International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 4, August 2019, pp. 2469~2480
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i4.pp2469-2480

Area and power efficient VLSI architecture of mode decision in integer motion estimation for HEVC video coding standard

El Ansari Abdessamad[1], Nejmeddine Bahri[2], Anass Mansouri[3], Nouri Masmoud[4], Ahaitouf Ali[5]
[1,5]Faculty of Sciences and Technology, LERSI, Laboratory, University of Sidi Mohammed Ben Abdellah Fez, Morocco
[2,4] National School of Engineers/LETI Laboratory, University of Sfax, Tunisia
[3]National School of Applied Sciences, LERSI, Laboratory, University of Sidi Mohammed Ben Abdellah Fez, Morocco

Article Info

Article history:

Received Apr 22, 2018
Revised Mar 8, 2019
Accepted Mar 12, 2019

Keywords:

HEVC
Integer Motion Estimation (IME)
Real time processing
Sum Absolute of the Difference (SAD)
Very-Large-Scale Integration (VLSI) architecture

ABSTRACT

In this paper, we propose a new parallel hardware architecture for the mode decision algorithm, that it is based on the Sum Absolute of the Difference (SAD) for compute the motion estimation, which is the most critical algorithm in the recent video encoding standard HEVC. In fact, this standard introduced new large variable block size of the motion estimation algorithm and therefore the SAD require a more reduced execution time in order to achieve the real time processing even for the ultra-high resolution sequences. The proposed accelerator executes the SAD algorithm in a parallel way for all sub-block prediction units (PUs) and coding unit (CU) whatever their sizes, which turns in a huge improvements in the performances, given that all the block sizes, PUs in each CU, are supported and processed in the same time. The Xilinx Artix-7 (Zynq-7000) FPGA is used for the prototyping and the synthesis of the proposed accelerator. The mode decision for the motion estimation scheme is implemented with 32K LUTs, 50K registers and 108Kb BRAMs. The implementation results show that our hardware architecture can achieve 30 frames per second of the 4K (3840×2160) resolutions in real time processing at 115.15MHz.

Corresponding Author:

El Ansari Abdessamad,
Faculty of Sciences and Technology of Fez, Laboratory of Renewable Energy & Smart Systems,
Sidi Mohammed Ben Abdellah University,
P.O. Box 2202 Fez, 00212626243355, Morocco.
Email: abdessamad.elansari@usmba.ac.ma

1. INTRODUCTION

High Efficiency Video Coding (HEVC) is the new standard video coding proposed in January 2013 by the ISO/IEC and ITU-T. It produces efficiency coding up to 50% compared to its predecessor H.264/AVC [1, 2]. This new standard introduces improvements in the encoder to work with ultra-high resolution like 4k (3840×2160 pixels) and 8k (7680×4320 pixels). The marked improvements are: (i) the intra prediction that works with 35 modes instead of 9 modes in H.264; (ii) the seven-tap and eight-tap filters used for the motion precision, whereas only a six-tap is used in H.264/AVC; and (iii) the block partition structure based on coding tree units (CTUs), allowing prediction refinement with variable block size from 16×16 to 64×64 against the fixed Macroblock size 16 × 16 in the old standard H.264/AVC.

Many works have been devoted to implementing the HEVC standard by using both software and hardware facilities to reach real time processing for the ultra-HD (high definition) applications. The video coding algorithm is known by high computing complexity that forces designers to choose between the architectures of the more popular Digital Signal Processing (DSP) and Field Programmable Gate Array (FPGA), which are the more used embedded platforms. Currently, the recent embedded systems offer, in most cases, both homogenous and heterogeneous architectures leading to many implementation possibilities, including both simple and mixed software and/or hardware implementations. However, even if DSPs have an

advantage in the speedup of high processing algorithms, they have limitations, for instance, the needed time for some sophisticated programs loading, while FPGAs can easily surround such problems, especially for intensive tasks, having many loading steps.

Many efforts are necessary to identify which of the hardware or software implementation can be more advantageous for each task. For instance, it is preferable to convert the most time consuming part in a hardware part, as the case of tasks using communication exchanges between blocks. Sometimes, mixing the advantages of software and hardware can be more beneficial in the whole video coding process. The way is first to profile the software project to pinpoint the more cumbersome parts and to decide which kind of design is more adapted. Fortunately, many embedded development boards offer the mixing design facility among which is the Chip Zynq 7000 from the Xilinx family, based on 28 nm CMOS technology. The inter prediction is the more loading time block of the encoder, because it contains the algorithms of motion estimation and motion compensation [3]. The motion estimation algorithm still the same as for the H.264/AVC, based on the cost SAD with the ability to process the biggest PUs sizes announced in the HEVC.

In a previous work [4], we have compared an implementation of a whole HEVC encoder in both processors, ARM and Intel in order to study the software complexity. So, in Intel processor is faster about 10 times than the embedded processor ARM, however, the execution steel too far from real time processing conditions, especially for high resolution. To overcome this problem, we focus on the hardware implementation in this work. The reference software of the standard HEVC test Model (HM) is available on-line [5]. The TComRDCost class, which it is used for RD cost computation and it includes the all functions for SAD computation. The TComRDCost takes the longest execution time of HM encoder, when compared to the other classes. That takes about 40% of the completely encoding time [3].

