An Efficient VLSI Design of CAVLC Encoder

Rohan Mukherjee[1], Anupam Banerjee[2], Avirup Maulik[3], Indrajit Chakrabarty[1], Pranab Kumar Dutta[3], Ajoy Kumar Ray[1]
[1]Department of Electronics & Electrical Communication Engineering
[2]Advanced Technology Development Center
[3]Department of Electrical Engineering
Indian Institute of Technology Kharagpur
Kharagpur – 721302, West Bengal, India
mukherjee.rohan666@gmail.com, mr.anupambanerjee@gmail.com, avirupmaulik@gmail.com, indrajit.chakrabarti@gmail.com, pkd@ee.iitkgp.ernet.in, akray@ece.iitkgp.ernet.in

Abstract – **The present research work introduces a novel VLSI design of Context-Based Adaptive Variable Length Coding (CAVLC) entropy encoder. The proposed design is intended to be used for H.264/AVC CAVLC entropy encoder. The architecture is designed to be used as a part of a comprehensive H.264 video coding system. This hardware works in parallel and pipelined fashion and demonstrates a fair trade-off between speed and area. The proposed CAVLC encoder performs at an optimum clock frequency of 134 MHz when incorporated in Xilinx 9.2i, Virtex-5 technology. It offers higher speed performance when compared to the existing CAVLC architectures. While considering HD-1080 format video sequence, the proposed architecture satisfies the required real time processing requirement.**

Keywords- CAVLC encoder; H.264; FPGA

I. INTRODUCTION

Entropy encoding plays an invaluable part in attaining compression in video coding standards. Entropy encoding takes care of the statistical redundancies in a sequence and eliminates them. We find that the MPEG, H.261/3 video coding standard use Variable Length Coding wherein the probability of occurrence of each symbol is determined and is assigned a codeword accordingly. H.264 [1], an advanced video coding standard uses context-based adaptive variable length coding (CAVLC) in order to augment the efficiency of entropy encoding by 5-10%. The CAVLC method takes advantage of inter symbol correlation in order to eliminate the statistical redundancy by introducing adaptability to the variable length coding tables. The generated table can be appropriately modified during the process of encoding according to the quantized DCT coefficients of the video sequence. The encoding efficiency of CAVLC is much superior to the different coding standards used previously owing primarily to its context adaptive nature of encoding. Needless to say the CAVLC has greater computational complexity. Owing to run-length coding in CAVLC, strings of zeros are efficiently represented. The strings of -1s or 1s present in the high-frequency coefficients are addressed as trailing ones (T1s). On careful examination we find that there exists a correlation between corresponding neighboring blocks and the non-zero coefficients. The methodology by which non-zero coefficients are encoded in the look-up table is primarily based on the measure of the newly coded DCT coefficients and on the neighboring blocks. Corresponding to each block, the codeword that is to be entropy encoded with the help of CAVLC consists of 5 parts- (i) The quantity of T1s and non-zero coefficients (Coeff_Token), (ii) the sign of individual T1s (iii) the total count of zeroes (iv) the magnitude of the left over non-zero coefficients (v) the run of zeroes for the earlier encoded non-zero coefficient [1, 2, 3]. A majority of the CAVLC encoders constitute a statistical buffer that contains the values for a look-up table (LUT) search procedure following the zigzag scan [4]. Eighty percent of this statistic buffer is consumed by the level and run first-in-first out buffers. In [5] Rahman *et al.* have reduced the area in the CAVLC encoder by removing the level and run FIFOs. Further optimization is obtained by using arithmetic table elimination (ATE) and split VLC LUT procedures. With the help of the two techniques the syntax elements are generated. Additionally the ATE technique is used to reduce the area. Using 6.5K logic gates in Virtex 2 FPGA the architecture can handle CIF videos in real time situations. In [6] Chien *et al.* have proposed an architecture wherein the throughput is enhanced by using serial-input-parallel-output (SIPO) buffers and a forward-biased parallel coding (FPC) approach. This architecture meets the necessary requirements of real-time processing of HD-1080 video format by using 9724 gates in 0.18 µm CMOS technology. In [7] Chen *et al.* have put forward a two-stage pipelined architecture that has adopted a zero skip technique. Although an increase in throughput is registered, an increase in chip area is observed due to the presence of the pipeline registers. In [8] Huang *et al.* have proposed an architecture that has considered statistical and regularity parameters and in process reduced the computation time for the "nC" parameter (a parameter defined by the standard for selection of VLC table). This procedure lowers the area needed and the proposed architecture can process real-time videos of HD-1080 format with the help of 16K gates in 0.18 µm CMOS technology.

