RFCAVLC8t: a Reference Frame Compression Algorithm for Video Coding Systems

Dieison Silveira, Mateus Grellert, Luciano Agostini
{dssilveira, mgdsilva, agostini}@inf.ufpel.edu.br

Group of Architectures and Integrated Circuits
Federal University of Pelotas

Abstract

This paper presents the Reference Frame Context Adaptive Variable-Length Compressor eight tables (RFCAVLC8t) for video coding systems. RFCAVLC8t aims to reduce the external memory bandwidth required to carry out this process. Six experiments were performed, all based on adaptations of the Huffman algorithm, and the best experiment achieved an average compression rate of more than 30% without any loss in quality for all targeted resolutions. This result is similar to the best solutions proposed in the literature, but it is the only one without losses in this process. The presented RFCAVLC8t splits the reference frames in 4x4 blocks and compresses these blocks using one of eight static code tables in a context-adaptive way.

1. Introduction

The recent technological innovations introduced an increasing demand for videos with better quality and higher resolutions. However, these videos require a large volume of data to be represented, and this information needs to be stored and eventually transmitted. In addition, digital videos are supported by many current electronic devices (such as smart phones and digital television set-top-boxes). With this scenario in mind, video coding represents a fundamental process in order to make digital videos support feasible. The video coding process comprehends a variety of tools and techniques that were and still are being developed, so there is a great research activity in this area. Through such techniques, digital videos are represented with a much smaller volume of data at the cost of little to no loss in quality. By associating a particular set of coding tools, video coding standards can be derived, from which H.264/AVC [1] is pointed out as the most recent consolidated video coding standard [2]. Moreover, HEVC (High Efficiency Video Coding) is the video coding standard that is currently being developed by expert professionals on the area [3]. The goal aimed by HEVC is to double the compression rate when compared to H.264/AVC, for the same video quality [3].

Motion Estimation (ME) is the most important process on recent video encoders, because most of the compression gains achieved during the entire coding process are produced at this stage. This is explained by the fact that ME is responsible for exploiting the similarities that are detected among temporal neighboring frames of a video and, since videos are usually displayed at a 30 fps rate, such similarities occur frequently [2]. Nevertheless, the compression gains introduced by ME are penalized by its complexity and by the significant amount of external memory bandwidth required for this process. This bandwidth comes from the fact that the encoded frames (called reference frames) must be stored in an off-chip memory to be accessed again when ME is applied on the next frames. For this reason, an encoder design must take into account that one of the greatest bottlenecks in this type of system is on this communication between the external memory and the ME core [4]. Besides being a bottleneck for the coding efficiency, external memory bandwidth also implies unwanted energy consumption, and this is a critical issue when battery-based devices are considered.

Many solutions that aim to reduce memory issues associated to ME can be found in the literature. Such solutions are based on two main approaches: (1) external memory bandwidth reduction through data reuse strategies using caches; and (2) bandwidth reduction via reference frames compression before they are stored in the off-chip memory. The main advantage of reference frame compression when compared to the other approach is that the former reduces the number of reading and writing accesses, while the latter reduces the number of reading accesses only. However, it is important to emphasize that both solutions can be used together, significantly decreasing the number of accesses to the external memory.

This paper presents the Reference Frame Context Adaptive Variable-Length Compressor (RFCAVLC8t), a solution based on reference frames compression, which is capable of reducing, on average, by more than 30% the volume of data that is stored in the off-chip memory, consequently decreasing the number of reading and writing operations required to access this data. The presented RFCAVLC8t is a hardware-friendly context adaptive coding based on Huffman algorithm [5]. Its coding process is performed without any information loss, avoiding drift. A software implementation of the proposed reference frame coder/decoder in C++ was used to analyze the gains in compression of each adaptation explored.

