An FPGA-based Integer Motion Estimator for real time HEVC UHD video encoding

E. Alcocer, O. López-Granado, R. Gutiérrez, and M.P. Malumbres

Abstract—**This paper presents the architecture of a high parallel integer motion estimation of the H.265/HEVC encoder. The motion estimation, and in particular the search algorithm, is one of the encoder critical blocks due to the overwhelming complexity required to accurately remove video temporal redundancy. The proposed architecture is based on both a novel memory controller and an innovative structure of an adder tree. As a result, our proposal reduces the overall video HEVC encoding time significantly. The proposed system has been designed in VHDL, synthesized and implemented by means of the Xilinx FPGA, Virtex-7 XC7VX550T-3FFG1158. Our design achieves encoding frame rates up to 30 fps at UHD-4K video formats.**

Index Terms—**H.265/HEVC, FPGA, Motion estimation, SAD architecture, Video coding.**

I. Introducción

THE new High Efficiency Video Coding (HEVC) standard was introduced aiming to improve the compression efficiency, achieving the same video quality than the H.264/AVC (high profile) at approximately half the bit-rate. However, the Motion Estimation (ME) module is by far the most complex task of encoder, requiring more than 90% of the encoding time. This is due to (a) a large set of Coding Tree Unit (CTU) partitioning modes, and also (b) to the greater CTU sizes respect to its predecessor, the H.264/AVC [1].

For these reasons, several hardware architectures have been proposed to speed up the HEVC ME module, reducing the overall encoder complexity as much as possible. The Integer-pel Motion Estimation (IME) block is in charge of motion estimation and it is composed of (a) an integer motion search algorithm, and (b) a Rate/Distortion (R/D) optimization procedure that optimally reduces the temporal redundancy found at the video sequence. In most of the works found in the literature, the proposed IME hardware architectures are only focused on the motion search algorithm since it takes most of the computational complexity of the IME block.

There are a lot of motion search algorithms that can be used to find the motion in video sequences. The most popular in hardware implementations is the Full Search (FS) algorithm. It follows greedy behavior by searching for motion at all points of the established search area of a reference frame, and, as a consequence, it is able to provide an optimal result (i.e., a motion vector that minimizes the residual error of the actual CTU).

The architecture proposals in [1, 2, 3, 4, 5] present an IME hardware block using FS strategy. In [1], a Sum of Absolute Diferences (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 standard, being able to run as fast as 30 fps at 2k video resolutions. The work presented in [2] proposes a SAD unit that computes all CTU partitions, achieving the same frame rates as previous work at 4k video formats. In their proposal, the search area has the same size as the maximum CTU, being implemented on an Application Specific Integrated Circuit (ASIC). In [3], 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. Different configurable search areas are studied in [4], achieving a maximum frame rate of 57 fps for a 720p video resolution. Several SAD units implemented on FPGA are described in [5], with different levels of parallelization, but no data about search area size, memory management, or how they obtain the minimum SAD are included.

In addition, similar hardware ME architectures have also been studied for the previous H264/AVC standard in [6, 7, 8, 9], which are of interest for our work due to the high similarity of the IME block architecture in both standards.

Therefore, our purpose is to design a new hardware architecture that may perform IME computation in a fast and accurate way in order to significantly reduce the computation cost of the overall encoder.

Regarding the ME process, each video frame is subdivided and partitioned into basic coding units called CUs. The coding structure in HEVC consists of CUs with a maximum size of 64x64 pixels, as large as that of CTUs, which can be recursively divided in picture squares until achieving a block size of 8x8 pixels. Each coding unit (CU) consists of prediction units (Intra or Inter) and its size can vary from the maximum size of the CU to 4x4 pixels for Intra prediction, or to 4x8 or 8x4 for Inter prediction, supporting 8 partitioning modes. Prediction units of sizes 2Nx2N and NxN are called square motion partitions (Square); 2NxN and Nx2N as Symmetric Motion Partitions (SMP); and 2NxnU, 2NxnD, nLx2N, and nRx2N as Asymmetric Motion Partitions (AMP). The total number of different partitions for a 64x64 CTU is more than 600, and for each of these partitions, the HEVC encoder performs one ME process in order to determine the best CTU partitioning in terms of bit rate and video quality.