In the current context, a novel architectural design for the CAVLC encoder has been put forward that considers the bit rate requirement of H.264. The intended design aspires to optimize the area by using a clock of high frequency. The proposed architecture is aimed at being used as a component

978-1-5090-1134-6/17/$31.00 ©2017 IEEE

of a complete system as a result of which the speed factor has been considered to be of utmost importance. In process we have simultaneously tried to achieve a fair tradeoff between speed and area and the comparison is provided in the results section. The CAVLC encoder works in a pipelined fashion wherein corresponding data elements are parallelly distributed among different encoding units. Parts of the codeword are produced simultaneously by different modules and are set in different set of registers. The final codes are assembled and packed by the packing units. Thus high speed data requirements for video conferencing can be achieved. The pipelined architecture increases the throughput. The latency is also reduced as the encoder does not wait for the codeword to be generated in its entirety before transmission. The look-up tables of each encoding module are kept in the ROM and are retrieved accordingly. Instead of keeping full look-up tables (LUTs) as necessitated by the H.264 standard, our proposed architecture generates and keeps modified LUTs in the ROM. The codes from the draft of H.264 are analyzed and it was found that some kind of common structure of words exists. The similarity in structure is implemented with the help of ROM and designed combinational circuits. This technique allows lesser values to be stored in the ROM and in process results in area reduction. Additionally, certain LUTs have also been replaced by select combinational circuits which save time that is necessarily needed to access the ROM. The proposed work also presents a novel algorithm for level encoding. The proposed design has been verified and synthesized for FPGA with the help of Xilinx 9.2i. The proposed architecture quite efficiently matches up to the real-time processing demand that is required for H.264 video encoding on HD 1080 format.

The remaining part of this paper consists of three sections. Section II presents the proposed architecture for the CAVLC encoder. Section III presents the results and compares the proposed architecture with other state of the art techniques. Finally, section IV concludes the work.

II. ARCHITECTURE

Fig. 1 presents the inputs and output corresponding to the VLSI execution of the CAVLC encoder. The encoder takes a 4×4 transformed, quantized block as input and gives a serial bit stream as an output. A ROM block stores the various look-up tables used by the CAVLC and is accessed as and when necessary. The encoder is also provided

with an asynchronous Reset pin to clear all the register contents if it is needed. Each quantized DCT coefficient is described using 8 bits of information. The output signal End of Code Word (EoCW) specifies the termination of encoding of one block, signaling that the subsequent set of inputs can be provided along with the internal reset signal generation.

The proposed architecture comprises of seven distinct modules as shown in Fig. 2. Five of the functional modules, namely, (1) Coeff_Token, (2) Trailing Ones, (3) Level, (4) Total Zero, (5) Run module constitute the 5 parts of the output codeword. The Output module (shown as VLC packer in Fig. 2) which is the sixth segment, presents a serial representation of the codeword and the seventh module, the

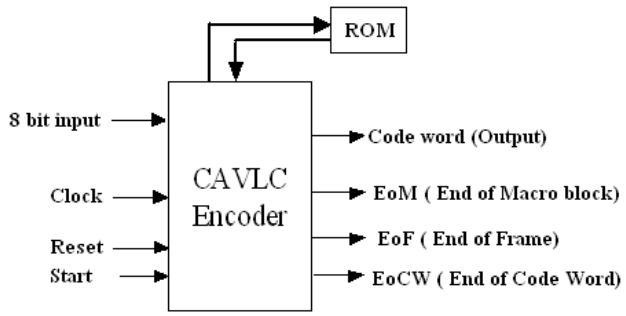

Fig. 1: CAVLC Encoder

FSM controller, generates the necessary control signals that trigger the remaining six modules at the appropriate time instances. Fig. 2 portrays the data components of the architecture. The residual coefficients which act as inputs are applied to the CAVLC scanning block in a zigzag fashion. Subsequently, elements are provided to the five different encoding modules in parallel. In the next step, the packing units assemble the final codes. On the basis of the number of non-zero coefficients present in the blocks situated to the top and to the left of the present block in the frame of the video sequence, the look-up table in order to encode Coeff_Token is determined.

