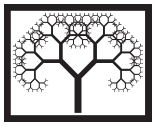

Proceedings of the Sixth International Conference on
Parallel, Distributed, GPU and Cloud Computing for Engineering,
P. Iványi and B.H.V Topping (Editors)
Civil-Comp Press, Stirlingshire, Scotland

Paper 29

Evaluation of FPGA-based motion estimation module for HEVC video coding standard

O. LÓPEZ[1], R. GUTIERREZ[1], E. ALCOCER[1], V. GALIANO[1], H. MIGALLON[1], G. VAN WALLENDAEL[2], M.P. MALUMBRES[1]
[1]Miguel Hernández University, Spain
[2]Multimedia Lab Ghent University, Belgium

Abstract

Motion estimation is one of the most complex task of HEVC video encoder, requiring most of the encoding time mainly due to (a) a large set of Coding Tree Unit partitioning modes, (b) the presence of multiple reference frames, and (c) the varying size of Coding Units in comparison with its predecessor H264/AVC. Besides, HEVC uses Variable Block Size Motion Estimation to obtain advanced coding efficiency. In this work, we design and evaluate a hardware IME design when applied to a particular System-On-Chip platform, taking into account the impact on the Rate/Distortion performance when applying different CTU sizes and the effect of the DMA transfers in the CTU encoding time.

Results show that the overall video encoding time could be reduced a 77% when using our hardware IME module, being this kind of accelerators an interesting cost effective solution to speed-up last generation video encoders.

Keywords: HEVC, FPGA, video coding, motion estimation, special purpose hardware

1 Introduction

The High Efficiency Video Coding (HEVC) standard [1] was launched on January 2013 by the Joint Collaborative Team on Video Coding (JCT-VC) in order to replace the previous H.264/AVC [2] standard with the intention to overcome with nowadays and future multimedia market trends like 4K and 8K definition video content and high quality color depth at 10 bit. HEVC improves the coding efficiency over its predecessor (H.264/AVC) by a factor of almost twice for an equivalent visual quality [3].

Regarding complexity, HEVC encoder is several times more complex than the H.264/AVC encoder [4]. As in previous video standards, Motion Estimation (ME) is the most complex task of encoder, requiring more than 90% of the encoding time [5]. In HEVC, ME is even more complex due to several issues such as the use of a large set of Coding Tree Unit (CTU) partitioning modes and the presence of multiple reference frames in comparison with previous H264/AVC video coding standard. Besides, HEVC adopts Variable Block Size Motion

Estimation (VBSME) to obtain advanced coding efficiency, but it comes at the expense of an increment in the computational complexity.

Several hardware architectures has been proposed in the literature in order to speed up the HEVC ME module to reduce the overall encoder complexity. The Integer-pel Motion Estimation (IME) block is in charge of motion estimation and in most of the state-of-the-art proposals, the IME hardware architectures are only focused on the motion search algorithm since it takes most of the computational time of the IME block. Usually, the most used motion search algorithm in hardware implementations is the Full Search (FS) algorithm. It searches at all points of the established search area of a reference frame, and, as a consequence, it is able to provide an optimal result, obtaining a vector that minimizes the residual error of the current CTU prediction).

Authors in [5–9] present an IME hardware block using FS strategy. In [5], a Sum of Absolute Differences (SAD) unit on a Field-Programmable Gate Array (FPGA) is proposed, being able to test all partition modes of a CTU except the set of asymmetric partition modes. Authors fixed a search area size lower than the one established by the HEVC standard, being able to run as fast as 30 fps at 2k video resolutions. In the hardware module presented in [7], the maximum CTU size is reduced to 32x32 with a search area size of ± 23 pixels. This architecture is implemented on an FPGA device and achieves 30 fps at 1080p video resolutions. In [8], different search areas are studied, achieving a maximum frame rate of 57 fps for a 720p video resolution.

