Volume 6, Issue 1, January 2016 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

# Use of Internal RAM for Improved CAVLC Performance

**Kshama R. Vaze**[*]
E&TC Department, GEC,
Goa University, Goa, India

**Prof. V. K. Joseph**
Electrical Department, GEC,
Goa University, Goa, India

***Abstract— Video Compression is a necessity in today's life to bring down the storage requirements to manageable levels and to transmit data with existing channel capacities. H.264 is one of the recent video compression standards. H.264 Encoder consists of various functional modules such as Intra prediction processor, an Integer transform, Quantization processor and a CAVLC (Context Adaptive Variable Length Coder). Amongst these modules, CAVLC is a lossless compression technique. Paper proposes use of RAM to speed up operation of CAVLC unit.***

***Keywords— H.264, Intra prediction processor, Integer transform, Quantization processor, CAVLC.***

## I. INTRODUCTION

Video compression has become necessity today to save storage and bandwidth. This requirement for compression has led to development of various video compression standards and techniques. H.264 is one of the recent video compression standards. H.264 encoder comprises of prediction, transform, quantization and encoding processes to produce compressed bit stream. H.264 decoder involves the complementary processes of decoding, dequantization, inverse transform, and reconstruction to produce displayable video sequence. Paper focuses on CAVLC (Context Adaptive Variable Length Coder) unit of H.264 encoder. CAVLC is lossless stage of H.264 encoder. CAVLC improves coding efficiency over simple VLC. Codewords used by CAVLC are stored in LUTs (Lookup Tables) and choice of codeword depends on frequency of occurrence of symbol it represents. Essentially each symbol uses multiple VLC tables that are adapted to the symbol's context to create a codeword. Paper proposes an architecture that uses RAM to store previously coded 4x4 blocks of nonzero coefficients which are used in the next stage. This use of RAM speeds up processing of CAVLC.

## II. ENTROPY CODING IN H.264

For the coding of the quantized transform coefficients, H.264 offers two entropy coding methods: Context-Adaptive Variable Length Coding (CAVLC) and Context-Adaptive Binary Arithmetic Coding (CABAC).

**Context Adaptive Variable Length Coding**

After the zig-zag scan, the number of trailing zeros and trailing ones are considered. Depending on the statistics of the neighbouring blocks, it uses one of four available VLC tables to encode this data. The encoder then codes the coefficients in reverse order, choosing one of six possible Exp-Golomb code tables defined in the standard.

**Context Adaptive Binary Arithmetic Coding**

This method has the following desirable properties:

- It is possible to assign a non-integer number of bits to each symbol, e.g. transform coefficient, which is advantageous for symbols with higher probability than 0.5.
- The adaptive arithmetic code allows the code to change as the statistics of the input changes.

The implementation complexity is kept to a minimum by utilizing only shifts and table look-ups.
CAVLC is supported in all H.264 profiles, unlike CABAC which is not supported in Baseline and Extended profiles.
On the downside, CABAC entropy coding is significantly more computationally intensive than CAVLC.

## III. CAVLC ALGORITHM

A video sequence is input to the integer transform and quantization processor. Where the Y, Cb, Cr components are intra predicted, transformed and quantized. Quantized coefficients are then fed to CAVLC module which assigns variable length codes to get the desired compressed bit stream. The quantized coefficients are 4x4 blocks of data which is scanned in zigzag manner and then fed to CAVLC processor after quantization.

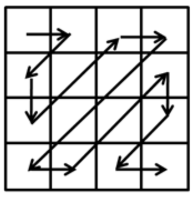

Fig.1 Zigzag scan order

CAVLC takes every 4x4 block of transformed and quantized coefficients and reorders the data into a vector using the zig- zag scan. When encoding intra predicted and chroma macroblocks, H.264 encodes the DC coefficients (the top left most coefficient of a 4x4 block) of a macroblock separately from the rest of the coefficients.

The coefficient vectors are then analysed and converted into a binary bit stream by the CAVLC. CAVLC is optimized for encoding the transformed and quantized residual data. The binary bit streams contain 5 sections as listed below.

**1. Coefficient Token**:

The coefficient token is a VLC that contains information concerning the total number of non-zero coefficients in the vector, in addition to the number of "Trailing Ones". The token is determined using a LUT.

**2. Trailing Ones**:

A string of positive and negative ones often occur at the end of the coefficient vector. Since they are so common, CAVLC makes a special case for up to three trailing ones. A single bit for each trailing one is used to encode the sign of that trailing one, (1 for negative, 0 for positive) in reverse zig zag scan order.

**3. Total Zeros**:

This portion of the bit stream contains information regarding the number of zeros that are embedded within the non-zero coefficients. A VLC is determined using an LUT and based on the number of non-zero coefficients and embedded zeros.