The CAVLC encoder uses a buffer which maintains this information. The CAVLC encoder has numerous look-up tables and while a particular look up table transforms to another it is ensured that the value corresponding to the coded coefficient is in more than an empirically determined threshold. Based on the look-up table, the number of non-zero coefficients and the total number of T1s the ROM address is evaluated and the corresponding codeword is redeemed from it. On the basis of the sign of the T1 the Trailing One encoder appends a 0 or a 1 to the output codeword. A zero represents a positive sign, whereas a one represents a negative sign. Simultaneously, the Level encoder codes for the non-zero coefficients. The generated codeword is subsequently placed

into the output buffer in order to be transmitted in a serial manner. The Coeff_Token encoder section sends the total number of zeroes to be encoded to the Total Zero encoder. The necessary address is created and the codeword is redeemed from the ROM. The Run encoder acquires the run information corresponding to every non-zero coefficient from the Coeff_Token encoder. Finally, the codedwod is constructed serially with the help of the Output module. In order to produce the output codeword without any break, the output buffer is run at a slower clock speed in comparison to the speed considered to work on remaining modules. The slowdown is calibrated such that corresponding to the worst case (considering only 1 bit requires transmission from one module) the time is adequate to create the output corresponding to the remaining modules. In order to ensure that the output module transmits out the codeword serially without any stalling, it is configured to operate at a lower frequency than the other modules. The chief components of the proposed CAVLC encoder are described next.

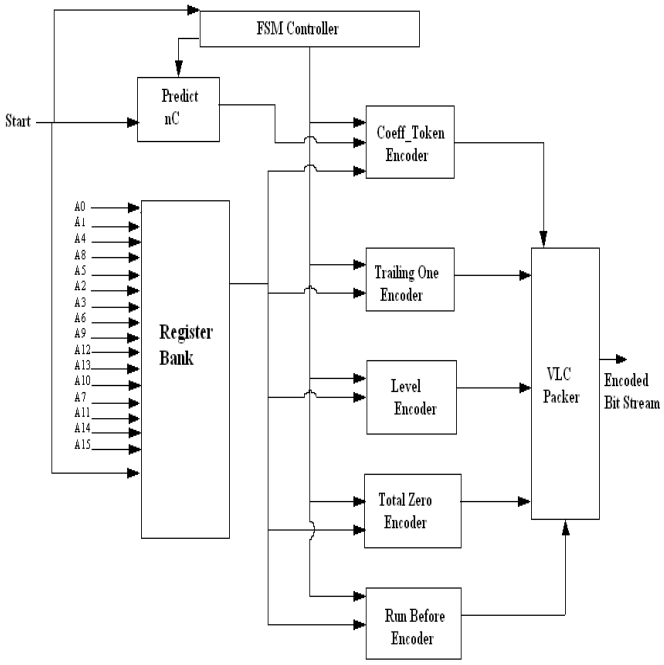

Fig. 2: Components of CAVLC Encoder

A. Coeff_Token Encoder

The Coeff_Token is the most important module of the CAVLC encoder shown in Fig. 3. It generates all the necessary data for the remaining modules. The Coeff_Token module receives as input a 4×4 array of transformed quantized coefficients in raster scan order. These coefficients are then converted to a zigzag order by a state counter. For each incoming coefficient 'in', the state counter specifies one of the sixteen locations in the register bank in a zigzag order. At every clock cycle, the coefficient that is read out of the register bank is specified by the 4-bit "mux select" select line. The individual bits of these coefficients are OR-ed together to determine if it is non zero or not. The signal is then passed on to the Non Zero counter which is a simple up counter. The Non Zero counter counts the aggregate of non-zero coefficients. The output of the register bank along with the appropriate control signal is passed on to the other sub-modules of the Coeff _Token module to produce the remaining values to be encoded. The block diagram of the T1 encoder is shown in Fig. 4. Going by the CAVLC standards, no more than 3 frequencies can be encoded as T1s. In the process of storing non-zero coefficients, corresponding coefficients are looked for 0, -1 or 1 and the moment one of the three are encountered, a T1 counter is accreted. If the value of the non-zero coefficient is not equal to 0, -1 or 1, the counter is reset. Consequentially, if the value of the counter is more than 3, the value is set to 3.