In a previous work [10], authors presented a new hardware architecture that performs IME computation using FPGA technology, which is based in two innovative techniques: The first one is a new SAD adder tree structure, and the second one is a new memory scan order. This proposal is able to achieve encoding frame rates up to 116 fps and 30 fps at 2K and 4K video resolutions, respectively. The new SAD adder tree structure performs the additions at the first level of the CTU quad tree, starting from the maximum size of the CTU, and halving the amount of additions at the next quad tree levels. This approach is different from the rest of state-of-the-art approaches where usually the CTU is first divided into smaller blocks for consecutive accumulations, keeping the same additions in each step and thus requiring a higher number of steps to acquire all SADs. With that new proposal, authors took advantage of the resources provided by the FPGA, obtaining the minimum possible latency when calculating SADs of all levels and partitions for a CTU. In this manner, SADs that corresponds to asymmetric partitions are obtained in a fast and efficient way. With respect to the second innovative technique, the new memory scan order, a series of reconfigurable shift registers and processing elements are responsible for storing the necessary pixels of both reference and current frames, keeping them always available to calculate both the SADs and Motion Vectors (MVs) of a CTU. In this way, authors avoid external memory accesses since available data are highly reused by reconfiguring the displacement in an efficient way.

In this work, we implemented and evaluate the IME design presented in [10] when applied to a specific evaluation board. We have slightly modified the design to adapt it to the selected board to finally provide three implementations: (a) one unit working with a maximum CTU size of 32x32 pixels (CTU_32), (b) four CTU_32 units working in parallel, and (c) one unit working with a maximum CTU size of 64x64 pixels (CTU_64). All of them have fixed the motion search area size to the one proposed in the standard. We have evaluated the three versions, studying the hardware resources required, clock frequency operation, the end-to-end delay chain of the hardware modules (Input DMA transfer, motion estimation computation

and output DMA transfer), and the R/D performance of hardware proposals with respect the software ones.

The rest of the paper is organized as follows. Section 2 presents a brief overview of the architecture design while in Section 3, numerical experiments analyzing the results of our hardware design over an evaluation board are presented. Finally, in Section 4 some conclusions and future work are drawn.

2 Hardware Architecture Description

In this section, we present a brief overview of a complete IME design in a System-On-Chip (SoC) platform that consist of two well-defined parts, a Processing System (PS) based on an ARM processor and several hard peripherals like Ethernet, USB, etc., and a FPGA Programmable Logic (PL). Our architecture has been modeled in VHDL, and it has been synthesized, simulated, implemented, and tested on the Xilinx SoC, Zynq-7 Mini-ITX Motherboard XC7Z100 (xc7z100ffg900-2). The correctness of our design was tested and verified with the HEVC HM 14 reference model [11].

In the proposed architecture, ARM processor manages the transfer between the IME SAD module and a Double Data Rate (DDR) memory which stores both reference and current frames, by a Direct Memory Access (DMA) module. ARM processor works at 666.66 MHz, and the DDR at 533.33 MHz, whereas the clock frequency of the PL is restricted by the maximum frequency of the SAD HEVC module which is the responsible for the IME calculation.

Regarding the IME process, each video frame is subdivided and partitioned into basic coding units called CTUs. The coding structure in HEVC consists of CUs with a maximum size of 64x64 pixels, as large as that of CTUs (Coding Tree Units), which can be recursively divided in picture squares until achieving a block size of 8x8 pixels. Each coding unit (CU) consists of Prediction Units (PUs) whose size can vary from the maximum size of the CU to 4x8 or 8x4 for Inter prediction, supporting 8 partitioning modes [12]. In our proposal, the SAD HEVC module responsible for IME calculation can be configured to work with CTU sizes of 64x64 and 32x32. In the case of 64x64 CTU size, the PL can work at 200 MHz whereas with a 32x32 CTU size the PL clock frequency was restricted by the evaluation board used, 250 MHz, although the module could work at a maximum frequency of 333 MHz. Therefore the maximum frequency is limited to 200MHz.

Our SAD HEVC module consists of (a) internal memory areas to allocate CU pixels of current frame and the pixels belonging to the search area in the reference frame, (b) a distortion block where pixels belonging to both CUs are subtracted, (c) a Sum of Absolute Difference (SAD) adder tree block, and (e) an accumulative comparator block that saves the minimum SAD values and its corresponding MVs for all CU partitions, as shown in Figure 1. For more details of this module, see [10].

In Table 1, we show the hardware resources used to implement our SAD HEVC module for maximum CTU sizes of 64x64 and 32x32, respectively, on a Zynq-7 Mini-ITX Motherboard XC7Z100 FPGA. As shown, our SAD HEVC module requires a 56.5% and 15.3% of the total used area for CTU_64 and CTU_32 implementations, respectively. Furthermore, the 4xCTU_32 implementation is able to compute 4 CTUs of 32x32 at the same time using 59.7% of the available area.