2. State of the art and related works

Video coding can be performed through different combinations of tools and techniques. Therefore, video-coding standards must be created in order to maintain the coding and decoding sides coherent. Such standards explicitly define which procedures are adopted in every coding and decoding step. The most recent video-coding standard is called H.264/AVC. This standard is capable of achieving greater compression rates than those of its predecessors, since it brought a number of innovations, such as quarter-precision Fractional Motion Estimation, Variable Block Sizes, among others. Nonetheless, it relies on high computational complexity and large numbers of external memory accesses in order to do so [4].

Aside from H.264/AVC, video-coding experts are currently developing a new standard, the HEVC, which has its final draft expected for January, 2012. HEVC also introduces a plethora of innovations to the video-coding process, e.g., new coding structures (Tree Units, Coding Units, among others), Variable Size Transforms, Arithmetic Entropy Coding, etc. Its beta release is already achieving compression rates close to 50% higher than H.264/AVC for the same quality, but its complexity is also higher [3].

Regarding digital videos, an important trend that must be noted is that resolution grows constantly. Besides significantly increasing the coding time needed to encode and decode a video, high resolutions represent an increase on the data volume that traffics between the ME core and the off-chip memory, consequently affecting coding performance. Moreover, there is a particular extension of H.264/AVC called MVC (Multiview Video Coding), related to compression of 3D videos, a kind of media that has obtained a significant interest on the market. 3D video coding adds a new kind of estimation during the coding process, the Disparity Estimation, which also requires a great off-chip memory bandwidth to be processed.

Within the current scenario, solutions that aim to reduce the off-chip memory bandwidth become imperative to any video coding system that prioritizes performance or that is energy-aware. Some solutions that address to this matter are found in the literature [6] [7]. The work proposed in [6] applies an in-loop compression of reference frames. This technique serves the dual purpose of reducing memory bandwidth and memory size. The particular algorithm that is used for compressing reference frames in this paper is a fixed-length compression algorithm based on block scalar quantization, called min-max scalar quantization (MMSQ) scheme. This scheme saves from 25% to 37.5% of the memory size used to store reference frames and an estimated 25% to 37.5% savings in memory transfer bandwidth, depending on which configuration is applied. Both solutions achieve little quality degradation, but the resolution of the videos used was 704x480 pixels, so results for 720p or higher resolution videos cannot be estimated.

Work [7] proposes a solution based on a previously proposed algorithm MMSQ [6]. The improvement proposed by [7] reduces the losses of the MMSQ process by storing them, so that these errors can be returned to the correspondent blocks during the Motion Compensation (MC) process. This solution is interesting, but it is lossy and it is not compatible with architectures that apply the ME and MC processes jointly. Architectures that implement ME/MC combined can skip the external memory access performed by MC by storing the residual block at the moment ME finds the best match for the current block, reducing the off-chip memory bandwidth of the coding process.

The main contribution of this work is to propose a lossless reference frame compression method, preserving video quality for any targeted resolution. Additionally, since memory accesses are known to consume a significant amount of energy in many digital systems, the RFCAVLC8t is a suitable solution for energy-aware video encoders. Furthermore, the proposed solution is compliant with any video-coding standard currently known.

3. Proposed Solution

The algorithm proposed in this work is a hardware-friendly adaptation of the Huffman coding algorithm, which was entitled Reference Frame Context Adaptive Variable-Length Compressor (RFCAVLC8t). Huffman algorithm is a famous data compression technique based on statistical methods widely used in image and text compressors [5].

Traditionally, Huffman is implemented using data trees, which are created following specific rules. Through such rules, the patterns of data that occur more often are coded with a smaller amount of bits. In order to do so, the algorithm requires a two-pass analysis: in the first pass, statistical calculations are performed; in the second one, the algorithm builds a tree based on the results acquired in the previous pass. This kind of implementation is not suitable for video-coding hardware systems, where throughput is a critical constraint. Therefore, hardware implementations of a Huffman coder are usually adapted using a static Huffman table to bypass the statistical analysis.

This work proposes a context adaptive solution using eight static Huffman table to compress the reference frames. Each table is suitable for a different set of frame features, so a decision strategy must be applied. This solution uses an idea similar to that presented in the CAVLC entropy encoder of the H.264/AVC standard [2], but in a simplified fashion.

