A Two-way Parallel CAVLC Encoder for 4Kx2K H.264/AVC

Huibo Zhong, Sha Shen, Yibo Fan, Xiaoyang Zeng

State Key Lab of ASIC and System, Fudan University, Shanghai 201203, China
Email:fanyibo@fudan.edu.cn

Abstract

This paper presents a high performance design for Context-Based Adaptive Variable Length-Coding (CAVLC) used in the H.264/AVC standard. To reduce the cycles of processing one macroblock (MB), a two-stage residual encoder is proposed to make the scan and encode stage work simultaneously. The scan engine scans two coefficients at each cycle. Parallel encoder for two levels and parallel encoder for two runs are adopted to accelerate the encoder engine. Only 228 cycles at most are needed to process one MB. Due to the existence coded block pattern (CBP) decided skip block mode, our experiment shows only 160 cycles are needed on the average. The proposed CAVLC encoder can support 4Kx2K @30fps (frames per second) real-time encoding at 250 MHz and the gate count is only about 16k.

Index Terms: H.264/AVC, CAVLC, entropy encoding, level encoder

1. Introduction

H.264/AVC [1] [2] is the latest international video coding standard, this new standard significantly improves compression performance compared with previous standards because hosts of new technologies are used. H.264/AVC specifies two kinds of entropy coding methods that are Context-Based Adaptive Variable Length Coding (CAVLC) and Context-based Binary Arithmetic Coding (CABAC). CAVLC yields a higher coding efficiency than conventional VLC coding due to the context adaptive feature. On the other hand, because of the data dependency in CAVLC, it results in a complex encoding flow in hardware implementation.

As both the display size of flat-panel and video resolution become increasingly larger, encoders are desired to support higher bitrate video stream, however, the entropy coding process often becomes the bottleneck of the whole encoder since its bit-serial processing is hard to be speeded up by increasing parallelism or pipelining. Varieties of VLSI architectures for CAVLC were proposed in [3-5]. The design in [3] is capable of processing 1080@30fps video in real time, it requires 500 cycles for high-quality applications and about 200 cycles for low bitrate applications. Yi et al. [4] proposed a parallel structure and their work can process 1080@60fps video in real time (YUV=4:4:4), it requires about 323 cycles for some sequences. Literature [5] adopted a forward order for computing the CAVLC parameters and it takes 256 cycles to process 1080@60fps video (YUV=4:2:2). However, due to the complexities and data dependencies of CAVLC, it is hard to design an encoder to support high resolution applications such as 4Kx2K video or higher.

In this paper, to reduce the cycles of processing one macroblock (MB), we focus on increasing the hardware parallelism for CAVLC. A two-stage CAVLC residual encoder architecture is proposed to make the scan and encode stage work simultaneously. A paralleled scan engine which can scan two coefficients is proposed. In the same way, a paralleled "levels" encoder is proposed to accelerate the speed of encoder engine. Two "Run_before" symbols are encoded simultaneously with "levels". With the methods above, only 8 cycles are needed by each stage of the CAVLC. The proposed work is implemented in Verilog HDL and synthesized using SMIC 0.13um standard CMOS technology. The maximum operation frequency amounts to 250 MHz and the total logic gate count is only 16K. The proposed architecture can support 4Kx2K @30fps (frames per second) real-time encoding.

The rest of this paper is organized as follows. Section 2 briefly introduces the background of CAVLC. The proposed design is presented in section 3. Section 4 shows the performance analysis and comparison with previous works. And finally, conclusion is drawn in section 5.

2. Background of CAVLC

As mentioned above, in the H.264/AVC standard, two entropy coding methods (CAVLC and CABAC) are included. In the CAVLC entropy coding, only the quantized transform coefficients of the residual images are coded. Other information, i.e., macroblock (MB) header, is coded using Exp-Golomb codes. The detail of CAVLC is shown in [6]. CAVLC encoder encodes the elements listed as follows:

1) Coeff_token: the first VLC symbol in the CAVLC stream is Coeff_token. It encodes both the total number of nonzero coefficients (TotalCoeff) and the number of trailing ones (TrailingOnes). TotalCoeff ranged from 0 to 16, and the range of TrailingOnes is from 0 to 3. If the block has more than 3 trailing ones the rest are coded as

978-1-61284-193-9/11/$26.00 ©2011 IEEE

normal coefficient. There are five tables used for Coeff_token encoding. The selection of the tables is according to the value of nC which is calculated according to the upper and left coded 4x4 block.

2) Sign_trail: The signs of the trailing ones are the following symbols to be coded. Each sign is coded into one single bit. Bit 0 is assigned for positive and bit 1 is assigned for negative ones. The Sign_trail is coded in reverse zigzag scan order.