**4. Levels**:

The value of each of the remaining non-zero coefficients in reverse order is found in the levels portion of the CAVLC bit stream. Each level can be encoded using one of 7 LUTs, or through stream processing. The length of each level increases as the absolute value of the coefficients increase.

**5. Run Before**:

For every non-zero coefficient (until all the embedded zeros have been accounted for) a few bits are appended to the bit stream that tells the decoder how many zeros "run before" the current coefficient.

## IV. PROPOSED ARCHITECTURE OF CAVLC

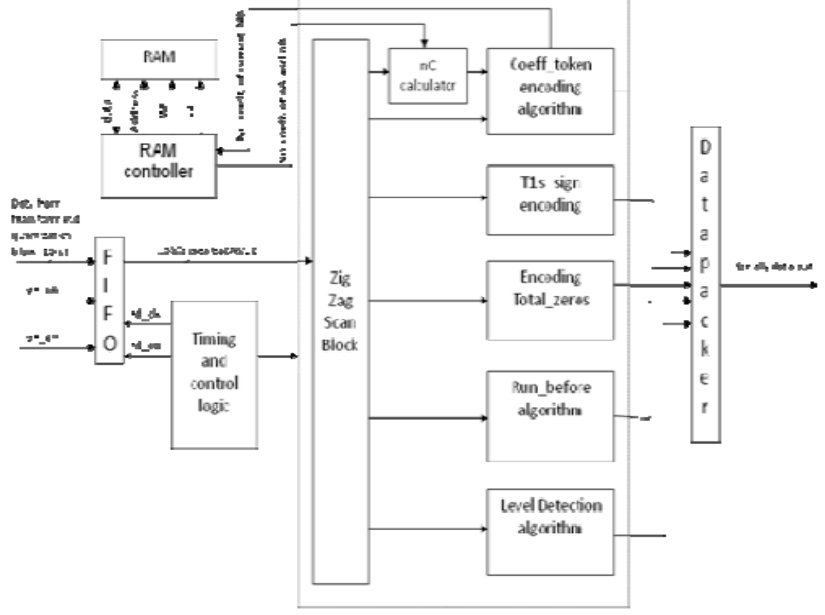

Fig. 2 Cavlc Architecture

**FIFO (First in first out)** :

In this proposed architecture FIFO is used to receive and storing data coming from transform and quantization blocks of H.264 codec. We can say that FIFO will provide medium for coming data to pass to the CAVLC encoder. The incoming 10 bit data from transform and quantization blocks is writing to FIFO using write clock (wr_clk) and write enable (wr_en). When this data is needed by CAVLC encoder it will read from FIFO using read clock (rd_clk) and read enable (rd_en). This outgoing 10bit data is going to the zigzag scan module of CAVLC encoder.

**Timing and control logic:**

This module will decide on which time which module is perform its function. Control module is used to control the overall operating sequence orders and generate the control signals those are necessary for each functional unit. When previous stage (transfers and quantize unit) finishes one block and writes data to FIFO, and then Control module will activate the CAVLC to start reading data from FIFO for encoding.

**Zigzag Scan block:**

When the encoding processing starts, residuals and the type of blocks are sent to zigzag scan module under the control of control module. The backward scan technology is adopted, which scans 4×4 macroblock coefficients in following manners.

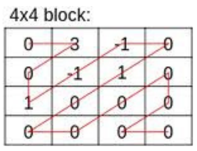

Fig.3 4×4 block

**RAM and RAM controller:**
RAM is used to store number of coefficients of all encoded macroblock which are needed to calculate nC. This nC calculator will take stored number of coefficients of upper macroblock and left macroblock of current encoding macroblock.RAM controller controlling access of data from RAM according to read or write operation by providing data to nC calculator and taking data from coeff_token to store number of coefficient of current macroblock.

**nC calculator:**
CAVLC also predicts the number of coefficients in the current array using the context variables nA and nB. These two variables represent the number of non zero coefficients in the neighbouring 4x4 blocks, in particular the blocks that lie above (nA) and to the left (nB) of the current 4x4 block.

**Coeff_token block:**
The Coeff_token generates the code by calculating the values of Totalcoeffs, number of T1 and nC value from the nC generator block. After taking the nC value from the nC generator block, the Coeff_token block implemented VLC tables and FLC table instead.

**T1 sign encoding:**
This block takes input from zigzag scan block in reverse order. Then from last coefficient it finds trailing one and encodes is sign as 1 for negative and 0 for positive. Up to maximum three trailing ones it will encode sign and after that it takes coefficient as level coefficient. will be the code output and length depending on the value of the Totalzeros and Totalcoeffs.