In the inter-prediction blocks, the SAD for the motion estimation algorithm is the most important and time-consuming step. Its hardware implementation can be helpful for the video encoder. In the literature, many works are proposed to support the SAD architecture for several application domains, such as computer vision, like motion detection for image processing [6], on the system video surveillance based motion detection and recognition in [7], that it implements on an embedded board based on XC2V1000 FPGA and motion estimation for video compression standards [8, 9, 10, 11, 12, 13, 14]. All the proposed architectures aim to reach real-time processing for higher resolutions sequences with the highest possible operating frequency and to compute the maximum inter-prediction blocs whatever their sizes. Walter et al. [8] reported on a hardware architecture with a high-throughput for low power SAD calculation functioning up to real-time encoding process for the resolution of 720×480 pixels. Their design was developed and synthesized using two CMOS technologies, 180nm and 65nm. They used two metrics to compare their obtained results, the maximum throughput and the minimum consumed energy per operation. They reported the average of the consumed power using 65 nm technology decreases by 53% compared to 180 nm technology, considering the needed lower supply voltage. In addition, the new technology 65 nm achieved the high frequency because of the higher throughput. In [9], Rehman et al. proposed efficient hardware architecture for the SAD algorithm, designed for image processing. The advantage of this architecture is the decreasing absolute difference additions for the 4×4 pixels block sizes. Using an FPGA with Xilinx XC2V1000, they reached a frequency of 133 MHz and used 657 LUTs with single 4×4 block, and thereafter, they use the developed 4 × 4 engine for processing the biggest blocks. Another SAD dedicated hardware architecture has been reported by Kalamiros and Lygouras [10] for the stereo vision acceleration, especially for computing the method of local correlation. The proposed architecture was implemented on FPGA Cyclone II EP2C35 device, working at 100 MHz, and their proposed SAD accelerator takes 23900 logic elements to reach 162 frames per second in VGA resolution (640×480) coded on 8-bit depth.

Zhenyu et al. [11] designed two hardware architectures for the SAD algorithm, with the H.264/AVC standard supporting the variable block size technique. The first architecture, based on the propagated partial SAD, reaches 231.6 MHz-operating frequency at a cost of 84.1K gates, and the second is a tree based SAD architecture achieving 204.8 MHz with 88.5k gates. The two designs were synthesized on a 0.18µm 1P6M CMOS technology, and the architecture level approaches were used to improve performance. In addition, Purnachand et al. [12] developed a hardware architecture for the HEVC video coding standard. The synthesis was performed on a Xilinx Virtex-5 FPGA, takes 291.27K of gate counts, and reaches 171.9 MHz as operating frequency. Their accelerator is based on parallel processing and designed to support all PUs block sizes, including blocks for Asymmetric Motion Partitioning (AMP). Xu et al. [13] and synthetized in Xilinx Virtex-6 XC6VLX-550T FPGA proposed another hardware SAD accelerator. Their architecture contains two parallel sub-architectures to process 1080p with 30 frames per second (fps) reaching real-time video coding. However, the accelerator can only compute 16×16 PU and the corresponding sub-blocks. The synthesis shows an operating frequency of 110 MHz with 55346 LUTs, 19744 registers, and 148kB of BRAM. More recently, Medhat et al. [14] proposed a parallel hardware SAD accelerator for the motion estimation, synthetized on a Xilinx Virtix-7 XC7VX550T FPGA. An operating frequency of 458 MHz has been reported with 39901 LUTs and 24957 registers. With this architecture, a 2K resolution with 30 fps can be processed, which is the success

key of this architecture. Other customized solution have been also published such the VLSI pipelined multiplierless fixed-point 8x8 DCTDesign of a Fast Pipelined 8x8 Discrete Cosine Transform other blocks [15]. Authors have implemented and compared the DCT using two architectures, non-pipeline and a 2-stage pipeline and their simulation results show that throughput can be improved by almost a factor of two, from around 1 Giga pixels/s to 1.8 Giga pixels/s, at a small cost of roughly 14% more resources (i.e. pipeline registers).

In this work, we propose a new, highly efficient VLSI architecture for the SAD decision mode, implemented in the HM of HEVC. This architecture can process all sizes of PUs, including AMP, which is used to increase the coding efficiency of video sequences having areas that are more irregular. The new architecture is designed to load all the input pixels from the reference and current blocks in a first step, making them available during the processing process. Then the proposed design highly reduces exchanges with the memory and the memory accesses and finally leads to a parallel computing of all PUs simultaneously, reducing the latency time.

2. HEVC MODE DECISION OF MOTION ESTIMATION ALGORITHM

2.1. Coding structure

The HEVC introduces more improvement than H.264/AVC [1, 2]. The most importance is the highly flexible and efficient block partitioning structure by introducing four block concepts: Coding Tree Unit (CTU), coding unit (CU), prediction unit (PU), and transform unit (TU). The first step is the partitioning structure of the pictures, which can be partitioned in slices and tiles; the slice is partitioned into CTUs. In H.264, the CTUs are analogue to a fixed macroblock size of 16×16, whereas in HEVC, they can be sized to 16×16, 32×32 and 64×64 to accept several video sequences. The size of a CTU is fixed in the main encoder configuration file and can be partitioned into multiple squared CUs regions as shown in Figure 1.

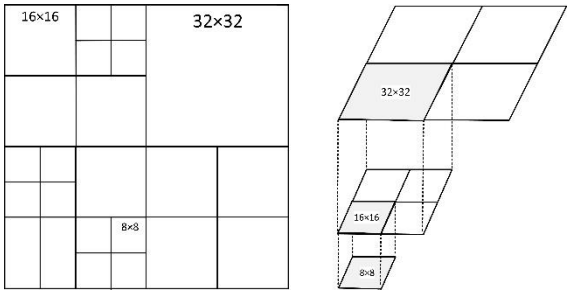

Figure 1. Coding structure of the CU in the HEVC encoder

The number of CUs in a CTU is mainly determined by the required resolution. The small CU is 8×8 and can be scaled up to 64×64. The CU can be divided into Prediction Units. This block can be used for the intra and inter block prediction, and its size varies from 4×4 to 64×64. After the PU, an appropriate size of the transform Unit, TU, is determined for the Integer Direct Cosine Transform (IDCT) and the quantification.