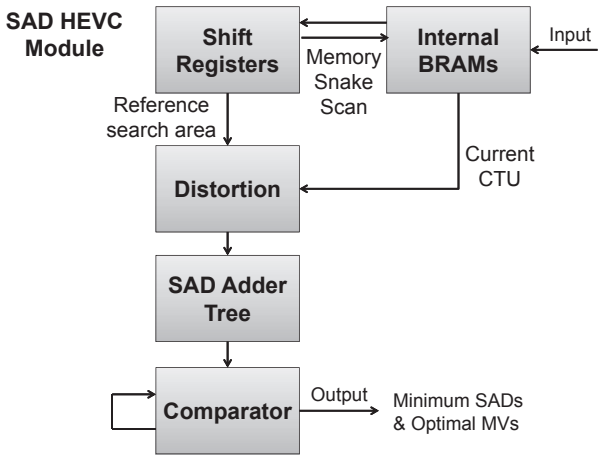

Figure 1: Hardware SAD HEVC module

Table 1: Utilization resources on Mini-ITX

Resources	CTU_64	CTU_32	4xCTU_32	Available
LUTs	156880	42405	165700	277400
Flip-flops	180249	50466	190240	554800
Block-RAMs	41,5	25,5	102	755

3 Numerical experiments

Motion estimation is a task integrated in the Inter Prediction module of the HEVC encoder. We have considered the Low Delay P (LD-P) coding mode in our evaluation test. In LD-P coding mode, the first picture is encoded as an intra-picture, whereas the following pictures are encoded as generalized unidirectional pictures (inter). This coding structure is the most popular for video conferencing, designed for interactive real-time communication.

Given the previous configuration, we have performed several experiments of the HEVC IME in order to observe how the CTU size impact on the R/D performance and coding complexity of the HEVC encoder. We have chosen CTU sizes of 64x64 and 32x32, and their corresponding Search Range (SR) sizes as 100% of the CTU size. In addition, four video sequences from the HEVC common conditions video set were selected: RaceHorses (832x480-30 fps), ParkScene (1920x1080-24 fps), Traffic (2560x1600-30 fps) and PeopleOnStreet (2560x1600-30 fps). In order to perform these tests, we have used the HEVC HM 14 reference model [11]. The HEVC reference software was compiled with Visual Studio 2010 and run over a PC platform with an Intel Core i7-6800K CPU 3.40GHz with 16GB RAM.

First of all, we have analyzed the impact of the CTU size on the R/D performance using the Bjontegaard metric (BD-rate) [13]. So as to obtain the BD-rate we have compressed all tested video sequences at four compression levels (QP values): 22, 27, 32, and 37. The R/D curve used as reference will be the one obtained with the HEVC reference software using a 64x64 CTU size. As can be seen in Table 2 better performance is obtained when using CTUs of 64x64 size. The use of lower CTU sizes has a penalty in R/D of up to 4.3% of BD-rate, being more noticeable for higher resolution video sequences.

Table 2: % BD-Rate comparison between CTU sizes of 64x64 and 32x32

Video Sequence	% BD-Rate
RaceHorses	2.4
ParkScene	3.8
PeopleOnStreet	3.7
Traffic	4.3

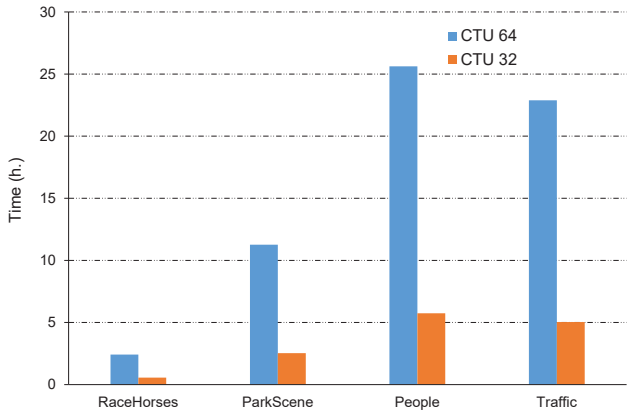

Figure 2: Encoding time (30 frames) using both CTU sizes 64x64 and 32x32 for all tested video sequences.