B. Trailing One Encoder

The Trailing One Encoder segment appends either a 0 or 1 to the output code word on the basis of the sign of the T1s. One stands for a negative sign and Zero stands for a positive sign. The blocks for the trailing one encoder are represented in Fig. 4.

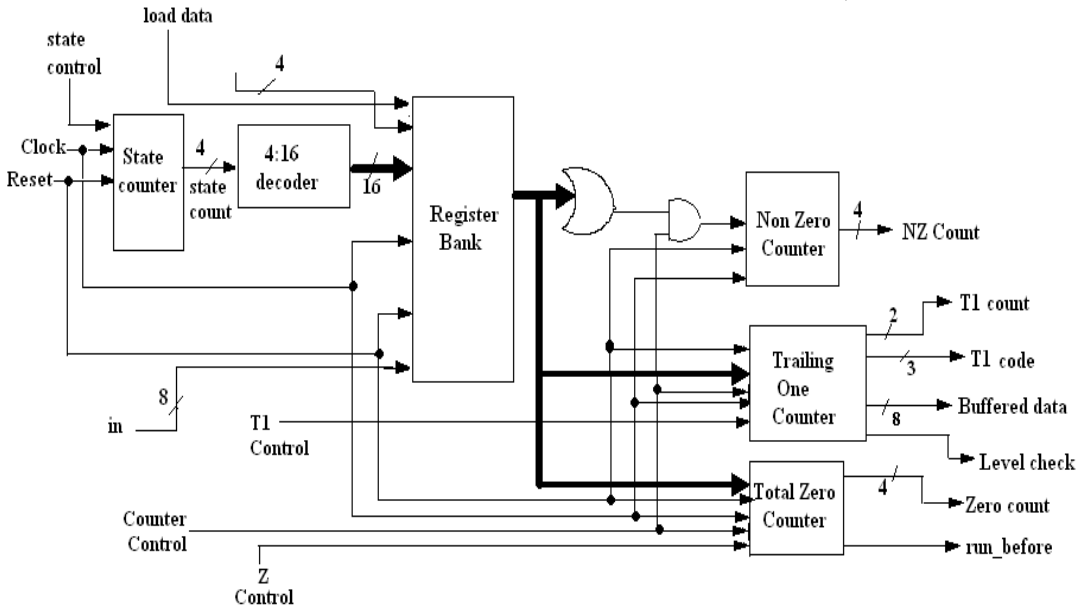

Fig. 3: Design of Coeff_Token Encoder

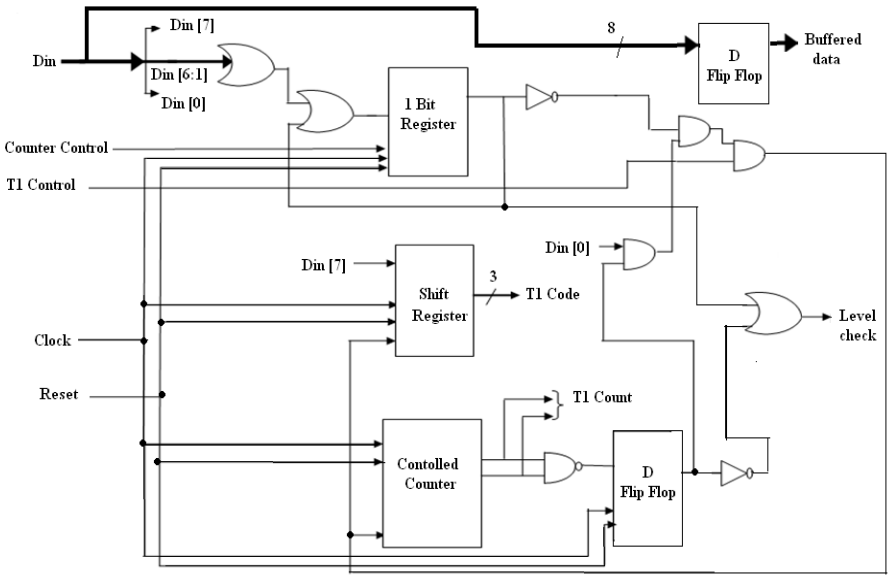

Fig. 4: Construction of Trailing One Encoder

C. *Level Encoder*

The Level Encoder encodes for the non-zero coefficients. According to the CAVLC algorithm, the look-up tables that are employed to encode each quantized DCT coefficient are changed based on the previously encoded coefficient. Seven look-up tables are considered and the transformation from one to another takes place once the value of the encoded coefficient is more than an empirically determined threshold. The seven look-up tables are created with the help of the level encoder wherein the non-zero coefficient is coded accordingly. The algorithm for level encoding is presented in Fig. 5 [9]. 'OC' and 'OL' stands for the magnitude and length of the output code word, 'vlctable' denotes the index number of the look-up table, 'mag-level' stands for the magnitude of the level, 'sign' corresponds to the sign of the encoded coefficient. For vlctable varying between 1 and 6 (included), 5 parameters are interpreted and are computed depending on the value of the encoded coefficient and on the corresponding index of the look-up table. The algorithm mentioned in Fig. 5 has been employed to evaluate the said parameters and in process a variable length table has been considered. The codeword generated is transferred into the output buffer in order to be transmitted serially. Once each block has been encoded, the magnitude of the current encoded coefficient is compared with the empirically determined threshold. Based on the comparison with the empirically determined threshold the index value of the look-up table is altered.

```
for vlctable = 0 do
  if mag-level < 8 then
    OL = 2 *mag-level + sign -1
    OC = 1
  end if
  if 8≤mag-level < 16 then
    OL = 19
    OC = 10000 | ((mag-level - 8) ≪1) |sign
  end if
  if mag-level ≥16 then
    OL = 28
    OC = (1 ≪12) | ((mag-level - 8) ≪ 1) | sign
  end if
end for

for 1≤ vlctable ≤6 do
  Shift = vlctable - 1
  Escape = (15≪ shift) + 1
  numPrefix = (mag-level -1) ≫shift
  suffmask = ~((-1) ≪shift)
