The Hardware Design and Implementation for CAVLC and Exp-Golomb in H.264/AVC

Jing Chen[1],Yongliang Chen[1],Hantao Zhu[1],Chunchun Sui[1],PuFeng Wu[2],XixinCao[1]

School of Software and Microelectronics, Peking University[1]
The School of Management,Xi'an Jiaotong University[2]
Email: cxx@ss.pku.edu.cn, wupf@xaut.edu.cn

***Abstract*—Context-based adaptive variable-length coding (CAVLC) is a new and important feature of the latest video coding standard H.264/AVC. Together with Exp-Golomb(which is also called UVLC), CAVLC is used to code source data, i.e., quantized transformed coefficients (QTCs), and header data, respectively. In this paper, a hardware architecture for CAVLC and Exp-Golomb in the H.264/AVC codec is proposed, the main concept is the three-stage block pipelining scheme for this architecture. The simulation results show that our CAVLC and Exp-Golomb system can run at 31.2MHz and 156.6 MHz clock frequency respectively.**

***Keywords*—Context-based adaptive variable length coding, Exp-Golomb, H.264**

1. Introduction

The new video coding standard H.264/AVC [1] significantly outperforms previous standards in compression performance. It aims at a wide range of applications such as storage, entertainment, multimedia short message, videophone, videoconference, HDTV broadcasting, and internet streaming. Compared with MPEG-4, H.263, and MPEG-2, H.264/AVC can achieve 39%, 49%, and 64% bit-rate reductions, respectively [2].

CAVLC coding for quantized transform residues and Exp-Golomb coding for the other syntax elements are important in the baseline profile of the H.264/AVC codec since this profile targets at low complexity and low cost applications so that only CAVLC and Exp-Golomb are allowed to be the entropy coding tools.

In recent years, there are a bunch of papers dealing with researching and developing fast algorithms or Hardware acceleration programs adaptable to H. 264 /VAC standards, most of which concerned prediction and decoding, but few of them has focused on entropy coding of H. 264 /AVC[3-5]. This paper mainly concentrates on entropy coding algorithms of H. 264 standards and hardware realization of entropy coder.

This paper presents a hardware architecture of CAVLC and Exp-Golomb for H.264/AVC.The organization of this paper is as follows: The algorithm of CAVLC and Exp-Golomb are discussed in Section 2. The implementation of hardware design is described in Section 3. The simulation results are presented in Section 4. Finally, a summary is given in the last section.

2. Fundamentals

2.1 Exp-Golomb Algorithm

The Exp-Golomb algorithm is a variable length coder (VLC) algorithm used by H.264 and AVS. What the Exp-Golomb encoder processes is the syntax elements, for example, NAL (Network Abstraction Layer), SPS(sequence parameter set), PPS(picture parameters set), Sliceheader and Macroblock and so on. These are all syntax elements that have not been quantified. Compared with the CAVLC, the advantages of the Exp-Golomb algorithm are displayed at two aspects: the first advantage is that it has relatively low complexity of hardware, and it can analysis code bits without look-up table, according to the closure formula; the other is that it can determinate the K-step Exp-Golomb coding flexibly, according to the probability distribution of the coding element [1].

The Exp-Golomb algorithm is a kind of VLC with certain rules to construct bits. It can be described in five items as follows:

1) The value of "i" is determinated through the formula 1, if a variable value is code.

$$\sum_{j=0}^{i-1} 2^{j+k} \le code < \sum_{j=0}^{i} 2^{j+k} \qquad (1)$$

2) The prefix of the bits is made up of "i" zeros. The zero is just a sign, only the number of zero is meaningful.
3) The digit of the "1", as the separator, is inserted in the bits to differ from the digit of "0" from the step (2).
4) The tail of the bits can be calculated by the formula as follows:

$$code-\sum_{j=0}^{i-1}2^{j+k}$$

In H.264, we define k=0,it is clear that the structure of the bits follows the rules, which can be expressed as the formula has showed:

［M zeros］［1］［INFO］

INFO, which is M bits, includes the effective information. The first codeword does not include the prefix zero and the tail zero; the codeword of 1 and 2 have a INFO field of 1 bit; the codeword of 3~6 have a INFO field of 2 bits. Followed by analogy. The length of every codeword is 2M+1 bits.

$$M=\log_2(code\_num+1) \quad (2)$$

$$INFO=code\_num+1-2^M \quad (3)$$

There are three methods for the parameter v to map to the code_num.

1) ue(v): it is unsigned mapping, and code_num is v. It is used in parameters such as macro block type and reference frame index.

2) se(v): it is signed mapping, used in the residual of motion vector and quantitative parameters. The mapping rule is:

$$code\_num=2|v| \quad (v<0)$$

$$code\_num=2|v|-1 \quad (v>0)$$

3) me(v): it is mark mapping. The parameter v is mapping to code_num, according to a table which is defined in the criterion. That is used in the parameter of coded_block_pattrn. More details is showed in reference [1].

