FPGA based High Performance CAVLC Implementation for H.264 Video Coding

Arun Kumar Pradhan
Trident Academy of Technology
Bhubaneswar,India

Lalit Kumar Kanoje
Trident Academy of Technology
Bhubaneswar,India

Biswa Ranjan Swain
Trident Academy of Technology
Bhubaneswar,India

ABSTRACT

Context-based adaptive variable-length coding (CAVLC) is an important feature of the latest video coding standard H.264/AVC. The coding technique using conventional CAVLC based on area efficient design, the second is on low power design architecture will lead to low throughput. In this paper, an efficient CAVLC design is proposed. The main concept is the FPGA based pipelining scheme for parallel processing of two 4x4 blocks. When one block is processed by the scanning engine to collect the required symbols, its previous block is handled by the coding engine to translate symbols into bit stream. Our block based pipelined architecture doubles the throughput of CAVLC at high bit rates. The proposed architecture can make a real time processing of 1920X1080 @ 30fps. With the synthesis constraint of a 200MHz clock using altera cyclone-II FPGA.

Keywords—Context-based adaptive variable-length coding (CAVLC), H.264/AVC, Zig-Zag Scanning, block pipeline.

1. INTRODUCTION

In order to achieve the higher compression ratio, Context-Based Adaptive Variable Length Coding (CAVLC) is adopted as one of entropy encoder in MPEG-4 AVC/H.264 [1], [2]. 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, it results a complex CAVLC encoding in hardware implementation. At the same time, throughput is the other concern in CAVLC, especially for higher resolution video such as HDTV. The encoder overall diagram is shown in figure 1.

While dealing with higher resolution of video information, encoding (or decoding) with an efficient CAVLC encoder is important. In this paper, an efficient and low power CAVLC encoder is proposed for video coding applications of MPEG-4 AVC/H.264. Two main concepts are proposed to increase the throughput. One is the combination of scan phase and coding phase; the other is the block-based pipelining by the associated input buffer.

As shown in figure 1 the encoder contains different blocks starting from video input to bit stream generation. In H.264 video encoder the output block is entropy coding, entropy coding is widely used in lossless compression. H.264/AVC adopted two approaches for entropy coding employed

One is CAVLC and CABAC. CAVLC is supported in all H.264 profiles, unlike CABAC which is not supported in Baseline and extended profiles. There are various profiles present like base line, main, extended base line supports intra and inter coding using I-Slices and P-Slices and entropy coding with CAVLC. The CAVLC designs can be classified into four types first is emphasizing on area efficient design, high performance, low power design, FPGA based design.

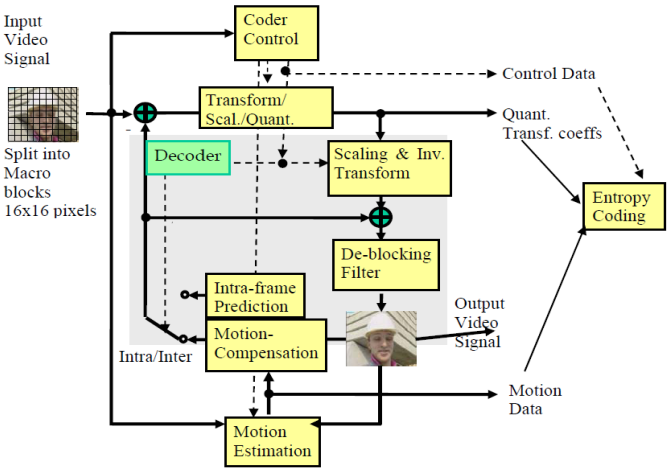

Figure 1 H.264 Encoder Diagram

2. H.264 CAVLC ALGORITHM OVERVIEW

Figure 2 shows the coding order of a macro block including Luma and chroma.CAVLC algorithm is used to encode transformed and quantized residual luminance and chrominance blocks in a macroblock in the order shown in Figure 2. Block -1 is formed by the DC coefficients of 4x4 luminance blocks only for the macroblocks that are coded in 16x16 Intra Mode. Blocks 16 and 17 are formed by the DC coefficients of 4x4 chrominance blocks for all the macro blocks. All the transformed and quantized 4x4 and 2x2 blocks for a macro block are given as inputs to CAVLC algorithm in the order shown in Figure 2. CAVLC algorithm processes each 4x4 block in zig-zag scan order and each 2x2 block in raster scan order. It encodes each block in the following five steps [2, 3, 4].