After the R/D evaluation, we will measure the impact of CTU size in the encoding time. In Figure 2 we show the total time required to encode 30 frames (1 second) at a QP 37 of all tested video sequences using CTU sizes of 64x64 and 32x32. As shown, the total encoding time when the maximum CTU size is set to 64x64 is more than 4 times than the time required for the case of 32x32 CTU size, requiring more than 22 hours to compress a 4k video sequence. Also, in Figure 3, we show the percentage of the overall encoding time that was spent by HEVC reference software using the FS algorithm to perform the ME, when the R/D optimization is disabled. Depending on both the maximum CTU size and the quantization parameter (QP), the percentage of time spent by the ME process ranges from 85% to 97.8% in our tests. Those results agree with the ones obtained in [5]. As expected, the encoder spends more time in the IME module when the CTU size is higher. Therefore, a hardware design performing the IME computation in a fast and accurate way makes sense in order to reduce the overall encoding time as much as possible.

In order to evaluate the impact of the inclusion of our hardware IME proposal in the HEVC, we have measured the number of cycles required for a) transferring both current CTU and its reference window, b) IME processing and c) transferring SADs and motion vectors (MVs) of all PUs. In Table 3 we show the number of cycles required to perform the IME process for a given CTU, where, as previously said, all SADs and MVs of all prediction units including asymmetric partitions are calculated at the same time. So, the time required to process one CTU, will be the time required to transfer the CTU and the search area, the processing time required by our HEVC SAD module, and the time required to return the processing results. As shown, the most part of the CTU coding time corresponds to the DMA transfer of both the CTU and the corresponding search window of the reference frame form the DDR memory to

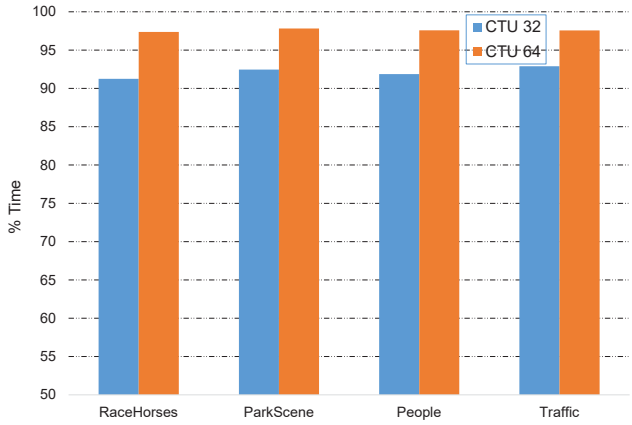

Figure 3: Percentage of SW reference encoding time required for SAD module with a Full-Search strategy.

the internal memory of the hardware IME. In Table 3 we can see the total time required by our hardware IME module to perform the motion estimation process of a given CTU. Considering those times, the hardware IME module is able to process 13959 CTUs per second in the 32x32 size case and 3264 in the 64x64 size case running at 200MHz.

Table 3: Required cycles and time for the hardware IME

	DMA send	DMA receive	SADs and MVs computing	Total cycles	Total Time (ms) at 200MHz
CTU_32	10069	151	4139	14359	0.07179
CTU_64	44276	605	16400	61281	0.30640

Table 4: Reference frames used in a GOP

Frame 1	-1 -5 -9 -13
Frame 2	-1 -2 -6 -10
Frame 3	-1 -3 -7 -11
Frame 4	-1 -4 -8 -12

As stated before, in LD-P coding mode, the first frame is encoded as an I-frame and the rest of frames are encoded as P frames. P frames perform the motion estimation using previous encoded and decoded pictures as reference frames. In HEVC, P frames can use multiple reference frames. In this paper, a group of pictures (GOP) consists of four P frames where each one can use up to four reference frames. In Table 4 we show the references used by each P frame. At the beginning of encoding a video sequence, the first P frame has only one available reference, the I frame (relative picture order is -1). Second, third and fourth P frames will use two reference frames. However, after encoding several GOPs, P frames inside that GOPs will use four reference frames. As previously mentioned, the times shown in Table 3, are the ones required to perform the motion estimation of one CTU over one reference frame. So, when multiple reference frames are available, that motion estimation process must be repeated for the different reference frames, reducing up to 4 times the number of CTUs per second that can be computed if four reference frames are used.

Table 5: Frames per second computed by hardware IME at 200 MHz

video	CTU_32		4xCTU_32	CTU_64	
resolution	CTUs	fps	fps	CTUs	fps
832x480	390	8.93	35.71	104	7.85
1920x1080	2040	1.70	6.83	510	1.60
2560x1600	4000	0.87	3.48	1000	0.82

In order to deal with that situation, we have developed a SAD HEVC module containing 4 CTU_32 modules, each one with each own DMA channel, which is able to compute 4 CTUs of 32x32 at the same time. Performing the motion estimation over the four reference frames at the same time will reduce the encoding time a 77% on average. Table 5 shows the number of frames per second (fps) that can be computed for different resolution video sequences if our IME module were used. Also shown in Table 5, the number of CTUs per frame to process depending on both the video resolution and the CTU size. As can be seen, using the 4xCTU_32 module, we can compute 4 times more fps, reaching real-time encoding for the lowest resolution video sequence.