For this work, every table contains the codes for each value of luminance sample. Since luminance samples are typically represented with eight bits per sample, each table has 256 entries, one for each possible luminance value. The values for each table were obtained separately and they correspond to the average occurrence of each luminance sample, according to an analysis with ten 1080p (1920x1080 pixels) sequences that presented different motion and lighting characteristics.

The average sample is computed for each block, and then one of eight tables is selected. In this case, 3 extra bits are needed to inform which table was used to encode the uncompressed information. Table decision is performed as follows: if the average sample calculated for the current block is between 0 and 31, then table 1 is selected; if the average is between 32 and 63, table 2 is used; when the value of the average sample is between 64 and 95, the third table is consulted; if the average is between 96 and 127, the block will be encoded according to the fourth table; if the average is between 128 and 159, table 5 is used; if the average is between 160 and 191, table 6 is consulted; if the average is between 192 and 223, table 7 is used; lastly, if the average is between 224 and 255, the block will be encoded according to the table 8.

Fig. 1 presents the flowchart of the RFCAVLC8t compression algorithm. The RFCAVLC8t fetches a 4x4 block and calculates the average sample value related to this block. Based on the resulting average, the most suitable table is used to compress the block. After that, the compressed block is stored in the external memory and the same process is applied for the next blocks of the frame.

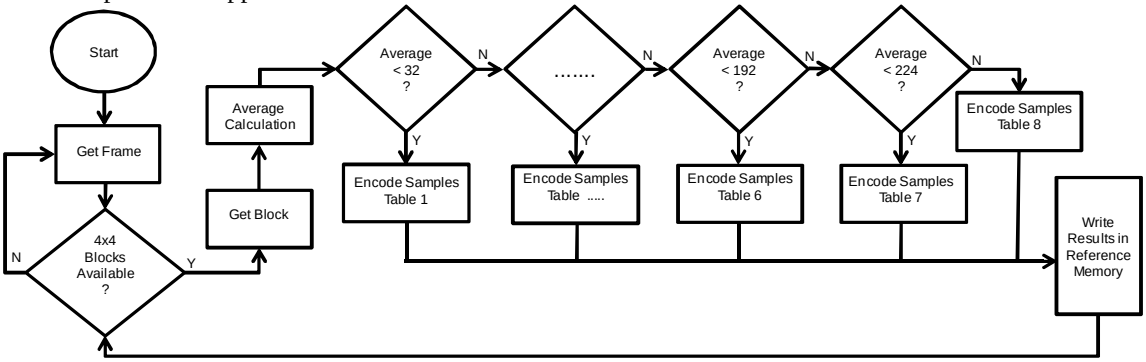

Figure 1. RFCAVLC8t table decision flowchart using 8 static tables

The decoding phase consists in reading the coded frame from the reference frames memory and applying the reverse process of the encoding phase. The frame is interpreted as a list of 32-bits words. Translation is performed using a greedy strategy, i.e., it consumes the bits from the list, reading one word per turn. The three first bits in the word are used to select the right table. The remaining bits are processed using this table. If the sequence of bits is fully consumed and the table did not reach its end, the next word is read from the external memory and the decoding continues until the end of the table is reached.

4. Results and Comparisons With Related Works

Table I presents the detailed results obtained with RFCAVLC8t for each video sequence cited before. The compression rates reached for each sequence indicate also the external memory bandwidth reduction achieved with RFCAVLC8t. As Table I shows, a compression rate variation occurs among different video sequences. This is explained by the fact that some sequences present a higher amount of homogeneous areas than others, which favors the compression achieved using the RFCAVLC8t solution. The remaining sequences achieved compression rates closer to the average of 30.38%. Among these eight sequences, four of them (In To Trees, Tennis, Rolling Tomatoes and Tractor) were not used in the analysis that was performed to generate the RFCAVLC8t tables. This was done in order to prove that this solution achieves good compression rates regardless of whether the sequence was used to generate the static tables or not.