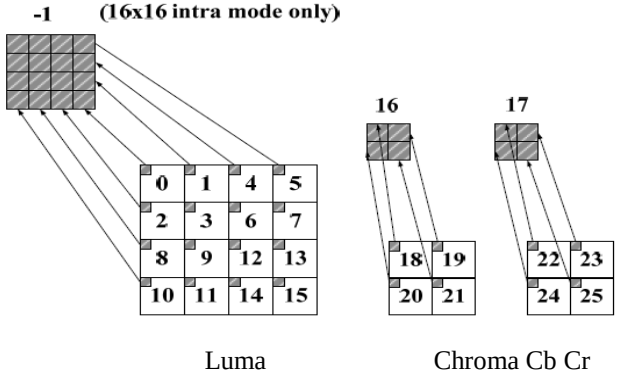

Figure 2 Coding Order of Blocks in a Macro block

A. Step 1: Coeff_Token-:

It generates coeff_token, the variable length code that encodes both the number of non-zero coefficients (TotalCoeff) and the number of trailing ±1 values (TrailingOnes) in a block. Since the highest non-zero coefficients after the zig-zag scan are often sequences of ±1, CAVLC algorithm encodes the number of high-frequency ±1 coefficients (TrailingOnes) in coeff_token. Since the number of non-zero coefficients in neighbouring blocks is correlated, CAVLC algorithm generates coeff_token for a block context adaptively. It uses one of the four different VLC tables for generating the coeff_token for a block based on the number of nonzero coefficients in the neighboring blocks as follows. Figure 3 shows the calculation of nC parameter based on the number of non-zero coefficients in the left-hand and upper previously coded blocks, nA and nB respectively.

nC = round ((nA + nB) /2).If both block is available.

nC = nB; If upper block is available.

nC = nA; if left block is available.

nC = 0 if neither is available.

As a special case, for 2x2 dc chroma blocks, nC is always set to -1. It, then, selects the VLC table that will be used for generating the coeff_token based on the value of nC as shown in Table I.

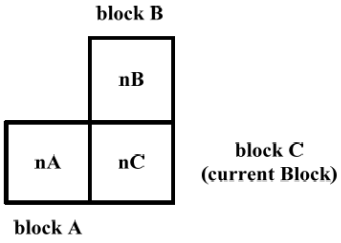

Figure 3 The relationship between block A, block B and block C

TABLE I VLC TABLE SELECTION

nC	VLC Table for coeff_token
0,1	Table 1
2,3	Table 2
4,5,6,7	Table 3
8 or above	Table 4

B. Step 2:TrailingOne Sign:-

It encodes the sign of each TrailingOne with a single bit in reverse order starting with the highest-frequency TrailingOne.

C. Step 3:Level :-

It encodes the level (sign and magnitude) of each remaining non-zero coefficient in the block in reverse order starting with the highest frequency coefficient and working back towards the DC coefficient. The code word for a level consists of a prefix and a suffix. Since the magnitude of non-zero coefficients tends to be larger near the DC coefficient and smaller towards the higher frequencies, CAVLC algorithm adapts the suffix length for the level parameter depending on recently-coded level magnitudes. It sets the suffix length for the first level, except in some special cases, to 0. It then increments the current suffix length, if the magnitude of the current level is larger than a predefined threshold for this suffix length. CAVLC algorithm generates the code length and the codeword for the current level based on its suffix length. When the suffix length for a level is 0, its codeword does not include a suffix. Otherwise, the codeword for the level includes a suffix. The codeword for a level always includes a prefix, but the prefix for a level is generated using different equations in the two cases; when the suffix length for the level is 0 versus when the suffix length for the level is greater than 0 [4].

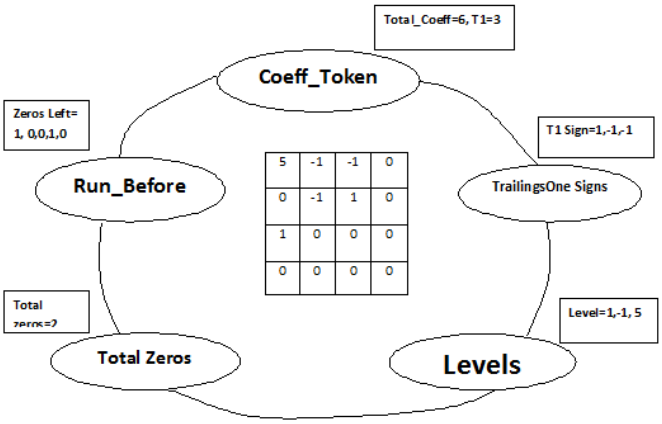

Figure 4 CAVLC Encoding Flow

D. Step 4: Total Zeros :-

It encodes the total number of zeros before the last non-zero coefficient (Total_Zeros) using a VLC table.Figure 4 shows the encoding flow of a CAVLC algorithm.