In the HEVC encoder, each CU can be divided into multiple PUs, and each PU is characterized by a motion estimation vector. In addition, when the CU size is equal to 2N×2N with N an integer value among 32, 16, 8 or 4 depending on the depth (0, 1, 2 and 3) in the corresponding coding tree structure, eight PU splitting types are defined as illustrated in Figure 2: two squared region (PART_2N×2N and PART_N×N) and six rectangular ones, two symmetric (PART_2N×N and PART_N×2N), and four asymmetric named AMP (PART_2N×nU, PART_2N×nD, PART_nL×2N, and PART_nR×2N) where U, D, L, and R mean Up, Down, Left, and Right, respectively and n represents the smaller size in AMP partition modes [16].

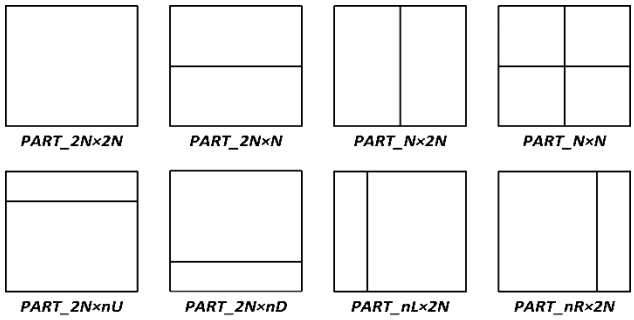

Figure 2. Symmetric and asymmetric block portioning of the PU

2.2. Mode decision SAD algorithm in HEVC

The critical part of the all-video encoder standards exploiting temporal redundancy is the Motion Estimation (ME). The recent HEVC has a more complex motion estimation algorithm than H.264/AVC, due to the pixel density in the high picture resolution processing (8K and 4K) and the new PU block partitioning. The SAD processing is performed for the integer ME (IME), and it is calculated for all sizes for the PUs candidates. It is computed based on the information coming from the current or original block (Bo) of the PU and the predicted or the reference block (Br) obtained by the motion vector predictor derivation process. The calculation of the SAD is performed using (1).

$$SAD = \sum_{i=1}^{K} \sum_{j=1}^{L} |B_r(i, j) - B_o(i, j)| \quad (1)$$

Where i and j are the corresponding pixel in the Bo and Br blocks. K and L are the width and height of each search window, respectively. indicates the pixel of the reference block and Bo(i,j) indicates the pixel of the original (current) block. The computation complexity is in the number of calls and the different size SAD for IME of the inter prediction. Hence, a hardware accelerator of SAD is required.

3. PROPOSED SAD HARDWARE ARCHITECTURE

The SAD is a cost function used to compute distortion between two blocks. Several architectures for the SAD calculation were proposed in the literature. They are mainly based on one of the following three kinds of designs, parallel, Pipeline, and sequential. The sequential design uses accumulator to store the results of the absolute value in each clock cycle. It uses a minimum hardware usage, but takes more than one clock cycle to give the output results; subsequently, it is impossible to reach up real-time processing due to the higher resolutions treatment in our cases. The pipeline architecture, as it uses a series of connected stages, takes in input a vector of pixels instead of one pixel input and generates several results on one line; for instance, in [9], the authors use an input vector of 128-bit presents 16 pixels 8-bit sample of 4×4 block sample. So the pipeline architecture is faster than the sequential, but it is not enough to reach the real time for high resolution. In [9], the proposed architecture is designed in pipelined stages; however, it reaches a rate of 30 fps only for small sequences with 720×480 pixels with the configuration level 3 of H.264/AVC encoder.

The parallel design is faster than both sequential and pipeline, because it is constructed on several slices, which are processed independently of each other; consequently the area consumption is increased. In addition, it's possible to mix these types; for instance, the total architecture works parallel, and the silices design work in sequential or pipelined mode like in ref. [11]. For the use of fully parallel architecture, a highly optimized design is needed to reduce the consumed area and to keep higher performances.

The parallelism in the hardware architecture is explored to increase the accelerator throughput to compute all SAD operations on four clock cycles, but the number of input samples pixels increases and the number of adders in the depth tree stage increases. An optimization is necessary to reduce the consumed area and the delay in reaching the real time encoding system, particularly for 4K applications.

The SAD algorithm implementation is achieved in three main steps; first the absolute difference between two pixels is calculated by the processing unit (PrU) designed in Figure 3. Next, the sum of these differences is calculated in one step for four input pixels using the deigned processing element (PE) presented in Figure 4. Finally, for the whole SAD algorithm, the PE is implemented 1024 times to allow processing of all block sizes from 4×8 & 8×4 to 64×64.

3.1. Architecture of absolute difference

The PrU, given the absolute difference (abs), operates by comparison between the two pixels coming from the reference and current images. If the difference is positive or equal to zero, the abs is directly obtained; if not, the two compliments of the negative term are used to make the result positive. This operation of pixel comparison can be described by the following (2):

$$abs = \begin{cases} B_r - B_o, & B_r - B_o \geq 0 \\ not(B_r - B_o) + 1, & B_r - B_o < 0 \end{cases} \quad (2)$$

The proposed hardware architecture of PrU is presented in Figure 3. It consists of an optimal circuit using one multiplexer (MUX) for which the selection input corresponds to one of the two states of the comparator (Comp) outputs (i.e. Positive or negative value). Depending on the sign of the comparison, the MUX is given in each case the absolute difference coming from the output of the subtractor (SUB). The equation not(Br-Bo)+1 is implemented using one inverter followed by one 1 bit adder. The results are always 8 bits-coded. The registers (R) are used to store the values of the processed pixels. To decrease the hardware cost, the design of PrU is optimized by the optimization the comparator.

