Microelectronics Journal 67 (2017) 43–49

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

An efficient hardware architecture of CAVLC encoder based on stream processing

Milica Orlandić*, Kjetil Svarstad

Department of Electronic Systems NTNU - Norwegian University of Science and Technology Trondheim, Norway

ARTICLE INFO

Keywords:
H.264/AVC
Entropy encoding
CAVLC
FPGA
Bitstream packing
Hardware implementation.

ABSTRACT

The paper presents an efficient implementation of Context-Adaptive Variable Length Coding (CAVLC) entropy encoder in H.264/AVC standard. The architecture is designed with a parallel structure targeting real-time video compression. The intensive memory access demand in the syntax element coding stage is lowered by using the proposed arithmetic table elimination technique. The packing stage implementation is interleaved with syntax element generation stage and includes fast methods for syntax elements concatenation. The register update method performs concatenation of the bitstream of previously processed sub-blocks and the syntax codewords of the currently processed sub-block. The CAVLC encoder processes 4 × 4 sub-block coefficients in parallel, introducing the initial latency of 12 clock cycles, after which the full pipeline of the data encoding on the sub-block level is performed, and 16 residuals are processed at each clock cycle. The achieved high throughput allows the encoding core to perform real-time processing of 8 K UHD (4320 p) video sequences with a frame rate of 30 frames/s.

1. Introduction

Video compression is an intensive-computational application involving several stages such as transform coding, prediction algorithms and entropy coding. After the video sequence is compressed to a series of residuals in the prediction and transform stages, entropy encoding uses the statistical properties to compress data. The number of bits produced by entropy encoders is logarithmically proportional to the probability of the data. Entropy encoders are of serial nature due to data dependencies between the encoding elements, and achieving high throughput of entropy coding process represents a challenge nowadays. Existing video players, despite great efforts, cannot provide support for high bitrates dictated by increasing resolutions, and the design of high performance entropy encoder architectures on dedicated parallel hardware represents a solution.

The various types of information produced in the encoder stages, such as residuals from transform coding stage or mode flags from prediction stage, are referred as the syntax elements. In the H.264/AVC standard [1] a number of entropy encoding techniques are defined for different types of syntax elements and video standard profiles. Depending on the profile, residual encoding can be performed by Context-Adaptive Variable Length Coding (CAVLC) [2] and Context-Based Adaptive Binary Arithmetic Coding (CABAC) [3]. CABAC achieves bitrate savings when compared to CAVLC, but its higher complexity meets challenges in achieving efficient execution for high bitrate video content. Based on compression-complexity tradeoff compared to C-ABAC, CAVLC is deployed in the Baseline and Extended profile of H.264/AVC.

A number of hardware designs for CAVLC have been proposed in order to meet the high throughput requirements of high- definition coding. Recent works on CAVLC FPGA implementations [4–8] and ASIC designs [9–12] are reported in literature. Moon et al. [4] propose a solution for decoding *Run_before* codeword in video length decoding (VLD) without the look-up tables. Lo et al. [5] combine two entropy decoding methods in H.264/AVC standard in the shared implementation. The shared components between CABAC and CAVLC are context adaptation module, input and line buffers, whereas level computation in CAVLC level decoder and CABAC inverse binarization and look-up tables for other phases of entropy encoding within both standards are not adapted and merged into the common component. Ramos et al. [6] propose an implementation characterized by 2-pixel input parallelism, and the focus is on the speed up of syntax element encoding, in particular *Levels* encoding stage. Licciardo et al. [7] focus on minimizing area cost by using an arithmetic table elimination technique, however the operating frequency is the limiting factor for high definition content real-time processing. Hoffman et al. [8] present an implementation for surveillance applications with high frame rate characterized by a modified dual-coefficient scanning method. The level information and

* Corresponding author.
E-mail address: milica.orlandic@ntnu.no (M. Orlandić).

http://dx.doi.org/10.1016/j.mejo.2017.07.013
Received 2 March 2017; Received in revised form 30 May 2017; Accepted 24 July 2017

the number of zeros which run before each non-zero are determined during the scanning phase in order to reduce the number of clock cycles. Hsia et al. [11] propose a direct forward algorithm instead of backward tracking.

The paper is structured as follows: Section 2 presents details of C-AVLC, and in particular, suggested adaptations of computationally and resource demanding phases. The details of implementation of both encoding and packing phases are presented in Section 3. The overall supporting SoC architecture for testing, logic utilization and performance analysis are presented in Section 4. Finally, the conclusions are given in Section 5.

2. CAVLC encoding in H.264/AVC standard

Entropy coders in video compression standards convert a series of elements of video sequence such as transform coefficients, headers or motion vectors into bitstream suitable for transmission or storage. The CAVLC coding is based on the common Variable Length Coding (VLC) method which defines a codebook by assigning a code to each symbol. Average size of each coded symbol can be minimised by assigning shorter codes to the frequent symbols. A variation of VLC which introduces context-based adaptivity is defined in H.264/AVC standard. Context-based adaptivity introduces strong inter-symbol dependency and limits the use of parallelism in the encoder implementation. The data dependency exists among the coefficients on the 4×4 level, but also within the neighboring sub-blocks. The relationship between an actively encoded block and previously coded blocks is defined based on current block statistics. The CAVLC encoding process is partitioned into three phases: pre-processing including block scan and generation of flags and parameters, syntax element encoding and bitstream formation.