3) Levels: The nonzero coefficients excluding the trailing ones in reverse zigzag scan order are encoded as levels. Each level is encoded by one of the seven VLC table depending on the magnitude of each successive encoded level.

4) Total_zeros: Total_zeros encodes the total number of zero coefficients proceeding to the first nonzero coefficient in reverse zigzag scan order for each 4x4 (2x2) block. There are two separate VLC tables for the luma and chroma residual blocks.

5) Run_before: Run_before encodes the number of zeros preceding each nonzero coefficient in reverse zigzag order by a VLC table. There are two cases where the run_before of a coefficient does not to be encoded: either there are no more zeros left or this is the last coefficient of which run_before is left to encode.

Compared with conventional VLC coding, CAVLC yields a higher coding efficiencies due to the context-adaptive feature. However, this context-adaptive feature introduces many data dependencies in CAVLC such as : 1).the symbols mentioned above, such as Coeff_token, Sign_trail and so on, can only be obtained after the statistical process of each 4x4 block is finished, which makes the scan and encode stage hard to be paralleled. 2). To encode Coeff_token, nC should be determined at first, however, the value of nC is calculated according to the upper and left coded 4x4 block. 3). As for the levels in each 4x4 block, there are seven VLC table, the table selection depends on the previous table number and the magnitude of the successive encoded Level. Because of the data dependencies in CAVLC, it is hard to propose a CAVLC encoder for high resolution application such as 4Kx2K or larger.

3. The Proposed CAVLC Architecture

Fig.1 shows the proposed two-stage CAVLC residual encoder. In the scan stage, the Address Generator generates the scan address according to the MB type and CBP code. The residual coefficients are fetched from the residual buffer in the backward. Then the scan engine collects the required statistics such as TrailingOnes, TotalCoeffs, TotalZeros, and so on. The run-level symbols are extracted by the run-level detector and stored in the statistic Register Array. The encode engine encodes the symbols obtained by the scan engine by looking up corresponding tables. According to the encoded codewords, the bitstream packer chooses the codewords and packs them into bitstream. The following of this section describes the proposed CAVLC encoder in detail.

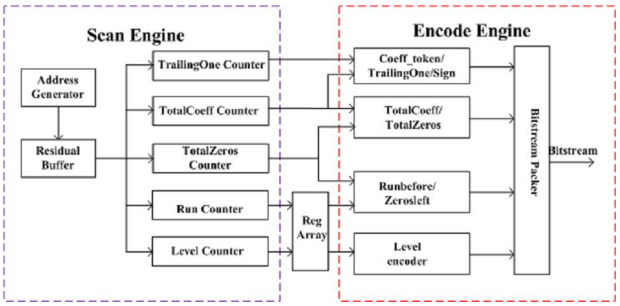

Fig.1 CAVLC Residual Encoder

3.1 Scan engine

Scan engine is used to figure out the CAVLC parameters. Generally, CAVLC parameters are calculated in reverse zigzag scan order. There are totally 384 coefficients inside one MB, so scan stage may become a bottleneck. To accelerate the speed of the whole encoder, a parallel structure for computing CAVLC parameters is necessary.

As shown in (1) and (2), s0, s1 are the state bits indicating whether the coefficient equals to zero. c0, c1 are the state bits indicating whether the coefficient equals to ±1. By the bits of these four states, all required CAVLC symbols can be figured out. According to s0, s1, Total_Coeff, TotalZeros, Run_before can be obtained by certain FSM. The computation of TrailingOnes is according to the value of c0, c1. The detail of the architecture is shown in Fig. 2.

$$\begin{cases} if\ coeff\_0 = 0, & s0 = 0 \\ else & s0 = 1 \\ if\ coeff\_1 = 0, & s1 = 0 \\ else & s1 = 1 \end{cases} \tag{1}$$

$$\begin{cases} if\ coeff\_0 = \pm 1, & c0 = 1 \\ else & c0 = 0 \\ if\ coeff\_1 = \pm 1, & c1 = 1 \\ else & c1 = 0 \end{cases} \tag{2}$$