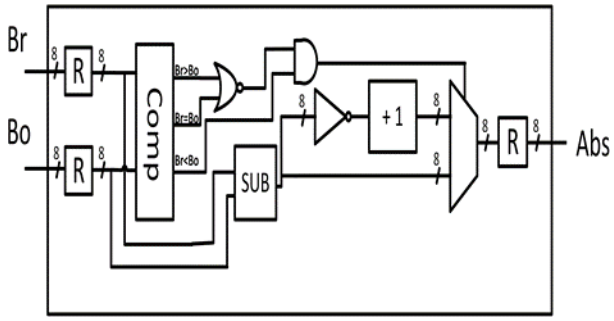

Figure 3. Processing unit (PrU) for the calculation of the absolute difference value

The main function of Comp is to compare two input signals and to return the results on three outputs, each coded in 1-bit. The Comp is a 8-bit comparator. To reduce the used area, it contains the logic gates in the 1-bit comparator as shown in Figure 4 and uses the principal 2:1 compressor described in ref [17]. In our architecture [18], the Com construction is based on a tree of 1-bit comparator using the method of 2:1 compressor to compare two A and B input pixels coded in 8-bit and delivering the results of the comparison of the output.

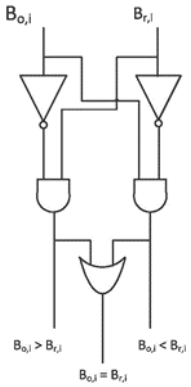

Figure 4. 1-bit comparator using in the Comp

The PE, as shown in Figure 5, is composed from four PrU, seven registers, and three adders compute the sum of absolute difference of eight pixels, four from the reference Block and four from the original. The result is given by the output signal S4x1, which is now coded in 10-bits and obtained in one clock cycle.

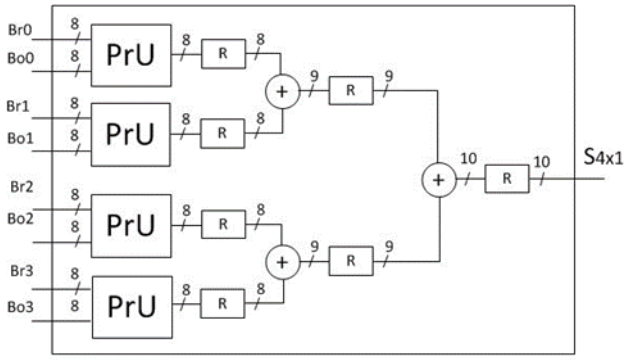

Figure 5. Processing element (PE) for eight pixel inputs and one output

The global SAD calculation architecture is presented in Figure 6. It contains:

1. Three 4K single port RAM blocks, the two first (2×1024×8 bits) are dedicated to the pixels of the current and original block storage, and the third is employed to store the results of the computed SADs.
2. 1024 PE arranged in a parallel way to start functioning at the same time, and calculating line by line the partial sum of the absolute differences (PSAD) for the eight selected pixels.
3. A three adder blocks to deliver the SAD by summing the PSAD two by two until the final result. This part is named combination addition in Figure 5.

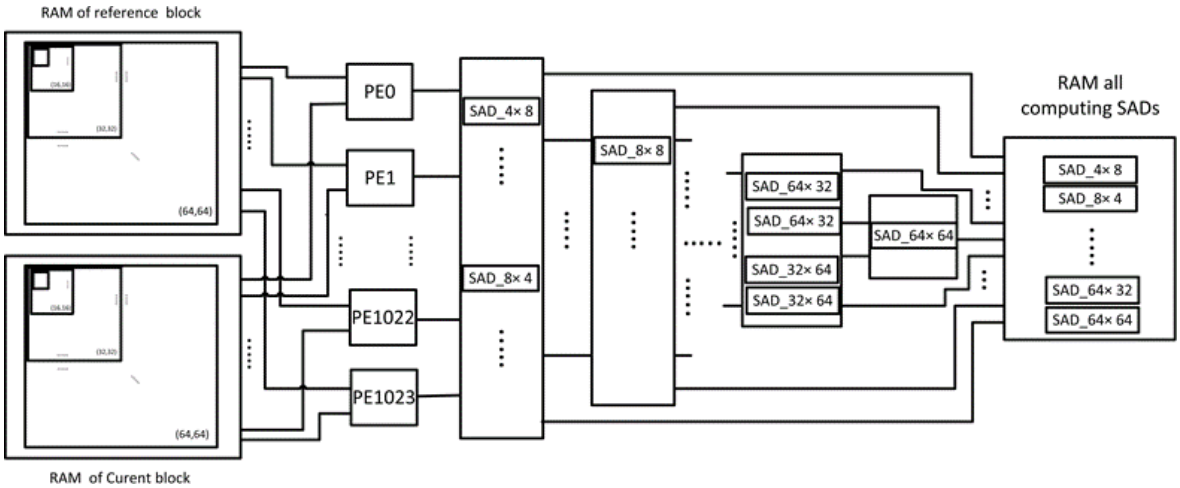

Figure 6. The global architecture of the SAD algorithm in IME for HEVC encoder

In the hardware architecture, the PrUs is organized line by line in such a way that, from eight PrUs, two small PUs 8×4 and 4×8 can be constructed. Moreover the proposed architecture is designed for the largest size block 64×64 and can easily be adapted for the other smallest size blocks with adequate memory storage of the starting pixels giving in Figure 7.