4 Conclusion

In this work, we have presented a full hardware IME architecture. We have analyzed how the CTU size impact on the HEVC performance in terms of Rate/Distortion and processing time. As shown, there are small differences in R/D, being 4.3% the maximum BD-rate increment when we use a CTU size of 32x32 for higher resolution video sequences. Regarding complexity, CTU size have a noticeable impact over the total encoding time, being the faster configuration the one that uses a 32x32 CTU size. Remark, that the ME module requires between 85% to 97.8% of the total encoding time, so our IME hardware module will significantly reduce the overall encoding time.

Respect to the evaluation of the hardware IME proposal, we have measured the time required by DMA transfers as well as the computing time. The results show that up to 13928 and 3263 CTUs per second can be processed for CTU sizes of 32x32 and 64x64, respectively, working at 200MHz. Also, it can be observed that the system bottleneck resides in the DMA transfer process, requiring more than 70% of the total processing time of a single CTU.

Furthermore, we have presented a SAD HEVC module for the case of 32x32 CTU size, containing 4 IME modules, each one with each own DMA channel. This new module is able to compute 4 CTUs of 32x32 at the same time. Using this hardware module on a HEVC LD-P coding mode configuration, we can reduce a 77% on average the total encoding time, because this module is able to compute the motion estimation for all reference frames at the same time.

As future work, we will reduce the DMA transfer time, reusing part of the search area for the IME calculation in contiguous CUs, only transferring the new reference pixels required at every moment and packing in a 64 bit word eight pixels of the reference search area.

Acknowledgment

This research was supported by the Spanish Ministry of Economy and Competitiveness under Grant TIN2015-66972-C5-4-R co-financed by FEDER funds.

References

[1] B. Bross, W. Han, J. Ohm, G. Sullivan, Y.-K. Wang, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 10," *Document JCTVC-L1003 of JCT-VC, Geneva*, January 2013.

[2] ITU-T and ISO/IEC JTC 1, "Advanced video coding for generic audiovisual services," *ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC) version 16, 2012*, 2012.

[3] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *Circuits and systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1648 –1667, December 2012.

[4] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1685–1696, 2012.

[5] A. Medhat, A. Shalaby, M. S. Sayed, and M. Elsabrouty, "A highly parallel sad architecture for motion estimation in hevc encoder," in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS'14)*, Ishigaki, Nov. 2014, pp. 280–283.

[6] J. Byun, Y. Jung, and J. Kim, "Design of integer motion estimator of hevc for asymmetric motion-partitioning mode and 4k-uhd," *Electronics Letters*, vol. 49, no. 18, pp. 1142–1143, 2013.

[7] X. Yuan, L. Jinsong, G. Liwei, Z. Zhi, and R. K. Teng, "A high performance vlsi architecture for integer motion estimation in hevc," in *IEEE 10th International Conference on ASIC (ASICON'13)*, Shenzhen, Oct. 2013, pp. 1–4.

[8] T. D'huys, "Reconfigurable data flow engine for hevc motion estimation," in *IEEE International Conference on Image Processing (ICIP'14)*, Paris, Oct. 2014, pp. 1223–1227.

[9] A. N. Purnachand Nalluri, Luis Nero Alves, "High speed sad architectures for variable block size motion estimation in hevc video coding," in *IEEE International Conference on Image Processing (ICIP'14)*, Paris, Oct. 2014, pp. 1233–1237.

[10] E. Alcocer, R. Gutierrez, O. Lopez-Granado, and M. P. Malumbres, "Design and implementation of an efficient hardware integer motion estimator for an hevc video encoder," *Journal of Real-Time Image Processing*, pp. 1–11, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11554-016-0572-4

[11] HEVC software repository HM–14.0 reference model, https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags/HM-14.0.

[12] I. Kim, J. Min, T. Lee, W. Han, and J. Park, “Block partitioning structure in the hevc standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1697–1706, Dec 2012.

[13] G. Bjøntegaard, “Document vceg-m33: Calculation of average psnr differences between rd-curves,” ITU-T VCEG Meeting, Austin, Texas, USA, Tech. Rep, Tech. Rep., 2001.