E. *Step 5:Run Before :-*

It encodes the number of zeros preceding each non-zero coefficient (Run_Before) in reverse order starting with the highest frequency coefficient. Since after transformation and quantization, blocks typically contain mostly zeros, CAVLC algorithm uses run level coding to represent strings of zeros compactly.

During encoding the residual data of a block, forward zigzag scan order is first processed; and then it is coded in the direction of backward scan order (from the highest ac value to dc value of current 4X 4 blocks). An example of CAVLC encoding is provided. In Fig. 5, the quantized transform coefficients of a 4 X 4 block are shown.

0	3	-1	0
0	-1	1	0
1	0	0	0
0	0	0	0

Figure 5 CAVLC Example

The reordered data of the block are in the form of (0,3,0,1,-1, -1,0,1, 0, 0, 0, 0, 0, 0, 0, 0). The encoding procedure of the example is also shown in Table II.

Table II CAVLC Encoding example

Parameter	Example (value)	Example (code)
coeff_token	TotalCoeff = 5 TrailingOnes=3	0000000110
Trailing_Ones sign flag	+, - ,-	0,1,1
level	Level(1) = +1 Level(0) = +3	1 0010
total_zero	3	111
run_before	Zerosleft= 3, Runb=1 Zerosleft= 2, Runb=0 Zerosleft= 2, Runb=0 Zerosleft= 2, Runb=1 Zerosleft= 1, Runb=1	10 1 1 01 No code

The five main parameters are encoded in sequence; and finally, the transmitted bit stream for this block is

0000100011100101111101101

3. PROPOSED HARDWARE ARCHITECTURE

The proposed CAVLC architecture includes the following main blocks. They are the input Register file, Reverse Zigzag Address Generation logic, nC generator, Trailing One Counter, Non-Zero Coefficient Counter, Total Zero Counter, Level counter, Run Before Regfile, Trailing One counter, Coeff_Token Table Selection Unit. The proposed over all architecture is shown in figure 7.

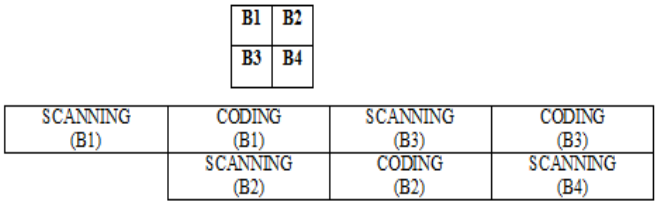

Figure 6 Scanning and Coding Pipe line phase

The CAVLC is one of the entropy coding methods which is used in the latest H.264 video standard, encodes the transformed quantized residual data. In a real time video encoder the residual data after quantization stored in a buffer. During the encoding of residual data, forward zigzag scan order from the dc value to the highest ac value of current 4 X 4 blocks is performed. So we proposed the dual-buffer with block pipelining architecture in which when one block of data is zigzag scanned at the same time other block data is coded to generate bit stream. Figure 6 shows the block pipe line stage of four blocks. At first block B1 is zigzag scaned then while coding of B1 the block B2 is scanned. So as shown in the figure 6 while buffer A used for coding buffer B is used for scanning of the new 4 x 4 block.