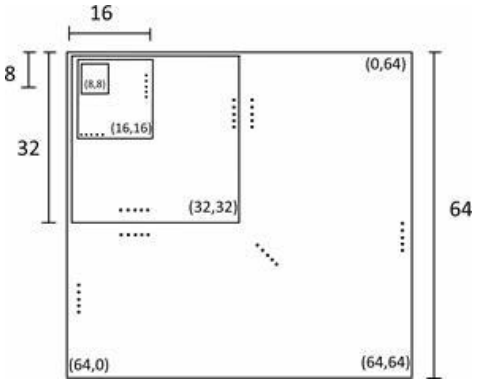

Figure 7. Location of data input pixels Sub-CU 64×64

3.2. High speed adder CLA

The SAD algorithm is based on a basic arithmetic operation, since it comprises double summations. To make it faster, several proposed architectures use the carry propagation for the binary addition known by its high performances. Therefore, its implementation is adequate for Very Large Scale Integration (VLSI).

In references [17] and [18], the authors used two faster parallel adders, Ripple Carry Adders (RCA) and Carry Lookahead Adders (CLA), and they make a comparison between them based on the delay and the area consumption. First, RCA takes more delay than the CLA because it has a lengthy propagation (through all adders); it starts from the first adder and it finishes at the last one. The CLA has no carry propagation because the carry output of current adder is connected to the carry input bit of the next adder. In the area, the RCA required less surface than the CLA, since the last has a scalar architecture and it needed more components. Giving that we are interested in a rapid system, the CLA is more adopted in our hardware accelerator for designing all adders, even though it increases the needed area, but without affecting the total surface; moreover, it is simple to design, given that the CLA architecture uses basic gate logic such as AND, OR and XOR.

Our CLA comprises two levels. The first level is used to compute the partial (intermediate) values, Gi and Pi, and generate the terms, SUM(i) and Ci+1 as output, defined by the equation (3), (4), (5) and (6) as presented in Figure 8, which proposed a section of the whole CLA design. The second level is the total design of CLA as shown in Figure 9, which computes each carry, the total sum, and indicates the overflow of addition as output. When each carry is available, the partial sum is calculated simultaneously and then the overflow of the addition is calculated at the end.

$$G_i = A(i).B(i) \tag{3}$$

$$P_i = A(i) \oplus B(i) \tag{4}$$

$$Cout_i = G_i + (P_i.Cin_i). \tag{5}$$

$$Sum(i) = P_i \oplus C_i \tag{6}$$

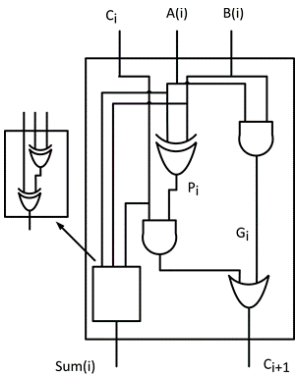

Figure 8. Proposed section of CLA adder

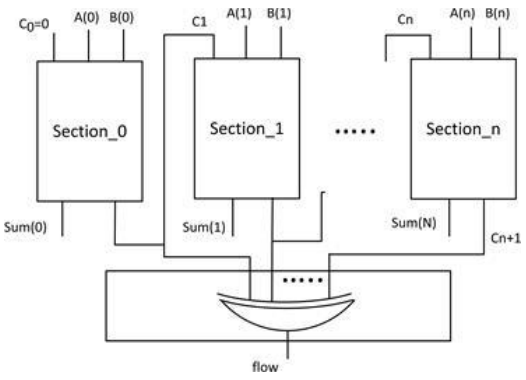

Figure 9. Proposed CLA n-bit adder based on the architecture of CLA adder

Cout_i (or Cn+1) and Cin_i (or Cn) correspond respectively to the carry output and the carry input of section N. As illustrated in Figure 9, each Cout of a given section is connected to Cin of the subsequent one. Gn and Pn are respectively the generated and the propagated carries.

In Figure 9, our high parallel architecture starts processing from 1024 PEs and arrives to the SAD_64×64, different to the case in [19], that can compute the size of windows differently. The SAD_64×64 is the value of SAD corresponds to the bigger bock 64×64, and generates all its corresponding small SAD simultaneously. To summarize our approach, one computes both smaller blocks 8×4 and 4×8, presented by values SAD_4×8 and SAD_8×4 respectively. They are computed by the addition of the eight PEs corresponding to each block. After computing all smaller blocks inside 64 × 64, the next step computes the SAD value (SAD_8×8) of 8×8 block by adding two smaller blocks already processed. Following the same method, the block is processed from the previous SAD values to arrive at the value SAD_64×64 of the bigger block. Finally, the design is arranged in the tree. An example is given in Figure 10, showing SAD_8×8 value is processed by adding two SAD_8×4.