There are many kinds of algorithms for block-based IME. The most accurate strategy is the FS algorithm which exhaustively searches motion for all prediction unit blocks at every single point of the established search area. Due to computational regularity and excellent video quality, FS motion estimation is commonly employed in hardware implementations [10]. Therefore, we will focus our work towards the design and hardware implementation of an FS algorithm that is able to significantly speed up the motion estimation process of the HEVC encoder without losing R/D performance.

Our proposed architecture is presented for several FS algorithms with different configuration sizes of both CTU and search area, in order to study the impact, in terms of speedup, of using our IME FPGA-based accelerator.

The rest of the paper is organized as follows. Section 2 presents the HEVC time profiling. Section 3 describes the architecture design of the proposed ME system while in Section 4, implementation results are provided. Finally, in Section 5 some conclusions are drawn.

II. HEVC Time PROFILING

ME is a task integrated in the HEVC Inter Prediction. For this reason, we have considered the Low Delay (LD) coding structure as configuration mode. In LD, the first picture is encoded as an intra-picture. Other pictures are encoded as generalized pictures (inter). This coding structure is the most popular for video conferencing and streaming, designed for interactive real-time communication.

Given the previous configuration, we have performed several time profiling experiments of the HEVC IME in order to observe how both parameters, the CTU size and search area size impact on the R/D performance of the HEVC encoder. We have choosen CTU sizes of 64x64 and 32x32, and their corresponding search area sizes 128x128 and 64x64, and 64x64 and 32x32, respectively. In addition, two video sequences from the HEVC common conditions video set were selected: RaceHorses at 832x480 resolution (30 fps) and BasketballDrive at 1920x1080 (50 fps).

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 i7-3770 CPU 3.40GHz with 16GB RAM.

Firstly, we have measured the time of the IME software module using a full search algorithm when a video sequence is encoded. The percentage of IME time spent by encoder, without the R/D optimization procedure, vary depending on the video sequence, the search area size, and the CTU size, as shown in Fig. 1 and Fig. 2.

In both Fig. 1 and Fig. 2 the percentage of IME time in the total video encoding with CTU sizes of 32x32 and 64x64 respectively, is shown. As can be seen, the encoder generally spends more time in the IME module when both the CTU size and Search Range (SR) are higher. In addition, in the case of the BasketballDrive video sequence with a greater resolution, this percentage of IME time is increased for every configuration size, as expected.

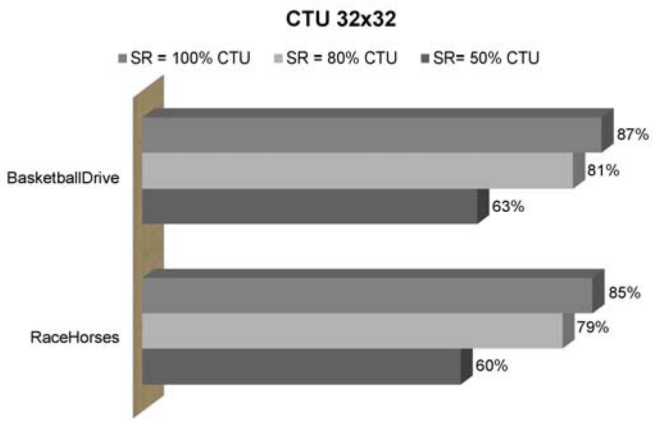

Fig 1. Percentage of IME time of the total video sequences encoding with a CTU size of 32x32

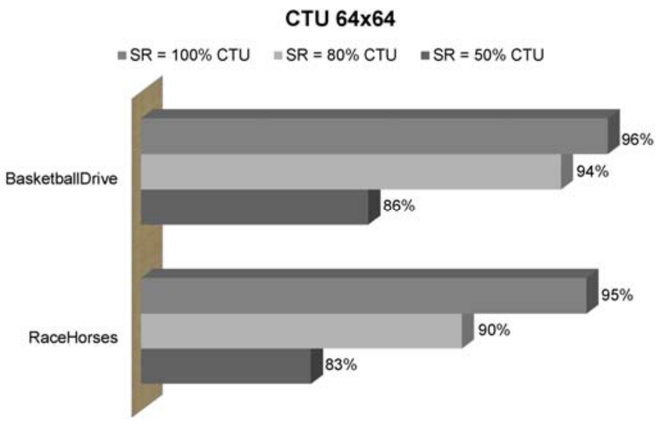

Fig 2. Percentage of IME time of the total video sequences encoding with a CTU size of 64x64

Therefore, a hardware design performing the IME computation in a fast and accurate way makes sense in order to reduce the IME complexity and as a consequence the overall encoder time as much as possible. For instance, for a video sequence like BasketballDrive and setting the CTU size to 64x64, the search range to 64 (default values in the HEVC reference software), and a full search algorithm, the total encoding time of 10 frames is 17 hours of which 16,32 are due to the IME process.