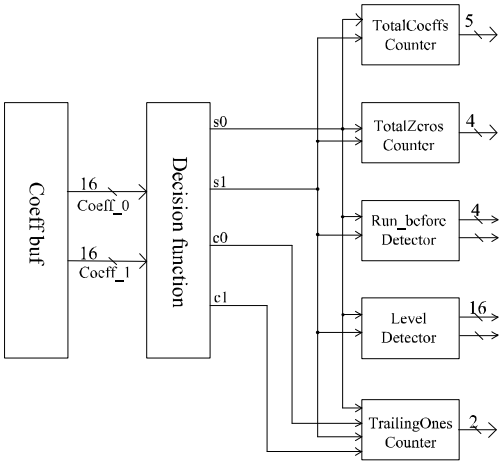

Fig.2 Parallel architecture for parameter computing

During the scan stage, the statistic data such as Levels and Run_before have to be stored in the buffer for the later encoding stage. In Chen's[3] work, a ping-pong buffer is used, by switching the ping-pong mode, scanning and coding of the 4x4 blocks within an MB can be processed simultaneously. However, the ping pong buffer makes the statistic buffer occupied the most part of the CAVLC encoder. A Register Array is adopted to reduce the required buffer size. Due to the scan engine and encode engine consume the same cycles, a fully pipelined process can be achieved by using only half size of the required ping-pong buffer used in [3].

3.2 Encode engine

The encode engine encodes the data which are obtained by the scan engine. It is comprised of Level encoder, Run_before encoder, Coeff_token and TrailingOne encoder, TotalZeros encoder. In H.264/AVC standard, seven VLC tables are used to define the codewords of level symbols. The number of execution cycles required for encoding level symbols depends on the number of nonzero coefficients in block. For a 4x4 block, there may be at most 16 levels. Therefore, when encoding a block data, the number of cycles required in coding stage may exceed that in scanning stage by serial symbol encoding. Many previous works encode the level symbols in sequential because the evaluation of each coefficient level depends on *vlc*, which is derived from the previously coded coefficient level as (3) shows. Let $inVLC \in \{0, 3, 6, 12, 24, 48, 65535\}$, *vlc* denotes the index of the VLC table. And to achieve higher throughput, parallel level encoding is necessary. The limit of data dependency can be effectively eliminated by using the VLCN look-ahead technique so that coefficients can be processed simultaneously.

$$\begin{aligned} &if\,(vlc = 0) \\ &\quad vlc++; \\ &if\,(abs(level) > incVLC[vlc]) \\ &\quad vlc++; \end{aligned} \tag{3}$$

Fig.3 shows the architecture of proposed levels coding element with VLCN look-ahead. In the proposed design, two level symbols are fetched from the Level Register Buffer simultaneously, and two level encoders are used for parallel encoding. Noted that the encode engine will consume the same cycles as the scan engine does if this structure is used. The controller of the level encoder controls the two level encoders according to the value of TotalCoeffs and TrailingOnes. VLCN look-ahead computes the two *vlc* codes for the corresponding level encoder according to the absolute value of the levels and previous *vlc*. The two level encoders then encode the levels and generate *level_prefix* and *level_suffix*. And finally the level codeword packer packs all the two pairs of *level_prefix* and *level_suffix* into level codeword.

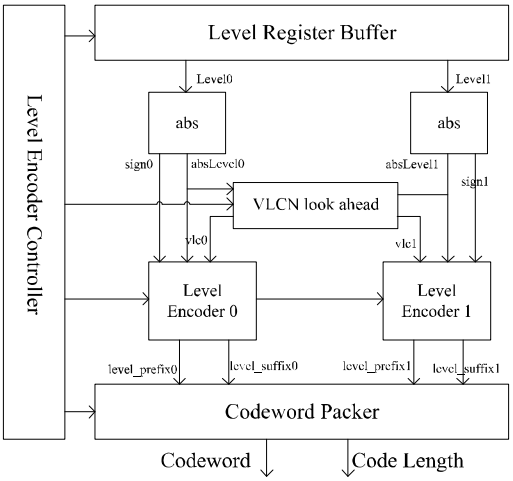

Fig.3 Proposed Parallel Level Encoder Architecture

In the same way, run_before encoder should also be processed parallel as level encoder does. Two Run_before symbols are fetched from Run Register array. Two codewords are produced by looking up the Run_Before&Zerosleft table. This codewords are packed and stored in a register until all the levels are encoded. Then run codeword is packed by the Bitstream Packer.

3.3 CAVLC architecture

After the description of all the sub modules of the CAVLC encoder, The CAVLC architecture is shown in Fig.4. The controller controls the Scan Engine to

calculate the CAVLC parameters; the results are stored in the Statistic Buffer. The whole symbol encoder can be divided into four parts: Exp_Golomb Encoder, Level Encoder, Coeff_Token and TrailingOne Encoder and TotalZeros & Run Encoder. The codewords encoded by the four parts are then chosen and packed by the Bitstream Packer. The area of the Statistic Buffer, Level Encoder and Bitstream Packer occupy about 90% of the CAVLC encoder.