2.2 CAVLC

The quantized transform residues are coded by the CAVLC unit. The MB is divided into 4×4 blocks, and the 4×4 blocks are processed one after another in the defined order .In CAVLC, a block is scanned in a backward zig-zag order. Once a nonzero residue is found, the number of preceding zeros (known as run) and the nonzero residue (known as level) are transmitted as a joint symbol. An end-of-block (EOB) or related last-jointed run-level symbol will terminate the bitstream of a block if the remaining residues are zeros. The scanning phase and the coding phase can be fully pipelined for every residual pixel.

The first symbol represents the number of total nonzero coefficients (TotalCoeffs) and the number of trailing ones (Trailingones). The following symbols are signs of trailing ones, levels, number of total zeros (Zerolefts, excluding the zeros after the last nonzero coefficient in the forward zig-zag order),and runs. Since the last several nonzero residues are usually in the unit magnitude, the Trailingones s with sign symbols can replace the level symbols of trailing ones for fewer bits. With TotalCoeffs and Zerolefts, the EOB is not required. Besides, when the transmitted runs reaches Zerolefts, the runs for the rest levels must be zeros and thus can be saved. When the last level is reached, the run of the last level must be equal to Zerolefts minus all previous runs and thus can be saved as well.

In addition to the reduction of symbol amount, context-based adaptability is the most important key to improve the entropy coding performance. In CAVLC, each category of symbols has several context-based adaptive VLC tables, and the selection of these tables depends on the statistics of the block's content and previous transmitted symbols to match the most probable statistics [1].

3. Hardware Architecture Design

3.1 Hardware Architecture of Exp-Golomb

The hardware Sketch of Exp-Golomb is showed in Figure 1.

According to the two main sub branch, the sketch is divided into two parts by a red line. Firstly, let's look at the sub branch 1, which is on the left of the red line: syntax element adds 1, then we can get the exp_colomb. One of exp_colomb and syntax element is sent to a register of 16 bits, reg_syntax_info. This is decided by uvlc mode. After shifted left (16 – reg_len) bits, reg_syntax_info is sent to syntax_info, which is 16 bits.

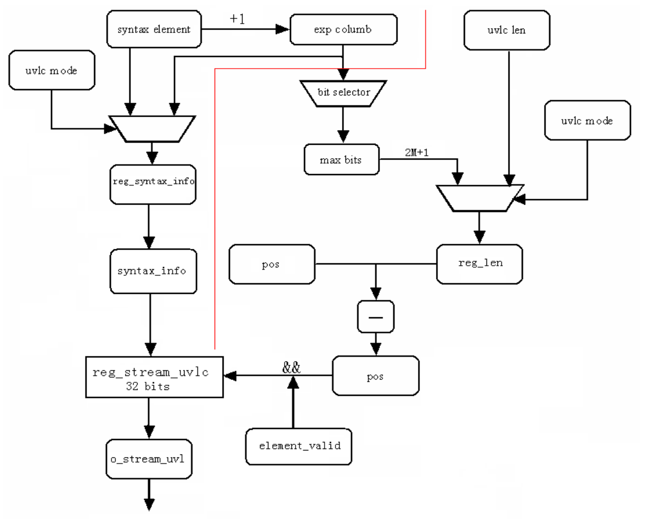

Figure1 The sketch of the Exp-Golomb Encoder

Then let's turn to the other sub branch, the right part of the red line. This is very important in Exp-Golomb. The length of the suffix, which is called max_bits, can be calculated by comparators. So the effective length of Exp-Golomb is (2*max_bits+1). Then the uvlc mode will decide which one is needed, uvlc_len or (2*max_bits+1). The result will be put in reg_len. A pointer variable is defined to point the position where syntax_info starts to store. When the next data arrives, and is stored at reg_stream_uvlc from the top digit, the pointer points at pos_reg_len.

3.2 Proposed Hardware Architecture Scheme Of CAVLC

The proposed hardware architechiture of CAVLC is showde as in Figure 2.In CAVLC unit, a MB is divided into 16 4×4 blocks, and the 4×4 blocks are processed one after another in the defined order. Each 4×4 block is processed through two phases, the scanning phase and the coding phase. In the scanning phase, the residues are read from the residue buffer in the backward zig-zag order. Then, the run-level symbols and required statistics are extracted by the level detector and stored in the statistic buffer. In the coding phase, the symbols are translated into codewords by the corresponding class of tables. The selection of VLC tables within a class is according to the related statistics and the previously transmitted symbols.