  Suffix = (mag-level - 1) & suffmask
  if mag-level < escape then
    OL = numPrefix + vlctable + 1
    OC = (1 ≪(shift + 1)) | (suffix)≪ 1) | sign
  else
    OL = 28
    OC = (1 ≪12) | (escape)≪1) | sign
  end if
end for
```

Fig. 5: Algorithm for Level Encoding

D. *Total Zero Encoder*

Here we consider a 4×4 quantized block and the total number of zeroes in it is encoded by the total zero encoder. As the total

number of coefficients in the block is known, the zeroes succeeding the last non-zero coefficient are skipped. Consequently, the total number of zeroes succeeding the last non zero coefficient will equal to 16 - total non-zero coefficients - zeros before last non-zero coefficient. The encoding of Total Zeros is calculated depending on the total quantity of zeros and the total coefficients. The H.264 standard vouchsafes the fact that the length of the encoded bit stream present in the look-up table does not vary to a considerable extent. Hence, a combinatorial circuit is used to replace the LUT used in order to store the length of the encoded data. This strategy conserves the time needed to acquire the LUTs and in process simultaneously reduces the area.

E. Run Encoder

The total number of outstanding zeroes determines the encoding of the run parameter for each non-zero coefficient. The CAVLC algorithm takes help from different tables in order to encode the run information. The run encoder produces the appropriate address and retrieves the code word from the tables stored in ROM after having accessed the run information of each non-zero coefficient from the level buffer.

F. Output Encoder

This module has five buffers, each storing the code words from different encoder modules. Each of the five buffers gets the values from the corresponding modules in parallel as the encoding is done in parallel. The control signal from the modules to the buffers depends on the length of the codeword. It then transmits out the codeword serially, decrementing the value of the length buffer at every clock cycle. Once the codeword is transmitted out, it sends a trigger signal to FSM controller indicating that the current codeword has been transmitted completely. The FSM then triggers the next module which has to output its codeword and the process repeats. In order to ensure that the output module transmits the codeword serially, without any stalling, it is configured to operate at a lower frequency than the other modules.