On the other hand, in order to reduce the complexity, allowing faster versions with reduced consumption, the CTU size and the search area should be reduced as much as possible. To evaluate this aspect, we have analyzed the impact of these parameters on the R/D (rate/distortion) performance by the Bjontegaard metric calculation (BD-rate). Bjontegaard's metric allows computing the average per cent saving in bitrate between two rate-distortion curves. The rate-distortion curves have been obtained for the compression levels (QP values): 22, 27, 32, and 37, taking into account a reference curve with typical configuration given by the HEVC software model, HM-14, with a CTU size of 64x64 and a search range of 64 (128x128 search area size).

In Fig. 3 and Fig. 4 it can be observed the percentage of BD-rate obtained with different CTU and SR sizes for the video sequences RaceHorses and BasketballDrive, respectively.

As can be seen, there are slight differences between the SR size for a given CTU size, being greater the difference as both the CTU size decreases and the video sequence resolution increases. Anyway, differences between parameters are negligible, being the maximum around 10% of the bitrate increased for a given PSNR. Although BD-rate differences may depend on the video content, similar results were obtained with other video sequences.

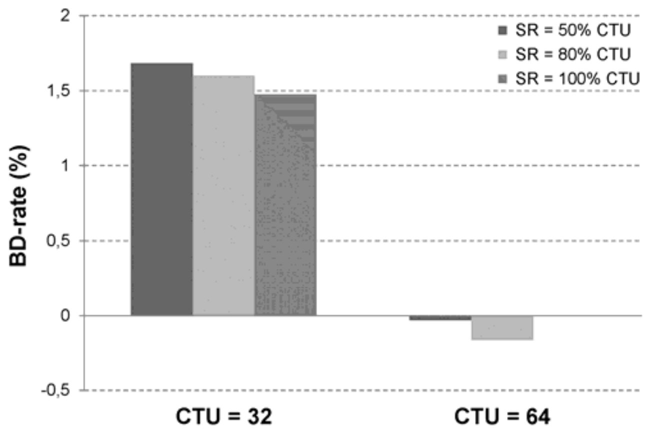

Fig 3. Bjontegaard metric calculation of RaceHorses video sequence with different CTU and search range sizes for the reference rate-distortion curve with a CTU size and a search range of 64

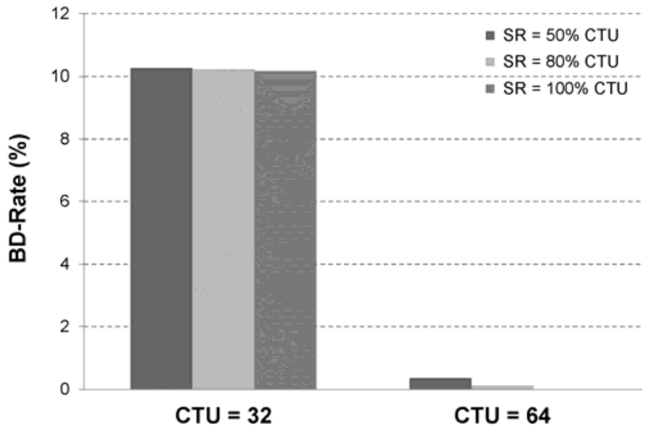

Fig 4. Bjontegaard metric calculation of BasketballDrive video sequence with different CTU and search range sizes for the reference rate-distortion curve with a CTU size and a search range of 64

III. HARDWARE ARCHITECTURE

Our system consists of (a) memory areas to allocate pixels of the current coding unit and the pixels belonging to the search area in the reference frame, (b) 64 processing units (PU), (c) 4096 processing elements (PE), (d) a Sum of Absolute Difference (SAD) adder tree block, and (e) a comparison block that saves the minimum SAD values and its corresponding motion vectors (MVs) for all CTU partitions. Each PE computes the distortion of both current and reference pixel. One PU consists of 64 PEs, which calculates the distortion values of a column of 64 pixels. In each cycle, current and reference pixels columns are delivered to 64 PUs, being able to compute the pixel distortion values of a 64x64 block (maximum CTU size) just in one clock cycle. The SAD Adder Tree Block (SATB) calculates the SADs for all the CU partitions using the results obtained from a block of 64x64 pixel distortions. Fig. 5 shows the structure of the proposed architecture.