Table 1. Results archived for RFCAVLC8t

Sequence	Compression Rate (%)	Sequence	Compression Rate(%)
Kimono	31.17	Tennis	31.71
In To Trees	30.55	Rolling Tomatoes	31.27
Station2	29.20	Sunflower	31.30
Pedestrian Area	30.33	Tractor	27.50
Average		**30.38%**	

Tab. 2 presents a bandwidth reduction comparison between RFCAVLC8t and related works. Two metrics were used in the comparison: the average bandwidth reduction, and the average quality degradation. It is important to note that this work is the only lossless solution among all compared works.

Table 2. Comparisons with related works

	This Work	**MMSQ 6bpp [6]**	**MMSQ 5bbp [6]**	**MMSQEC [7]**
Average Bandwidth Reduction	**30,38%**	25%	37,5%	23%
ΔPSNR	**0.00**	-0.043	-0.159	-0.01

Based on Tab. 2, it is possible to conclude that this work presents a competitive compression rate without introducing any quality loss. RFCAVLC8t achieved better results than work [7], even without introducing a difference in quality. Work [6] presented the greatest compression among others, but with a greater loss in quality as well. Additionally, both related works used only low-resolution sequences for the results that were presented, while the results presented in this work were all generated using high resolution videos. Another very important issue that was not cited in the related works is that the error generated in the reference frames will cause two collateral effects: (1) the references used in encoder and decoder sides will be different, and this can cause a very expressive quality degradation in the decoder side; and (2) the error inserted in the reference frames will be propagated among other frames of the video sequence. Thus, the evaluation of the PSNR loss comparing the reference frame before and after the compression process is not a robust metric to evaluate the impacts that this loss can cause in the global coding system.

5. Conclusion and Future Works

This work presented a solution for memory bandwidth reduction in video coding systems through reference frames compression. The RFCAVLC8t uses a lossless context adaptive variable-length method with eight static code tables and blocks with 4x4 samples to reach the desired bandwidth reduction. The decision of which table will be used is simple and it is based on the average value of each 4x4 bock. RFCAVLC8t achieves an average memory bandwidth reduction of more than 30%. This bandwidth is reduced for both reading and writing operations. This method is inspired in the Huffman coding algorithm and it is fully compliant with state-of-the-art coding standards, such as the H.264/AVC and the HEVC.

Considering all presented results, it is possible to infer that the solution presented in this paper introduces positive impacts on throughput and energy consumption as well when used in a complete video coder system.

Comparing with related works, this is the only solution that does not insert degradations in the video process, since the used method is lossless. Even with the use of lossless compression, the RFCAVLC8t is able to reach a compression rate and a bandwidth reduction near to that presented in related works.

As future work, the proposed module is implemented using a hardware description language, such as VHDL. This description will be synthesized to FPGA hardware and its performance will be evaluated.

6. References

[1] JVT of ITU-T VCEG and ISO/IEC MPEG, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[2] I. E. G. Richardson, H.264/AVC and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia. Chichester: John Wiley and Sons, 2003.

[3] Joint Collaborative Team on Video Coding (JCT-VC), available at < http://phenix.int-evry.fr/jct/index.php> December, 2011.

[4] L. V. Agostini, “Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H.264/AVC”. Ph.D. dissertation, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil, 2007.

[5] D. Salomon, Data Compression – The Complete Reference. 3rd ed. New York: Springer, 2004.

[6] M. Budagavi, M. Zhou. “Video Coding Using Compressed Reference Frames”. IEEE ICASSP 2008, pp. 1165-1168 May 2008, Las Vegas, NV, USA.

[7] A. D. Gupte, B. Amrutur, M. M. Mehendale, A. V. Rao, and M. Budagavi, “Memory Bandwidth and Power Reduction Using Lossy Reference Frame Compression in Video Encoding”, IEEE TCSVT, vol. 21, no. 2, pp.225-230, February 2011, Bengaluru, India.

[8] Institut für Informationsverarbeitung: Test media, available at <ftp://ftp.tnt.uni-hannover.de/>, January, 2012.