**Level encode block:**
The level block encodes the nonzero coefficients. Level gives the level of each nonzero coefficient. The format of the level code is the level prefix followed by bit 1 and that followed by level suffix. Level prefix is the array of zeros that are in level code and level suffix is the array of bits. The level input is taken for each nonzero coefficient and this block generates the level prefix, length of level prefix, level suffix and length of level suffix based on the look up tables. These are then sent to the data packer.

**Run before block:**
The run before block encodes the number of zeros preceding each nonzero coefficient in reverse zigzag order by taking the inputs run before, Totalcoeffs and Totalzeros. The block calculates the run before for each nonzero coefficient. It determines the number of zeros left by subtracting the run before from Totalzeros. Depending on the value of the zeros left and the run before, the code and the length of the code are determined from the look up table.

**Data Packer:**
The data packer receives all the codewords generated in the encoding phase, and when enabled, creates the bit stream. The Data Packer block gets T1's Sign inputs from trailing one sign encoding clock; Coefftoken and length of the token from the Coefftoken block; Totalzeros code and its length from the Totalzeros block; Level suffix, level prefix and their lengths from the level block; Run before code and its length for each nonzero coefficient from run before block. The Data Packer generator reads in the inputs and their lengths and appends all the code based on their lengths together in order to generate the encoded bitstream.