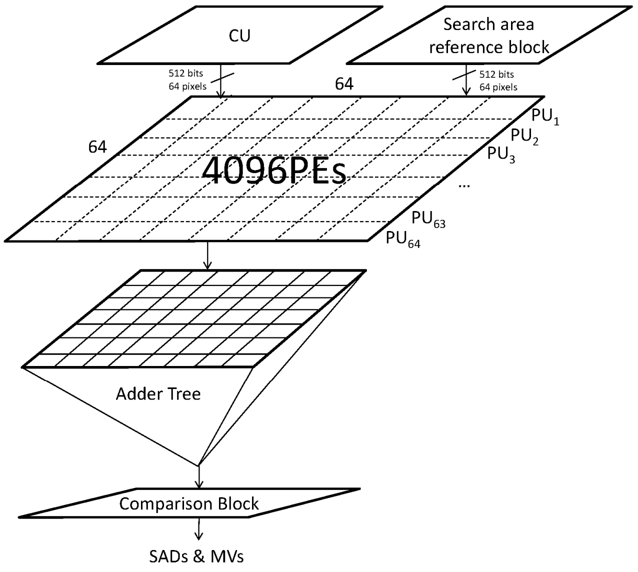

Fig 5. Figure 2. General structure of architecture proposed

The system is designed as a pipeline process. The memory reading process and shift registers propagation require only one clock cycle. The PUs use one cycle, the SATB requires eleven additional clock cycles and the comparison block needs one additional clock cycle. Finally, the proposed architecture uses the 14 clock cycles above mentioned plus 64 clock cycles corresponding to the initial preload of the shift registers and as many clock cycles as pixels the search area have, in order to process the integer motion estimation (IME) of all CUs from 64x64 to 8x4 and/or 4x8 and all its partitions.

A. Memory read controller

The search area is the region of the reference frame where full search motion estimation will run, evaluating the distortion found at every single point of this area. The search area is just centered on the location of current CU (see Fig. 6). For instance, a ±64 pixels search range represents a 128x128 pixels search window or search area.

In order to provide a high data reuse, a snake scan order and a reconfigurable data path with 64 propagation registers are adopted. At first step, ‘D’, a shift register fetches a row of

64 pixels from the BRAM at each clock cycle, setting the propagation of shift registers as downward. After loading 64 rows in the registers (a 64x64 CU), a 64-pixel column is delivered to each PU from both shift registers and current CTU memory bank, to compute the corresponding first SAD. After that, a new 64-pixel row is loaded in the shift registers discarding the first row, so we can proceed to compute the new CU just displaced one pixel in downward direction. From this moment, at each clock cycle we obtain the SAD corresponding to each positioned CU at each search area position. When computing the last CU in the downward scanning direction which corresponds to the last 64 rows of the search area, the next CU will be the one resulting from shifting one pixel/column to the right, step ’R’. Just after computing this CU, the memory controller will proceed to compute CUs in the upward direction, step ’U’, following the same criteria as in step ‘D’. This procedure will iterate until all searching CU positions in the search area have been processed, getting all the SADs for each CU at every clock cycle.

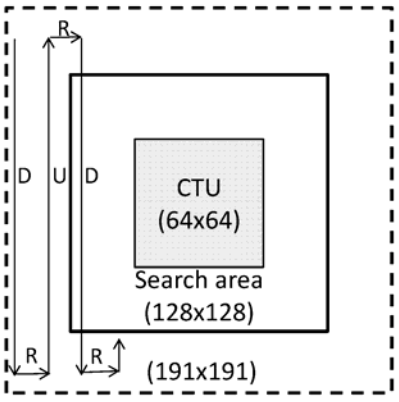

Fig 6. Memory scan order

B. *SAD adder tree block (SATB)*

The SATB block obtains the CTU SAD values for each CTU partition from 64x64 (maximum CTU size) to 4x8/8x4 as determined by HEVC standard, including both the Asymmetric Mode Partitions (AMP) and Symmetric Mode Partitions (SMP). After receiving from PUs the 64x64 SADs associated to the current search area position, a succession of aggregation stages are performed to compute the corresponding SAD values for all the CTU partitions (a total number of 677), as shown in Fig. 7. At each stage, all pairs of consecutive columns/rows are added, reducing to half the width/height of the resulting partition. This SAD aggregation process is followed until the last partition size is reached (1x1), ie. the SAD corresponding to the 64x64 partition. At intermediate stages, the SADs of the rest of partitions are stored, as shown at Fig. 7. The SATB is also designed as a pipeline, so with an eleven clock cycles delay, 677 SADs corresponding to each CU block are delivered to the next comparison block every single clock cycle.