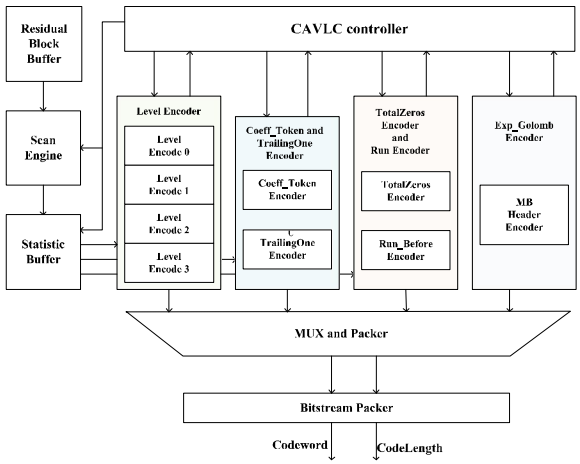

Fig. 4 Proposed CAVLC encoder Architecture

4. Performance Analysis and Comparison

The proposed architecture has been implemented in Verilog HDL, synthesized using SMIC 0.13um standard CMOS technology. The circuit occupies about 16k gates at a core speed of 250 MHz The comparison on hardware cost and processing speed of the proposed design with the existing design [3~5] is shown as table I.

Table I Comparison of the proposed design with others

	Chen[3]	Yi[4]	Hsia[5]	Proposed
technology (um)	0.18	0.09	0.18	0.13
frequency	100M	227M	125M	250M
Gate count	23K	66K	15K	16K
Speed (cycles/MB)	500	323	256/ 308*1	228
Max. Specication	1080 @30fps	1080 @60fps	1080 @30fps	4kx2k @30fps
Video format (YUV)	4:2:0	4:4:4	4:2:2	4:2:0

*1: the actual cycles computed by the method in [5].

From table I, we can see that the throughput of the proposed design is 228 cycles/MB, which is smaller than all [3~5] does. The design in [5] declare that their design only need 256 cycles, this number is estimate by (NC+4)x32 where NC denotes the none zero count in one 4x4 block and is set to 4, however the value of NC could be bigger for high quality video; the total 4x4 blocks in a MB presented in [5] is 32 which means it didn't include the Chorma_DC blocks, and the MB header encoding and core latency have not been reckoned with. By using the method in [5] we can figure out the actual cycles is 308 cycles, which is larger than our proposed. In practice, with the method using the coded block pattern (CBP) to determine whether the blocks need to be encoded or not, the average cycles will be smaller than the worst case. According to our experiment results, our design takes about 160 cycles to encode one MB on the average.

The hardware cost of our design is 16k, which is smaller than Chen's [3] and Yi's [4]. Hsia's [5] design did not including the MB header encoder, so it is a litter smaller than our work; however, our design achieves higher frequency and throughput.

5. Conclusion

This paper has presented a high-throughput CAVLC encoder for the H.264/AVC system. To reduce the required cycles of processing one MB, some parallel schemes are used. With those schemes, a new CAVLC encoder architecture is proposed. Compared with other previous methods, this architecture has the following advantage: 1) it greatly reduces the cycles to process one MB, at most only 228 cycles are needed. 2) It achieves high coding speed with a less hardware cost. This high-speed CAVLC encoder can process 4Kx2K@30fps video in real time.

References

[1] ITU-T, H.264, Advanced video coding for generic audiovisual services, March 2005.

[2] Joint Video Team (JVT) of ITU-T VCEG and ISO/ IEC MPEG, "Joint Model (JM) Reference Software Version 15.1*", http://iphome.hhi.de/suehring.*

[3] T. C. Chen, Y. W. Huang, C. Y. Tsai, B. Y. Hsieh, and L. G. Chen, "Architecture design of context-based adaptive variable-length coding for H.264/AVC," *IEEE Trans. on Circuits Syst. II, Exp. Briefs*, vol. 53, no. 9, pp. 832-836, Sep. 2006.

[4] Y. Yi and B. C. Song, "High-speed CAVLC encoder for 1080p 60-Hz H.264 codec," *IEEE Signal Process. Lett.*, vol. 15, pp. 891–894, 2008.

[5] S.C Hsia and W.S Liao, "Fowward Computations for Context-Adaptive Variable-Length Coding Design," *IEEE Trans. on Circuits Syst. II, Exp. Briefs*, vol.57 no.8, pp. 637-641, Aug 2010.

[6] G. Bjontegaardand K. Lillevold, "Context-adaptive VLC (CVLC) coding of coefficients," JVT Document JVT-C028. Fairfax, VA,2002.