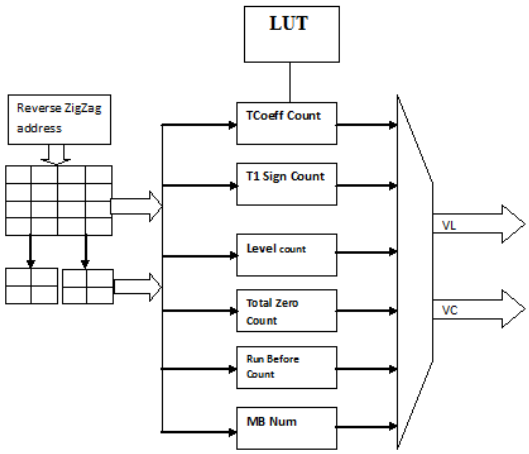

Figure 7 Overall proposed Architecture

F. VLC Counters and Reverse Zig-zag Ordering

CAVLC hardware 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 (TotalCoeff). TrailingOnes counter is used to store the number of trailing ±1 values (TrailingOnes). TotalZeros 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 TrailingOnes. TrailingOnes register file is used to store the sign of each TrailingOne. Level register file is used to store the level (sign and magnitude) of each non-zero coefficient other than the TrailingOnes. RunBefores register file is used to store the number of zeros preceding each non-zero coefficient. [4, 5]

CAVLC hardware begins the encoding for a 4x4 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. Figure 8 shows 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. [3, 5]

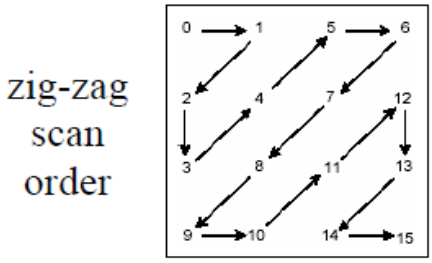

Figure 8 Zig-Zag Scan Process

It takes 16 cycles to read the coefficients and store the corresponding information in the counters and register files for each 4x4 luminance block in the macro blocks that are not coded in 16x16 Intra Mode. Figure 10 shows the block diagram of H.264/AVC baseline profile entropy coding engine. The same process takes 16 cycles for block -1 and 15 cycles for the other 4x4 luminance blocks in the macro blocks that are coded in 16x16 Intra Mode. Because DC coefficients in 4x4 luminance blocks in these macro blocks are coded in block -1. The same process takes 4 cycles for 2x2 chrominance blocks 16 and 17, and 15 cycles for the 4x4 chrominance blocks in all the macro blocks due to the same reason. [6, 7]

G. Zero Skipping by CBP Look-Ahead :

The symbol count of residues decreases with the increasing of quantization parameter. In this situation, the throughput of the dual-buffer architecture will be confined by the scanning Phase. To further improve our design, a zero-skipping technique is applied. When the residues within an 8 x 8 block are all zero, it is not necessary for the 4x4 blocks inside to be coded in this situation. We can save time and power by skipping the redundant scanning process. In this method, the CBP in the MB header is used for the skipping decision.

H. Real Time Requirement

In order to achieve the real time processing for H.264 video coding on HD1080 video coding the CAVLC encoder should run over 94 MHz when one transform quantized residual data is encoded per cycle. That is the average number of processing cycle per MB must be less than 384 cycle for meeting the real time processing requirement.

4. SIMULATION AND IMPLEMENTATION

The Basic dual-buffer architecture with block pipelining and zero skipping technique is used. Four sequences in the QCIF 30-fps format with different characteristics are used. Foreman is a general sequence with medium motion. Mobile calendar is highly textured and has complex motion. Weather has a static background and a sudden fast-moving person. Stefan has large global motions caused by the camera. Compared with the basic architecture, the dual-buffer architecture with the block-pipelining figure 11 scheme can process the scanning Phase and the coding phase of two neighboring 4 x 4 blocks in parallel and thus enhances the hardware utilization. It can almost half the processing cycles of CAVLC when the quantized residue energy is still large in high-bit-rate situations. However, when prediction is fine or in low-bit-rate situations, most residues are zero and the scanning phase dominates the processing cycles. The zero-skipping technique according to CBP can further improve the design by saving the redundant scanning.

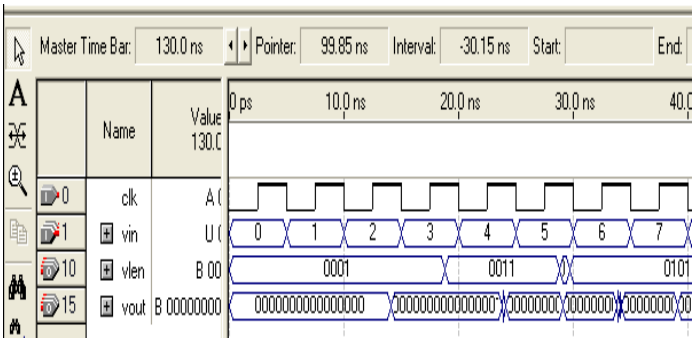

Figure 9 Simulation Waveform (Exgolomb Code)