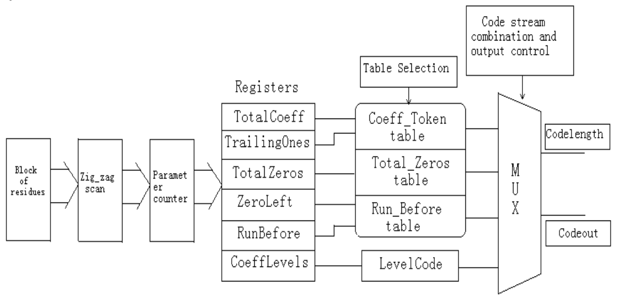

Figure 2 Proposed Hardware Architecture of CAVLC

The first stage, Zig_zag scan. Before coding a block of residues, these residues should be put in a right order by zig_zag scan. Here a reverse zig_zag scan is used directly to achieve it. It will take 16 clock cycles for a 4×4 block zig_zag scan.

The second stage, Parameter counter. There are 6 syntax elements that need to be coded in CAVLC unit, they are TotalCoeff、Trailingones、TotalZeros、ZeroLeft、RunBefore and CoeffLevels. And this unit will calculate these syntax elements and export some of them .

The last stage, Level Coding and Tables saving. In this section, VLC tables are saved by ROMs in order to access them random. And meanwhile some other ROMs are used to keep track of factor mark "coeff token". LevelCode hardware architecture is showde as in Figure 3.

In the Figure3, the "level table" unit is used to code nonzero coefficient(including prefix and length) and calculate codeword. When this unit starts to work, firstly it codes factor mark "coeff token" and enter it into bit stream packet unit. Then, ROM unit of bypass controller will detect residual factor to see whether it is 1 or not. If it is, the unit will neglect it and go on with next factor until it has detected "1" for 3 times, by then the resulting codes of symbol Trailingones(T1s) would be output to bit stream packet module. If, in this process, the nonzero coefficient detected is not 1, then export the detected codes of symbol Trailingones and shift to factor level coding.

After this, code the factors popped up from stack in sequence and until the pop-up value from stack is 0.

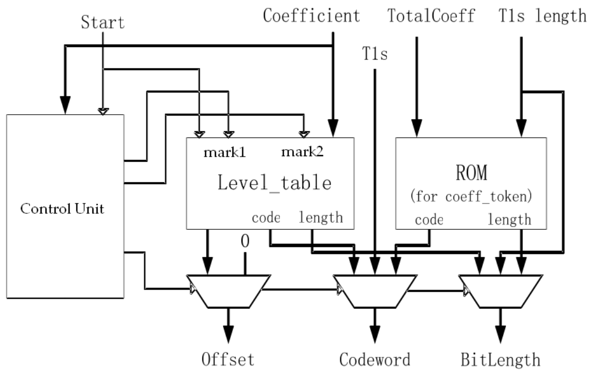

Figure 3 LevelCode Hardware Architecture Sketch

4. Simulation and Verification

We described our CAVLC and Exp-Golomb architecture design by Verilog HDL and synthesized the circuit using the EDA software—Synplify Pro 8.1 and find the estimated frequency of CAVLC and Exp-Golomb are 31.2MHz and 156.6 MHz. The proposed solution of reference[5] uses 84,902 LUTs with a coding clock of 31.9 MHz. However, lack of effective optimizations causes waste of hardware resources. Comparatively, the proposed solution of this paper achieves a better balance between hardware resources and performance. More details is showed as Table 1.

Table 1. FPGA Resource Comparison

FPGA resource				
	LUTS	REGS	Critical path/ns	CLK/MHz
Exp_Golomb	1207	643	4.94	202.429
Exp_Golomb in this papar	272	580	6.38	156.6
CAVLC in referenc[4]	84902	15622	31.36	31.9
CAVLC in this papar	5459	982	32.03	31.2

5. Conclusions

In this paper, a three-stage block pipelining architecture scheme for CAVLC and Exp-Golomb is proposed. The scheme is proposed to reduce hardware consumption and improve the performance. In order to achieve these aims of our design, a three-stage block pipelining architecture is proposed. The architecture proposed in this paper could be used for various application .

REFERENCES

[1] J. V. Team, "Draft ITU-T Recommendation and Final DraftInternational Standard of Joint Video Specification," , ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.
[2] A.Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G.J.Sullivan. " Performance comparison of video coding standards using Lagragian coder control," in Proc. ICIP, 2002, pp. 501–504.
[3] Tung-Chien Chen, Yu-Wen Huang. Architecture Design of Context-Based Adaptive Variable-Length Coding for H.264/AVC, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 53, NO. 9, SEPTEMBER 2006.
[4] Yanling Chen, Xixin Cao.A Memory-Efficient CAVLC Decoding Scheme for H.264/AVG Feb. 17-20, 2008 ICACT 2008
[5] CH IEN CH IH2DA, LU KENG2PO, SH IH YI2HUNG. A high performance CAVLC encoder design for MPEG24 AVC /H. 264 video coding applications[J]. ISCAS. Greece: [s. n.], 2006: 3838 - 3841.