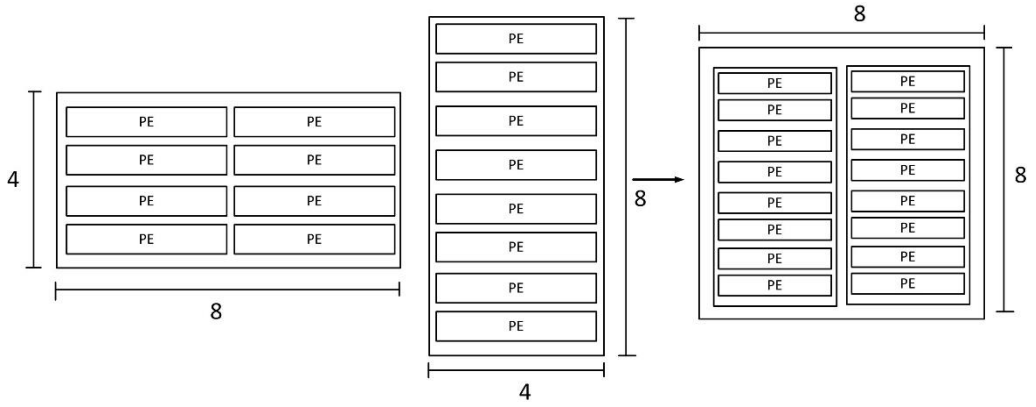

Figure 10. Construction of the SADs 8×4 and 4×8 from PEs and also the construction of 8×8 from two 8×4

4. EXPERIMENTAL RESULTS

In the first step, the proposed architecture for the SAD algorithm was implemented in C++ language, then it was compiled to be validated and to reduce its complexity. Afterwards, the accelerator hardware design is developed using the hardware description language VHDL, tested using Mentor Graphics ModelSim PE before being synthetized in the FPGA Artex-7.

The first test was performed on some test video sequences proposed by the standard. The SAD was validated using three technics, first with reference software of HEVC HM15.0 version [5], the second by the software C++ executed under Linux Ubuntu, and the last from the proposed hardware design. The three obtained results are in rather good agreement.

The frames of sequence Basketballdrive (1080p) and others are used in this test, and the value of QP is fixed at 32. For more configurations of the execution, we have been inspired by the work presented in Ref. [20], which provide a performance evaluation of distributed HEVC encoding, and was focused on the effects of the video partitioning method on overall distributed encoding performance using a partitioning scheme based on GOP in order to measure the encoder efficiency. Then the Random-access only configuration file is selected from three configurations (Random-access, low-delay and intra only), because it is most suitable for our case. All these characteristics and even more information are mentioned in the configuration file proposed by the standard in [21].

The proposed design block for architecture for the SAD algorithm in HEVC is first coded in a high-level language C++ and it is implemented on both processors, ARM cortex-a9 embedded in Zynq device and Intel core i3 CPU clocked 667MHz and 2.40 GHz, respectively, to be validated and to reduce the computational complexity. Second, this design is also coded in VHDL language and tested using Mentor Graphics ModelSim PE, then the hardware architecture prototyped on Artex-7 FPGA (xc7z020-1clg400) from Xilinx, and acceleration hardware was verified on Microzed board.

4.1 Synthesis results and analysis

The proposed architecture was synthesized using recent design environment (Xilinx Vivado 2016.3) like using in the previous work for ICT algorithm inside the HEVC encoder [22] and in other work [23]. The synthesis results for our architecture are presented in Table 1. The design works with a maximum frequency up to 115.15MHz, and it consumes 32752 LUTs, corresponding to 61.56% of device resources (53200 LUTs), and it uses 5077 registers (4.77%) from the 106400 available on the board. It takes 108Kb in the embedded memory BRAM, and in addition, our design exploits both embedded multiplexers on the FPGA, F7 MUX and F8 MUX by 1577 and 481, respectively. However, it needs no DSPs from the 220 existing DSPs in the FPGA.

The proposed design was simulated-using ModelSim. It was shown that the architecture could, in one clock cycle, compute the SAD for all block sizes up to 64×64, including AMP. Giving the reached frequency 115.15 MHz, our hardware design can process 30 frames per second (fps) for 4K resolution calculated from the number of search points estimated for 30fps@4k in section II divided by the operating frequency, and it can reach up to 120 fps for the full high resolution calculated in the same way. The design can be improved to reach more than this number of frames per second with a cost of additional resource consumption from the device.

Table 1. The synthesis results of the proposed architecture

Available SAD block	8x4 & 4x8 to 64×64
Slice LUTs	32752
Slice registers	5077
Maximum frequency	115.15Mhz
DSP	0
F7 MUX	1577
F8 MUX	481
Memory	108Kb RAM

4.2 Comparison with other works

In Table 2, we report published results [12, 13, 14], [24] and [25] to compare our performances. We selected related works on FPGA mentioning the same characteristics, such as slice LUTs, slice registers, maximum frequency, frame rate per second, and the PUs size block used for the SAD computing in each work.

Table 2. The synthesis results of the proposed architecture

	Our work	[12]	[13]	[14]	[24]	[25]
Technology	28 nm Artix-7	65 nm Virtex-5	40 nm Virtex-6	28 nm Virtex-7	90 nm Virtex-4	120 nm Virtex 2
LUTs	33K	15K	55k	39K	12K	657
Registers	5K	20K	19K	24K	7K	-
Max frequency	115.15Mhz	171.94Mhz	110Mhz	458.7Mhz	32Mhz	133Mhz
Serach window	±64	-	±24	±20	±16	-
Frames rate	4K@30fps	-	1080p@30fps	2K@30fps	HD@5.34	1024 x 1024@127fps
PUs	ALL	ALL	4×8 to 32×32	ALL	32×32	4×4

In [12], the whole hardware architecture for SAD algorithm is presented, in order to compare our performance with its work, we limited the comparison only on the consume area obtained in the Virtex-5 65 nm because the authors not indicate how its design the numbers of frame can be processed. so, its maximum frequency is increased about 1.5 times than our frequency. For LUTs, it tikes 18K less than our work, however, the results for registers shows that have 15K more than our results. Yuan et al. [13] present a parallel SAD architecture synthesized on Virtex-6 40nm. They used more LUTs resources and registers compared with our case, but they reported a low maximum frequency, only 110 MHz. The frames computing per second could be up to 30 fps just full HD (1080p). Medhat el al. [14] on an FPGA Virtex-7 with 28 nm, the same technology used in this work, has reported another parallel design. Even if their results seem comparable to ours, our design tikes 118% of LUTs less than its work, however, our result of the registers consuming shows that have 380% more than our results with a lower maximum frequency (398% more than our frequency). They exploit five times more slice registers, and they can process 30fps for the 2K resolution, using only a small search window, 104×104 pixels.