G. Additional Buffer

To decide for the look-up tables to encode Coeff_Token, data of the blocks located to the left and the top of the present block, if available, are needed. Thus, we need a buffer size equal to the total numbers of blocks to store the information of all the blocks. The buffer size can be reduced by doing in-place computations. Using this method, a buffer size equal to the number of blocks in one row is only required. This is explained as follows. The information of a block, after encoding, is stored in the buffer at the position equal to the position of the block in the row. For the first block in the first row there are no blocks to be looked for, and thus the buffer is not accessed. The information of this block after encoding is stored in the buffer at index zero. The subsequent blocks in the first row contain only blocks located to their left to be looked. Since the blocks are encoded sequentially, accessing the buffer at positions left of the current index position would give the required information. For the blocks in the subsequent rows, the data of the block located to the left (if available) of the present encoding block is present in the buffer at a position one left to the present index position and the data of the block located at the top of the present encoding block is present in the buffer at present index position.

III. RESULTS

The architecture proposed in this paper is designed with the help of Verilog HDL. The simulation is performed using ISim (version 9.2i) and is synthesized using Xilinx ISE (version 9.2i). The targeted device for architecture is Virtex 5(5vlx110ff676-3). The critical path delay is obtained as 7.46 ns and the maximum operating frequency achieved is 134 MHz. Table I shows the comparison of the maximum achievable frequency with other existing works. The comparison vouchsafes for the gain in speed while considering relevant state of the art architectures.

TABLE I: COMPARISON WITH EXISTING WORKS

	Tech	Max. Frequency (MHz)	Target Format
Chen [7]	0.18µm	100	HD 1080@ 30 fps
Rahman [5]	Virtex-II	53	CIF @ 30fps
Huang [8]	0.18µm	100	HD 1080@ 30 fps
Kim [10]	FPGA	100	HD 1080@ 30 fps
Lai [11]	0.35µm	66	QCIF @ 30fps
Kim [12]	0.18µm	140	HD 1080@ 30 fps
Albanese [13]	Spartan 3	63	HD 1080@ 30 fps
Hmida [14]	FPGA	65	CIF @ 30fps
Joshi [16]	Virtex 4	143	CIF @ 30fps
Proposed	Virtex 5	134	HD 1080@ 30 fps

The architecture proposed by Huang et al. [8] when implemented on ASIC gives a much more increased speed than the FPGA implementation. The same can be concluded for this proposed work. The architecture by Hmida et al. [14] tries to reduce the hardware by considering a new technique of level encoding. But the design can operate at much lower clock speed. The design by Kim et al. [12] achieves a slight better frequency than the proposed work by parallel processing and performing the scanning in both direction using two sets of buffer. But this incurs an extra hardware cost. The design by Joshi et al. [16] gains a higher frequency at the cost of extra hardware resources. Although Albanese et al. have proposed [13] a new table compression technique which helps in reduction of area, the design cannot surpass the present work in terms of speed. The comparison of area utilization has been presented in Table II. The proposed architecture is synthesized using three different target platforms for correctness of comparisons. The concept of reducing the buffer size by doing in-place computation can give better result in terms of area reduction when implemented on ASIC.

TABLE II: COMPARISON OF AREA UTILIZATION WITH EXISTING WORKS

Tech.	Virtex 2		Virtex 5		Virtex 4	
	[5]	Present Work	[15]	Present Work	[16]	Present Work
Slices	-	-	13%	1%	4%	1%
CLBs	3 %	1%	-	-	-	-
LUTs	3%	1%	16%	1%	4%	1%
BRAMs	-	-	0.3%	0.6%	-	-
IOBs	46%	16%	-	-	13%	15%
DFFs	1%	1%	-	-	1%	1%

IV. CONCLUSION

This present body of work presents a new VLSI architecture of CAVLC encoder for H.264/AVC baseline profile encoder. Our simulation results demonstrate better speed in comparison to other state of the art technologies. While evaluating the performance of the proposed architecture with respect to the area factor, results indicate that the factor is considerably optimized. The optimization of the area parameter is reported since the present architecture is designed to be used as a part of a complete H.264 video coding system for video applications. In process the latency factor is also optimized. In conclusion, the proposed architecture effectively meets the real-time processing demand of H.264 video encoding on HD 1080 format.