After the transform and the quantization stages, high-frequency regions typically contain coefficients with low values, whereas the levels of non-zero coefficients tend to be larger towards the low-frequency region. The zig-zag scan order proposed by the standard tends to group significant coefficients around DC coefficient as presented in Fig. 1. The coding parameters are extracted by backward tracking from the vector of coefficients obtained by zig-zag reordering.

The five syntax element codewords are defined by the standard as follows:

- **CoefToken**- Encoding of the number of total coefficients *TotalCoef* and the number of high-frequency +/- coefficients also called 'Trailing 1s' (*NR_T1*),
- **Sign**- Encoding of the sign of each *NR_T1*, where signs of *NR_T1* elements are coded with a single bit in reverse order from the highest frequency in *NR_T1*, and the length of *Sign* codeword is *NR_T1*.
- **Levels**- Encoding of the levels of the remaining non-zero coefficients.*Levels* are values of non-zero coefficients. Encoding of this phase is context-adaptive since the successive level coding depends on the magnitude of the previously coded level.

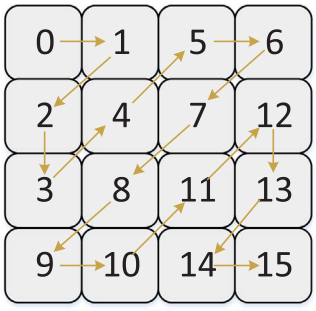

Fig. 1. Zigzag scan in H.264/AVC.

- **TotalZeros**- Encoding of the total number of zeros before the last coefficient.

Total number of zeros, *TotalZeros*, is the sum of zeros preceding the highest non-zero coefficient in the scan order array.

- **Run _ before**- Encoding runs of zeros, where the codeword is determined by the number of zeros in between two consecutive non-zero coefficient together with the number of remaining zeros in the vector.

Parameter *TotalCoef* can take values in a range $[0-16]$, whereas *NR_T1* is in the range $[0-3]$. Standard defines that if there are more than three trailing values, only last three are considered as *NR_T1*, and the other coefficients are coded as normal coefficients. From the implementation viewpoint, *CoefToken* is obtained from 2D variable-length code tables inherited from variable length coding in MPEG-2 based on the number of non-zero coefficients and on the number of trailing coefficients. There are four context-dependent NUM_VLCx look-up tables for *CoefToken* codeword generation for Luma components (three variable-length code tables and one fixed-length code table). The choice of a table is determined by statistics of the neighboring blocks and depends on a number of non-zero elements in the left and upper coded blocks.

The vector *Levels* is built as follows:

$$Code_{Levels} = [\underbrace{0\cdots0}_{p.length}1\underbrace{x\cdots xs}_{s.length}], \quad (1)$$

where *s* is the sign of the coefficients in the vector *Levels*. The string of zeros followed by a stop bit '1' is *Prefix*, whereas the sequence of bits following the stop bit is *Suffix* and the sign of the coefficient. The number of bits for *Prefix* and *Suffix* are given by parameters p_length and s_length. There are defined seven tables Lev_VLCx for coding levels syntax elements selected by an update step specified by parameter *N* and threshold values defined for each table.

The *TotalZeros* parameter counts the number of zeros in between the first non-zero coefficient and last non-zero coefficient. Depending on parameters *TotalCoef* and *TotalZeros* parameters, *TotalZeros* codeword is encoded using TZ_VLC look-up table - **TZ VLC**$(TotalCoef, TotalZeros+1)$.

The final codeword sequence, Run_before, is derived from a look-up table **RBVLC**$(1+Run(i), Zerospar)$, where the parameters *Run* and *Zerospar* are the number of zeros between each non-zero coefficient, and the number of embedded zeros yet to be encoded until the last non-zero element, respectively.

3. Implementation of the CAVLC entropy encoder

The block diagram of the proposed CAVLC encoding system architecture is given in Fig. 2. The zig-zag reorder scan module receives sixteen coefficients corresponding to the 4×4 quantized residual sub-block as inputs and performs coefficient reordering within one clock cycle. A number of flags required for the encoding process of the syntax elements are generated:

- *Flag Nonzero* - Non-zero elements,
- *Flag Ones* - Coefficients with value ±1,
- *Flag Sign* - Sign of the coefficients,
- *Flag FirstNonzero* - First nonzero element in reverse order,
- and *Flag TR1* - Trailing ones.

And a number of parameters used in the syntax element encoding phase are computed based on the set flags, such as:

- *Nonzero* - Non-zero coefficient vector,
- *TotalCoef* - Number of non-zero coefficients,