Compared to our architecture, Muralidhar and Ramaro [24] report on a lower consumed area (LUT and register) on the Virtex-4 FPGA, but they stayed so far from the real time processing, with a rate just 5.32 fps for the HD resolution; their circuit is still only well adopted for the H.264/AVC video coding standard.

Compared to existing works, our designpresents a good performance that explores the parallelism in the hardware architecture, which is the way to reach highthroughput for 4K encoding in real-time. Therefore, our circuit presents a good coding and efficiency regarding the resource consumption. Joshi et al. [25] presented an architecture for SAD in HEVC standard. it uses pipelined design with parallel processing with 16 processing unit in order to reduce computation time of the execution and to increase the performance. The hardware design is prototyped and simulated on Xilinx Virtex-5 FPGA for XC5VLX20T family. and also is synthesized on 65 nm technology and reaches a maximum clock frequency of 475.21 MHz. They did not give a the number of coded frames, and the highest frequency obtained led to a more consumption of energy. They only reached just 127 fps for 1024×1024 image resolution, but their design is still limited to the middle resolution.

5. CONCLUSION

In this paper, we designed a novel high parallel accelerator for the SAD algorithm hardware implementation. This concerns the new HEVC standard, and it is implemented on an FPGA Artix-7 Xilinx Zynq-7000. The design is parallel and allows the acceleration of integer motion estimation with no modification in the motion estimation algorithm. The proposed architecture has a maximum operating frequency up to 115.15 MHz and can process the well-known 4K video sequences with high resolution 3840×2160 within the real-time requirements. Compared with previous works, this design presents a high speed and low area cost architecture suitable for video encoders over a wide range of video applications. The design is simple and perfectly adapted for the HEVC encoder. In the next work, our proposed architecture will be used as an engine for the next proposed circuit for an inter prediction block of a HEVC encoder, keeping the real-time processing for higher resolutions, 8K and 4K.

ACKNOWLEDGEMENTS

This work is partially supported by the Moroccan-Tunisian program n017/TM 24 titled Study and design of an embedded system for Ultra High Definition compression on a dedicated multi-component architecture, as Application: Digital TV. The authors would like to thank the Moroccan and Tunisian governments for their financial support.

REFERENCES

[1] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standardsincluding high efficiency video coding (HEVC)," *Circuits Syst. Video Technol. IEEE Trans. On*, vol. 22, no. 12, pp. 16691684, 2012.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *Circuits Syst. Video Technol. IEEE Trans. On*, vol. 22, no. 12, pp. 16491668, 2012.

[3] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *Circuits Syst. Video Technol. IEEE Trans. On*, vol. 22, no. 12, pp. 16851696, 2012.

[4] El Ansari, Abdessamad, Ali Ahaitouf, and Anass Mansouri. "An implementation comparison of the HEVC encoder on two embedded processors," *Wireless Technologies, Embedded and Intelligent Systems (WITS), 2017 International Conference on IEEE*, 2017.

[5] svn HEVCSoftware - Revision 4598: /tags/HM-15.0. [Online]. Available: , <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-15.0/>

[6] J.-H. Hsieh, J.-H. Huang, and H.-R. Wang, "DVFS-aware motion estimation design scheme based on bandwidthratedistortion optimization in application processor systems," *Integr. VLSI J.*, vol. 57, pp. 7480, 2017.

[7] Moussa, Mourad, Nesrine Bdioui, and Ali Douik, "Motion Detection and Clustering Using PCA and NN in Color Image Sequence," *Telecommunication Computing Electronics and Control (TELKOMNIKA)*, 16.2, 747-754, 2018.

[8] F. L. Walter, C. M. Diniz, and S. Bampi, "Synthesis and comparison of low-power high-throughput architectures for SAD calculation, Analog Integr," *Circuits Signal Process.*, vol. 73, no. 3, pp. 873884, 2012.

[9] S. Rehman, R. Young, C. Chatwin, and P. Birch, "An FPGA based generic framework for high speed sum of absolute difference implementation," *Eur. J. Sci. Res.*, vol. 33, no. 1, p. 629, 2009.

[10] J. Kalomiros and J. Lygouras, "Comparative study of local SAD and dynamic programming for stereo processing using dedicated hardware," *EURASIP J. Adv. Signal Process.*, vol. 2009, no. 1, pp. 118, 2010.

[11] Z. Liu, S. Goto, and T. Ikenaga, "Optimization of Propagate Partial SAD and SAD tree motion estimation hardwired engine for H. 264," in *Computer Design, 2008. ICCD 2008. IEEE International Conference*, pp.328333, 2008.

[12] P. Nalluri, L. N. Alves, and A. Navarro, "A novel SAD architecture for variable block size motion estimation in HEVC video coding, in System on Chip (SoC)," *2013 International Symposium*, pp. 14, 2013.

[13] X. Yuan, L. Jinsong, G. Liwei, Z. Zhi, and R. K. Teng, "*A high performance VLSI architecture for integer motion estimation in HEVC,"* in *ASIC (ASICON), 2013 IEEE 10th International Conference*, pp. 14, 2013.