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time
1	Worst-case tsu	N/A	None	5.699 ns
2	Worst-case tco	N/A	None	7.772 ns
3	Worst-case th	N/A	None	0.538 ns
4	Clock Setup: 'clk'	N/A	None	Restricted to 420.17 MHz (period = 2.380 ns)
5	Total number of failed paths			

Figure 10 Time Analyzer Summary (Exgolomb Code)

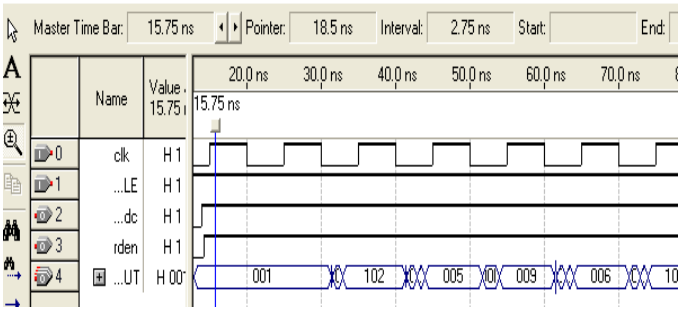

Figure 11 Simulation Waveform Zig Zag Scan

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time
1	Worst-case tsu	N/A	None	-0.286 ns
2	Worst-case tco	N/A	None	9.131 ns
3	Worst-case th	N/A	None	0.827 ns
4	Clock Setup: 'clk'	N/A	None	Restricted to 420.17 MHz (period = 2.380 ns)
5	Total number of failed paths			

Figure 12 Time Analyzer Summary (Zig Zag Scan)

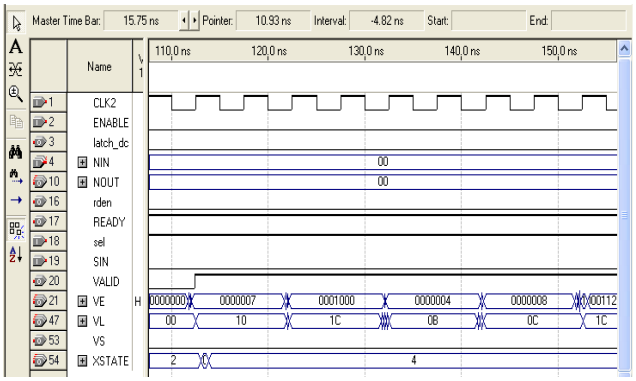

Figure 13 Simulation Waveform (Combine Block)

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time
1	Worst-case tsu	N/A	None	8.401 ns
2	Worst-case tco	N/A	None	8.825 ns
3	Worst-case th	N/A	None	-0.125 ns
4	Clock Setup: 'CLK'	N/A	None	105.17 MHz (period = 9.508 ns)
5	Clock Setup: 'CLK2'	N/A	None	203.33 MHz (period = 4.918 ns)
6	Total number of failed paths			

Figure 14 Time Analyzer Summary (Combine Block)

I. Implementation Results

The proposed entropy coding engine with dual-block pipelined architecture, zigzag address scan generator, the designed architecture is synthesized and simulated in 0.35 µm process technology using Synopsys tools. Table III shows the gate count profile. To achieve full hardware utilization by the dual-buffer architecture, two block statistic buffers are required. Two types of memories are required. The coefficient memory and bit stream memory are used as input and output buffers for system consideration. The upper 4 x 4 block total coefficient memory is used to store. The 2nd memory is used to process the next 4 x 4 block. Table 1 listed the throughput of proposed architecture.

TABLE II Throughput of Proposed Design with Others

Sl No	Frequency MHz	Frame format	Size	Proposed MB/s
1	200	QCIF	176x144	4856
2	200	CIF	352x288	1214
3	200	SDTV	1280x720	133
4	200	HDTV	1920x1080	59

Quartus II Version	9.0 Build 235 06/17/2009 SP 2 SJ Web Edition
Revision Name	final
Top-level Entity Name	final
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	1,939 / 33,216 (6 %)
Total combinational functions	1,727 / 33,216 (5 %)
Dedicated logic registers	600 / 33,216 (2 %)
Total registers	600
Total pins	53 / 475 (11 %)
Total virtual pins	0
Total memory bits	384 / 483,840 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 15 Compilation report of overall block