C. *Comparison Block*

Finally, the comparison block should keep the minimum SAD values for each CTU partition with their corresponding motion vectors (search area locations). So, it will compare all incoming SADs from the adder tree block with the minimum SADs previously found. After comparing the SADs from the last search area location, the minimum SADs for each partition and the associated motion vectors are obtained.

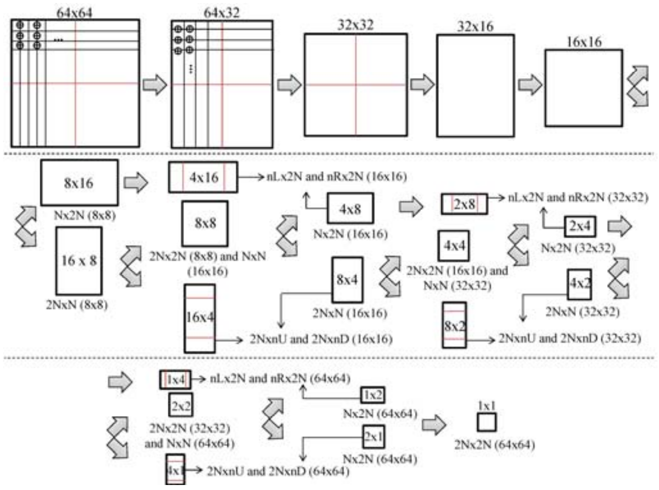

Fig 7. Structure of SAD Adder Tree Block

IV. IMPLEMENTATION RESULTS

Our architecture has been modeled in VHDL, and it has been synthesized, simulated, and implemented on the Xilinx FPGA, Virtex-7 XC7VX550T-3FFG1158. The correctness of our design was tested and verified with the HEVC HM 14 reference model.

To evaluate the performance and efficiency of our design, we have parametrized our IME architecture to allow different configurations, such as (a) the maximum CTU size with values of 64x64 and 32x32, and (b) the size of the search range of the reference frame with values defined as the 50%, 80%, and 100% of the CTU size.

TABLE I
FRAME RATE FOR DIFFERENT CONFIGURATIONS

CTU size	32			64		
SR size	16	26	32	32	52	64
Clock (MHz)		318			247	
Fps at 2K	141	55	37	116	45	30
Fps at 4K	37	15	10	30	12	8

In Table 1, we show the resulting operating frequency, and the system throughput in terms of the maximum frame rate under different video formats (2K, and 4K), for different configurations of CTU and SR sizes. Our design can operate at the frequency of 247 MHz and 318 MHz for a 64x64 CTU and a 32x32 CTU, respectively. It enables the encoder to carry out the IME process with a 64x64 CTU size and a search area of 128x128 pixels (SR of size 64), as the HM14 reference model establishes, obtaining a throughput of 30 fps at 2K video formats (2K@30fps).

Our proposal is able to process video sequences in real time for 2K resolutions in all tested configurations, and also with 4K video formats if the search area size is the same as the CTU size (SR is 50% of CTU), as can be seen in Table 1.

In Table 2, the resources used to implement our proposal for CTU sizes of 64x64 and 32x32, respectively, are shown. As can be seen, the slice area required by slice registers, LUTs, and BRAMs increases four times linearly with the increase of the maximum CTU size, as expected.

TABLE II
IMPLEMENTATION RESULTS OF THE PROPOSED ARCHITECTURE

CTU size	32	64
No. of slice registers	36864 (5.32%)	144302 (20.83%)
No. of slice LUTs	48531 (14.01%)	188664 (54.46%)
BRAMs	17 (1.35%)	33 (2.71%)
Available SAD partitions	4x8 up to 32x32	4x8 up to 64x64
CTUs/s when SR=100%CTU	77k CTUs	15k CTUs
CTUs/s when SR=80%CTU	116k CTUs	23k CTUs
CTUs/s when SR=50%CTU	298k CTUs	59k CTUs

In addition, we have compared our architecture with previous state-of-the art architectures.