REFERENCES

[1] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression—Video Coding for Next-Generation Multimedia.* New York: Wiley, 2003.

[2] Joint Video Team, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec H.264 and ISO/IEC 14496 AVC, March, 2005.

[3] N. Kamaci, and Y. Altunbasak, "Performance comparison of the emerging H.264 video coding standard with the existing standards", *Proceeding of IEEE International Conference on Multimedia & Expo,* vol.1, pp.345-348 July, 2003.

[4] T. Chen, Y. W. Huang, C. Y. Tsai et al., "Architecture design of context-based adaptive variable-length coding for h.264/avc", in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, pp. 832-836, Sept, 2006.

[5] C. A. Rahman, W. Badawy, "CAVLC Encoder Design for Real-Time Mobile Video Applications", in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol.64, pp. 873-877, Oct, 2007.

[6] C. D. Chien, K. P. Lu, Y. H. Shih, J. I. Guo, "A high performance cavlc encoder design for mpeg-4 avc/h.264 video coding applications", *Proceeding of International* Symposium on *Circuits and Systems,* pp. 3838 - 3841, May, 2006.

[7] T. Chen, Y. W. Huang, C. Y. Tsai et al., "Dual-block pipelined VLSI architecture of entropy coding for H.264/AVC baseline profile", *Proceeding of International Symposium on VLSI Design, Automation and Test,* pp. 271 - 274, April, 2005.

[8] F. M. Huang, S. F. Lei, "High performance and low cost entropy encoder for H.264/AVC baseline entropy coding", *Proceeding of International Conference on Communications, Circuits and Systems,* pp. 675 - 678, May, 2008.

[9] R. Mukherjee, I. Chakrabarti, S. Sengupta, "FPGA Based Architectural Implementation of Context-Based Adaptive Variable Length Coding (CAVLC) for H.264/AVC", Proceeding of IET International Conference on Information Science and Control Engineering, pp. 1-4, Dec, 2012.

[10] D. Kim, D. Har, E. Jung, H. Park, H. Shin, "'Implementation of High Performance CAVLC for H.264/AVC Video Codec", *Proceedings of System-on-Chip for Real-Time Applications, The 6th International Workshop*, pp. 20-23, Dec, 2006.

[11] Y. K Lai, C. C. Chou, Y. C. Chung, "A Simple and Cost Effective Video Encoder with Memory-Reducing CAVLC", *Proceeding of International Symposium on Circuits and System,* pp. 432 – 435, May, 2005.

[12] Y. J. Kim, K. Y. Wang, S. S. Lee, B. S. Kim, B. K. Choi, D. J. Chung, "Implementation of high efficient cavlc encoder for h.264/avc", *in IET Circuits, Devices & Systems,* vol. 3, pp. 116-124, June, 2009.

[13] L. F. Albanese, G. D. Licciardo, "An area reduced design of context-based adaptive variable length encoder suitable for embedded systems", *Proceedings of International Symposium on Communications and Mobile Network*, pp. 1– 4, Oct, 2010.

[14] A. B. Hmida, S. Dhahri, A. Zitoun, "A high performance architecture design of cavlc coding suitable for real-time applications", *Proceedings of World Cong. on Multimedia and Computer Science*, pp. 69– 74, 2013.

[15] F. L. L. Ramos, B. Zatt, T. L. Silva, A. Susin, S. Bampi, "A High Throughput CAVLC Hardware Architecture with Parallel Coefficient Processing for HDTV H.264/AVC Encoding", *Proceeding of 17th IEEE International Conference on Electronics, Circuits, and Systems,* pp. 587-590, Dec, 2010.

[16] A. M. Joshi, V. Mishra, R.M.Patrikar, "Low Complexity Hardware Implementation of Quantization and CAVLC for H.264 Encoder", *Proceeding of IEEE International Conference on Computational Intelligence and Computing Research,* pp. 1-5, Dec, 2014.