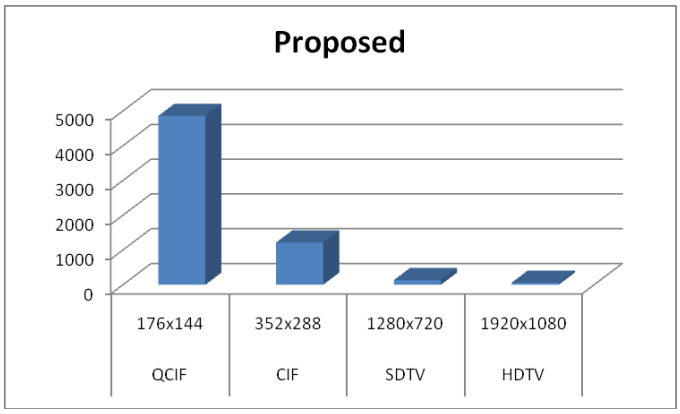

Figure 16 Result of proposed Architecture

5. CONCLUSIONS

This paper has focused on the high throughput of the CAVLC encoder with zigzag scanning. The proposed architecture has a better performance than known alternatives. It also achieves a high throughput, with a low-cost memory requirement and hardware complexity.

In this paper, we presented a high performance and throughput hardware architecture for real-time implementation of H.264 CAVLC encoder. The proposed architecture is implemented in VHDL using Altera Cyclone-II FPGA. The VHDL RTL code is verified to work at 200 MHz.

6. REFERENCES

[1] WIEGAND T., SULLIVAN G.J., BJONTEGARARD G., LUTHRA A.: 'Overview of the H.264/AVC video coding standard', IEEE Trans. Circuits Syst. Video Technol., 2003, 13, (7),pp. 506–576

[2] ITU-T Rec.H.264/ISO/IEC 14496-10: 'Advanced video coding', March 2005

[3] CHEN T.-C., HUANG Y.-W., TSAI C.-Y., HSIEH B.-Y., CHEN L.-G.: 'Architecture design of context-based adaptive variable length coding for H.264/AVC'. Proc. TCSII'06, September 2006, vol. 53, pp. 832–836

[4] CHIEN C.-D., LU K.-P., SHIH Y.-H., GUO J.-I.: 'A high performance CAVLC encoder design for MPEG-4 AVC/H.264 video coding applications'. Proc. ISCAS'06, May 2006, p. 4

[5] TSAI M.-C., CHANG T.-S.: 'High performance context adaptive variable length coding encoder for MPEG-4 AVC/H.264 video coding'. Proc. APCCAS'06, December 2006, pp. 586–589

[6] KIM D., JUNG E., PARK H., SHIN H., HAR D.: 'Implementation of high performance CAVLC for H.264/AVC video codec'. Proc. 6th Int. Workshop on System-on-Chip for Real-Time Applications, December 2006, pp. 20–23

[7] TSAI C.-Y., CHEN T.-C., CHEN L.-G.: 'Low power entropy coding hardware design for H.264/AVC baseline profile encoder'. Proc. ICME'06, July 2006, pp. 1941–1944

[8] Y.S. Yi and B.C. Song, "High Speed CAVLC Encoder for 1080p 60Hz H.264 CODEC," Proc. ISCAS 2008.

[9] C. D. Chien, K. P. Lu, Y. H. Shin, and J. I. Guo, "A High performance CAVLC Encoder Design for MPEG-4 AVC/H.264 Video Coding Applications," Proc. ISCAS 2006.

[10] T. C. Chen, Y. W. Huang, C. Y. Tsai, B. Y. Hsieh, and L. G. Chen, "Dual-block-pipelined VLSI Architecture of Entropy Coding for H.264/AVC Baseline Profile," Proc. International Symposium on VLSI Design, Automation and Test (VLSI-DAT), pp.271-274, 2005.

[11] Chang Su Han and Jae Hun Lee, "Area Efficient And High Throughput CAVLC Encoder For 1920x1080@30p H.264/Avc", SAMSUNG ELECTRONICS CO., LTD., South Korea;ICCE.org,p-1-7,2009.

[12] WeiJun Lu, Ying Li, DunShan Yu, Xing Zhang; 'Vlsi Implementation of an Entropy Encoder for H.264/AVC Baseline', Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on Digital Object Identifier: 10.1109/ICIEA.2008.4582753 Publication Year: 2008, Page(s): 1422 - 1425