## V. SIMULATION RESULTS

### Coeff_token block Simulation

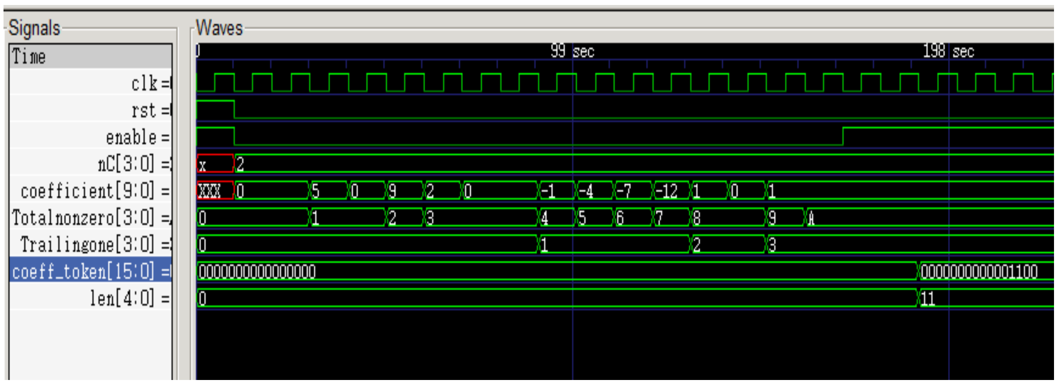

Fig.4. coefficient token simulation

**T1 sign encoding simulation**

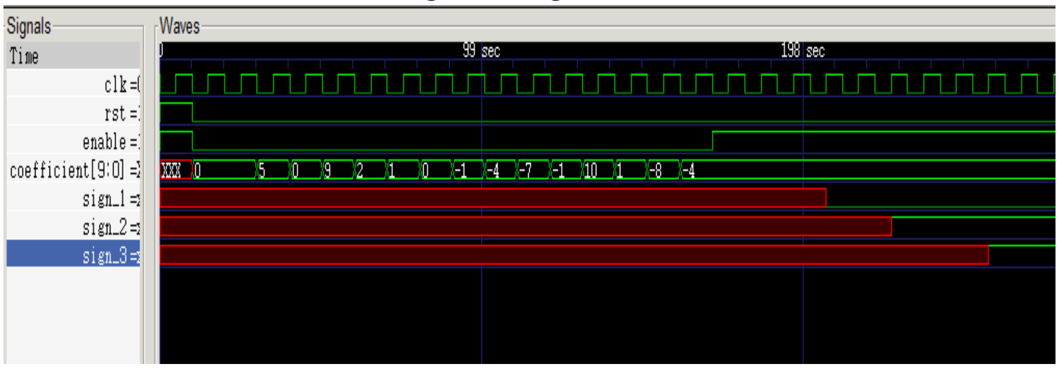

Fig.5. Trailing ones sign encoding simulation

**Total Zeros block Simulation**

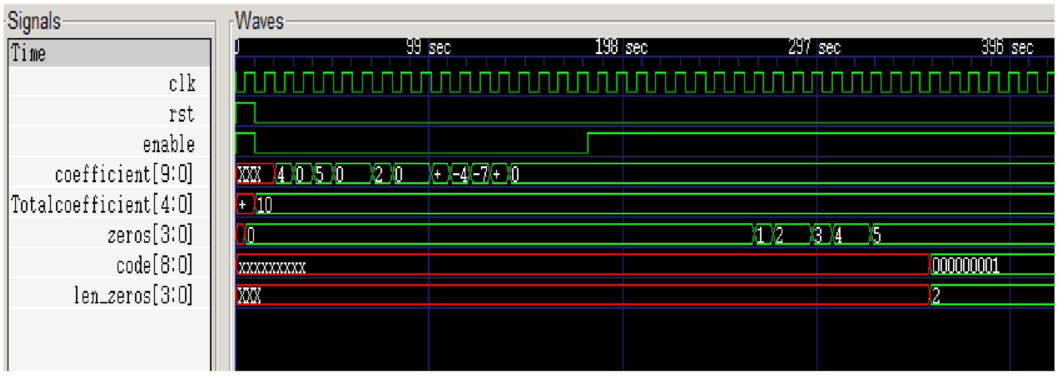

Fig.6. Total embedded zeros simulation

**Run before Simulation:**

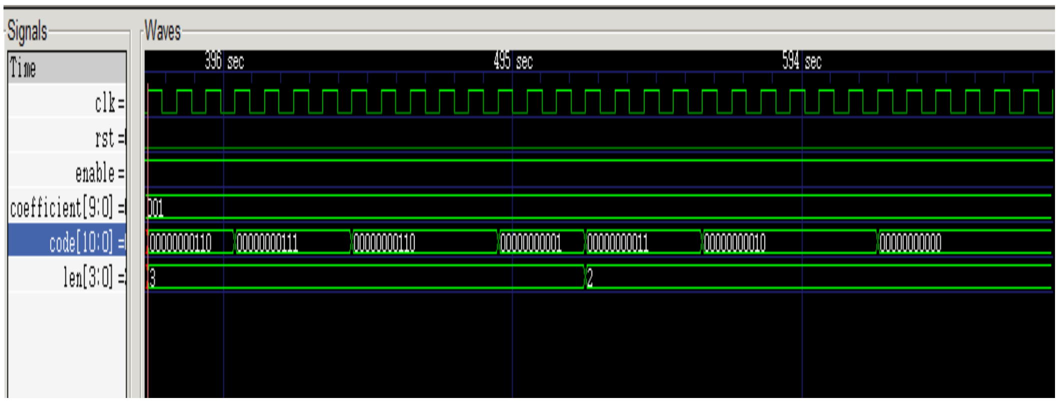

Fig.7. Runbefore simulation

## VI. CONCLUSION

CAVLC is the entropy coding method adopted in H.264/AVC. In this paper, I present technique for CAVLC VLSI implementation which has internal RAM for storing nA and nB. All modules of CAVLC are worked in parallel so there is no need to wait for completion of any module. The CAVLC architecture was designed and verified with simulations using Modelsim. Verilog HDL description of the architecture was verified for functionality with simulation.

## ACKNOWLEDGMENT

Firstly, I would like to sincerely thank my project guide, Prof. **V.K. Joseph** for providing me the required guidance throughout my project-work.
I would also like to thank Mr. **Nitesh Guinde** (Associate Professor, Electronics and Telecommunication Engineering, GEC, Goa) for helping me to understand concepts of Video Compression.
I also wish to thank Prof**. Hasan Ali Virani**, H.O.D, Electronics and Telecommunication department for his motivation.
I am grateful to Mr. **V. N. Shet**, Principal, Goa Engineering College, for his support and encouragement.
I want to thank all my friends who have knowingly or unknowingly extended their support for my project.
Finally, I would like to thank my family members who have always been a great source of inspiration and support for me.

## REFERENCES

[1] Iain E. G. Richardson, "*H.264 and MPEG -4 video compression*"

[2] Li Luo, Zheying Li, Qinmei Yu, "*A VLSI Implementation for CAVLC for H.264/AVC*", School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China, 100044; College of Information, Beijing Union University, Beijing, China, 100101.

[3] Yi, Y., Song, B. C., "*High-Speed CAVLC Encoder for 1080p 60-Hz H.264 Codec*", Signal Processing Letters. IEEE.vol.15, pp.891-894(2008).

[4] T.-H. Tsai S.-P.Chang T.-L. Fang., "*Highly efficient CAVLC encoder for MPEG-4 AVC/H.264*", Department of Electrical Engineering, National Central University, Chung-Li 320, Taiwan, Republic of China .

[5] N. Keshaveni, S.Ramachandran and K.S.Gurumurthy, "*Implementation of Context Adaptive Variable Length Coder for H.264 Video Encoder*", Proc. International Journal of Recent Trends in Engineering, November 2009.