Regarding a CTU size of 64, Medhat et al. [1] present a parallel SAD block for the HEVC integer-pel full search architecture without supporting AMP modes with a search area of 104x104 pixels. Their design can operate at the frequency of 458.7 MHz. The operating frequency of our proposal with the same technology and configurations is almost two times lower. However, our architecture including AMP modes is capable of processing 45 fps at 2K video formats instead of 30 fps as obtained by the proposed design in [1]. On the other hand, D'huys [4] proposes a reconfigurable design for HEVC motion estimation which can operate at the frequency of 150 MHz, whereas the operation frequency of our proposal is 159 MHz. His architecture is compared with our proposal, setting a common search area size to 64x64 pixels and the same technology. The architecture presented in [4] is able to process a video resolution of 720p at 57 fps, whereas our architecture is capable of processing 173 fps at the same video resolution.

Regarding a CTU size of 32, Yuan et al. [3] present a IME full search architecture with a search area size of 48x48 pixels. With the same configuration and technology, our proposal can provide a higher operation frequency, achieving throughput of 43 fps at 1080p video resolution, whereas the architecture presented in [3] is able to achieve 30 fps at 1080p video formats.

Considering the presented results, our architecture overcomes the performance of previous state-of-the-art architectures.

V. CONCLUSION

In this work we present a configurable hardware fast integer motion estimation block for the HEVC which supports AMP mode, implemented over a Virtex-7 FPGA. Our system is able to encode 2K video formats at 116 fps and 4K at 30 fps for a 64x64 CTU, which represents a huge complexity reduction of the HEVC video encoding process, achieving real-time encoding for high definition video contents and beyond.

ACKNOWLEDGMENT

This research was supported by the Spanish Ministry of Economy and Competitiveness under grant TIN2015-66972-C5-4-R.

REFERENCES

[1] Medhat A., Shalaby A., Sayed M.S., Elsabrouty M. (2014) A Highly Parallel SAD Architecture for Motion Estimation in HEVC Encoder. IEEE Asia Pacic Conf. Circuits Syst. (APCCAS). Ishigaki, nov, pp. 280-283

[2] Byun J., Jung Y., Kim J. (2013) Design of integer motion estimator of HEVC for asymmetric motion-partitioning mode and 4K-UHD. Electronics Letters. Vol:49, No:18, pp. 1142-1143

[3] Yuan X., Jinsong L., Liwei G., Zhi Z., Teng R.K.F. (2013) A High Performance VLSI Architecture for Integer Motion Estimation in HEVC. IEEE 10th Int. Conf. ASIC (ASICON). Shenzhen, oct, pp. 1-4

[4] D'huys T. (2014) Reconfigurable data flow engine for HEVC motion estimation. IEEE Int. Conf. Image Processing (ICIP). Paris, oct, pp. 1223-1227

[5] Nalluri P., Alves L.N., Navarro A. (2014) High Speed SAD Architectures for Variable Block Size Motion Estimation in HEVC Video Coding. IEEE Int. Conf. Image Processing (ICIP). Paris, oct, pp. 1233-1237

[6] Elhamzi W., Dubois J., Miteran J. (2014) An eficient low-cost FPGA implementation of a configurable motion estimation for H.264 video coding. Springer Journal of Real-Time Processing. Vol:9, No:1, pp. 19-30

[7] Moorthy T., Ye A. (2008) A scalable architecture for variable block size motion estimation on field-programmable gate arrays. IEEE Canadian Conf. Electrical and Computer Engineering (CCECE). Niagara Falls, may, pp. 1303-1308

[8] Kthiri M., Kadionik P., Levi H., Loukil H., Atitallah B., Masmoudi N. (2010) An FPGA Implementation of Motion Estimation Algorithm for H.264/AVC. IEEE 5th Int. Symp. I/V Communications and Mobile Network (ISVC). Rabat, sep, pp. 1-4

[9] Pastuszak G., Jakubowski M. (2013) Adaptive Computationally Scalable Motion Estimation for the Hardware H.264/AVC Encoder. IEEE Trans. Circuits Syst. Video Technol. Vol:23, No:5, pp. 802-812

[10] Lin Y.L.S., Kao C.Y., Kuo H.C., Hen J.W. (2010) VLSI Design for Video Coding - H.264/AVC Encoding from Standard Specification to Chip. Springer, New York

[11] 17. HEVC software repository HM-14.0 reference model. Available: https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags//HM-14.0 (2014). Accessed 2 May 2014