[14] A. Medhat, A. Shalaby, M. S. Sayed, M. Elsabrouty, and F. Mehdipour, "*A highly parallel SAD architecture for motion estimation in HEVC encoder,"* in *Circuits and Systems (APCCAS) IEEE Asia Pacific Conference on 2014*, pp. 280283, 2014.
[15] Zabidi, N. M., & Ab Rahman, A. A. H, "VLSI Design of a Fast Pipelined 8x8 Discrete Cosine Transform," *International Journal of Electrical and Computer Engineering (IJECE)*, 7(3), 1430-1435, 2017.
[16] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, "Block partitioning structure in the HEVC standard," *Circuits Syst. Video Technol. IEEE Trans. On*, vol. 22, no. 12, pp. 16971706, 2012.
[17] L. Yufei, F. Xiubo, and W. Qin, "A high-performance low cost SAD architecture for video coding," *Consum. Electron. IEEE Trans. On*, vol. 53, no. 2, pp. 535541, 2007.
[18] J. Vanne, E. Aho, T. D. Hamalainen, and K. Kuusilinna, "A high-performance sum of absolute difference implementation for motion estimation," *Circuits Syst. Video Technol. IEEE Trans. On*, vol. 16, no. 7, pp. 876883, 2006.
[19] A. Kulkarni and T. Mohsenin, "Low Overhead Architectures for OMP Compressive Sensing Reconstruction Algorithm," *in IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 6, pp. 1468-1480,. doi: 10.1109/TCSI.2017.2648854, June 2017.
[20] Jeon, Myunghoon, and Byoung-Dai Lee. "Toward Content-Aware Video Partitioning Methods for Distributed HEVC Video Encoding," *International Journal of Electrical and Computer Engineering (IJECE)*, 5.3, 569-578, 2015.
[21] Bossen, Frank. "*Common test conditions and software reference configurations*," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 5th meeting, Jan. 2011. 2011.
[22] El ansari, Abdessamad, Anass Mansouri, and Ali Ahaitouf, "*An Efficient VLSI Architecture Design for Integer DCT in HEVC Standard," Proc. of The 13th ACS/IEEE International conference on computer systems and Applications (AICCSA 2016)*, Agadir, Moroco, November 29th to December 2nd, 2016.
[23] Rani, Archana, and Naresh Grover, "An Enhanced FPGA Based Asynchronous Microprocessor Design Using VIVADO and ISIM," *Bulletin of Electrical Engineering and Informatics*, 7.2, 199-208, 2018.
[24] P. Muralidhar and C. Ramarao, "Efficient architecture for global elimination algorithm for H.264 motion estimation, Sadhana, pp. 18.
[25] Joshi, Amit M., Mohd Samar Ansari, and Chitrakant Sahu. "*VLSI Architecture of High Speed SAD for High Efficiency Video Coding (HEVC) Encoder," Circuits and Systems (ISCAS), 2018 IEEE International Symposium on. IEEE*, 2018.

BIOGRAPHIES OF AUTHORS

El Ansari Abdessamad received her PhD in embedded systems from the faculty of science and technology (FST) of Fez Morocco, in 2019. His Master degree from the FST of Fez, in 2012, he joined the team micro- electronics and embedded systems of the laboratory of energies and smart system. His research interests include video coding and implementation of video coding standard on embedded processors (ARM and DSP) and VLSI architecture (FPGA).

Nejmeddine Bahri he is currently an associate professor in the Faculty of Sciences of Monastir-Tunisia. In 2015, he obtained his PhD in Computer Science from Paris-EST University and a PhD in Electrical Engineering from the National School of Engineers of Sfax (ENIS). In 2010, he received the Master degree in electronics and in 2009 he obtained the engineering degree in electrical engineering from ENIS. He is a member of Sfax Laboratory of Electronics and Information Technology. His current research activities include video and image processing on embedded processors, algorithms implementation on parallel architectures.

Anass Mansouri received M.S. and Ph.D degrees in Microelectronics and Telecommunication from Faculty of sciences & technology, Fes, Morocco, in 2005 and 2009, respectively. He is an Assistant Professor in National School of Applied Sciences, Fes. His major research interests include VLSI and embedded architectures design, video and image Processing.

Nouri Masmoudi received electrical engineering degree from the Faculty of Sciences and Techniques-Sfax, Tunisia, in 1982, the DEA degree from the National Institute of Applied Sciences-Lyon and University Claude Bernard—Lyon, France in 1984. From 1986 to 1990, he prepared his thesis at the laboratory of Power Electronics (LEP) at the National School of Engineers of sfax (ENIS). He received his Ph.D. degree from the National School Engineering of Tunis (ENIT), Tunisia in 1990. From 1990 to 2000, he is currently a professor at the electrical engineering department—ENIS. Since 2000, he has been a group leader 'Circuits and Systems' in the Laboratory of Electronics and Information Technology. Since 2003, he is responsible for the Electronic Master Program at ENIS. His research activities have been devoted to several topics: Design, Telecommunication, Embedded Systems, Information Technology, Video Coding and Image Processing.

Ali Ahaitouf, teacher and researcher in the University of Sidi Mohammed Ben Abdellah University of Fes. He obtained his PhD in electronics since 1998. He is director of the laboratory of energies and smart system and the head of the research team microelectronics and embedded systems. His research field of research covers microelectronics and solar compounds, digital and analog design of the integrated circuits, Image and data compression. He managed many research multilateral projects, related to analog design optimlization, electronics component characterization and optimization as well as solar energy under concentration (CPV).