused on high throughput implementation of CAVLC encoder. It presented a high performance architecture for real-time implementation of H.264 CAVLC encoder which requires low-cost memory and low hardware complexity. Reverse zigzag scanning determines the necessary information for encoding a block by reading and analyzing each coefficient only once. This reduces the power consumption by reducing the switching activity on the input buffer address and data signals. The proposed architecture is implemented in VHDL using Nexys-4 board, based on the latest Artix-7 FPGA XC7AlOOT- CSG324 from Xilinx.

REFERENCES

[1] Arun Kumar Pradhan, Lalit Kumar Kanoje and Biswa Ranjan Swain, "FPGA based High Performance CAVLC Implementation for H.264 Video Coding", International Journal of Computer Applications,Vol.69, No.10, PP. 23-28, May 2013.

[2] Asma Ben Hmida, Salah Dhahri, and Abdelkrim Zitouni, "A High Performance Architecture Design of CAVLC Coding Suitable for Real-Time Applications", World Cong. on Multimedia and Computer Science, 2013.

[3] Chih-Da Chien, Keng-Po Lu, Yi-Hung Shih, and Jiun-In Guo, "A High Performance CAVLC Encoder Design for MPEG-4 AVC/H.264 Video Coding Applications", IEEE International Symposium on Circuits and Systems, PP. 3838-3841, 2006.

[4] Choudhury A. Rahman, and Wael Badawy, "CAVLC Encoder Design for Real-Time Mobile Video Applications", IEEE Transactions on Circuits and Systems—II, Vol. 54, No. 10, PP. 873-877, OCTOBER 2007.

[5] Iole Moccagatta, Salma Soudagar, Jie Liang and Homer Chen, " Error-Resilient Coding in JPEG-2000 and MPEG-4", IEEE Journal on Selected Areas in Communications, Vol. 18, No. 6, PP. 899-914, JUNE 2000.

[6] Klaus Schoffmann, Markus Fauster, Oliver Lampl, and Laszlo Bosz ormenyi , " An Evaluation of Parallelization Concepts for Baseline-Profile Compliant H.264/AVC Decoders". LNCS 4641, pp. 782–791, 2007. Springer-Verlag Berlin Heidelberg. 2007.

[7] N. Keshaveni, S. Ramachandran and K.S. Gurumurthy, "Implementation of Context Adaptive Variable Length Coder for H.264 Video Encoder", International Journal of Recent Trends in Engineering, Vol 2, No. 5, PP. 341-345, November 2009.

[8] Ngoc-Mai Nguyen, Xuan-Tu Tran, Pascal Vivet and Suzanne Lesecq, " An Efficient Context Adaptive Variable Length Coding Architecture for H.264/AVC Video Encoders", International Conference on Advanced Technologies for Communications (ATC), PP. 158-164, 2012.

[9] Sangyoon Park, Kyeongyuk Min, Jongwha Chong, "The New Memory-Efficient Hardware Architecture of CAVLD in H.264/AVC for Mobile System", Communications and Information Technology, 2009. 9th International Symposium on ISCIT, PP. 204-207. 2009.

[10] Shau-Yin Tseng, Tien-Wei Hsieh, "A Pattern-Search Method for H.264/AVC CAVLC decoding", Multimedia and Expo, 2006 IEEE International Conference on, pp 1073 – 1076, July 2006.

[11] Taheni Damak, Imen Werda, Mohamed Ali Ben Ayed, Nouri Masmoudi4, "Context adaptive variable length decoding optimization and implementation on tms320c64 dsp for h.264/avc", Science Journal of Circuits, Systems and Signal Processing, Vol. 2, No. 1, PP. 6-15, 2013.

[12] Tony Gladvin George, Dr.N.Malmurugan, "A New Fast Architecture for HD H.264 CAVLC multi-syn Decoder and its FPGA Implementation", International Conference on Computational Intelligence and Multimedia Applications, Sivakasi, India. ICCIMA 2007.

[13] Tung-Chien Chen, Yu-Wen Huang, Chuan-Yung Tsai, Bing-Yu Hsieh, and Liang-Gee Chen, "Architecture Design of Context-Based Adaptive Variable-Length Coding for H.264/AVC", IEEE Transactions on Circuits and Systems—II, Vol. 53, No. 9, PP. 832-836, September 2006.

[14] . K. Lai, C. C. Chou, and Y. C. Chung, "A simple and cost effective video encoder with memory-reducing CAVLC," in Proc. IEEE ISCAS'05, Vol. 1, pp. 432–435, 2005.